

Randomised Algorithms

Lecture 1: Introduction to Course & Introduction to Chernoff Bounds

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Outline

Introduction

Topics and Syllabus

A (Very) Brief Reminder of Probability Theory

Basic Examples

Introduction to Chernoff Bounds

Randomised Algorithms

What? Randomised Algorithms utilise random bits to compute their output.

Randomised Algorithms

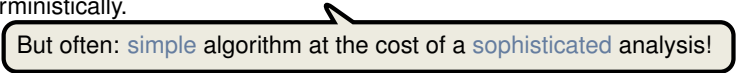
What? Randomised Algorithms utilise random bits to compute their output.

Why? Randomised Algorithms often provide an efficient (and elegant!) solution or approximation to a problem that is costly (or impossible) to solve deterministically.

Randomised Algorithms

What? Randomised Algorithms utilise random bits to compute their output.

Why? Randomised Algorithms often provide an efficient (and elegant!) solution or approximation to a problem that is costly (or impossible) to solve deterministically.



But often: simple algorithm at the cost of a sophisticated analysis!

Randomised Algorithms

What? Randomised Algorithms utilise random bits to compute their output.

Why? Randomised Algorithms often provide an efficient (and elegant!) solution or approximation to a problem that is costly (or impossible) to solve deterministically.

“... If somebody would ask me, what in the last 10 years, what was the most important change in the study of algorithms I would have to say that people getting really familiar with randomised algorithms had to be the winner.”

- Donald E. Knuth (in Randomization and Religion)



Randomised Algorithms

What? Randomised Algorithms utilise random bits to compute their output.

Why? Randomised Algorithms often provide an efficient (and elegant!) solution or approximation to a problem that is costly (or impossible) to solve deterministically.

“... If somebody would ask me, what in the last 10 years, what was the most important change in the study of algorithms I would have to say that people getting really familiar with randomised algorithms had to be the winner.”

- Donald E. Knuth (in *Randomization and Religion*)



How? This course aims to strengthen your knowledge of probability theory and apply this to analyse examples of randomised algorithms.

Randomised Algorithms

What? Randomised Algorithms utilise random bits to compute their output.

Why? Randomised Algorithms often provide an efficient (and elegant!) solution or approximation to a problem that is costly (or impossible) to solve deterministically.

“... If somebody would ask me, what in the last 10 years, what was the most important change in the study of algorithms I would have to say that people getting really familiar with randomised algorithms had to be the winner.”

- Donald E. Knuth (in *Randomization and Religion*)



How? This course aims to strengthen your knowledge of probability theory and apply this to analyse examples of randomised algorithms.

What if I (initially) don't care about randomised algorithms?

Many of the techniques in this course (Markov Chains, Concentration of Measure, Spectral Theory) are very relevant to other popular areas of research and employment such as Data Science and Machine Learning.

Some stuff you should know...

In this course we will assume some basic knowledge of **probability**:

- random variable
- computing expectations and variances
- notions of independence and conditional probabilities
- “general” idea of how to compute probabilities (manipulating, counting and **estimating**)



Some stuff you should know...

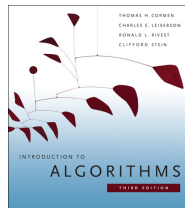
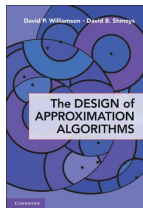
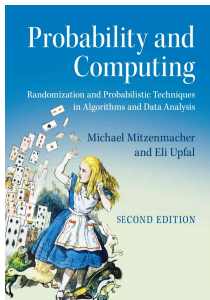
In this course we will assume some basic knowledge of **probability**:

- random variable
- computing expectations and variances
- notions of independence and conditional probabilities
- “general” idea of how to compute probabilities (manipulating, counting and **estimating**)



You should also be familiar with basic **computer science**, **mathematics** knowledge such as:

- graphs
- basic algorithms (sorting, graph algorithms etc.)
- matrices, norms and vectors



- (★) **Michael Mitzenmacher and Eli Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, 2nd edition, 2017**
- David P. Williamson and David B. Shmoys. The Design of Approximation Algorithms, Cambridge University Press, 2011
- Cormen, T.H., Leiserson, C.D., Rivest, R.L. and Stein, C. Introduction to Algorithms. MIT Press (3rd ed.), 2009
(We will adopt some of the labels (e.g., Theorem 35.6) from this book in Lectures 6-10)

Outline

Introduction

Topics and Syllabus

A (Very) Brief Reminder of Probability Theory

Basic Examples

Introduction to Chernoff Bounds

1 Introduction (Lecture)

- Intro to Randomised Algorithms; Logistics; Recap of Probability; Examples.

1 Introduction (Lecture)

- Intro to Randomised Algorithms; Logistics; Recap of Probability; Examples.

Lectures 2-5 focus on probabilistic tools and techniques.

2–3 Concentration (Lectures)

- Concept of Concentration; Recap of Markov and Chebyshev; Chernoff Bounds and Applications; Extensions: Hoeffding's Inequality and Method of Bounded Differences; Applications.

4 Markov Chains and Mixing Times (Lecture)

- Recap; Stopping and Hitting Times; Properties of Markov Chains; Convergence to Stationary Distribution; Variation Distance and Mixing Time

5 Hitting Times and Application to 2-SAT (Lecture)

- Reversible Markov Chains and Random Walks on Graphs; Cover Times and Hitting Times on Graphs (Example: Paths and Grids); A Randomised Algorithm for 2-SAT Algorithm

1 Introduction (Lecture)

- Intro to Randomised Algorithms; Logistics; Recap of Probability; Examples.

Lectures 2-5 focus on probabilistic tools and techniques.

2–3 Concentration (Lectures)

- Concept of Concentration; Recap of Markov and Chebyshev; Chernoff Bounds and Applications; Extensions: Hoeffding's Inequality and Method of Bounded Differences; Applications.

4 Markov Chains and Mixing Times (Lecture)

- Recap; Stopping and Hitting Times; Properties of Markov Chains; Convergence to Stationary Distribution; Variation Distance and Mixing Time

5 Hitting Times and Application to 2-SAT (Lecture)

- Reversible Markov Chains and Random Walks on Graphs; Cover Times and Hitting Times on Graphs (Example: Paths and Grids); A Randomised Algorithm for 2-SAT Algorithm

Lectures 6-8 introduce linear programming, a (mostly) deterministic but very powerful technique to solve various optimisation problems.

6–7 Linear Programming (Lectures)

- Introduction to Linear Programming, Applications, Standard and Slack Forms, Simplex Algorithm, Finding an Initial Solution, Fundamental Theorem of Linear Programming

8 Travelling Salesman Problem (Interactive Demo)

- Hardness of the general TSP problem, Formulating TSP as an integer program; Classical TSP instance from 1954; Branch & Bound Technique to solve integer programs using linear programs

We then see how we can efficiently combine linear programming with randomised techniques, in particular, rounding:

9–10 Randomised Approximation Algorithms (Lectures)

- MAX-3-CNF and Guessing, Vertex-Cover and Deterministic Rounding of Linear Program, Set-Cover and Randomised Rounding, Concluding Example: MAX-CNF and Hybrid Algorithm

We then see how we can efficiently combine linear programming with randomised techniques, in particular, rounding:

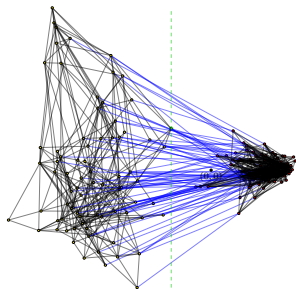
9–10 Randomised Approximation Algorithms (Lectures)

- MAX-3-CNF and Guessing, Vertex-Cover and Deterministic Rounding of Linear Program, Set-Cover and Randomised Rounding, Concluding Example: MAX-CNF and Hybrid Algorithm

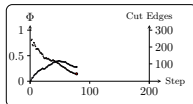
Lectures 11-12 cover a more advanced topic with ML flavour:

11–12 Spectral Graph Theory and Spectral Clustering (Lectures)

- Eigenvalues, Eigenvectors and Spectrum; Visualising Graphs; Expansion; Cheeger's Inequality; Clustering and Examples; Analysing Mixing Times



- Step: 78
- Threshold: -0.0336
- Partition Sizes: 78/122
- Cut Edges: 84
- Conductance: 0.1448



Outline

Introduction

Topics and Syllabus

A (Very) Brief Reminder of Probability Theory

Basic Examples

Introduction to Chernoff Bounds

Recap: Probability Space

In probability theory we wish to evaluate the likelihood of certain results from an experiment. The setting of this is the **probability space** $(\Omega, \Sigma, \mathbf{P})$.

Recap: Probability Space

In probability theory we wish to evaluate the likelihood of certain results from an experiment. The setting of this is the **probability space** $(\Omega, \Sigma, \mathbf{P})$.

Components of the Probability Space $(\Omega, \Sigma, \mathbf{P})$

- The **Sample Space** Ω contains all the possible, mutually exclusive **outcomes** $\omega_1, \omega_2, \dots$ of the experiment.
- The **Event Space** Σ is the power-set of Ω containing **events**, which are combinations of outcomes (subsets of Ω including \emptyset and Ω).
- The **Probability Measure** \mathbf{P} is a function from Σ to \mathbb{R} satisfying
 - (i) $0 \leq \mathbf{P}[\mathcal{E}] \leq 1$, for all $\mathcal{E} \in \Sigma$
 - (ii) $\mathbf{P}[\Omega] = 1$
 - (iii) If $\mathcal{E}_1, \mathcal{E}_2, \dots \in \Sigma$ are pairwise disjoint ($\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ for all $i \neq j$) then

$$\mathbf{P}\left[\bigcup_{i=1}^{\infty} \mathcal{E}_i\right] = \sum_{i=1}^{\infty} \mathbf{P}[\mathcal{E}_i].$$

Recap: Random Variables

A **random variable** X on $(\Omega, \Sigma, \mathbf{P})$ is a function $X : \Omega \rightarrow \mathbb{R}$ mapping each sample “outcome” to a real number.

Intuitively, random variables are the “**observables**” in our experiment.

Recap: Random Variables

A **random variable** X on $(\Omega, \Sigma, \mathbf{P})$ is a function $X : \Omega \rightarrow \mathbb{R}$ mapping each sample “outcome” to a real number.

Intuitively, random variables are the “**observables**” in our experiment.

Examples of random variables

- The **number of heads** in three coin flips $X_1, X_2, X_3 \in \{0, 1\}$ is:

$$X_1 + X_2 + X_3$$

Recap: Random Variables

A **random variable** X on $(\Omega, \Sigma, \mathbf{P})$ is a function $X : \Omega \rightarrow \mathbb{R}$ mapping each sample “outcome” to a real number.

Intuitively, random variables are the “**observables**” in our experiment.

Examples of random variables

- The **number of heads** in three coin flips $X_1, X_2, X_3 \in \{0, 1\}$ is:

$$X_1 + X_2 + X_3$$

- The **indicator random variable** $\mathbf{1}_{\mathcal{E}}$ of an event $\mathcal{E} \in \Sigma$ given by

$$\mathbf{1}_{\mathcal{E}}(\omega) = \begin{cases} 1 & \text{if } \omega \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

Recap: Random Variables

A **random variable** X on $(\Omega, \Sigma, \mathbf{P})$ is a function $X : \Omega \rightarrow \mathbb{R}$ mapping each sample “outcome” to a real number.

Intuitively, random variables are the “**observables**” in our experiment.

Examples of random variables

- The **number of heads** in three coin flips $X_1, X_2, X_3 \in \{0, 1\}$ is:

$$X_1 + X_2 + X_3$$

- The **indicator random variable** $\mathbf{1}_{\mathcal{E}}$ of an event $\mathcal{E} \in \Sigma$ given by

$$\mathbf{1}_{\mathcal{E}}(\omega) = \begin{cases} 1 & \text{if } \omega \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

For the indicator random variable $\mathbf{1}_{\mathcal{E}}$ we have $\mathbf{E}[\mathbf{1}_{\mathcal{E}}] = \mathbf{P}[\mathcal{E}]$.

Recap: Random Variables

A **random variable** X on $(\Omega, \Sigma, \mathbf{P})$ is a function $X : \Omega \rightarrow \mathbb{R}$ mapping each sample “outcome” to a real number.

Intuitively, random variables are the “**observables**” in our experiment.

Examples of random variables

- The **number of heads** in three coin flips $X_1, X_2, X_3 \in \{0, 1\}$ is:

$$X_1 + X_2 + X_3$$

- The **indicator random variable** $\mathbf{1}_{\mathcal{E}}$ of an event $\mathcal{E} \in \Sigma$ given by

$$\mathbf{1}_{\mathcal{E}}(\omega) = \begin{cases} 1 & \text{if } \omega \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

For the indicator random variable $\mathbf{1}_{\mathcal{E}}$ we have $\mathbf{E}[\mathbf{1}_{\mathcal{E}}] = \mathbf{P}[\mathcal{E}]$.

- The **number of sixes** of two dice throws $X_1, X_2 \in \{1, 2, \dots, 6\}$ is

$$\mathbf{1}_{X_1=6} + \mathbf{1}_{X_2=6}$$

Recap: Boole's Inequality (Union Bound)

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

Recap: Boole's Inequality (Union Bound)

Union Bound is one of the most basic probability inequalities, yet it is extremely useful and easy to apply!

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

Recap: Boole's Inequality (Union Bound)

Union Bound is one of the most basic probability inequalities, yet it is extremely useful and easy to apply!

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

A Proof using Indicator Random Variables:

Recap: Boole's Inequality (Union Bound)

Union Bound is one of the most basic probability inequalities, yet it is extremely useful and easy to apply!

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

A Proof using Indicator Random Variables:

1. Let $\mathbf{1}_{\mathcal{E}_i}$ be the random variable that takes value 1 if \mathcal{E}_i holds, 0 otherwise

Recap: Boole's Inequality (Union Bound)

Union Bound is one of the most basic probability inequalities, yet it is extremely useful and easy to apply!

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

A Proof using Indicator Random Variables:

1. Let $\mathbf{1}_{\mathcal{E}_i}$ be the random variable that takes value 1 if \mathcal{E}_i holds, 0 otherwise
2. $\mathbf{E} [\mathbf{1}_{\mathcal{E}_i}] = \mathbf{P} [\mathcal{E}_i]$ (Check this)

Recap: Boole's Inequality (Union Bound)

Union Bound is one of the most basic probability inequalities, yet it is extremely useful and easy to apply!

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

A Proof using Indicator Random Variables:

1. Let $\mathbf{1}_{\mathcal{E}_i}$ be the random variable that takes value 1 if \mathcal{E}_i holds, 0 otherwise
2. $\mathbf{E} [\mathbf{1}_{\mathcal{E}_i}] = \mathbf{P} [\mathcal{E}_i]$ (Check this)
3. It is clear that $\mathbf{1}_{\bigcup_{i=1}^n \mathcal{E}_i} \leq \sum_{i=1}^n \mathbf{1}_{\mathcal{E}_i}$ (Check this)

Recap: Boole's Inequality (Union Bound)

Union Bound is one of the most basic probability inequalities, yet it is extremely useful and easy to apply!

Union Bound

Let $\mathcal{E}_1, \dots, \mathcal{E}_n$ be a collection of events in Σ . Then

$$\mathbf{P} \left[\bigcup_{i=1}^n \mathcal{E}_i \right] \leq \sum_{i=1}^n \mathbf{P} [\mathcal{E}_i].$$

A Proof using Indicator Random Variables:

1. Let $\mathbf{1}_{\mathcal{E}_i}$ be the random variable that takes value 1 if \mathcal{E}_i holds, 0 otherwise
2. $\mathbf{E} [\mathbf{1}_{\mathcal{E}_i}] = \mathbf{P} [\mathcal{E}_i]$ (Check this)
3. It is clear that $\mathbf{1}_{\bigcup_{i=1}^n \mathcal{E}_i} \leq \sum_{i=1}^n \mathbf{1}_{\mathcal{E}_i}$ (Check this)
4. Taking expectation completes the proof.

Outline

Introduction

Topics and Syllabus

A (Very) Brief Reminder of Probability Theory

Basic Examples

Introduction to Chernoff Bounds

A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem



A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

- **Given:** Undirected graph $G = (V, E)$

A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

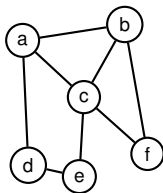
- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.

A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.

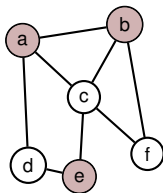


A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.



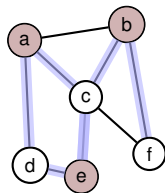
$$S = \{a, b, e\}$$
$$e(S, S^c) = 6$$

A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.



$$S = \{a, b, e\}$$
$$e(S, S^c) = 6$$

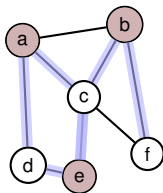
A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.

Applications:



$$S = \{a, b, e\}$$
$$e(S, S^c) = 6$$

A Randomised Algorithm for MAX-CUT (1/2)

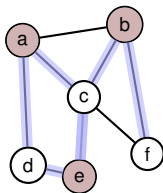
$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.

Applications:

- network or chip design
- machine learning
- statistical physics



$$S = \{a, b, e\}$$
$$e(S, S^c) = 6$$

A Randomised Algorithm for MAX-CUT (1/2)

$E(A, B)$: set of edges with one endpoint in $A \subseteq V$ and the other in $B \subseteq V$.

MAX-CUT Problem

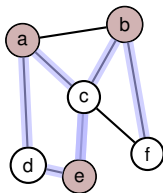
- **Given:** Undirected graph $G = (V, E)$
- **Goal:** Find $S \subseteq V$ such that $e(S, S^c) := |E(S, S^c)|$ is maximised.

Applications:

- network or chip design
- machine learning
- statistical physics

Comments:

- MAX-CUT is NP-hard
- It is different from the **clustering** problem, where we want to find a **sparse cut**
- Note that the **MIN-CUT** problem is solvable in polynomial time!



$$S = \{a, b, e\}$$
$$e(S, S^c) = 6$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

This kind of “random guessing” will appear often in this course!

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

— Proposition —

RANDOMMAXCUT(G) gives a 2-approximation using time $O(n)$.

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMCUT(G) gives a 2-approximation using time $O(n)$.

A Randomised Algorithm for MAX-CUT (2/2)

RANDMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDMAXCUT(G) gives a 2-approximation using time $O(n)$.



Question:

1. What is the sample space Ω here?
2. Which quantity do we need to analyse?

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

A Randomised Algorithm for MAX-CUT (2/2)

RANDMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\mathbf{E} [e(S, S^c)]$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\mathbf{E} [e(S, S^c)] = \mathbf{E} \left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}} \right]$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\begin{aligned}\mathbf{E} [e(S, S^c)] &= \mathbf{E} \left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}} \right] \\ &= \sum_{\{u,v\} \in E} \mathbf{E} [\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}]\end{aligned}$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\begin{aligned}\mathbf{E}[e(S, S^c)] &= \mathbf{E}\left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}\right] \\&= \sum_{\{u,v\} \in E} \mathbf{E}[\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}] \\&= \sum_{\{u,v\} \in E} \mathbf{P}[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}]\end{aligned}$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDMAXCUT(G) gives a **2-approximation** using time $O(n)$.

Proof:

- We need to analyse the **expectation** of $e(S, S^c)$:

$$\begin{aligned}\mathbf{E}[e(S, S^c)] &= \mathbf{E}\left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}\right] \\&= \sum_{\{u,v\} \in E} \mathbf{E}[\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}] \\&= \sum_{\{u,v\} \in E} \mathbf{P}[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}] \\&= 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S, v \in S^c]\end{aligned}$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\begin{aligned}\mathbf{E}[e(S, S^c)] &= \mathbf{E}\left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}\right] \\&= \sum_{\{u,v\} \in E} \mathbf{E}[\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}] \\&= \sum_{\{u,v\} \in E} \mathbf{P}[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}] \\&= 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S, v \in S^c] = 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S] \cdot \mathbf{P}[v \in S^c]\end{aligned}$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\begin{aligned} \mathbf{E}[e(S, S^c)] &= \mathbf{E}\left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}\right] \\ &= \sum_{\{u,v\} \in E} \mathbf{E}[\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}] \\ &= \sum_{\{u,v\} \in E} \mathbf{P}[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}] \\ &= 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S, v \in S^c] = 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S] \cdot \mathbf{P}[v \in S^c] = |E|/2. \end{aligned}$$

A Randomised Algorithm for MAX-CUT (2/2)

RANDMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDMAXCUT(G) gives a 2-approximation using time $O(n)$.

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\begin{aligned} \mathbf{E}[e(S, S^c)] &= \mathbf{E}\left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}\right] \\ &= \sum_{\{u,v\} \in E} \mathbf{E}[\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}] \\ &= \sum_{\{u,v\} \in E} \mathbf{P}[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}] \\ &= 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S, v \in S^c] = 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S] \cdot \mathbf{P}[v \in S^c] = |E|/2. \end{aligned}$$

- Since for any $S \subseteq V$, we have $e(S, S^c) \leq |E|$, the proof is complete.

A Randomised Algorithm for MAX-CUT (2/2)

RANDOMMAXCUT(G)

- 1: Start with $S \leftarrow \emptyset$
- 2: **For** each $v \in V$, add v to S with probability $1/2$
- 3: **Return** S

Ratio between optimal and expected value of our solution is ≤ 2 (more on this in Lecture 9)

RANDOMMAXCUT(G) gives a 2-approximation using time $O(n)$.

Later: learn stronger tools that imply concentration around the expectation!

Proof:

- We need to analyse the expectation of $e(S, S^c)$:

$$\begin{aligned} \mathbf{E}[e(S, S^c)] &= \mathbf{E}\left[\sum_{\{u,v\} \in E} \mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}\right] \\ &= \sum_{\{u,v\} \in E} \mathbf{E}[\mathbf{1}_{\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}}] \\ &= \sum_{\{u,v\} \in E} \mathbf{P}[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}] \\ &= 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S, v \in S^c] = 2 \sum_{\{u,v\} \in E} \mathbf{P}[u \in S] \cdot \mathbf{P}[v \in S^c] = |E|/2. \end{aligned}$$

- Since for any $S \subseteq V$, we have $e(S, S^c) \leq |E|$, the proof is complete.

Example: Coupon Collector



Source: <https://www.express.co.uk/life-style/life/567954/Discount-codes-money-saving-vouchers-coupons-mum>

Coupon Collector Problem

Suppose that there are n coupons to be collected from the cereal box. Every morning you open a new cereal box and get one coupon. We assume that each coupon appears with the same probability in the box.

Example: Coupon Collector



Source: <https://www.express.co.uk/life-style/life/567954/Discount-codes-money-saving-vouchers-coupons-mum>

This is a very important example in the design and analysis of randomised algorithms.

Coupon Collector Problem

Suppose that there are n coupons to be collected from the cereal box. Every morning you open a new cereal box and get one coupon. We assume that each coupon appears with the same probability in the box.

Example: Coupon Collector



Source: <https://www.express.co.uk/life-style/life/567954/Discount-codes-money-saving-vouchers-coupons-mum>

This is a very important example in the design and analysis of randomised algorithms.

Coupon Collector Problem

Suppose that there are n coupons to be collected from the cereal box. Every morning you open a new cereal box and get one coupon. We assume that each coupon appears with the same probability in the box.

Example Sequence for $n = 8$: 7, 6, 3, 3, 3, 2, 5, 4, 2, 4, 1, 4, 2, 1, 4, 3, 1, 4, 8 ✓

Example: Coupon Collector



Source: <https://www.express.co.uk/life-style/life/567954/Discount-codes-money-saving-vouchers-coupons-mum>

This is a very important example in the design and analysis of randomised algorithms.

Coupon Collector Problem

Suppose that there are n coupons to be collected from the cereal box. Every morning you open a new cereal box and get one coupon. We assume that each coupon appears with the same probability in the box.

Example Sequence for $n = 8$: 7, 6, 3, 3, 3, 2, 5, 4, 2, 4, 1, 4, 2, 1, 4, 3, 1, 4, 8 ✓

Exercise ([Ex. 1.11])

Example: Coupon Collector



Source: <https://www.express.co.uk/life-style/life/567954/Discount-codes-money-saving-vouchers-coupons-mum>

This is a very important example in the design and analysis of randomised algorithms.

Coupon Collector Problem

Suppose that there are n coupons to be collected from the cereal box. Every morning you open a new cereal box and get one coupon. We assume that each coupon appears with the same probability in the box.

Example Sequence for $n = 8$: 7, 6, 3, 3, 3, 2, 5, 4, 2, 4, 1, 4, 2, 1, 4, 3, 1, 4, 8 ✓

Exercise ([Ex. 1.11])

In this course: $\log n = \ln n$

1. Prove it takes $n \sum_{k=1}^n \frac{1}{k} \approx n \log n$ expected boxes to collect all coupons

Example: Coupon Collector



Source: <https://www.express.co.uk/life-style/life/567954/Discount-codes-money-saving-vouchers-coupons-mum>

This is a very important example in the design and analysis of randomised algorithms.

Coupon Collector Problem

Suppose that there are n coupons to be collected from the cereal box. Every morning you open a new cereal box and get one coupon. We assume that each coupon appears with the same probability in the box.

Example Sequence for $n = 8$: 7, 6, 3, 3, 3, 2, 5, 4, 2, 4, 1, 4, 2, 1, 4, 3, 1, 4, 8 ✓

Exercise ([Ex. 1.11])

In this course: $\log n = \ln n$

1. Prove it takes $n \sum_{k=1}^n \frac{1}{k} \approx n \log n$ expected boxes to collect all coupons
2. Use **Union Bound** to prove that the probability it takes more than $n \log n + cn$ boxes to collect all n coupons is $\leq e^{-c}$.

Hint: It is useful to remember that $1 - x \leq e^{-x}$ for all x

Outline

Introduction

Topics and Syllabus

A (Very) Brief Reminder of Probability Theory

Basic Examples

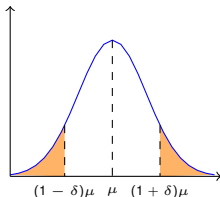
Introduction to Chernoff Bounds

Concentration Inequalities

- **Concentration** refers to the phenomena where random variables are very close to their mean
- This is very useful in randomised algorithms as it ensures an **almost** deterministic behaviour

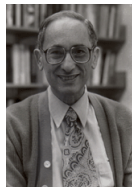
Concentration Inequalities

- **Concentration** refers to the phenomena where random variables are very close to their mean
- This is very useful in randomised algorithms as it ensures an **almost** deterministic behaviour
- It gives us the best of two worlds:
 1. **Randomised Algorithms:** Easy to Design and Implement
 2. **Deterministic Algorithms:** They do what they claim

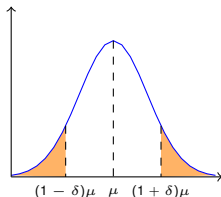


Chernoff Bounds: A Tool for Concentration (1952)

- Chernoff's bounds are “strong” bounds on the tail probabilities of **sums of independent random variables**
- random variables can be **discrete** (or continuous)
- usually these bounds decrease **exponentially** as opposed to a polynomial decrease in Markov's or Chebyshev's inequality (see example)

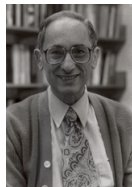


Hermann Chernoff (1923-)

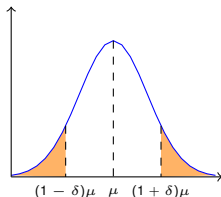


Chernoff Bounds: A Tool for Concentration (1952)

- Chernoff's bounds are “strong” bounds on the tail probabilities of **sums of independent random variables**
- random variables can be **discrete** (or continuous)
- usually these bounds decrease **exponentially** as opposed to a polynomial decrease in Markov's or Chebyshev's inequality (see example)
- easy to apply, but **requires independence**
- have found various applications in:

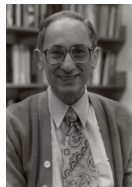


Hermann Chernoff (1923-)

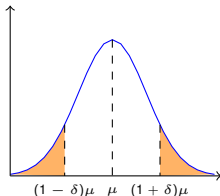


Chernoff Bounds: A Tool for Concentration (1952)

- Chernoff's bounds are “strong” bounds on the tail probabilities of **sums of independent random variables**
- random variables can be **discrete** (or continuous)
- usually these bounds decrease **exponentially** as opposed to a polynomial decrease in Markov's or Chebyshev's inequality (see example)
- easy to apply, but **requires independence**
- have found various applications in:
 - Randomised Algorithms and Statistics
 - Random Projections and Dimensionality Reduction
 - Complexity Theory and Learning Theory (e.g., PAC-learning)
 - \vdots



Hermann Chernoff (1923-)



Recap: Markov and Chebyshev

Markov's Inequality

If X is a non-negative random variable, then for any $a > 0$,

$$\mathbf{P}[X \geq a] \leq \mathbf{E}[X] / a.$$

Chebyshev's Inequality

If X is a random variable, then for any $a > 0$,

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq a] \leq \mathbf{V}[X] / a^2.$$

Recap: Markov and Chebyshev

Markov's Inequality

If X is a non-negative random variable, then for any $a > 0$,

$$\mathbf{P}[X \geq a] \leq \mathbf{E}[X] / a.$$

Chebyshev's Inequality

If X is a random variable, then for any $a > 0$,

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq a] \leq \mathbf{V}[X] / a^2.$$

- Let $f : \mathbb{R} \rightarrow [0, \infty)$ and **increasing**, then $f(X) \geq 0$, and thus

$$\mathbf{P}[X \geq a] \leq \mathbf{P}[f(X) \geq f(a)] \leq \mathbf{E}[f(X)] / f(a).$$

Recap: Markov and Chebyshev

Markov's Inequality

If X is a non-negative random variable, then for any $a > 0$,

$$\mathbf{P}[X \geq a] \leq \mathbf{E}[X] / a.$$

Chebyshev's Inequality

If X is a random variable, then for any $a > 0$,

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq a] \leq \mathbf{V}[X] / a^2.$$

- Let $f : \mathbb{R} \rightarrow [0, \infty)$ and **increasing**, then $f(X) \geq 0$, and thus

$$\mathbf{P}[X \geq a] \leq \mathbf{P}[f(X) \geq f(a)] \leq \mathbf{E}[f(X)] / f(a).$$

- Similarly, if $g : \mathbb{R} \rightarrow [0, \infty)$ and **decreasing**, then $g(X) \geq 0$, and thus

$$\mathbf{P}[X \leq a] \leq \mathbf{P}[g(X) \geq g(a)] \leq \mathbf{E}[g(X)] / g(a).$$

Recap: Markov and Chebyshev

Markov's Inequality

If X is a non-negative random variable, then for any $a > 0$,

$$\mathbf{P}[X \geq a] \leq \mathbf{E}[X] / a.$$

Chebyshev's Inequality

If X is a random variable, then for any $a > 0$,

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq a] \leq \mathbf{V}[X] / a^2.$$

- Let $f : \mathbb{R} \rightarrow [0, \infty)$ and **increasing**, then $f(X) \geq 0$, and thus

$$\mathbf{P}[X \geq a] \leq \mathbf{P}[f(X) \geq f(a)] \leq \mathbf{E}[f(X)] / f(a).$$

- Similarly, if $g : \mathbb{R} \rightarrow [0, \infty)$ and **decreasing**, then $g(X) \geq 0$, and thus

$$\mathbf{P}[X \leq a] \leq \mathbf{P}[g(X) \geq g(a)] \leq \mathbf{E}[g(X)] / g(a).$$

Chebyshev's inequality (or Markov) can be obtained by choosing $f(X) := (X - \mu)^2$ (or $f(X) := X$, respectively).

Markov and Chebyshev use the first and second moment of the random variable. Can we keep going?

Markov and Chebyshev use the first and second moment of the random variable. Can we keep going?

- Yes!

Markov and Chebyshev use the first and second moment of the random variable. Can we keep going?

- Yes!

We can consider the first, second, third and more moments! That is the basic idea behind the Chernoff Bounds

Our First Chernoff Bound

Chernoff Bounds (General Form, Upper Tail)

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu. \quad (\star)$$

Our First Chernoff Bound

Chernoff Bounds (General Form, Upper Tail)

Suppose X_1, \dots, X_n are **independent Bernoulli** random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu. \quad (\star)$$

While (\star) is one of the easiest (and most generic) Chernoff bounds to derive, the bound is complicated and hard to apply...

Our First Chernoff Bound

Chernoff Bounds (General Form, Upper Tail)

Suppose X_1, \dots, X_n are **independent Bernoulli** random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu. \quad (\star)$$

By substitution, this implies that for any $t > \mu$,

$$\mathbf{P}[X \geq t] \leq e^{-\mu} \left(\frac{e\mu}{t} \right)^t.$$

While (\star) is one of the easiest (and most generic) Chernoff bounds to derive, the bound is complicated and hard to apply...

Example: Coin Flips (1/3)

- Consider throwing a fair coin n times and count the total number of heads

Example: Coin Flips (1/3)

- Consider throwing a fair coin n times and count the total number of heads
- $X_i \in \{0, 1\}$, $X = \sum_{i=1}^n X_i$ and $\mathbf{E}[X] = n \cdot 1/2 = n/2$

Example: Coin Flips (1/3)

- Consider throwing a fair coin n times and count the total number of heads
- $X_i \in \{0, 1\}$, $X = \sum_{i=1}^n X_i$ and $\mathbf{E}[X] = n \cdot 1/2 = n/2$
- The Chernoff Bound gives for any $\delta > 0$,

$$\mathbf{P}[X \geq (1 + \delta)(n/2)] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^{n/2}.$$

Example: Coin Flips (1/3)

- Consider throwing a **fair coin** n times and count the **total number of heads**
- $X_i \in \{0, 1\}$, $X = \sum_{i=1}^n X_i$ and $\mathbf{E}[X] = n \cdot 1/2 = n/2$
- The **Chernoff Bound** gives for any $\delta > 0$,

$$\mathbf{P}[X \geq (1 + \delta)(n/2)] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^{n/2}.$$

- The above expression equals 1 only for $\delta = 0$, and then it gives a value strictly less than 1 (check this!)
- ⇒ The inequality is **exponential in n** , (for fixed δ) which is much better than Chebyshev's inequality.

Example: Coin Flips (1/3)

- Consider throwing a **fair coin** n times and count the **total number of heads**
- $X_i \in \{0, 1\}$, $X = \sum_{i=1}^n X_i$ and $\mathbf{E}[X] = n \cdot 1/2 = n/2$
- The **Chernoff Bound** gives for any $\delta > 0$,

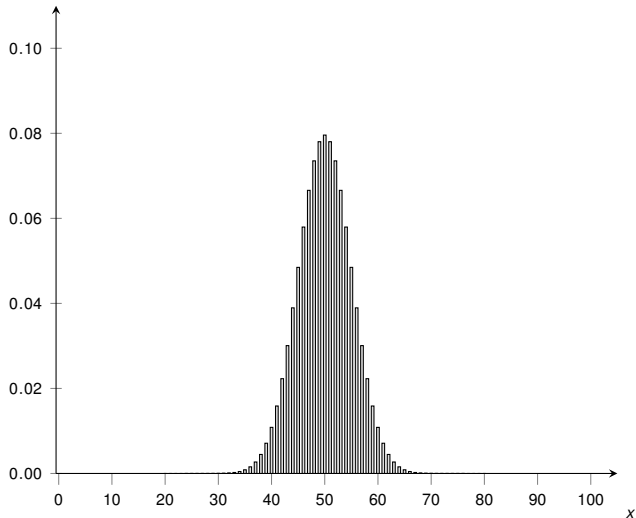
$$\mathbf{P}[X \geq (1 + \delta)(n/2)] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^{n/2}.$$

- The above expression equals 1 only for $\delta = 0$, and then it gives a value strictly less than 1 (check this!)
- ⇒ The inequality is **exponential in n** , (for fixed δ) which is much better than Chebyshev's inequality.

What about a **concrete value** of n , say $n = 100$?

Example: Coin Flips (2/3)

$$P[\text{Bin}(100, 1/2) = x]$$



Example: Coin Flips (3/3)

Consider $n = 100$ independent coin flips. We wish to find an upper bound on the probability that the number of heads is greater or equal than 75.

Example: Coin Flips (3/3)

Consider $n = 100$ independent coin flips. We wish to find an upper bound on the probability that the number of heads is greater or equal than 75.

- Markov's inequality: $\mathbf{E}[X] = 100/2 = 50$.

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq 2/3 = 0.666.$$

Example: Coin Flips (3/3)

Consider $n = 100$ independent coin flips. We wish to find an upper bound on the probability that the number of heads is greater or equal than 75.

- Markov's inequality: $\mathbf{E}[X] = 100/2 = 50$.

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq 2/3 = \mathbf{0.666}.$$

- Chebyshev's inequality: $\mathbf{V}[X] = \sum_{i=1}^{100} \mathbf{V}[X_i] = 100 \cdot (1/2)^2 = 25$.

$$\mathbf{P}[|X - \mu| \geq t] \leq \frac{\mathbf{V}[X]}{t^2},$$

and plugging in $t = 25$ gives an upper bound of $25/25^2 = 1/25 = \mathbf{0.04}$, much better than what we obtained by Markov's inequality.

Example: Coin Flips (3/3)

Consider $n = 100$ independent coin flips. We wish to find an upper bound on the probability that the number of heads is greater or equal than 75.

- Markov's inequality: $\mathbf{E}[X] = 100/2 = 50$.

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq 2/3 = \mathbf{0.666}.$$

- Chebyshev's inequality: $\mathbf{V}[X] = \sum_{i=1}^{100} \mathbf{V}[X_i] = 100 \cdot (1/2)^2 = 25$.

$$\mathbf{P}[|X - \mu| \geq t] \leq \frac{\mathbf{V}[X]}{t^2},$$

and plugging in $t = 25$ gives an upper bound of $25/25^2 = 1/25 = \mathbf{0.04}$, much better than what we obtained by Markov's inequality.

- Chernoff bound: setting $\delta = 1/2$ gives

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq \left(\frac{e^{1/2}}{(3/2)^{3/2}} \right)^{50} = \mathbf{0.004472}.$$

Example: Coin Flips (3/3)

Consider $n = 100$ independent coin flips. We wish to find an upper bound on the probability that the number of heads is greater or equal than 75.

- Markov's inequality: $\mathbf{E}[X] = 100/2 = 50$.

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq 2/3 = \mathbf{0.666}.$$

- Chebyshev's inequality: $\mathbf{V}[X] = \sum_{i=1}^{100} \mathbf{V}[X_i] = 100 \cdot (1/2)^2 = 25$.

$$\mathbf{P}[|X - \mu| \geq t] \leq \frac{\mathbf{V}[X]}{t^2},$$

and plugging in $t = 25$ gives an upper bound of $25/25^2 = 1/25 = \mathbf{0.04}$, much better than what we obtained by Markov's inequality.

- Chernoff bound: setting $\delta = 1/2$ gives

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq \left(\frac{e^{1/2}}{(3/2)^{3/2}} \right)^{50} = \mathbf{0.004472}.$$

- Remark: The exact probability is $\mathbf{0.00000028 \dots}$

Example: Coin Flips (3/3)

Consider $n = 100$ independent coin flips. We wish to find an upper bound on the probability that the number of heads is greater or equal than 75.

- Markov's inequality: $\mathbf{E}[X] = 100/2 = 50$.

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq 2/3 = \mathbf{0.666}.$$

- Chebyshev's inequality: $\mathbf{V}[X] = \sum_{i=1}^{100} \mathbf{V}[X_i] = 100 \cdot (1/2)^2 = 25$.

$$\mathbf{P}[|X - \mu| \geq t] \leq \frac{\mathbf{V}[X]}{t^2},$$

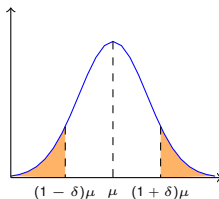
and plugging in $t = 25$ gives an upper bound of $25/25^2 = 1/25 = \mathbf{0.04}$, much better than what we obtained by Markov's inequality.

- Chernoff bound: setting $\delta = 1/2$ gives

$$\mathbf{P}[X \geq 3/2 \cdot \mathbf{E}[X]] \leq \left(\frac{e^{1/2}}{(3/2)^{3/2}} \right)^{50} = \mathbf{0.004472}.$$

- Remark: The exact probability is $\mathbf{0.00000028 \dots}$

Chernoff bound yields a much better result (but needs independence!)



Randomised Algorithms

Lecture 2: Concentration Inequalities, Application to Balls-into-Bins

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

How to Derive Chernoff Bounds

Application 1: Balls into Bins

Appendix: More on Moment Generating Functions (non-examinable)

General Recipe for Deriving Chernoff Bounds

Recipe

The **three main steps** in deriving Chernoff bounds for sums of **independent** random variables $X = X_1 + \dots + X_n$ are:

General Recipe for Deriving Chernoff Bounds

Recipe

The **three main steps** in deriving Chernoff bounds for sums of **independent** random variables $X = X_1 + \dots + X_n$ are:

1. Instead of working with X , we switch to the **moment generating function** $e^{\lambda X}$, $\lambda > 0$ and apply Markov's inequality $\leadsto \mathbf{E} [e^{\lambda X}]$

General Recipe for Deriving Chernoff Bounds

Recipe

The **three main steps** in deriving Chernoff bounds for sums of **independent** random variables $X = X_1 + \dots + X_n$ are:

1. Instead of working with X , we switch to the **moment generating function** $e^{\lambda X}$, $\lambda > 0$ and apply Markov's inequality $\leadsto \mathbf{E} [e^{\lambda X}]$
2. Compute an upper bound for $\mathbf{E} [e^{\lambda X}]$ (using independence)

General Recipe for Deriving Chernoff Bounds

Recipe

The **three main steps** in deriving Chernoff bounds for sums of **independent** random variables $X = X_1 + \dots + X_n$ are:

1. Instead of working with X , we switch to the **moment generating function** $e^{\lambda X}$, $\lambda > 0$ and apply Markov's inequality $\leadsto \mathbf{E} [e^{\lambda X}]$
2. Compute an upper bound for $\mathbf{E} [e^{\lambda X}]$ (using independence)
3. Optimise value of λ to obtain best tail bound

Chernoff Bound: Proof

Chernoff Bound (General Form, Upper Tail)

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu.$$

Proof:

Chernoff Bound: Proof

Chernoff Bound (General Form, Upper Tail)

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu.$$

Proof:

1. For $\lambda > 0$,

$$\mathbf{P}[X \geq (1 + \delta)\mu] \underset{e^{\lambda x} \text{ is incr}}{=} \mathbf{P}\left[e^{\lambda X} \geq e^{\lambda(1+\delta)\mu}\right] \underset{\text{Markov}}{\leq} e^{-\lambda(1+\delta)\mu} \mathbf{E}\left[e^{\lambda X}\right]$$

Chernoff Bound: Proof

Chernoff Bound (General Form, Upper Tail)

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right]^\mu.$$

Proof:

1. For $\lambda > 0$,

$$\mathbf{P}[X \geq (1 + \delta)\mu] \stackrel{\text{e}^{\lambda X} \text{ is incr}}{=} \mathbf{P}\left[e^{\lambda X} \geq e^{\lambda(1+\delta)\mu}\right] \stackrel{\text{Markov}}{\leq} e^{-\lambda(1+\delta)\mu} \mathbf{E}\left[e^{\lambda X}\right]$$

$$2. \mathbf{E}\left[e^{\lambda X}\right] = \mathbf{E}\left[e^{\lambda \sum_{i=1}^n X_i}\right] \stackrel{\text{indep}}{=} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X_i}\right]$$

Chernoff Bound: Proof

Chernoff Bound (General Form, Upper Tail)

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$ it holds that

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq \left[\frac{e^\delta}{(1 + \delta)^{(1 + \delta)}} \right]^\mu.$$

Proof:

1. For $\lambda > 0$,

$$\mathbf{P}[X \geq (1 + \delta)\mu] \underset{\text{e}^{\lambda X} \text{ is incr}}{=} \mathbf{P}\left[e^{\lambda X} \geq e^{\lambda(1 + \delta)\mu}\right] \underset{\text{Markov}}{\leq} e^{-\lambda(1 + \delta)\mu} \mathbf{E}\left[e^{\lambda X}\right]$$

$$2. \mathbf{E}\left[e^{\lambda X}\right] = \mathbf{E}\left[e^{\lambda \sum_{i=1}^n X_i}\right] \underset{\text{indep}}{=} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X_i}\right]$$

3.

$$\mathbf{E}\left[e^{\lambda X_i}\right] = e^\lambda p_i + (1 - p_i) = 1 + p_i(e^\lambda - 1) \underset{1+x \leq e^x}{\leq} e^{p_i(e^\lambda - 1)}$$

Chernoff Bound: Proof

Chernoff Bound: Proof

1. For $\lambda > 0$,

$$\mathbf{P}[X \geq (1 + \delta)\mu] \underset{\text{e}^{\lambda x} \text{ is incr}}{=} \mathbf{P}\left[e^{\lambda X} \geq e^{\lambda(1+\delta)\mu}\right] \underset{\text{Markov}}{\leq} e^{-\lambda(1+\delta)\mu} \mathbf{E}\left[e^{\lambda X}\right]$$

$$2. \mathbf{E}\left[e^{\lambda X}\right] = \mathbf{E}\left[e^{\lambda \sum_{i=1}^n X_i}\right] \underset{\text{indep}}{=} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X_i}\right]$$

3.

$$\mathbf{E}\left[e^{\lambda X_i}\right] = e^{\lambda} p_i + (1 - p_i) = 1 + p_i(e^{\lambda} - 1) \underset{1+x \leq e^x}{\leq} e^{p_i(e^{\lambda} - 1)}$$

Chernoff Bound: Proof

1. For $\lambda > 0$,

$$\mathbf{P}[X \geq (1 + \delta)\mu] \stackrel{\text{e}^{\lambda x} \text{ is incr}}{=} \mathbf{P}\left[e^{\lambda X} \geq e^{\lambda(1+\delta)\mu}\right] \stackrel{\text{Markov}}{\leq} e^{-\lambda(1+\delta)\mu} \mathbf{E}\left[e^{\lambda X}\right]$$

$$2. \mathbf{E}\left[e^{\lambda X}\right] = \mathbf{E}\left[e^{\lambda \sum_{i=1}^n X_i}\right] \stackrel{\text{indep}}{=} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X_i}\right]$$

3.

$$\mathbf{E}\left[e^{\lambda X_i}\right] = e^{\lambda} p_i + (1 - p_i) = 1 + p_i(e^{\lambda} - 1) \stackrel{1+x \leq e^x}{\leq} e^{p_i(e^{\lambda} - 1)}$$

4. Putting all together

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq e^{-\lambda(1+\delta)\mu} \prod_{i=1}^n e^{p_i(e^{\lambda} - 1)} = e^{-\lambda(1+\delta)\mu} e^{\mu(e^{\lambda} - 1)}$$

Chernoff Bound: Proof

1. For $\lambda > 0$,

$$\mathbf{P}[X \geq (1 + \delta)\mu] \stackrel{\text{e}^{\lambda x} \text{ is incr}}{=} \mathbf{P}\left[e^{\lambda X} \geq e^{\lambda(1+\delta)\mu}\right] \stackrel{\text{Markov}}{\leq} e^{-\lambda(1+\delta)\mu} \mathbf{E}\left[e^{\lambda X}\right]$$

$$2. \mathbf{E}\left[e^{\lambda X}\right] = \mathbf{E}\left[e^{\lambda \sum_{i=1}^n X_i}\right] \stackrel{\text{indep}}{=} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X_i}\right]$$

3.

$$\mathbf{E}\left[e^{\lambda X_i}\right] = e^{\lambda} p_i + (1 - p_i) = 1 + p_i(e^{\lambda} - 1) \stackrel{1+x \leq e^x}{\leq} e^{p_i(e^{\lambda} - 1)}$$

4. Putting all together

$$\mathbf{P}[X \geq (1 + \delta)\mu] \leq e^{-\lambda(1+\delta)\mu} \prod_{i=1}^n e^{p_i(e^{\lambda} - 1)} = e^{-\lambda(1+\delta)\mu} e^{\mu(e^{\lambda} - 1)}$$

5. Choose $\lambda = \log(1 + \delta) > 0$ to get the result.

Chernoff Bounds: Lower Tails

We can also use Chernoff Bounds to show a random variable is **not too small** compared to its mean:

Chernoff Bounds (General Form, Lower Tail)

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then, for any $0 < \delta < 1$ it holds that

$$\mathbf{P}[X \leq (1 - \delta)\mu] \leq \left[\frac{e^{-\delta}}{(1 - \delta)^{1-\delta}} \right]^\mu,$$

and thus, by substitution, for any $t < \mu$,

$$\mathbf{P}[X \leq t] \leq e^{-\mu} \left(\frac{e\mu}{t} \right)^t.$$

Exercise on Supervision Sheet

Hint: multiply both sides by -1 and repeat the proof of the Chernoff Bound

Nicer Chernoff Bounds

“Nicer” Chernoff Bounds

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then,

Nicer Chernoff Bounds

“Nicer” Chernoff Bounds

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then,

- For all $t > 0$,

$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$$

$$\mathbf{P}[X \leq \mathbf{E}[X] - t] \leq e^{-2t^2/n}$$

Nicer Chernoff Bounds

“Nicer” Chernoff Bounds

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then,

- For all $t > 0$,

$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$$

$$\mathbf{P}[X \leq \mathbf{E}[X] - t] \leq e^{-2t^2/n}$$

- For $0 < \delta < 1$,

$$\mathbf{P}[X \geq (1 + \delta)\mathbf{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbf{E}[X]}{3}\right)$$

$$\mathbf{P}[X \leq (1 - \delta)\mathbf{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbf{E}[X]}{2}\right)$$

Nicer Chernoff Bounds

“Nicer” Chernoff Bounds

Suppose X_1, \dots, X_n are independent Bernoulli random variables with parameter p_i . Let $X = X_1 + \dots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^n p_i$. Then,

- For all $t > 0$,

$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$$

$$\mathbf{P}[X \leq \mathbf{E}[X] - t] \leq e^{-2t^2/n}$$

- For $0 < \delta < 1$,

$$\mathbf{P}[X \geq (1 + \delta)\mathbf{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbf{E}[X]}{3}\right)$$

$$\mathbf{P}[X \leq (1 - \delta)\mathbf{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbf{E}[X]}{2}\right)$$

All upper tail bounds hold even under a **relaxed independence assumption**:
For all $1 \leq i \leq n$ and $x_1, x_2, \dots, x_{i-1} \in \{0, 1\}$,

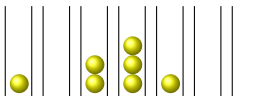
$$\mathbf{P}[X_i = 1 \mid X_1 = x_1, \dots, X_{i-1} = x_{i-1}] \leq p_i.$$

How to Derive Chernoff Bounds

Application 1: Balls into Bins

Appendix: More on Moment Generating Functions (non-examinable)

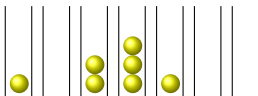
Balls into Bins



Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Balls into Bins

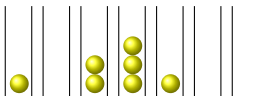


Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

- A very natural but also rich mathematical model

Balls into Bins

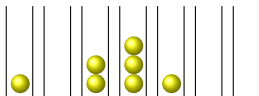


Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

- A very natural but also rich mathematical model
- In computer science, there are several interpretations:

Balls into Bins

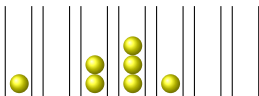


Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

- A very natural but also rich mathematical model
- In computer science, there are several interpretations:
 1. Bins are a hash table, balls are items
 2. Bins are processors and balls are jobs
 3. Bins are data servers and balls are queries

Balls into Bins



Balls into Bins Model

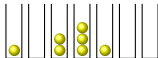
You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

- A very natural but also rich mathematical model
- In computer science, there are several interpretations:
 1. Bins are a hash table, balls are items
 2. Bins are processors and balls are jobs
 3. Bins are data servers and balls are queries



Exercise: Think about the relation between the Balls into Bins Model and the Coupon Collector Problem.

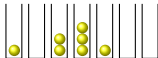
Balls into Bins: Bounding the Maximum Load (1/4)



Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Balls into Bins: Bounding the Maximum Load (1/4)

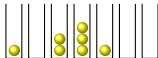


Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Question 1: How large is the maximum load if $m = 2n \log n$?

Balls into Bins: Bounding the Maximum Load (1/4)



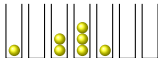
Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Question 1: How large is the maximum load if $m = 2n \log n$?

- Focus on an arbitrary single bin. Let X_i the indicator variable which is 1 iff ball i is assigned to this bin. Note that $p_i = \mathbf{P}[X_i = 1] = 1/n$.

Balls into Bins: Bounding the Maximum Load (1/4)



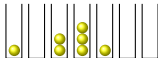
Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Question 1: How large is the maximum load if $m = 2n \log n$?

- Focus on an arbitrary single bin. Let X_i the indicator variable which is 1 iff ball i is assigned to this bin. Note that $p_i = \mathbf{P}[X_i = 1] = 1/n$.
- The total balls in the bin is given by $X := \sum_{i=1}^n X_i$.

Balls into Bins: Bounding the Maximum Load (1/4)



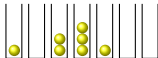
Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Question 1: How large is the maximum load if $m = 2n \log n$?

- Focus on an arbitrary single bin. Let X_i the indicator variable which is 1 iff ball i is assigned to this bin. Note that $p_i = \mathbf{P}[X_i = 1] = 1/n$.
- The total balls in the bin is given by $X := \sum_{i=1}^m X_i$.
- Since $m = 2n \log n$, then $\mu = \mathbf{E}[X] = 2 \log n$

Balls into Bins: Bounding the Maximum Load (1/4)



Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Question 1: How large is the maximum load if $m = 2n \log n$?

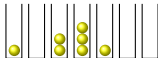
- Focus on an arbitrary single bin. Let X_i the indicator variable which is 1 iff ball i is assigned to this bin. Note that $p_i = \mathbf{P}[X_i = 1] = 1/n$.
- The total balls in the bin is given by $X := \sum_{i=1}^n X_i$.
- Since $m = 2n \log n$, then $\mu = \mathbf{E}[X] = 2 \log n$

$$\mathbf{P}[X \geq t] \leq e^{-\mu} (e\mu/t)^t$$

- By the Chernoff Bound,

$$\mathbf{P}[X \geq 6 \log n] \leq e^{-2 \log n} \left(\frac{2e \log n}{6 \log n} \right)^{6 \log n} \leq e^{-2 \log n} = n^{-2}$$

Balls into Bins: Bounding the Maximum Load (1/4)



Balls into Bins Model

You have m balls and n bins. Each ball is allocated in a bin picked independently and uniformly at random.

Question 1: How large is the maximum load if $m = 2n \log n$?

- Focus on an arbitrary single bin. Let X_i the indicator variable which is 1 iff ball i is assigned to this bin. Note that $p_i = \mathbf{P}[X_i = 1] = 1/n$.
- The total balls in the bin is given by $X := \sum_{i=1}^n X_i$.
- Since $m = 2n \log n$, then $\mu = \mathbf{E}[X] = 2 \log n$

here we could have used the “nicer” bounds as well!

$$\mathbf{P}[X \geq t] \leq e^{-\mu} (e\mu/t)^t$$

- By the Chernoff Bound,

$$\mathbf{P}[X \geq 6 \log n] \leq e^{-2 \log n} \left(\frac{2e \log n}{6 \log n} \right)^{6 \log n} \leq e^{-2 \log n} = n^{-2}$$

Balls into Bins: Bounding the Maximum Load (2/4)

- Let $\mathcal{E}_j := \{X(j) \geq 6 \log n\}$, that is, bin j receives at least $6 \log n$ balls.

Balls into Bins: Bounding the Maximum Load (2/4)

- Let $\mathcal{E}_j := \{X(j) \geq 6 \log n\}$, that is, bin j receives at least $6 \log n$ balls.
- We are interested in the probability that **at least** one bin receives at least $6 \log n$ balls \Rightarrow this is the event $\bigcup_{j=1}^n \mathcal{E}_j$

Balls into Bins: Bounding the Maximum Load (2/4)

- Let $\mathcal{E}_j := \{X(j) \geq 6 \log n\}$, that is, bin j receives at least $6 \log n$ balls.
- We are interested in the probability that **at least** one bin receives at least $6 \log n$ balls \Rightarrow this is the event $\bigcup_{j=1}^n \mathcal{E}_j$
- By the **Union Bound**,

$$\mathbf{P} \left[\bigcup_{j=1}^n \mathcal{E}_j \right] \leq \sum_{j=1}^n \mathbf{P}[\mathcal{E}_j] \leq n \cdot n^{-2} = n^{-1}.$$

Balls into Bins: Bounding the Maximum Load (2/4)

- Let $\mathcal{E}_j := \{X(j) \geq 6 \log n\}$, that is, bin j receives at least $6 \log n$ balls.
- We are interested in the probability that **at least** one bin receives at least $6 \log n$ balls \Rightarrow this is the event $\bigcup_{j=1}^n \mathcal{E}_j$
- By the **Union Bound**,

$$\mathbf{P} \left[\bigcup_{j=1}^n \mathcal{E}_j \right] \leq \sum_{j=1}^n \mathbf{P}[\mathcal{E}_j] \leq n \cdot n^{-2} = n^{-1}.$$

- Therefore **whp**, no bin receives at least $6 \log n$ balls

Balls into Bins: Bounding the Maximum Load (2/4)

- Let $\mathcal{E}_j := \{X(j) \geq 6 \log n\}$, that is, bin j receives at least $6 \log n$ balls.
- We are interested in the probability that **at least** one bin receives at least $6 \log n$ balls \Rightarrow this is the event $\bigcup_{j=1}^n \mathcal{E}_j$
- By the **Union Bound**,

$$\mathbf{P} \left[\bigcup_{j=1}^n \mathcal{E}_j \right] \leq \sum_{j=1}^n \mathbf{P}[\mathcal{E}_j] \leq n \cdot n^{-2} = n^{-1}.$$

- Therefore **whp**, no bin receives at least $6 \log n$ balls

whp stands for *with high probability*:

An event \mathcal{E} (that implicitly depends on an input parameter n) occurs **whp** if

$$\mathbf{P}[\mathcal{E}] \rightarrow 1 \text{ as } n \rightarrow \infty.$$

This is a very standard notation in randomised algorithms
but it may vary from author to author. **Be careful!**

Balls into Bins: Bounding the Maximum Load (2/4)

- Let $\mathcal{E}_j := \{X(j) \geq 6 \log n\}$, that is, bin j receives at least $6 \log n$ balls.
- We are interested in the probability that **at least** one bin receives at least $6 \log n$ balls \Rightarrow this is the event $\bigcup_{j=1}^n \mathcal{E}_j$
- By the **Union Bound**,

$$\mathbf{P} \left[\bigcup_{j=1}^n \mathcal{E}_j \right] \leq \sum_{j=1}^n \mathbf{P}[\mathcal{E}_j] \leq n \cdot n^{-2} = n^{-1}.$$

- Therefore **whp**, no bin receives at least $6 \log n$ balls
- By **pigeonhole principle**, the max loaded bin receives at least $2 \log n$ balls. Hence our bound is pretty sharp.

whp stands for *with high probability*:

An event \mathcal{E} (that implicitly depends on an input parameter n) occurs **whp** if

$$\mathbf{P}[\mathcal{E}] \rightarrow 1 \text{ as } n \rightarrow \infty.$$

This is a very standard notation in randomised algorithms
but it may vary from author to author. **Be careful!**

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

- Using the Chernoff Bound:

$$\mathbf{P}[X \geq t] \leq e^{-\mu} (e\mu/t)^t$$

$$\mathbf{P}[X \geq t] \leq e^{-1} \left(\frac{e}{t}\right)^t \leq \left(\frac{e}{t}\right)^t$$

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

- Using the Chernoff Bound:

$$\mathbf{P}[X \geq t] \leq e^{-\mu} (e\mu/t)^t$$

$$\mathbf{P}[X \geq t] \leq e^{-1} \left(\frac{e}{t}\right)^t \leq \left(\frac{e}{t}\right)^t$$

- By setting $t = 4 \log n / \log \log n$, we claim to obtain $\mathbf{P}[X \geq t] \leq n^{-2}$.

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

- Using the Chernoff Bound:

$$\mathbf{P}[X \geq t] \leq e^{-\mu} (e\mu/t)^t$$

$$\mathbf{P}[X \geq t] \leq e^{-1} \left(\frac{e}{t}\right)^t \leq \left(\frac{e}{t}\right)^t$$

- By setting $t = 4 \log n / \log \log n$, we claim to obtain $\mathbf{P}[X \geq t] \leq n^{-2}$.
- Indeed:

$$\left(\frac{e \log \log n}{4 \log n}\right)^{4 \log n / \log \log n} = \exp\left(\frac{4 \log n}{\log \log n} \cdot \log\left(\frac{e \log \log n}{4 \log n}\right)\right)$$

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

- Using the Chernoff Bound:

$$\mathbf{P}[X \geq t] \leq e^{-\mu} (e\mu/t)^t$$

$$\mathbf{P}[X \geq t] \leq e^{-1} \left(\frac{e}{t}\right)^t \leq \left(\frac{e}{t}\right)^t$$

- By setting $t = 4 \log n / \log \log n$, we claim to obtain $\mathbf{P}[X \geq t] \leq n^{-2}$.
- Indeed:

$$\left(\frac{e \log \log n}{4 \log n}\right)^{4 \log n / \log \log n} = \exp\left(\frac{4 \log n}{\log \log n} \cdot \log\left(\frac{e \log \log n}{4 \log n}\right)\right)$$

- The term inside the exponential is

$$\frac{4 \log n}{\log \log n} \cdot (\log(e/4) + \log \log \log n - \log \log n)$$

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

- Using the Chernoff Bound:

$$\mathbf{P}[X \geq t] \leq e^{-\mu}(e\mu/t)^t$$

$$\mathbf{P}[X \geq t] \leq e^{-1} \left(\frac{e}{t}\right)^t \leq \left(\frac{e}{t}\right)^t$$

- By setting $t = 4 \log n / \log \log n$, we claim to obtain $\mathbf{P}[X \geq t] \leq n^{-2}$.
- Indeed:

$$\left(\frac{e \log \log n}{4 \log n}\right)^{4 \log n / \log \log n} = \exp\left(\frac{4 \log n}{\log \log n} \cdot \log\left(\frac{e \log \log n}{4 \log n}\right)\right)$$

- The term inside the exponential is

$$\frac{4 \log n}{\log \log n} \cdot (\log(e/4) + \log \log \log n - \log \log n) \leq \frac{4 \log n}{\log \log n} \left(-\frac{1}{2} \log \log n\right),$$

This inequality only works for large enough n .

Balls into Bins: Bounding the Maximum Load (3/4)

Question 2: How large is the maximum load if $m = n$?

- Using the Chernoff Bound:

$$\mathbf{P}[X \geq t] \leq e^{-\mu}(e\mu/t)^t$$

$$\mathbf{P}[X \geq t] \leq e^{-1} \left(\frac{e}{t}\right)^t \leq \left(\frac{e}{t}\right)^t$$

- By setting $t = 4 \log n / \log \log n$, we claim to obtain $\mathbf{P}[X \geq t] \leq n^{-2}$.
- Indeed:

$$\left(\frac{e \log \log n}{4 \log n}\right)^{4 \log n / \log \log n} = \exp\left(\frac{4 \log n}{\log \log n} \cdot \log\left(\frac{e \log \log n}{4 \log n}\right)\right)$$

- The term inside the exponential is

$$\frac{4 \log n}{\log \log n} \cdot (\log(e/4) + \log \log \log n - \log \log n) \leq \frac{4 \log n}{\log \log n} \left(-\frac{1}{2} \log \log n\right),$$

obtaining that $\mathbf{P}[X \geq t] \leq n^{-4/2} = n^{-2}$.

This inequality only works for large enough n .

We just proved that

$$\mathbf{P}[X \geq 4 \log n / \log \log n] \leq n^{-2},$$

thus by the **Union Bound**, no bin receives more than $\Omega(\log n / \log \log n)$ balls with probability at least $1 - 1/n$. □

We just proved that

$$\mathbf{P}[X \geq 4 \log n / \log \log n] \leq n^{-2},$$

thus by the **Union Bound**, no bin receives more than $\Omega(\log n / \log \log n)$ balls with probability at least $1 - 1/n$. \square

- One can prove that **whp** at least one bin receives at least $c \log n / \log \log n$ balls, for some constant $c > 0$.

Conclusions

- If the number of balls is $2 \log n$ times n (the number of bins), then to distribute balls at random is a **good algorithm**

Conclusions

- If the number of balls is $2 \log n$ times n (the number of bins), then to distribute balls at random is a **good algorithm**
 - This is because the worst case maximum load is whp. $6 \log n$, while the average load is $2 \log n$

Conclusions

- If the number of balls is $2 \log n$ times n (the number of bins), then to distribute balls at random is a **good algorithm**
 - This is because the worst case maximum load is whp. $6 \log n$, while the average load is $2 \log n$
- For the case $m = n$, the algorithm is **not good**, since the maximum load is whp. $\Theta(\log n / \log \log n)$, while the average load is 1.

Conclusions

- If the number of balls is $2 \log n$ times n (the number of bins), then to distribute balls at random is a **good algorithm**
 - This is because the worst case maximum load is whp. $6 \log n$, while the average load is $2 \log n$
- For the case $m = n$, the algorithm is **not good**, since the maximum load is whp. $\Theta(\log n / \log \log n)$, while the average load is 1.

A Better Load Balancing Approach

For any $m \geq n$, we can improve this by sampling **two bins** in each step and then assign the ball into the bin with lesser load.

- If the number of balls is $2 \log n$ times n (the number of bins), then to distribute balls at random is a **good algorithm**
 - This is because the worst case maximum load is whp. $6 \log n$, while the average load is $2 \log n$
- For the case $m = n$, the algorithm is **not good**, since the maximum load is whp. $\Theta(\log n / \log \log n)$, while the average load is 1.

A Better Load Balancing Approach

For any $m \geq n$, we can improve this by sampling **two bins** in each step and then assign the ball into the bin with lesser load.

\Rightarrow for $m = n$ this gives a maximum load of $\log_2 \log n + \Theta(1)$ w.p. $1 - 1/n$.

Conclusions

- If the number of balls is $2 \log n$ times n (the number of bins), then to distribute balls at random is a **good algorithm**
 - This is because the worst case maximum load is whp. $6 \log n$, while the average load is $2 \log n$
- For the case $m = n$, the algorithm is **not good**, since the maximum load is whp. $\Theta(\log n / \log \log n)$, while the average load is 1.

A Better Load Balancing Approach

For any $m \geq n$, we can improve this by sampling **two bins** in each step and then assign the ball into the bin with lesser load.

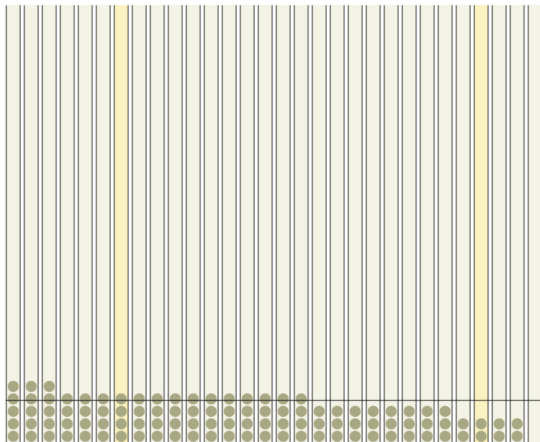
\Rightarrow for $m = n$ this gives a maximum load of $\log_2 \log n + \Theta(1)$ w.p. $1 - 1/n$.

This is called the **power of two choices**: It is a common technique to improve the performance of randomised algorithms (covered in Chapter 17 of the textbook by Mitzenmacher and Upfal)



*For “the discovery and analysis of balanced allocations, known as the **power of two choices**, and their extensive applications to practice.”*

*“These include **i-Google’s web index**, **Akamai’s overlay routing network**, and highly reliable **distributed data storage** systems used by Microsoft and Dropbox, which are all based on variants of the power of two choices paradigm. There are many other software systems that use balanced allocations as an important ingredient.”*



Sampled two bins u.a.r.

▶ Next Step Advance by 50 Go Trim Interval (ms): 1 ☒ Sort in each round ☒ Auto-trim ☒ Draw mean
Number of bins: 3 Capacity: 3 Reset Process: Batch size: 3 Noise (g): 5
Plot: Add Initialise configuration: Init

https://www.dimitrioslos.com/balls_and_bins/visualiser.html

How to Derive Chernoff Bounds

Application 1: Balls into Bins

Appendix: More on Moment Generating Functions (non-examinable)

Moment Generating Functions (non-examinable)

— Moment-Generating Function —

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Moment Generating Functions (non-examinable)

Moment-Generating Function

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Using power series of e and differentiating shows that $M_X(t)$ encapsulates all moments of X .

Moment Generating Functions (non-examinable)

Moment-Generating Function

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Using power series of e and differentiating shows that $M_X(t)$ encapsulates all moments of X .

Lemma

1. If X and Y are two r.v.'s with $M_X(t) = M_Y(t)$ for all $t \in (-\delta, +\delta)$ for some $\delta > 0$, then the distributions X and Y are identical.
2. If X and Y are **independent** random variables, then

$$M_{X+Y}(t) = M_X(t) \cdot M_Y(t).$$

Moment Generating Functions (non-examinable)

Moment-Generating Function

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Using power series of e and differentiating shows that $M_X(t)$ encapsulates all moments of X .

Lemma

1. If X and Y are two r.v.'s with $M_X(t) = M_Y(t)$ for all $t \in (-\delta, +\delta)$ for some $\delta > 0$, then the distributions X and Y are identical.
2. If X and Y are **independent** random variables, then

$$M_{X+Y}(t) = M_X(t) \cdot M_Y(t).$$

Proof of 2:

$$M_{X+Y}(t) = \mathbf{E} \left[e^{t(X+Y)} \right] = \mathbf{E} \left[e^{tX} \cdot e^{tY} \right] \stackrel{(!)}{=} \mathbf{E} \left[e^{tX} \right] \cdot \mathbf{E} \left[e^{tY} \right] = M_X(t) M_Y(t) \quad \square$$

Randomised Algorithms

Lecture 3: Concentration Inequalities, Application to Quick-Sort, Extensions

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: More on Moment Generating Functions (non-examinable)

QUICKSORT (Input $A[1], A[2], \dots, A[n]$)

- 1: Pick an element from the array, the so-called **pivot**
- 2: **If** $n = 0$ or $n = 1$ **then**
- 3: **return** A
- 4: **else**
- 5: Create two subarrays A_1 and A_2 (without the pivot) such that:
- 6: A_1 contains the elements that are **smaller than the pivot**
- 7: A_2 contains the elements that are **greater (or equal) than the pivot**
- 8: QUICKSORT(A_1)
- 9: QUICKSORT(A_2)
- 10: **return** A

QUICKSORT (Input $A[1], A[2], \dots, A[n]$)

- 1: Pick an element from the array, the so-called **pivot**
- 2: **If** $n = 0$ or $n = 1$ **then**
- 3: **return** A
- 4: **else**
- 5: Create two subarrays A_1 and A_2 (without the pivot) such that:
- 6: A_1 contains the elements that are **smaller than the pivot**
- 7: A_2 contains the elements that are **greater (or equal) than the pivot**
- 8: QUICKSORT(A_1)
- 9: QUICKSORT(A_2)
- 10: **return** A

- **Example:** Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.

QUICKSORT (Input $A[1], A[2], \dots, A[n]$)

- 1: Pick an element from the array, the so-called **pivot**
- 2: **If** $n = 0$ or $n = 1$ **then**
- 3: **return** A
- 4: **else**
- 5: Create two subarrays A_1 and A_2 (without the pivot) such that:
- 6: A_1 contains the elements that are **smaller than the pivot**
- 7: A_2 contains the elements that are **greater (or equal) than the pivot**
- 8: QUICKSORT(A_1)
- 9: QUICKSORT(A_2)
- 10: **return** A

- **Example:** Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
⇒ $A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$

QUICKSORT (Input $A[1], A[2], \dots, A[n]$)

- 1: Pick an element from the array, the so-called **pivot**
- 2: **If** $n = 0$ or $n = 1$ **then**
- 3: **return** A
- 4: **else**
- 5: Create two subarrays A_1 and A_2 (without the pivot) such that:
- 6: A_1 contains the elements that are **smaller than the pivot**
- 7: A_2 contains the elements that are **greater (or equal) than the pivot**
- 8: QUICKSORT(A_1)
- 9: QUICKSORT(A_2)
- 10: **return** A

- **Example:** Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
⇒ $A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$
- **Worst-Case Complexity** (number of comparisons) is $\Theta(n^2)$,

QUICKSORT (Input $A[1], A[2], \dots, A[n]$)

- 1: Pick an element from the array, the so-called **pivot**
- 2: **If** $n = 0$ or $n = 1$ **then**
- 3: **return** A
- 4: **else**
- 5: Create two subarrays A_1 and A_2 (without the pivot) such that:
- 6: A_1 contains the elements that are **smaller than the pivot**
- 7: A_2 contains the elements that are **greater (or equal) than the pivot**
- 8: QUICKSORT(A_1)
- 9: QUICKSORT(A_2)
- 10: **return** A

- **Example:** Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
⇒ $A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$
- **Worst-Case Complexity** (number of comparisons) is $\Theta(n^2)$,
while **Average-Case Complexity** is $O(n \log n)$.

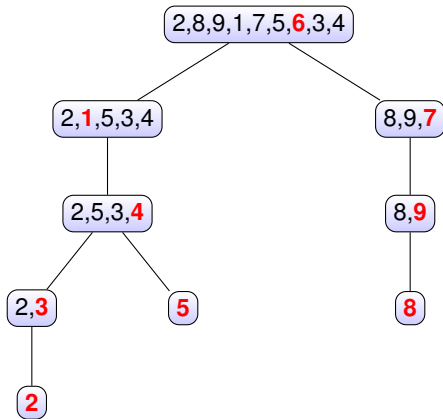
QUICKSORT (Input $A[1], A[2], \dots, A[n]$)

- 1: Pick an element from the array, the so-called **pivot**
- 2: **If** $n = 0$ or $n = 1$ **then**
- 3: **return** A
- 4: **else**
- 5: Create two subarrays A_1 and A_2 (without the pivot) such that:
- 6: A_1 contains the elements that are **smaller than the pivot**
- 7: A_2 contains the elements that are **greater (or equal) than the pivot**
- 8: QUICKSORT(A_1)
- 9: QUICKSORT(A_2)
- 10: **return** A

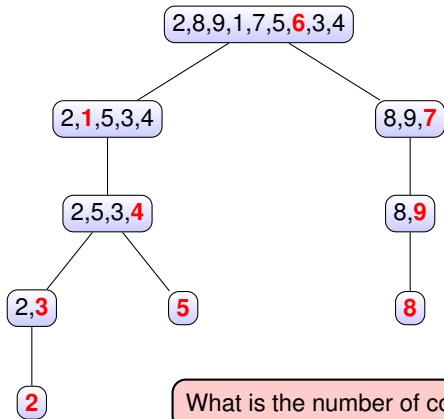
- **Example:** Let $A = (2, 8, 9, 1, 7, 5, 6, 3, 4)$ with $A[7] = 6$ as pivot.
⇒ $A_1 = (2, 1, 5, 3, 4)$ and $A_2 = (8, 9, 7)$
- **Worst-Case Complexity** (number of comparisons) is $\Theta(n^2)$,
while **Average-Case Complexity** is $O(n \log n)$.

We will now give a proof of this “well-known” result!

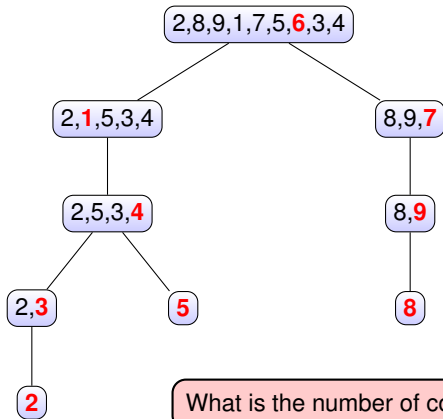
QuickSort: How to Count Comparisons



QuickSort: How to Count Comparisons



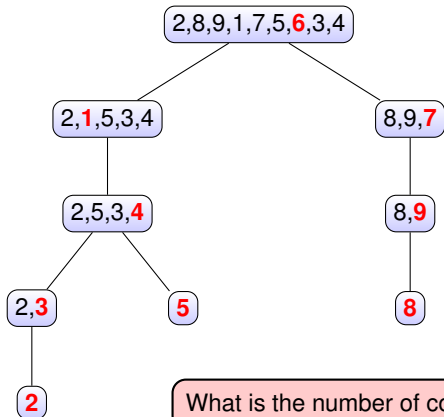
QuickSort: How to Count Comparisons



What is the number of comparisons?

Note that the number of comparison by QUICKSORT is equivalent to the sum of the depths of all nodes in the tree (why?).

QuickSort: How to Count Comparisons



What is the number of comparisons?

Note that the number of comparison by QUICKSORT is equivalent to the sum of the depths of all nodes in the tree (why?). In this case:

$$0 + 1 + 1 + 2 + 2 + 3 + 3 + 3 + 4 = 19.$$

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random.**

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random.**

This should be your standard answer in this course 😊

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random.**

This should be your standard answer in this course 😊

Let us analyse QUICKSORT with **random** pivots.

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random.**

This should be your standard answer in this course 😊

Let us analyse QUICKSORT with **random** pivots.

1. Assume A consists of n different numbers, w.l.o.g., $\{1, 2, \dots, n\}$

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random.**

This should be your standard answer in this course 😊

Let us analyse QUICKSORT with **random** pivots.

1. Assume A consists of n different numbers, w.l.o.g., $\{1, 2, \dots, n\}$
2. Let H_i be the **deepest level** where element i appears in the tree.
Then the number of comparison is $H = \sum_{i=1}^n H_i$

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random**.

This should be your standard answer in this course 😊

Let us analyse QUICKSORT with **random** pivots.

1. Assume A consists of n different numbers, w.l.o.g., $\{1, 2, \dots, n\}$
2. Let H_i be the **deepest level** where element i appears in the tree.
Then the number of comparison is $H = \sum_{i=1}^n H_i$
3. We will prove that there exists $C > 0$ such that

$$\mathbf{P}[H \leq Cn \log n] \geq 1 - n^{-1}.$$

Randomised QuickSort: Analysis (1/4)

How to pick a **good pivot**? We don't, **just pick one at random**.

This should be your standard answer in this course 😊

Let us analyse QUICKSORT with **random** pivots.

1. Assume A consists of n different numbers, w.l.o.g., $\{1, 2, \dots, n\}$
2. Let H_i be the **deepest level** where element i appears in the tree.
Then the number of comparison is $H = \sum_{i=1}^n H_i$
3. We will prove that there exists $C > 0$ such that

$$\mathbf{P}[H \leq Cn \log n] \geq 1 - n^{-1}.$$

4. Actually, we will prove sth slightly stronger:

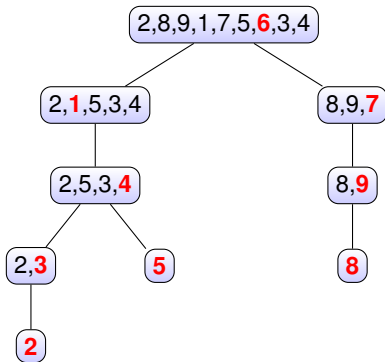
$$\mathbf{P}\left[\bigcap_{i=1}^n \{H_i \leq C \log n\}\right] \geq 1 - n^{-1}.$$

Randomised QuickSort: Analysis (2/4)

- Let P be a path from the root to the deepest level of some element

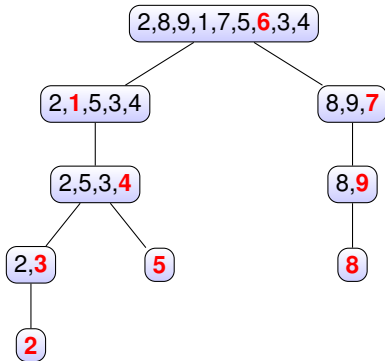
Randomised QuickSort: Analysis (2/4)

- Let P be a path from the root to the deepest level of some element



Randomised QuickSort: Analysis (2/4)

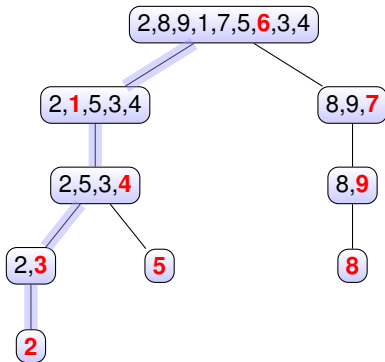
- Let P be a path from the root to the deepest level of some element



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

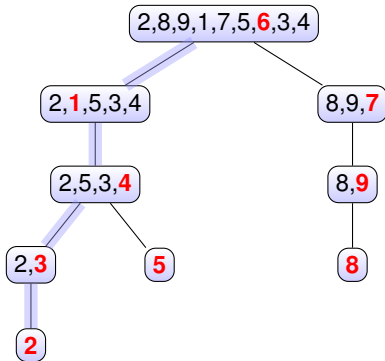
- Let P be a path from the root to the deepest level of some element



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

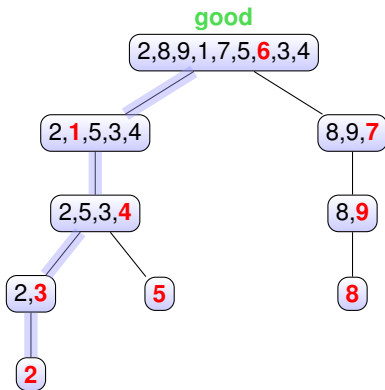
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

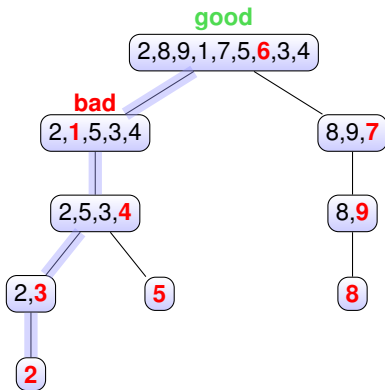
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

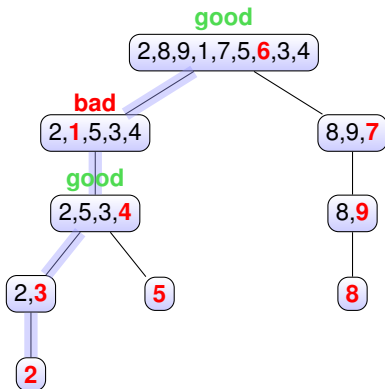
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

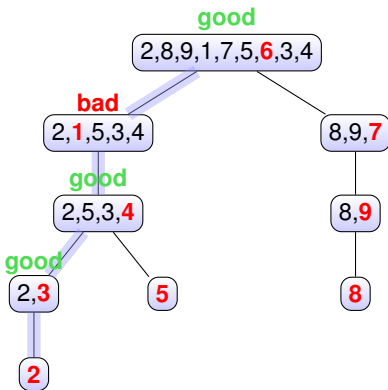
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

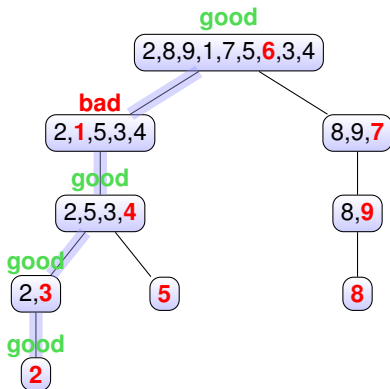
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

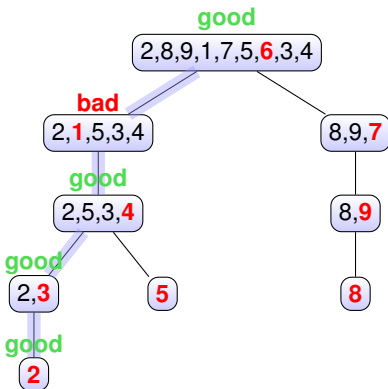
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

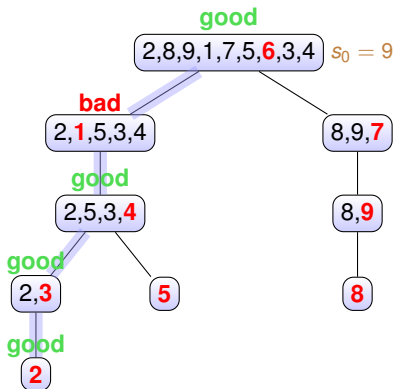
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**
- Further let s_t be the **size** of the array at level t in P .



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

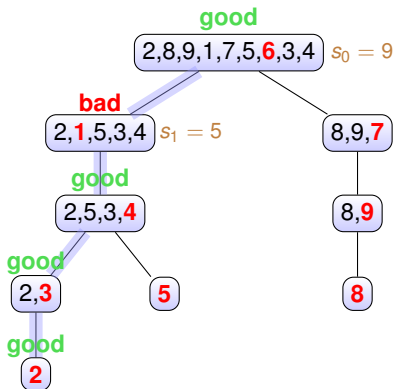
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**
- Further let s_t be the **size** of the array at level t in P .



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

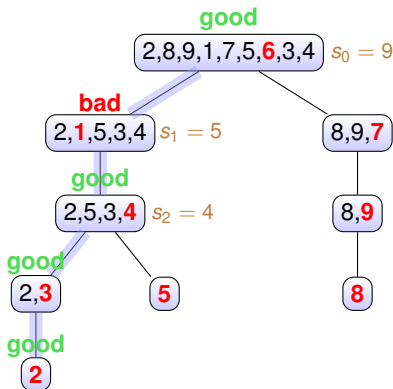
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**
- Further let s_t be the **size** of the array at level t in P .



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

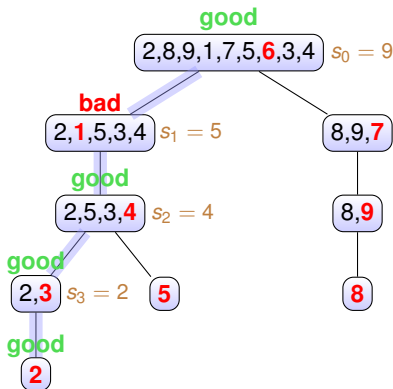
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**
- Further let s_t be the **size** of the array at level t in P .



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

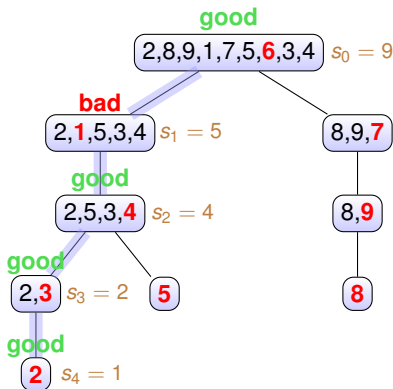
- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**
- Further let s_t be the **size** of the array at level t in P .



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (2/4)

- Let P be a path from the root to the deepest level of some element
 - A node in P is called **good** if the corresponding pivot partitions the array into two subarrays each of size at most $2/3$ of the previous one
 - otherwise, the node is **bad**
- Further let s_t be the **size** of the array at level t in P .



- Element 2: $(2, 8, 9, 1, 7, 5, 6, 3, 4) \rightarrow (2, 1, 5, 3, 4) \rightarrow (2, 5, 3, 4) \rightarrow (2, 3) \rightarrow (2)$

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case, good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case, bad** node: $s_{k+1} \leq s_k$.

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- **First Case**, **good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- **Second Case**, **bad** node: $s_{k+1} \leq s_k$.

⇒ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path P .

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- First Case, good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- Second Case, bad** node: $s_{k+1} \leq s_k$.

This even holds always,
i.e., deterministically!

\Rightarrow There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path P .

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- First Case, good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- Second Case, bad** node: $s_{k+1} \leq s_k$.

This even holds always,
i.e., deterministically!

⇒ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path P .

- Assume $|P| \geq C \log n$ for $C := 24$

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- First Case, good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- Second Case, bad** node: $s_{k+1} \leq s_k$.

This even holds always,
i.e., deterministically!

- ⇒ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path P .
- Assume $|P| \geq C \log n$ for $C := 24$
⇒ number of **bad** nodes in the first $24 \log n$ levels is more than $21 \log n$.

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \dots, n\}$ and construct the path $P = P(i)$ one level by one
- For P to proceed from level k to $k + 1$, the condition $s_k > 1$ is necessary

How far could such a path P possibly run until we have $s_k = 1$?

- We start with $s_0 = n$
- First Case, good** node: $s_{k+1} \leq \frac{2}{3} \cdot s_k$.
- Second Case, bad** node: $s_{k+1} \leq s_k$.

This even holds always,
i.e., deterministically!

⇒ There are at most $T = \frac{\log n}{\log(3/2)} < 3 \log n$ many **good** nodes on any path P .

- Assume $|P| \geq C \log n$ for $C := 24$

⇒ number of **bad** nodes in the first $24 \log n$ levels is more than $21 \log n$.

Let us now upper bound the probability that this “bad event” happens!

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the deepest level of element i .

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the deepest level of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the deepest level of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.

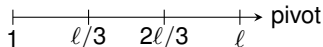
Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.

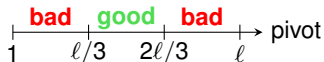
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$



Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the deepest level of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.

- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$



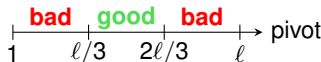
Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the deepest level of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)



Randomised QuickSort: Analysis (4/4)

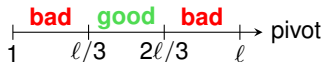
- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



Question: Edge Case: What if the path P does not reach level j ?

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



Question: Edge Case: What if the path P does not reach level j ?

Answer: We can then simply define X_j as 0 (deterministically).

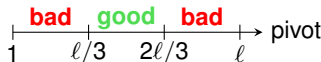
Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the deepest level of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)



Randomised QuickSort: Analysis (4/4)

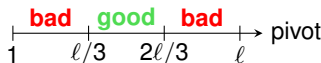
- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



We can now apply the “nicer” **Chernoff Bound**!

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)

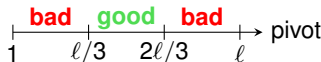


We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)

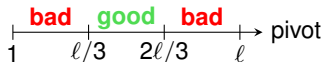


We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” **Chernoff Bounds**

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



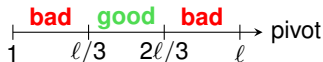
We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” **Chernoff Bounds**

$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$$

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



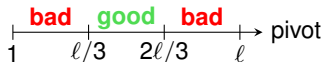
We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” Chernoff Bounds $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$

$$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n]$$

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



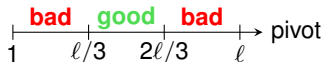
We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
- Then, by the “nicer” **Chernoff Bounds** $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$

$$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2 / (24 \log n)}$$

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)

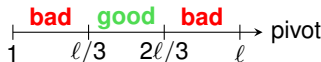


We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
 - Then, by the “nicer” Chernoff Bounds $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$
- $$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2 / (24 \log n)} \leq n^{-2}.$$

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)

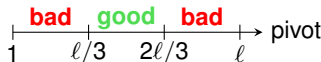


We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
 - Then, by the “nicer” Chernoff Bounds $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$
- $$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2 / (24 \log n)} \leq n^{-2}.$$
- Hence P has more than $24 \log n$ nodes with probability at most n^{-2} .

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)

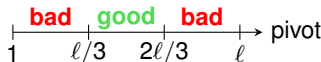


We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
 - Then, by the “nicer” **Chernoff Bounds** $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$
- $$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2 / (24 \log n)} \leq n^{-2}.$$
- Hence P has more than $24 \log n$ nodes with probability at most n^{-2} .
 - As there are in total n paths, by the **union bound**, the probability that at least one of them has more than $24 \log n$ nodes is at most n^{-1} .

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)

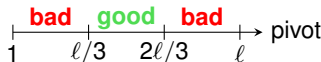


We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
 - Then, by the “nicer” Chernoff Bounds $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$
- $$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2 / (24 \log n)} \leq n^{-2}.$$
- Hence P has more than $24 \log n$ nodes with probability at most n^{-2} .
 - As there are in total n paths, by the **union bound**, the probability that at least one of them has more than $24 \log n$ nodes is at most n^{-1} .
 - This implies $\mathbf{P}[\bigcap_{i=1}^n \{H_i \leq 24 \log n\}] \geq 1 - n^{-1}$, as needed.

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ nodes of P to the **deepest level** of element i .
- For any level $j \in \{0, 1, \dots, 24 \log n - 1\}$, define an indicator variable X_j :
 - $X_j = 1$ if the node at level j is **bad**,
 - $X_j = 0$ if the node at level j is **good**.
- $\mathbf{P}[X_j = 1 \mid X_0 = x_0, \dots, X_{j-1} = x_{j-1}] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies **relaxed independence assumption** (Lecture 2)



We can now apply the “nicer” **Chernoff Bound**!

- We have $\mathbf{E}[X] \leq (2/3) \cdot 24 \log n = 16 \log n$
 - Then, by the “nicer” Chernoff Bounds $\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq e^{-2t^2/n}$
- $$\mathbf{P}[X > 21 \log n] \leq \mathbf{P}[X > \mathbf{E}[X] + 5 \log n] \leq e^{-2(5 \log n)^2 / (24 \log n)} \leq n^{-2}.$$
- Hence P has more than $24 \log n$ nodes with probability at most n^{-2} .
 - As there are in total n paths, by the **union bound**, the probability that at least one of them has more than $24 \log n$ nodes is at most n^{-1} .
 - This implies $\mathbf{P}[\bigcap_{i=1}^n \{H_i \leq 24 \log n\}] \geq 1 - n^{-1}$, as needed. \square

Randomised QuickSort: Final Remarks

- Well-known: any comparison-based sorting algorithm needs $\Omega(n \log n)$
- A classical result: **expected number** of comparison of **randomised QUICKSORT** is $2n \log n + O(n)$ (see, e.g., book by Mitzenmacher & Upfal)

- Well-known: any comparison-based sorting algorithm needs $\Omega(n \log n)$
- A classical result: **expected number** of comparison of **randomised QUICKSORT** is $2n \log n + O(n)$ (see, e.g., book by Mitzenmacher & Upfal)



Exercise: [Ex 2-3.6] Our upper bound of $O(n \log n)$ **whp** also immediately implies a $O(n \log n)$ bound on the expected number of comparisons!

Randomised QuickSort: Final Remarks

- Well-known: any comparison-based sorting algorithm needs $\Omega(n \log n)$
- A classical result: **expected number** of comparison of **randomised QUICKSORT** is $2n \log n + O(n)$ (see, e.g., book by Mitzenmacher & Upfal)



Exercise: [Ex 2-3.6] Our upper bound of $O(n \log n)$ **whp** also immediately implies a $O(n \log n)$ bound on the expected number of comparisons!

- It is possible to **deterministically** find the best pivot element that divides the array into two subarrays of the same size.
- The latter requires to compute the **median** of the array in linear time, which is not easy...
- The presented **randomised** algorithm for QUICKSORT is much **easier to implement!**

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: More on Moment Generating Functions (non-examinable)

- Besides **sums of independent Bernoulli** random variables, **sums of independent and bounded** random variables are very frequent in applications.

- Besides **sums of independent Bernoulli** random variables, **sums of independent and bounded** random variables are very frequent in applications.
- Unfortunately the distribution of the X_i may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.

- Besides **sums of independent Bernoulli** random variables, **sums of independent and bounded** random variables are very frequent in applications.
- Unfortunately the distribution of the X_i may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

- Besides **sums of independent Bernoulli** random variables, **sums of independent and bounded** random variables are very frequent in applications.
- Unfortunately the distribution of the X_i may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

— Hoeffding's Extension Lemma —

Let X be a random variable with mean 0 such that $a \leq X \leq b$. Then for all $\lambda \in \mathbb{R}$,

$$\mathbf{E} \left[e^{\lambda X} \right] \leq \exp \left(\frac{(b-a)^2 \lambda^2}{8} \right)$$

Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, **sums of independent and bounded** random variables are very frequent in applications.
- Unfortunately the distribution of the X_i may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

You can always consider
 $X' = X - \mathbf{E}[X]$

Hoeffding's Extension Lemma

Let X be a random variable with mean 0 such that $a \leq X \leq b$. Then for all $\lambda \in \mathbb{R}$,

$$\mathbf{E} \left[e^{\lambda X} \right] \leq \exp \left(\frac{(b-a)^2 \lambda^2}{8} \right)$$

Hoeffding's Extension

- Besides **sums of independent Bernoulli** random variables, **sums of independent and bounded** random variables are very frequent in applications.
- Unfortunately the distribution of the X_i may be unknown or hard to compute, thus it will be hard to compute the moment-generating function.
- Hoeffding's Lemma helps us here:

You can always consider
 $X' = X - \mathbf{E}[X]$

Hoeffding's Extension Lemma

Let X be a random variable with mean 0 such that $a \leq X \leq b$. Then for all $\lambda \in \mathbb{R}$,

$$\mathbf{E} \left[e^{\lambda X} \right] \leq \exp \left(\frac{(b-a)^2 \lambda^2}{8} \right)$$

We omit the proof of this lemma!

Hoeffding Bounds

Hoeffding's Inequality

Let X_1, \dots, X_n be independent random variables with mean μ_i such that $a_i \leq X_i \leq b_i$. Let $X = X_1 + \dots + X_n$, and let $\mu = \mathbf{E}[X] = \sum_{i=1}^n \mu_i$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Hoeffding Bounds

Hoeffding's Inequality

Let X_1, \dots, X_n be independent random variables with mean μ_i such that $a_i \leq X_i \leq b_i$. Let $X = X_1 + \dots + X_n$, and let $\mu = \mathbf{E}[X] = \sum_{i=1}^n \mu_i$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Proof Outline (skipped):

- Let $X'_i = X_i - \mu_i$ and $X' = X'_1 + \dots + X'_n$, then $\mathbf{P}[X \geq \mu + t] = \mathbf{P}[X' \geq t]$
- $\mathbf{P}[X' \geq t] \leq e^{-\lambda t} \prod_{i=1}^n \mathbf{E}\left[e^{\lambda X'_i}\right] \leq \exp\left[-\lambda t + \frac{\lambda^2}{8} \sum_{i=1}^n (b_i - a_i)^2\right]$
- Choose $\lambda = \frac{4t}{\sum_{i=1}^n (b_i - a_i)^2}$ to get the result.

This is not “magic” – you just need to optimise λ !

Method of Bounded Differences

Framework

Suppose, we have **independent** random variables X_1, \dots, X_n . We want to study the random variable:

$$f(X_1, \dots, X_n)$$

Framework

Suppose, we have **independent** random variables X_1, \dots, X_n . We want to study the random variable:

$$f(X_1, \dots, X_n)$$

Some examples:

1. $X = X_1 + \dots + X_n$ (our setting earlier)

Framework

Suppose, we have **independent** random variables X_1, \dots, X_n . We want to study the random variable:

$$f(X_1, \dots, X_n)$$

Some examples:

1. $X = X_1 + \dots + X_n$ (our setting earlier)
2. In **balls into bins**, X_i indicates where ball i is allocated, and $f(X_1, \dots, X_m)$ is the number of empty bins

Framework

Suppose, we have **independent** random variables X_1, \dots, X_n . We want to study the random variable:

$$f(X_1, \dots, X_n)$$

Some examples:

1. $X = X_1 + \dots + X_n$ (our setting earlier)
2. In **balls into bins**, X_i indicates where ball i is allocated, and $f(X_1, \dots, X_m)$ is the number of empty bins
3. In a **randomly generated graph**, X_i indicates if the i -th edge is present and $f(X_1, \dots, X_m)$ represents the number of connected components of G

Framework

Suppose, we have **independent** random variables X_1, \dots, X_n . We want to study the random variable:

$$f(X_1, \dots, X_n)$$

Some examples:

1. $X = X_1 + \dots + X_n$ (our setting earlier)
2. In **balls into bins**, X_i indicates where ball i is allocated, and $f(X_1, \dots, X_m)$ is the number of empty bins
3. In a **randomly generated graph**, X_i indicates if the i -th edge is present and $f(X_1, \dots, X_m)$ represents the number of connected components of G

In all those cases (and more) we can easily prove concentration of $f(X_1, \dots, X_n)$ around its mean by the so-called **Method of Bounded Differences**.

Method of Bounded Differences

A function f is called Lipschitz with parameters $\mathbf{c} = (c_1, \dots, c_n)$ if for all $i = 1, 2, \dots, n$,

$$|f(x_1, x_2, \dots, x_{i-1}, \mathbf{x}_i, x_{i+1}, \dots, x_n) - f(x_1, x_2, \dots, x_{i-1}, \tilde{\mathbf{x}}_i, x_{i+1}, \dots, x_n)| \leq c_i,$$

where x_i and \tilde{x}_i are in the domain of the i -th coordinate.

Method of Bounded Differences

A function f is called **Lipschitz with parameters** $\mathbf{c} = (c_1, \dots, c_n)$ if for all $i = 1, 2, \dots, n$,

$$|f(x_1, x_2, \dots, x_{i-1}, \mathbf{x}_i, x_{i+1}, \dots, x_n) - f(x_1, x_2, \dots, x_{i-1}, \tilde{\mathbf{x}}_i, x_{i+1}, \dots, x_n)| \leq c_i,$$

where x_i and \tilde{x}_i are in the domain of the i -th coordinate.

— McDiarmid's inequality —

Let X_1, \dots, X_n be **independent** random variables. Let f be **Lipschitz** with parameters $\mathbf{c} = (c_1, \dots, c_n)$. Let $X = f(X_1, \dots, X_n)$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

Method of Bounded Differences

A function f is called **Lipschitz with parameters** $\mathbf{c} = (c_1, \dots, c_n)$ if for all $i = 1, 2, \dots, n$,

$$|f(x_1, x_2, \dots, x_{i-1}, \mathbf{x}_i, x_{i+1}, \dots, x_n) - f(x_1, x_2, \dots, x_{i-1}, \tilde{\mathbf{x}}_i, x_{i+1}, \dots, x_n)| \leq c_i,$$

where x_i and \tilde{x}_i are in the domain of the i -th coordinate.

— McDiarmid's inequality —

Let X_1, \dots, X_n be **independent** random variables. Let f be **Lipschitz** with parameters $\mathbf{c} = (c_1, \dots, c_n)$. Let $X = f(X_1, \dots, X_n)$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

- Notice the similarity with Hoeffding's inequality! [[Exercise 2/3.14](#)]

Method of Bounded Differences

A function f is called **Lipschitz with parameters** $\mathbf{c} = (c_1, \dots, c_n)$ if for all $i = 1, 2, \dots, n$,

$$|f(x_1, x_2, \dots, x_{i-1}, \mathbf{x}_i, x_{i+1}, \dots, x_n) - f(x_1, x_2, \dots, x_{i-1}, \tilde{\mathbf{x}}_i, x_{i+1}, \dots, x_n)| \leq c_i,$$

where x_i and \tilde{x}_i are in the domain of the i -th coordinate.

— McDiarmid's inequality —

Let X_1, \dots, X_n be **independent** random variables. Let f be **Lipschitz** with parameters $\mathbf{c} = (c_1, \dots, c_n)$. Let $X = f(X_1, \dots, X_n)$. Then for any $t > 0$,

$$\mathbf{P}[X \geq \mu + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$\mathbf{P}[X \leq \mu - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

- Notice the similarity with Hoeffding's inequality! [[Exercise 2/3.14](#)]
- The proof is omitted here (it requires the concept of **martingales**).

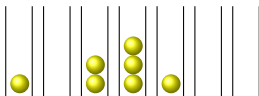
Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

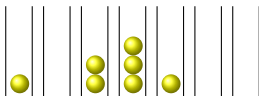
Appendix: More on Moment Generating Functions (non-examinable)

Application 3: Balls into Bins (again...)



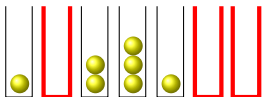
- Consider again m balls assigned uniformly at random into n bins.
- Enumerate the balls from 1 to m . Ball i is assigned to a random bin X_i

Application 3: Balls into Bins (again...)



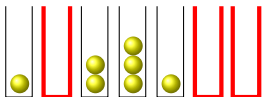
- Consider again m balls assigned uniformly at random into n bins.
- Enumerate the balls from 1 to m . Ball i is assigned to a random bin X_i
- Let Z be the number of empty bins (after assigning the m balls)
- $Z = Z(X_1, \dots, X_m)$ and Z is Lipschitz with $\mathbf{c} = (1, \dots, 1)$

Application 3: Balls into Bins (again...)



- Consider again m balls assigned uniformly at random into n bins.
- Enumerate the balls from 1 to m . Ball i is assigned to a random bin X_i
- Let Z be the number of empty bins (after assigning the m balls)
- $Z = Z(X_1, \dots, X_m)$ and Z is Lipschitz with $\mathbf{c} = (1, \dots, 1)$
(If we move one ball to another bin, number of empty bins changes by ≤ 1 .)

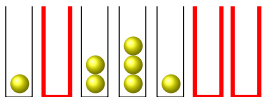
Application 3: Balls into Bins (again...)



- Consider again m balls assigned uniformly at random into n bins.
- Enumerate the balls from 1 to m . Ball i is assigned to a random bin X_i
- Let Z be the number of empty bins (after assigning the m balls)
- $Z = Z(X_1, \dots, X_m)$ and Z is Lipschitz with $\mathbf{c} = (1, \dots, 1)$
(If we move one ball to another bin, number of empty bins changes by ≤ 1 .)
- By McDiarmid's inequality, for any $t \geq 0$,

$$\mathbf{P}[|Z - \mathbf{E}[Z]| > t] \leq 2 \cdot e^{-2t^2/m}.$$

Application 3: Balls into Bins (again...)

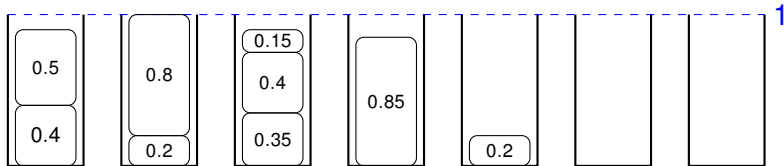


- Consider again m balls assigned uniformly at random into n bins.
- Enumerate the balls from 1 to m . Ball i is assigned to a random bin X_i
- Let Z be the number of empty bins (after assigning the m balls)
- $Z = Z(X_1, \dots, X_m)$ and Z is Lipschitz with $\mathbf{c} = (1, \dots, 1)$
(If we move one ball to another bin, number of empty bins changes by ≤ 1 .)
- By McDiarmid's inequality, for any $t \geq 0$,

$$\mathbf{P}[|Z - \mathbf{E}[Z]| > t] \leq 2 \cdot e^{-2t^2/m}.$$

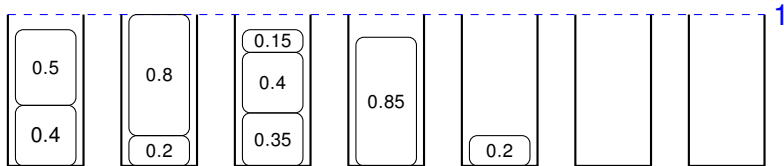
This is a decent bound, but for some values of m it is far from tight and stronger bounds are possible through a refined analysis.

Application 4: Bin Packing



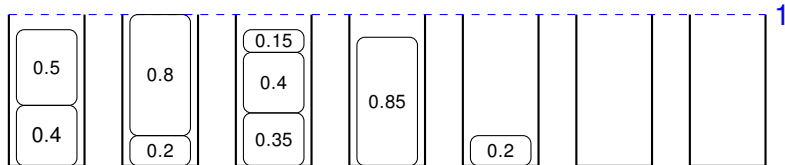
- We are given n items of sizes in the unit interval $[0, 1]$

Application 4: Bin Packing



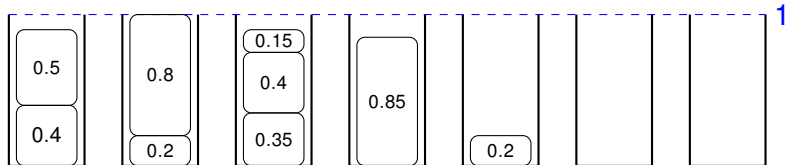
- We are given n items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the **fewest number of unit-capacity bins**

Application 4: Bin Packing



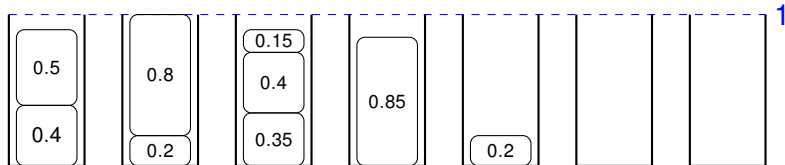
- We are given n items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the **fewest number of unit-capacity bins**
- Suppose the item sizes X_i are **independent random variables** in $[0, 1]$

Application 4: Bin Packing



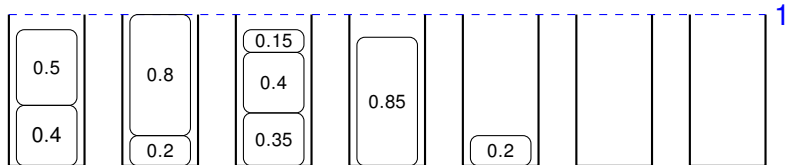
- We are given n items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the **fewest number of unit-capacity bins**
- Suppose the item sizes X_i are **independent random variables** in $[0, 1]$
- Let $B = B(X_1, \dots, X_n)$ be the **optimal number of bins**

Application 4: Bin Packing



- We are given n items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the **fewest number of unit-capacity bins**
- Suppose the item sizes X_i are **independent random variables** in $[0, 1]$
- Let $B = B(X_1, \dots, X_n)$ be the **optimal number of bins**
- The Lipschitz conditions holds with $\mathbf{c} = (1, \dots, 1)$. **Why?**

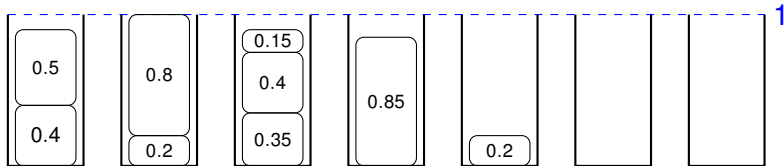
Application 4: Bin Packing



- We are given n items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the **fewest number of unit-capacity bins**
- Suppose the item sizes X_i are **independent random variables** in $[0, 1]$
- Let $B = B(X_1, \dots, X_n)$ be the **optimal number of bins**
- The Lipschitz conditions holds with $\mathbf{c} = (1, \dots, 1)$. **Why?**
- Therefore

$$\mathbf{P}[|B - \mathbf{E}[B]| \geq t] \leq 2 \cdot e^{-2t^2/n}.$$

Application 4: Bin Packing



- We are given n items of sizes in the unit interval $[0, 1]$
- We want to pack those items into the **fewest number of unit-capacity bins**
- Suppose the item sizes X_i are **independent random variables** in $[0, 1]$
- Let $B = B(X_1, \dots, X_n)$ be the **optimal number of bins**
- The Lipschitz conditions holds with $\mathbf{c} = (1, \dots, 1)$. **Why?**
- Therefore

$$\mathbf{P}[|B - \mathbf{E}[B]| \geq t] \leq 2 \cdot e^{-2t^2/n}.$$

This is a typical example where proving concentration is much easier than calculating (or estimating) the expectation!

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: More on Moment Generating Functions (non-examinable)

Moment Generating Functions (non-examinable)

— Moment-Generating Function —

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Moment Generating Functions (non-examinable)

Moment-Generating Function

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Using power series of e and differentiating shows that $M_X(t)$ encapsulates all moments of X .

Moment Generating Functions (non-examinable)

Moment-Generating Function

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Using power series of e and differentiating shows that $M_X(t)$ encapsulates all moments of X .

Lemma

1. If X and Y are two r.v.'s with $M_X(t) = M_Y(t)$ for all $t \in (-\delta, +\delta)$ for some $\delta > 0$, then the distributions X and Y are identical.
2. If X and Y are **independent** random variables, then

$$M_{X+Y}(t) = M_X(t) \cdot M_Y(t).$$

Moment Generating Functions (non-examinable)

Moment-Generating Function

The **moment-generating** function of a random variable X is

$$M_X(t) = \mathbf{E} \left[e^{tX} \right], \quad \text{where } t \in \mathbb{R}.$$

Using power series of e and differentiating shows that $M_X(t)$ encapsulates all moments of X .

Lemma

1. If X and Y are two r.v.'s with $M_X(t) = M_Y(t)$ for all $t \in (-\delta, +\delta)$ for some $\delta > 0$, then the distributions X and Y are identical.
2. If X and Y are **independent** random variables, then

$$M_{X+Y}(t) = M_X(t) \cdot M_Y(t).$$

Proof of 2:

$$M_{X+Y}(t) = \mathbf{E} \left[e^{t(X+Y)} \right] = \mathbf{E} \left[e^{tX} \cdot e^{tY} \right] \stackrel{(!)}{=} \mathbf{E} \left[e^{tX} \right] \cdot \mathbf{E} \left[e^{tY} \right] = M_X(t) M_Y(t) \quad \square$$

Randomised Algorithms

Lecture 4: Markov Chains and Mixing Times

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

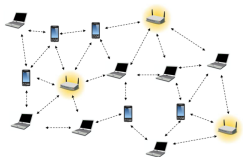
Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

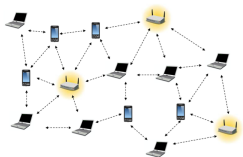
Appendix: Remarks on Mixing Time (non-examin.)

Applications of Markov Chains in Computer Science

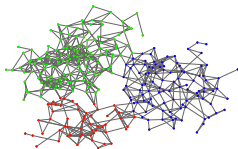


Broadcasting

Applications of Markov Chains in Computer Science

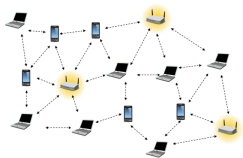


Broadcasting

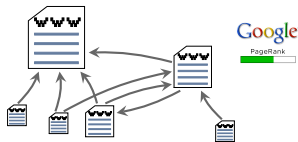


Clustering

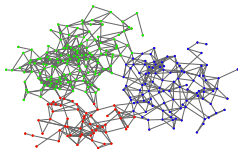
Applications of Markov Chains in Computer Science



Broadcasting

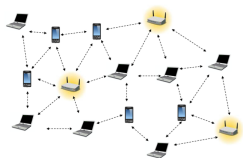


Ranking Websites

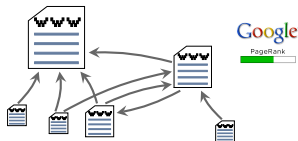


Clustering

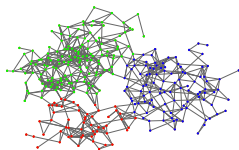
Applications of Markov Chains in Computer Science



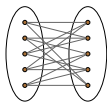
Broadcasting



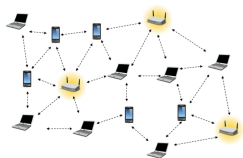
Ranking Websites



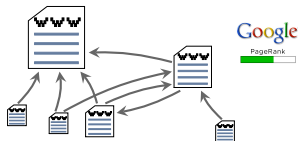
Clustering



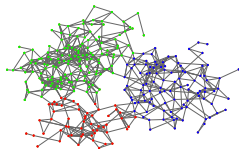
Applications of Markov Chains in Computer Science



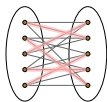
Broadcasting



Ranking Websites



Clustering

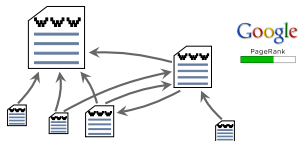


$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

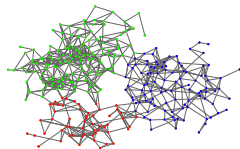
Applications of Markov Chains in Computer Science



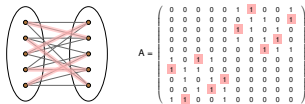
Broadcasting



Ranking Websites



Clustering

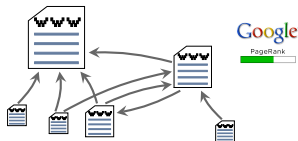


Sampling and Optimisation

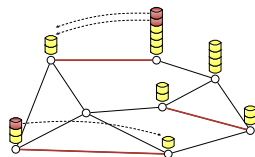
Applications of Markov Chains in Computer Science



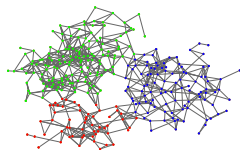
Broadcasting



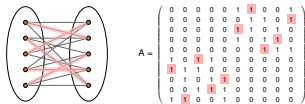
Ranking Websites



Load Balancing



Clustering

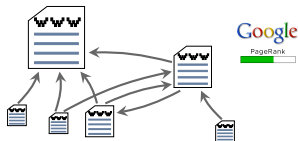


Sampling and Optimisation

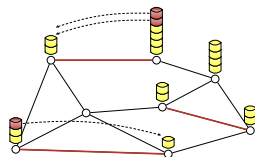
Applications of Markov Chains in Computer Science



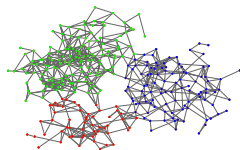
Broadcasting



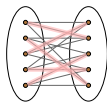
Ranking Websites



Load Balancing

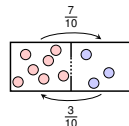


Clustering



Sampling and Optimisation

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Particle Processes

Markov Chains

Markov Chain (Discrete Time and State, Time Homogeneous)

We say that $(X_t)_{t=0}^{\infty}$ is a Markov Chain on State Space Ω with Initial Distribution μ and Transition Matrix P if:

Markov Chains

Markov Chain (Discrete Time and State, Time Homogeneous)

We say that $(X_t)_{t=0}^{\infty}$ is a Markov Chain on State Space Ω with Initial Distribution μ and Transition Matrix P if:

1. For any $x \in \Omega$, $\mathbf{P}[X_0 = x] = \mu(x)$.

Markov Chains

Markov Chain (Discrete Time and State, Time Homogeneous)

We say that $(X_t)_{t=0}^{\infty}$ is a **Markov Chain** on **State Space** Ω with **Initial Distribution** μ and **Transition Matrix** P if:

1. For any $x \in \Omega$, $\mathbf{P}[X_0 = x] = \mu(x)$.
2. The **Markov Property** holds: for all $t \geq 0$ and any $x_0, \dots, x_{t+1} \in \Omega$,

$$\begin{aligned}\mathbf{P}\left[X_{t+1} = x_{t+1} \mid X_t = x_t, \dots, X_0 = x_0\right] &= \mathbf{P}\left[X_{t+1} = x_{t+1} \mid X_t = x_t\right] \\ &:= P(x_t, x_{t+1}).\end{aligned}$$

Markov Chains

Markov Chain (Discrete Time and State, Time Homogeneous)

We say that $(X_t)_{t=0}^{\infty}$ is a **Markov Chain** on **State Space** Ω with **Initial Distribution** μ and **Transition Matrix** P if:

1. For any $x \in \Omega$, $\mathbf{P}[X_0 = x] = \mu(x)$.
2. The **Markov Property** holds: for all $t \geq 0$ and any $x_0, \dots, x_{t+1} \in \Omega$,

$$\begin{aligned}\mathbf{P}[X_{t+1} = x_{t+1} \mid X_t = x_t, \dots, X_0 = x_0] &= \mathbf{P}[X_{t+1} = x_{t+1} \mid X_t = x_t] \\ &:= P(x_t, x_{t+1}).\end{aligned}$$

From the definition one can deduce that (check!)

Markov Chains

Markov Chain (Discrete Time and State, Time Homogeneous)

We say that $(X_t)_{t=0}^{\infty}$ is a **Markov Chain** on **State Space** Ω with **Initial Distribution** μ and **Transition Matrix** P if:

1. For any $x \in \Omega$, $\mathbf{P}[X_0 = x] = \mu(x)$.
2. The **Markov Property** holds: for all $t \geq 0$ and any $x_0, \dots, x_{t+1} \in \Omega$,

$$\begin{aligned}\mathbf{P}[X_{t+1} = x_{t+1} \mid X_t = x_t, \dots, X_0 = x_0] &= \mathbf{P}[X_{t+1} = x_{t+1} \mid X_t = x_t] \\ &:= P(x_t, x_{t+1}).\end{aligned}$$

From the definition one can deduce that (check!)

- For all $t, x_0, x_1, \dots, x_t \in \Omega$,

$$\begin{aligned}\mathbf{P}[X_t = x_t, X_{t-1} = x_{t-1}, \dots, X_0 = x_0] \\ = \mu(x_0) \cdot P(x_0, x_1) \cdot \dots \cdot P(x_{t-2}, x_{t-1}) \cdot P(x_{t-1}, x_t).\end{aligned}$$

Markov Chains

Markov Chain (Discrete Time and State, Time Homogeneous)

We say that $(X_t)_{t=0}^{\infty}$ is a **Markov Chain** on **State Space** Ω with **Initial Distribution** μ and **Transition Matrix** P if:

1. For any $x \in \Omega$, $\mathbf{P}[X_0 = x] = \mu(x)$.
2. The **Markov Property** holds: for all $t \geq 0$ and any $x_0, \dots, x_{t+1} \in \Omega$,

$$\begin{aligned}\mathbf{P}[X_{t+1} = x_{t+1} \mid X_t = x_t, \dots, X_0 = x_0] &= \mathbf{P}[X_{t+1} = x_{t+1} \mid X_t = x_t] \\ &:= P(x_t, x_{t+1}).\end{aligned}$$

From the definition one can deduce that (check!)

- For all $t, x_0, x_1, \dots, x_t \in \Omega$,

$$\begin{aligned}\mathbf{P}[X_t = x_t, X_{t-1} = x_{t-1}, \dots, X_0 = x_0] \\ = \mu(x_0) \cdot P(x_0, x_1) \cdot \dots \cdot P(x_{t-2}, x_{t-1}) \cdot P(x_{t-1}, x_t).\end{aligned}$$

- For all $0 \leq t_1 < t_2, x \in \Omega$,

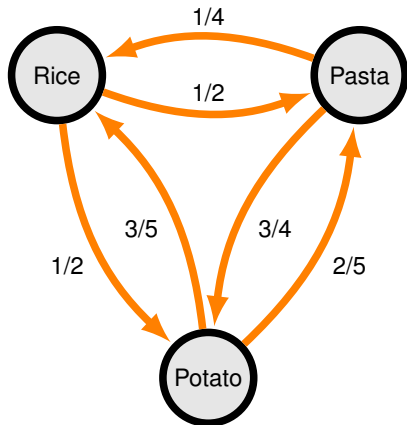
$$\mathbf{P}[X_{t_2} = x] = \sum_{y \in \Omega} \mathbf{P}[X_{t_2} = x \mid X_{t_1} = y] \cdot \mathbf{P}[X_{t_1} = y].$$

What does a Markov Chain Look Like?

Example: the carbohydrate served with lunch in the college cafeteria.

This has transition matrix:

$$P = \begin{array}{c} \begin{array}{ccc} \text{Rice} & \text{Pasta} & \text{Potato} \end{array} \\ \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{bmatrix} \end{array} \begin{array}{l} \text{Rice} \\ \text{Pasta} \\ \text{Potato} \end{array}$$



Transition Matrices and Distributions

The Transition Matrix P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1, 1) & \dots & P(1, n) \\ \vdots & \ddots & \vdots \\ P(n, 1) & \dots & P(n, n) \end{pmatrix}.$$

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1, 1) & \dots & P(1, n) \\ \vdots & \ddots & \vdots \\ P(n, 1) & \dots & P(n, n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \dots, \rho^t(n))$: **state vector** at time t (**row vector**).

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1,1) & \dots & P(1,n) \\ \vdots & \ddots & \vdots \\ P(n,1) & \dots & P(n,n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \dots, \rho^t(n))$: **state vector** at time t (**row vector**).
- Multiplying ρ^t by P corresponds to advancing the chain one step:

$$\rho^t(y) = \sum_{x \in \Omega} \rho^{t-1}(x) \cdot P(x, y) \quad \text{and thus} \quad \rho^t = \rho^{t-1} \cdot P.$$

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1,1) & \dots & P(1,n) \\ \vdots & \ddots & \vdots \\ P(n,1) & \dots & P(n,n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \dots, \rho^t(n))$: **state vector** at time t (**row vector**).
- Multiplying ρ^t by P corresponds to advancing the chain one step:

$$\rho^t(y) = \sum_{x \in \Omega} \rho^{t-1}(x) \cdot P(x, y) \quad \text{and thus} \quad \rho^t = \rho^{t-1} \cdot P.$$

- The **Markov Property** and line above imply that for any $t \geq 0$

$$\rho^t = \rho \cdot P^{t-1} \quad \text{and thus} \quad P^t(x, y) = \mathbf{P}[X_t = y \mid X_0 = x].$$

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1, 1) & \dots & P(1, n) \\ \vdots & \ddots & \vdots \\ P(n, 1) & \dots & P(n, n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \dots, \rho^t(n))$: **state vector** at time t (**row vector**).
- Multiplying ρ^t by P corresponds to advancing the chain one step:

$$\rho^t(y) = \sum_{x \in \Omega} \rho^{t-1}(x) \cdot P(x, y) \quad \text{and thus} \quad \rho^t = \rho^{t-1} \cdot P.$$

- The **Markov Property** and line above imply that for any $t \geq 0$

$$\rho^t = \rho \cdot P^{t-1} \quad \text{and thus} \quad P^t(x, y) = \mathbf{P}[X_t = y \mid X_0 = x].$$

Thus $\rho^t(x) = (\mu P^t)(x)$ and so $\rho^t = \mu P^t = (\mu P^t(1), \mu P^t(2), \dots, \mu P^t(n))$.

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1,1) & \dots & P(1,n) \\ \vdots & \ddots & \vdots \\ P(n,1) & \dots & P(n,n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \dots, \rho^t(n))$: **state vector** at time t (**row vector**).
- Multiplying ρ^t by P corresponds to advancing the chain one step:

$$\rho^t(y) = \sum_{x \in \Omega} \rho^{t-1}(x) \cdot P(x, y) \quad \text{and thus} \quad \rho^t = \rho^{t-1} \cdot P.$$

- The **Markov Property** and line above imply that for any $t \geq 0$

$$\rho^t = \rho \cdot P^{t-1} \quad \text{and thus} \quad P^t(x, y) = \mathbf{P}[X_t = y \mid X_0 = x].$$

Thus $\rho^t(x) = (\mu P^t)(x)$ and so $\rho^t = \mu P^t = (\mu P^t(1), \mu P^t(2), \dots, \mu P^t(n))$.

- Everything boils down to **deterministic** vector/matrix computations

Transition Matrices and Distributions

The **Transition Matrix** P of a Markov chain (μ, P) on $\Omega = \{1, \dots, n\}$ is given by

$$P = \begin{pmatrix} P(1,1) & \dots & P(1,n) \\ \vdots & \ddots & \vdots \\ P(n,1) & \dots & P(n,n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \dots, \rho^t(n))$: **state vector** at time t (**row vector**).
- Multiplying ρ^t by P corresponds to advancing the chain one step:

$$\rho^t(y) = \sum_{x \in \Omega} \rho^{t-1}(x) \cdot P(x, y) \quad \text{and thus} \quad \rho^t = \rho^{t-1} \cdot P.$$

- The **Markov Property** and line above imply that for any $t \geq 0$

$$\rho^t = \rho \cdot P^{t-1} \quad \text{and thus} \quad P^t(x, y) = \mathbf{P}[X_t = y \mid X_0 = x].$$

Thus $\rho^t(x) = (\mu P^t)(x)$ and so $\rho^t = \mu P^t = (\mu P^t(1), \mu P^t(2), \dots, \mu P^t(n))$.

- Everything boils down to **deterministic** vector/matrix computations
 \Rightarrow can replace ρ by any (load) vector and view P as a **balancing matrix**!

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Example - College Carbs Stopping times:

✓ “We had **rice** yesterday”

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Example - College Carbs Stopping times:

✓ “We had **rice** yesterday” $\leadsto \tau := \min \{t \geq 1 : X_{t-1} = \text{“rice”}\}$

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Example - College Carbs Stopping times:

- ✓ “We had **rice** yesterday” $\leadsto \tau := \min \{t \geq 1 : X_{t-1} = \text{“rice”}\}$
- ✗ “We are having **pasta** next Thursday”

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Example - College Carbs Stopping times:

- ✓ “We had **rice** yesterday” $\leadsto \tau := \min \{t \geq 1 : X_{t-1} = \text{“rice”}\}$
- ✗ “We are having **pasta** next Thursday”

For two states $x, y \in \Omega$ we call $h(x, y)$ the **hitting time** of y from x :

$$h(x, y) := \mathbf{E}_x[\tau_y] = \mathbf{E}[\tau_y \mid X_0 = x] \quad \text{where } \tau_y = \min\{t \geq 1 : X_t = y\}.$$

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Example - College Carbs Stopping times:

✓ “We had **rice** yesterday” $\leadsto \tau := \min \{t \geq 1 : X_{t-1} = \text{“rice”}\}$

✗ “We are having **pasta** next Thursday”

For two states $x, y \in \Omega$ we call $h(x, y)$ the **hitting time** of y from x :

$$h(x, y) := \mathbf{E}_x[\tau_y] = \mathbf{E}[\tau_y \mid X_0 = x] \quad \text{where } \tau_y = \min\{t \geq 1 : X_t = y\}.$$

Some distinguish between $\tau_y^+ = \min\{t \geq 1 : X_t = y\}$ and $\tau_y = \min\{t \geq 0 : X_t = y\}$

Stopping and Hitting Times

A non-negative integer random variable τ is a **stopping time** for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on X_0, \dots, X_s .

Example - College Carbs Stopping times:

✓ “We had **rice** yesterday” $\leadsto \tau := \min \{t \geq 1 : X_{t-1} = \text{“rice”}\}$

✗ “We are having **pasta** next Thursday”

For two states $x, y \in \Omega$ we call $h(x, y)$ the **hitting time** of y from x :

$$h(x, y) := \mathbf{E}_x[\tau_y] = \mathbf{E}[\tau_y \mid X_0 = x] \quad \text{where } \tau_y = \min\{t \geq 1 : X_t = y\}.$$

— A Useful Identity —

Hitting times are the solution to a **set of linear equations**:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in \Omega.$$

Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

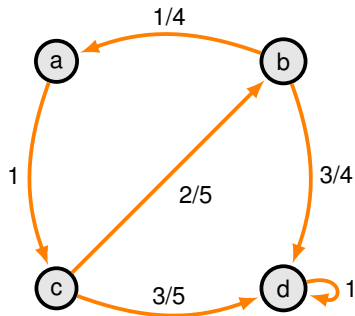
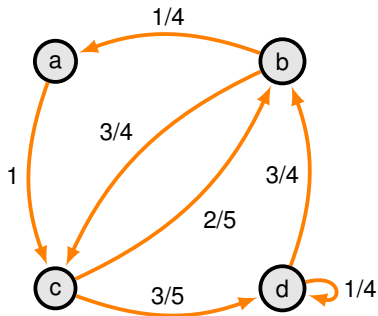
Appendix: Remarks on Mixing Time (non-examin.)

Irreducible Markov Chains

A Markov Chain is **irreducible** if for every pair of states $x, y \in \Omega$ there is an integer $k \geq 0$ such that $P^k(x, y) > 0$.

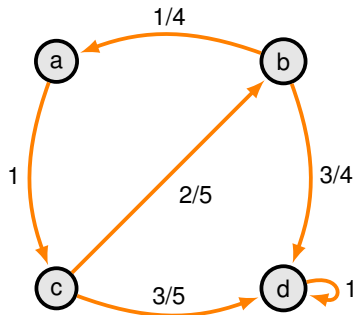
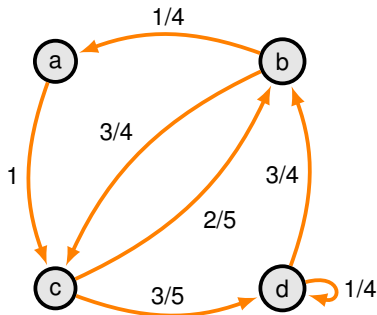
Irreducible Markov Chains

A Markov Chain is **irreducible** if for every pair of states $x, y \in \Omega$ there is an integer $k \geq 0$ such that $P^k(x, y) > 0$.



Irreducible Markov Chains

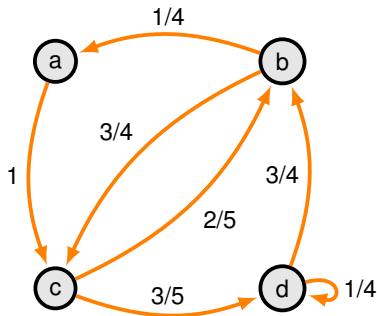
A Markov Chain is **irreducible** if for every pair of states $x, y \in \Omega$ there is an integer $k \geq 0$ such that $P^k(x, y) > 0$.



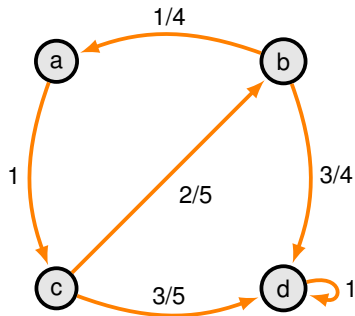
Question: Which of the two chains (if any) are irreducible?

Irreducible Markov Chains

A Markov Chain is **irreducible** if for every pair of states $x, y \in \Omega$ there is an integer $k \geq 0$ such that $P^k(x, y) > 0$.



✓ irreducible



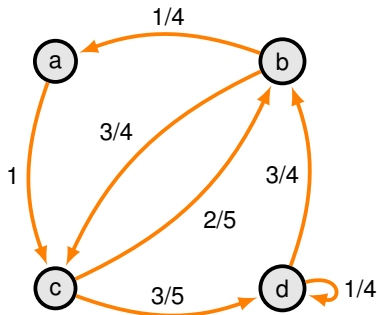
✗ not irreducible (thus reducible)



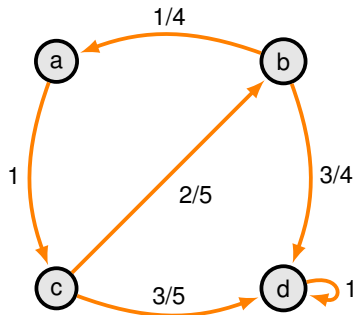
Question: Which of the two chains (if any) are irreducible?

Irreducible Markov Chains

A Markov Chain is **irreducible** if for every pair of states $x, y \in \Omega$ there is an integer $k \geq 0$ such that $P^k(x, y) > 0$.



✓ irreducible



✗ not irreducible (thus reducible)

Finite Hitting Time Theorem

For any states x and y of a **finite irreducible** Markov Chain $h(x, y) < \infty$.

Stationary Distribution

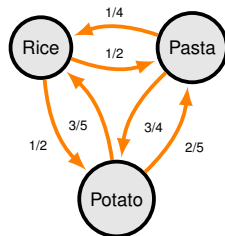
A probability distribution $\pi = (\pi(1), \dots, \pi(n))$ is the stationary distribution of a Markov Chain if $\pi P = \pi$ (π is a left eigenvector with eigenvalue 1)

Stationary Distribution

A probability distribution $\pi = (\pi(1), \dots, \pi(n))$ is the **stationary distribution** of a Markov Chain if $\pi P = \pi$ (π is a **left eigenvector** with eigenvalue 1)

College carbs example:

$$\begin{pmatrix} \frac{4}{13} & \frac{4}{13} & \frac{5}{13} \end{pmatrix}_{\pi} \cdot \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{pmatrix}_P = \begin{pmatrix} \frac{4}{13} & \frac{4}{13} & \frac{5}{13} \end{pmatrix}_{\pi}$$

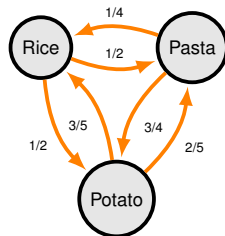


Stationary Distribution

A probability distribution $\pi = (\pi(1), \dots, \pi(n))$ is the **stationary distribution** of a Markov Chain if $\pi P = \pi$ (π is a **left eigenvector** with eigenvalue 1)

College carbs example:

$$\begin{pmatrix} \frac{4}{13} & \frac{4}{13} & \frac{5}{13} \end{pmatrix}_{\pi} \cdot \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{pmatrix}_P = \begin{pmatrix} \frac{4}{13} & \frac{4}{13} & \frac{5}{13} \end{pmatrix}_{\pi}$$



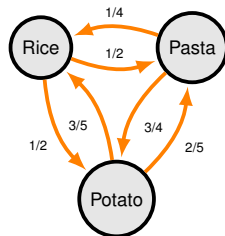
- A Markov Chain reaches **stationary distribution** if $\rho^t = \pi$ for some t .

Stationary Distribution

A probability distribution $\pi = (\pi(1), \dots, \pi(n))$ is the **stationary distribution** of a Markov Chain if $\pi P = \pi$ (π is a **left eigenvector** with eigenvalue 1)

College carbs example:

$$\begin{pmatrix} \frac{4}{13} & \frac{4}{13} & \frac{5}{13} \end{pmatrix}_{\pi} \cdot \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{pmatrix}_P = \begin{pmatrix} \frac{4}{13} & \frac{4}{13} & \frac{5}{13} \end{pmatrix}_{\pi}$$



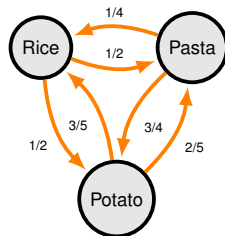
- A Markov Chain reaches **stationary distribution** if $\rho^t = \pi$ for some t .
- If reached, then it **persists**: If $\rho^t = \pi$ then $\rho^{t+k} = \pi$ for all $k \geq 0$.

Stationary Distribution

A probability distribution $\pi = (\pi(1), \dots, \pi(n))$ is the **stationary distribution** of a Markov Chain if $\pi P = \pi$ (π is a **left eigenvector** with eigenvalue 1)

College carbs example:

$$\underbrace{\left(\frac{4}{13}, \frac{4}{13}, \frac{5}{13} \right)}_{\pi} \cdot \underbrace{\begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{pmatrix}}_P = \underbrace{\left(\frac{4}{13}, \frac{4}{13}, \frac{5}{13} \right)}_{\pi}$$



- A Markov Chain reaches **stationary distribution** if $\rho^t = \pi$ for some t .
- If reached, then it **persists**: If $\rho^t = \pi$ then $\rho^{t+k} = \pi$ for all $k \geq 0$.

Existence and Uniqueness of a Positive Stationary Distribution

Let P be **finite, irreducible** M.C., then there **exists** a unique probability distribution π on Ω such that $\pi = \pi P$ and $\pi(x) = 1/h(x, x) > 0, \forall x \in \Omega$.

Periodicity

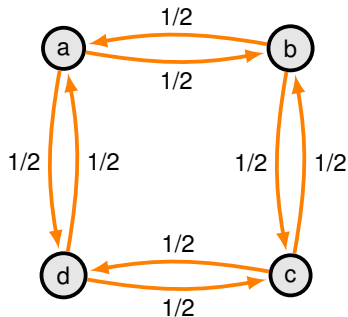
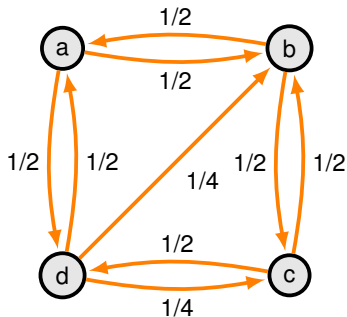
- A Markov Chain is **aperiodic** if for all $x \in \Omega$, $\gcd\{t \geq 1 : P^t(x, x) > 0\} = 1$.

Periodicity

- A Markov Chain is **aperiodic** if for all $x \in \Omega$, $\gcd\{t \geq 1 : P^t(x, x) > 0\} = 1$.
- Otherwise we say it is **periodic**.

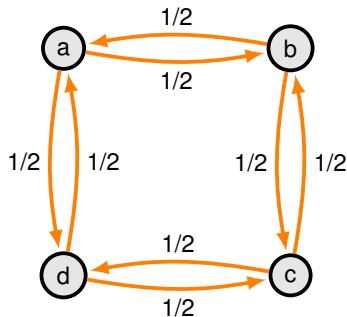
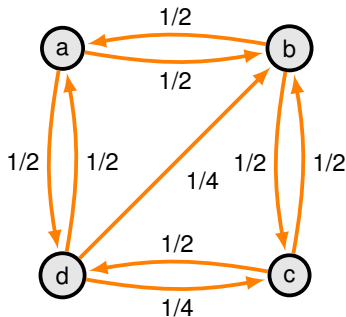
Periodicity

- A Markov Chain is **aperiodic** if for all $x \in \Omega$, $\gcd\{t \geq 1 : P^t(x, x) > 0\} = 1$.
- Otherwise we say it is **periodic**.



Periodicity

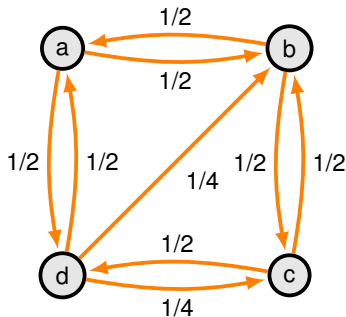
- A Markov Chain is **aperiodic** if for all $x \in \Omega$, $\gcd\{t \geq 1 : P^t(x, x) > 0\} = 1$.
- Otherwise we say it is **periodic**.



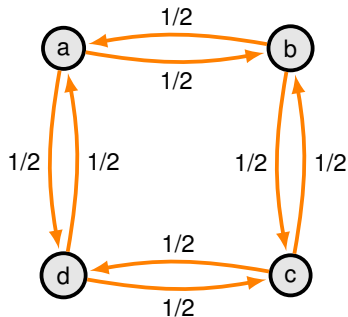
Question: Which of the two chains (if any) are aperiodic?

Periodicity

- A Markov Chain is **aperiodic** if for all $x \in \Omega$, $\gcd\{t \geq 1 : P^t(x, x) > 0\} = 1$.
- Otherwise we say it is **periodic**.



✓ Aperiodic



✗ Periodic



Question: Which of the two chains (if any) are aperiodic?

Convergence Theorem

Convergence Theorem

Let P be any finite, irreducible, aperiodic Markov Chain with stationary distribution π . Then for any $x, y \in \Omega$,

$$\lim_{t \rightarrow \infty} P^t(x, y) = \pi(y).$$

Convergence Theorem

Ergodic = Irreducible + Aperiodic

Convergence Theorem

Let P be any finite, irreducible, aperiodic Markov Chain with stationary distribution π . Then for any $x, y \in \Omega$,

$$\lim_{t \rightarrow \infty} P^t(x, y) = \pi(y).$$

Convergence Theorem

Ergodic = Irreducible + Aperiodic

Convergence Theorem

Let P be any finite, irreducible, aperiodic Markov Chain with stationary distribution π . Then for any $x, y \in \Omega$,

$$\lim_{t \rightarrow \infty} P^t(x, y) = \pi(y).$$

- mentioned before: For finite irreducible M.C.'s π exists, is unique and

$$\pi(y) = \frac{1}{h(y, y)} > 0.$$

Convergence Theorem

Ergodic = Irreducible + Aperiodic

Convergence Theorem

Let P be any finite, irreducible, aperiodic Markov Chain with stationary distribution π . Then for any $x, y \in \Omega$,

$$\lim_{t \rightarrow \infty} P^t(x, y) = \pi(y).$$

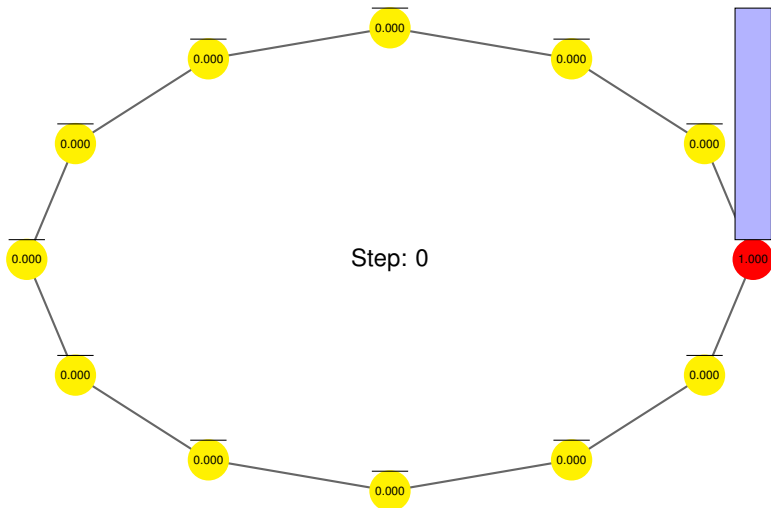
- mentioned before: For finite irreducible M.C.'s π exists, is unique and

$$\pi(y) = \frac{1}{h(y, y)} > 0.$$

- We will prove a simpler version of the Convergence Theorem after introducing Spectral Graph Theory.

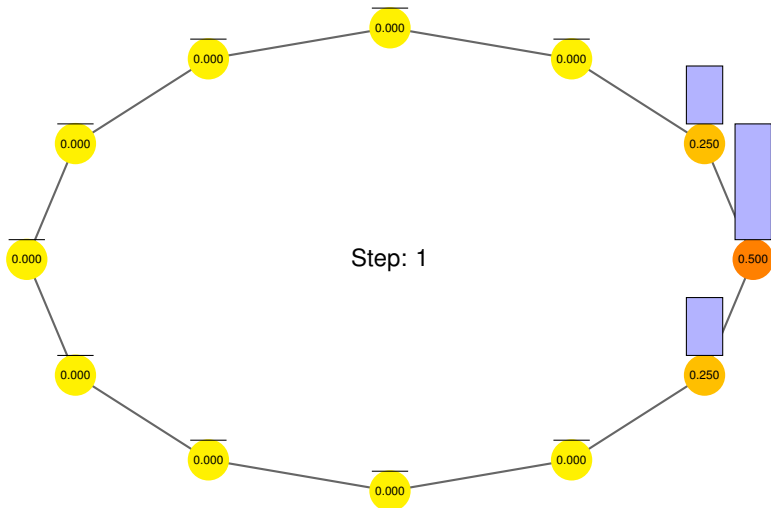
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



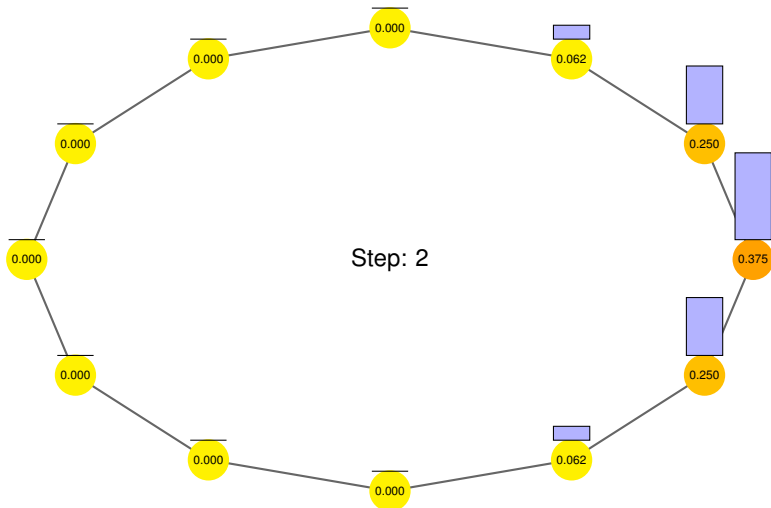
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



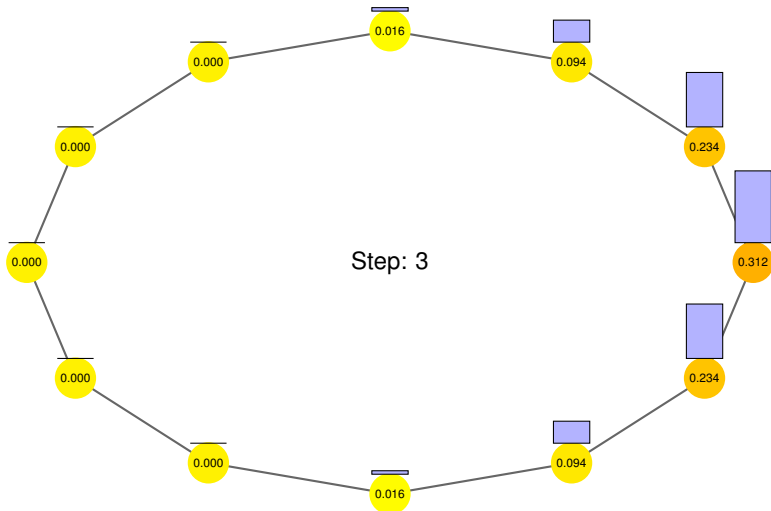
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



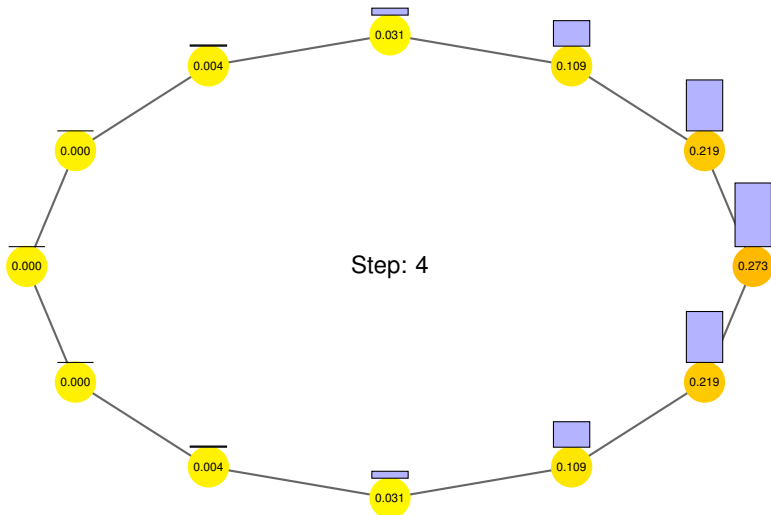
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



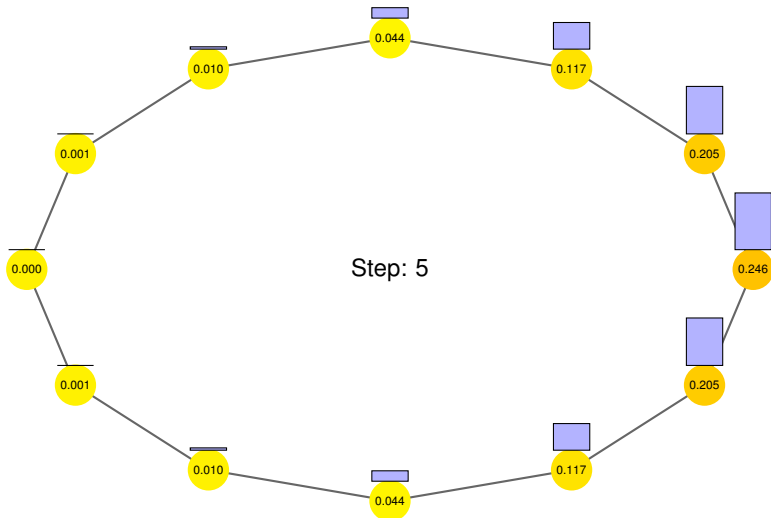
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



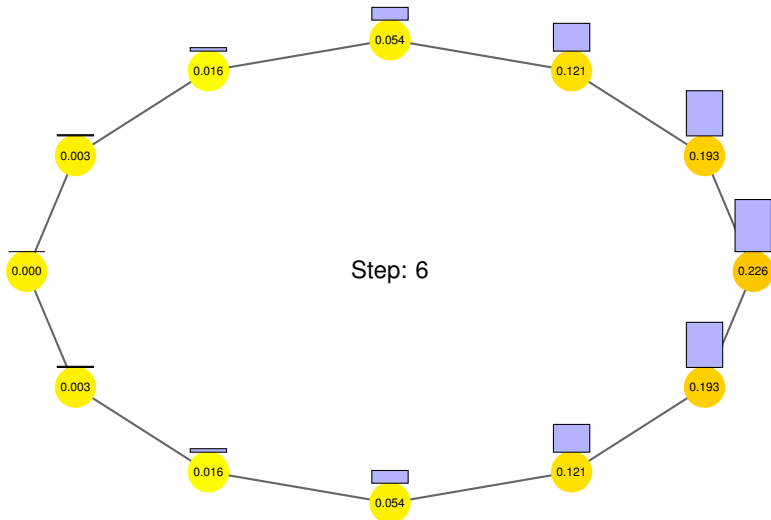
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



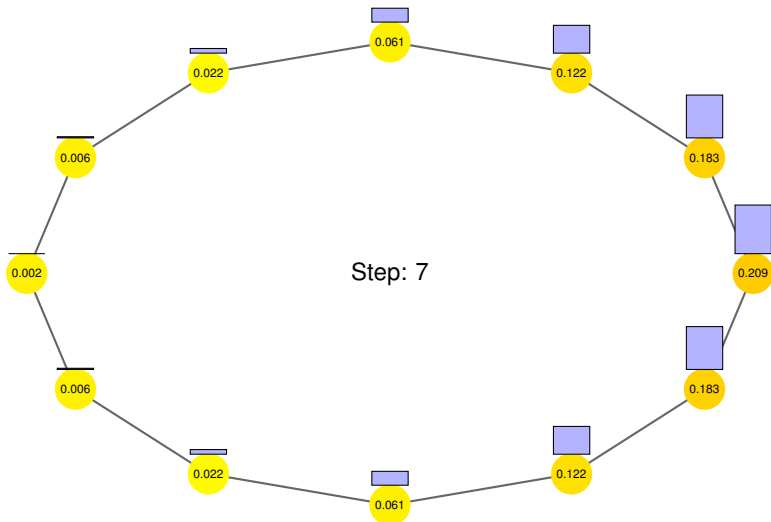
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



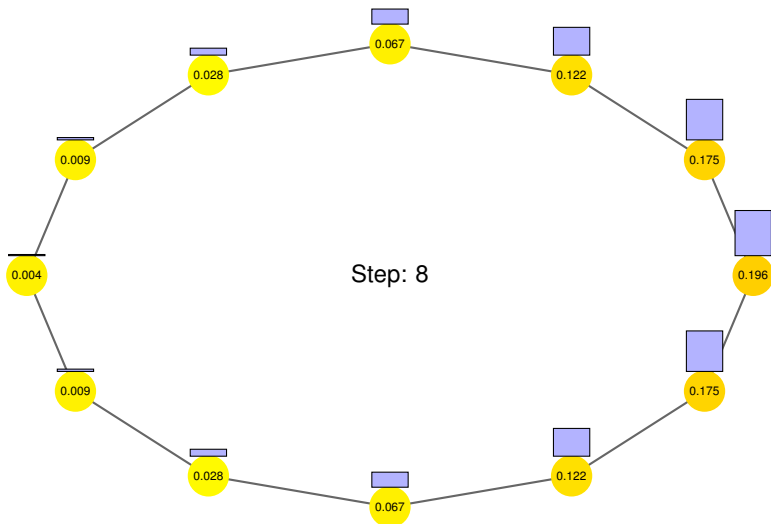
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



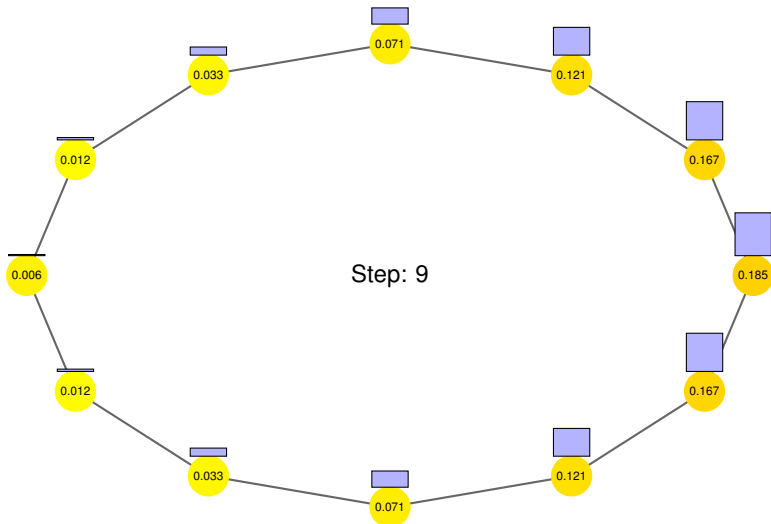
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



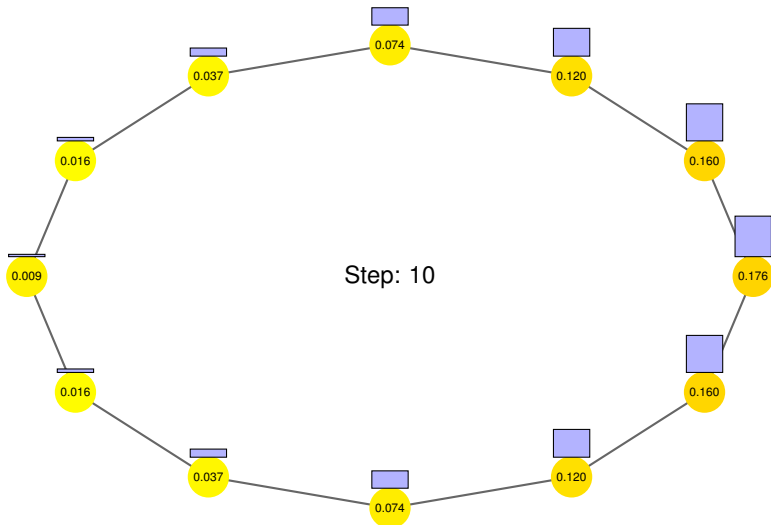
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



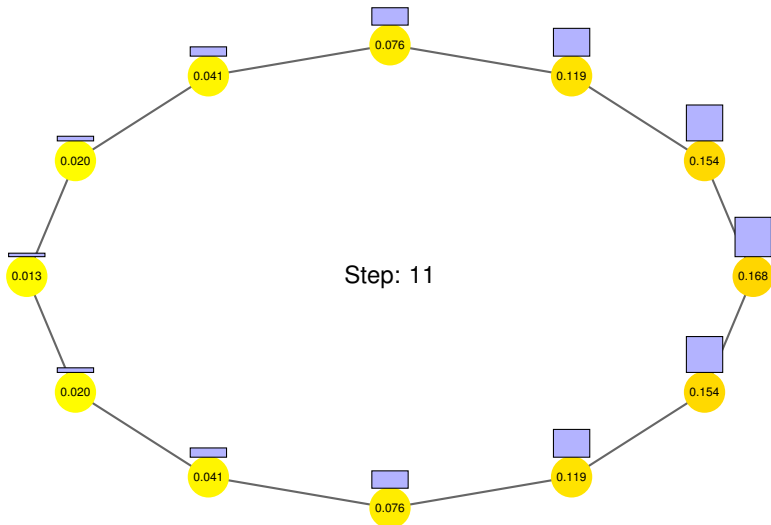
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



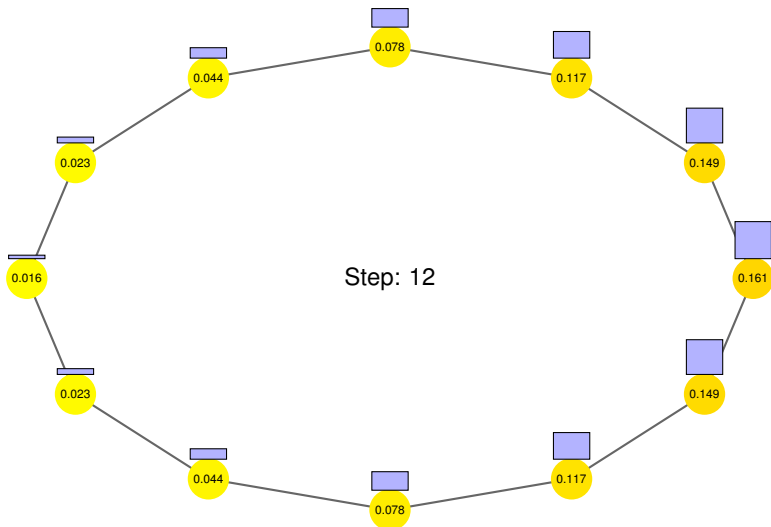
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



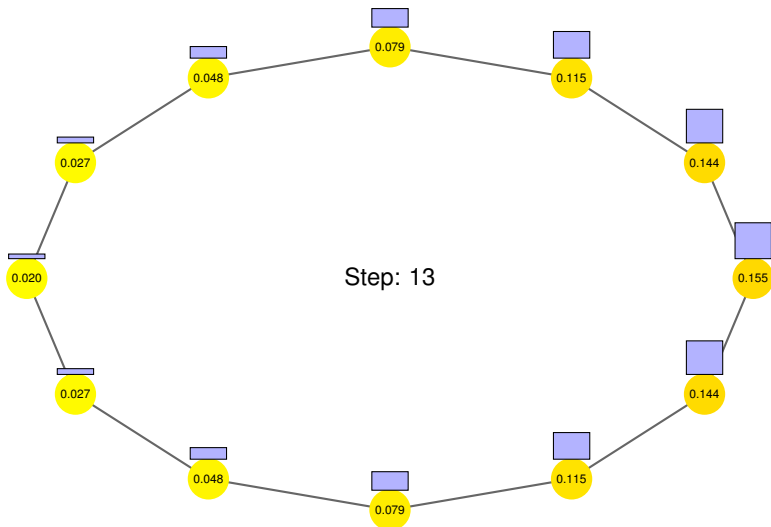
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



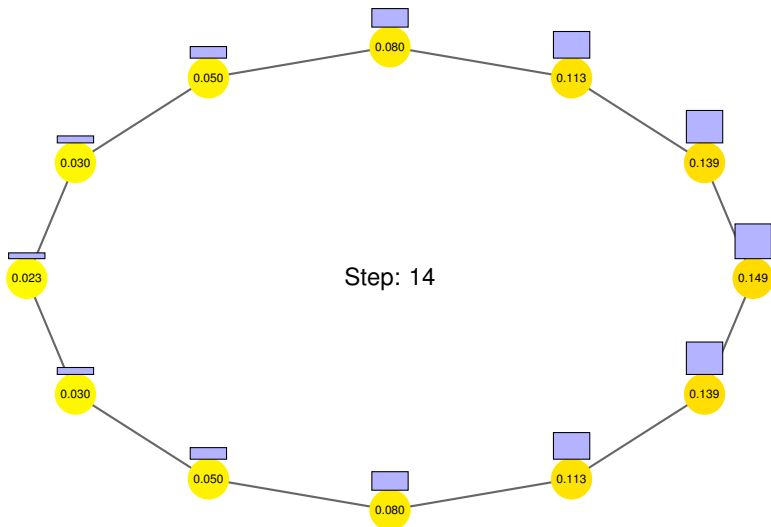
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



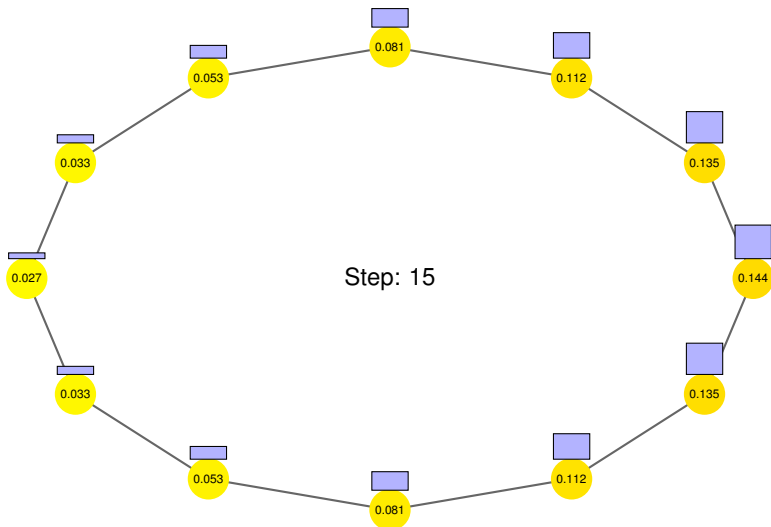
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



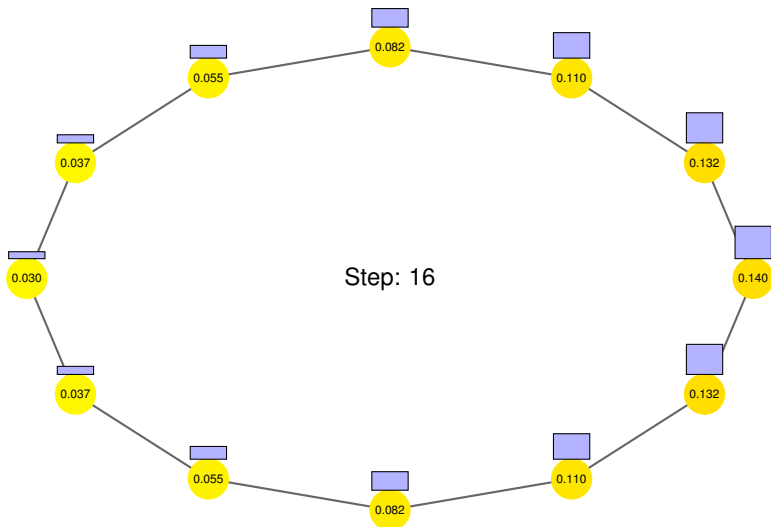
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



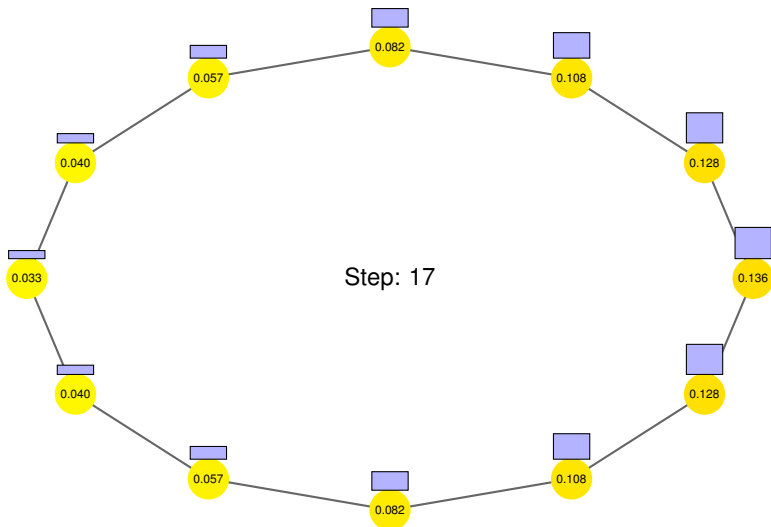
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



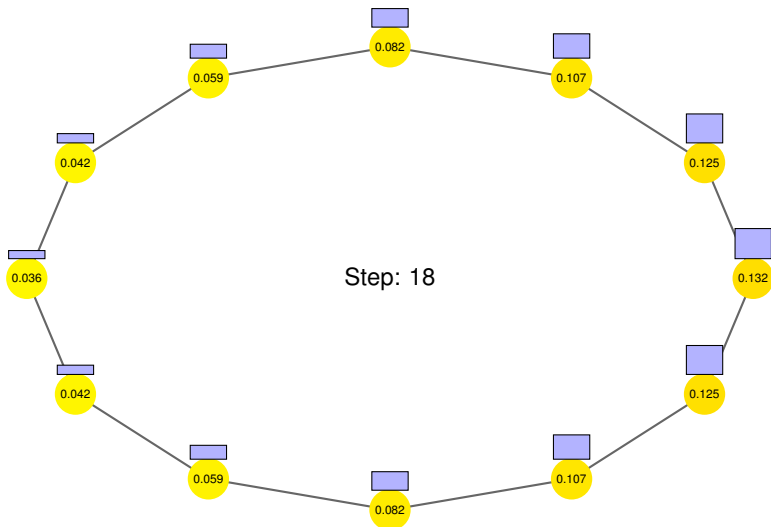
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



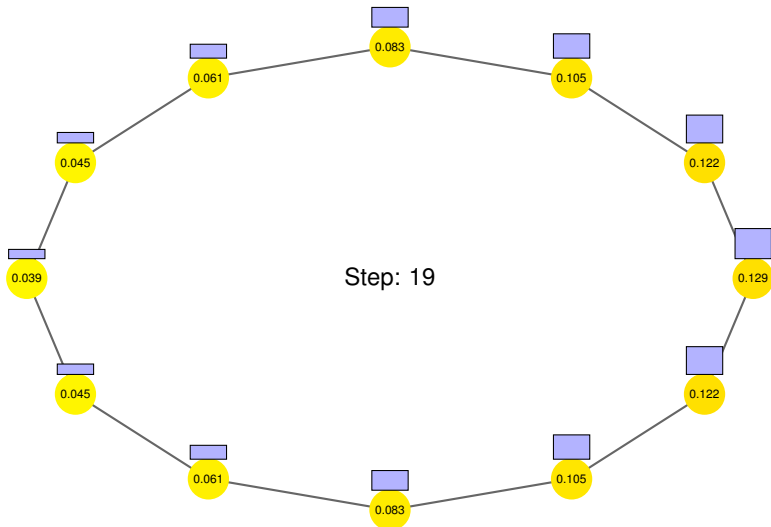
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



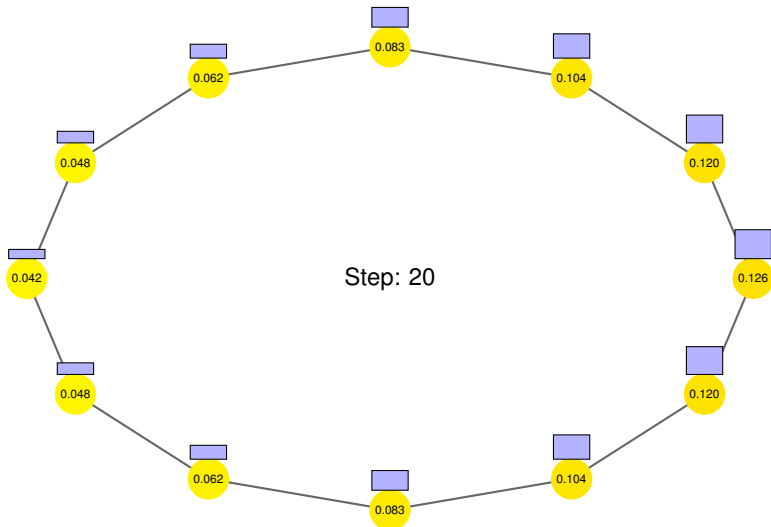
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



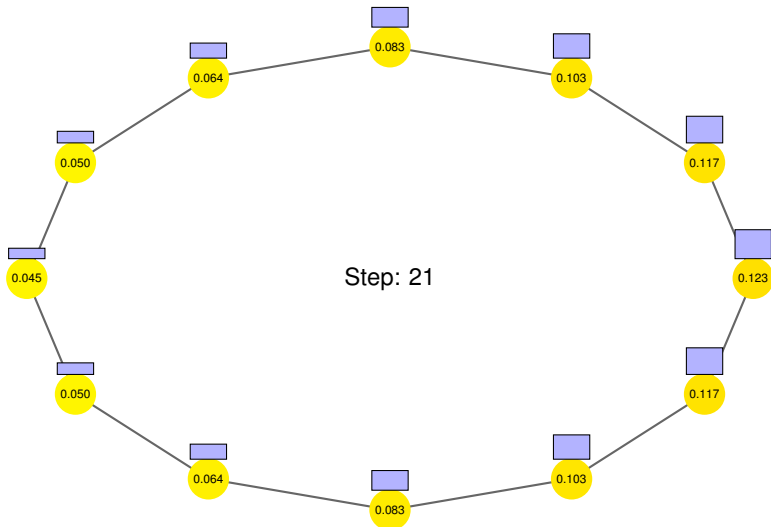
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



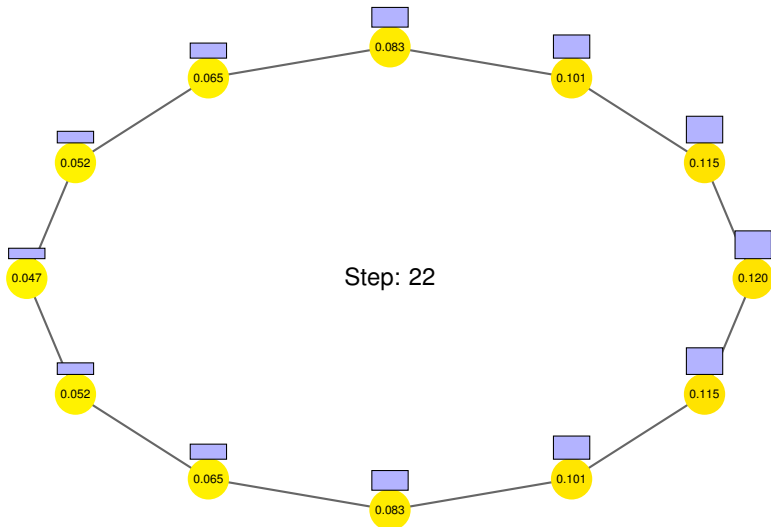
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



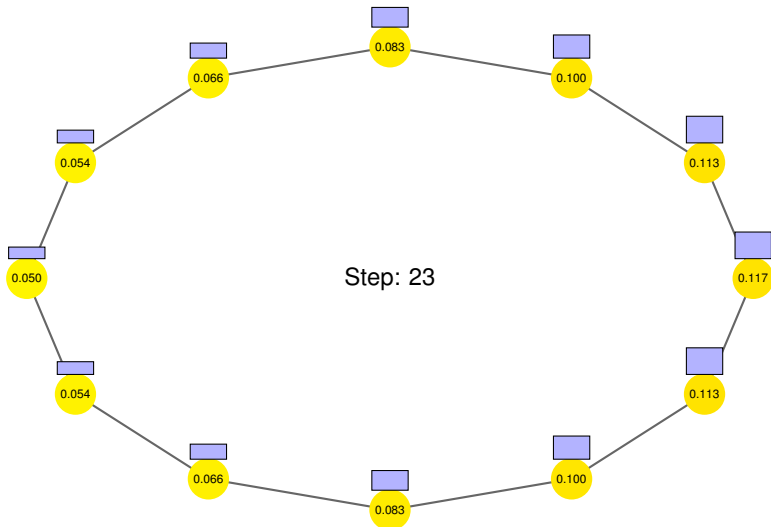
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



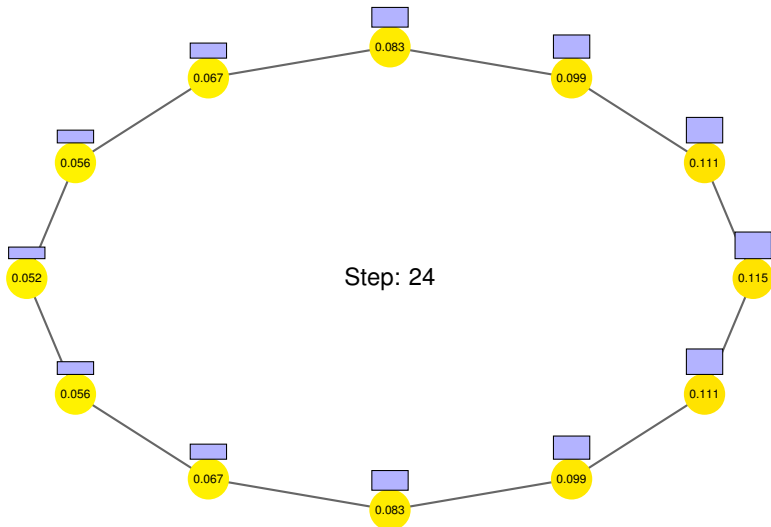
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



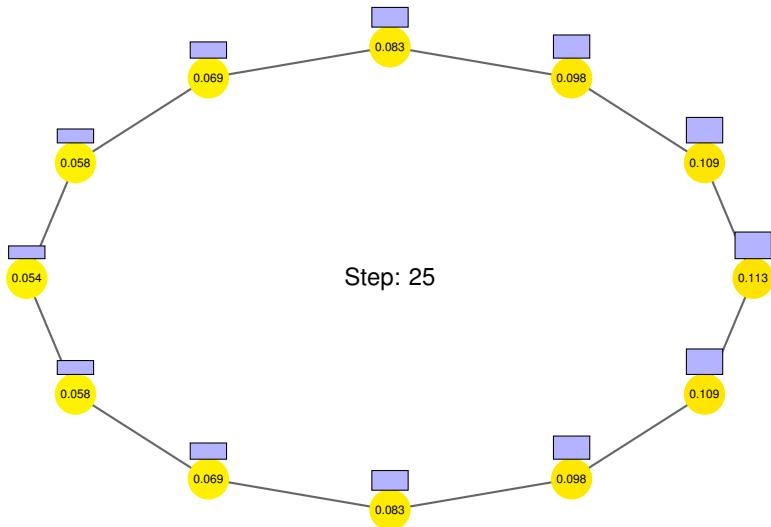
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



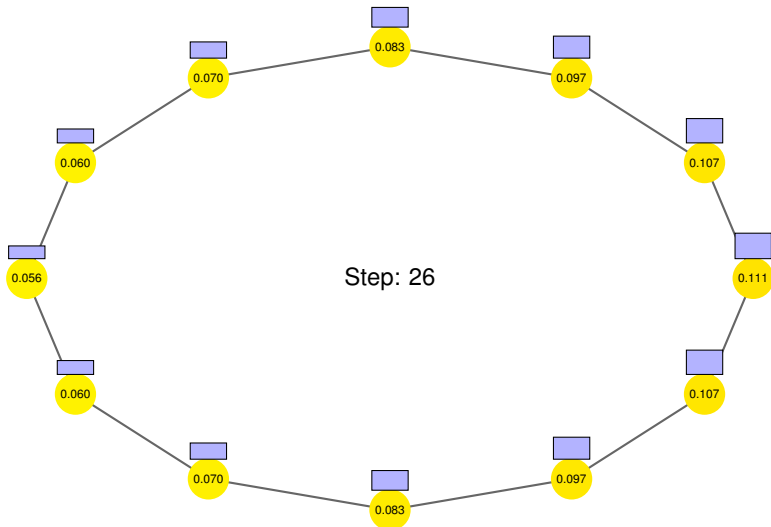
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



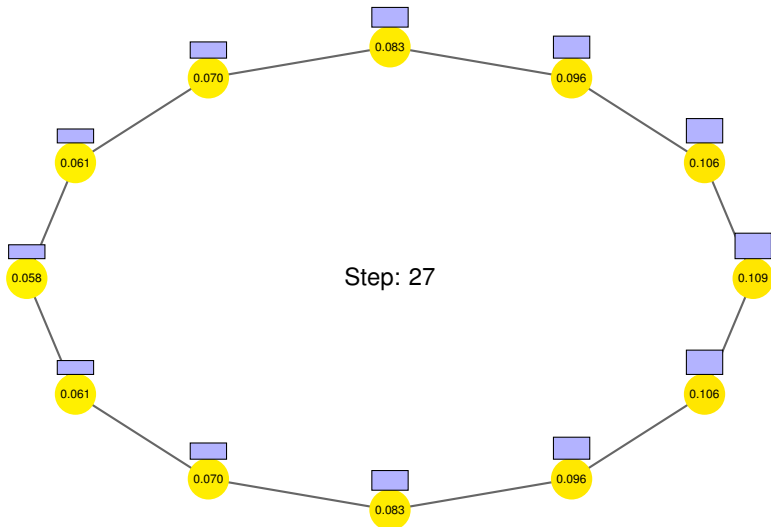
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



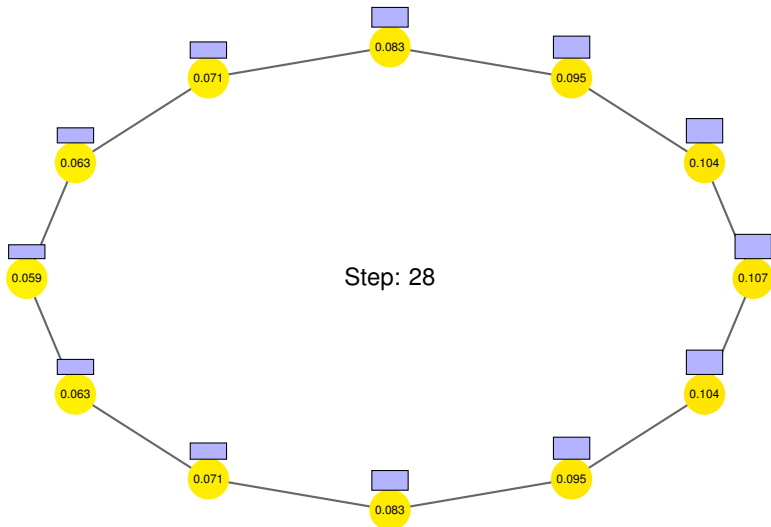
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



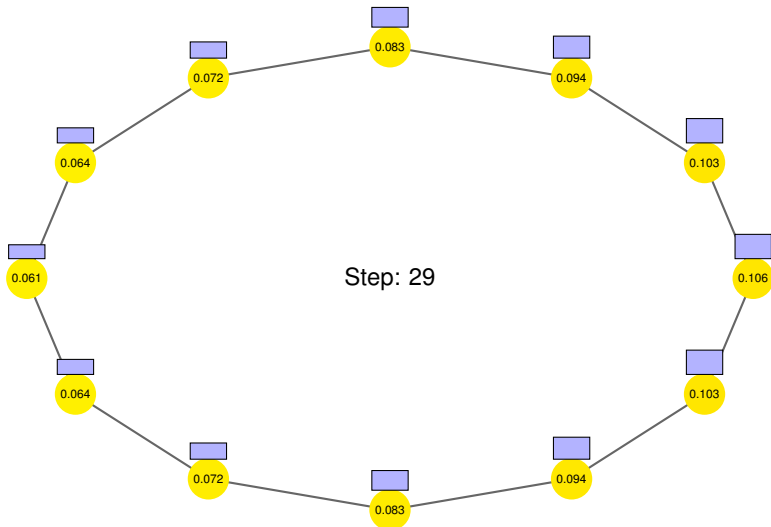
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



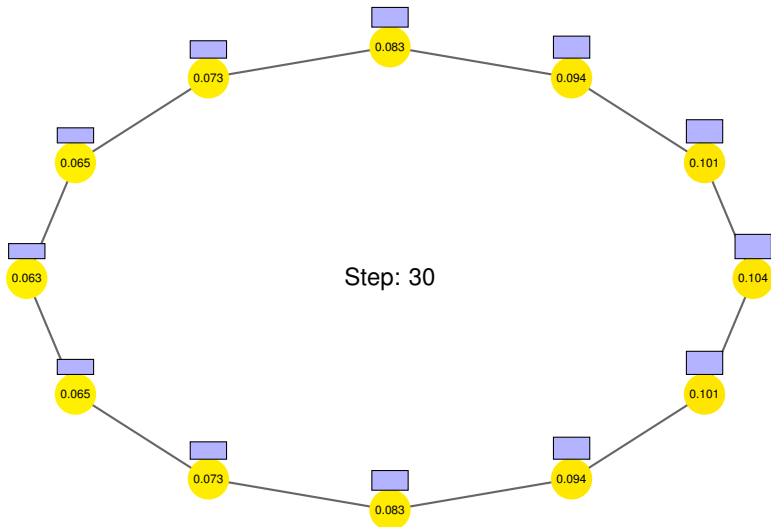
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



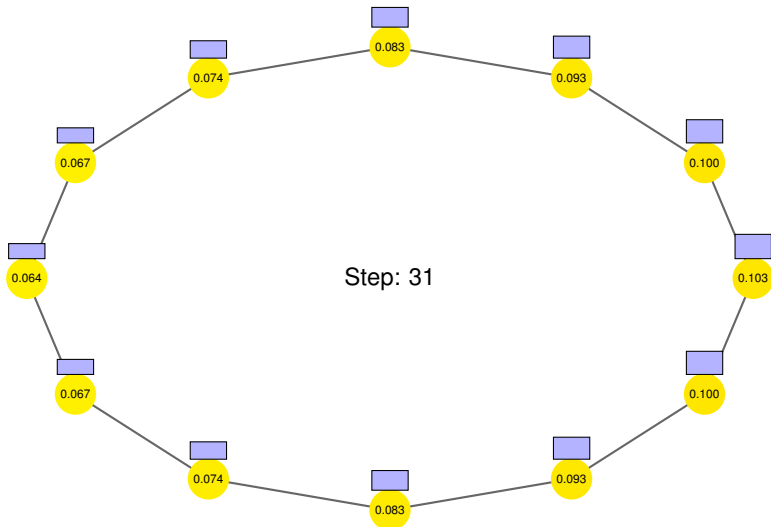
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



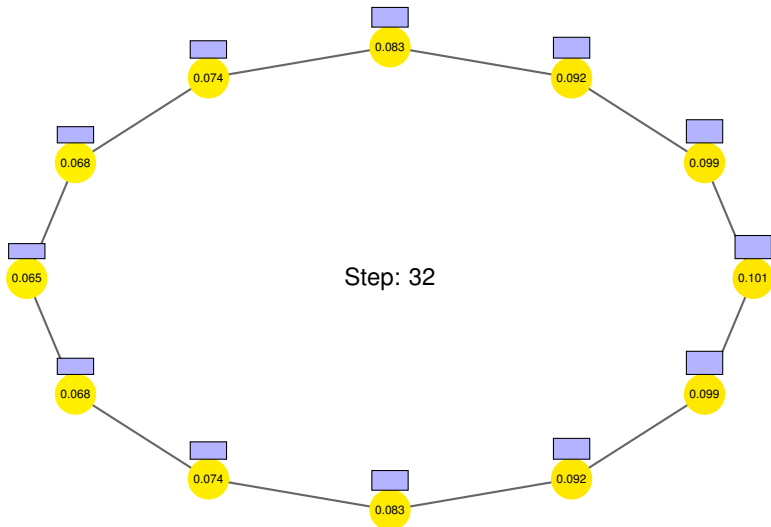
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



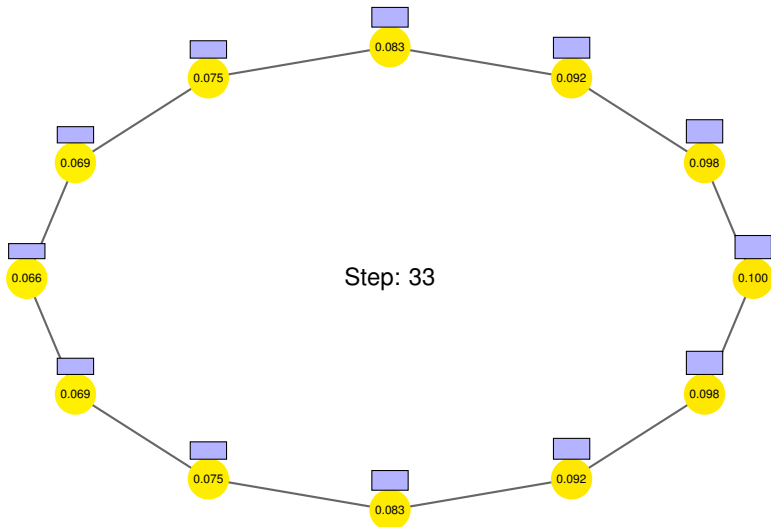
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



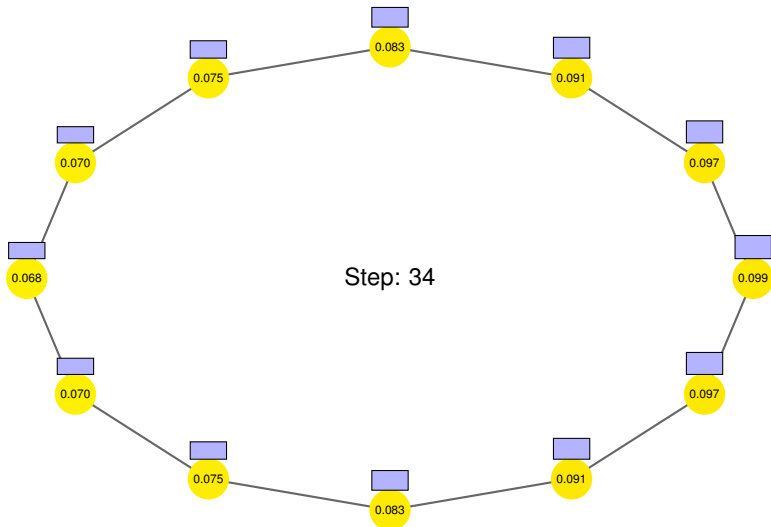
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



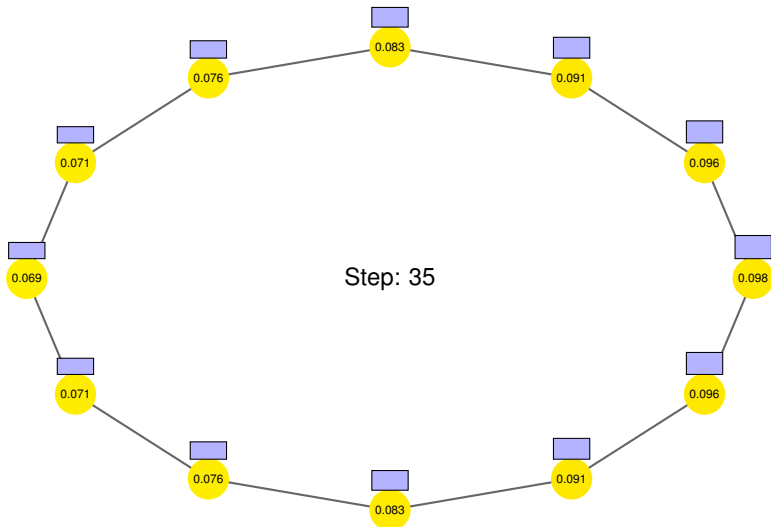
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



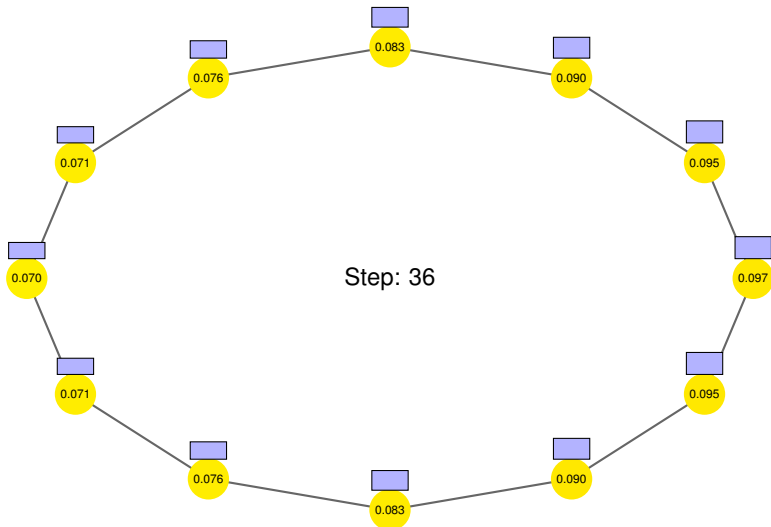
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



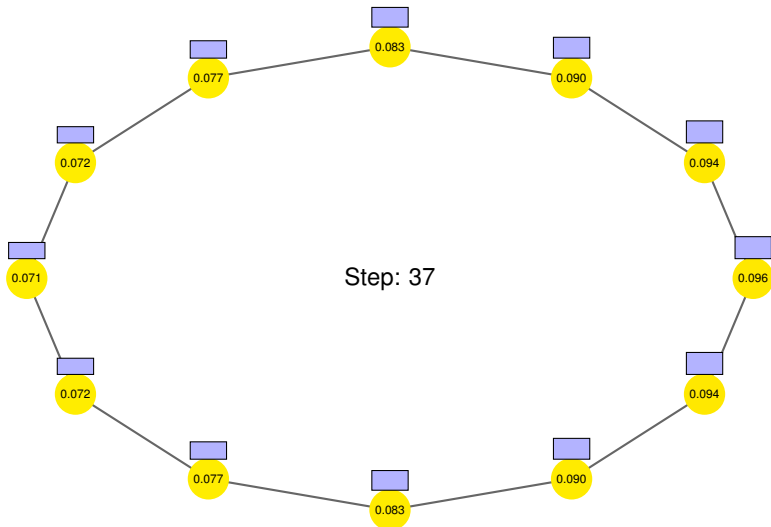
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



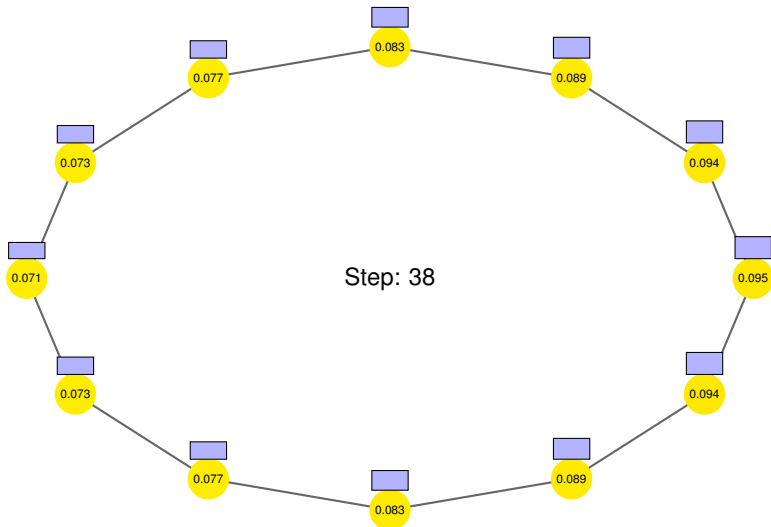
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



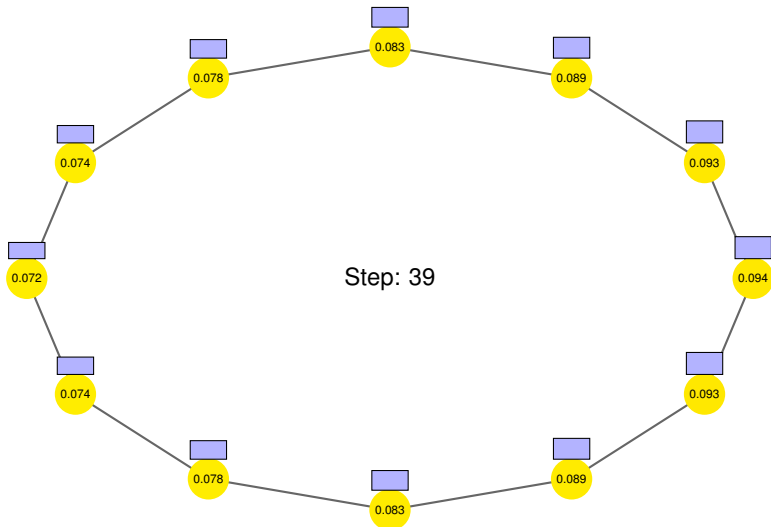
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



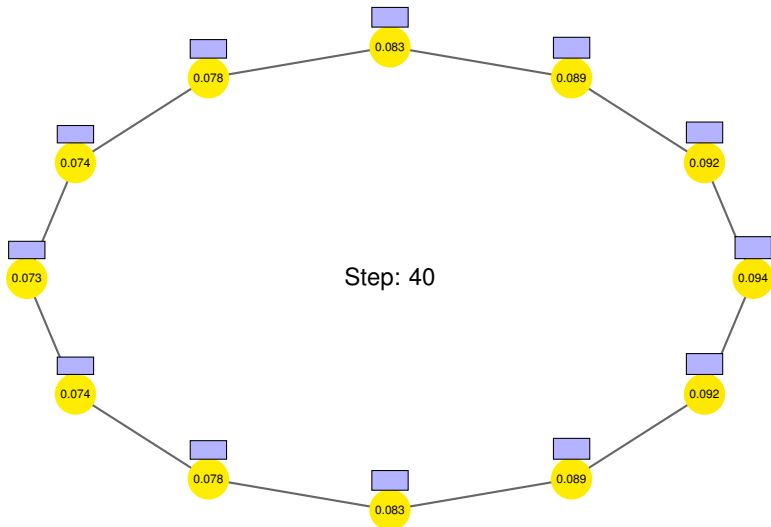
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



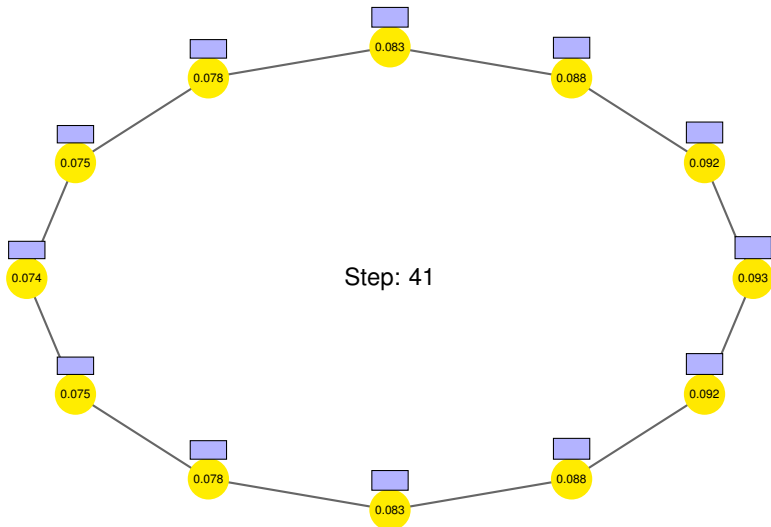
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



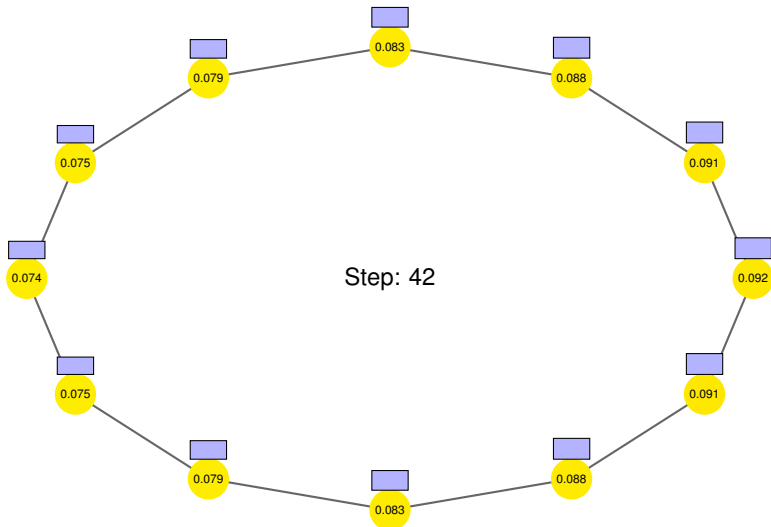
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



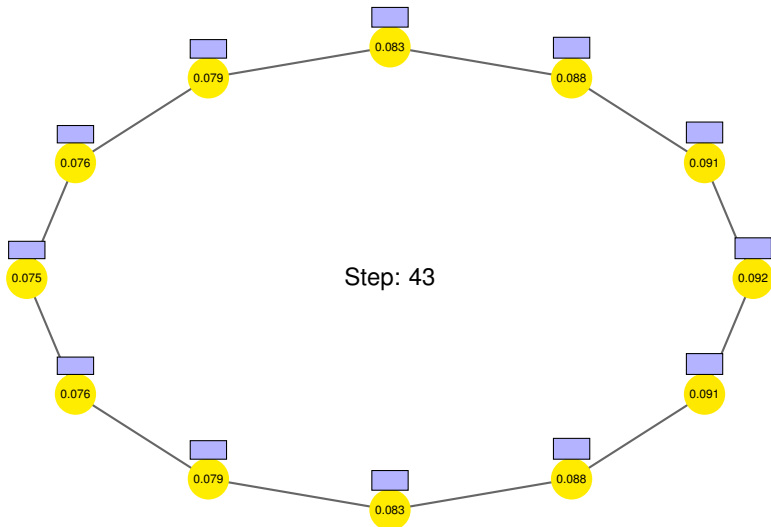
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



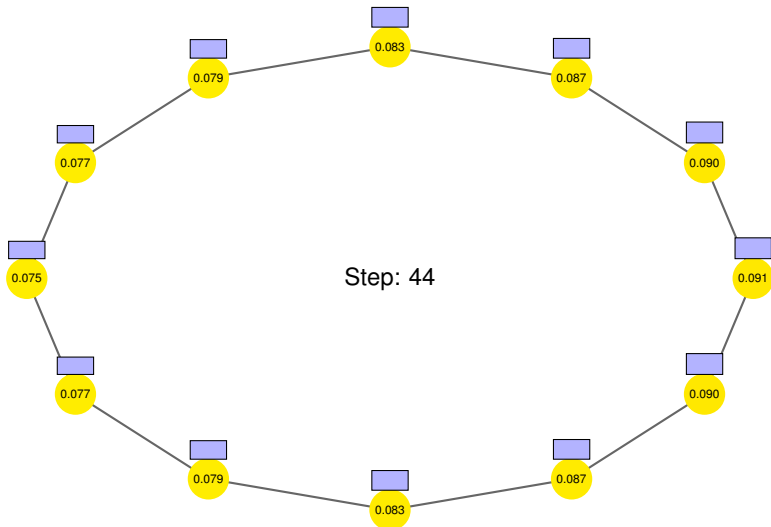
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



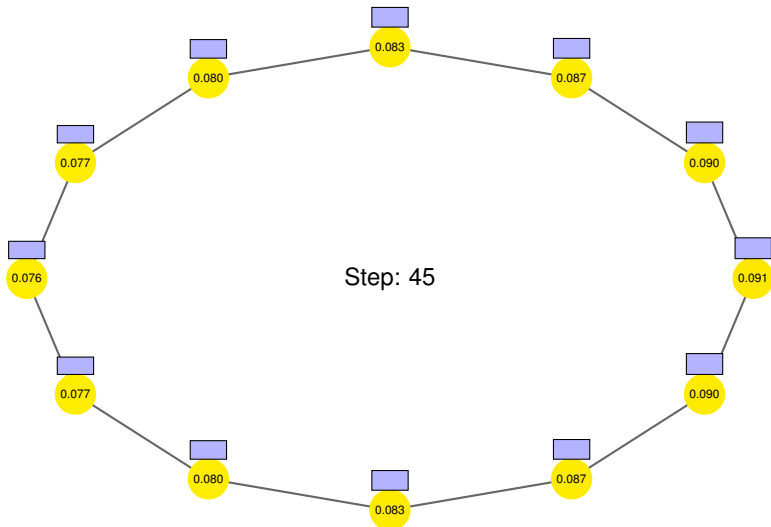
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



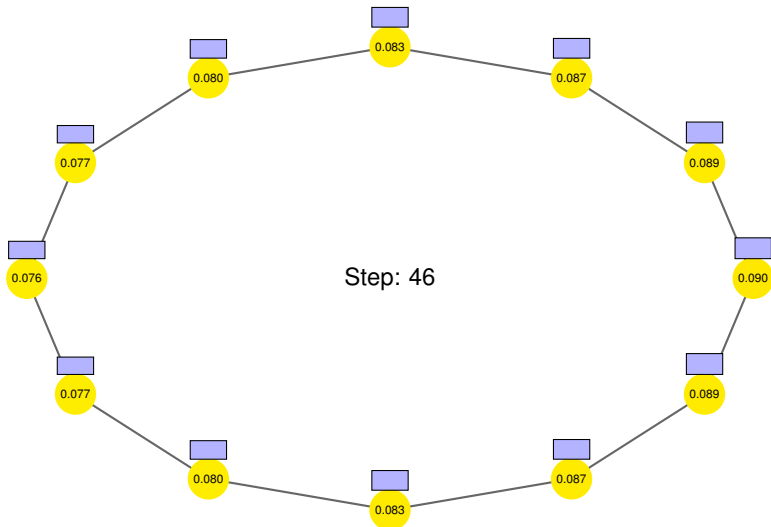
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



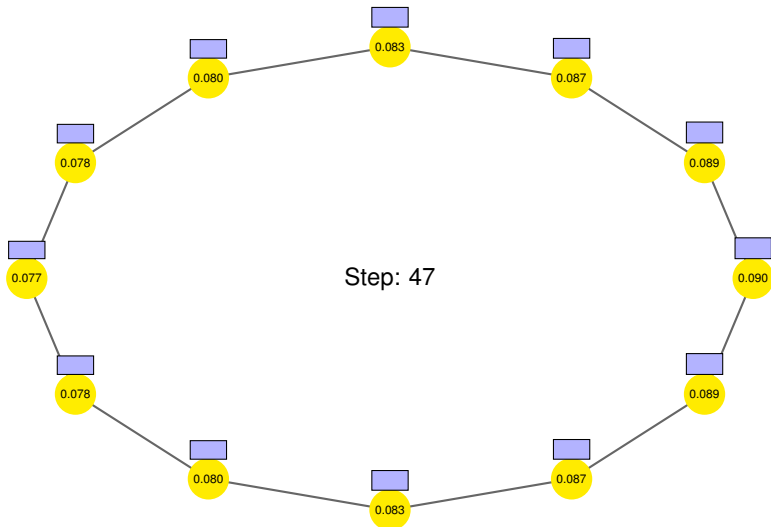
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



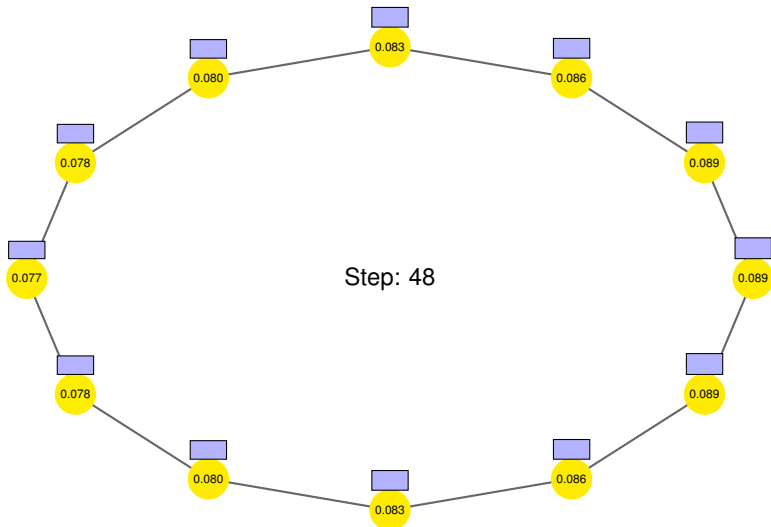
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



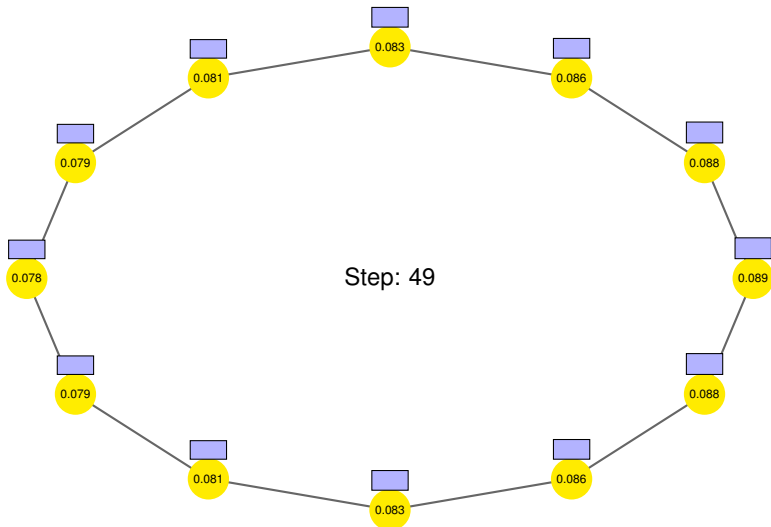
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



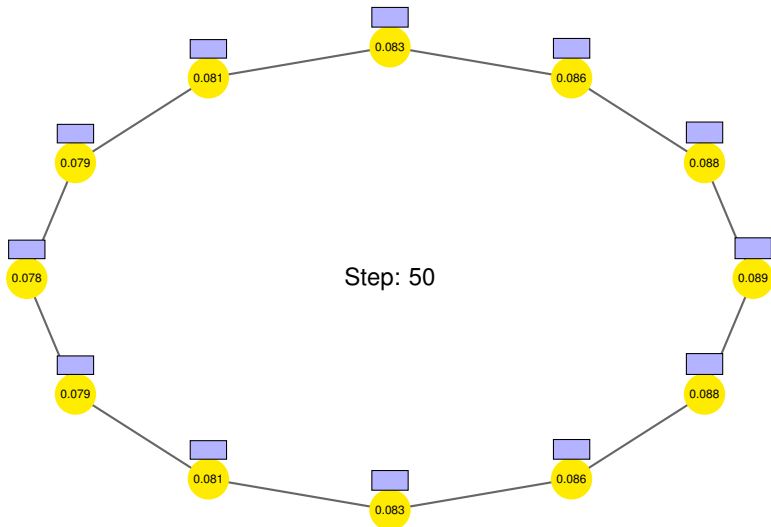
Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



Convergence to Stationarity (Example)

- **Markov Chain:** stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \dots, 12\}$ is $P^t(1, x)$.



Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

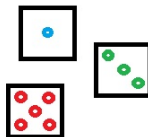
Appendix: Remarks on Mixing Time (non-examin.)

How Similar are Two Probability Measures?

Loaded Dice

- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



How Similar are Two Probability Measures?

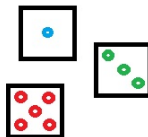
Loaded Dice

- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair?



How Similar are Two Probability Measures?

Loaded Dice

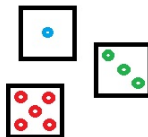
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair?

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

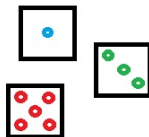
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair?

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

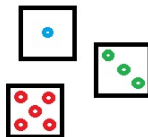
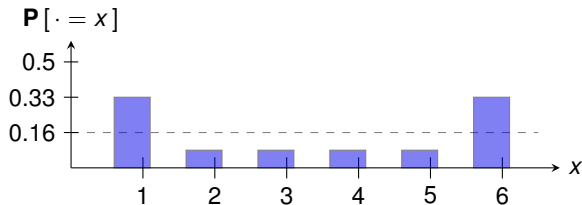
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair?

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

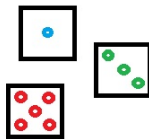
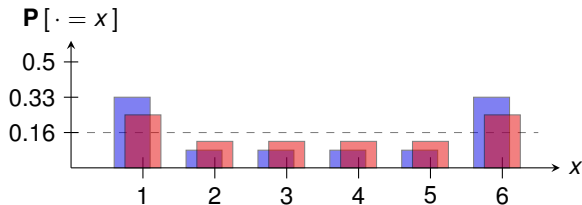
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair?

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

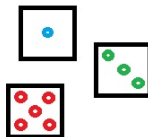
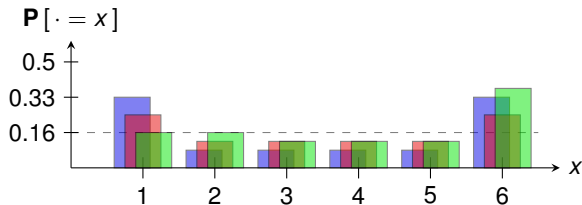
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair?

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

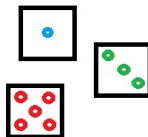
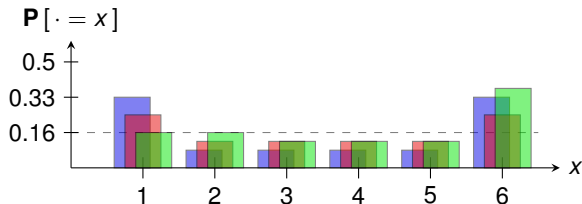
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair? Most choose A .

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

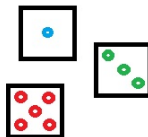
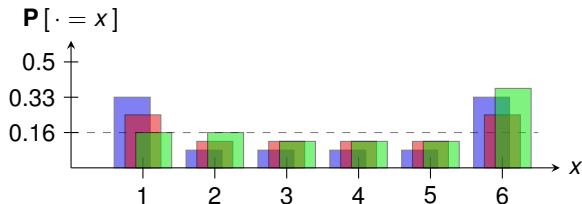
- You are presented three loaded (unfair) dice A, B, C :

x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair? Most choose A .
Why?

Question 2: Which dice is the most fair?



How Similar are Two Probability Measures?

Loaded Dice

- You are presented three loaded (unfair) dice A, B, C :

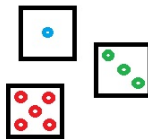
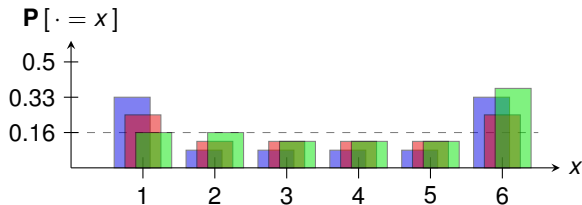
x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least fair? Most choose A .

Why?

Question 2: Which dice is the most fair? Dice B and C seem “fairer” than A but which is fairest?



How Similar are Two Probability Measures?

Loaded Dice

- You are presented three **loaded** (unfair) dice A, B, C :

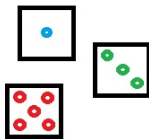
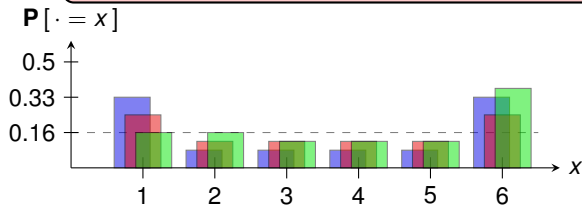
x	1	2	3	4	5	6
$P[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$P[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$P[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24



Question 1: Which dice is the least **fair**? Most choose **A**.
Why?

Question 2: Which dice is the most **fair**? Dice **B** and **C** seem “fairer” than **A** but which is **fairest**?

We need a **formal “fairness measure”** to compare probability distributions!



Total Variation Distance

The **Total Variation Distance** between two probability distributions μ and η on a countable state space Ω is given by

$$\|\mu - \eta\|_{tv} = \frac{1}{2} \sum_{\omega \in \Omega} |\mu(\omega) - \eta(\omega)|.$$

Total Variation Distance

The **Total Variation Distance** between two probability distributions μ and η on a countable state space Ω is given by

$$\|\mu - \eta\|_{tv} = \frac{1}{2} \sum_{\omega \in \Omega} |\mu(\omega) - \eta(\omega)|.$$

Loaded Dice: let $D = \text{Unif}\{1, 2, 3, 4, 5, 6\}$ be the law of a **fair dice**:

$$\|D - A\|_{tv} = \frac{1}{2} \left(2 \left| \frac{1}{6} - \frac{1}{3} \right| + 4 \left| \frac{1}{6} - \frac{1}{12} \right| \right) = \frac{1}{3}$$

Total Variation Distance

The **Total Variation Distance** between two probability distributions μ and η on a countable state space Ω is given by

$$\|\mu - \eta\|_{tv} = \frac{1}{2} \sum_{\omega \in \Omega} |\mu(\omega) - \eta(\omega)|.$$

Loaded Dice: let $D = \text{Unif}\{1, 2, 3, 4, 5, 6\}$ be the law of a **fair dice**:

$$\|D - A\|_{tv} = \frac{1}{2} \left(2 \left| \frac{1}{6} - \frac{1}{3} \right| + 4 \left| \frac{1}{6} - \frac{1}{12} \right| \right) = \frac{1}{3}$$

$$\|D - B\|_{tv} = \frac{1}{2} \left(2 \left| \frac{1}{6} - \frac{1}{4} \right| + 4 \left| \frac{1}{6} - \frac{1}{8} \right| \right) = \frac{1}{6}$$

$$\|D - C\|_{tv} = \frac{1}{2} \left(3 \left| \frac{1}{6} - \frac{1}{8} \right| + \left| \frac{1}{6} - \frac{9}{24} \right| \right) = \frac{1}{6}.$$

Total Variation Distance

The **Total Variation Distance** between two probability distributions μ and η on a countable state space Ω is given by

$$\|\mu - \eta\|_{tv} = \frac{1}{2} \sum_{\omega \in \Omega} |\mu(\omega) - \eta(\omega)|.$$

Loaded Dice: let $D = \text{Unif}\{1, 2, 3, 4, 5, 6\}$ be the law of a **fair dice**:

$$\|D - A\|_{tv} = \frac{1}{2} \left(2 \left| \frac{1}{6} - \frac{1}{3} \right| + 4 \left| \frac{1}{6} - \frac{1}{12} \right| \right) = \frac{1}{3}$$

$$\|D - B\|_{tv} = \frac{1}{2} \left(2 \left| \frac{1}{6} - \frac{1}{4} \right| + 4 \left| \frac{1}{6} - \frac{1}{8} \right| \right) = \frac{1}{6}$$

$$\|D - C\|_{tv} = \frac{1}{2} \left(3 \left| \frac{1}{6} - \frac{1}{8} \right| + \left| \frac{1}{6} - \frac{9}{24} \right| \right) = \frac{1}{6}.$$

Thus

$$\|D - B\|_{tv} = \|D - C\|_{tv} \quad \text{and} \quad \|D - B\|_{tv}, \|D - C\|_{tv} < \|D - A\|_{tv}.$$

So **A** is the least “fair”, however **B** and **C** are equally “fair” (in TV distance).

TV Distances and Markov Chains

Let P be a finite Markov Chain with stationary distribution π .

Let P be a finite Markov Chain with stationary distribution π .

- Let μ be a prob. vector on Ω (might be just one vertex) and $t \geq 0$. Then

$$P_{\mu}^t := \mathbf{P}[X_t = \cdot \mid X_0 \sim \mu],$$

is a probability measure on Ω .

Let P be a finite Markov Chain with stationary distribution π .

- Let μ be a prob. vector on Ω (might be just one vertex) and $t \geq 0$. Then

$$P_{\mu}^t := \mathbf{P}[X_t = \cdot \mid X_0 \sim \mu],$$

is a probability measure on Ω .

- [\[Exercise 4/5.5\]](#) For any μ ,

$$\left\| P_{\mu}^t - \pi \right\|_{tv} \leq \max_{x \in \Omega} \left\| P_x^t - \pi \right\|_{tv}.$$

TV Distances and Markov Chains

Let P be a finite Markov Chain with stationary distribution π .

- Let μ be a prob. vector on Ω (might be just one vertex) and $t \geq 0$. Then

$$P_{\mu}^t := \mathbf{P}[X_t = \cdot \mid X_0 \sim \mu],$$

is a probability measure on Ω .

- [Exercise 4/5.5] For any μ ,

$$\left\| P_{\mu}^t - \pi \right\|_{tv} \leq \max_{x \in \Omega} \left\| P_x^t - \pi \right\|_{tv}.$$

Convergence Theorem (Implication for TV Distance)

For any finite, irreducible, aperiodic Markov Chain

$$\lim_{t \rightarrow \infty} \max_{x \in \Omega} \left\| P_x^t - \pi \right\|_{tv} = 0.$$

TV Distances and Markov Chains

Let P be a finite Markov Chain with stationary distribution π .

- Let μ be a prob. vector on Ω (might be just one vertex) and $t \geq 0$. Then

$$P_{\mu}^t := \mathbf{P}[X_t = \cdot \mid X_0 \sim \mu],$$

is a probability measure on Ω .

- [Exercise 4/5.5] For any μ ,

$$\left\| P_{\mu}^t - \pi \right\|_{tv} \leq \max_{x \in \Omega} \left\| P_x^t - \pi \right\|_{tv}.$$

Convergence Theorem (Implication for TV Distance)

For any finite, irreducible, aperiodic Markov Chain

$$\lim_{t \rightarrow \infty} \max_{x \in \Omega} \left\| P_x^t - \pi \right\|_{tv} = 0.$$

We will see a similar result later after introducing spectral techniques (Lecture 12)!

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Question: How fast do they converge?

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Question: How fast do they converge?

Mixing Time

The **mixing time** $\tau_X(\epsilon)$ of a finite Markov Chain P with stationary distribution π is defined as

$$\tau_X(\epsilon) = \min \left\{ t \geq 0 : \left\| P_X^t - \pi \right\|_{tv} \leq \epsilon \right\},$$

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Question: How fast do they converge?

Mixing Time

The **mixing time** $\tau_x(\epsilon)$ of a finite Markov Chain P with stationary distribution π is defined as

$$\tau_x(\epsilon) = \min \left\{ t \geq 0 : \left\| P_x^t - \pi \right\|_{tv} \leq \epsilon \right\},$$

and,

$$\tau(\epsilon) = \max_x \tau_x(\epsilon).$$

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Question: How fast do they converge?

Mixing Time

The **mixing time** $\tau_X(\epsilon)$ of a finite Markov Chain P with stationary distribution π is defined as

$$\tau_X(\epsilon) = \min \left\{ t \geq 0 : \left\| P_X^t - \pi \right\|_{tv} \leq \epsilon \right\},$$

and,

$$\tau(\epsilon) = \max_X \tau_X(\epsilon).$$

- This is how long we need to wait until we are “ ϵ -close” to stationarity

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Question: How fast do they converge?

Mixing Time

The **mixing time** $\tau_X(\epsilon)$ of a finite Markov Chain P with stationary distribution π is defined as

$$\tau_X(\epsilon) = \min \left\{ t \geq 0 : \left\| P_X^t - \pi \right\|_{tv} \leq \epsilon \right\},$$

and,

$$\tau(\epsilon) = \max_X \tau_X(\epsilon).$$

- This is how long we need to wait until we are “ ϵ -close” to stationarity
- We often take $\epsilon = 1/4$, indeed let $t_{mix} := \tau(1/4)$

Mixing Time of a Markov Chain

Convergence Theorem: “Nice” Markov Chains converge to stationarity.

Question: How fast do they converge?

Mixing Time

The **mixing time** $\tau_X(\epsilon)$ of a finite Markov Chain P with stationary distribution π is defined as

$$\tau_X(\epsilon) = \min \left\{ t \geq 0 : \left\| P_X^t - \pi \right\|_{tv} \leq \epsilon \right\},$$

and,

$$\tau(\epsilon) = \max_X \tau_X(\epsilon).$$

- This is how long we need to wait until we are “ ϵ -close” to stationarity
- We often take $\epsilon = 1/4$, indeed let $t_{mix} := \tau(1/4)$

See final slides for some comments on why we choose $1/4$.

Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

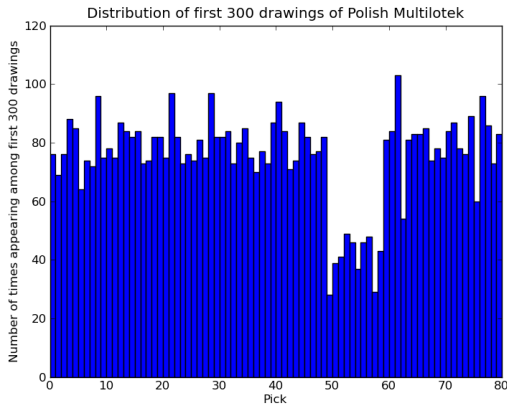
Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

Appendix: Remarks on Mixing Time (non-examin.)

Experiment Gone Wrong...



Thanks to Krzysztof Onak (pointer) and Eric Price (graph)

Source: Slides by Ronitt Rubinfeld

What is Card Shuffling?



Source: wikipedia

How long does it take to shuffle a deck of 52 cards?

What is Card Shuffling?



Source: wikipedia

How long does it take to shuffle a deck of 52 cards?



Persi Diaconis (Professor of Statistics and former Magician)

Source: www.soundcloud.com

What is Card Shuffling?



Source: wikipedia

How long does it take to **shuffle a deck of 52 cards**?



One of the leading experts in the field who has related card shuffling to many other mathematical problems.

Persi Diaconis (Professor of Statistics and former Magician)

Source: www.soundcloud.com

What is Card Shuffling?



Source: wikipedia

Here we will focus on one **shuffling scheme** which is easy to analyse.

How long does it take to **shuffle a deck of 52 cards**?

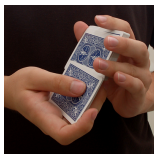


One of the leading experts in the field who has related card shuffling to many other mathematical problems.

Persi Diaconis (Professor of Statistics and former Magician)

Source: www.soundcloud.com

What is Card Shuffling?



Source: wikipedia

Here we will focus on one **shuffling scheme** which is easy to analyse.

How long does it take to **shuffle a deck of 52 cards**?

How quickly do we converge to the **uniform distribution** over all $n!$ permutations?



One of the leading experts in the field who has related card shuffling to many other mathematical problems.

Persi Diaconis (Professor of Statistics and former Magician)

Source: www.soundcloud.com

The Card Shuffling Markov Chain

TOPTORANDOMSHUFFLE (Input: A pile of n cards)

- 1: **For** $t = 1, 2, \dots$
- 2: Pick $i \in \{1, 2, \dots, n\}$ uniformly at random
- 3: Take the top card and insert it behind the i -th card

The Card Shuffling Markov Chain

TOPTORANDOMSHUFFLE (Input: A pile of n cards)

- 1: **For** $t = 1, 2, \dots$
- 2: Pick $i \in \{1, 2, \dots, n\}$ uniformly at random
- 3: Take the top card and insert it behind the i -th card

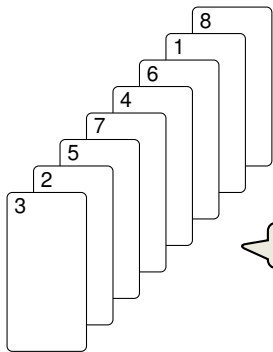
This is a slightly informal definition, so let us look at a small example...

The Card Shuffling Markov Chain

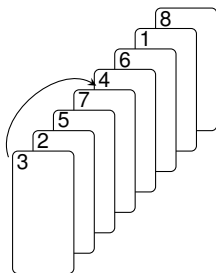
TOPTORANDOMSHUFFLE (Input: A pile of n cards)

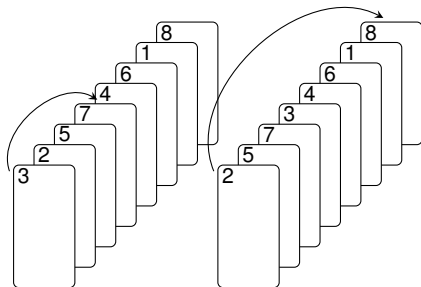
- 1: **For** $t = 1, 2, \dots$
- 2: Pick $i \in \{1, 2, \dots, n\}$ **uniformly at random**
- 3: Take the top card and insert it behind the i -th card

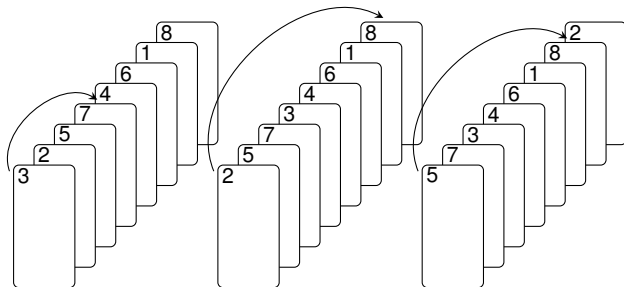
This is a slightly informal definition, so let us look at a small **example...**

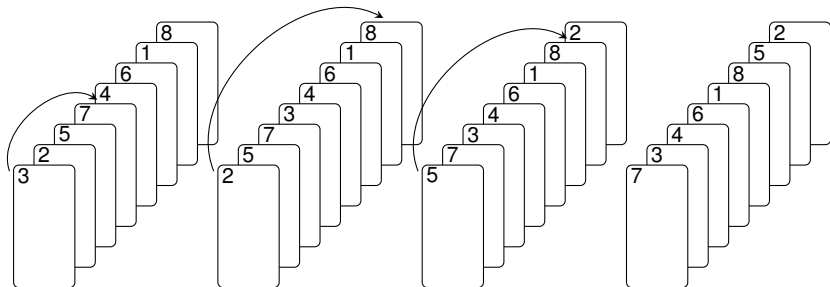


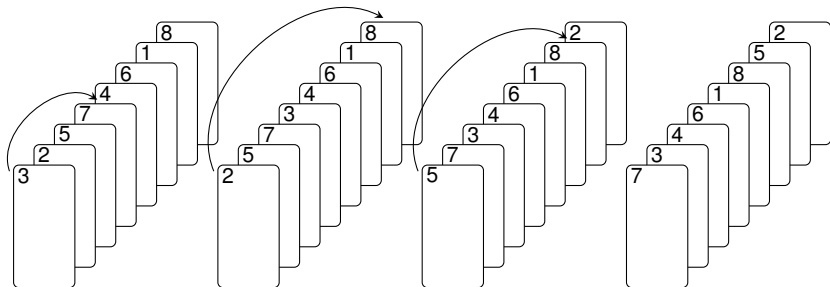
We will focus on this “small” set of cards ($n = 8$)



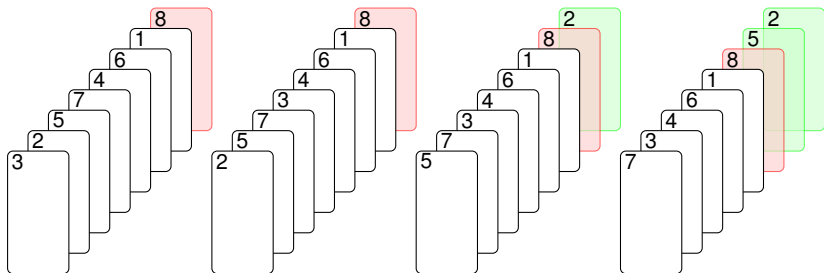




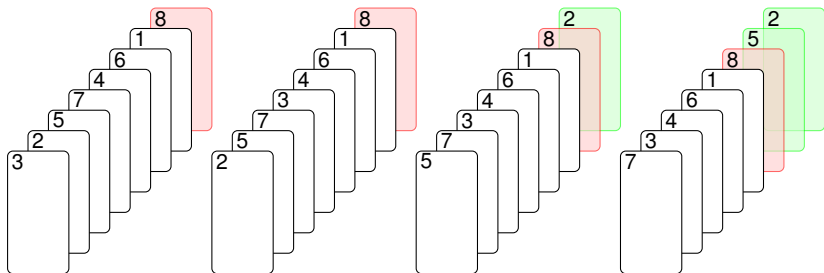




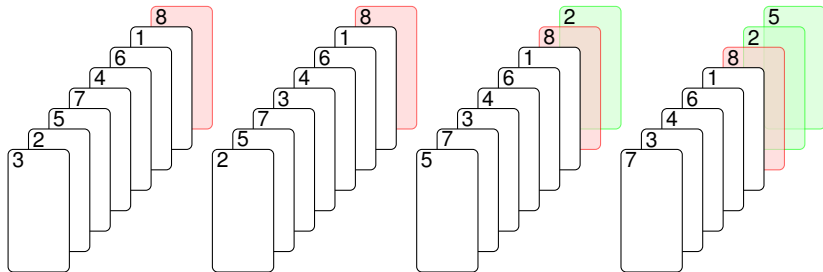
Even if we know which set of cards come after 8, every permutation is equally likely!

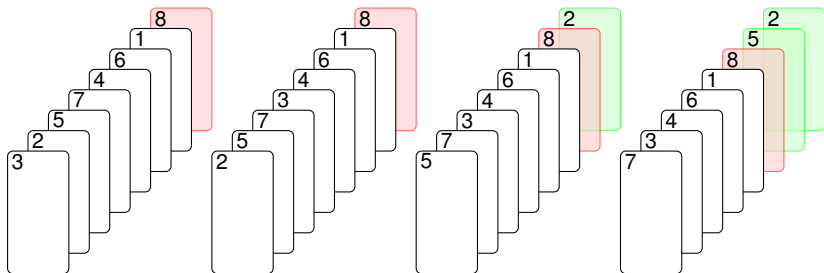


Even if we know which set of cards come after 8, every permutation is equally likely!

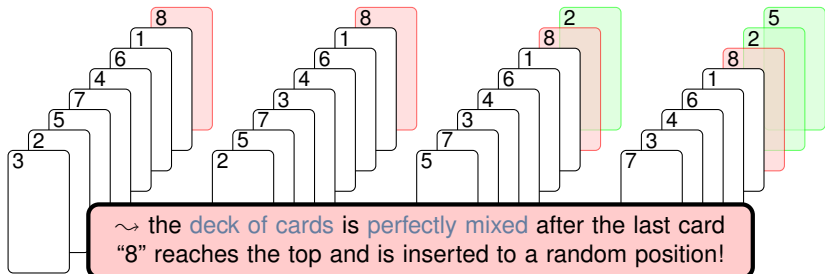


Even if we know which set of cards come after 8, every permutation is equally likely!

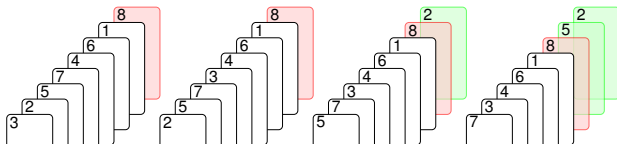




Even if we know which set of cards come after 8, every permutation is equally likely!

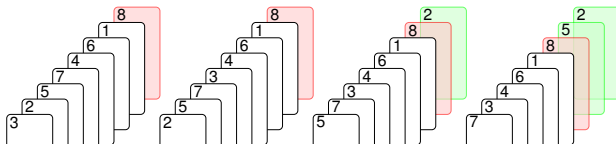


Analysing the Mixing Time (Intuition)



~> deck of cards is perfectly mixed after the last card "8" reaches the top and is inserted to a random position!

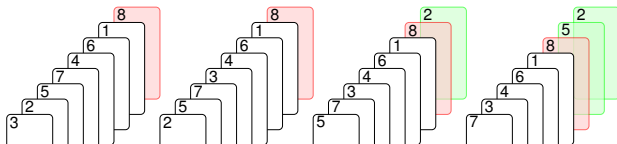
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?

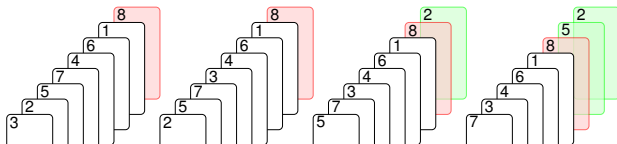
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step

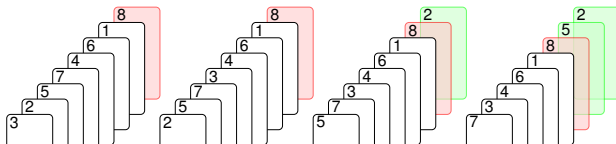
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$

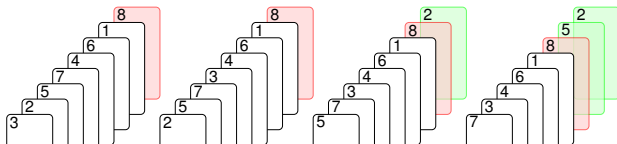
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$
- \vdots

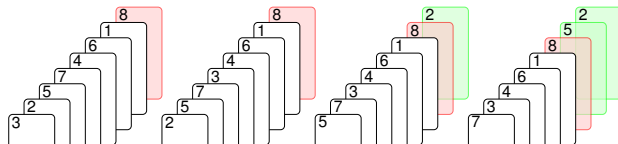
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$
- \vdots
- At the second position, card “ n ” moves up with probability $\frac{n-1}{n}$

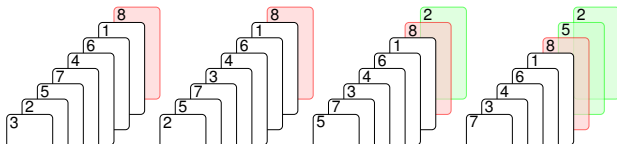
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$
- \vdots
- At the second position, card “ n ” moves up with probability $\frac{n-1}{n}$
- One final step to randomise card “ n ”

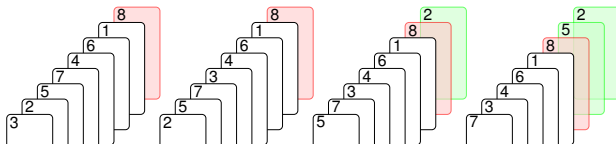
Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$
- \vdots
- At the second position, card “ n ” moves up with probability $\frac{n-1}{n}$
- One final step to randomise card “ n ” (with probability 1)

Analysing the Mixing Time (Intuition)

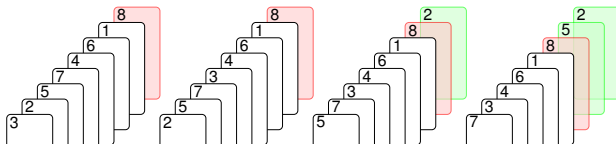


↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$
- \vdots
- At the second position, card “ n ” moves up with probability $\frac{n-1}{n}$
- One final step to randomise card “ n ” (with probability 1)

This is a “reversed” coupon collector process with n cards, which takes $n \log n$ in expectation.

Analysing the Mixing Time (Intuition)



↪ deck of cards is perfectly mixed after the last card “8” reaches the top and is inserted to a random position!

- How long does it take for the last card “ n ” to become top card?
- At the last position, card “ n ” moves up with probability $\frac{1}{n}$ at each step
- At the second last position, card “ n ” moves up with probability $\frac{2}{n}$
- \vdots
- At the second position, card “ n ” moves up with probability $\frac{n-1}{n}$
- One final step to randomise card “ n ” (with probability 1)

This is a “reversed” coupon collector process with n cards, which takes $n \log n$ in expectation.

Using the so-called coupling method, one could prove $t_{\text{mix}} \leq n \log n$.

Riffle Shuffle

Riffle Shuffle

1. Split a deck of n cards into two piles (thus the size of each portion will be Binomial)

Riffle Shuffle

Riffle Shuffle

1. **Split** a deck of n cards into two piles (thus the size of each portion will be **Binomial**)
2. **Riffle** the cards together so that the card drops from the left (or right) pile with probability proportional to the number of remaining cards

Riffle Shuffle

Riffle Shuffle

1. **Split** a deck of n cards into two piles (thus the size of each portion will be **Binomial**)
2. **Riffle** the cards together so that the card drops from the left (or right) pile with probability proportional to the number of remaining cards

a

A	2	3	4	5	6	7	8	9	10	J	Q	K
---	---	---	---	---	---	---	---	---	----	---	---	---

b

A	2	3	4	5	6	7	8	9	10	J	Q	K
---	---	---	---	---	---	---	---	---	----	---	---	---

c

A		2			3		4	5		6		
	7		8	9		10		J		Q	K	

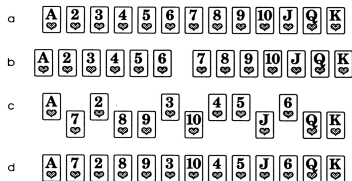
d

A	7	2	8	9	3	10	4	5	J	6	Q	K
---	---	---	---	---	---	----	---	---	---	---	---	---

Riffle Shuffle

Riffle Shuffle

1. Split a deck of n cards into two piles (thus the size of each portion will be Binomial)
2. Riffle the cards together so that the card drops from the left (or right) pile with probability proportional to the number of remaining cards



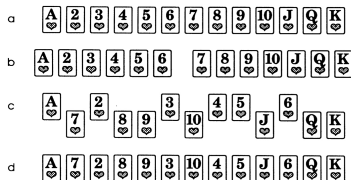
t	1	2	3	4	5	6	7	8	9	10
$\ P^t - \pi\ _{tv}$	1.000	1.000	1.000	1.000	0.924	0.614	0.334	0.167	0.085	0.043

Figure: Total Variation Distance for t riffle shuffles of 52 cards.

Riffle Shuffle

Riffle Shuffle

1. Split a deck of n cards into two piles (thus the size of each portion will be Binomial)
2. Riffle the cards together so that the card drops from the left (or right) pile with probability proportional to the number of remaining cards



The Annals of Applied Probability
1992, Vol. 2, No. 2, 294–313

TRAILING THE DOVETAIL SHUFFLE TO ITS LAIR

By DAVE BAYER¹ AND PERSI DIACONIS²

Columbia University and Harvard University

We analyze the most commonly used method for shuffling cards. The main result is a simple expression for the chance of any arrangement after any number of shuffles. This is used to give sharp bounds on the approach to randomness: $\frac{3}{2} \log_2 n + \theta$ shuffles are necessary and sufficient to mix up n cards.

Key ingredients are the analysis of a card trick and the determination of the idempotents of a natural commutative subalgebra in the symmetric group algebra.

t	1	2	3	4	5	6	7	8	9	10
$\ P^t - \pi\ _{TV}$	1.000	1.000	1.000	1.000	0.924	0.614	0.334	0.167	0.085	0.043

Figure: Total Variation Distance for t riffle shuffles of 52 cards.

Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

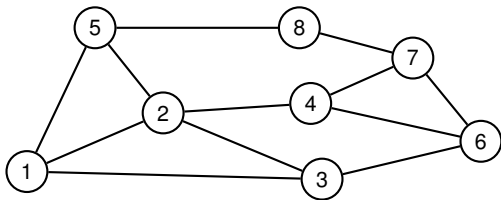
Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

Appendix: Remarks on Mixing Time (non-examin.)

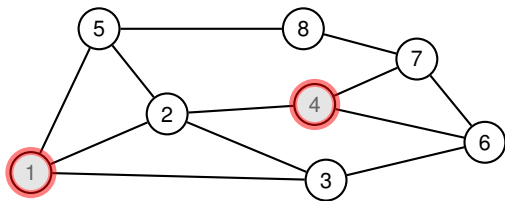
Markov Chain for Sampling Independent Sets (1/2) (non-examin.)



Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

Markov Chain for Sampling Independent Sets (1/2) (non-examin.)

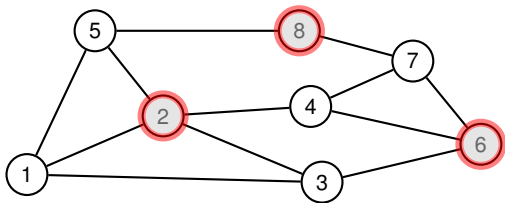


$S = \{1, 4\}$ is an independent set ✓

Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

Markov Chain for Sampling Independent Sets (1/2) (non-examin.)

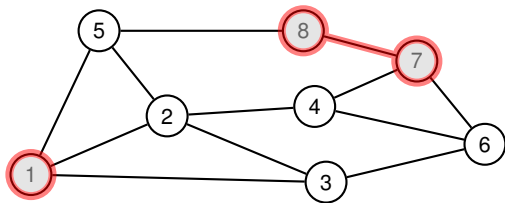


$S = \{2, 6, 8\}$ is an independent set ✓

Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

Markov Chain for Sampling Independent Sets (1/2) (non-examin.)

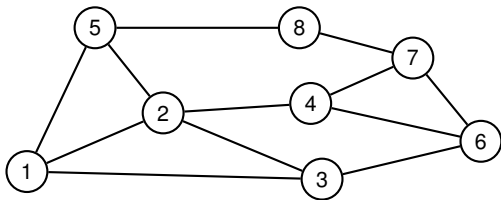


$S = \{1, 7, 8\}$ is **not** an independent set ✗

Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

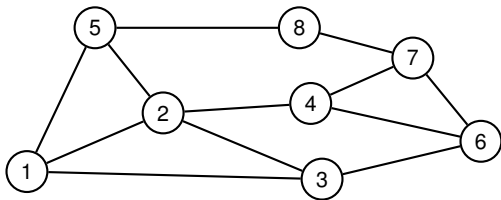
Markov Chain for Sampling Independent Sets (1/2) (non-examin.)



Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

Markov Chain for Sampling Independent Sets (1/2) (non-examin.)

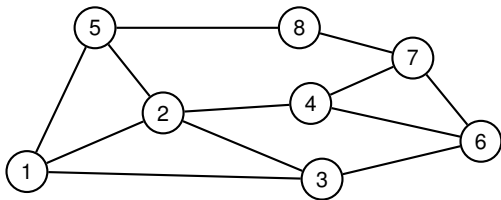


Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

How can we take a **sample** from the **space of all independent sets**?

Markov Chain for Sampling Independent Sets (1/2) (non-examin.)



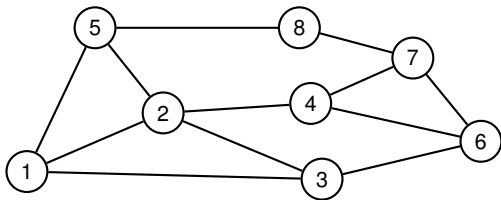
Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

How can we take a **sample** from the **space of all independent sets**?

Naive brute-force would take an insane amount of time (and space)!

Markov Chain for Sampling Independent Sets (1/2) (non-examin.)



Independent Set

Given an undirected graph $G = (V, E)$, an **independent set** is a subset $S \subseteq V$ such that there are no two vertices $u, v \in S$ with $\{u, v\} \in E(G)$.

How can we take a **sample** from the **space of all independent sets**?

Naive brute-force would take an insane amount of time (and space)!

We can use a **generic Markov Chain Monte Carlo** approach to tackle this problem!

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

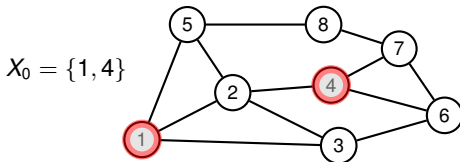
INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

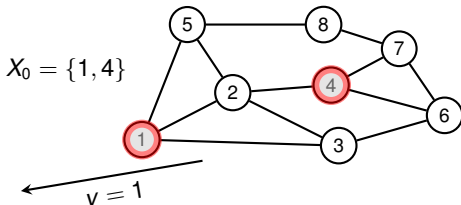
- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

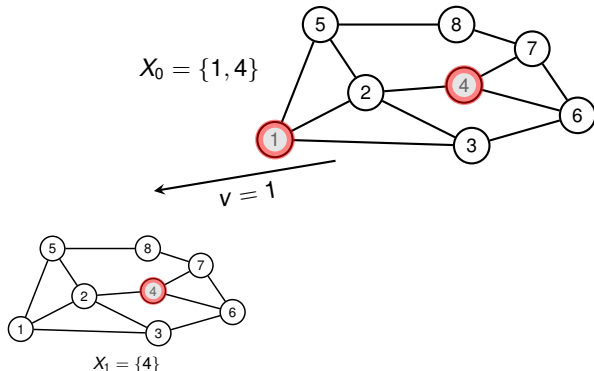
- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

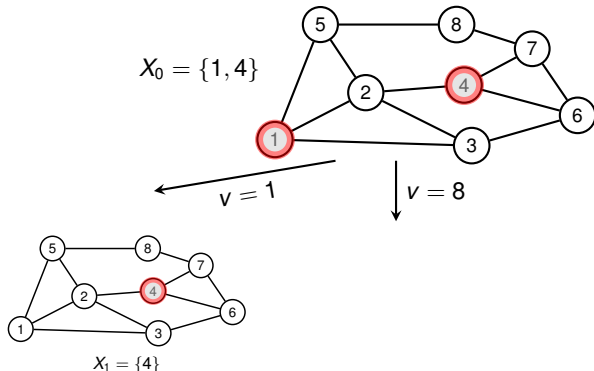
- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

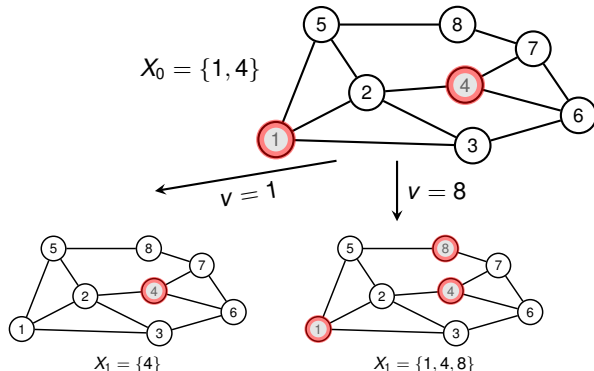
- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

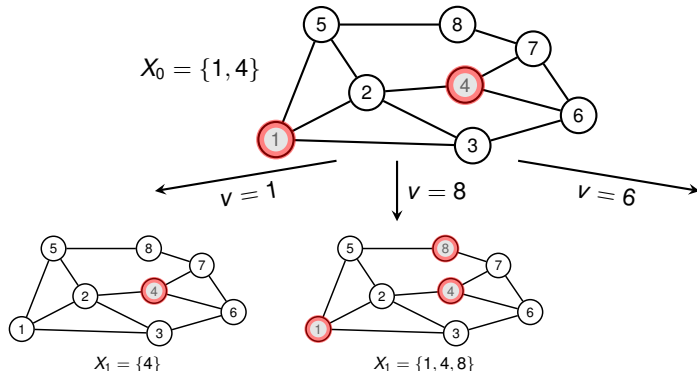
- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

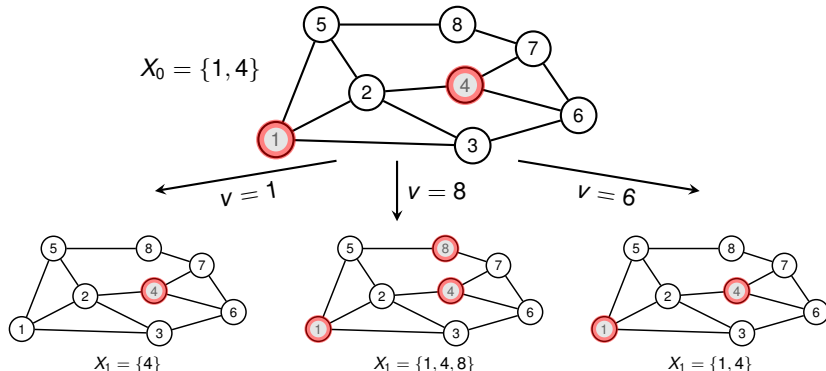
- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$



Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

- This is a **local** definition (no explicit definition of $P!$)

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

- This is a **local** definition (no explicit definition of P !)
- This chain is **irreducible** (every independent set is reachable)

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

- This is a **local** definition (no explicit definition of P !)
- This chain is **irreducible** (every independent set is reachable)
- This chain is **aperiodic** (Check!)

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

- This is a **local** definition (no explicit definition of $P!$)
- This chain is **irreducible** (every independent set is reachable)
- This chain is **aperiodic** (Check!)
- The **stationary distribution** is uniform, since $P_{u,v} = P_{v,u}$ (Check!)

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

- This is a **local** definition (no explicit definition of $P!$)
- This chain is **irreducible** (every independent set is reachable)
- This chain is **aperiodic** (Check!)
- The **stationary distribution** is uniform, since $P_{u,v} = P_{v,u}$ (Check!)

Key Question: What is the **mixing time** of this Markov Chain?

Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENTSETSAMPLER

- 1: Let X_0 be an arbitrary independent set in G
- 2: **For** $t = 0, 1, 2, \dots$:
- 3: Pick a vertex $v \in V(G)$ uniformly at random
- 4: **If** $v \in X_t$ **then** $X_{t+1} \leftarrow X_t \setminus \{v\}$
- 5: **elif** $v \notin X_t$ **and** $X_t \cup \{v\}$ is an independent set **then** $X_{t+1} \leftarrow X_t \cup \{v\}$
- 6: **else** $X_{t+1} \leftarrow X_t$

Remark

- This is a **local** definition (no explicit definition of $P!$)
- This chain is **irreducible** (every independent set is reachable)
- This chain is **aperiodic** (Check!)
- The **stationary distribution** is uniform, since $P_{u,v} = P_{v,u}$ (Check!)

Key Question: What is the **mixing time** of this Markov Chain?

not covered here, see the textbook by Mitzenmacher and Upfal

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

Appendix: Remarks on Mixing Time (non-examin.)

Further Remarks on the Mixing Time (non-examin.)

- One can prove $\max_x \|P_x^t - \pi\|_{tv}$ is non-increasing in t (this means if the chain is “ ϵ -mixed” at step t , then this also holds in future steps) [\[Mitzenmacher, Upfal, 12.3\]](#)

Further Remarks on the Mixing Time (non-examin.)

- One can prove $\max_X \|P_X^t - \pi\|_{tv}$ is non-increasing in t (this means if the chain is “ ϵ -mixed” at step t , then this also holds in future steps) [\[Mitzenmacher, Upfal, 12.3\]](#)
- We chose $t_{mix} := \tau(1/4)$, but other choices of ϵ are perfectly fine too (e.g, $t_{mix} := \tau(1/e)$ is often used); in fact, any constant $\epsilon \in (0, 1/2)$ is possible.

Further Remarks on the Mixing Time (non-examin.)

- One can prove $\max_x \|P_x^t - \pi\|_{tv}$ is non-increasing in t (this means if the chain is “ ϵ -mixed” at step t , then this also holds in future steps) [Mitzenmacher, Upfal, 12.3]
- We chose $t_{mix} := \tau(1/4)$, but other choices of ϵ are perfectly fine too (e.g., $t_{mix} := \tau(1/e)$ is often used); in fact, any constant $\epsilon \in (0, 1/2)$ is possible.

Remark: This freedom on how to pick ϵ relies on the sub-multiplicative property of a (version) of the variation distance. First, let

$$d(t) := \max_x \|P_x^t - \pi\|_{tv}$$

be the variation distance after t steps when starting from the worst state. Further, define

$$\bar{d}(t) := \max_{\mu, \nu} \|P_\mu^t - P_\nu^t\|_{tv}.$$

These quantities are related by the following double inequality

$$d(t) \leq \bar{d}(t) \leq 2d(t).$$

Further, $\bar{d}(t)$ is sub-multiplicative, that is for any $s, t \geq 1$,

$$\bar{d}(s+t) \leq \bar{d}(s) \cdot \bar{d}(t).$$

Hence for any fixed $0 < \epsilon < \delta < 1/2$ it follows from the above that

$$\tau(\epsilon) \leq \left\lceil \frac{\ln \epsilon}{\ln(2\delta)} \right\rceil \tau(\delta).$$

In particular, for any $\epsilon < 1/4$

$$\tau(\epsilon) \leq \left\lceil \log_2 \epsilon^{-1} \right\rceil \tau(1/4).$$

Further Remarks on the Mixing Time (non-examin.)

- One can prove $\max_x \|P_x^t - \pi\|_{tv}$ is non-increasing in t (this means if the chain is “ ϵ -mixed” at step t , then this also holds in future steps) [Mitzenmacher, Upfal, 12.3]
- We chose $t_{mix} := \tau(1/4)$, but other choices of ϵ are perfectly fine too (e.g., $t_{mix} := \tau(1/e)$ is often used); in fact, any constant $\epsilon \in (0, 1/2)$ is possible.

Remark: This freedom on how to pick ϵ relies on the sub-multiplicative property of a (version) of the variation distance. First, let

$$d(t) := \max_x \|P_x^t - \pi\|_{tv}$$

be the variation distance after t steps when starting from the worst state. Further, define

$$\bar{d}(t) := \max_{\mu, \nu} \|P_\mu^t - P_\nu^t\|_{tv}.$$

These quantities are related by the following double inequality

$$d(t) \leq \bar{d}(t) \leq 2d(t).$$

Further, $\bar{d}(t)$ is sub-multiplicative, that is for any $s, t \geq 1$,

$$\bar{d}(s+t) \leq \bar{d}(s) \cdot \bar{d}(t).$$

Hence for any fixed $0 < \epsilon < \delta < 1/2$ it follows from the above that

$$\tau(\epsilon) \leq \left\lceil \frac{\ln \epsilon}{\ln(2\delta)} \right\rceil \tau(\delta).$$

In particular, for any $\epsilon < 1/4$

$$\tau(\epsilon) \leq \left\lceil \log_2 \epsilon^{-1} \right\rceil \tau(1/4).$$

This 2 is the reason why we ultimately need $\epsilon < 1/2$ in this derivation. On the other hand, see [\[Exercise \(4/5\).8\]](#) why $\epsilon < 1/2$ is also necessary.

Further Remarks on the Mixing Time (non-examin.)

- One can prove $\max_x \|P_x^t - \pi\|_{tv}$ is non-increasing in t (this means if the chain is “ ϵ -mixed” at step t , then this also holds in future steps) [Mitzenmacher, Upfal, 12.3]
- We chose $t_{mix} := \tau(1/4)$, but other choices of ϵ are perfectly fine too (e.g., $t_{mix} := \tau(1/e)$ is often used); in fact, any constant $\epsilon \in (0, 1/2)$ is possible.

Remark: This freedom on how to pick ϵ relies on the sub-multiplicative property of a (version) of the variation distance. First, let

$$d(t) := \max_x \|P_x^t - \pi\|_{tv}$$

be the variation distance after t steps when starting from the worst state. Further, define

$$\bar{d}(t) := \max_{\mu, \nu} \|P_\mu^t - P_\nu^t\|_{tv}.$$

These quantities are related by the following double inequality

$$d(t) \leq \bar{d}(t) \leq 2d(t).$$

Further, $\bar{d}(t)$ is sub-multiplicative, that is for any $s, t \geq 1$,

$$\bar{d}(s+t) \leq \bar{d}(s) \cdot \bar{d}(t).$$

Hence for any fixed $0 < \epsilon < \delta < 1/2$ it follows from the above that

$$\tau(\epsilon) \leq \left\lceil \frac{\ln \epsilon}{\ln(2\delta)} \right\rceil \tau(\delta).$$

In particular, for any $\epsilon < 1/4$

$$\tau(\epsilon) \leq \left\lceil \log_2 \epsilon^{-1} \right\rceil \tau(1/4).$$

Hence smaller constants $\epsilon < 1/4$ only increase the mixing time by some constant factor.

This 2 is the reason why we ultimately need $\epsilon < 1/2$ in this derivation. On the other hand, see [\[Exercise \(4/5\).8\]](#) why $\epsilon < 1/2$ is also necessary.

Randomised Algorithms

Lecture 5: Random Walks, Hitting Times and Application to 2-SAT

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

The Ehrenfest Markov Chain

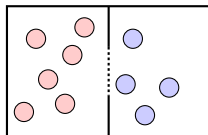
Ehrenfest Model

- A simple model for the exchange of molecules between two boxes

The Ehrenfest Markov Chain

Ehrenfest Model

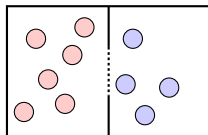
- A simple model for the exchange of molecules between two boxes



The Ehrenfest Markov Chain

Ehrenfest Model

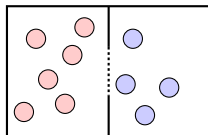
- A simple model for the exchange of molecules between two boxes
- We have d particles



The Ehrenfest Markov Chain

Ehrenfest Model

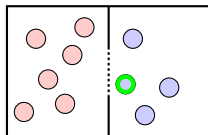
- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box



The Ehrenfest Markov Chain

Ehrenfest Model

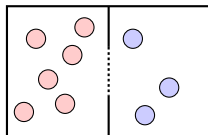
- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box



The Ehrenfest Markov Chain

Ehrenfest Model

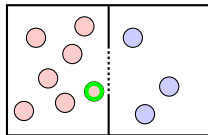
- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box



The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box

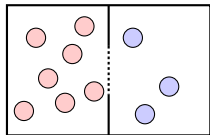


The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the **number of particles** in the red box, then:

$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$

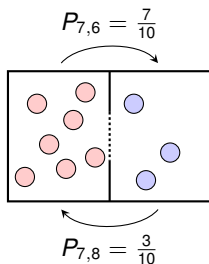


The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the **number of particles** in the red box, then:

$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$

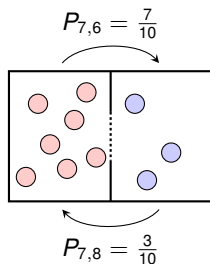


The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles
- At each step a particle is selected **uniformly at random** and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the **number of particles** in the red box, then:

$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$



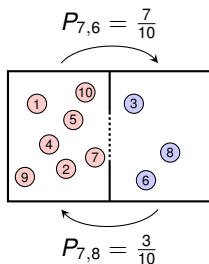
Let us now enlarge the state space by looking at each particle **individually**!

The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles labelled $1, 2, \dots, d$
- At each step a particle is selected uniformly at random and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the number of particles in the red box, then:

$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$



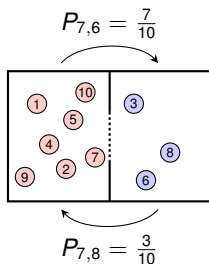
Let us now enlarge the state space by looking at each particle individually!

The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles labelled $1, 2, \dots, d$
- At each step a particle is selected uniformly at random and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the number of particles in the red box, then:

$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$



Let us now enlarge the state space by looking at each particle **individually**!

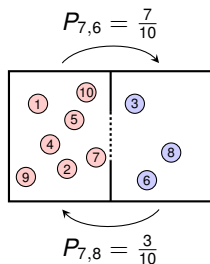
Random Walk on the Hypercube

The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles labelled $1, 2, \dots, d$
- At each step a particle is selected uniformly at random and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the number of particles in the red box, then:

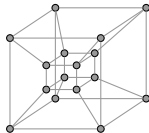
$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$



Let us now enlarge the state space by looking at each particle individually!

Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$

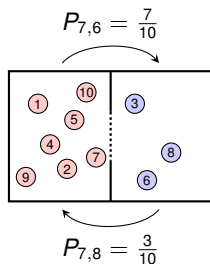


The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles labelled $1, 2, \dots, d$
- At each step a particle is selected uniformly at random and switches to the other box
- If $\Omega = \{0, 1, \dots, d\}$ denotes the number of particles in the red box, then:

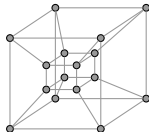
$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$



Let us now enlarge the state space by looking at each particle individually!

Random Walk on the Hypercube

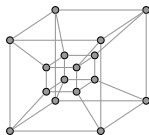
- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a random coordinate in $[d]$ and flip it



Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

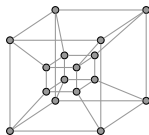


Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!



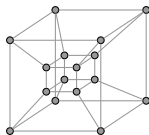
Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!



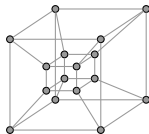
Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!



Lazy Random Walk (1st Version)

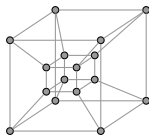
Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!



Lazy Random Walk (1st Version)

- At each step $t = 0, 1, 2 \dots$

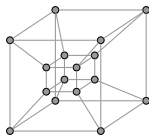
Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!



Lazy Random Walk (1st Version)

- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$

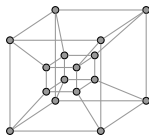
Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**

Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!



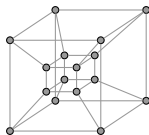
Lazy Random Walk (1st Version)

- At each step $t = 0, 1, 2, \dots$
 - Pick a **random** coordinate in $[d]$
 - With prob. $1/2$ **flip** coordinate.

Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**



Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!

Lazy Random Walk (1st Version)

- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$
 - With prob. $1/2$ **flip** coordinate.

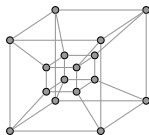
Lazy Random Walk (2nd Version)

- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$

Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**



Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!

Lazy Random Walk (1st Version)

- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$
 - With prob. $1/2$ **flip** coordinate.

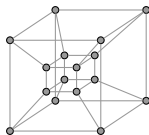
Lazy Random Walk (2nd Version)

- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$
 - Set coordinate to $\{0, 1\}$ **uniformly**.

Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a **random** coordinate in $[d]$ and **flip it**



Problem: This Markov Chain is **periodic**, as the number of ones always switches between odd to even!

Solution: Add **self-loops** to break periodic behaviour!

Lazy Random Walk (1st Version)

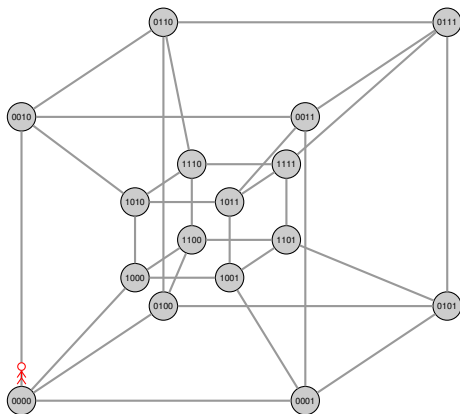
- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$
 - With prob. $1/2$ **flip** coordinate.

Lazy Random Walk (2nd Version)

- At each step $t = 0, 1, 2 \dots$
 - Pick a **random** coordinate in $[d]$
 - Set coordinate to $\{0, 1\}$ **uniformly**.

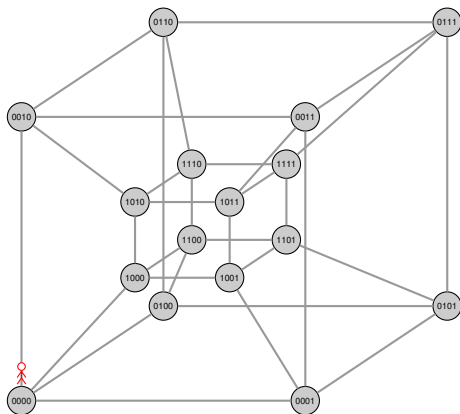
These two chains are equivalent!

Example of a Random Walk on a 4-Dimensional Hypercube



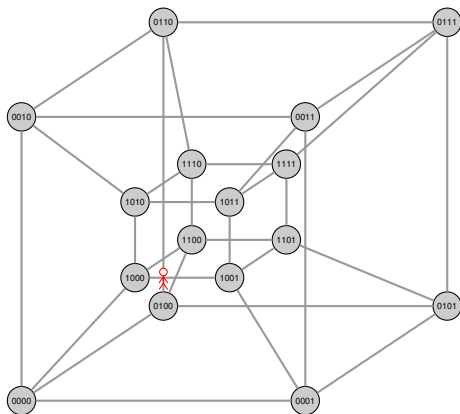
t	Coord.	X_t			
0		0	0	0	0

Example of a Random Walk on a 4-Dimensional Hypercube



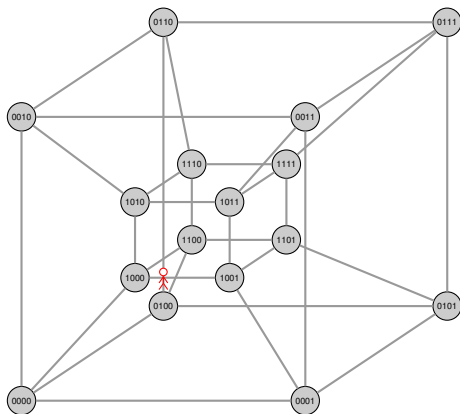
t	Coord.	X_t			
0	2	0	0	0	0
1		0	?	0	0

Example of a Random Walk on a 4-Dimensional Hypercube



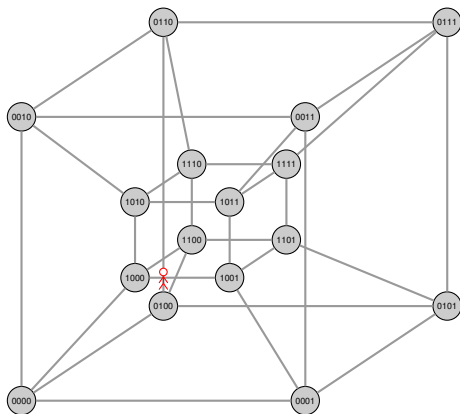
t	Coord.	X_t			
0	2	0	0	0	0
1		0	1	0	0

Example of a Random Walk on a 4-Dimensional Hypercube



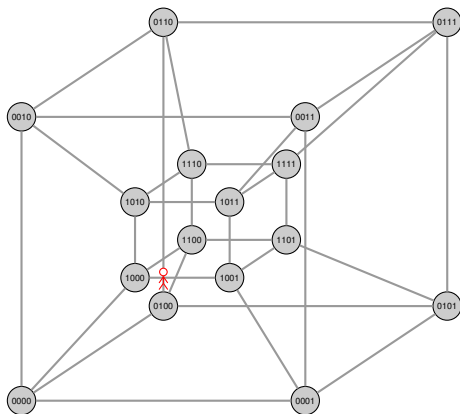
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2		0	1	?	0

Example of a Random Walk on a 4-Dimensional Hypercube



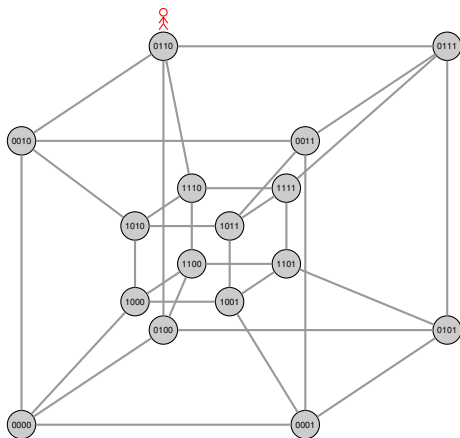
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2		0	1	0	0

Example of a Random Walk on a 4-Dimensional Hypercube



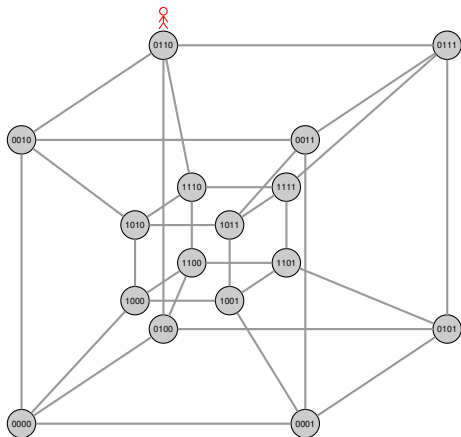
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3		0	1	?	0

Example of a Random Walk on a 4-Dimensional Hypercube



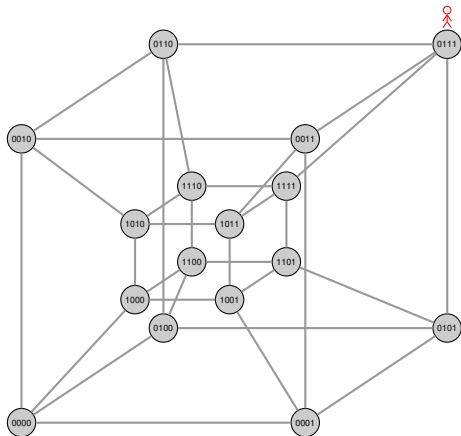
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3		0	1	1	0

Example of a Random Walk on a 4-Dimensional Hypercube



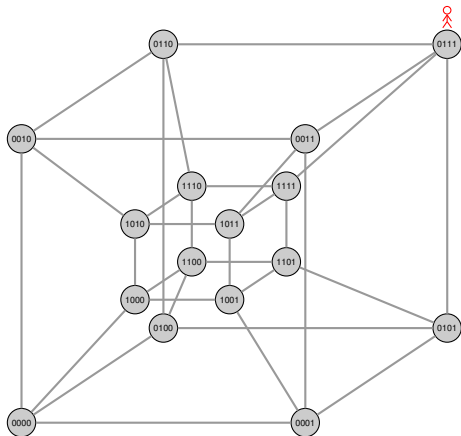
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4		0	1	1	?

Example of a Random Walk on a 4-Dimensional Hypercube



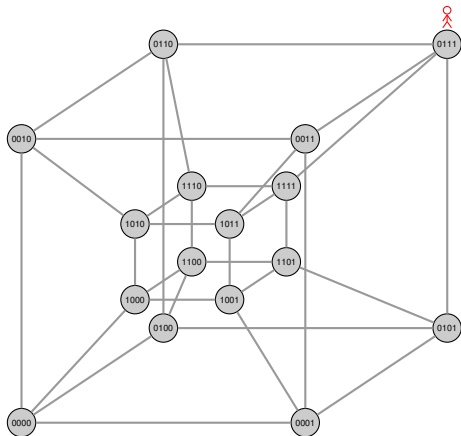
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	4	0	1	1	1

Example of a Random Walk on a 4-Dimensional Hypercube



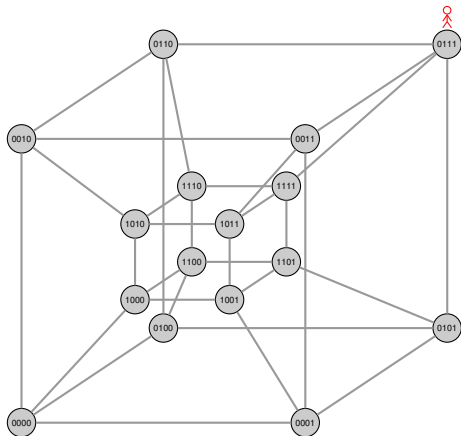
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5		0	?	1	1

Example of a Random Walk on a 4-Dimensional Hypercube



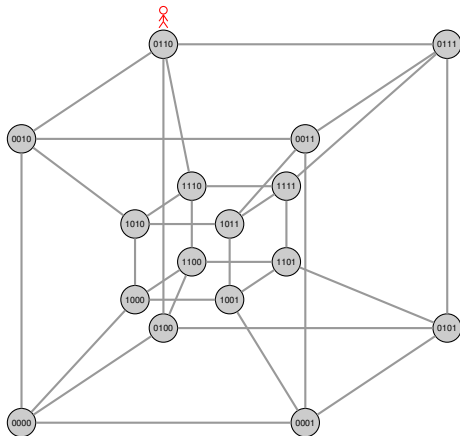
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5		0	1	1	1

Example of a Random Walk on a 4-Dimensional Hypercube



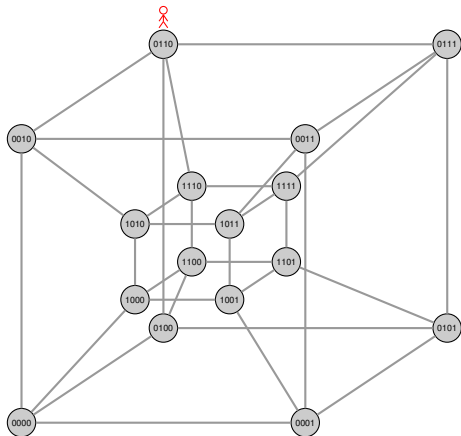
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6		0	1	1	?

Example of a Random Walk on a 4-Dimensional Hypercube



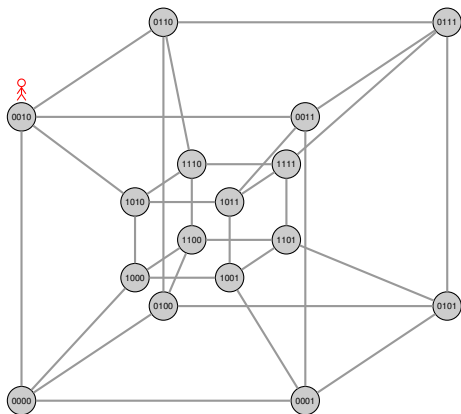
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6		0	1	1	0

Example of a Random Walk on a 4-Dimensional Hypercube



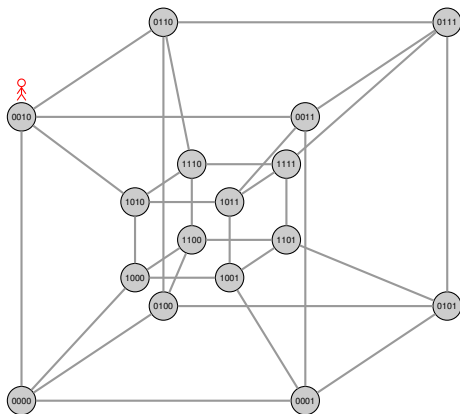
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7		0	?	1	0

Example of a Random Walk on a 4-Dimensional Hypercube



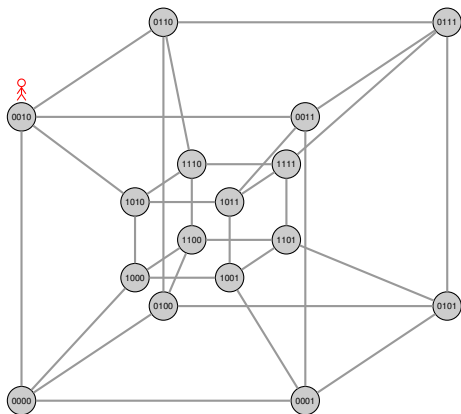
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7		0	0	1	0

Example of a Random Walk on a 4-Dimensional Hypercube



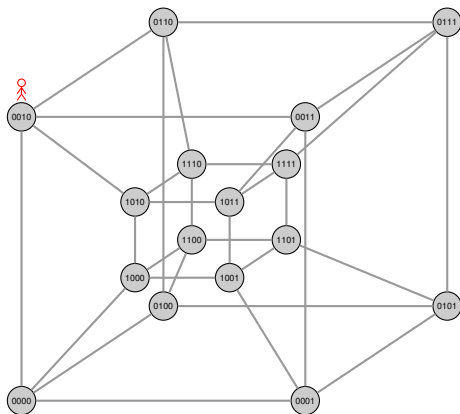
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8		0	0	1	?

Example of a Random Walk on a 4-Dimensional Hypercube



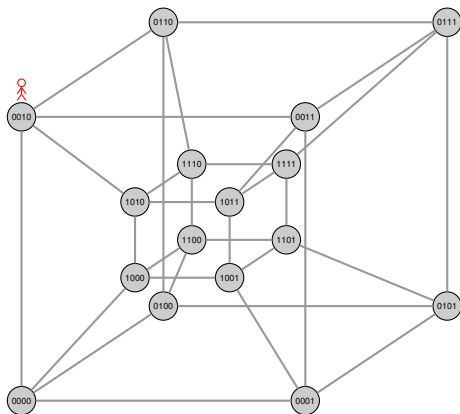
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8		0	0	1	0

Example of a Random Walk on a 4-Dimensional Hypercube



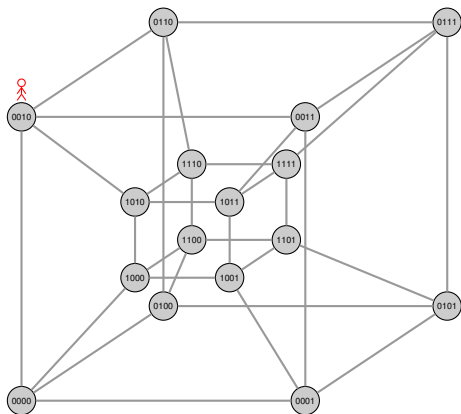
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9		0	0	?	0

Example of a Random Walk on a 4-Dimensional Hypercube



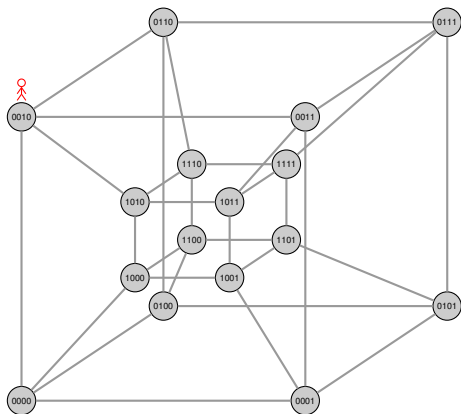
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9		0	0	1	0

Example of a Random Walk on a 4-Dimensional Hypercube



t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9	1	0	0	1	0
10	done!	?	0	1	0

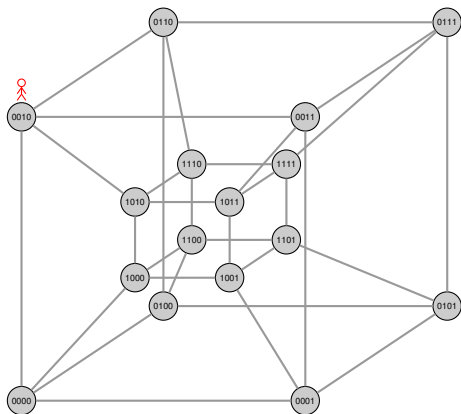
Example of a Random Walk on a 4-Dimensional Hypercube



t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9	1	0	0	1	0
10	done!	0	0	1	0

Once **all coordinates** have been **picked** at least once, the state is uniformly at random in $\{0, 1\}^d$.

Example of a Random Walk on a 4-Dimensional Hypercube

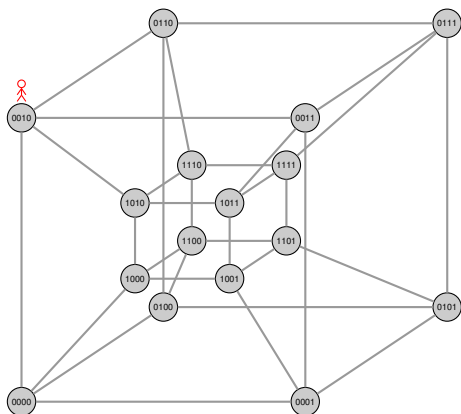


t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9	1	0	0	1	0
10	done!	0	0	1	0

Once **all coordinates** have been **picked** at least once, the state is uniformly at random in $\{0, 1\}^d$.

Coupon Collector \leadsto mixing time should be $O(d \log d)$

Example of a Random Walk on a 4-Dimensional Hypercube



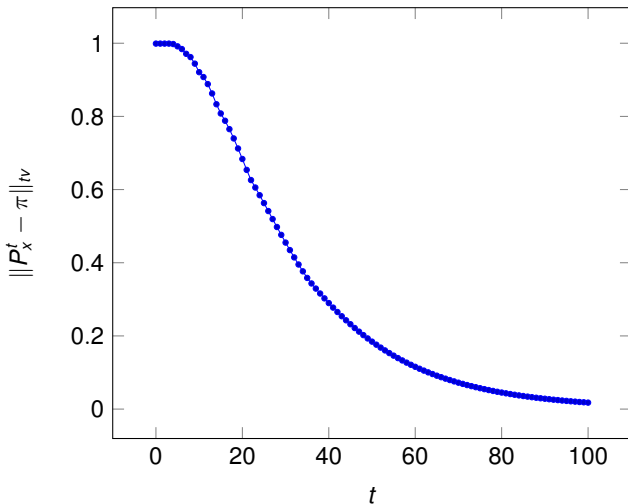
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9	1	0	0	1	0
10	done!	0	0	1	0

Once **all coordinates** have been **picked** at least once, the state is uniformly at random in $\{0, 1\}^d$.

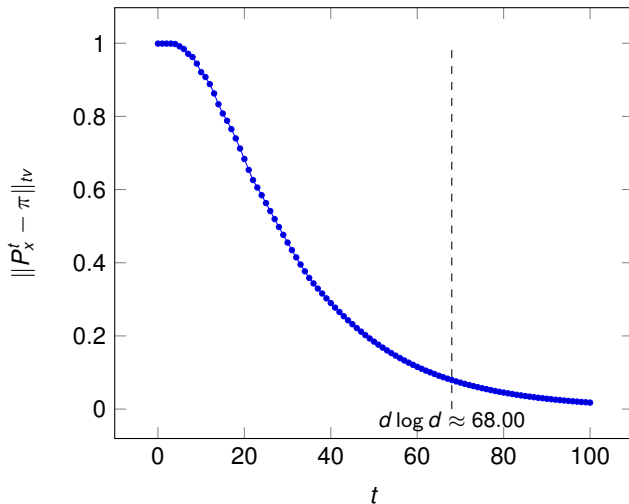
Coupon Collector \leadsto mixing time should be $O(d \log d)$

We won't formalise this argument here (see [\[Ex. 4/5.11\]](#))

Total Variation Distance of Random Walk on Hypercube ($d = 22$)



Total Variation Distance of Random Walk on Hypercube ($d = 22$)



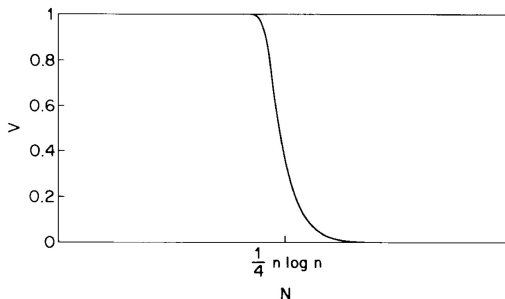


Fig. 1. The variation distance V as a function of N , for $n = 10^{12}$.

Source: "Asymptotic analysis of a random walk on a hypercube with many dimensions", P. Diaconis, R.L. Graham, J.A. Morrison; Random Structures & Algorithms, 1990.

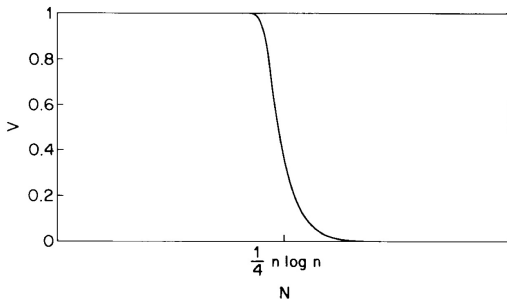


Fig. 1. The variation distance V as a function of N , for $n = 10^{12}$.

Source: "Asymptotic analysis of a random walk on a hypercube with many dimensions", P. Diaconis, R.L. Graham, J.A. Morrison; Random Structures & Algorithms, 1990.

- This is a numerical plot of a **theoretical bound**, where $d = 10^{12}$
(Minor Remark: This random walk is with a loop probability of $1/(d + 1)$)
- The variation distance exhibits a so-called **cut-off** phenomena:

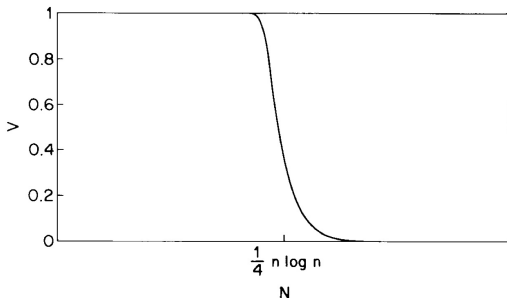


Fig. 1. The variation distance V as a function of N , for $n = 10^{12}$.

Source: "Asymptotic analysis of a random walk on a hypercube with many dimensions", P. Diaconis, R.L. Graham, J.A. Morrison; Random Structures & Algorithms, 1990.

- This is a numerical plot of a **theoretical bound**, where $d = 10^{12}$
(Minor Remark: This random walk is with a loop probability of $1/(d+1)$)
- The variation distance exhibits a so-called **cut-off** phenomena:
 - Distance remains close to its maximum value 1 until step $\frac{1}{4} n \log n - \Theta(n)$
 - Then distance moves close to 0 before step $\frac{1}{4} n \log n + \Theta(n)$

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

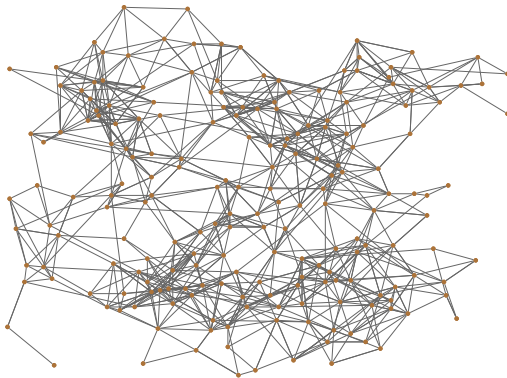
Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

Random Walks on Graphs

A **Simple Random Walk (SRW)** on a graph G is a Markov chain on $V(G)$ with

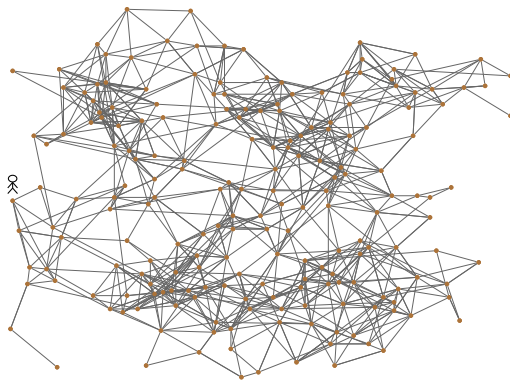
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

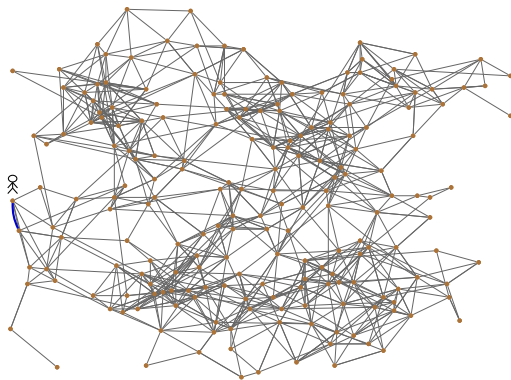
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

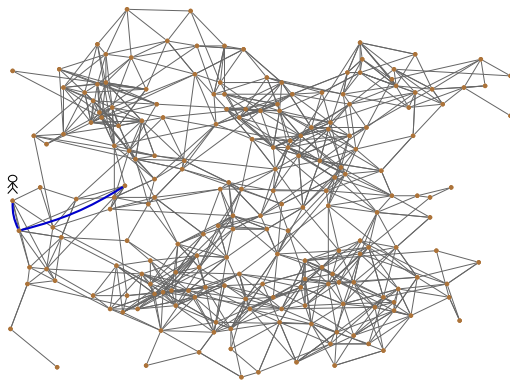
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

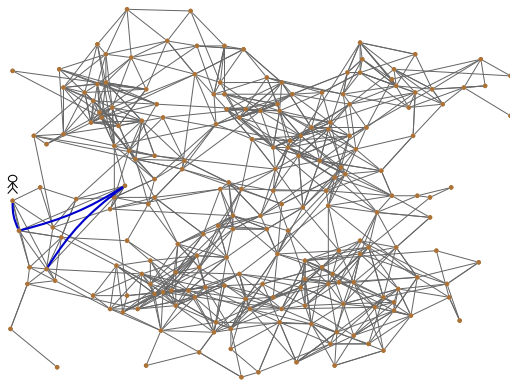
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

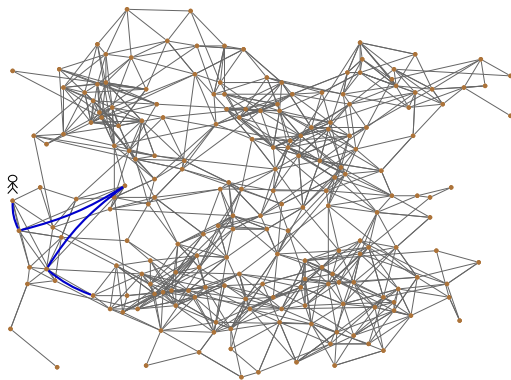
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

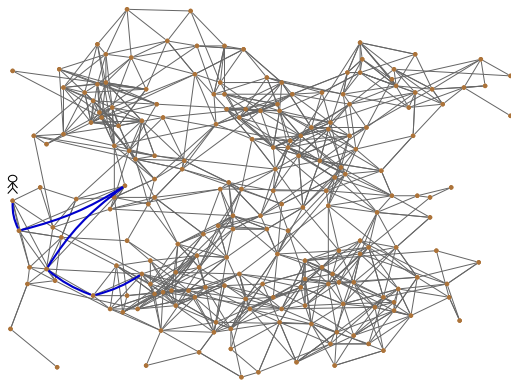
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

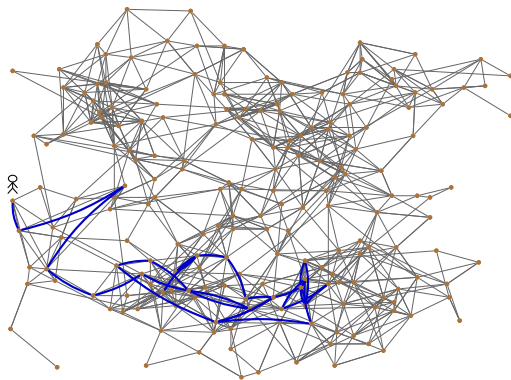
$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$



Random Walks on Graphs

A **Simple Random Walk (SRW)** on a graph G is a Markov chain on $V(G)$ with

$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$

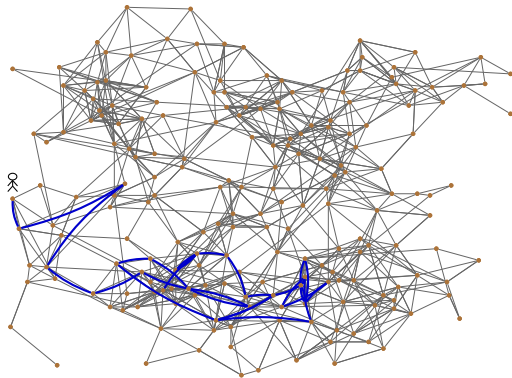


Random Walks on Graphs

A **Simple Random Walk (SRW)** on a graph G is a Markov chain on $V(G)$ with

$$P(u, v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}, \quad \text{and} \quad \pi(u) = \frac{\deg(u)}{2|E|}$$

Recall: $h(u, v) = \mathbf{E}_u[\min\{t \geq 1 : X_t = v\}]$ is the **hitting time** of v from u .



Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\tilde{P} = (P + I) / 2$,

$$\tilde{P}_{u,v} = \begin{cases} \frac{1}{2 \deg(u)} & \text{if } \{u, v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\tilde{P} = (P + I) / 2$,

$$\tilde{P}_{u,v} = \begin{cases} \frac{1}{2 \deg(u)} & \text{if } \{u, v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

P - SRW matrix
 I - Identity matrix.

Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\tilde{P} = (P + I) / 2$,

$$\tilde{P}_{u,v} = \begin{cases} \frac{1}{2 \deg(u)} & \text{if } \{u, v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

P - SRW matrix
 I - Identity matrix.

Fact: For any graph G the LRW on G is **aperiodic**.

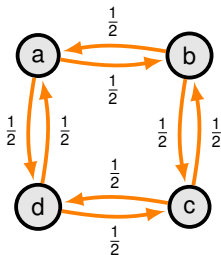
Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\tilde{P} = (P + I) / 2$,

$$\tilde{P}_{u,v} = \begin{cases} \frac{1}{2 \deg(u)} & \text{if } \{u, v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

P - SRW matrix
 I - Identity matrix.

Fact: For any graph G the LRW on G is **aperiodic**.



SRW on C_4 , *Periodic*

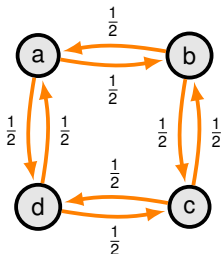
Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\tilde{P} = (P + I) / 2$,

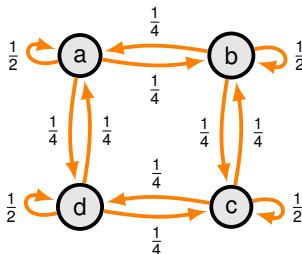
$$\tilde{P}_{u,v} = \begin{cases} \frac{1}{2 \deg(u)} & \text{if } \{u, v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

P - SRW matrix
 I - Identity matrix.

Fact: For any graph G the LRW on G is **aperiodic**.



SRW on C_4 , *Periodic*



LRW on C_4 , *Aperiodic*

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

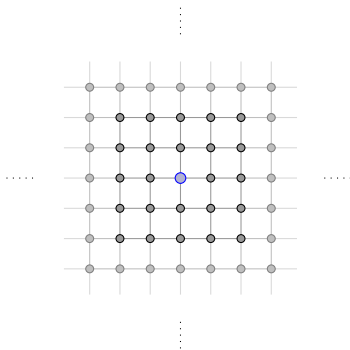
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

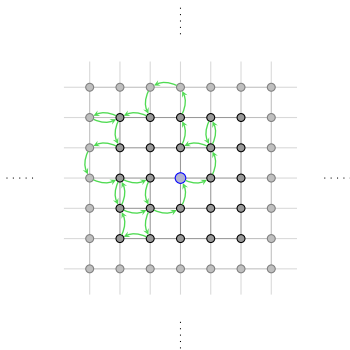
Infinite 2D Grid



1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

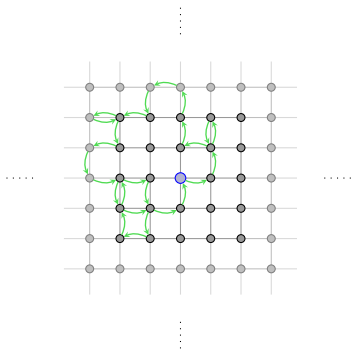
Infinite 2D Grid



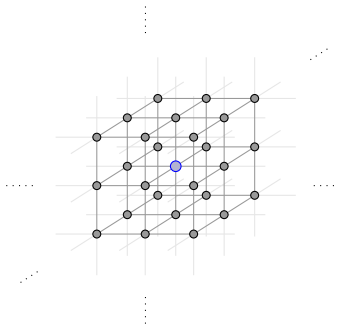
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

Infinite 2D Grid



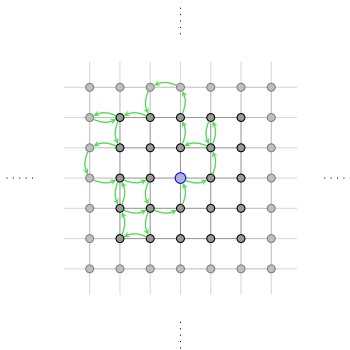
Infinite 3D Grid



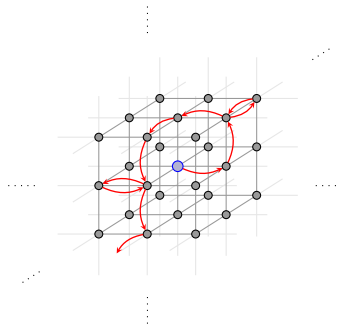
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

Infinite 2D Grid



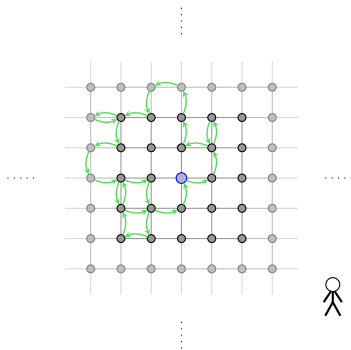
Infinite 3D Grid



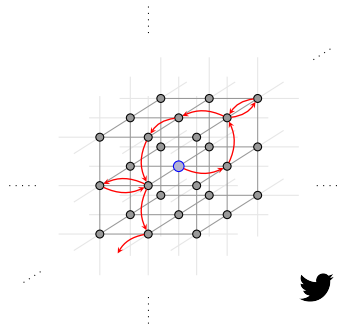
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

Infinite 2D Grid



Infinite 3D Grid

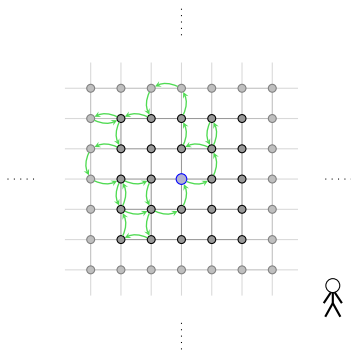


“A drunk man will find his way home, but a drunk bird may get lost forever.”

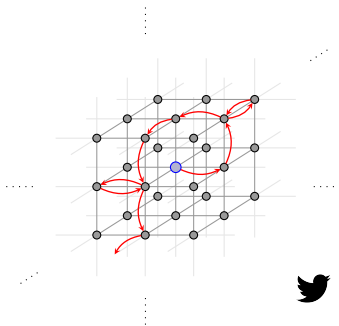
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

Infinite 2D Grid



Infinite 3D Grid



"A drunk man will find his way home, but a drunk bird may get lost forever."

But for any regular (finite) graph, the **expected return time** to u is $1/\pi(u) = n$

SRW Random Walk on Two-Dimensional Grids: Animation

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.



Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

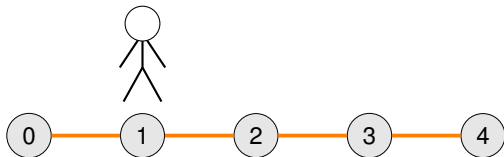


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

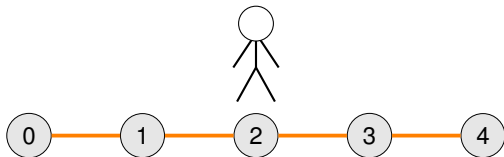


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

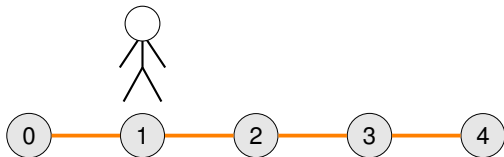


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

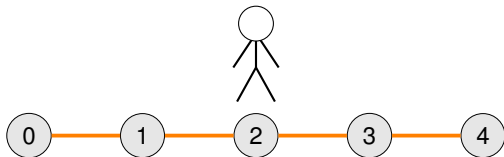


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

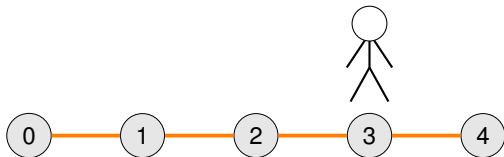


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

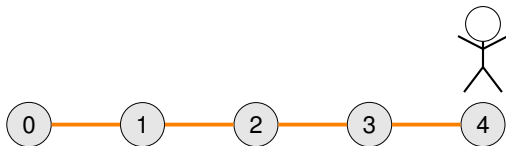


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

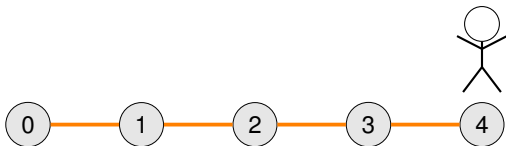


Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (1/2)

The n -path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.



Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.



Exercise: [\[Exercise 4/5.15\]](#) What happens for the LRW on P_n ?

Random Walk on a Path (2/2)

— Proposition —

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$.

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1)$$

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k-1)}{2} + \frac{f(k+1)}{2} \quad \text{for } 1 \leq k \leq n-1.$$

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k-1)}{2} + \frac{f(k+1)}{2} \quad \text{for } 1 \leq k \leq n-1.$$

System of n independent equations in n unknowns, so has a unique solution.

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k-1)}{2} + \frac{f(k+1)}{2} \quad \text{for } 1 \leq k \leq n-1.$$

System of n independent equations in n unknowns, so has a **unique** solution.

Thus it suffices to check that $f(k) = n^2 - k^2$ satisfies the above.

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k-1)}{2} + \frac{f(k+1)}{2} \quad \text{for } 1 \leq k \leq n-1.$$

System of n independent equations in n unknowns, so has a **unique** solution.

Thus it suffices to check that $f(k) = n^2 - k^2$ satisfies the above. Indeed

$$f(0) = 1 + f(1) = 1 + n^2 - 1^2 = n^2,$$

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x, y) \stackrel{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k-1)}{2} + \frac{f(k+1)}{2} \quad \text{for } 1 \leq k \leq n-1.$$

System of n independent equations in n unknowns, so has a **unique** solution.

Thus it suffices to check that $f(k) = n^2 - k^2$ satisfies the above. Indeed

$$f(0) = 1 + f(1) = 1 + n^2 - 1^2 = n^2,$$

and for any $1 \leq k \leq n-1$ we have,

$$f(k) = 1 + \frac{n^2 - (k-1)^2}{2} + \frac{n^2 - (k+1)^2}{2} = n^2 - k^2. \quad \square$$

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

SAT Problems

A **Satisfiability (SAT)** formula is a logical expression that's the conjunction (AND) of a set of **Clauses**, where a clause is the disjunction (OR) of **Literals**.

SAT Problems

A **Satisfiability (SAT)** formula is a logical expression that's the conjunction (AND) of a set of **Clauses**, where a clause is the disjunction (OR) of **Literals**.

A **Solution** to a SAT formula is an assignment of the variables to the values **True** and **False** so that all the clauses are satisfied.

SAT Problems

A **Satisfiability (SAT)** formula is a logical expression that's the conjunction (AND) of a set of **Clauses**, where a clause is the disjunction (OR) of **Literals**.

A **Solution** to a SAT formula is an assignment of the variables to the values **True** and **False** so that all the clauses are satisfied.

Example:

$$\text{SAT: } (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

SAT Problems

A **Satisfiability (SAT)** formula is a logical expression that's the conjunction (AND) of a set of **Clauses**, where a clause is the disjunction (OR) of **Literals**.

A **Solution** to a SAT formula is an assignment of the variables to the values **True** and **False** so that all the clauses are satisfied.

Example:

SAT: $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$

Solution: $x_1 = \text{True}, \quad x_2 = \text{False}, \quad x_3 = \text{False} \quad \text{and} \quad x_4 = \text{True}.$

SAT Problems

A **Satisfiability (SAT)** formula is a logical expression that's the conjunction (AND) of a set of **Clauses**, where a clause is the disjunction (OR) of **Literals**.

A **Solution** to a SAT formula is an assignment of the variables to the values **True** and **False** so that all the clauses are satisfied.

Example:

SAT: $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$

Solution: $x_1 = \text{True}, \quad x_2 = \text{False}, \quad x_3 = \text{False} \quad \text{and} \quad x_4 = \text{True}.$

- If each clause has k literals we call the problem **k -SAT**.
- In general, determining if a SAT formula has a solution is **NP-hard**
- In practice solvers are fast and used to great effect
- A huge amount of problems can be posed as a SAT:

SAT Problems

A **Satisfiability (SAT)** formula is a logical expression that's the conjunction (AND) of a set of **Clauses**, where a clause is the disjunction (OR) of **Literals**.

A **Solution** to a SAT formula is an assignment of the variables to the values **True** and **False** so that all the clauses are satisfied.

Example:

SAT: $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$

Solution: $x_1 = \text{True}, \quad x_2 = \text{False}, \quad x_3 = \text{False} \quad \text{and} \quad x_4 = \text{True}.$

- If each clause has k literals we call the problem **k -SAT**.
- In general, determining if a SAT formula has a solution is **NP-hard**
- In practice solvers are fast and used to great effect
- A huge amount of problems can be posed as a SAT:
 - Model checking and hardware/software verification
 - Design of experiments
 - Classical planning
 - ...

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
- 2: **Repeat up to $2n^2$ times**

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
- 2: **Repeat up to $2n^2$ times**
- 3: Pick an **arbitrary** unsatisfied clause

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
- 2: **Repeat up to $2n^2$ times**
- 3: Pick an arbitrary unsatisfied clause
- 4: Choose a random literal and switch its value

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
- 2: **Repeat up to $2n^2$ times**
- 3: Pick an arbitrary unsatisfied clause
- 4: Choose a random literal and switch its value
- 5: **If** formula is satisfied **then return** “Satisfiable”
- 6: **return** “Unsatisfiable”

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an arbitrary unsatisfied clause
 - 4: Choose a random literal and switch its value
 - 5: **If** formula is satisfied **then return** “Satisfiable”
 - 6: **return** “Unsatisfiable”
- Call each loop of (2) a step. Let A_i be the variable assignment at step i .

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an arbitrary unsatisfied clause
 - 4: Choose a random literal and switch its value
 - 5: **If** formula is satisfied **then return** “Satisfiable”
 - 6: **return** “Unsatisfiable”
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

2-SAT

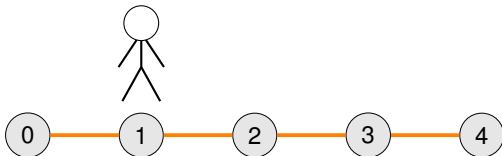
RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F F T F T



$$\alpha = (T, T, F, T).$$

t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

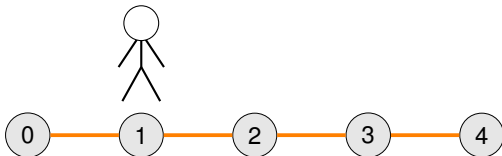
RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F F T F T



$$\alpha = (T, T, F, T).$$

t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

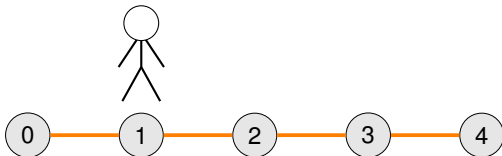
- 1: Start with an arbitrary truth assignment
- 2: **Repeat up to $2n^2$ times**
- 3: Pick an **arbitrary** unsatisfied clause
- 4: Choose a random **literal** and **switch** its value
- 5: **If** formula is satisfied **then return** "Satisfiable"
- 6: **return** "Unsatisfiable"

- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
- Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

F T T T F **F** F T F T



$$\alpha = (T, T, F, T).$$

t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

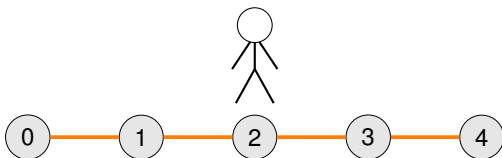
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

F F T T F T F T F T

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

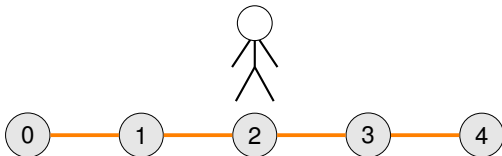
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

F F T T F T F T F T

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

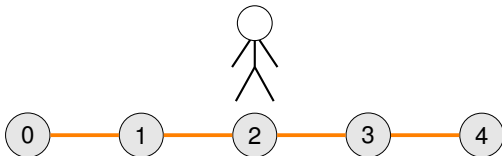
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

F F T T F T F T F T

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

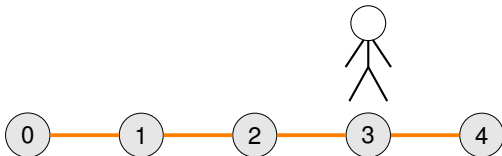
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

T F F T T T F T F F

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F
2	T	T	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

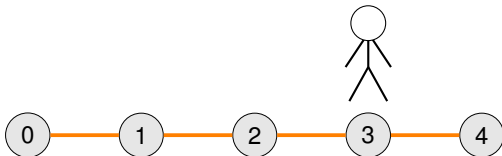
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (\overline{x_4} \vee \overline{x_1})$$

T F F T T T F T F F

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F
2	T	T	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

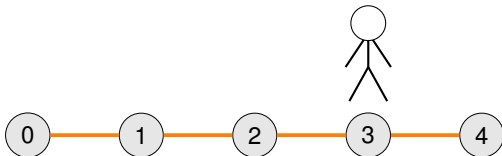
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

T F F T T T F T **F** F

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F
2	T	T	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

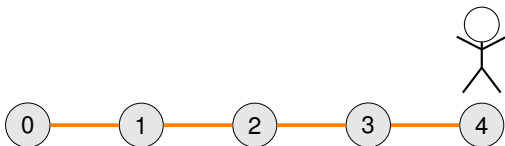
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

T F F T T T T T T F

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F
2	T	T	F	F
3	T	T	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

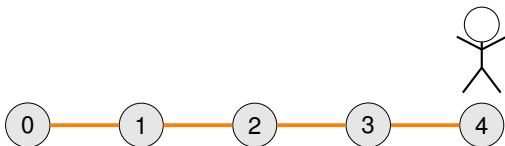
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be **any solution** and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 : Solution Found

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

T F F T T T T T T F

$$\alpha = (T, T, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F
2	T	T	F	F
3	T	T	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

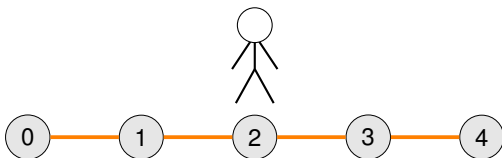
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F F F F T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

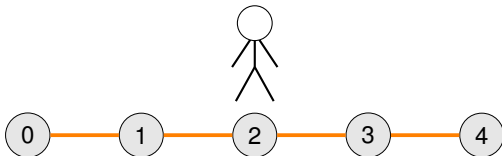
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F F F F T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

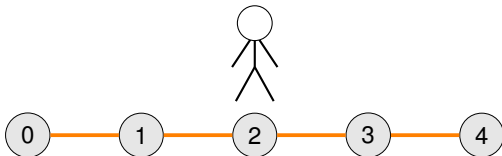
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F F F F T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

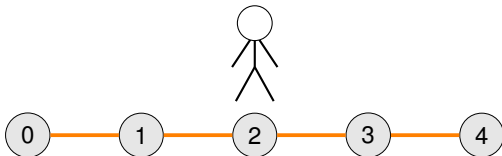
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F **F** F F T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

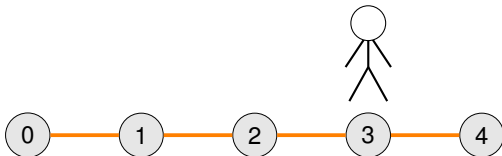
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F T F T T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
- 2: **Repeat up to $2n^2$ times**
- 3: Pick an **arbitrary** unsatisfied clause
- 4: Choose a random **literal** and **switch** its value
- 5: **If** formula is satisfied **then return** "Satisfiable"
- 6: **return** "Unsatisfiable"

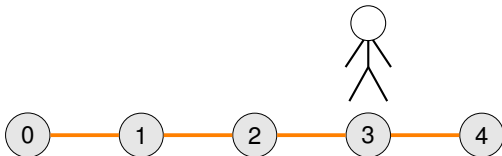
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F F T F T T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

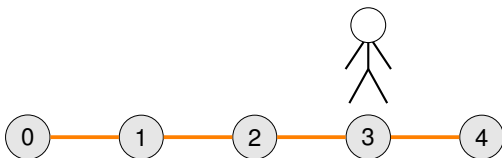
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F T T T F **F** T F T T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

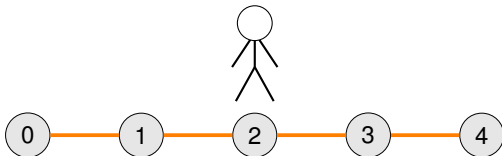
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F F T T F T T F T T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	T	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

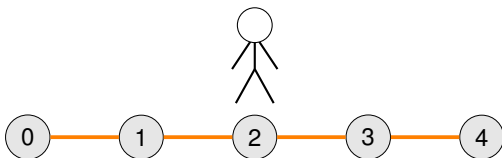
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F F T T F T T F T T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	T	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

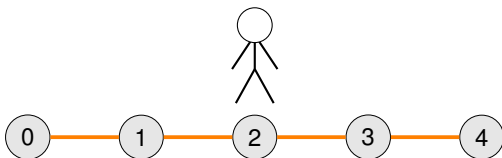
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

F F T T F T T F T T

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	T	F	T

2-SAT

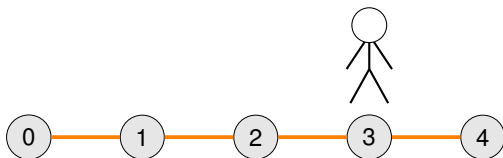
RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 :

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

T F F T T T T F T F



$$\alpha = (T, F, F, T).$$

t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	T	F	T
3	T	T	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

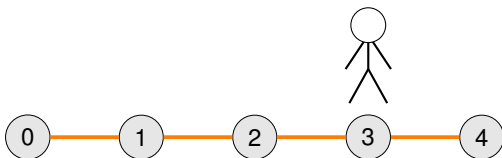
- 1: Start with an arbitrary truth assignment
 - 2: **Repeat up to $2n^2$ times**
 - 3: Pick an **arbitrary** unsatisfied clause
 - 4: Choose a random **literal** and **switch** its value
 - 5: **If** formula is satisfied **then return** "Satisfiable"
 - 6: **return** "Unsatisfiable"
- Call each loop of (2) a **step**. Let A_i be the variable assignment at step i .
 - Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 : (Another) Solution Found

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee x_3) \wedge (x_4 \vee \overline{x_1})$$

T F F T T T T F T F

$$\alpha = (T, F, F, T).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	T	F	T
3	T	T	F	T

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

(i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k + 1 \mid X_i = k] \geq 1/2$

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k + 1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k - 1 \mid X_i = k] \leq 1/2$.

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k + 1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k - 1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k + 1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k - 1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k + 1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k - 1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process X_i is complicated to describe in full; however by (i) – (iii) we can **bound** it by Y_i (SRW on the n -path from 0).

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n-1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k+1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k-1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process X_i is complicated to describe in full; however by (i) – (iii) we can **bound** it by Y_i (SRW on the n -path from 0). This gives (see also [Ex 4/5.16])

$$\mathbf{E}[\text{time to find sol}] \leq \mathbf{E}_0[\min\{t : X_t = n\}] \leq \mathbf{E}_0[\min\{t : Y_t = n\}] = h(0, n) = n^2.$$

□

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n-1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k+1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k-1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process X_i is complicated to describe in full; however by (i) – (iii) we can **bound** it by Y_i (SRW on the n -path from 0). This gives (see also [Ex 4/5.16])

$\mathbf{E}[\text{time to find sol}] \leq \mathbf{E}_0[\min\{t : X_t = n\}] \leq \mathbf{E}_0[\min\{t : Y_t = n\}] = h(0, n) = n^2$.

Running for $2n^2$ steps and using Markov's inequality yields: □

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n-1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k+1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k-1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process X_i is complicated to describe in full; however by (i) – (iii) we can **bound** it by Y_i (SRW on the n -path from 0). This gives (see also [Ex 4/5.16])

$$\mathbf{E}[\text{time to find sol}] \leq \mathbf{E}_0[\min\{t : X_t = n\}] \leq \mathbf{E}_0[\min\{t : Y_t = n\}] = h(0, n) = n^2.$$

Running for $2n^2$ steps and using Markov's inequality yields: □

Proposition

Provided a solution exists, RANDOMISED-2-SAT will return a valid solution in $O(n^2)$ steps with probability at least $1/2$.

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is **satisfiable**, then the **expected number of steps** before RANDOMISED-2-SAT outputs a valid solution is at most n^2 .

Proof: Fix any solution α , then for any $i \geq 0$ and $1 \leq k \leq n-1$,

- (i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$
- (ii) $\mathbf{P}[X_{i+1} = k+1 \mid X_i = k] \geq 1/2$
- (iii) $\mathbf{P}[X_{i+1} = k-1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus **solution** found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process X_i is complicated to describe in full; however by (i) – (iii) we can **bound** it by Y_i (SRW on the n -path from 0). This gives (see also [Ex 4/5.16])

$$\mathbf{E}[\text{time to find sol}] \leq \mathbf{E}_0[\min\{t : X_t = n\}] \leq \mathbf{E}_0[\min\{t : Y_t = n\}] = h(0, n) = n^2.$$

□



Exercise: (difficult, beyond this course) What happens to the above analysis if we apply the same algorithm to 3-SAT?

Boosting Success Probabilities

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p . Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Boosting Success Probabilities

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p . Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Proof: Recall that $1 - p \leq e^{-p}$ for all real p . Let $t = \lceil \frac{C}{p} \log n \rceil$ and observe

$$\begin{aligned} \mathbf{P}[t \text{ runs all fail}] &\leq (1 - p)^t \\ &\leq e^{-pt} \\ &\leq n^{-C}, \end{aligned}$$

thus the probability one of the runs succeeds is at least $1 - n^{-C}$. □

Boosting Success Probabilities

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p . Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Proof: Recall that $1 - p \leq e^{-p}$ for all real p . Let $t = \lceil \frac{C}{p} \log n \rceil$ and observe

$$\begin{aligned} \mathbf{P}[t \text{ runs all fail}] &\leq (1 - p)^t \\ &\leq e^{-pt} \\ &\leq n^{-C}, \end{aligned}$$

thus the probability one of the runs succeeds is at least $1 - n^{-C}$. □

RANDOMISED-2-SAT

There is a $O(n^2 \log n)$ -step algorithm for 2-SAT which succeeds w.h.p.

Randomised Algorithms

Lecture 6: Linear Programming: Introduction

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Boosting Success Probabilities (Last Lecture)

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p . Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p . Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Proof: Recall that $1 - p \leq e^{-p}$ for all real p . Let $t = \lceil \frac{C}{p} \log n \rceil$ and observe

$$\begin{aligned} \mathbf{P}[t \text{ runs all fail}] &\leq (1 - p)^t \\ &\leq e^{-pt} \\ &\leq n^{-C}, \end{aligned}$$

thus the probability one of the runs succeeds is at least $1 - n^{-C}$. □

Boosting Success Probabilities

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p . Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Proof: Recall that $1 - p \leq e^{-p}$ for all real p . Let $t = \lceil \frac{C}{p} \log n \rceil$ and observe

$$\begin{aligned} \mathbf{P}[t \text{ runs all fail}] &\leq (1 - p)^t \\ &\leq e^{-pt} \\ &\leq n^{-C}, \end{aligned}$$

thus the probability one of the runs succeeds is at least $1 - n^{-C}$. □

RANDOMISED-2-SAT

There is a $O(n^2 \log n)$ -step algorithm for 2-SAT which succeeds w.h.p.

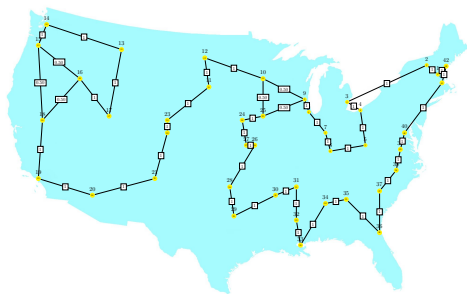
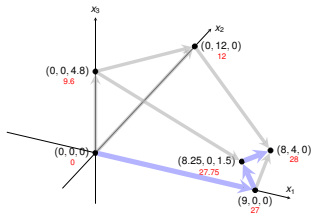
Boosting Success Probabilities (Last Lecture)

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms



- **linear programming** is a powerful tool in **optimisation**
- inspired more sophisticated techniques such as **quadratic optimisation**, **convex optimisation**, **integer programming** and **semi-definite programming**
- we will later use the connection between linear and **integer programming** to tackle several problems (Vertex-Cover, Set-Cover, TSP, satisfiability)

Outline

Boosting Success Probabilities (Last Lecture)

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms

What are Linear Programs?

Linear Programming (informal definition)

- maximise or minimise an objective, given limited resources (competing constraints)
- constraints are specified as (in)equalities
- objective function and constraints are **linear**

A Simple Example of a Linear Optimisation Problem

- Laptop

A Simple Example of a Linear Optimisation Problem

- Laptop
 - selling price to retailer: 1,000 GBP

A Simple Example of a Linear Optimisation Problem



- Laptop
 - selling price to retailer: 1,000 GBP
 - glass: 4 units

A Simple Example of a Linear Optimisation Problem



■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units

A Simple Example of a Linear Optimisation Problem



■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit

A Simple Example of a Linear Optimisation Problem



■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit

■ Smartphone

A Simple Example of a Linear Optimisation Problem



■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit

■ Smartphone

- selling price to retailer: 1,000 GBP

A Simple Example of a Linear Optimisation Problem



■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit

A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit

A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units

A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units

- You have a daily supply of:

A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units



■ You have a daily supply of:

- glass: 20 units

A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units



■ You have a daily supply of:

- glass: 20 units
- copper: 10 units



A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units



■ You have a daily supply of:

- glass: 20 units
- copper: 10 units
- rare-earth elements: 14 units



A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



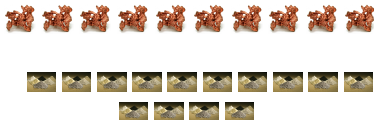
■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units



■ You have a daily supply of:

- glass: 20 units
- copper: 10 units
- rare-earth elements: 14 units
- (and enough of everything else...)



A Simple Example of a Linear Optimisation Problem

■ Laptop

- selling price to retailer: 1,000 GBP
- glass: 4 units
- copper: 2 units
- rare-earth elements: 1 unit



■ Smartphone

- selling price to retailer: 1,000 GBP
- glass: 1 unit
- copper: 1 unit
- rare-earth elements: 2 units



■ You have a daily supply of:

- glass: 20 units
- copper: 10 units
- rare-earth elements: 14 units
- (and enough of everything else...)



How to maximise your daily earnings?

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program

- Given a_1, a_2, \dots, a_n and a set of variables x_1, x_2, \dots, x_n , a **linear function** f is defined by

$$f(x_1, x_2, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n.$$

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program

- Given a_1, a_2, \dots, a_n and a set of variables x_1, x_2, \dots, x_n , a **linear function** f is defined by

$$f(x_1, x_2, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n.$$

- Linear Equality:** $f(x_1, x_2, \dots, x_n) = b$
- Linear Inequality:** $f(x_1, x_2, \dots, x_n) \begin{matrix} \geq \\ \leq \end{matrix} b$

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program

- Given a_1, a_2, \dots, a_n and a set of variables x_1, x_2, \dots, x_n , a **linear function** f is defined by

$$f(x_1, x_2, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n.$$

- Linear Equality:** $f(x_1, x_2, \dots, x_n) = b$

- Linear Inequality:** $f(x_1, x_2, \dots, x_n) \geq b$

Linear Constraints

The Linear Program

Linear Program for the Production Problem

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program

- Given a_1, a_2, \dots, a_n and a set of variables x_1, x_2, \dots, x_n , a **linear function** f is defined by

$$f(x_1, x_2, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n.$$

- Linear Equality:** $f(x_1, x_2, \dots, x_n) = b$
- Linear Inequality:** $f(x_1, x_2, \dots, x_n) \geq b$ or $f(x_1, x_2, \dots, x_n) \leq b$
- Linear-Programming Problem:** either minimise or maximise a linear function subject to a set of linear constraints

Linear Constraints

Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

Finding the Optimal Production Schedule

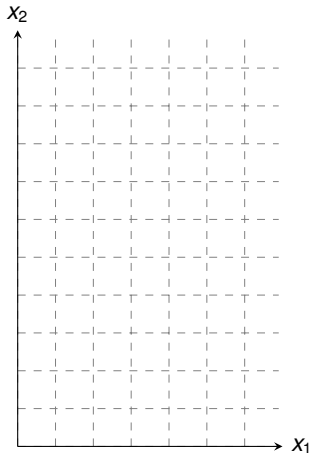
$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

Any setting of x_1 and x_2 satisfying all constraints is a feasible solution

Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

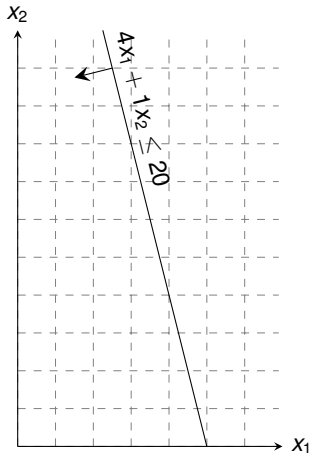
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & & x_1 & + & x_2 & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

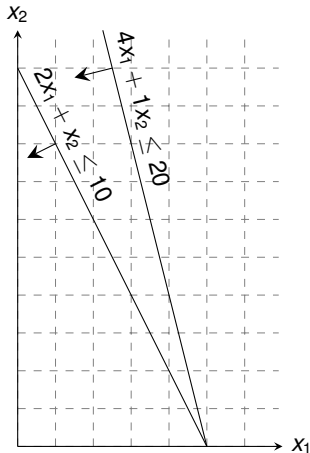
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & & x_1 & + & x_2 & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

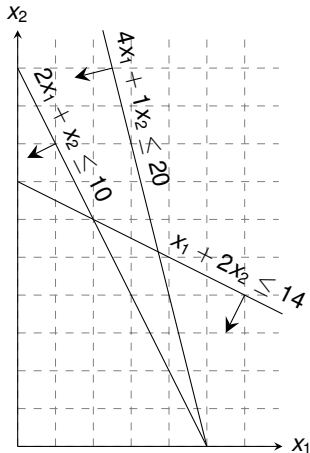
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

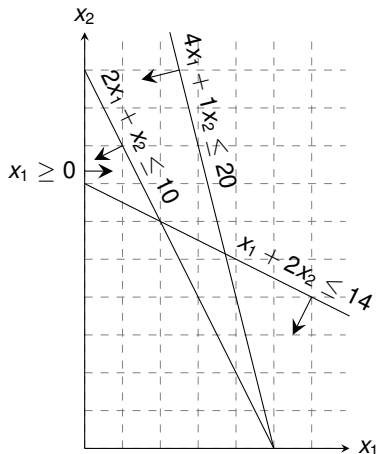
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

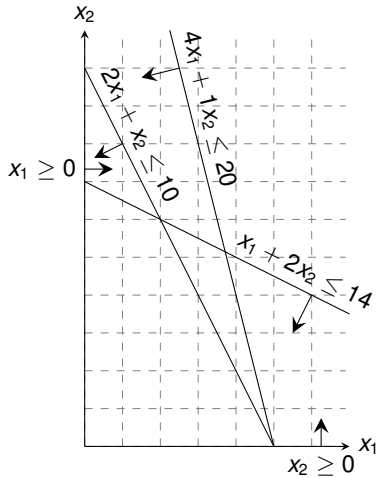
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

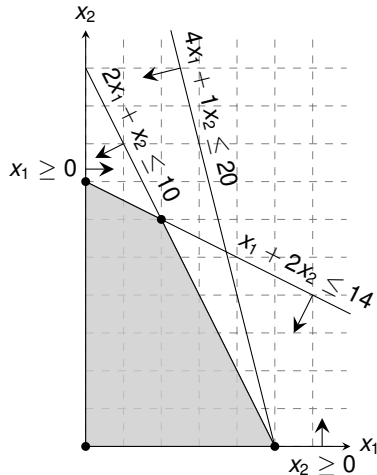
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

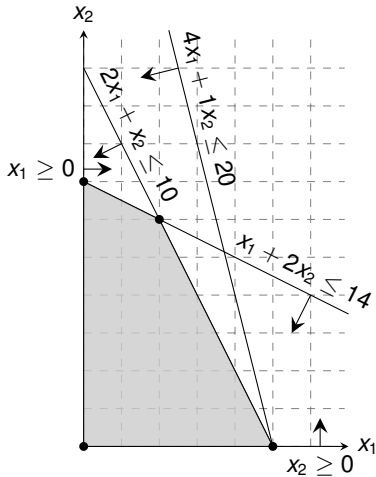
Any setting of x_1 and x_2 satisfying all constraints is a feasible solution



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

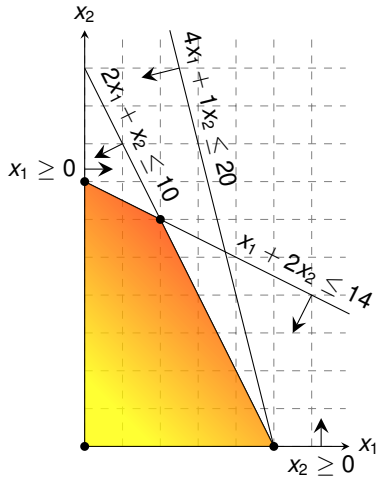
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

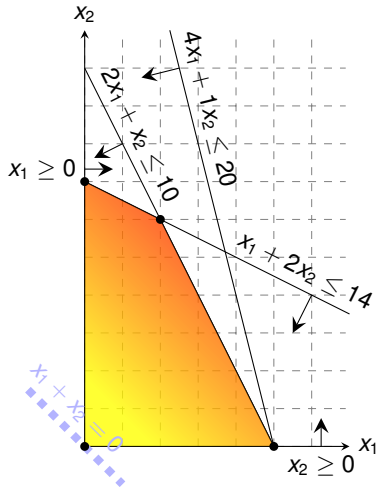
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

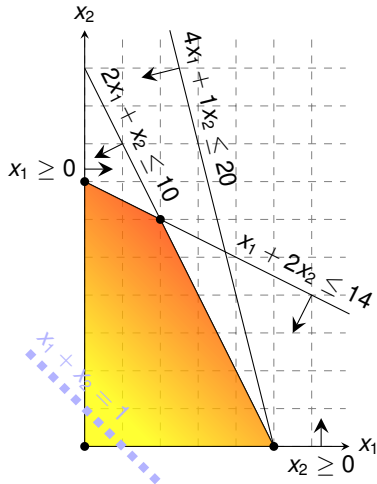
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

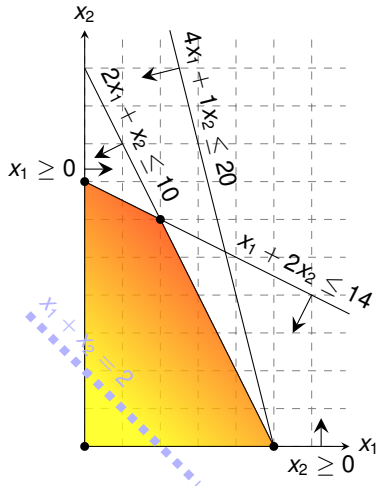
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

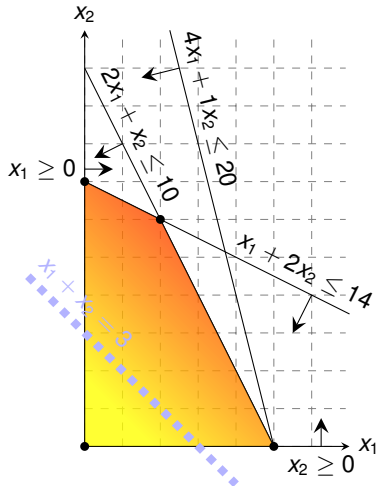
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

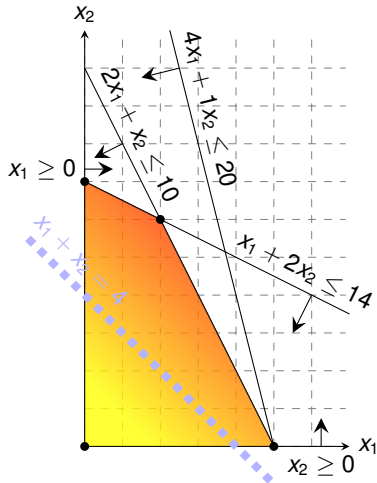
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

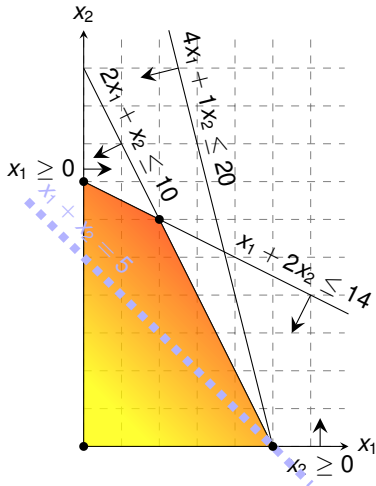
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

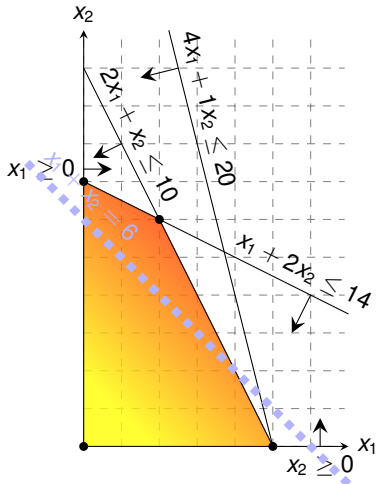
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

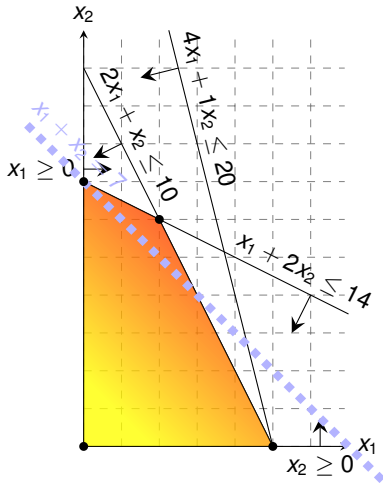
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

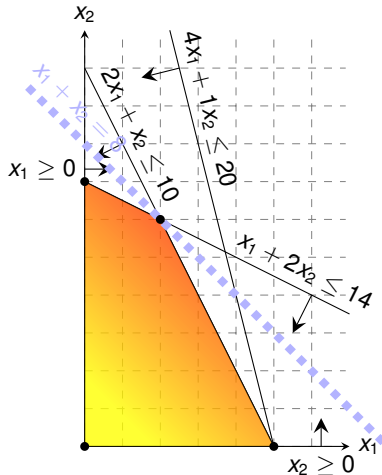
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

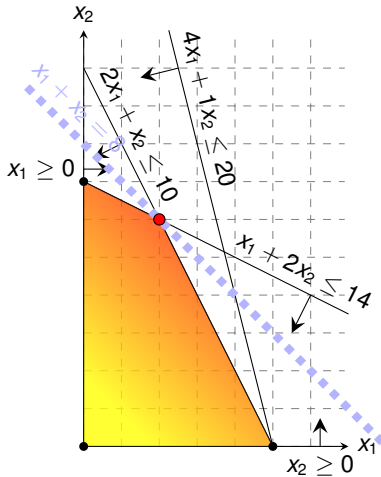
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

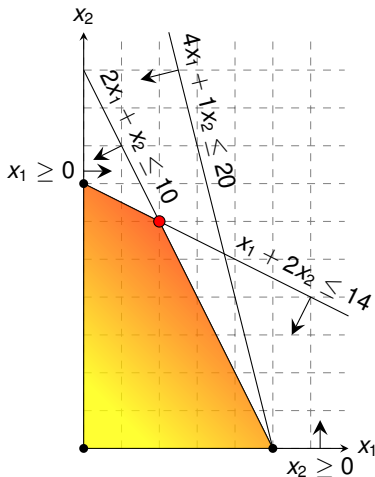
Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far as possible.

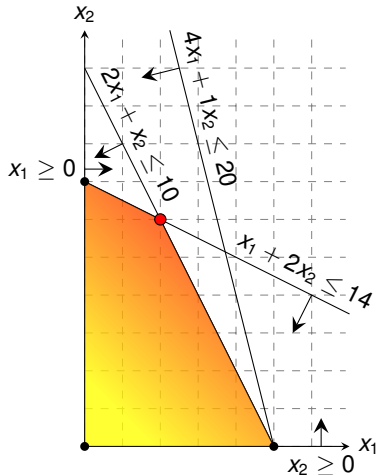


Question: Which aspect did we ignore in the formulation of the linear program?

Finding the Optimal Production Schedule

$$\begin{array}{llllll} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far as possible.



While the same approach also works for higher-dimensions, we need to take a more systematic and algebraic procedure.

Outline

Boosting Success Probabilities (Last Lecture)

Introduction

A Simple Example of a Linear Program

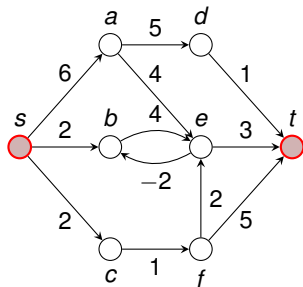
Formulating Problems as Linear Programs

Standard and Slack Forms

Shortest Paths

Single-Pair Shortest Path Problem

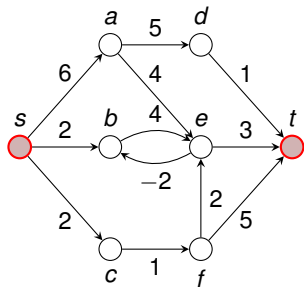
- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$



Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

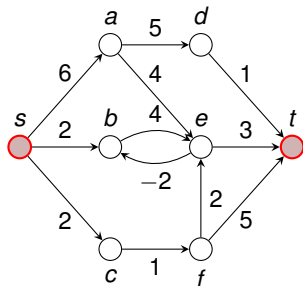


Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$ is **minimised**.

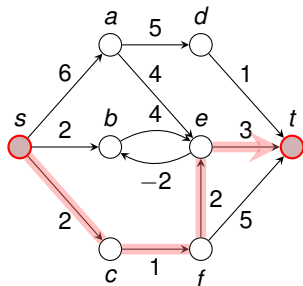


Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$ is **minimised**.

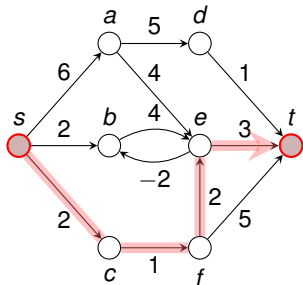


Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$ is **minimised**.



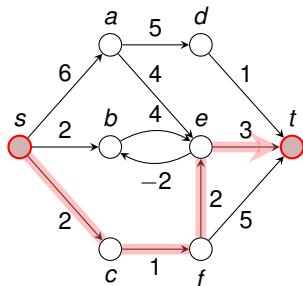
Exercise: Translate the SPSP problem into a linear program!

Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$ is **minimised**.



Shortest Paths as LP

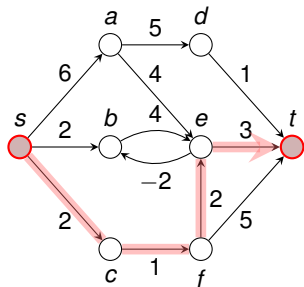
subject to

Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$ is **minimised**.



Shortest Paths as LP

subject to

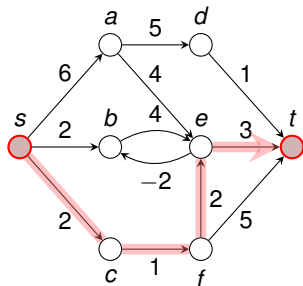
$$\begin{aligned} d_v &\leq d_u + w(u, v) && \text{for each edge } (u, v) \in E, \\ d_s &= 0. \end{aligned}$$

Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$ is **minimised**.



Shortest Paths as LP

maximise d_t

subject to

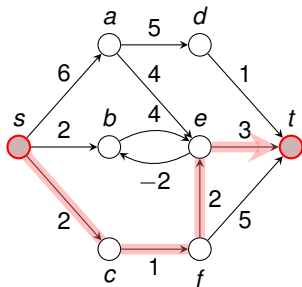
$$\begin{aligned} d_v &\leq d_u + w(u, v) && \text{for each edge } (u, v) \in E, \\ d_s &= 0. \end{aligned}$$

Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{k-1}, v_k)$ is **minimised**.



Shortest Paths as LP

maximise
subject to

$$\begin{aligned} d_v &\leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \\ d_s &= 0. \end{aligned}$$

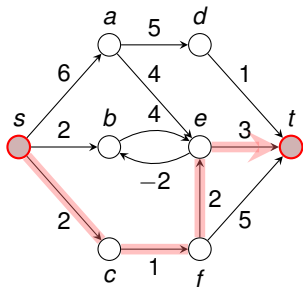
this is a **maximisation problem!**

Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{k-1}, v_k)$ is **minimised**.



Shortest Paths as LP

maximise
subject to

$$\begin{aligned} d_v &\leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \\ d_s &= 0. \end{aligned}$$

this is a **maximisation problem!**

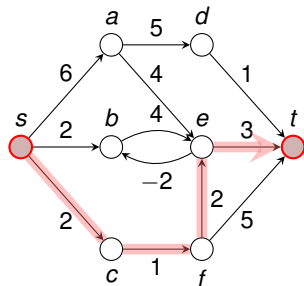
Recall: When BELLMAN-FORD terminates, all these inequalities are satisfied.

Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of **minimum weight** from s to t in G

$p = (v_0 = s, v_1, \dots, v_k = t)$ such that $w(p) = \sum_{i=1}^k w(v_{k-1}, v_k)$ is **minimised**.



Shortest Paths as LP

maximise d_t
subject to

$$d_v \leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E,$$

$$d_s = 0.$$

this is a **maximisation problem!**

Recall: When BELLMAN-FORD terminates, all these inequalities are satisfied.

Solution \bar{d} satisfies $\bar{d}_v = \min_{u: (u,v) \in E} \{\bar{d}_u + w(u, v)\}$

Maximum Flow

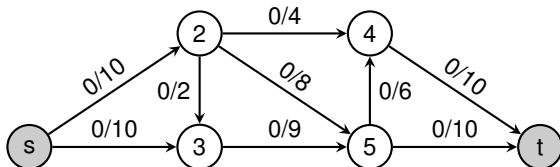
Maximum Flow Problem

- **Given:** directed graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$

Maximum Flow

Maximum Flow Problem

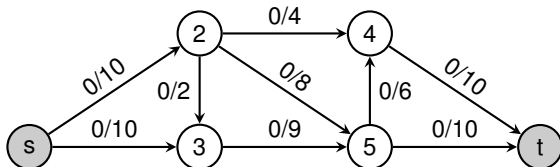
- **Given:** directed graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$



Maximum Flow

Maximum Flow Problem

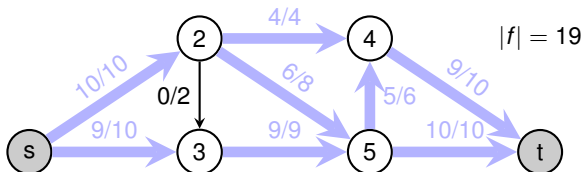
- **Given:** directed graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$
- **Goal:** Find a **maximum flow** $f : V \times V \rightarrow \mathbb{R}$ from s to t which satisfies the capacity constraints and flow conservation



Maximum Flow

Maximum Flow Problem

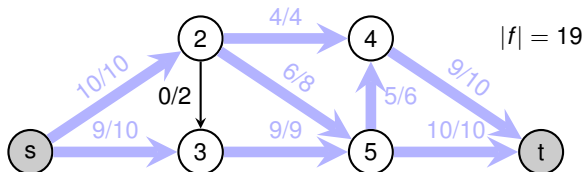
- **Given:** directed graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$
- **Goal:** Find a **maximum flow** $f : V \times V \rightarrow \mathbb{R}$ from s to t which satisfies the capacity constraints and flow conservation



Maximum Flow

Maximum Flow Problem

- **Given:** directed graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$
- **Goal:** Find a **maximum flow** $f : V \times V \rightarrow \mathbb{R}$ from s to t which satisfies the capacity constraints and flow conservation



Maximum Flow as LP

maximise
subject to

$$\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\begin{aligned} f_{uv} &\leq c(u, v) && \text{for each } u, v \in V, \\ \sum_{v \in V} f_{vu} &= \sum_{v \in V} f_{uv} && \text{for each } u \in V \setminus \{s, t\}, \\ f_{uv} &\geq 0 && \text{for each } u, v \in V. \end{aligned}$$

Minimum-Cost Flow

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem



Minimum-Cost Flow

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem

- **Given:** directed graph $G = (V, E)$ with capacities $c : E \rightarrow \mathbb{R}^+$, pair of vertices $s, t \in V$, **cost function** $a : E \rightarrow \mathbb{R}^+$, **flow demand of d units**

Minimum-Cost Flow

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem

- **Given:** directed graph $G = (V, E)$ with capacities $c : E \rightarrow \mathbb{R}^+$, pair of vertices $s, t \in V$, **cost function** $a : E \rightarrow \mathbb{R}^+$, **flow demand of d units**
- **Goal:** Find a **flow** $f : V \times V \rightarrow \mathbb{R}$ from s to t with $|f| = d$ while **minimising the total cost** $\sum_{(u,v) \in E} a(u,v)f_{uv}$ incurred by the flow.

Minimum-Cost Flow

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem

- **Given:** directed graph $G = (V, E)$ with capacities $c : E \rightarrow \mathbb{R}^+$, pair of vertices $s, t \in V$, **cost function** $a : E \rightarrow \mathbb{R}^+$, **flow demand** of d units
- **Goal:** Find a **flow** $f : V \times V \rightarrow \mathbb{R}$ from s to t with $|f| = d$ while **minimising the total cost** $\sum_{(u,v) \in E} a(u,v)f_{uv}$ incurred by the flow.

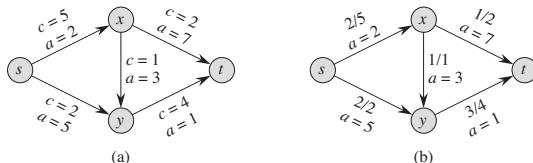


Figure 29.3 (a) An example of a minimum-cost-flow problem. We denote the capacities by c and the costs by a . Vertex s is the source and vertex t is the sink, and we wish to send 4 units of flow from s to t . (b) A solution to the minimum-cost flow problem in which 4 units of flow are sent from s to t . For each edge, the flow and capacity are written as flow/capacity.

Minimum-Cost Flow

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem

- **Given:** directed graph $G = (V, E)$ with capacities $c : E \rightarrow \mathbb{R}^+$, pair of vertices $s, t \in V$, **cost function** $a : E \rightarrow \mathbb{R}^+$, **flow demand** of d units
- **Goal:** Find a **flow** $f : V \times V \rightarrow \mathbb{R}$ from s to t with $|f| = d$ while **minimising the total cost** $\sum_{(u,v) \in E} a(u,v)f_{uv}$ incurred by the flow.

Optimal Solution with total cost:

$$\sum_{(u,v) \in E} a(u,v)f_{uv} = (2 \cdot 2) + (5 \cdot 2) + (3 \cdot 1) + (7 \cdot 1) + (1 \cdot 3) = 27$$

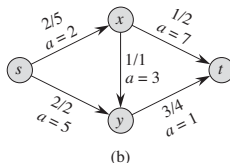
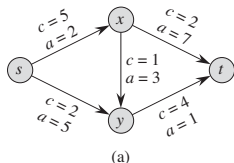


Figure 29.3 (a) An example of a minimum-cost-flow problem. We denote the capacities by c and the costs by a . Vertex s is the source and vertex t is the sink, and we wish to send 4 units of flow from s to t . (b) A solution to the minimum-cost flow problem in which 4 units of flow are sent from s to t . For each edge, the flow and capacity are written as flow/capacity.

Minimum Cost Flow as LP

minimise $\sum_{(u,v) \in E} a(u,v) f_{uv}$

subject to

$$\begin{aligned} f_{uv} &\leq c(u,v) && \text{for } u, v \in V, \\ \sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} &= 0 && \text{for } u \in V \setminus \{s, t\}, \\ \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} &= d, \\ f_{uv} &\geq 0 && \text{for } u, v \in V. \end{aligned}$$

Minimum Cost Flow as LP

minimise $\sum_{(u,v) \in E} a(u,v) f_{uv}$

subject to

$$\begin{aligned} f_{uv} &\leq c(u,v) && \text{for } u, v \in V, \\ \sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} &= 0 && \text{for } u \in V \setminus \{s, t\}, \\ \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} &= d, \\ f_{uv} &\geq 0 && \text{for } u, v \in V. \end{aligned}$$

Real power of Linear Programming comes
from the ability to solve **new problems!**

Outline

Boosting Success Probabilities (Last Lecture)

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms

Standard and Slack Forms

Standard Form

maximise $\sum_{j=1}^n c_j x_j$

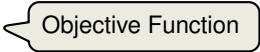
subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \text{for } i = 1, 2, \dots, m$$

$$x_j \geq 0 \quad \text{for } j = 1, 2, \dots, n$$

Standard and Slack Forms

Standard Form

maximise $\sum_{j=1}^n c_j x_j$ 

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \text{for } i = 1, 2, \dots, m$$
$$x_j \geq 0 \quad \text{for } j = 1, 2, \dots, n$$

Standard and Slack Forms

Standard Form

maximise $\sum_{j=1}^n c_j x_j$

Objective Function

subject to

$n + m$ constraints

$$\left\{ \begin{array}{ll} \sum_{j=1}^n a_{ij} x_j \leq b_i & \text{for } i = 1, 2, \dots, m \\ x_j \geq 0 & \text{for } j = 1, 2, \dots, n \end{array} \right.$$

Standard and Slack Forms

Standard Form

maximise $\sum_{j=1}^n c_j x_j$ Objective Function

subject to

$n + m$ constraints $\left\{ \begin{array}{ll} \sum_{j=1}^n a_{ij} x_j \leq b_i & \text{for } i = 1, 2, \dots, m \\ x_j \geq 0 & \text{for } j = 1, 2, \dots, n \end{array} \right.$

Non-Negativity Constraints

Standard and Slack Forms

Standard Form

maximise $\sum_{j=1}^n c_j x_j$

Objective Function

subject to

$n + m$ constraints $\left\{ \begin{array}{ll} \sum_{j=1}^n a_{ij} x_j \leq b_i & \text{for } i = 1, 2, \dots, m \\ x_j \geq 0 & \text{for } j = 1, 2, \dots, n \end{array} \right.$

Non-Negativity Constraints

Standard Form (Matrix-Vector-Notation)

maximise $c^T x$

Inner product of two vectors

subject to

$Ax \leq b$

Matrix-vector product

$x \geq 0$

Converting Linear Programs into Standard Form

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.
2. There might be variables without **nonnegativity constraints**.
3. There might be **equality constraints**.
4. There might be **inequality constraints** (with \geq instead of \leq).

Converting Linear Programs into Standard Form

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.
2. There might be variables without **nonnegativity constraints**.
3. There might be **equality constraints**.
4. There might be **inequality constraints** (with \geq instead of \leq).

Goal: Convert linear program into an **equivalent** program which is in standard form

Converting Linear Programs into Standard Form

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.
2. There might be variables without **nonnegativity constraints**.
3. There might be **equality constraints**.
4. There might be **inequality constraints** (with \geq instead of \leq).

Goal: Convert linear program into an **equivalent** program which is in standard form

Equivalence: a correspondence (not necessarily a bijection) between solutions.

Converting into Standard Form (1/5)

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.

Converting into Standard Form (1/5)

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.

minimise $-2x_1 + 3x_2$
subject to

$$\begin{array}{rclcl} x_1 & + & x_2 & = & 7 \\ x_1 & - & 2x_2 & \leq & 4 \\ x_1 & & & \geq & 0 \end{array}$$

Converting into Standard Form (1/5)

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.

$$\begin{array}{l} \text{minimise} \quad -2x_1 + 3x_2 \\ \text{subject to} \end{array}$$

$$\begin{array}{rclcl} x_1 & + & x_2 & = & 7 \\ x_1 & - & 2x_2 & \leq & 4 \\ x_1 & & & \geq & 0 \end{array}$$



Negate objective function

Converting into Standard Form (1/5)

Reasons for a LP not being in standard form:

1. The objective might be a **minimisation** rather than **maximisation**.

$$\begin{array}{llll} \text{minimise} & -2x_1 & + & 3x_2 \\ \text{subject to} & & & \end{array}$$

$$\begin{array}{rclcl} x_1 & + & x_2 & = & 7 \\ x_1 & - & 2x_2 & \leq & 4 \\ x_1 & & & \geq & 0 \end{array}$$

Negate objective function

$$\begin{array}{llll} \text{maximise} & 2x_1 & - & 3x_2 \\ \text{subject to} & & & \end{array}$$

$$\begin{array}{rclcl} x_1 & + & x_2 & = & 7 \\ x_1 & - & 2x_2 & \leq & 4 \\ x_1 & & & \geq & 0 \end{array}$$

Converting into Standard Form (2/5)

Reasons for a LP not being in standard form:

2. There might be variables without nonnegativity constraints.

Converting into Standard Form (2/5)

Reasons for a LP not being in standard form:

2. There might be variables without nonnegativity constraints.

$$\begin{array}{llllll} \text{maximise} & 2x_1 & - & 3x_2 & & \\ \text{subject to} & & & & & \\ & x_1 & + & x_2 & = & 7 \\ & x_1 & - & 2x_2 & \leq & 4 \\ & x_1 & & & \geq & 0 \end{array}$$

Converting into Standard Form (2/5)

Reasons for a LP not being in standard form:

2. There might be variables without nonnegativity constraints.

maximise
subject to

$$2x_1 - 3x_2$$

$$x_1 + x_2 = 7$$

$$x_1 - 2x_2 \leq 4$$

$$x_1 \geq 0$$



Replace x_2 by two non-negative variables x'_2 and x''_2

Converting into Standard Form (2/5)

Reasons for a LP not being in standard form:

2. There might be variables without nonnegativity constraints.

maximise
subject to

$$2x_1 - 3x_2$$

$$x_1 + x_2 = 7$$

$$x_1 - 2x_2 \leq 4$$

$$x_1 \geq 0$$



Replace x_2 by two non-negative variables x'_2 and x''_2

maximise
subject to

$$2x_1 - 3x'_2 + 3x''_2$$

$$x_1 + x'_2 - x''_2 = 7$$

$$x_1 - 2x'_2 + 2x''_2 \leq 4$$

$$x_1, x'_2, x''_2 \geq 0$$

Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

maximise
subject to

$$2x_1 - 3x'_2 + 3x''_2$$

$$x_1 + x'_2 - x''_2 = 7$$

$$x_1 - 2x'_2 + 2x''_2 \leq 4$$

$$x_1, x'_2, x''_2 \geq 0$$

Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$x_1 + x_2' - x_2'' = 7$$

$$x_1 - 2x_2' + 2x_2'' \leq 4$$

$$x_1, x_2', x_2'' \geq 0$$

↓ Replace each equality
by two inequalities.

Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$\begin{array}{rclcl} x_1 & + & x_2' & - & x_2'' & = & 7 \\ x_1 & - & 2x_2' & + & 2x_2'' & \leq & 4 \\ x_1, x_2', x_2'' & & & & & \geq & 0 \end{array}$$

Replace each equality
by two inequalities.

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$\begin{array}{rclcl} x_1 & + & x_2' & - & x_2'' & \leq & 7 \\ x_1 & + & x_2' & - & x_2'' & \geq & 7 \\ x_1 & - & 2x_2' & + & 2x_2'' & \leq & 4 \\ x_1, x_2', x_2'' & & & & & \geq & 0 \end{array}$$

Converting into Standard Form (4/5)

Reasons for a LP not being in standard form:

4. There might be inequality constraints (with \geq instead of \leq).

Converting into Standard Form (4/5)

Reasons for a LP not being in standard form:

4. There might be inequality constraints (with \geq instead of \leq).

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$x_1 + x_2' - x_2'' \leq 7$$

$$x_1 + x_2' - x_2'' \geq 7$$

$$x_1 - 2x_2' + 2x_2'' \leq 4$$

$$x_1, x_2', x_2'' \geq 0$$

Converting into Standard Form (4/5)

Reasons for a LP not being in standard form:

4. There might be **inequality constraints** (with \geq instead of \leq).

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$x_1 + x_2' - x_2'' \leq 7$$

$$x_1 + x_2' - x_2'' \geq 7$$

$$x_1 - 2x_2' + 2x_2'' \leq 4$$

$$x_1, x_2', x_2'' \geq 0$$

Negate respective inequalities.



Converting into Standard Form (4/5)

Reasons for a LP not being in standard form:

4. There might be **inequality constraints** (with \geq instead of \leq).

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$x_1 + x_2' - x_2'' \leq 7$$

$$x_1 + x_2' - x_2'' \geq 7$$

$$x_1 - 2x_2' + 2x_2'' \leq 4$$

$$x_1, x_2', x_2'' \geq 0$$

Negate respective inequalities.

maximise
subject to

$$2x_1 - 3x_2' + 3x_2''$$

$$x_1 + x_2' - x_2'' \leq 7$$

$$-x_1 - x_2' + x_2'' \leq -7$$

$$x_1 - 2x_2' + 2x_2'' \leq 4$$

$$x_1, x_2', x_2'' \geq 0$$

Converting into Standard Form (5/5)

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$x_1 + x_2 - x_3 \leq 7$$

$$-x_1 - x_2 + x_3 \leq -7$$

$$x_1 - 2x_2 + 2x_3 \leq 4$$

$$x_1, x_2, x_3 \geq 0$$

Converting into Standard Form (5/5)

Rename variable names (for consistency).

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$\begin{array}{rcccccccl} x_1 & + & x_2 & - & x_3 & \leq & 7 \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\ x_1, x_2, x_3 & & & & & \geq & 0 \end{array}$$

Converting into Standard Form (5/5)

Rename variable names (for consistency).

maximise
subject to

$$\begin{array}{rcccccccl} 2x_1 & - & 3x_2 & + & 3x_3 & & & \\ x_1 & + & x_2 & - & x_3 & \leq & 7 & \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 & \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 & \\ x_1, x_2, x_3 & & & & & \geq & 0 & \end{array}$$

It is always possible to convert a linear program into standard form.

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^n a_{ij}x_j \leq b_i$ be an inequality constraint

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^n a_{ij}x_j \leq b_i$ be an inequality constraint
- Introduce a **slack variable** s by

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^n a_{ij}x_j \leq b_i$ be an inequality constraint
- Introduce a **slack variable** s by

$$s = b_i - \sum_{j=1}^n a_{ij}x_j$$

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^n a_{ij}x_j \leq b_i$ be an inequality constraint
- Introduce a **slack variable** s by

$$s = b_i - \sum_{j=1}^n a_{ij}x_j$$

$$s \geq 0.$$

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^n a_{ij}x_j \leq b_i$ be an inequality constraint
- Introduce a **slack variable** s by

s measures the slack between the two sides of the inequality.

$$s = b_i - \sum_{j=1}^n a_{ij}x_j$$
$$s \geq 0.$$

Converting Standard Form into Slack Form (1/3)

Goal: Convert **standard form** into **slack form**, where all constraints except for the non-negativity constraints are equalities.

For the **simplex algorithm**, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^n a_{ij}x_j \leq b_i$ be an inequality constraint
- Introduce a **slack variable** s by

s measures the slack between the two sides of the inequality.

$$s = b_i - \sum_{j=1}^n a_{ij}x_j$$
$$s \geq 0.$$

- Denote slack variable of the i -th inequality by x_{n+i}

Converting Standard Form into Slack Form (2/3)

$$\begin{array}{llllllll} \text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & \\ \text{subject to} & & & & & & & \\ & x_1 & + & x_2 & - & x_3 & \leq & 7 \\ & -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\ & x_1, x_2, x_3 & & & & & \geq & 0 \end{array}$$

Converting Standard Form into Slack Form (2/3)

maximise
subject to

$$\begin{array}{ccccccc} 2x_1 & - & 3x_2 & + & 3x_3 & & \\ x_1 & + & x_2 & - & x_3 & \leq & 7 \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\ & & & & & \geq & 0 \end{array}$$

x_1, x_2, x_3

Introduce slack variables



Converting Standard Form into Slack Form (2/3)

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$\begin{array}{rclclcl} x_1 & + & x_2 & - & x_3 & \leq & 7 \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\ x_1, x_2, x_3 & & & & & \geq & 0 \end{array}$$

Introduce slack variables

subject to

$$x_4 = 7 - x_1 - x_2 + x_3$$

Converting Standard Form into Slack Form (2/3)

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$\begin{array}{rcccccccl} x_1 & + & x_2 & - & x_3 & \leq & 7 \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\ x_1, x_2, x_3 & & & & & \geq & 0 \end{array}$$

Introduce slack variables



subject to

$$\begin{array}{rcccccccl} x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \end{array}$$

Converting Standard Form into Slack Form (2/3)

maximise
subject to

$$\begin{array}{rcccccccl} 2x_1 & - & 3x_2 & + & 3x_3 & & & \\ x_1 & + & x_2 & - & x_3 & \leq & 7 & \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 & \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 & \\ x_1, x_2, x_3 & & & & & \geq & 0 & \end{array}$$

Introduce slack variables



subject to

$$\begin{array}{rcccccccl} x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \end{array}$$

Converting Standard Form into Slack Form (2/3)

maximise
subject to

$$\begin{array}{rcccccccl} 2x_1 & - & 3x_2 & + & 3x_3 & & & \\ x_1 & + & x_2 & - & x_3 & \leq & 7 & \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 & \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 & \\ x_1, x_2, x_3 & & & & & \geq & 0 & \end{array}$$

Introduce slack variables



subject to

$$\begin{array}{rcccccccl} x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 & \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 & \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 & \\ x_1, x_2, x_3, x_4, x_5, x_6 & & & & & & & \geq & 0 & \end{array}$$

Converting Standard Form into Slack Form (2/3)

$$\begin{array}{llllll} \text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 \\ \text{subject to} & & & & & \\ & x_1 & + & x_2 & - & x_3 & \leq & 7 \\ & -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\ & x_1, x_2, x_3 & & & & & \geq & 0 \end{array}$$

Introduce slack variables

$$\begin{array}{llllllllll} \text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & & & \\ \text{subject to} & & & & & & & & & \\ & x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ & x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ & x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \\ & x_1, x_2, x_3, x_4, x_5, x_6 & & & & & & \geq & 0 \end{array}$$

Converting Standard Form into Slack Form (3/3)

$$\begin{array}{llllllllll} \text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & & & \\ \text{subject to} & & & & & & & & & \\ & x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ & x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ & x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \\ & x_1, x_2, x_3, x_4, x_5, x_6 & & & & & \geq & 0 & & \end{array}$$

Converting Standard Form into Slack Form (3/3)

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$x_4 = 7 - x_1 - x_2 + x_3$$

$$x_5 = -7 + x_1 + x_2 - x_3$$

$$x_6 = 4 - x_1 + 2x_2 - 2x_3$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$



Use variable z to denote objective function
and omit the nonnegativity constraints.

Converting Standard Form into Slack Form (3/3)

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$x_4 = 7 - x_1 - x_2 + x_3$$

$$x_5 = -7 + x_1 + x_2 - x_3$$

$$x_6 = 4 - x_1 + 2x_2 - 2x_3$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$



Use variable z to denote objective function
and omit the nonnegativity constraints.

z	$=$		$2x_1$	$-$	$3x_2$	$+$	$3x_3$	
x_4	$=$	7	$-$	x_1	$-$	x_2	$+$	x_3
x_5	$=$	-7	$+$	x_1	$+$	x_2	$-$	x_3
x_6	$=$	4	$-$	x_1	$+$	$2x_2$	$-$	$2x_3$

Converting Standard Form into Slack Form (3/3)

maximise
subject to

$$2x_1 - 3x_2 + 3x_3$$

$$x_4 = 7 - x_1 - x_2 + x_3$$

$$x_5 = -7 + x_1 + x_2 - x_3$$

$$x_6 = 4 - x_1 + 2x_2 - 2x_3$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$

Use variable z to denote objective function
and omit the nonnegativity constraints.

$$\begin{array}{rcll} z & = & 2x_1 - 3x_2 + 3x_3 \\ x_4 & = & 7 - x_1 - x_2 + x_3 \\ x_5 & = & -7 + x_1 + x_2 - x_3 \\ x_6 & = & 4 - x_1 + 2x_2 - 2x_3 \end{array}$$

This is called **slack form**.

Basic and Non-Basic Variables

$$\begin{array}{rclclclcl} Z & = & & & 2x_1 & - & 3x_2 & + & 3x_3 \\ x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \end{array}$$

Basic and Non-Basic Variables

$$\begin{array}{rclclclcl} Z & = & & & 2x_1 & - & 3x_2 & + & 3x_3 \\ x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \end{array}$$

Basic Variables: $B = \{4, 5, 6\}$

Basic and Non-Basic Variables

$$\begin{array}{rclclclcl} Z & = & & & 2x_1 & - & 3x_2 & + & 3x_3 \\ x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \end{array}$$

Basic Variables: $B = \{4, 5, 6\}$

Non-Basic Variables: $N = \{1, 2, 3\}$

Basic and Non-Basic Variables

$$\begin{array}{rclclclcl} z & = & & 2x_1 & - & 3x_2 & + & 3x_3 \\ x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \end{array}$$

Basic Variables: $B = \{4, 5, 6\}$

Non-Basic Variables: $N = \{1, 2, 3\}$

Slack Form (Formal Definition)

Slack form is given by a tuple (N, B, A, b, c, v) so that

$$\begin{aligned} z &= v + \sum_{j \in N} c_j x_j \\ x_i &= b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B, \end{aligned}$$

and all variables are non-negative.

Basic and Non-Basic Variables

$$\begin{array}{rclclclcl} z & = & & 2x_1 & - & 3x_2 & + & 3x_3 \\ x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\ x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\ x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \end{array}$$

Basic Variables: $B = \{4, 5, 6\}$

Non-Basic Variables: $N = \{1, 2, 3\}$

Slack Form (Formal Definition)

Slack form is given by a tuple (N, B, A, b, c, v) so that

$$\begin{aligned} z &= v + \sum_{j \in N} c_j x_j \\ x_i &= b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B, \end{aligned}$$

and all variables are non-negative.

Variables/Coefficients on the right hand side are indexed by B and N .

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form Notation

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form Notation

- $B = \{1, 2, 4\}$, $N = \{3, 5, 6\}$

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form Notation

- $B = \{1, 2, 4\}, N = \{3, 5, 6\}$

-

$$A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix}$$

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form Notation

- $B = \{1, 2, 4\}, N = \{3, 5, 6\}$

-

$$A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix}$$

-

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix},$$

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form Notation

- $B = \{1, 2, 4\}, N = \{3, 5, 6\}$

-

$$A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix}$$

-

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix}, \quad c = \begin{pmatrix} c_3 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} -1/6 \\ -1/6 \\ -2/3 \end{pmatrix}$$

Slack Form (Example)

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Slack Form Notation

- $B = \{1, 2, 4\}, N = \{3, 5, 6\}$

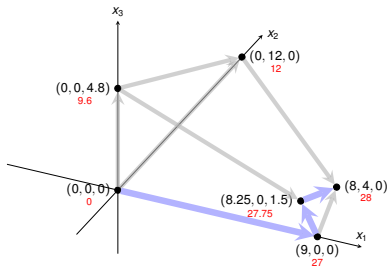
-

$$A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix}$$

-

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix}, \quad c = \begin{pmatrix} c_3 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} -1/6 \\ -1/6 \\ -2/3 \end{pmatrix}$$

- $v = 28$



Randomised Algorithms

Lecture 7: Linear Programming: Simplex Algorithm

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)

Simplex Algorithm: Introduction

Simplex Algorithm

- classical method for solving linear programs (Dantzig, 1947)
- usually fast in practice although worst-case runtime not polynomial
- iterative procedure somewhat similar to Gaussian elimination

Simplex Algorithm

- classical method for solving linear programs (Dantzig, 1947)
- usually fast in practice although worst-case runtime not polynomial
- iterative procedure somewhat similar to Gaussian elimination

Basic Idea:

- Each iteration corresponds to a “basic solution” of the slack form
- All non-basic variables are 0, and the basic variables are determined from the equality constraints
- Each iteration converts one slack form into an equivalent one while the objective value will not decrease
- Conversion (“pivoting”) is achieved by switching the roles of one basic and one non-basic variable

Simplex Algorithm: Introduction

Simplex Algorithm

- classical method for solving linear programs (Dantzig, 1947)
- usually fast in practice although worst-case runtime not polynomial
- iterative procedure somewhat similar to Gaussian elimination

Basic Idea:

- Each iteration corresponds to a “basic solution” of the slack form
- All non-basic variables are 0, and the basic variables are determined from the equality constraints
- Each iteration converts one slack form into an equivalent one while the objective value will not decrease
- Conversion (“pivoting”) is achieved by switching the roles of one basic and one non-basic variable

In that sense, it is a **greedy algorithm**.

Extended Example: Conversion into Slack Form

$$\begin{array}{llllllll} \text{maximise} & 3x_1 & + & x_2 & + & 2x_3 & & \\ \text{subject to} & & & & & & & \\ & x_1 & + & x_2 & + & 3x_3 & \leq & 30 \\ & 2x_1 & + & 2x_2 & + & 5x_3 & \leq & 24 \\ & 4x_1 & + & x_2 & + & 2x_3 & \leq & 36 \\ & & & x_1, x_2, x_3 & & & \geq & 0 \end{array}$$

Extended Example: Conversion into Slack Form


$$\begin{array}{llllllll} \text{maximise} & 3x_1 & + & x_2 & + & 2x_3 & & \\ \text{subject to} & & & & & & & \\ & x_1 & + & x_2 & + & 3x_3 & \leq & 30 \\ & 2x_1 & + & 2x_2 & + & 5x_3 & \leq & 24 \\ & 4x_1 & + & x_2 & + & 2x_3 & \leq & 36 \\ & & & x_1, x_2, x_3 & & & \geq & 0 \end{array}$$

↓
Conversion into slack form

Extended Example: Conversion into Slack Form

$$\begin{array}{llllll} \text{maximise} & 3x_1 & + & x_2 & + & 2x_3 \\ \text{subject to} & & & & & \\ & x_1 & + & x_2 & + & 3x_3 & \leq & 30 \\ & 2x_1 & + & 2x_2 & + & 5x_3 & \leq & 24 \\ & 4x_1 & + & x_2 & + & 2x_3 & \leq & 36 \\ & & & x_1, x_2, x_3 & & & \geq & 0 \end{array}$$

Conversion into slack form


$$\begin{array}{rcll} Z & = & & 3x_1 + x_2 + 2x_3 \\ x_4 & = & 30 & - x_1 - x_2 - 3x_3 \\ x_5 & = & 24 & - 2x_1 - 2x_2 - 5x_3 \\ x_6 & = & 36 & - 4x_1 - x_2 - 2x_3 \end{array}$$

Extended Example: Iteration 1

$$Z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

Extended Example: Iteration 1

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

Basic solution: $(\overline{x_1}, \overline{x_2}, \dots, \overline{x_6}) = (0, 0, 0, 30, 24, 36)$

Extended Example: Iteration 1

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (0, 0, 0, 30, 24, 36)$

This basic solution is **feasible**

Extended Example: Iteration 1

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

Basic solution: $(\overline{x_1}, \overline{x_2}, \dots, \overline{x_6}) = (0, 0, 0, 30, 24, 36)$

This basic solution is **feasible**

Objective value is 0.

Extended Example: Iteration 1

Increasing the value of x_1 would increase the objective value.

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (0, 0, 0, 30, 24, 36)$

This basic solution is **feasible**

Objective value is 0.

Extended Example: Iteration 1

Increasing the value of x_1 would increase the objective value.

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

The third constraint is the tightest and limits how much we can increase x_1 .

Extended Example: Iteration 1

Increasing the value of x_1 would increase the objective value.

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

The third constraint is the tightest and limits how much we can increase x_1 .

Switch roles of x_1 and x_6 :

Extended Example: Iteration 1

Increasing the value of x_1 would increase the objective value.

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

The third constraint is the tightest and limits how much we can increase x_1 .

Switch roles of x_1 and x_6 :

- Solving for x_1 yields:

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}.$$

Extended Example: Iteration 1

Increasing the value of x_1 would increase the objective value.

$$z = 3x_1 + x_2 + 2x_3$$

$$x_4 = 30 - x_1 - x_2 - 3x_3$$

$$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

The third constraint is the tightest and limits how much we can increase x_1 .

Switch roles of x_1 and x_6 :

- Solving for x_1 yields:

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}.$$

- Substitute this into x_1 in the other three equations

Extended Example: Iteration 2

$$\begin{array}{rclclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

Extended Example: Iteration 2

$$\begin{array}{rclclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (9, 0, 0, 21, 6, 0)$ with objective value 27

Extended Example: Iteration 2

Increasing the value of x_3 would increase the objective value.

$$\begin{array}{rclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (9, 0, 0, 21, 6, 0)$ with objective value 27

Extended Example: Iteration 2

Increasing the value of x_3 would increase the objective value.

$$\begin{array}{rclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

The third constraint is the tightest and limits how much we can increase x_3 .

Extended Example: Iteration 2

Increasing the value of x_3 would increase the objective value.

$$\begin{array}{rclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

The third constraint is the tightest and limits how much we can increase x_3 .

Switch roles of x_3 and x_5 :

Extended Example: Iteration 2

Increasing the value of x_3 would increase the objective value.

$$\begin{array}{rclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

The third constraint is the tightest and limits how much we can increase x_3 .

Switch roles of x_3 and x_5 :

- Solving for x_3 yields:

$$x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} - \frac{x_6}{8}.$$

Extended Example: Iteration 2

Increasing the value of x_3 would increase the objective value.

$$\begin{array}{rclclcl} z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\ x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\ x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\ x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} \end{array}$$

The third constraint is the tightest and limits how much we can increase x_3 .

Switch roles of x_3 and x_5 :

- Solving for x_3 yields:

$$x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} - \frac{x_6}{8}.$$

- Substitute this into x_3 in the other three equations

Extended Example: Iteration 3

$$\begin{aligned} z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \end{aligned}$$

Extended Example: Iteration 3

$$\begin{aligned} z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \end{aligned}$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (\frac{33}{4}, 0, \frac{3}{2}, \frac{69}{4}, 0, 0)$ with objective value $\frac{111}{4} = 27.75$

Extended Example: Iteration 3

Increasing the value of x_2 would increase the objective value.

$$\begin{aligned} z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \end{aligned}$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (\frac{33}{4}, 0, \frac{3}{2}, \frac{69}{4}, 0, 0)$ with objective value $\frac{111}{4} = 27.75$

Extended Example: Iteration 3

Increasing the value of x_2 would increase the objective value.

$$\begin{aligned} z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \end{aligned}$$

The second constraint is the tightest and limits how much we can increase x_2 .

Extended Example: Iteration 3

Increasing the value of x_2 would increase the objective value.

$$\begin{aligned} z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \end{aligned}$$

The second constraint is the tightest and limits how much we can increase x_2 .

Switch roles of x_2 and x_3 :

Extended Example: Iteration 3

Increasing the value of x_2 would increase the objective value.

$$\begin{aligned} z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \end{aligned}$$

The second constraint is the tightest and limits how much we can increase x_2 .

Switch roles of x_2 and x_3 :

- Solving for x_2 yields:

$$x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3}.$$

Extended Example: Iteration 3

Increasing the value of x_2 would increase the objective value.

$$\begin{aligned}z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}\end{aligned}$$

The second constraint is the tightest and limits how much we can increase x_2 .

Switch roles of x_2 and x_3 :

- Solving for x_2 yields:

$$x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3}.$$

- Substitute this into x_2 in the other three equations

Extended Example: Iteration 4

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Extended Example: Iteration 4

$$\begin{array}{rclclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (8, 4, 0, 18, 0, 0)$ with objective value 28

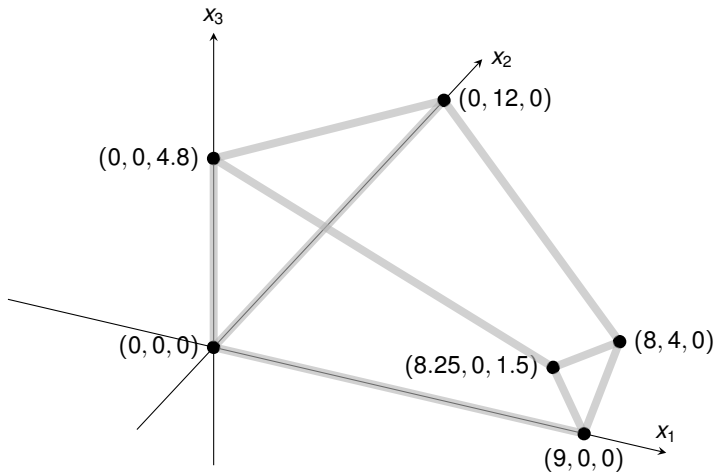
Extended Example: Iteration 4

All coefficients are negative, and hence this basic solution is **optimal!**

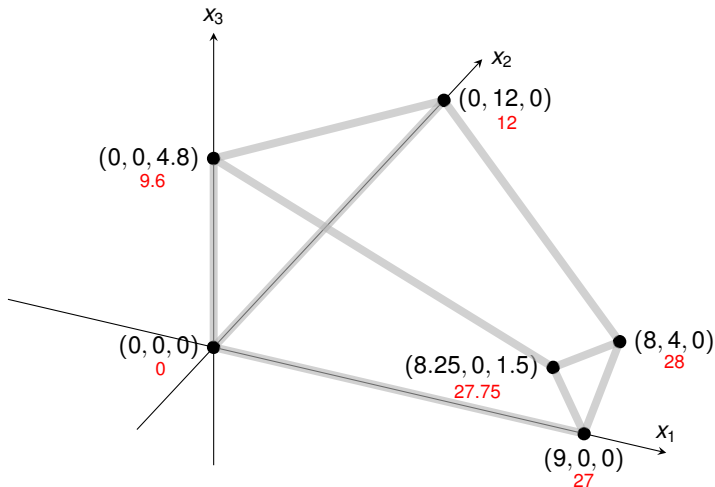
$$\begin{array}{rclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Basic solution: $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_6) = (8, 4, 0, 18, 0, 0)$ with objective value 28

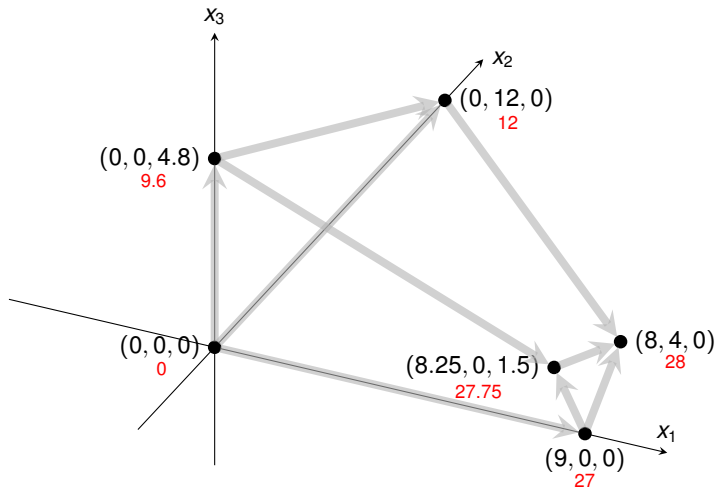
Extended Example: Visualization of SIMPLEX



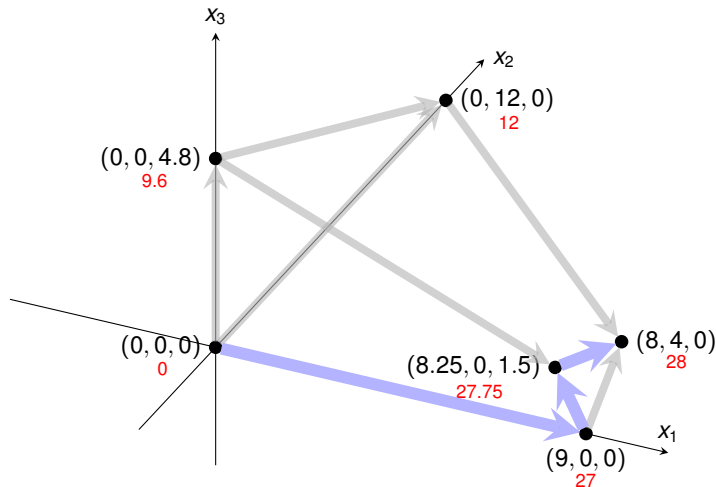
Extended Example: Visualization of SIMPLEX



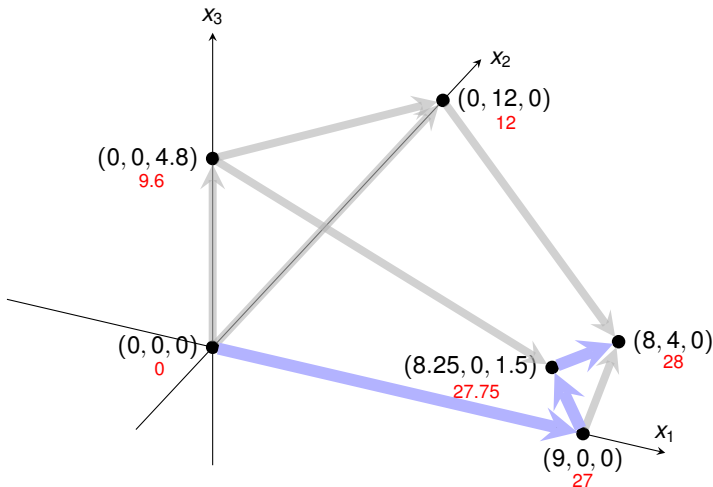
Extended Example: Visualization of SIMPLEX



Extended Example: Visualization of SIMPLEX



Extended Example: Visualization of SIMPLEX



Exercise: [Ex. 6/7.6] How many basic solutions (including non-feasible ones) are there?

Extended Example: Alternative Runs (1/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

Extended Example: Alternative Runs (1/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

↓ Switch roles of x_2 and x_5

Extended Example: Alternative Runs (1/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

Switch roles of x_2 and x_5
↓

$$\begin{array}{rclclclcl} z & = & 12 & + & 2x_1 & - & \frac{x_3}{2} & - & \frac{x_5}{2} \\ x_2 & = & 12 & - & x_1 & - & \frac{5x_3}{2} & - & \frac{x_5}{2} \\ x_4 & = & 18 & - & x_2 & - & \frac{x_3}{2} & + & \frac{x_5}{2} \\ x_6 & = & 24 & - & 3x_1 & + & \frac{x_3}{2} & + & \frac{x_5}{2} \end{array}$$

Extended Example: Alternative Runs (1/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

Switch roles of x_2 and x_5
↓

$$\begin{array}{rclclclcl} z & = & 12 & + & 2x_1 & - & \frac{x_3}{2} & - & \frac{x_5}{2} \\ x_2 & = & 12 & - & x_1 & - & \frac{5x_3}{2} & - & \frac{x_5}{2} \\ x_4 & = & 18 & - & x_2 & - & \frac{x_3}{2} & + & \frac{x_5}{2} \\ x_6 & = & 24 & - & 3x_1 & + & \frac{x_3}{2} & + & \frac{x_5}{2} \end{array}$$

Switch roles of x_1 and x_6
↓

Extended Example: Alternative Runs (1/2)

$$\begin{array}{rclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

Switch roles of x_2 and x_5
▼

$$\begin{array}{rclclcl} z & = & 12 & + & 2x_1 & - & \frac{x_3}{2} & - & \frac{x_5}{2} \\ x_2 & = & 12 & - & x_1 & - & \frac{5x_3}{2} & - & \frac{x_5}{2} \\ x_4 & = & 18 & - & x_2 & - & \frac{x_3}{2} & + & \frac{x_5}{2} \\ x_6 & = & 24 & - & 3x_1 & + & \frac{x_3}{2} & + & \frac{x_5}{2} \end{array}$$

Switch roles of x_1 and x_6
▼

$$\begin{array}{rclclcl} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

↓ Switch roles of x_3 and x_5
▼

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

↓
Switch roles of x_3 and x_5

$$\begin{array}{rclclclcl} z & = & \frac{48}{5} & + & \frac{11x_1}{5} & + & \frac{x_2}{5} & - & \frac{2x_5}{5} \\ x_4 & = & \frac{78}{5} & + & \frac{x_1}{5} & + & \frac{x_2}{5} & + & \frac{3x_5}{5} \\ x_3 & = & \frac{24}{5} & - & \frac{2x_1}{5} & - & \frac{2x_2}{5} & - & \frac{x_5}{5} \\ x_6 & = & \frac{132}{5} & - & \frac{16x_1}{5} & - & \frac{x_2}{5} & + & \frac{2x_3}{5} \end{array}$$

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclclcl} z & = & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

↓ Switch roles of x_3 and x_5

$$\begin{array}{rclclclcl} z & = & \frac{48}{5} & + & \frac{11x_1}{5} & + & \frac{x_2}{5} & - & \frac{2x_5}{5} \\ x_4 & = & \frac{78}{5} & + & \frac{x_1}{5} & + & \frac{x_2}{5} & + & \frac{3x_5}{5} \\ x_3 & = & \frac{24}{5} & - & \frac{2x_1}{5} & - & \frac{2x_2}{5} & - & \frac{x_5}{5} \\ x_6 & = & \frac{132}{5} & - & \frac{16x_1}{5} & - & \frac{x_2}{5} & + & \frac{2x_3}{5} \end{array}$$

Switch roles of x_1 and x_6

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclcl}
 z & = & & 3x_1 & + & x_2 & + & 2x_3 \\
 x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
 x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
 x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
 \end{array}$$

↓ Switch roles of x_3 and x_5

$$\begin{array}{rclclcl}
 z & = & \frac{48}{5} & + & \frac{11x_1}{5} & + & \frac{x_2}{5} & - & \frac{2x_5}{5} \\
 x_4 & = & \frac{78}{5} & + & \frac{x_1}{5} & + & \frac{x_2}{5} & + & \frac{3x_5}{5} \\
 x_3 & = & \frac{24}{5} & - & \frac{2x_1}{5} & - & \frac{2x_2}{5} & - & \frac{x_5}{5} \\
 x_6 & = & \frac{132}{5} & - & \frac{16x_1}{5} & - & \frac{x_2}{5} & + & \frac{2x_3}{5}
 \end{array}$$

Switch roles of x_1 and x_6

$$\begin{array}{rclclcl}
 z & = & \frac{111}{4} & + & \frac{x_2}{16} & - & \frac{x_5}{8} & - & \frac{11x_6}{16} \\
 x_1 & = & \frac{33}{4} & - & \frac{x_2}{16} & + & \frac{x_5}{8} & - & \frac{5x_6}{16} \\
 x_3 & = & \frac{3}{2} & - & \frac{3x_2}{8} & - & \frac{x_5}{4} & + & \frac{x_6}{8} \\
 x_4 & = & \frac{69}{4} & + & \frac{3x_2}{16} & + & \frac{5x_5}{8} & - & \frac{x_6}{16}
 \end{array}$$

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclclcl}
 z & = & & 3x_1 & + & x_2 & + & 2x_3 \\
 x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
 x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
 x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
 \end{array}$$

↓ Switch roles of x_3 and x_5

$$\begin{array}{rclclclclcl}
 z & = & \frac{48}{5} & + & \frac{11x_1}{5} & + & \frac{x_2}{5} & - & \frac{2x_5}{5} \\
 x_4 & = & \frac{78}{5} & + & \frac{x_1}{5} & + & \frac{x_2}{5} & + & \frac{3x_5}{5} \\
 x_3 & = & \frac{24}{5} & - & \frac{2x_1}{5} & - & \frac{2x_2}{5} & - & \frac{x_5}{5} \\
 x_6 & = & \frac{132}{5} & - & \frac{16x_1}{5} & - & \frac{x_2}{5} & + & \frac{2x_3}{5}
 \end{array}$$

Switch roles of x_1 and x_6

Switch roles of x_2 and x_3

$$\begin{array}{rclclclclcl}
 z & = & \frac{111}{4} & + & \frac{x_2}{16} & - & \frac{x_5}{8} & - & \frac{11x_6}{16} \\
 x_1 & = & \frac{33}{4} & - & \frac{x_2}{16} & + & \frac{x_5}{8} & - & \frac{5x_6}{16} \\
 x_3 & = & \frac{3}{2} & - & \frac{3x_2}{8} & - & \frac{x_5}{4} & + & \frac{x_6}{8} \\
 x_4 & = & \frac{69}{4} & + & \frac{3x_2}{16} & + & \frac{5x_5}{8} & - & \frac{x_6}{16}
 \end{array}$$

Extended Example: Alternative Runs (2/2)

$$\begin{array}{rclclcl}
 z & = & & 3x_1 & + & x_2 & + & 2x_3 \\
 x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
 x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
 x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
 \end{array}$$

↓ Switch roles of x_3 and x_5

$$\begin{array}{rclclcl}
 z & = & \frac{48}{5} & + & \frac{11x_1}{5} & + & \frac{x_2}{5} & - & \frac{2x_5}{5} \\
 x_4 & = & \frac{78}{5} & + & \frac{x_1}{5} & + & \frac{x_2}{5} & + & \frac{3x_5}{5} \\
 x_3 & = & \frac{24}{5} & - & \frac{2x_1}{5} & - & \frac{2x_2}{5} & - & \frac{x_5}{5} \\
 x_6 & = & \frac{132}{5} & - & \frac{16x_1}{5} & - & \frac{x_2}{5} & + & \frac{2x_3}{5}
 \end{array}$$

Switch roles of x_1 and x_6

Switch roles of x_2 and x_3

$$\begin{array}{rclclcl}
 z & = & \frac{111}{4} & + & \frac{x_2}{16} & - & \frac{x_5}{8} & - & \frac{11x_6}{16} \\
 x_1 & = & \frac{33}{4} & - & \frac{x_2}{16} & + & \frac{x_5}{8} & - & \frac{5x_6}{16} \\
 x_3 & = & \frac{3}{2} & - & \frac{3x_2}{8} & - & \frac{x_5}{4} & + & \frac{x_6}{8} \\
 x_4 & = & \frac{69}{4} & + & \frac{3x_2}{16} & + & \frac{5x_5}{8} & - & \frac{x_6}{16}
 \end{array}$$

$$\begin{array}{rclclcl}
 z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\
 x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\
 x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\
 x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2}
 \end{array}$$

Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)

The Pivot Step Formally

PIVOT(N, B, A, b, c, v, l, e)

```
1  // Compute the coefficients of the equation for new basic variable  $x_e$ .
2  let  $\hat{A}$  be a new  $m \times n$  matrix
3   $\hat{b}_e = b_l / a_{le}$ 
4  for each  $j \in N - \{e\}$ 
5       $\hat{a}_{ej} = a_{lj} / a_{le}$ 
6   $\hat{a}_{el} = 1 / a_{le}$ 
7  // Compute the coefficients of the remaining constraints.
8  for each  $i \in B - \{l\}$ 
9       $\hat{b}_i = b_i - a_{ie} \hat{b}_e$ 
10     for each  $j \in N - \{e\}$ 
11          $\hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej}$ 
12      $\hat{a}_{il} = -a_{ie} \hat{a}_{el}$ 
13 // Compute the objective function.
14  $\hat{v} = v + c_e \hat{b}_e$ 
15 for each  $j \in N - \{e\}$ 
16      $\hat{c}_j = c_j - c_e \hat{a}_{ej}$ 
17  $\hat{c}_l = -c_e \hat{a}_{el}$ 
18 // Compute new sets of basic and nonbasic variables.
19  $\hat{N} = N - \{e\} \cup \{l\}$ 
20  $\hat{B} = B - \{l\} \cup \{e\}$ 
21 return ( $\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v}$ )
```

The Pivot Step Formally

PIVOT(N, B, A, b, c, v, l, e)

```
1  // Compute the coefficients of the equation for new basic variable  $x_e$ .
2  let  $\hat{A}$  be a new  $m \times n$  matrix
3   $\hat{b}_e = b_l / a_{le}$ 
4  for each  $j \in N - \{e\}$ 
5       $\hat{a}_{ej} = a_{lj} / a_{le}$ 
6   $\hat{a}_{el} = 1 / a_{le}$ 
7  // Compute the coefficients of the remaining constraints.
8  for each  $i \in B - \{l\}$ 
9       $\hat{b}_i = b_i - a_{ie} \hat{b}_e$ 
10     for each  $j \in N - \{e\}$ 
11          $\hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej}$ 
12      $\hat{a}_{il} = -a_{ie} \hat{a}_{el}$ 
13 // Compute the objective function.
14  $\hat{v} = v + c_e \hat{b}_e$ 
15 for each  $j \in N - \{e\}$ 
16      $\hat{c}_j = c_j - c_e \hat{a}_{ej}$ 
17  $\hat{c}_l = -c_e \hat{a}_{el}$ 
18 // Compute new sets of basic and nonbasic variables.
19  $\hat{N} = N - \{e\} \cup \{l\}$ 
20  $\hat{B} = B - \{l\} \cup \{e\}$ 
21 return ( $\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v}$ )
```

Rewrite “tight” equation
for entering variable x_e .

The Pivot Step Formally

PIVOT(N, B, A, b, c, v, l, e)

```
1  // Compute the coefficients of the equation for new basic variable  $x_e$ .
2  let  $\hat{A}$  be a new  $m \times n$  matrix
3   $\hat{b}_e = b_l / a_{le}$ 
4  for each  $j \in N - \{e\}$ 
5       $\hat{a}_{ej} = a_{lj} / a_{le}$ 
6   $\hat{a}_{el} = 1 / a_{le}$ 
7  // Compute the coefficients of the remaining constraints.
8  for each  $i \in B - \{l\}$ 
9       $\hat{b}_i = b_i - a_{ie} \hat{b}_e$ 
10     for each  $j \in N - \{e\}$ 
11          $\hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej}$ 
12          $\hat{a}_{il} = -a_{ie} \hat{a}_{el}$ 
13 // Compute the objective function.
14  $\hat{v} = v + c_e \hat{b}_e$ 
15 for each  $j \in N - \{e\}$ 
16      $\hat{c}_j = c_j - c_e \hat{a}_{ej}$ 
17  $\hat{c}_l = -c_e \hat{a}_{el}$ 
18 // Compute new sets of basic and nonbasic variables.
19  $\hat{N} = N - \{e\} \cup \{l\}$ 
20  $\hat{B} = B - \{l\} \cup \{e\}$ 
21 return ( $\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v}$ )
```

Rewrite “tight” equation
for entering variable x_e .

Substituting x_e into
other equations.

The Pivot Step Formally

PIVOT(N, B, A, b, c, v, l, e)

```
1  // Compute the coefficients of the equation for new basic variable  $x_e$ .
2  let  $\hat{A}$  be a new  $m \times n$  matrix
3   $\hat{b}_e = b_l / a_{le}$ 
4  for each  $j \in N - \{e\}$ 
5       $\hat{a}_{ej} = a_{lj} / a_{le}$ 
6   $\hat{a}_{el} = 1 / a_{le}$ 
7  // Compute the coefficients of the remaining constraints.
8  for each  $i \in B - \{l\}$ 
9       $\hat{b}_i = b_i - a_{ie} \hat{b}_e$ 
10     for each  $j \in N - \{e\}$ 
11          $\hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej}$ 
12      $\hat{a}_{il} = -a_{ie} \hat{a}_{el}$ 
13 // Compute the objective function.
14  $\hat{v} = v + c_e \hat{b}_e$ 
15 for each  $j \in N - \{e\}$ 
16      $\hat{c}_j = c_j - c_e \hat{a}_{ej}$ 
17  $\hat{c}_l = -c_e \hat{a}_{el}$ 
18 // Compute new sets of basic and nonbasic variables.
19  $\hat{N} = N - \{e\} \cup \{l\}$ 
20  $\hat{B} = B - \{l\} \cup \{e\}$ 
21 return ( $\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v}$ )
```

Rewrite “tight” equation
for entering variable x_e .

Substituting x_e into
other equations.

Substituting x_e into
objective function.

The Pivot Step Formally

PIVOT(N, B, A, b, c, v, l, e)

```
1  // Compute the coefficients of the equation for new basic variable  $x_e$ .
2  let  $\hat{A}$  be a new  $m \times n$  matrix
3   $\hat{b}_e = b_l / a_{le}$ 
4  for each  $j \in N - \{e\}$ 
5       $\hat{a}_{ej} = a_{lj} / a_{le}$ 
6   $\hat{a}_{el} = 1 / a_{le}$ 
7  // Compute the coefficients of the remaining constraints.
8  for each  $i \in B - \{l\}$ 
9       $\hat{b}_i = b_i - a_{ie} \hat{b}_e$ 
10     for each  $j \in N - \{e\}$ 
11          $\hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej}$ 
12      $\hat{a}_{il} = -a_{ie} \hat{a}_{el}$ 
13 // Compute the objective function.
14  $\hat{v} = v + c_e \hat{b}_e$ 
15 for each  $j \in N - \{e\}$ 
16      $\hat{c}_j = c_j - c_e \hat{a}_{ej}$ 
17  $\hat{c}_l = -c_e \hat{a}_{el}$ 
18 // Compute new sets of basic and nonbasic variables.
19  $\hat{N} = N - \{e\} \cup \{l\}$ 
20  $\hat{B} = B - \{l\} \cup \{e\}$ 
21 return  $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$ 
```

Rewrite “tight” equation for entering variable x_e .

Substituting x_e into other equations.

Substituting x_e into objective function.

Update non-basic and basic variables

The Pivot Step Formally

PIVOT(N, B, A, b, c, v, l, e)

```
1  // Compute the coefficients of the equation for new basic variable  $x_e$ .
2  let  $\hat{A}$  be a new  $m \times n$  matrix
3   $\hat{b}_e = b_l / a_{le}$ 
4  for each  $j \in N - \{e\}$  Need that  $a_{le} \neq 0$ !
5       $\hat{a}_{ej} = a_{lj} / a_{le}$ 
6   $\hat{a}_{el} = 1 / a_{le}$ 
7  // Compute the coefficients of the remaining constraints.
8  for each  $i \in B - \{l\}$ 
9       $\hat{b}_i = b_i - a_{ie} \hat{b}_e$ 
10     for each  $j \in N - \{e\}$ 
11          $\hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej}$ 
12      $\hat{a}_{il} = -a_{ie} \hat{a}_{el}$ 
13 // Compute the objective function.
14  $\hat{v} = v + c_e \hat{b}_e$ 
15 for each  $j \in N - \{e\}$ 
16      $\hat{c}_j = c_j - c_e \hat{a}_{ej}$ 
17  $\hat{c}_l = -c_e \hat{a}_{el}$ 
18 // Compute new sets of basic and nonbasic variables.
19  $\hat{N} = N - \{e\} \cup \{l\}$ 
20  $\hat{B} = B - \{l\} \cup \{e\}$ 
21 return  $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$ 
```

Rewrite “tight” equation for entering variable x_e .

Substituting x_e into other equations.

Substituting x_e into objective function.

Update non-basic and basic variables

Effect of the Pivot Step (extra material, non-examinable)

— Lemma 29.1 —

Consider a call to $\text{PIVOT}(N, B, A, b, c, v, l, e)$ in which $a_{le} \neq 0$. Let the values returned from the call be $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$, and let \bar{x} denote the basic solution after the call. Then

Effect of the Pivot Step (extra material, non-examinable)

Lemma 29.1

Consider a call to $\text{PIVOT}(N, B, A, b, c, v, l, e)$ in which $a_{le} \neq 0$. Let the values returned from the call be $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$, and let \bar{x} denote the basic solution after the call. Then

1. $\bar{x}_j = 0$ for each $j \in \hat{N}$.
2. $\bar{x}_e = b_l / a_{le}$.
3. $\bar{x}_i = b_i - a_{ie} \hat{b}_e$ for each $i \in \hat{B} \setminus \{e\}$.

Effect of the Pivot Step (extra material, non-examinable)

Lemma 29.1

Consider a call to $\text{PIVOT}(N, B, A, b, c, v, l, e)$ in which $a_{le} \neq 0$. Let the values returned from the call be $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$, and let \bar{x} denote the basic solution after the call. Then

1. $\bar{x}_j = 0$ for each $j \in \hat{N}$.
2. $\bar{x}_e = b_l / a_{le}$.
3. $\bar{x}_i = b_i - a_{ie} \hat{b}_e$ for each $i \in \hat{B} \setminus \{e\}$.

Proof:

Effect of the Pivot Step (extra material, non-examinable)

— Lemma 29.1 —

Consider a call to $\text{PIVOT}(N, B, A, b, c, v, l, e)$ in which $a_{le} \neq 0$. Let the values returned from the call be $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$, and let \bar{x} denote the basic solution after the call. Then

1. $\bar{x}_j = 0$ for each $j \in \hat{N}$.
2. $\bar{x}_e = b_l / a_{le}$.
3. $\bar{x}_i = b_i - a_{ie} \hat{b}_e$ for each $i \in \hat{B} \setminus \{e\}$.

Proof:

1. holds since the basic solution always sets all non-basic variables to zero.
2. When we set each non-basic variable to 0 in a constraint

$$x_i = \hat{b}_i - \sum_{j \in \hat{N}} \hat{a}_{ij} x_j,$$

we have $\bar{x}_i = \hat{b}_i$ for each $i \in \hat{B}$. Hence $\bar{x}_e = \hat{b}_e = b_l / a_{le}$.

3. After substituting into the other constraints, we have

$$\bar{x}_i = \hat{b}_i = b_i - a_{ie} \hat{b}_e.$$

Effect of the Pivot Step (extra material, non-examinable)

— Lemma 29.1 —

Consider a call to $\text{PIVOT}(N, B, A, b, c, v, l, e)$ in which $a_{le} \neq 0$. Let the values returned from the call be $(\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})$, and let \bar{x} denote the basic solution after the call. Then

1. $\bar{x}_j = 0$ for each $j \in \hat{N}$.
2. $\bar{x}_e = b_l / a_{le}$.
3. $\bar{x}_i = b_i - a_{ie} \hat{b}_e$ for each $i \in \hat{B} \setminus \{e\}$.

Proof:

1. holds since the basic solution always sets all non-basic variables to zero.
2. When we set each non-basic variable to 0 in a constraint

$$x_i = \hat{b}_i - \sum_{j \in \hat{N}} \hat{a}_{ij} x_j,$$

we have $\bar{x}_i = \hat{b}_i$ for each $i \in \hat{B}$. Hence $\bar{x}_e = \hat{b}_e = b_l / a_{le}$.

3. After substituting into the other constraints, we have

$$\bar{x}_i = \hat{b}_i = b_i - a_{ie} \hat{b}_e. \quad \square$$

Questions:

- How do we determine whether a linear program is feasible?
- What do we do if the linear program is feasible, but the initial basic solution is not feasible?
- How do we determine whether a linear program is unbounded?
- How do we choose the entering and leaving variables?

Questions:

- How do we determine whether a linear program is feasible?
- What do we do if the linear program is feasible, but the initial basic solution is not feasible?
- How do we determine whether a linear program is unbounded?
- How do we choose the entering and leaving variables?

Example before was a particularly nice one!

The formal procedure SIMPLEX

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

The formal procedure **SIMPLEX**

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

Returns a slack form with a feasible basic solution (if it exists)

The formal procedure SIMPLEX

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

Returns a slack form with a feasible basic solution (if it exists)

The formal procedure SIMPLEX

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

Returns a slack form with a feasible basic solution (if it exists)

Main Loop:

The formal procedure **SIMPLEX**

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l = \infty$ 
11         return “unbounded”
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

Returns a slack form with a feasible basic solution (if it exists)

Main Loop:

- terminates if all coefficients in objective function are **non-positive**
- Line 4 picks entering variable x_e with **positive** coefficient
- Lines 6 – 9 pick the tightest constraint, associated with x_l
- Line 11 returns “unbounded” if there are no constraints
- Line 12 calls PIVOT, switching roles of x_l and x_e

The formal procedure **SIMPLEX**

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l = \infty$ 
11         return "unbounded"
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

Returns a slack form with a feasible basic solution (if it exists)

Main Loop:

- terminates if all coefficients in objective function are **non-positive**
- Line 4 picks entering variable x_e with **positive** coefficient
- Lines 6 – 9 pick the tightest constraint, associated with x_l
- Line 11 returns "unbounded" if there are no constraints
- Line 12 calls PIVOT, switching roles of x_l and x_e

Return corresponding solution.

The formal procedure **SIMPLEX**

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
12     else ( $N, B, A, b, c, v$ ) = PIVOT( $N, B, A, b, c, v, l, e$ )
13 for  $i = 1$  to  $n$ 
14     if  $i \in B$ 
15          $\bar{x}_i = b_i$ 
16     else  $\bar{x}_i = 0$ 
17 return ( $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ )
```

Returns a slack form with a feasible basic solution (if it exists)

Lemma 29.2

Suppose the call to INITIALIZE-SIMPLEX in line 1 returns a slack form for which the basic solution is feasible. Then if SIMPLEX returns a solution, it is a feasible solution. If SIMPLEX returns “unbounded”, the linear program is unbounded.

The formal procedure **SIMPLEX**

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
```

Returns a slack form with a feasible basic solution (if it exists)

Proof is based on the following three-part loop invariant:

— Lemma 29.2 —

Suppose the call to INITIALIZE-SIMPLEX in line 1 returns a slack form for which the basic solution is feasible. Then if SIMPLEX returns a solution, it is a feasible solution. If SIMPLEX returns “unbounded”, the linear program is unbounded.

The formal procedure **SIMPLEX**

SIMPLEX(A, b, c)

```
1  ( $N, B, A, b, c, v$ ) = INITIALIZE-SIMPLEX( $A, b, c$ )
2  let  $\Delta$  be a new vector of length  $m$ 
3  while some index  $j \in N$  has  $c_j > 0$ 
4      choose an index  $e \in N$  for which  $c_e > 0$ 
5      for each index  $i \in B$ 
6          if  $a_{ie} > 0$ 
7               $\Delta_i = b_i / a_{ie}$ 
8          else  $\Delta_i = \infty$ 
9      choose an index  $l \in B$  that minimizes  $\Delta_i$ 
10     if  $\Delta_l == \infty$ 
11         return “unbounded”
```

Returns a slack form with a feasible basic solution (if it exists)

Proof is based on the following three-part loop invariant:

1. the slack form is always equivalent to the one returned by **INITIALIZE-SIMPLEX**,
2. for each $i \in B$, we have $b_i \geq 0$,
3. the basic solution associated with the (current) slack form is feasible.

Lemma 29.2

Suppose the call to **INITIALIZE-SIMPLEX** in line 1 returns a slack form for which the basic solution is feasible. Then if **SIMPLEX** returns a solution, it is a feasible solution. If **SIMPLEX** returns “unbounded”, the linear program is unbounded.

Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)

Finding an Initial Solution

$$\begin{array}{llllll} \text{maximise} & 2x_1 & - & x_2 & & \\ \text{subject to} & & & & & \\ & 2x_1 & - & x_2 & \leq & 2 \\ & x_1 & - & 5x_2 & \leq & -4 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

Finding an Initial Solution

maximise
subject to

$$2x_1 - x_2$$

$$2x_1 - x_2 \leq 2$$

$$x_1 - 5x_2 \leq -4$$

$$x_1, x_2 \geq 0$$



Conversion into slack form

Finding an Initial Solution

maximise
subject to

$$2x_1 - x_2$$

$$2x_1 - x_2 \leq 2$$

$$x_1 - 5x_2 \leq -4$$

$$x_1, x_2 \geq 0$$

Conversion into slack form

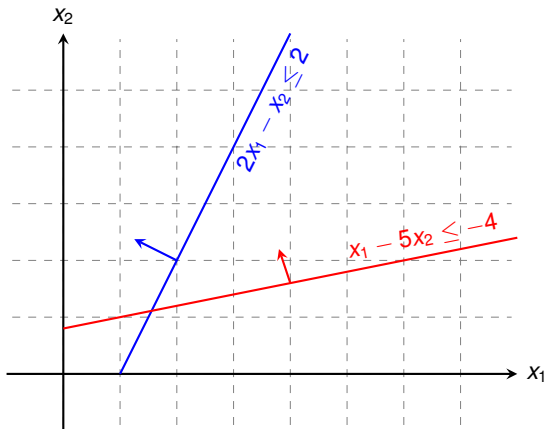
$$\begin{aligned} z &= & 2x_1 &- x_2 \\ x_3 &= 2 &- 2x_1 &+ x_2 \\ x_4 &= -4 &- x_1 &+ 5x_2 \end{aligned}$$

Basic solution $(x_1, x_2, x_3, x_4) = (0, 0, 2, -4)$ is not feasible!

Geometric Illustration

maximise
subject to

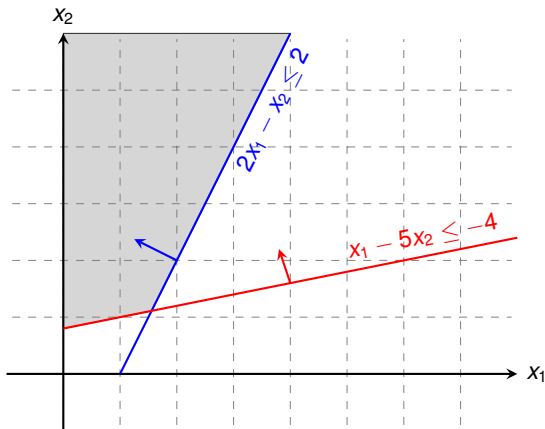
$$\begin{array}{rclcl} 2x_1 & - & x_2 & & \\ 2x_1 & - & x_2 & \leq & 2 \\ x_1 & - & 5x_2 & \leq & -4 \\ x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise
subject to

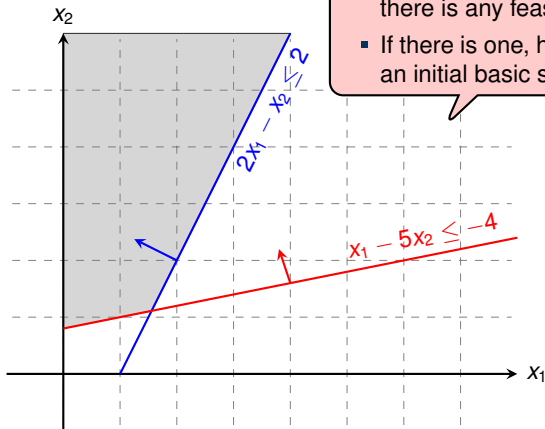
$$\begin{array}{rclcl} 2x_1 & - & x_2 & & \\ 2x_1 & - & x_2 & \leq & 2 \\ x_1 & - & 5x_2 & \leq & -4 \\ x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise
subject to

$$\begin{array}{rclcl} 2x_1 & - & x_2 & & \\ 2x_1 & - & x_2 & \leq & 2 \\ x_1 & - & 5x_2 & \leq & -4 \\ x_1, x_2 & & & \geq & 0 \end{array}$$



Questions:

- How to determine whether there is any feasible solution?
- If there is one, how to determine an initial basic solution?

Formulating an Auxiliary Linear Program

maximise $\sum_{j=1}^n c_j x_j$
subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 1, 2, \dots, n \end{aligned}$$

Formulating an Auxiliary Linear Program

maximise $\sum_{j=1}^n c_j x_j$
subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 1, 2, \dots, n \end{aligned}$$

↓
Formulating an Auxiliary Linear Program

Formulating an Auxiliary Linear Program

maximise $\sum_{j=1}^n c_j x_j$
subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 1, 2, \dots, n \end{aligned}$$

↓
Formulating an Auxiliary Linear Program

maximise $-x_0$
subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j - x_0 &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 0, 1, \dots, n \end{aligned}$$

Formulating an Auxiliary Linear Program

maximise $\sum_{j=1}^n c_j x_j$
subject to

$$\begin{aligned}\sum_{j=1}^n a_{ij} x_j &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 1, 2, \dots, n\end{aligned}$$

↓ Formulating an Auxiliary Linear Program

maximise $-x_0$
subject to

$$\begin{aligned}\sum_{j=1}^n a_{ij} x_j - x_0 &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 0, 1, \dots, n\end{aligned}$$

— Lemma 29.11 —

Let L_{aux} be the auxiliary LP of a linear program L in standard form. Then L is feasible if and only if the optimal objective value of L_{aux} is 0.

Formulating an Auxiliary Linear Program

maximise $\sum_{j=1}^n c_j x_j$
subject to

$$\begin{aligned}\sum_{j=1}^n a_{ij} x_j &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 1, 2, \dots, n\end{aligned}$$

↓ Formulating an Auxiliary Linear Program

maximise $-x_0$
subject to

$$\begin{aligned}\sum_{j=1}^n a_{ij} x_j - x_0 &\leq b_i && \text{for } i = 1, 2, \dots, m, \\ x_j &\geq 0 && \text{for } j = 0, 1, \dots, n\end{aligned}$$

— Lemma 29.11 —

Let L_{aux} be the auxiliary LP of a linear program L in standard form. Then L is feasible if and only if the optimal objective value of L_{aux} is 0.

Proof. Exercise!

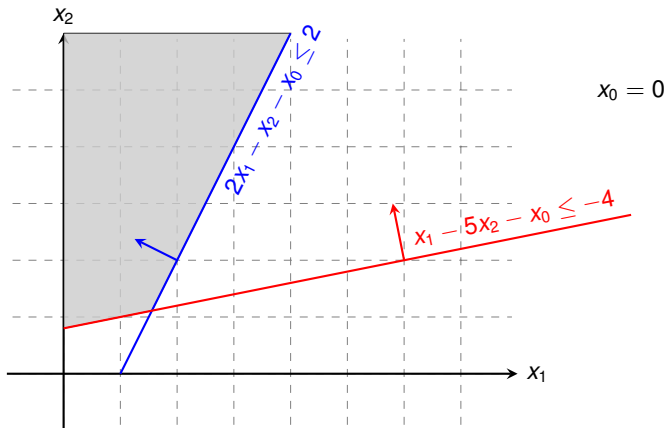
- Let us illustrate the role of x_0 as “distance from feasibility”

- Let us illustrate the role of x_0 as “distance from feasibility”
- We’ll also see that increasing x_0 enlarges the feasible region

Geometric Illustration

maximise $-x_0$
subject to

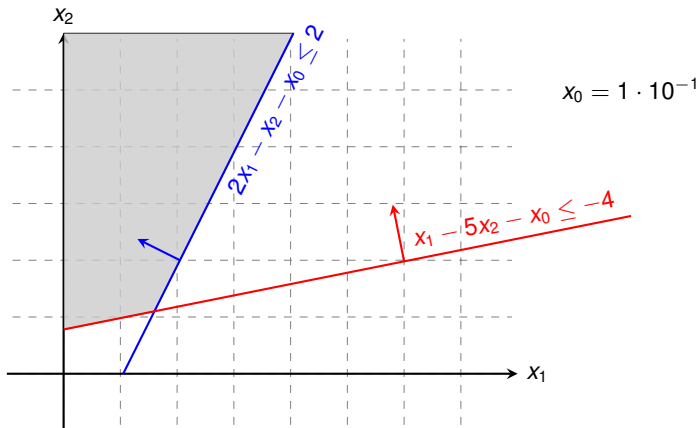
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

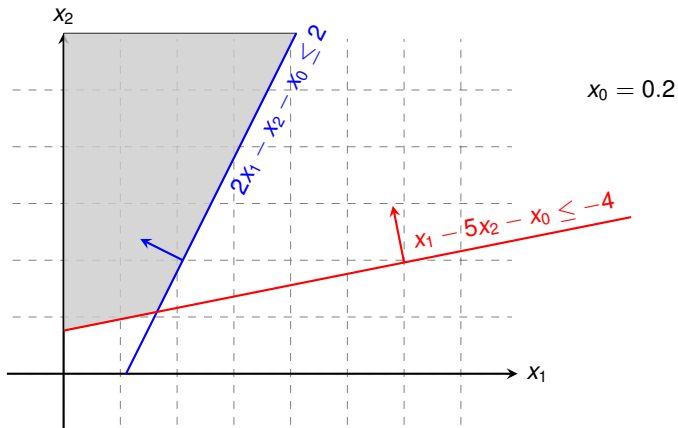
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

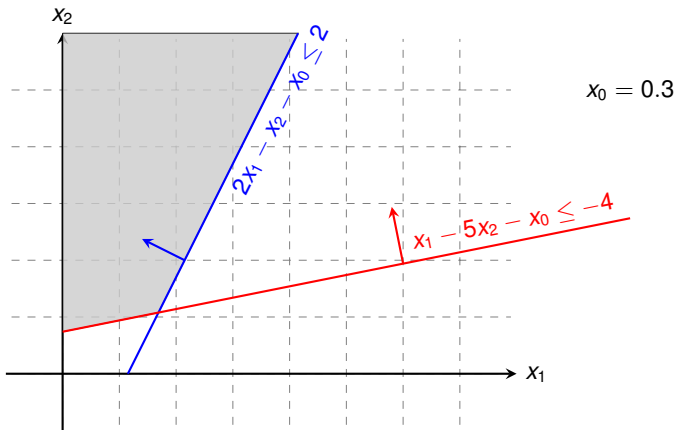
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

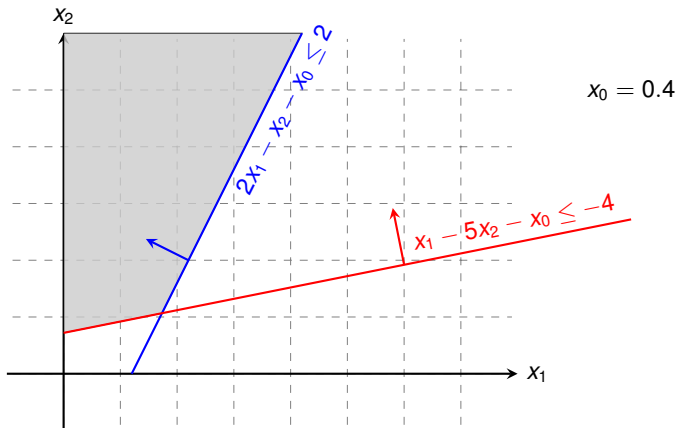
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

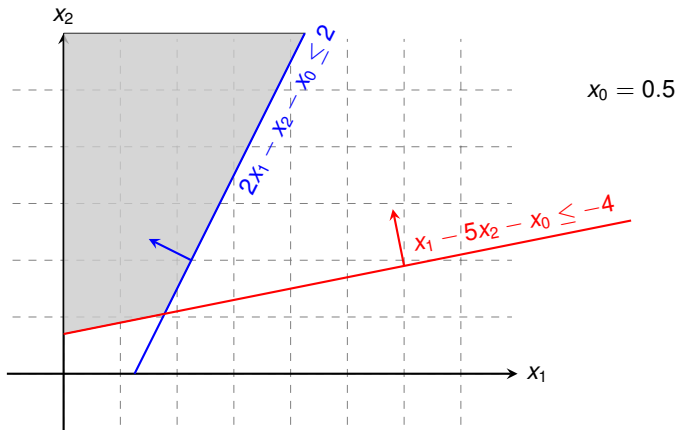
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

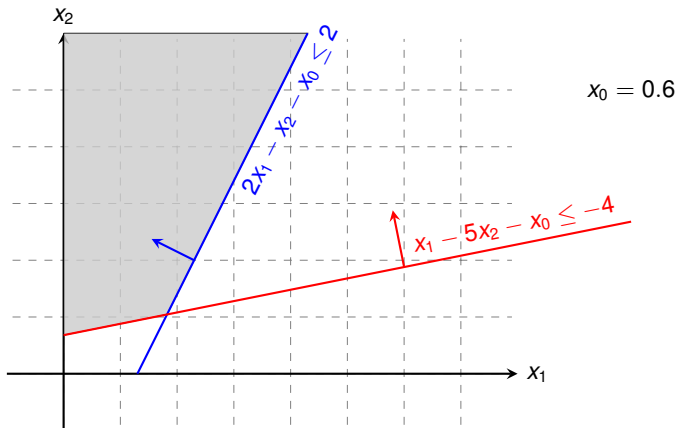
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

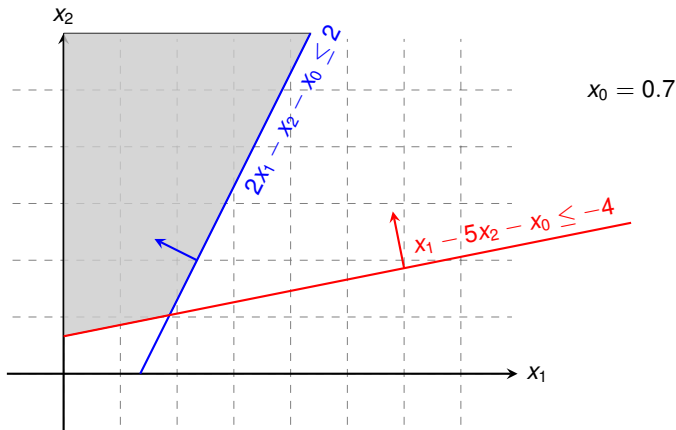
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

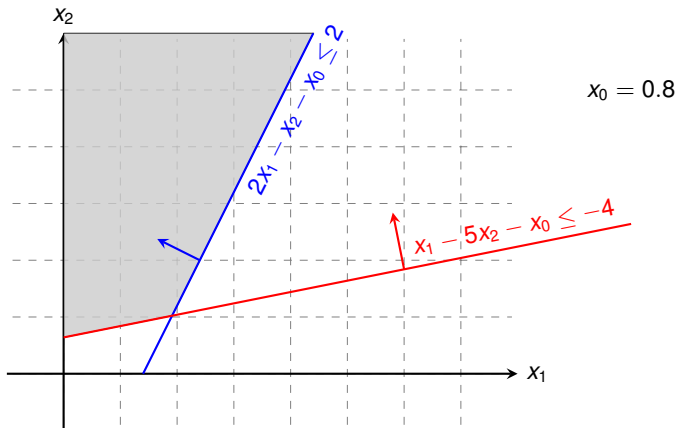
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

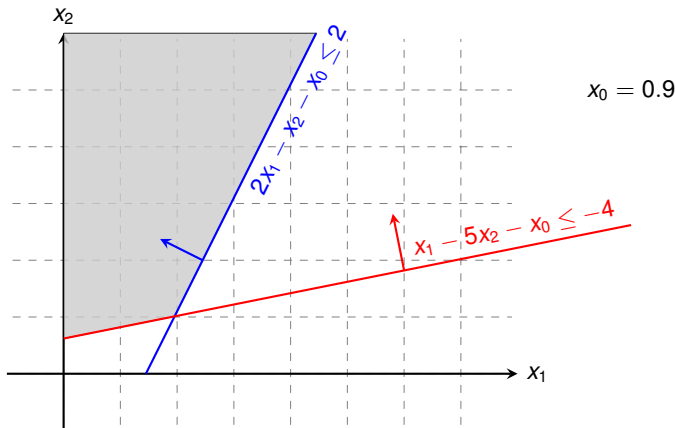
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

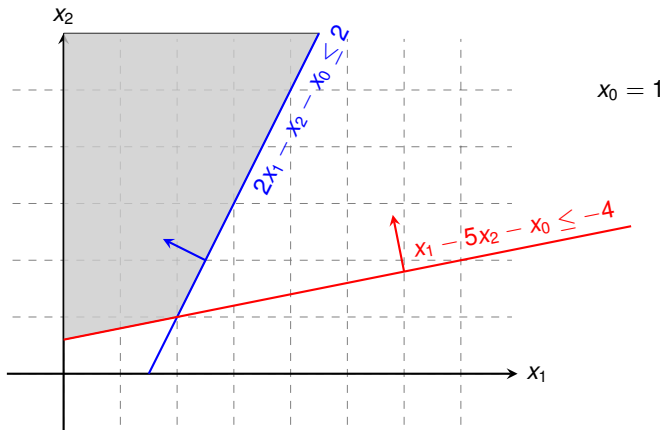
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

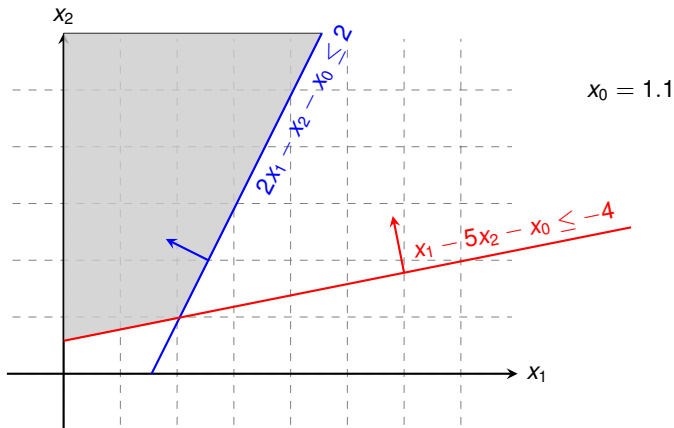
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

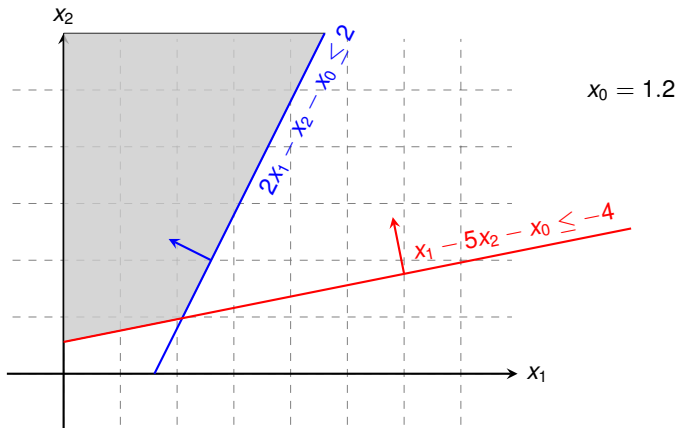
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

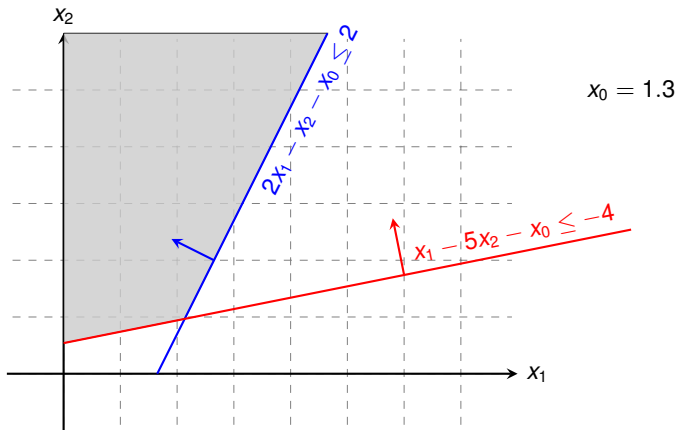
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

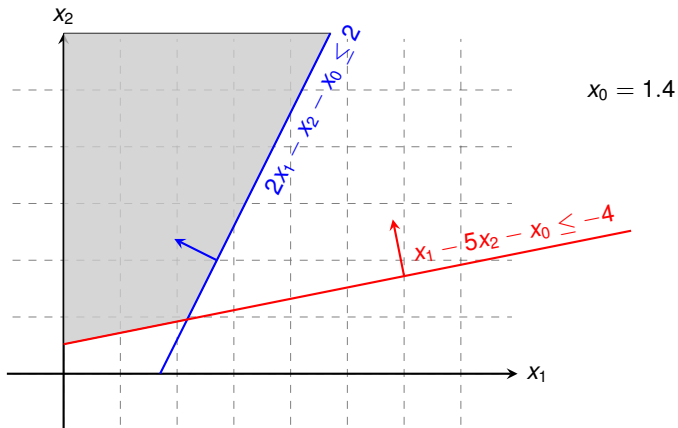
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

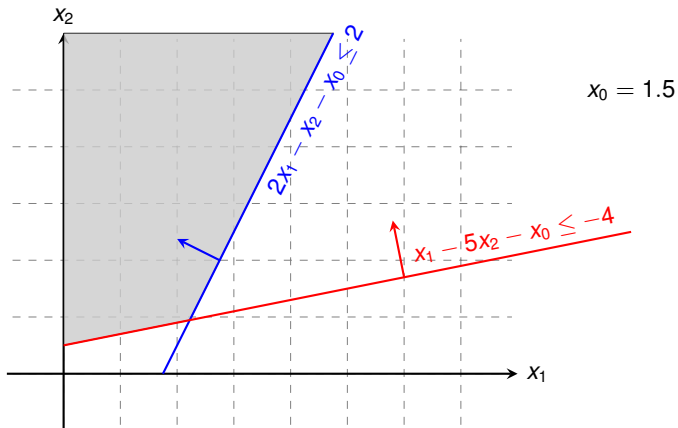
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

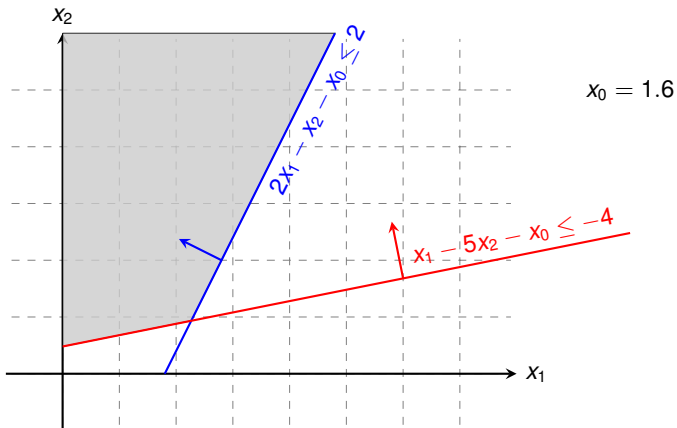
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

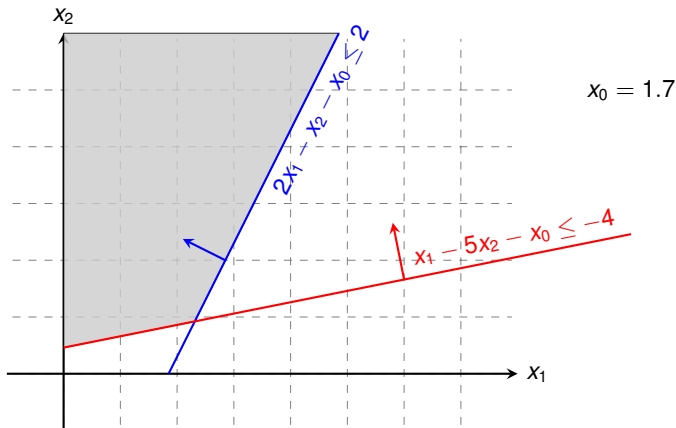
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

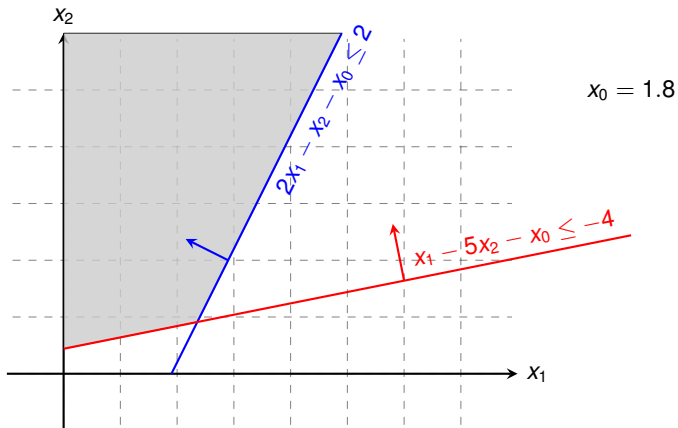
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

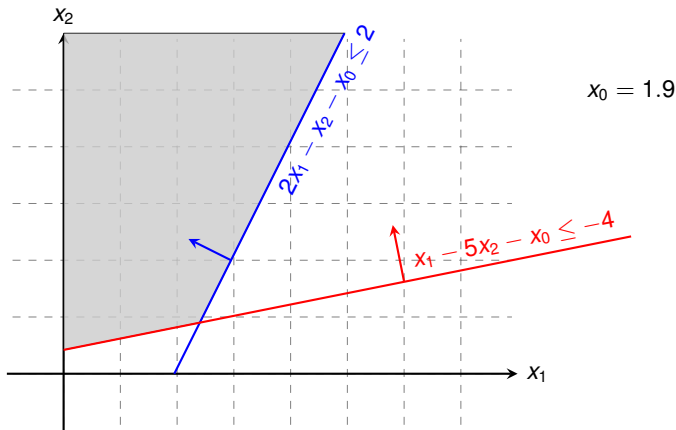
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

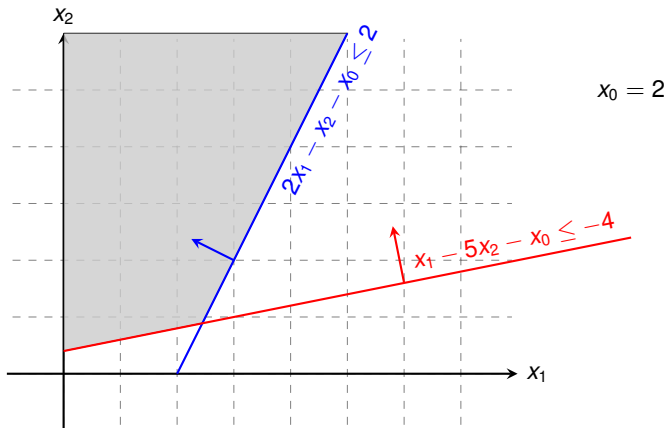
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

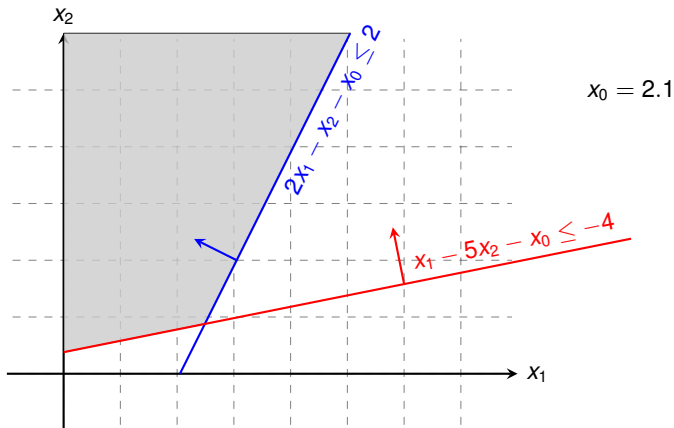
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

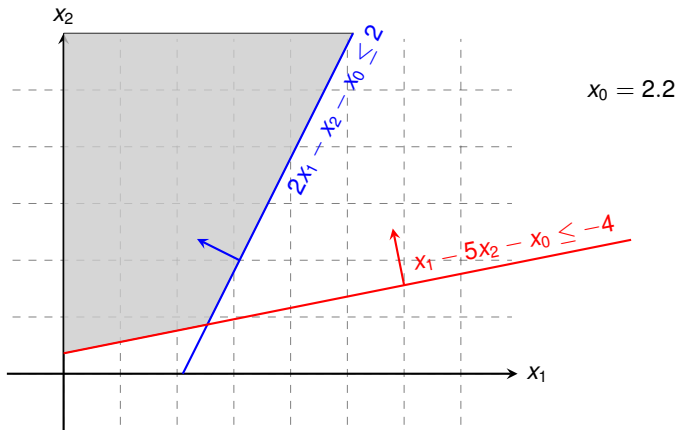
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

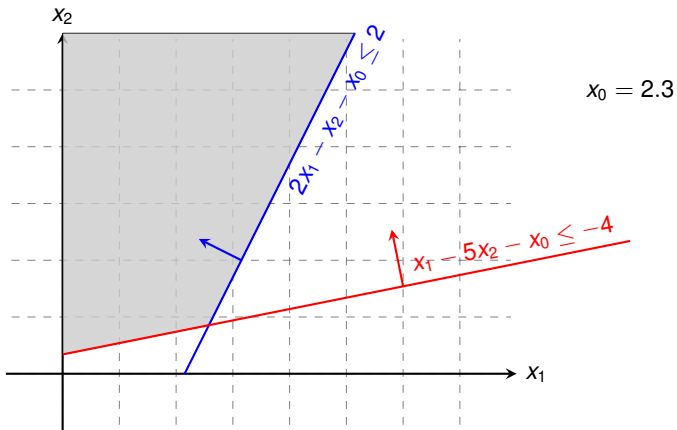
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

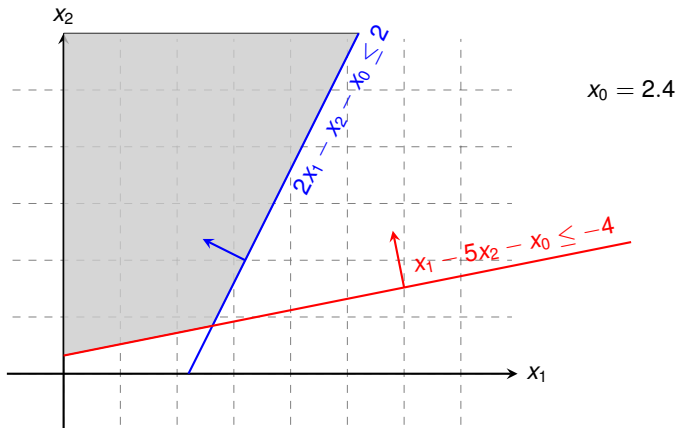
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

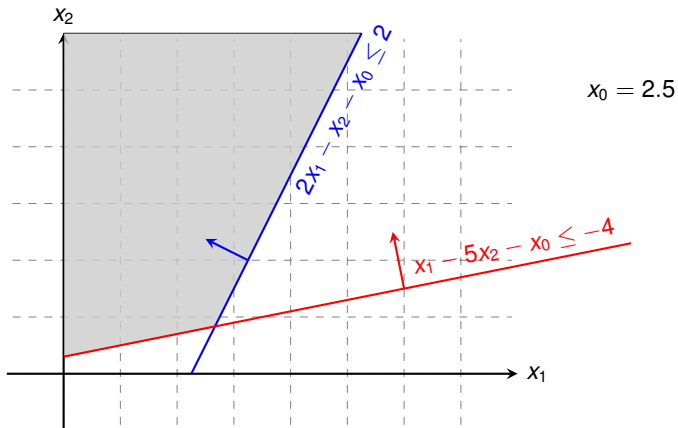
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

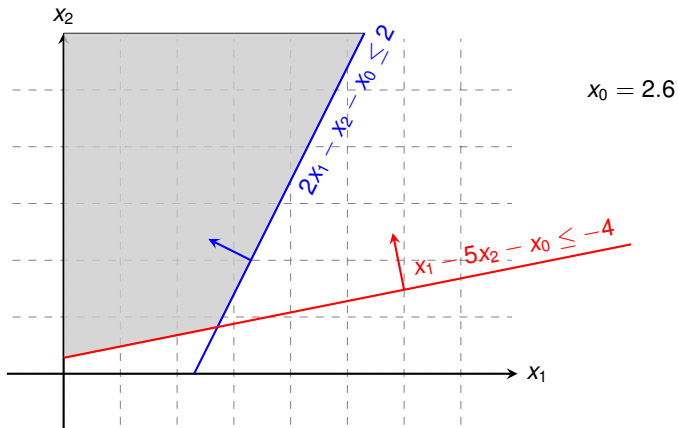
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

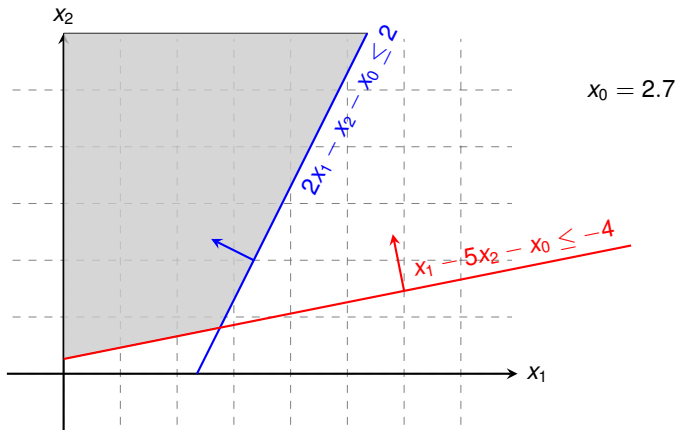
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

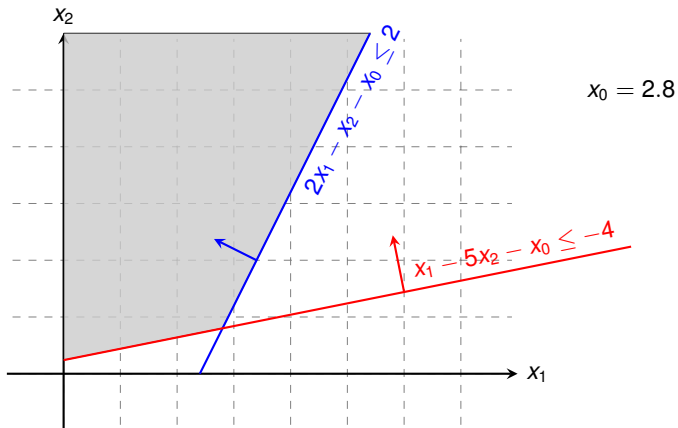
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

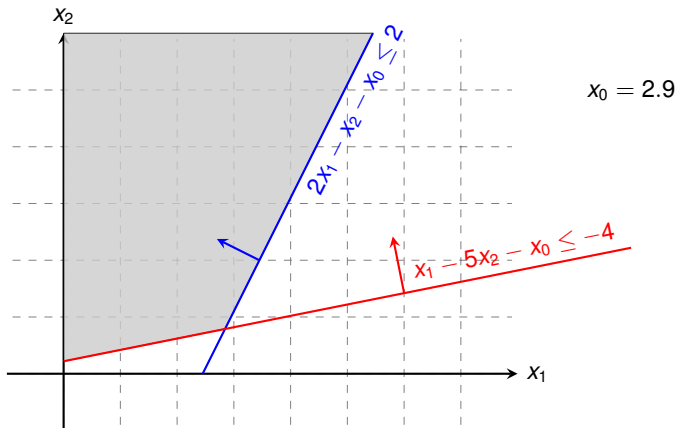
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

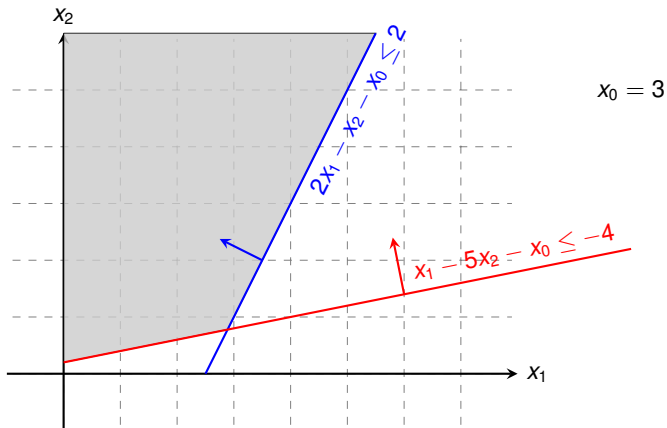
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

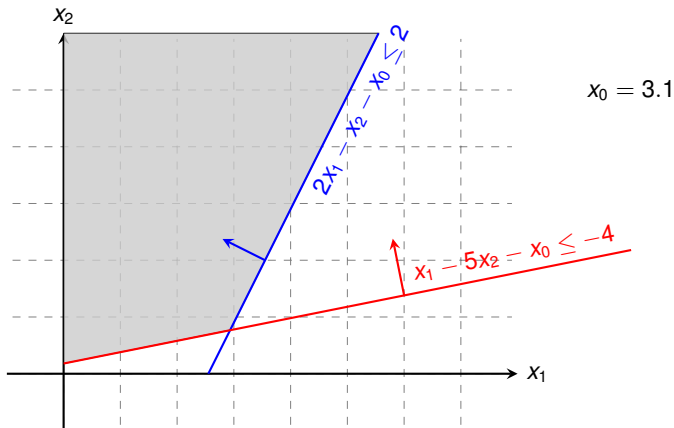
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

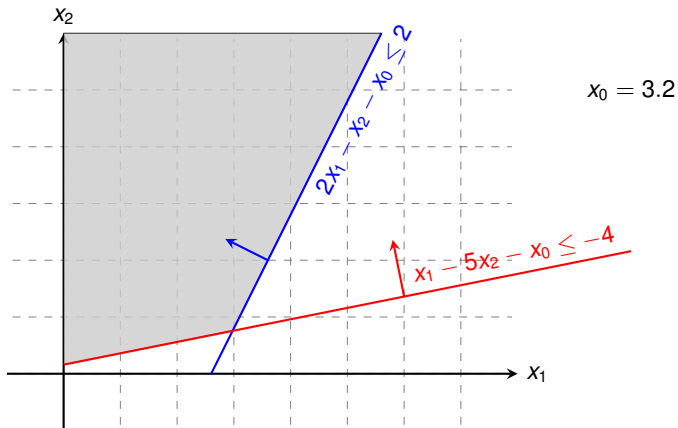
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

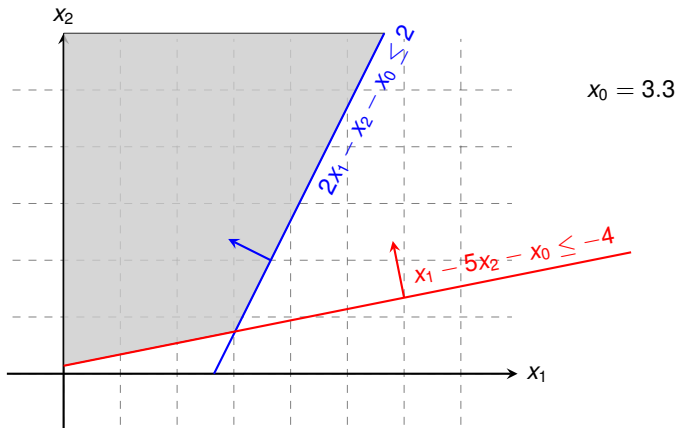
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

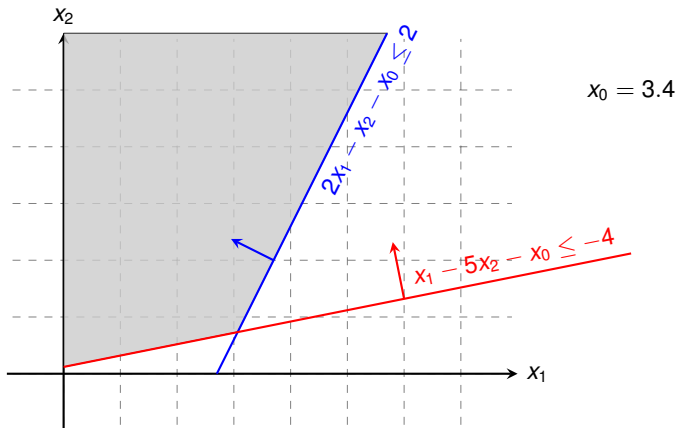
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
 subject to

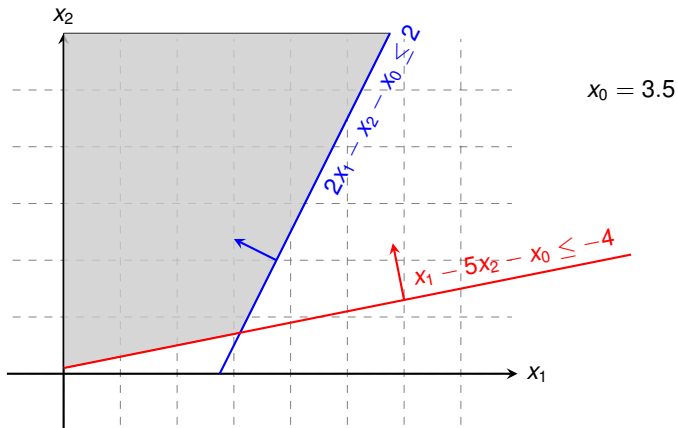
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

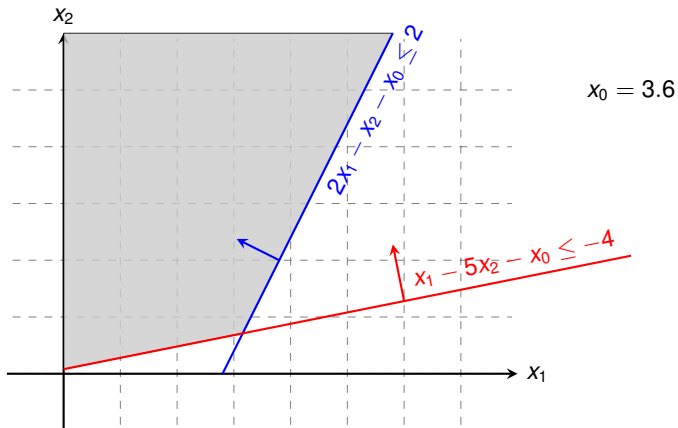
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

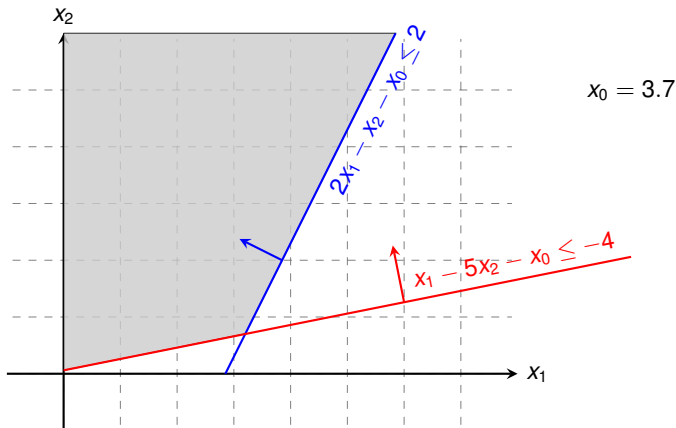
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

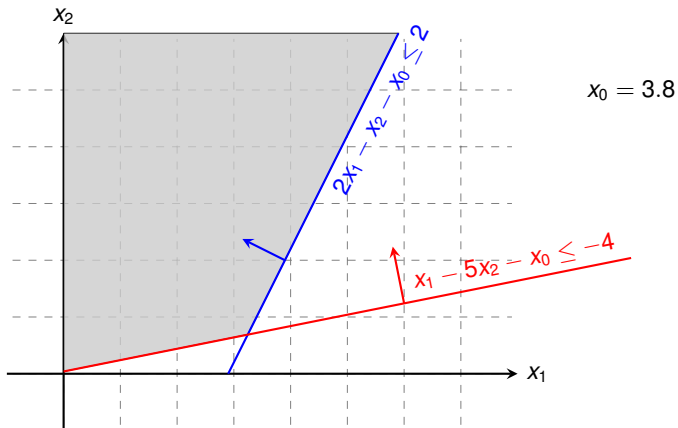
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

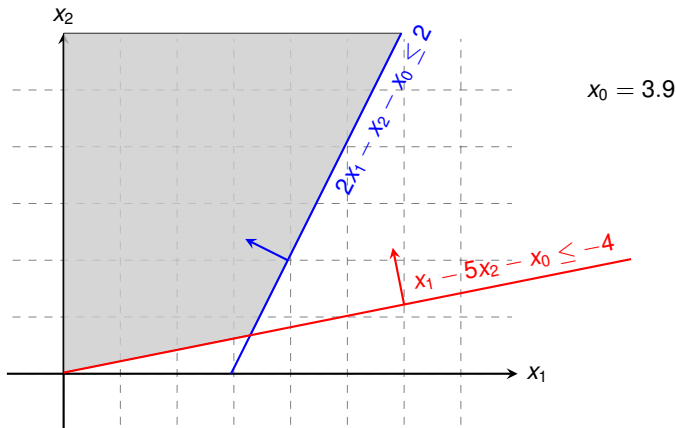
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

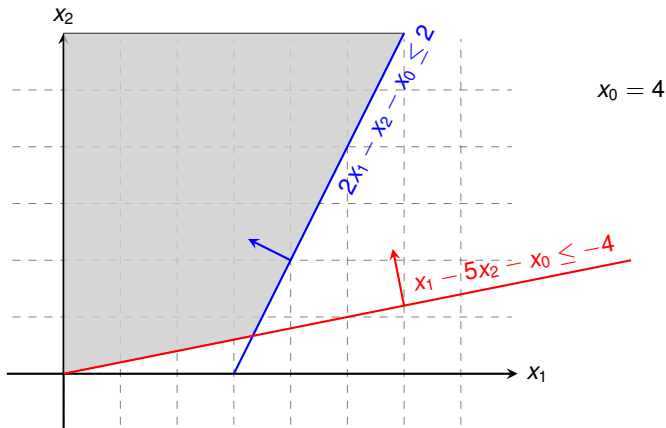
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



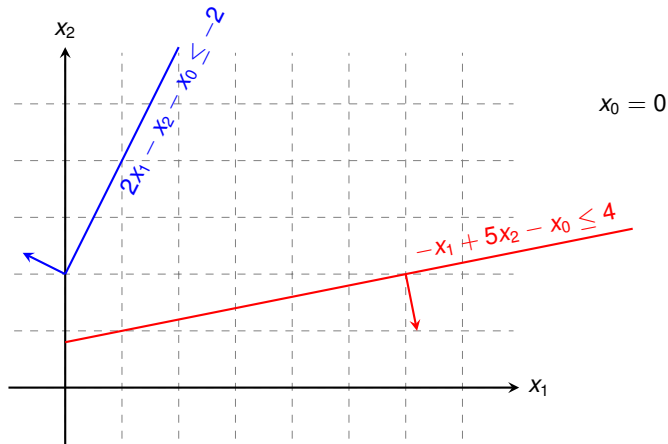
- Let us now modify the original linear program so that it is not feasible

- Let us now modify the original linear program so that it is **not feasible**
- ⇒ Hence the auxiliary linear program has only a solution for a sufficiently large $x_0 > 0$!

Geometric Illustration

maximise $-x_0$
subject to

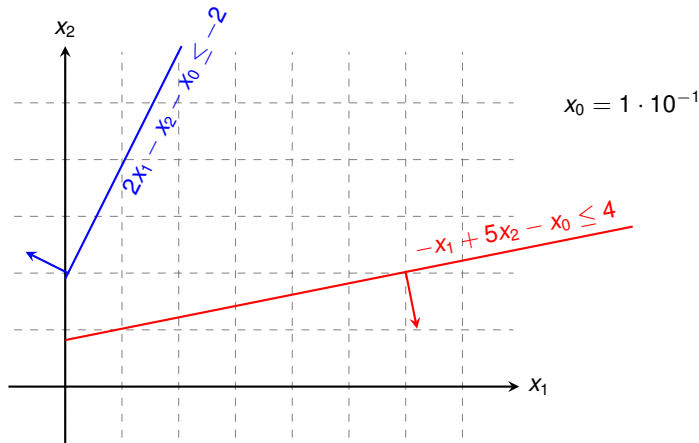
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

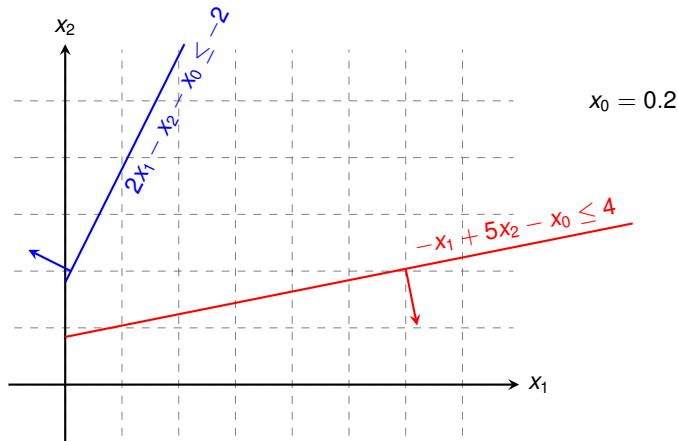
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

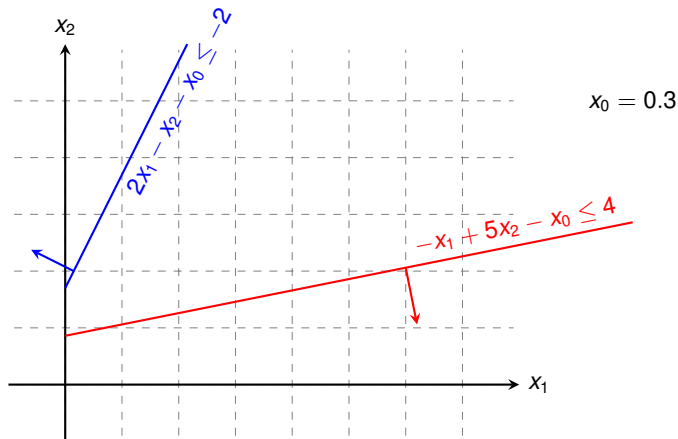
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

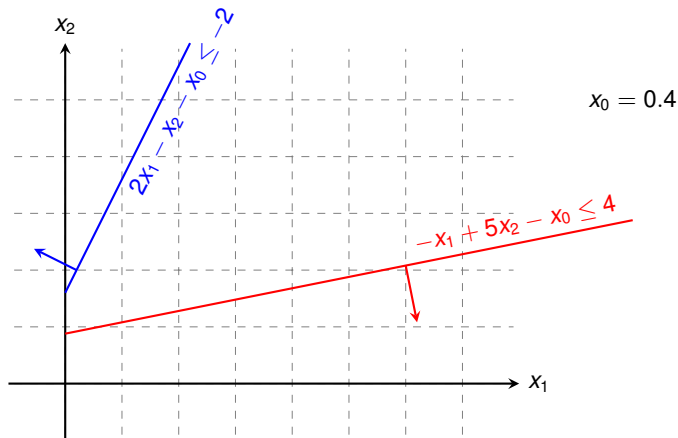
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

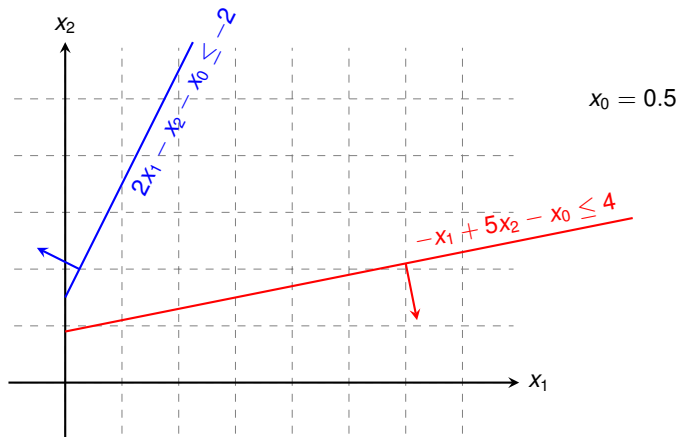
$$\begin{array}{rcccccccl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

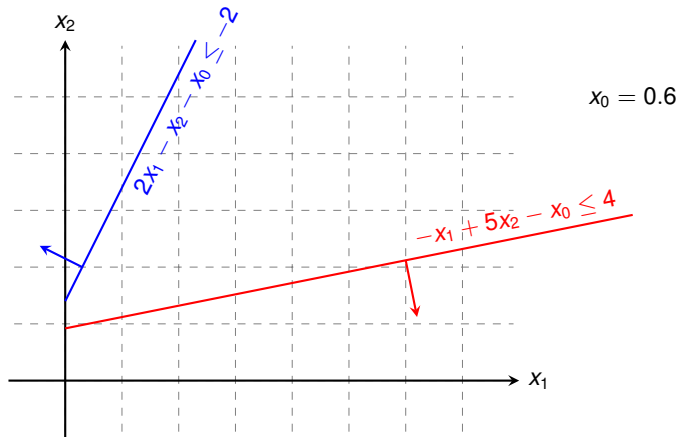
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

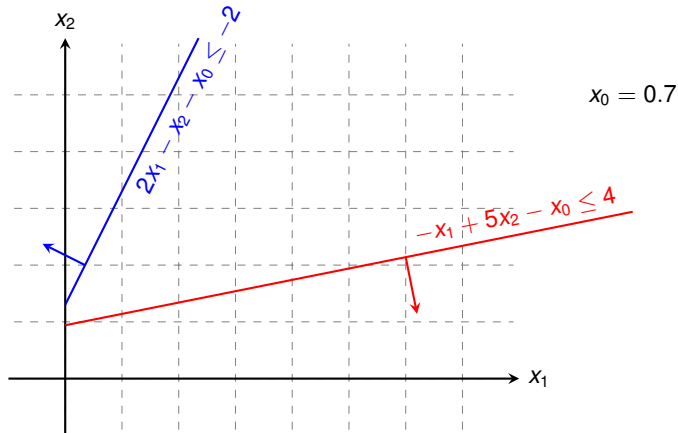
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

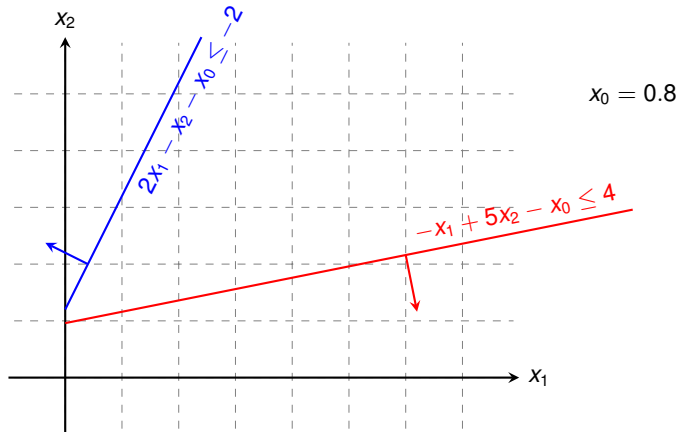
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

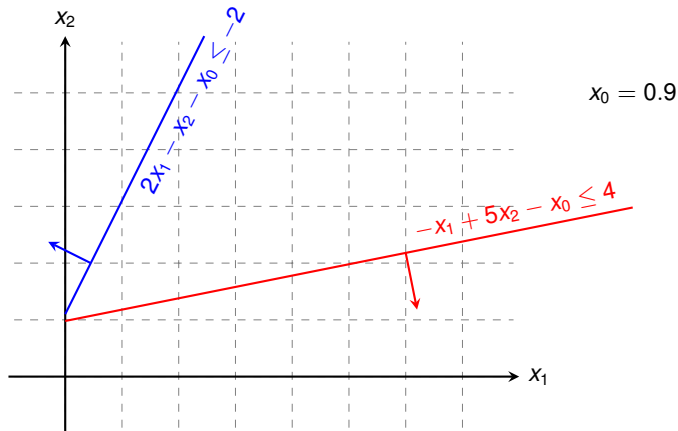
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

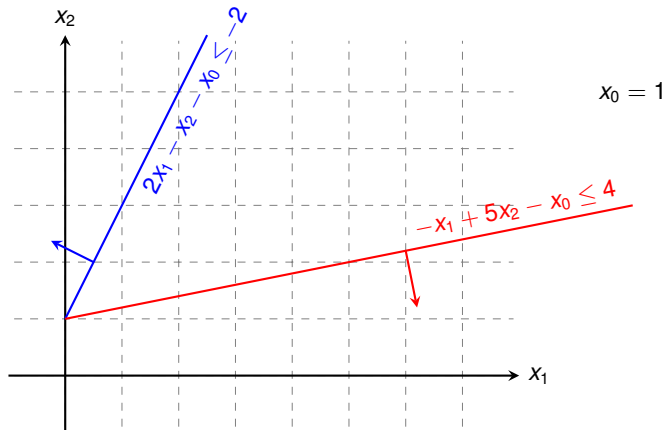
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

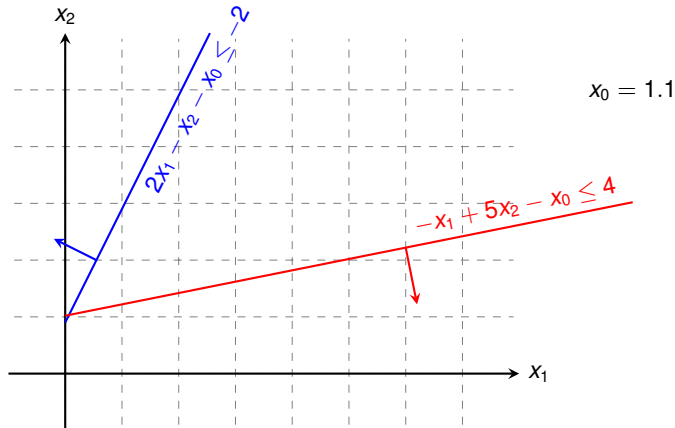
$$\begin{array}{rcccccccl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

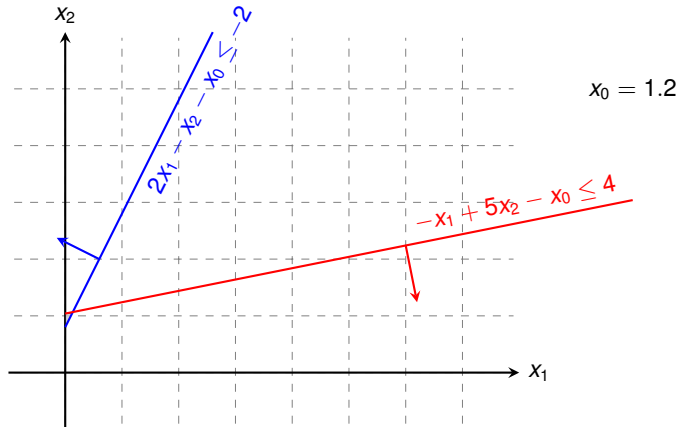
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

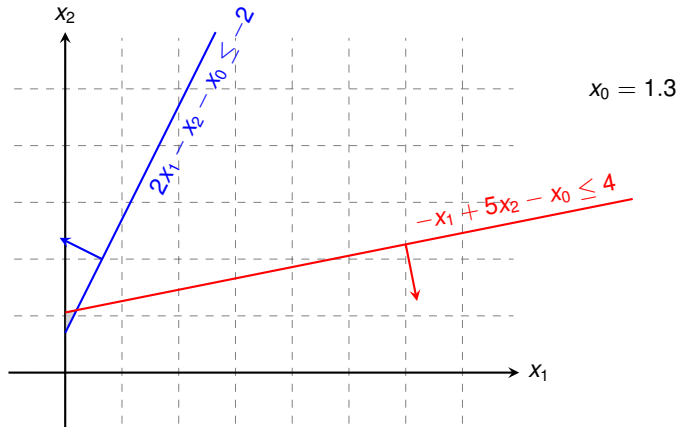
$$\begin{array}{rcccccccl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

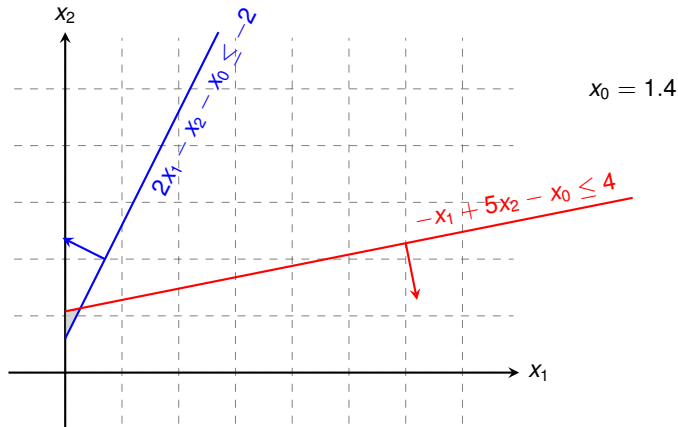
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

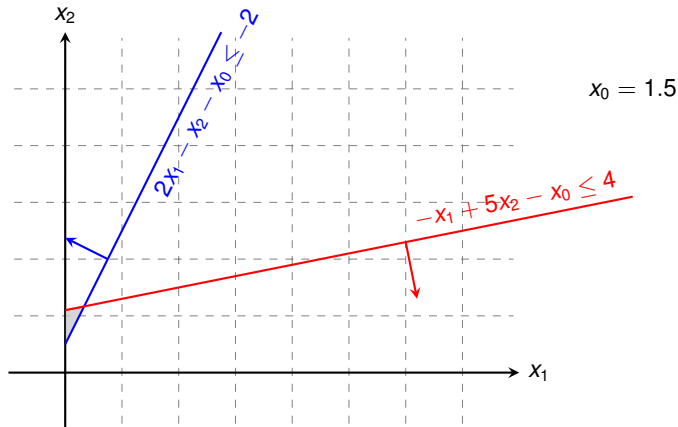
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

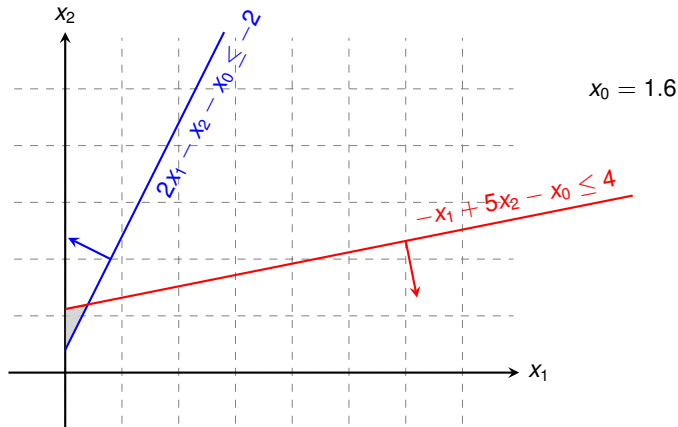
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

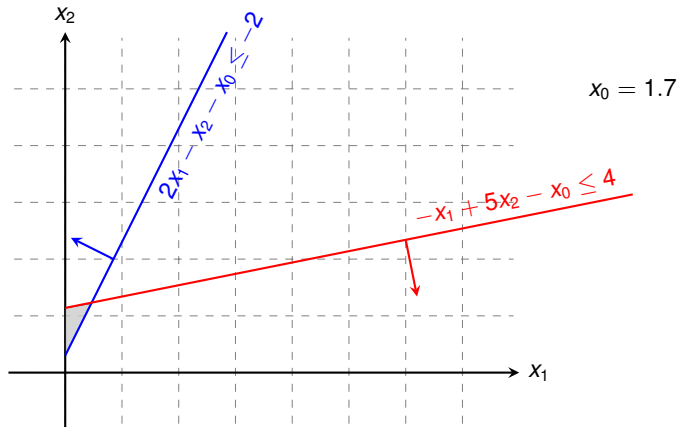
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

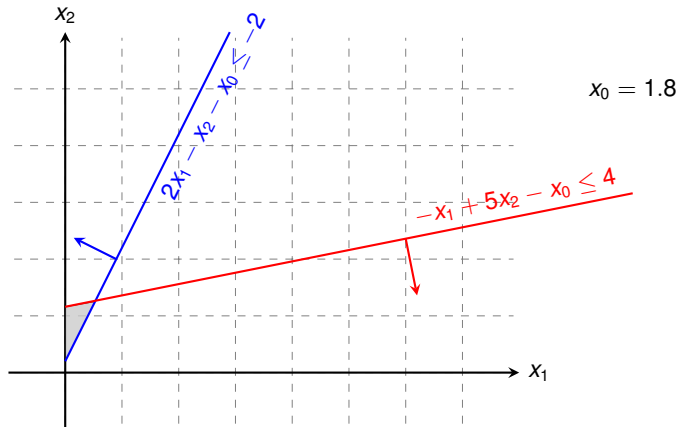
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

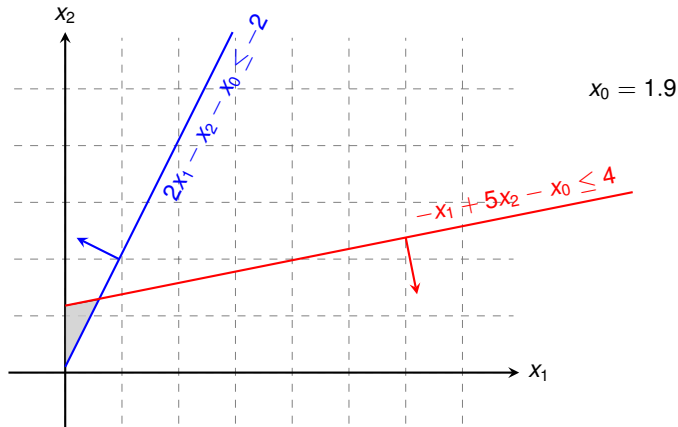
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

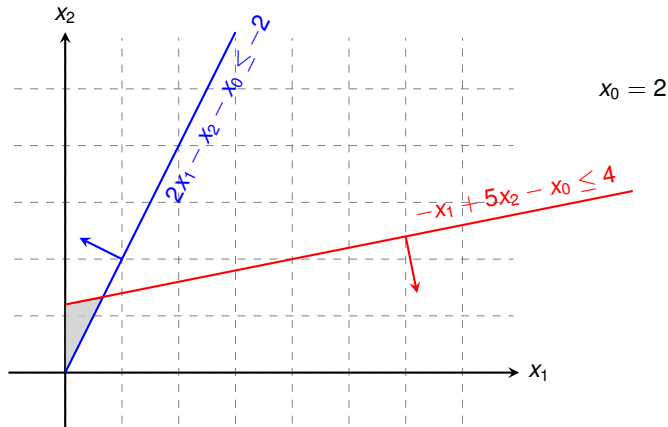
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

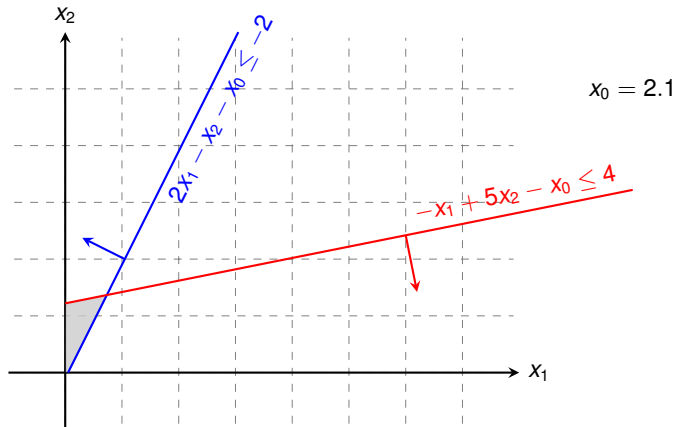
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

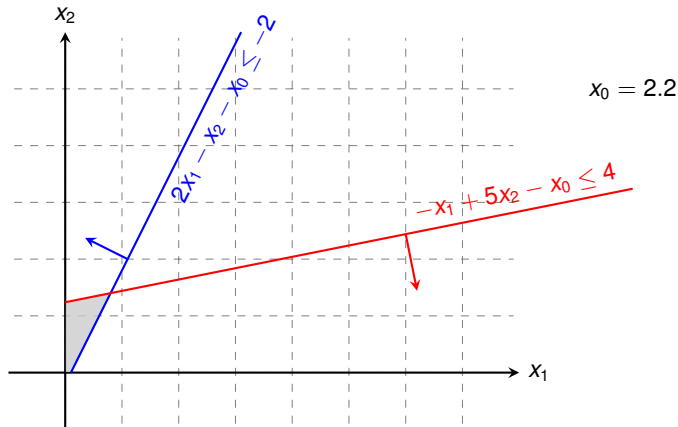
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

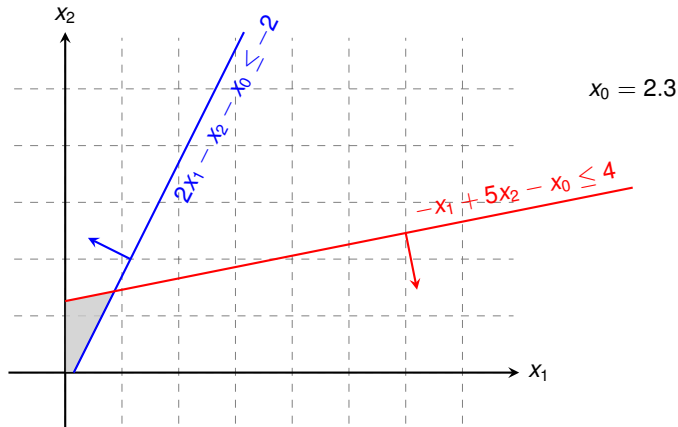
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

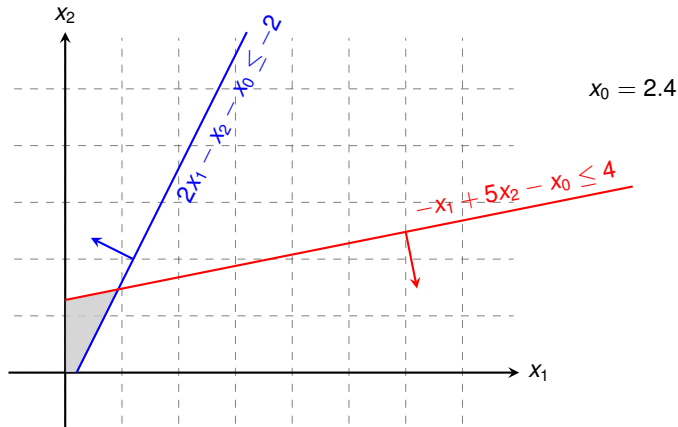
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

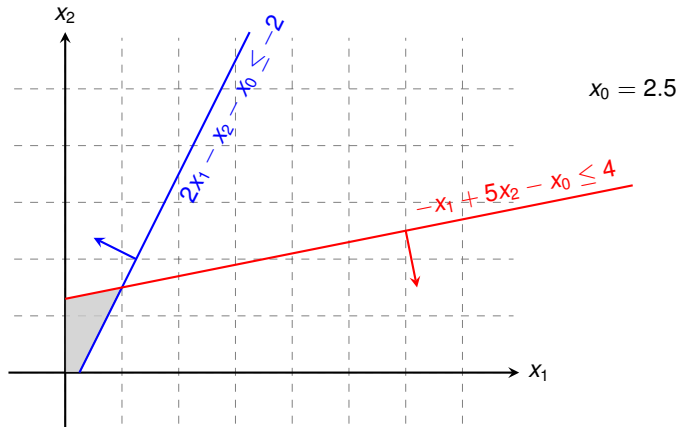
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

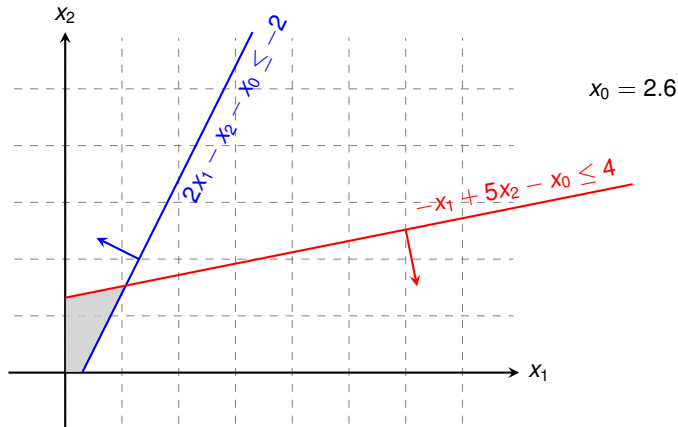
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

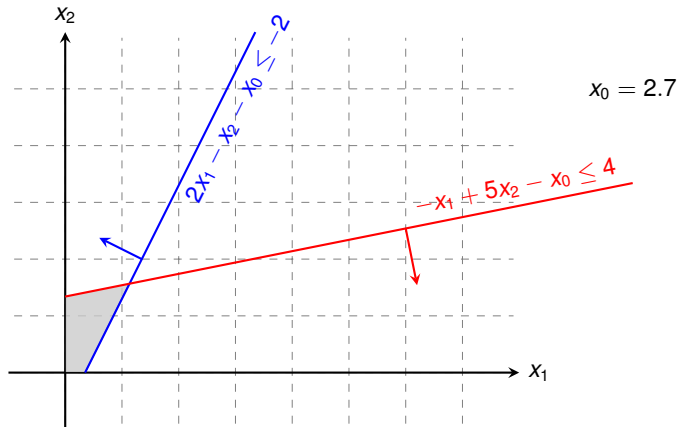
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

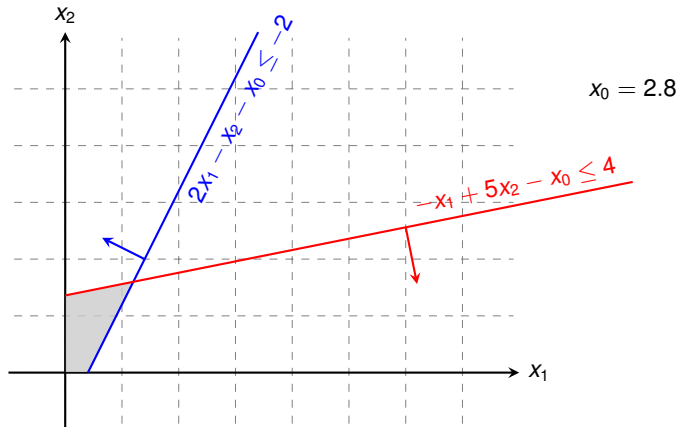
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

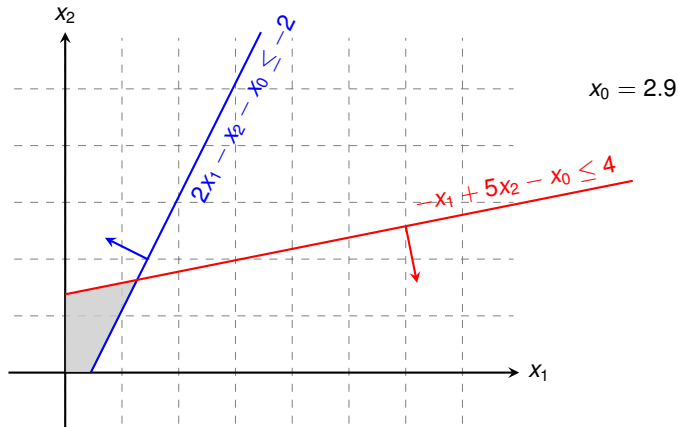
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

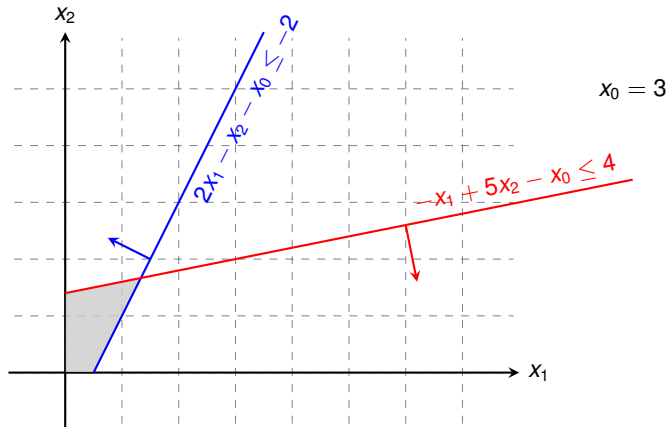
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise
subject to

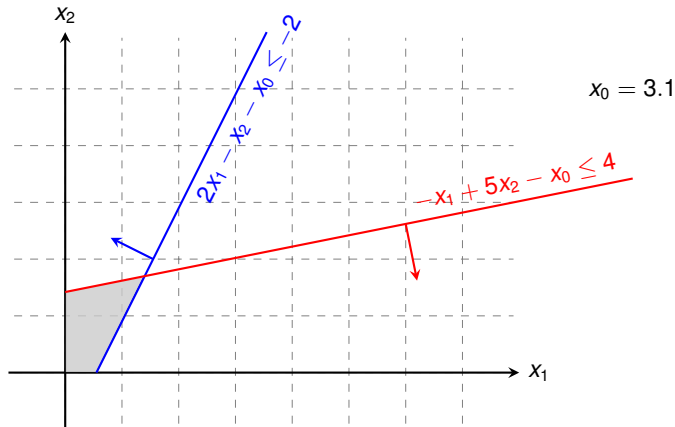
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

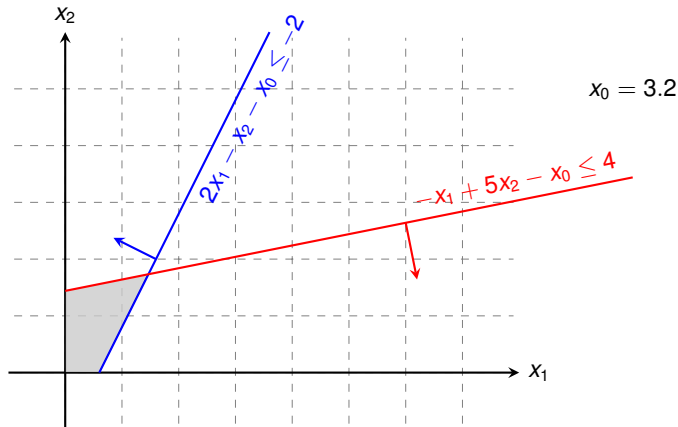
$$\begin{array}{rcccccccl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

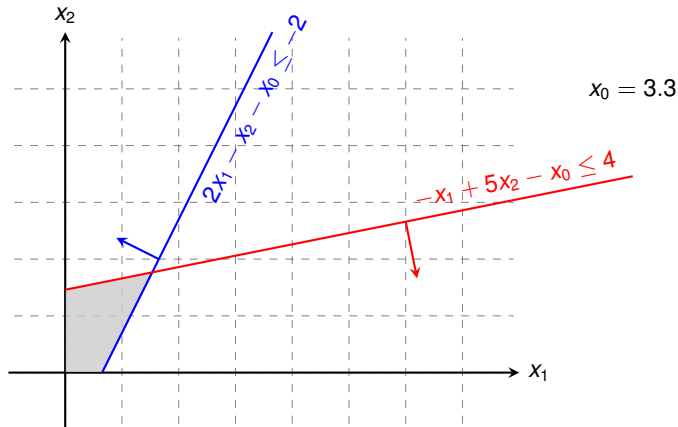
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

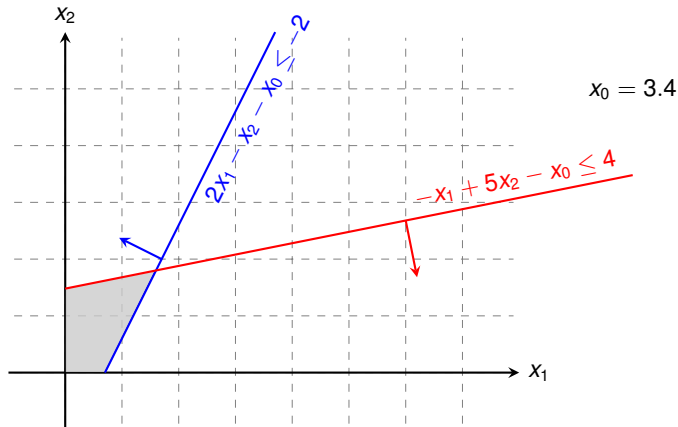
$$\begin{array}{rcccccccl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

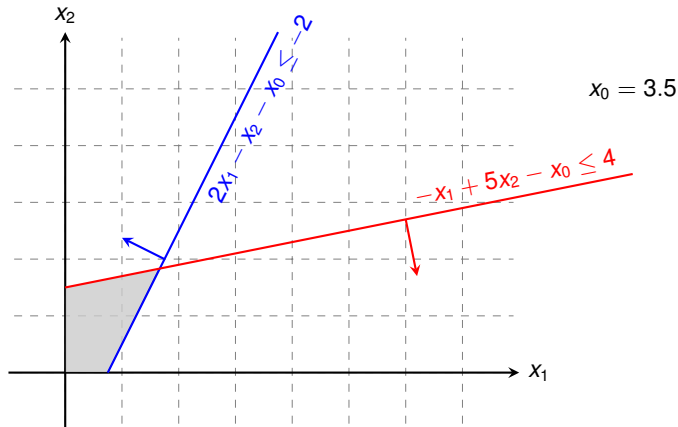
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

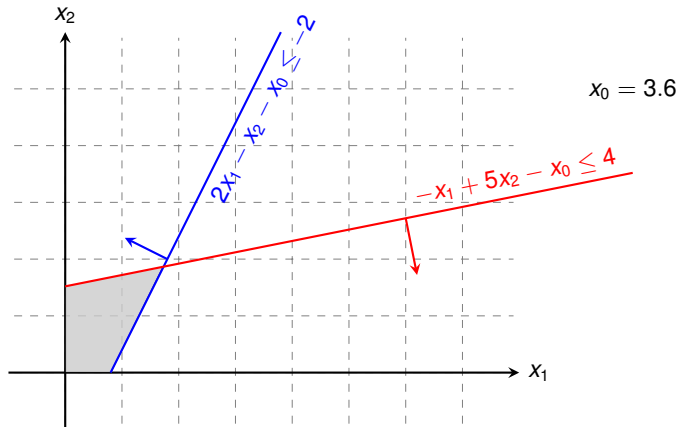
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

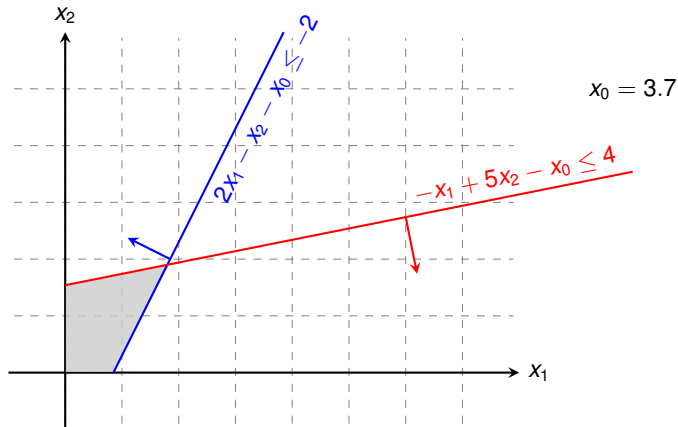
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

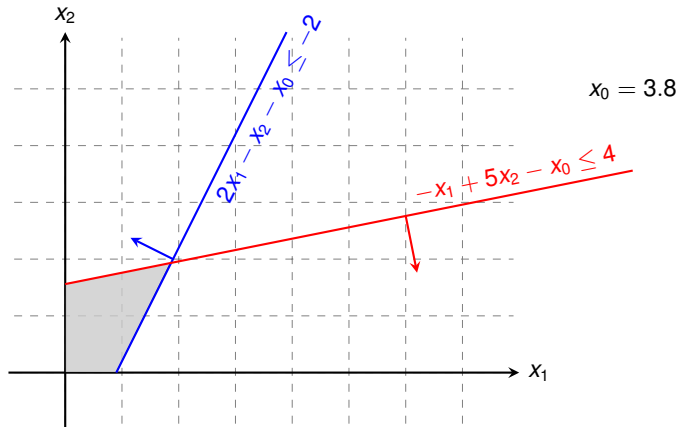
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

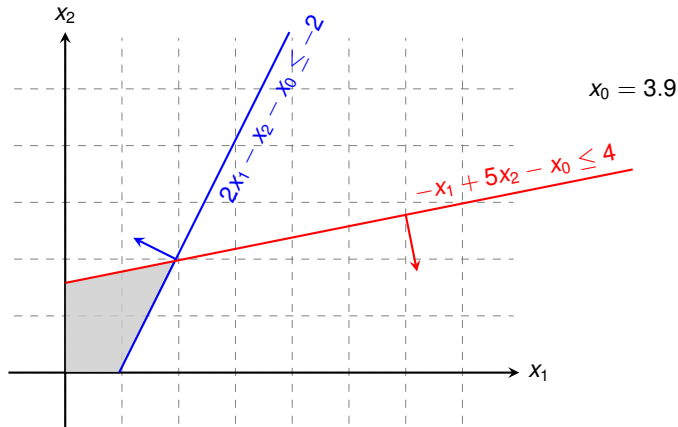
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

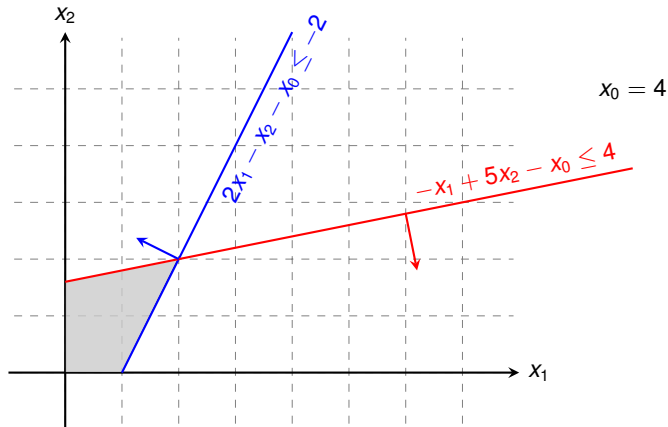
$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



Geometric Illustration

maximise $-x_0$
subject to

$$\begin{array}{rclclcl} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ x_0, x_1, x_2 & & & & & \geq & 0 \end{array}$$



INITIALIZE-SIMPLEX(A, b, c)

- 1 let k be the index of the minimum b_i
- 2 **if** $b_k \geq 0$ // is the initial basic solution feasible?
- 3 **return** ($\{1, 2, \dots, n\}, \{n+1, n+2, \dots, n+m\}, A, b, c, 0$)
- 4 form L_{aux} by adding $-x_0$ to the left-hand side of each constraint
and setting the objective function to $-x_0$
- 5 let (N, B, A, b, c, v) be the resulting slack form for L_{aux}
- 6 $l = n + k$
- 7 // L_{aux} has $n + 1$ nonbasic variables and m basic variables.
- 8 $(N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, 0)$
- 9 // The basic solution is now feasible for L_{aux} .
- 10 iterate the **while** loop of lines 3–12 of SIMPLEX until an optimal solution
to L_{aux} is found
- 11 **if** the optimal solution to L_{aux} sets \bar{x}_0 to 0
- 12 **if** \bar{x}_0 is basic
- 13 perform one (degenerate) pivot to make it nonbasic
- 14 from the final slack form of L_{aux} , remove x_0 from the constraints and
restore the original objective function of L , but replace each basic
variable in this objective function by the right-hand side of its
associated constraint
- 15 **return** the modified final slack form
- 16 **else return** “infeasible”

Test solution with $N = \{1, 2, \dots, n\}$, $B = \{n+1, n+2, \dots, n+m\}$, $\bar{x}_i = b_i$ for $i \in B$, $\bar{x}_i = 0$ otherwise.

INITIALIZE-SIMPLEX(A, b, c)

- 1 let k be the index of the minimum b_i
- 2 **if** $b_k \geq 0$ // is the initial basic solution feasible?
- 3 **return** ($\{1, 2, \dots, n\}, \{n+1, n+2, \dots, n+m\}, A, b, c, 0$)
- 4 form L_{aux} by adding $-x_0$ to the left-hand side of each constraint
and setting the objective function to $-x_0$
- 5 let (N, B, A, b, c, v) be the resulting slack form for L_{aux}
- 6 $l = n + k$
- 7 // L_{aux} has $n + 1$ nonbasic variables and m basic variables.
- 8 $(N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, 0)$
- 9 // The basic solution is now feasible for L_{aux} .
- 10 iterate the **while** loop of lines 3–12 of SIMPLEX until an optimal solution
to L_{aux} is found
- 11 **if** the optimal solution to L_{aux} sets \bar{x}_0 to 0
- 12 **if** \bar{x}_0 is basic
- 13 perform one (degenerate) pivot to make it nonbasic
- 14 from the final slack form of L_{aux} , remove x_0 from the constraints and
restore the original objective function of L , but replace each basic
variable in this objective function by the right-hand side of its
associated constraint
- 15 **return** the modified final slack form
- 16 **else return** “infeasible”

INITIALIZE-SIMPLEX(A, b, c)

- 1 let k be the index of the minimum b_i
- 2 **if** $b_k \geq 0$ // is the initial basic solution feasible?
- 3 **return** ($\{1, 2, \dots, n\}, \{n+1, n+2, \dots, n+m\}, A, b, c, 0$)
- 4 form L_{aux} by adding $-x_0$ to the left-hand side of each constraint
and setting the objective function to $-x_0$
- 5 let (N, B, A, b, c, v) be the resulting slack form for L_{aux}
- 6 $l = n + k$
- 7 // L_{aux} has $n + 1$ nonbasic variables and m basic variables.
- 8 $(N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, 0)$
- 9 // The basic solution is now feasible for L_{aux} .
- 10 iterate the **while** loop of lines 3–12 of SIMPLEX until an optimal solution
to L_{aux} is found
- 11 **if** the optimal solution to L_{aux} sets \bar{x}_0 to 0
- 12 **if** \bar{x}_0 is basic
- 13 perform one (degenerate) pivot to make it nonbasic
- 14 from the final slack form of L_{aux} , remove x_0 from the constraints and
restore the original objective function of L , but replace each basic
variable in this objective function by the right-hand side of its
associated constraint
- 15 **return** the modified final slack form
- 16 **else return** “infeasible”

Test solution with $N = \{1, 2, \dots, n\}$, $B = \{n+1, n+2, \dots, n+m\}$, $\bar{x}_i = b_i$ for $i \in B$, $\bar{x}_i = 0$ otherwise.

ℓ will be the leaving variable so
that x_ℓ has the most negative value.

INITIALIZE-SIMPLEX(A, b, c)

- 1 let k be the index of the minimum b_i
- 2 **if** $b_k \geq 0$ // is the initial basic solution feasible?
- 3 **return** ($\{1, 2, \dots, n\}, \{n+1, n+2, \dots, n+m\}, A, b, c, 0$)
- 4 form L_{aux} by adding $-x_0$ to the left-hand side of each constraint
and setting the objective function to $-x_0$
- 5 let (N, B, A, b, c, v) be the resulting slack form for L_{aux}
- 6 $l = n + k$
- 7 // L_{aux} has $n + 1$ nonbasic variables and m basic variables.
- 8 $(N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, 0)$
- 9 // The basic solution is now feasible for L_{aux} .
- 10 iterate the **while** loop of lines 3–12 of SIMPLEX until an optimal solution
to L_{aux} is found
- 11 **if** the optimal solution to L_{aux} sets \bar{x}_0 to 0
- 12 **if** \bar{x}_0 is basic
- 13 perform one (degenerate) pivot to make it nonbasic
- 14 from the final slack form of L_{aux} , remove x_0 from the constraints and
restore the original objective function of L , but replace each basic
variable in this objective function by the right-hand side of its
associated constraint
- 15 **return** the modified final slack form
- 16 **else return** “infeasible”

Test solution with $N = \{1, 2, \dots, n\}$, $B = \{n+1, n+2, \dots, n+m\}$, $\bar{x}_i = b_i$ for $i \in B$, $\bar{x}_i = 0$ otherwise.

ℓ will be the leaving variable so that x_ℓ has the most negative value.

Pivot step with x_ℓ leaving and x_0 entering.

INITIALIZE-SIMPLEX(A, b, c)

- 1 let k be the index of the minimum b_i
- 2 **if** $b_k \geq 0$ // is the initial basic solution feasible?
- 3 **return** ($\{1, 2, \dots, n\}, \{n+1, n+2, \dots, n+m\}, A, b, c, 0$)
- 4 form L_{aux} by adding $-x_0$ to the left-hand side of each constraint
and setting the objective function to $-x_0$
- 5 let (N, B, A, b, c, v) be the resulting slack form for L_{aux}
- 6 $l = n + k$
- 7 // L_{aux} has $n + 1$ nonbasic variables and m basic variables.
- 8 $(N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, 0)$
- 9 // The basic solution is now feasible for L_{aux} .
- 10 **iterate the while loop** of lines 3–12 of SIMPLEX until an optimal solution
to L_{aux} is found
- 11 **if** the optimal solution to L_{aux} sets \bar{x}_0 to 0
- 12 **if** \bar{x}_0 is basic
- 13 perform one (degenerate) pivot to make it nonbasic
- 14 from the final slack form of L_{aux} , remove x_0 from the constraints and
restore the original objective function of L , but replace each basic
variable in this objective function by the right-hand side of its
associated constraint
- 15 **return** the modified final slack form
- 16 **else return** “infeasible”

Test solution with $N = \{1, 2, \dots, n\}$, $B = \{n+1, n+2, \dots, n+m\}$, $\bar{x}_i = b_i$ for $i \in B$, $\bar{x}_i = 0$ otherwise.

ℓ will be the leaving variable so that x_ℓ has the most negative value.

Pivot step with x_ℓ leaving and x_0 entering.

This pivot step does not change the value of any variable.

Example of INITIALIZE-SIMPLEX (1/3)

$$\begin{array}{llllll} \text{maximise} & 2x_1 & - & x_2 & & \\ \text{subject to} & & & & & \\ & 2x_1 & - & x_2 & \leq & 2 \\ & x_1 & - & 5x_2 & \leq & -4 \\ & & & x_1, x_2 & \geq & 0 \end{array}$$

Example of INITIALIZE-SIMPLEX (1/3)

maximise
subject to

$$2x_1 - x_2$$

$$2x_1 - x_2 \leq 2$$

$$x_1 - 5x_2 \leq -4$$

$$x_1, x_2 \geq 0$$

↓ Formulating the auxiliary linear program
(as basic solution would not be feasible!)

Example of INITIALIZE-SIMPLEX (1/3)

maximise
subject to

$$\begin{array}{rclcl} 2x_1 & - & x_2 & & \\ x_1 & - & 5x_2 & & \\ x_1, x_2 & & & & \end{array} \begin{array}{l} \\ \\ \geq \end{array} \begin{array}{l} 2 \\ -4 \\ 0 \end{array}$$

↓ Formulating the auxiliary linear program
(as basic solution would not be feasible!)

maximise
subject to

$$\begin{array}{rclclcl} & & & - & x_0 & & \\ 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_1, x_2, x_0 & & & & & \geq & 0 \end{array}$$

Example of INITIALIZE-SIMPLEX (1/3)

maximise
subject to

$$\begin{array}{rclcl} 2x_1 & - & x_2 & & \\ 2x_1 & - & x_2 & \leq & 2 \\ x_1 & - & 5x_2 & \leq & -4 \\ x_1, x_2 & & & \geq & 0 \end{array}$$

↓ Formulating the auxiliary linear program
(as basic solution would not be feasible!)

maximise
subject to

$$\begin{array}{rclclcl} & & & - & x_0 & \\ 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ x_1, x_2, x_0 & & & & & \geq & 0 \end{array}$$

↓ Converting into slack form

Example of INITIALIZE-SIMPLEX (1/3)

$$\begin{array}{llll}
 \text{maximise} & 2x_1 & - & x_2 \\
 \text{subject to} & & & \\
 & 2x_1 & - & x_2 \leq 2 \\
 & x_1 & - & 5x_2 \leq -4 \\
 & x_1, x_2 & & \geq 0
 \end{array}$$

\downarrow Formulating the auxiliary linear program
 (as basic solution would not be feasible!)

$$\begin{array}{llllll}
 \text{maximise} & & & & - & x_0 \\
 \text{subject to} & & & & & \\
 & 2x_1 & - & x_2 & - & x_0 \leq 2 \\
 & x_1 & - & 5x_2 & - & x_0 \leq -4 \\
 & x_1, x_2, x_0 & & & & \geq 0
 \end{array}$$

\downarrow Converting into slack form

$$\begin{array}{llllllll}
 Z & = & & & & & - & x_0 \\
 x_3 & = & 2 & - & 2x_1 & + & x_2 & + x_0 \\
 x_4 & = & -4 & - & x_1 & + & 5x_2 & + x_0
 \end{array}$$

Example of INITIALIZE-SIMPLEX (1/3)

$$\begin{array}{llll} \text{maximise} & 2x_1 & - & x_2 \\ \text{subject to} & & & \\ & 2x_1 & - & x_2 \leq 2 \\ & x_1 & - & 5x_2 \leq -4 \\ & x_1, x_2 & & \geq 0 \end{array}$$

Formulating the auxiliary linear program
(as basic solution would not be feasible!)

$$\begin{array}{llllll} \text{maximise} & & & & - & x_0 \\ \text{subject to} & & & & & \\ & 2x_1 & - & x_2 & - & x_0 \leq 2 \\ & x_1 & - & 5x_2 & - & x_0 \leq -4 \\ & x_1, x_2, x_0 & & & & \geq 0 \end{array}$$

Basic solution
(0, 0, 0, 2, -4) not feasible!

Converting into slack form

$$\begin{array}{rclllll} Z & = & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 + x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 + x_0 \end{array}$$

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rclclclcl} Z & = & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rclclclcl} Z & = & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$



Pivot with x_0 entering and x_4 leaving

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rcllclclcl} Z & = & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$



Pivot with x_0 entering and x_4 leaving

$$\begin{array}{rcllclclcl} Z & = & -4 & - & x_1 & + & 5x_2 & - & x_4 \\ x_0 & = & 4 & + & x_1 & - & 5x_2 & + & x_4 \\ x_3 & = & 6 & - & x_1 & - & 4x_2 & + & x_4 \end{array}$$

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rcllclclcl} Z & = & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$



Pivot with x_0 entering and x_4 leaving

$$\begin{array}{rcllclclcl} Z & = & -4 & - & x_1 & + & 5x_2 & - & x_4 \\ x_0 & = & 4 & + & x_1 & - & 5x_2 & + & x_4 \\ x_3 & = & 6 & - & x_1 & - & 4x_2 & + & x_4 \end{array}$$

Basic solution (4, 0, 0, 6, 0) is feasible!

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rcllclclcl} Z & = & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$

↓
Pivot with x_0 entering and x_4 leaving

$$\begin{array}{rcllclclcl} Z & = & -4 & - & x_1 & + & 5x_2 & - & x_4 \\ x_0 & = & 4 & + & x_1 & - & 5x_2 & + & x_4 \\ x_3 & = & 6 & - & x_1 & - & 4x_2 & + & x_4 \end{array}$$

Basic solution (4, 0, 0, 6, 0) is feasible!

↓
Pivot with x_2 entering and x_0 leaving

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rclclclcl} Z & = & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$

↓
Pivot with x_0 entering and x_4 leaving

$$\begin{array}{rclclclcl} Z & = & -4 & - & x_1 & + & 5x_2 & - & x_4 \\ x_0 & = & 4 & + & x_1 & - & 5x_2 & + & x_4 \\ x_3 & = & 6 & - & x_1 & - & 4x_2 & + & x_4 \end{array}$$

Basic solution (4, 0, 0, 6, 0) is feasible!

↓
Pivot with x_2 entering and x_0 leaving

$$\begin{array}{rclclclcl} Z & = & & - & x_0 & & & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rcllclclcl} Z & = & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$

Pivot with x_0 entering and x_4 leaving

$$\begin{array}{rcllclclcl} Z & = & -4 & - & x_1 & + & 5x_2 & - & x_4 \\ x_0 & = & 4 & + & x_1 & - & 5x_2 & + & x_4 \\ x_3 & = & 6 & - & x_1 & - & 4x_2 & + & x_4 \end{array}$$

Basic solution (4, 0, 0, 6, 0) is feasible!

Pivot with x_2 entering and x_0 leaving

$$\begin{array}{rcllclclcl} Z & = & & - & x_0 \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

Optimal solution has $x_0 = 0$, hence the initial problem was feasible!

Example of INITIALIZE-SIMPLEX (3/3)

$$\begin{array}{rclclclcl} Z & = & & - & x_0 & & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} & + \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} & + \frac{x_4}{5} \end{array}$$

Example of INITIALIZE-SIMPLEX (3/3)

$$\begin{array}{rclclclcl} Z & = & & - & x_0 & & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} & + \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} & + \frac{x_4}{5} \end{array}$$



Set $x_0 = 0$ and express objective function by non-basic variables

Example of INITIALIZE-SIMPLEX (3/3)

$$\begin{array}{rclclcl} Z & = & & - & x_0 & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} + \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} + \frac{x_4}{5} \end{array}$$

$$2x_1 - x_2 = 2x_1 - \left(\frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \right)$$

Set $x_0 = 0$ and express objective function by non-basic variables

$$\begin{array}{rclclcl} Z & = & -\frac{4}{5} & + & \frac{9x_1}{5} & - & \frac{x_4}{5} \\ x_2 & = & \frac{4}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

Example of INITIALIZE-SIMPLEX (3/3)

$$\begin{array}{rclclcl} Z & = & & - & x_0 & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} + \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} + \frac{x_4}{5} \end{array}$$

$$2x_1 - x_2 = 2x_1 - \left(\frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5}\right)$$

Set $x_0 = 0$ and express objective function by non-basic variables

$$\begin{array}{rclclcl} Z & = & -\frac{4}{5} & + & \frac{9x_1}{5} & - & \frac{x_4}{5} \\ x_2 & = & \frac{4}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

Basic solution $(0, \frac{4}{5}, \frac{14}{5}, 0)$, which is feasible!

Example of INITIALIZE-SIMPLEX (3/3)

$$\begin{array}{rclclcl} Z & = & & - & x_0 & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} + \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} + \frac{x_4}{5} \end{array}$$

$$2x_1 - x_2 = 2x_1 - \left(\frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5}\right)$$

Set $x_0 = 0$ and express objective function by non-basic variables

$$\begin{array}{rclclcl} Z & = & -\frac{4}{5} & + & \frac{9x_1}{5} & - & \frac{x_4}{5} \\ x_2 & = & \frac{4}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

Basic solution $(0, \frac{4}{5}, \frac{14}{5}, 0)$, which is feasible!

Lemma 29.12

If a linear program L has no feasible solution, then INITIALIZE-SIMPLEX returns “infeasible”. Otherwise, it returns a valid slack form for which the basic solution is feasible.

Fundamental Theorem of Linear Programming

Theorem 29.13 (Fundamental Theorem of Linear Programming)

For any linear program L , given in standard form, either:

1. L is infeasible \Rightarrow SIMPLEX returns “infeasible”.
2. L is unbounded \Rightarrow SIMPLEX returns “unbounded”.
3. L has an optimal solution with a finite objective value
 \Rightarrow SIMPLEX returns an optimal solution with a finite objective value.

Small Technicality: need to equip SIMPLEX with an “anti-cycling strategy” (see extra slides)

Fundamental Theorem of Linear Programming

Theorem 29.13 (Fundamental Theorem of Linear Programming)

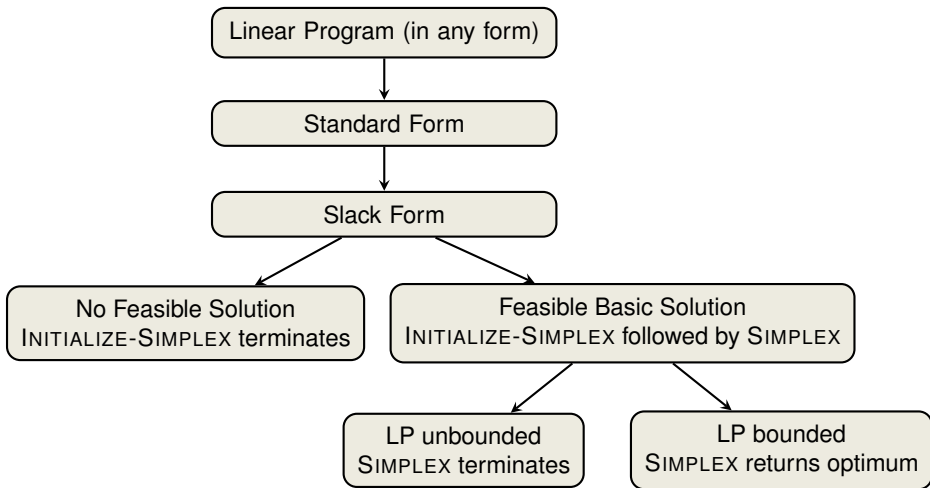
For any linear program L , given in standard form, either:

1. L is infeasible \Rightarrow SIMPLEX returns “infeasible”.
2. L is unbounded \Rightarrow SIMPLEX returns “unbounded”.
3. L has an optimal solution with a finite objective value
 \Rightarrow SIMPLEX returns an optimal solution with a finite objective value.

Small Technicality: need to equip SIMPLEX with an “anti-cycling strategy” (see extra slides)

Proof requires the concept of **duality**, which is not covered in this course (for details see CLRS3, Chapter 29.4)

Workflow for Solving Linear Programs



Linear Programming and Simplex: Summary and Outlook

Linear Programming



Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds

Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of [Integer Programming](#), to be discussed in later lectures

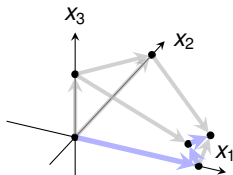
Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of **Integer Programming**, to be discussed in later lectures

Simplex Algorithm

- **In practice**: usually terminates in polynomial time, i.e., $O(m + n)$



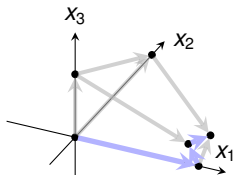
Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of **Integer Programming**, to be discussed in later lectures

Simplex Algorithm

- **In practice**: usually terminates in polynomial time, i.e., $O(m + n)$
- **In theory**: even with anti-cycling may need exponential time



Linear Programming and Simplex: Summary and Outlook

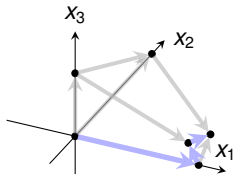
Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of **Integer Programming**, to be discussed in later lectures

Simplex Algorithm

- **In practice**: usually terminates in polynomial time, i.e., $O(m + n)$
- **In theory**: even with anti-cycling may need exponential time

Research Problem: Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?



Linear Programming and Simplex: Summary and Outlook

Linear Programming

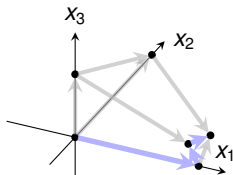
- extremely versatile tool for modelling problems of all kinds
- basis of **Integer Programming**, to be discussed in later lectures

Simplex Algorithm

- **In practice**: usually terminates in polynomial time, i.e., $O(m + n)$
- **In theory**: even with anti-cycling may need exponential time

Research Problem: Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?

Polynomial-Time Algorithms



Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of **Integer Programming**, to be discussed in later lectures

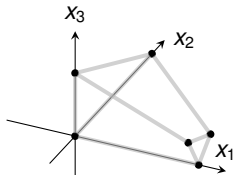
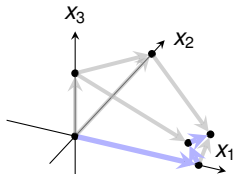
Simplex Algorithm

- **In practice**: usually terminates in polynomial time, i.e., $O(m + n)$
- **In theory**: even with anti-cycling may need exponential time

Research Problem: Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?

Polynomial-Time Algorithms

- **Interior-Point Methods**: traverses the interior of the feasible set of solutions (not just vertices!)



Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of **Integer Programming**, to be discussed in later lectures

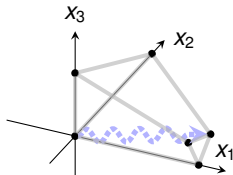
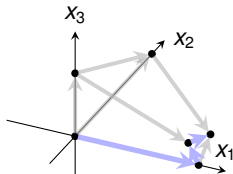
Simplex Algorithm

- **In practice**: usually terminates in polynomial time, i.e., $O(m + n)$
- **In theory**: even with anti-cycling may need exponential time

Research Problem: Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?

Polynomial-Time Algorithms

- **Interior-Point Methods**: traverses the interior of the feasible set of solutions (not just vertices!)



1.2 Famous Failures and the Need for Alternatives

For many problems a bit beyond the scope of an undergraduate course, the downside of worst-case analysis rears its ugly head. This section reviews four famous examples in which worst-case analysis gives misleading or useless advice about how to solve a problem. These examples motivate the alternatives to worst-case analysis that are surveyed in Section 1.4 and described in detail in later chapters of the book.

1.2.1 The Simplex Method for Linear Programming

Perhaps the most famous failure of worst-case analysis concerns linear programming, the problem of optimizing a linear function subject to linear constraints (Figure 1.1). Dantzig proposed in the 1940s an algorithm for solving linear programs called the *simplex method*. The simplex method solves linear programs using greedy local

Source: "Beyond the Worst-Case Analysis of Algorithms" by Tim Roughgarden, 2020

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

$$Z = \quad \quad \quad x_1 + x_2 + x_3$$

$$x_4 = 8 - x_1 - x_2$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

$$Z = \quad \quad \quad x_1 \quad + \quad x_2 \quad + \quad x_3$$

$$x_4 = 8 \quad - \quad x_1 \quad - \quad x_2$$

$$x_5 = \quad \quad \quad \quad \quad x_2 \quad - \quad x_3$$

↓ Pivot with x_1 entering and x_4 leaving

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

$$Z = \quad \quad \quad x_1 + x_2 + x_3$$

$$x_4 = 8 - x_1 - x_2$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

↓ Pivot with x_1 entering and x_4 leaving

$$Z = 8 \quad \quad \quad + x_3 - x_4$$

$$x_1 = 8 - x_2 \quad \quad \quad - x_4$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

$$Z = \quad \quad \quad x_1 + x_2 + x_3$$

$$x_4 = 8 - x_1 - x_2$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

↓ Pivot with x_1 entering and x_4 leaving

$$Z = 8 \quad \quad \quad + x_3 - x_4$$

$$x_1 = 8 - x_2 \quad \quad \quad - x_4$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

↓ Pivot with x_3 entering and x_5 leaving

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

$$Z = \quad \quad \quad x_1 + x_2 + x_3$$

$$x_4 = 8 - x_1 - x_2$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

↓ Pivot with x_1 entering and x_4 leaving

$$Z = 8 \quad \quad \quad + x_3 - x_4$$

$$x_1 = 8 - x_2 \quad \quad \quad - x_4$$

$$x_5 = \quad \quad \quad x_2 - x_3$$

↓ Pivot with x_3 entering and x_5 leaving

$$Z = 8 + x_2 - x_4 - x_5$$

$$x_1 = 8 - x_2 - x_4$$

$$x_3 = \quad \quad \quad x_2 \quad \quad \quad - x_5$$

Termination

Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

$$\begin{array}{rclclcl} Z & = & & x_1 & + & x_2 & + & x_3 \\ x_4 & = & 8 & - & x_1 & - & x_2 & \\ x_5 & = & & & & x_2 & - & x_3 \end{array}$$

↓ Pivot with x_1 entering and x_4 leaving

$$\begin{array}{rclclcl} Z & = & 8 & & + & x_3 & - & x_4 \\ x_1 & = & 8 & - & x_2 & & - & x_4 \\ x_5 & = & & x_2 & - & x_3 & & \end{array}$$

Cycling: If additionally slack form at two iterations are identical, SIMPLEX fails to terminate!

↓ Pivot with x_3 entering and x_5 leaving

$$\begin{array}{rclclcl} Z & = & 8 & + & x_2 & - & x_4 & - & x_5 \\ x_1 & = & 8 & - & x_2 & - & x_4 & \\ x_3 & = & & x_2 & & & - & x_5 \end{array}$$



Exercise: Execute one more step of the Simplex Algorithm on the tableau from the previous slide.

Cycling: SIMPLEX may fail to terminate.

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies



Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies

1. **Bland's rule:** Choose entering variable with smallest index

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies

1. **Bland's rule:** Choose entering variable with smallest index
2. **Random rule:** Choose entering variable uniformly at random

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

— Anti-Cycling Strategies —

1. **Bland's rule:** Choose entering variable with smallest index
2. **Random rule:** Choose entering variable uniformly at random
3. **Perturbation:** Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies

1. **Bland's rule:** Choose entering variable with smallest index
2. **Random rule:** Choose entering variable uniformly at random
3. **Perturbation:** Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Replace each b_i by $\hat{b}_i = b_i + \epsilon_i$, where $\epsilon_i \gg \epsilon_{i+1}$ are all small.

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies

1. **Bland's rule:** Choose entering variable with smallest index
2. **Random rule:** Choose entering variable uniformly at random
3. **Perturbation:** Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Replace each b_i by $\hat{b}_i = b_i + \epsilon_i$, where $\epsilon_i \gg \epsilon_{i+1}$ are all small.

Lemma 29.7

Assuming INITIALIZE-SIMPLEX returns a slack form for which the basic solution is feasible, SIMPLEX either reports that the program is unbounded or returns a feasible solution in at most $\binom{n+m}{m}$ iterations.

Termination and Running Time

It is theoretically possible, but very rare in practice.

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies

1. **Bland's rule:** Choose entering variable with smallest index
2. **Random rule:** Choose entering variable uniformly at random
3. **Perturbation:** Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Replace each b_i by $\hat{b}_i = b_i + \epsilon_i$, where $\epsilon_i \gg \epsilon_{i+1}$ are all small.

Lemma 29.7

Assuming INITIALIZE-SIMPLEX returns a slack form for which the basic solution is feasible, SIMPLEX either reports that the program is unbounded or returns a feasible solution in at most $\binom{n+m}{m}$ iterations.

Every set B of basic variables uniquely determines a slack form, and there are at most $\binom{n+m}{m}$ unique slack forms.

Randomised Algorithms

Lecture 8: Solving a TSP Instance using Linear Programming

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



Outline

Introduction

Examples of TSP Instances

Demonstration

The Traveling Salesman Problem (TSP)

*Given a set of **cities** along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.*

The Traveling Salesman Problem (TSP)

*Given a set of **cities** along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.*

Formal Definition



The Traveling Salesman Problem (TSP)

*Given a set of **cities** along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.*

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$

The Traveling Salesman Problem (TSP)

*Given a set of **cities** along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.*

Formal Definition

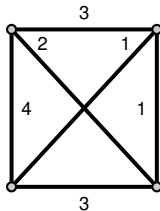
- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

The Traveling Salesman Problem (TSP)

*Given a set of **cities** along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.*

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

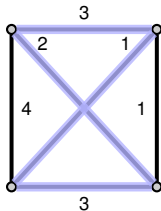


The Traveling Salesman Problem (TSP)

*Given a set of **cities** along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.*

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.



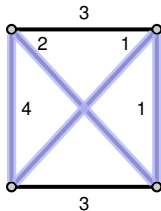
$$3 + 2 + 1 + 3 = 9$$

The Traveling Salesman Problem (TSP)

Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.



$$2 + 4 + 1 + 1 = 8$$

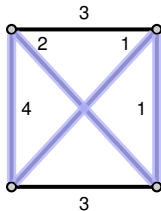
The Traveling Salesman Problem (TSP)

Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!



$$2 + 4 + 1 + 1 = 8$$

The Traveling Salesman Problem (TSP)

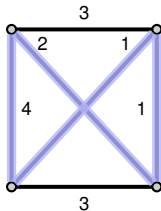
Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n - 1)!/2$



$$2 + 4 + 1 + 1 = 8$$

The Traveling Salesman Problem (TSP)

Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

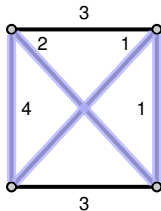
Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n - 1)!/2$

Special Instances



$$2 + 4 + 1 + 1 = 8$$

The Traveling Salesman Problem (TSP)

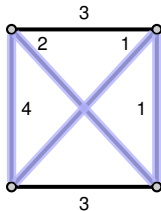
Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n - 1)!/2$



$$2 + 3 + 1 + 1 = 7$$

Special Instances

- **Metric TSP:** costs satisfy triangle inequality:

$$\forall u, v, w \in V: \quad c(u, w) \leq c(u, v) + c(v, w).$$

The Traveling Salesman Problem (TSP)

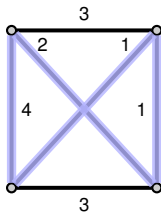
Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n - 1)!/2$



$$2 + 4 + 1 + 1 = 8$$

Special Instances

- **Metric TSP:** costs satisfy triangle inequality:

$$\forall u, v, w \in V: \quad c(u, w) \leq c(u, v) + c(v, w).$$

Even this version is NP hard (Ex. 35.2-2)

The Traveling Salesman Problem (TSP)

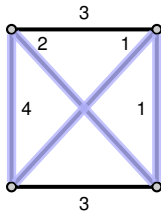
Given a set of *cities* along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- **Given:** A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- **Goal:** Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n - 1)!/2$



$$2 + 4 + 1 + 1 = 8$$

Special Instances

- **Metric TSP:** costs satisfy triangle inequality:

$$\forall u, v, w \in V: \quad c(u, w) \leq c(u, v) + c(v, w).$$

Even this version is NP hard (Ex. 35.2-2)

- **Euclidean TSP:** cities are points in the Euclidean space, costs are equal to their (rounded) Euclidean distance

Outline

Introduction

Examples of TSP Instances

Demonstration

33 city contest (1964)

HELP! WE'RE LOST!

HELP "CAR 54"...AND WIN CASH
54...\$1,000 PRIZES
ONE...\$10,000 GRAND PRIZE

START
FINISH

Map by Rand McNally

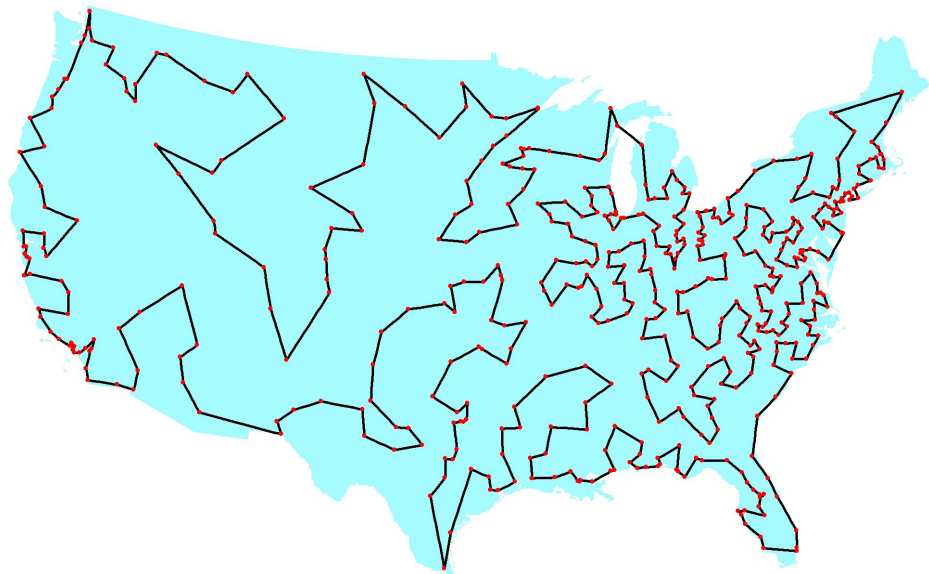
Help Toody and Muldoon find the shortest round trip route to visit all 33 locations shown on the map.
All you do is draw connecting straight lines from location to location to show the shortest round trip route.

HERE'S THE CORRECT START...
Begin at Chicago, Illinois. From there, lines show correct route as far as Erie, Pennsylvania. Next, do you go to Carlisle, Pennsylvania or Wana, West Virginia? Check the easy instructions on back of this entry blank for details.

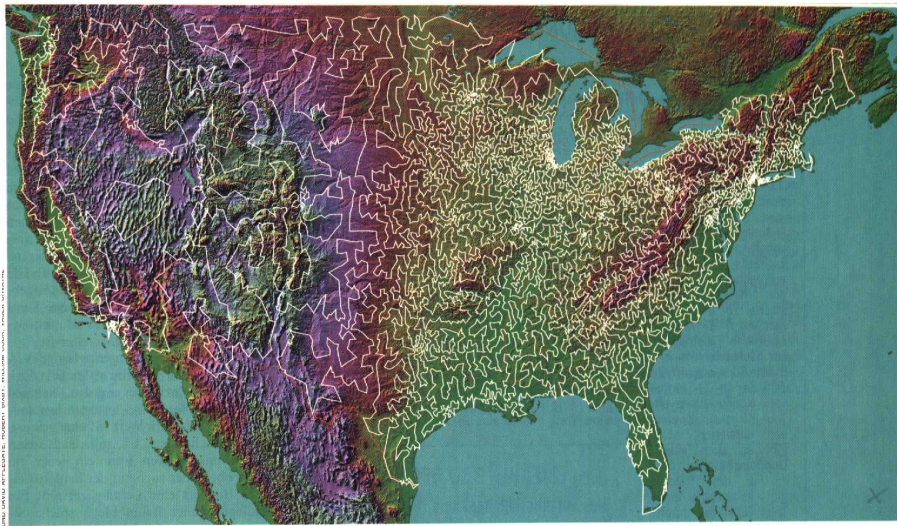
© PROCTER & GAMBLE 1962

OFFICIAL RULES ON REVERSE SIDE

532 cities (1987 [Padberg, Rinaldi])



13,509 cities (1999 [Applegate, Bixby, Chavatal, Cook])



SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM*

G. DANTZIG, R. FULKERSON, AND S. JOHNSON

The Rand Corporation, Santa Monica, California

(Received August 9, 1954)

It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington, D. C., has the shortest road distance.

THE TRAVELING-SALESMAN PROBLEM might be described as follows: Find the shortest route (tour) for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the original point of departure. More generally, given an n by n symmetric matrix $D=(d_{IJ})$, where d_{IJ} represents the 'distance' from I to J , arrange the points in a cyclic order in such a way that the sum of the d_{IJ} between consecutive points is minimal. Since there are only a finite number of possibilities (at most $\frac{1}{2}(n-1)!$) to consider, the problem is to devise a method of picking out the optimal arrangement which is reasonably efficient for fairly large values of n . Although algorithms have been devised for problems of similar nature, e.g., the optimal assignment problem,^{3,7,8} little is known about the traveling-salesman problem. We do not claim that this note alters the situation very much; what we shall do is outline a way of approaching the problem that sometimes, at least, enables one to find an optimal path and prove it so. In particular, it will be shown that a certain arrangement of 49 cities, one in each of the 48 states and Washington, D. C., is best, the d_{IJ} used representing road distances as taken from an atlas.

The 42 (49) Cities

- | | | |
|--------------------------|--------------------------|------------------------|
| 1. Manchester, N. H. | 18. Carson City, Nev. | 34. Birmingham, Ala. |
| 2. Montpelier, Vt. | 19. Los Angeles, Calif. | 35. Atlanta, Ga. |
| 3. Detroit, Mich. | 20. Phoenix, Ariz. | 36. Jacksonville, Fla. |
| 4. Cleveland, Ohio | 21. Santa Fe, N. M. | 37. Columbia, S. C. |
| 5. Charleston, W. Va. | 22. Denver, Colo. | 38. Raleigh, N. C. |
| 6. Louisville, Ky. | 23. Cheyenne, Wyo. | 39. Richmond, Va. |
| 7. Indianapolis, Ind. | 24. Omaha, Neb. | 40. Washington, D. C. |
| 8. Chicago, Ill. | 25. Des Moines, Iowa | 41. Boston, Mass. |
| 9. Milwaukee, Wis. | 26. Kansas City, Mo. | 42. Portland, Me. |
| 10. Minneapolis, Minn. | 27. Topeka, Kans. | A. Baltimore, Md. |
| 11. Pierre, S. D. | 28. Oklahoma City, Okla. | B. Wilmington, Del. |
| 12. Bismarck, N. D. | 29. Dallas, Tex. | C. Philadelphia, Penn. |
| 13. Helena, Mont. | 30. Little Rock, Ark. | D. Newark, N. J. |
| 14. Seattle, Wash. | 31. Memphis, Tenn. | E. New York, N. Y. |
| 15. Portland, Ore. | 32. Jackson, Miss. | F. Hartford, Conn. |
| 16. Boise, Idaho | 33. New Orleans, La. | G. Providence, R. I. |
| 17. Salt Lake City, Utah | | |

Combinatorial Explosion



(42-1)!/2

NATURAL LANGUAGE MATH INPUT EXTENDED KEYBOARD EXAMPLES UPLOAD RANDOM

Input

$$\frac{1}{2} (42 - 1)!$$

n! is the factorial function

Result

1672626330658190355408503102672037583257600000000

Scientific notation

$$1.6726263306581903554085031026720375832576 \times 10^{49}$$

Number name [Full name](#)

16 quindillion ...

Number length

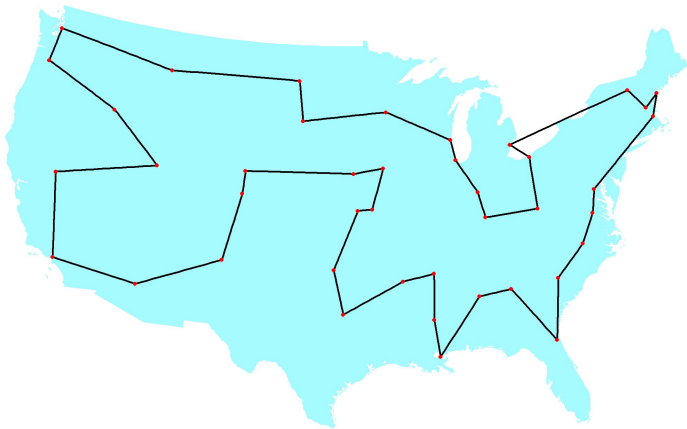
50 decimal digits

Alternative representations [More](#)

$$\frac{1}{2} (42 - 1)! = \frac{\Gamma(42)}{2}$$
$$\frac{1}{2} (42 - 1)! = \frac{\Gamma(42, 0)}{2}$$
$$\frac{1}{2} (42 - 1)! = \frac{(1)_{41}}{2}$$

Solution of this TSP problem

Dantzig, Fulkerson and Johnson found an optimal tour through 42 cities.



http://www.math.uwaterloo.ca/tsp/history/img/dantzig_big.html

Road Distances

TABLE I

ROAD DISTANCES BETWEEN CITIES IN ADJUSTED UNITS

The figures in the table are mileages between the two specified numbered cities, less 11, divided by 17, and rounded to the nearest integer.

[illegible]

Road Distances

Hence this is an instance of the **Metric TSP**, but not **Euclidean TSP**.

TABLE I

ROAD DISTANCES BETWEEN CITIES IN ADJUSTED UNITS

The figures in the table are mileages between the two specified numbered cities, less 11, divided by 17, and rounded to the nearest integer.

[illegible]

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned} \sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42 \end{aligned}$$

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned} \sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42 \end{aligned}$$

Constraints $x(i, j) \in \{0, 1\}$ are not allowed in a LP!

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned} \sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42 \end{aligned}$$

Constraints $x(i, j) \in \{0, 1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned} \sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42 \end{aligned}$$

Constraints $x(i, j) \in \{0, 1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i, j) \in (0, 1)$:

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned} \sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42 \end{aligned}$$

Constraints $x(i, j) \in \{0, 1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i, j) \in (0, 1)$:
 - Add $x(i, j) = 0$ to the LP, solve it and recurse
 - Add $x(i, j) = 1$ to the LP, solve it and recurse
 - Return best of these two solutions

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned}\sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42\end{aligned}$$

Constraints $x(i, j) \in \{0, 1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i, j) \in (0, 1)$:
 - Add $x(i, j) = 0$ to the LP, solve it and recurse
 - Add $x(i, j) = 1$ to the LP, solve it and recurse
 - Return best of these two solutions
- If solution of LP integral, return objective value

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j)x(i, j)$

subject to

$$\begin{aligned} \sum_{j < i} x(i, j) + \sum_{j > i} x(j, i) &= 2 && \text{for each } 1 \leq i \leq 42 \\ 0 \leq x(i, j) &\leq 1 && \text{for each } 1 \leq j < i \leq 42 \end{aligned}$$

Constraints $x(i, j) \in \{0, 1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i, j) \in (0, 1)$:
 - Add $x(i, j) = 0$ to the LP, solve it and recurse
 - Add $x(i, j) = 1$ to the LP, solve it and recurse
 - Return best of these two solutions
- If solution of LP integral, return objective value

Bound-Step: If the best known integral solution so far is better than the solution of a LP, no need to explore branch further!

Outline

Introduction

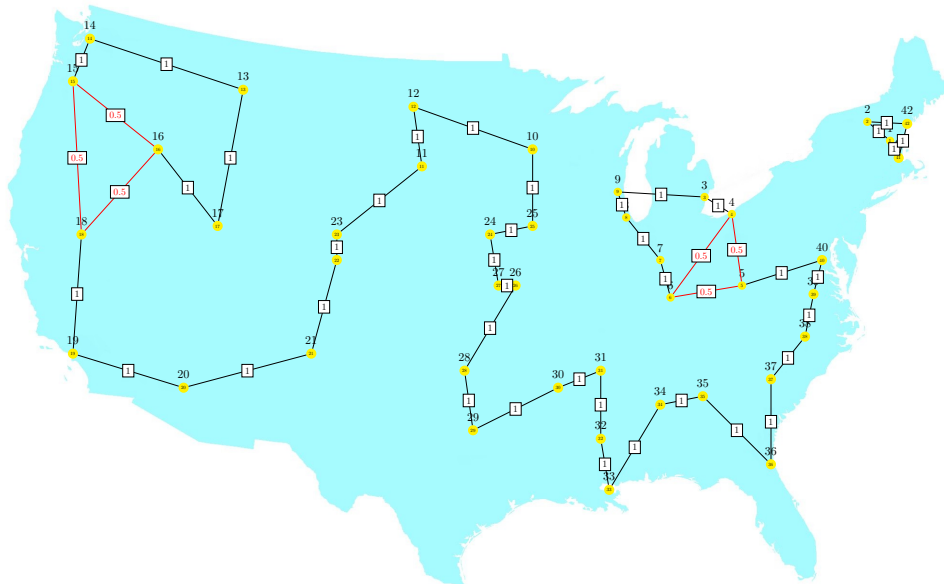
Examples of TSP Instances

Demonstration

In the following, there are a few different runs of the demo.

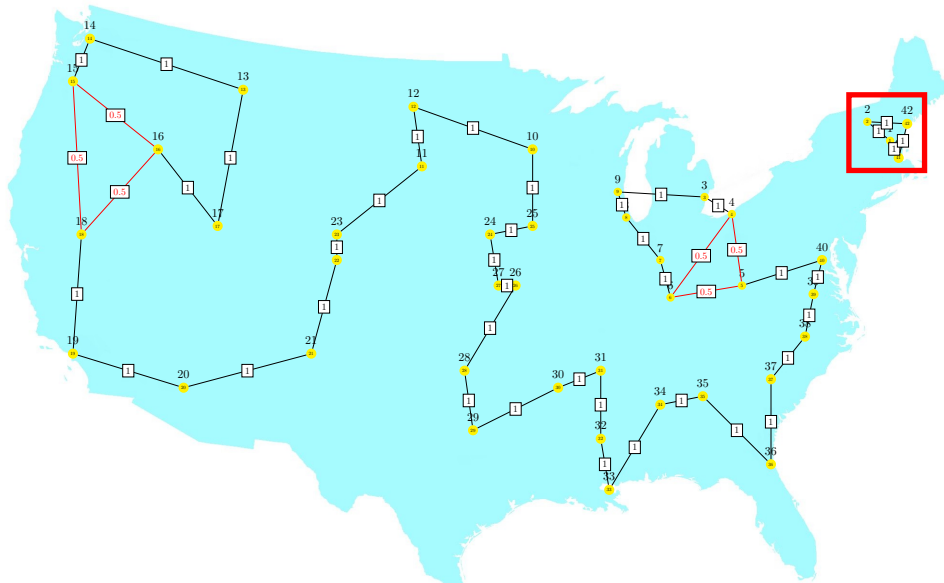
Iteration 1:

Objective value: -641.000000 , 861 variables, 945 constraints, 1809 iterations



Iteration 1: Eliminate Subtour 1, 2, 41, 42

Objective value: -641.000000 , 861 variables, 945 constraints, 1809 iterations

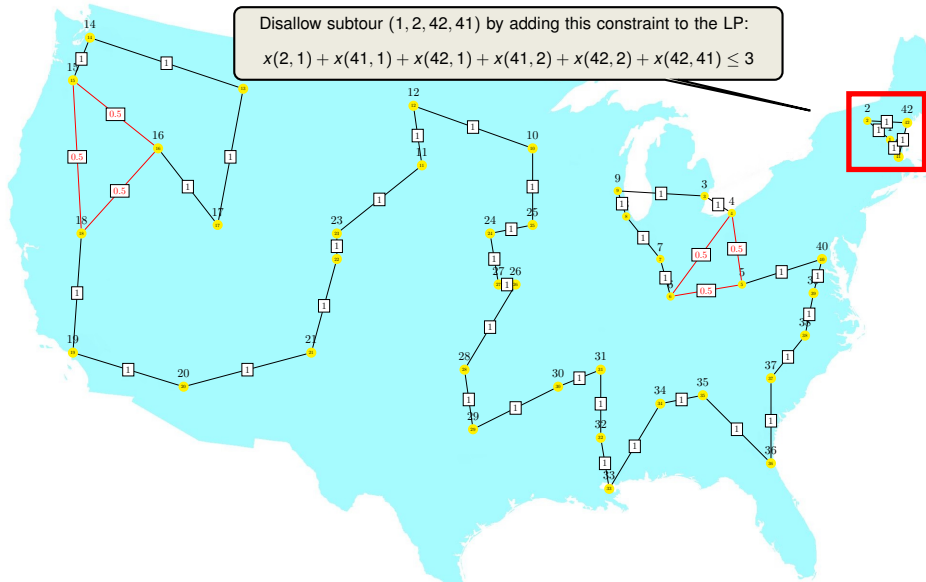


Iteration 1: Eliminate Subtour 1, 2, 41, 42

Objective value: -641.000000 , 861 variables, 945 constraints, 1809 iterations

Disallow subtour (1, 2, 42, 41) by adding this constraint to the LP:

$$x(2, 1) + x(41, 1) + x(42, 1) + x(41, 2) + x(42, 2) + x(42, 41) \leq 3$$



Iteration 1: Eliminate Subtour 1, 2, 41, 42

Objective value: -641.000000, 861 variables, 945 constraints, 1809 iterations

Disallow subtour (1, 2, 42, 41) by adding this constraint to the LP:

$$x(2, 1) + x(41, 1) + x(42, 1) + x(41, 2) + x(42, 2) + x(42, 41) \leq 3$$

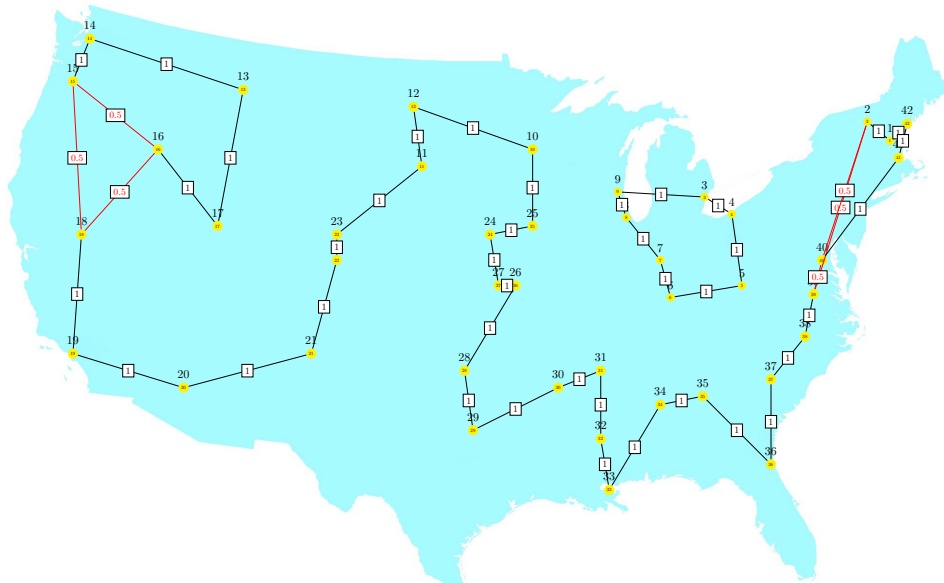
Equivalent to: $S = \{1, 2, 41, 42\}$,

$$\sum_{i \in S, j \in V \setminus S} x(\max(i, j), \min(i, j)) \geq 2$$



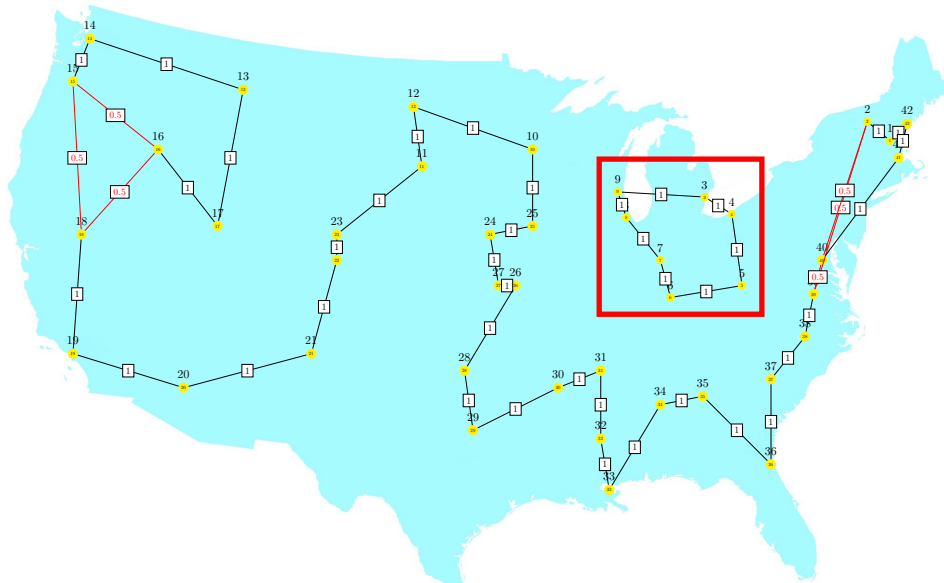
Iteration 2:

Objective value: -676.000000 , 861 variables, 946 constraints, 1802 iterations



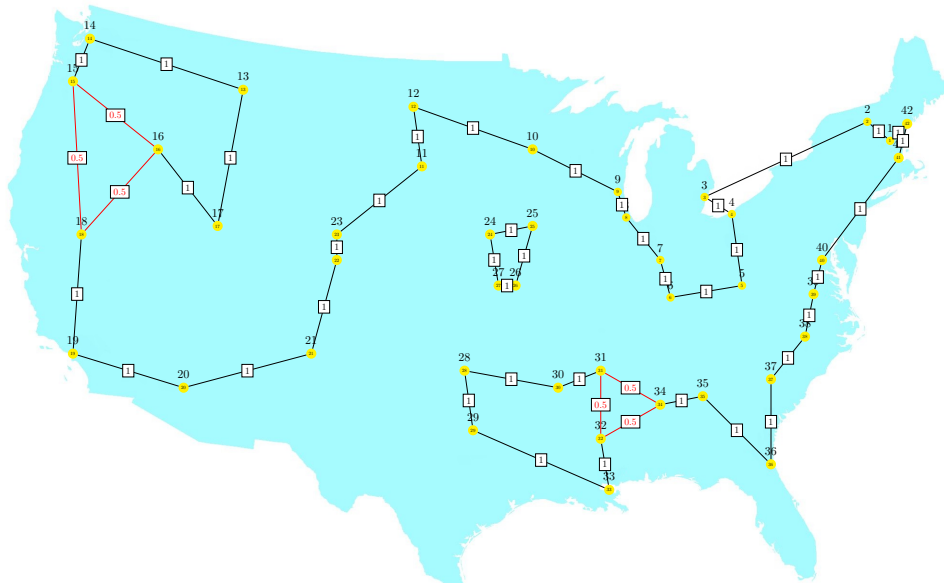
Iteration 2: Eliminate Subtour 3 – 9

Objective value: -676.000000 , 861 variables, 946 constraints, 1802 iterations



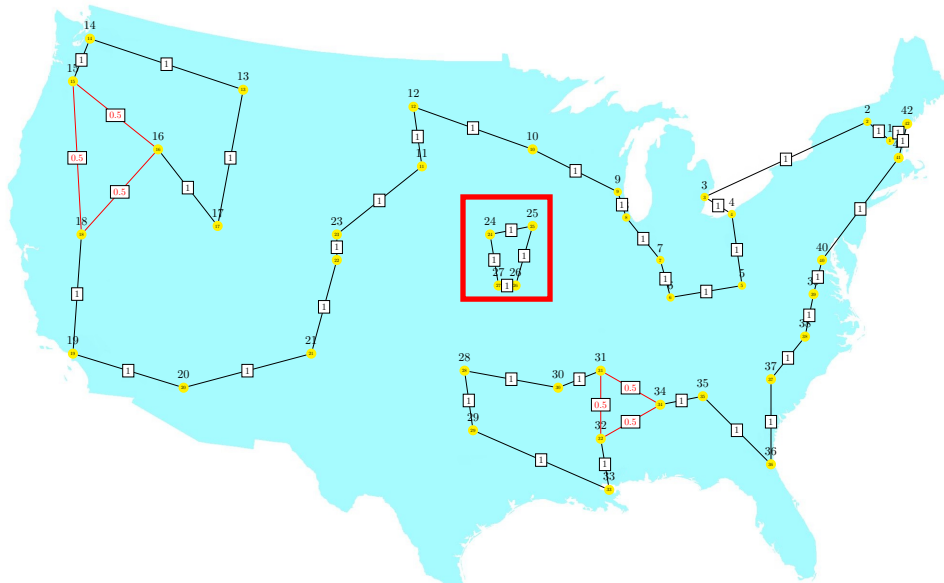
Iteration 3:

Objective value: -681.000000 , 861 variables, 947 constraints, 1984 iterations



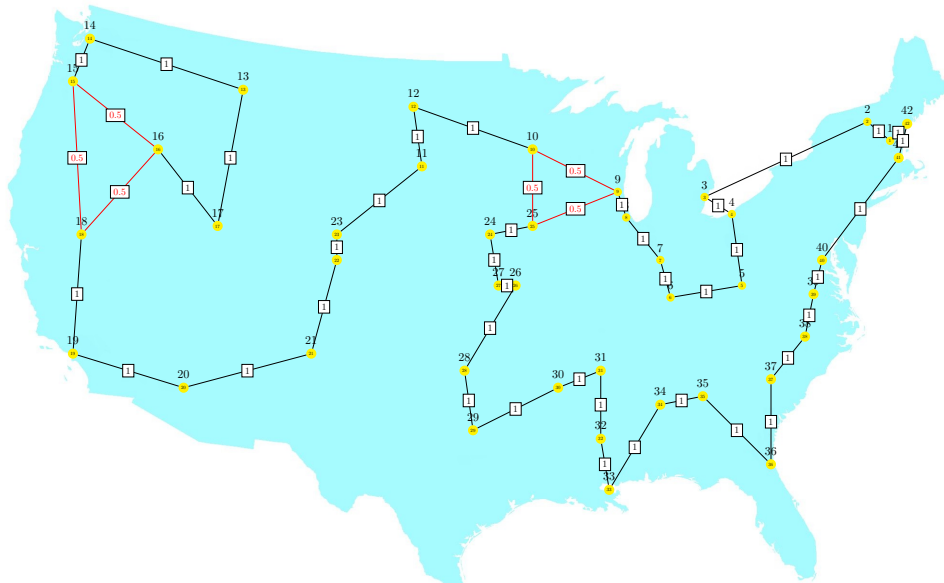
Iteration 3: Eliminate Subtour 24, 25, 26, 27

Objective value: -681.000000, 861 variables, 947 constraints, 1984 iterations



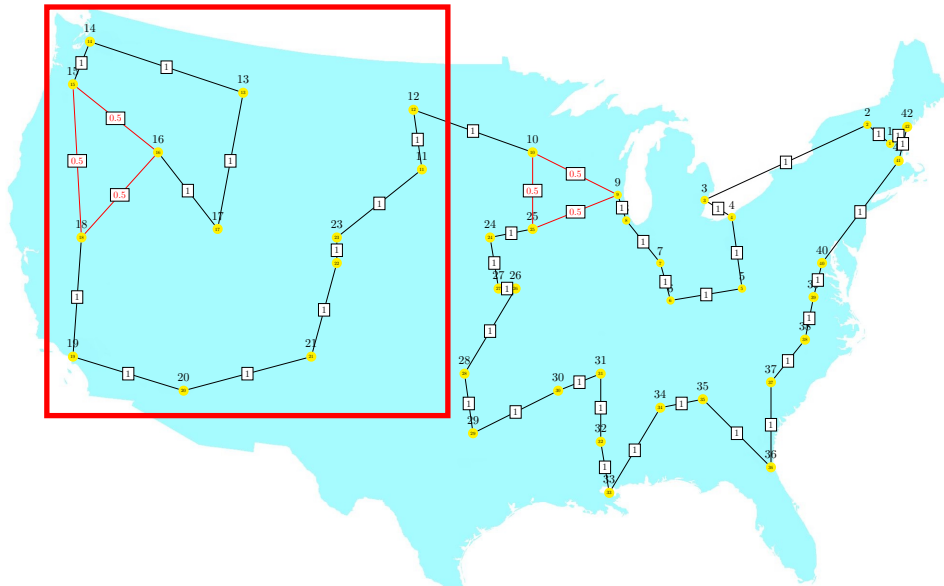
Iteration 4:

Objective value: -682.500000 , 861 variables, 948 constraints, 1492 iterations



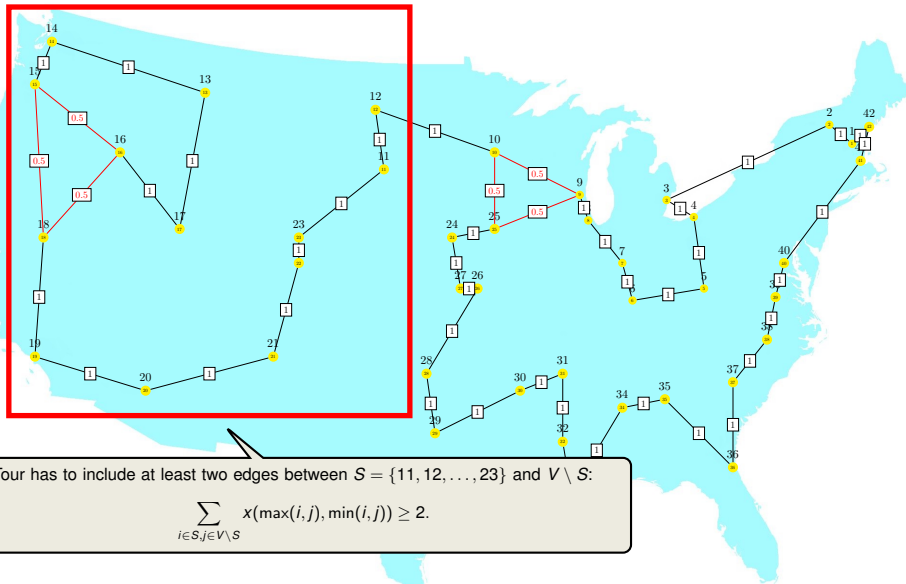
Iteration 4: Eliminate Cut 11 – 23

Objective value: -682.500000 , 861 variables, 948 constraints, 1492 iterations



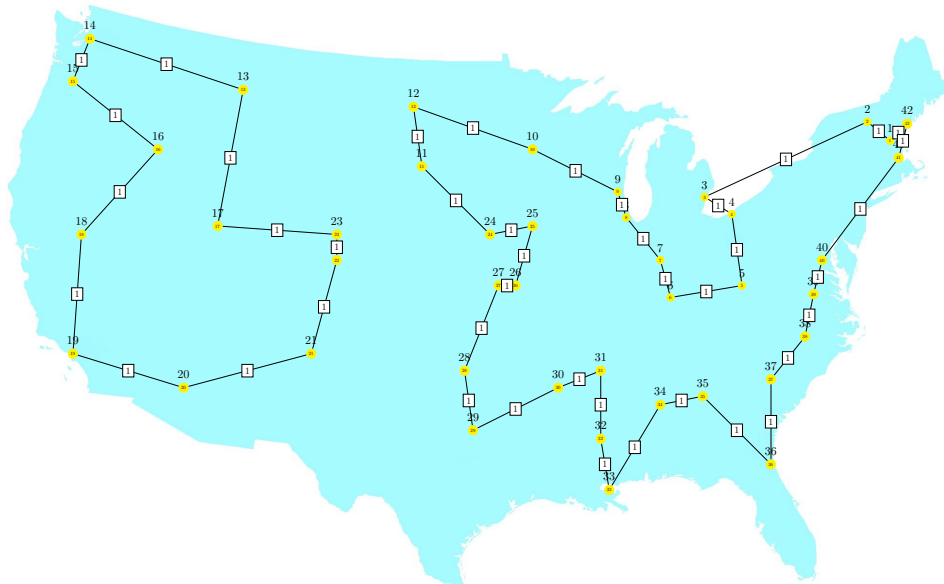
Iteration 4: Eliminate Cut 11 – 23

Objective value: -682.500000, 861 variables, 948 constraints, 1492 iterations



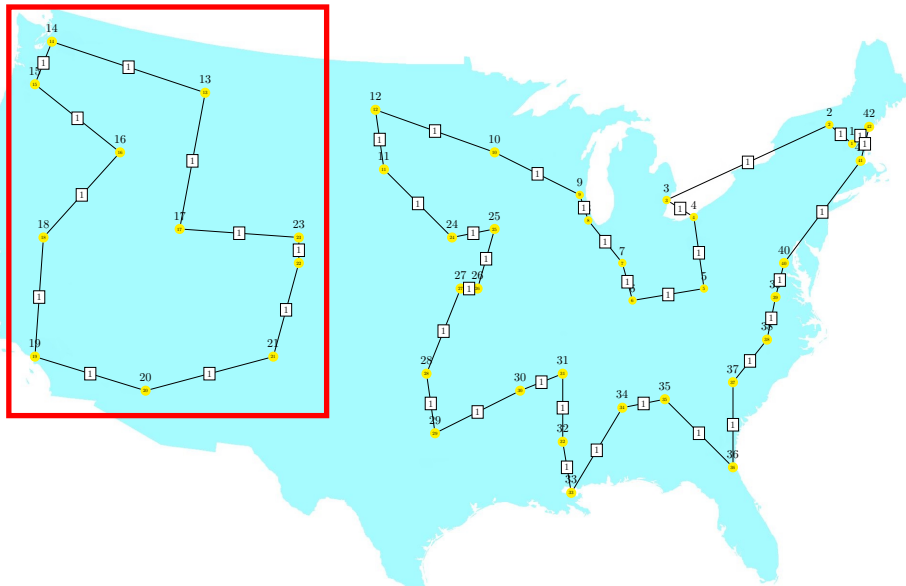
Iteration 5:

Objective value: -686.000000 , 861 variables, 949 constraints, 2446 iterations



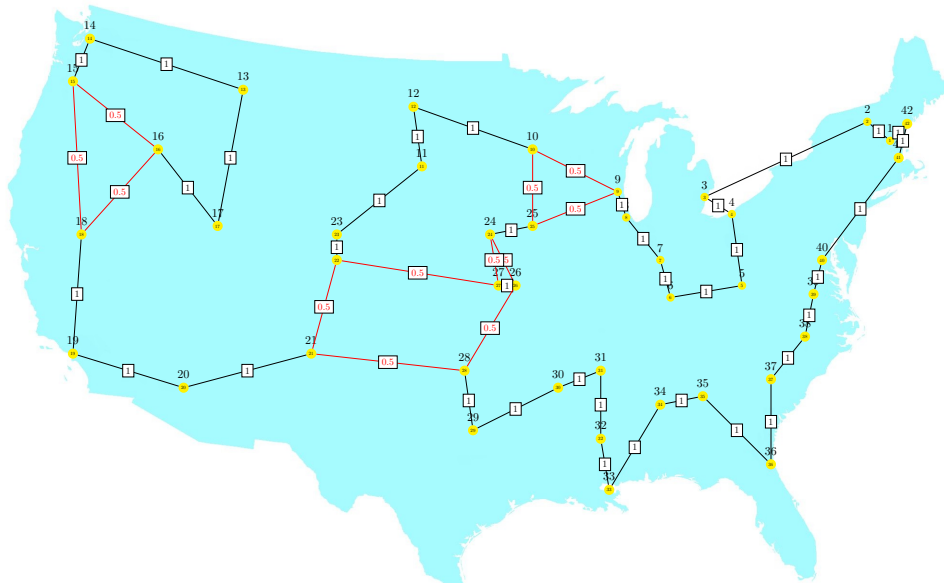
Iteration 5: Eliminate Subtour 13 – 23

Objective value: -686.000000 , 861 variables, 949 constraints, 2446 iterations



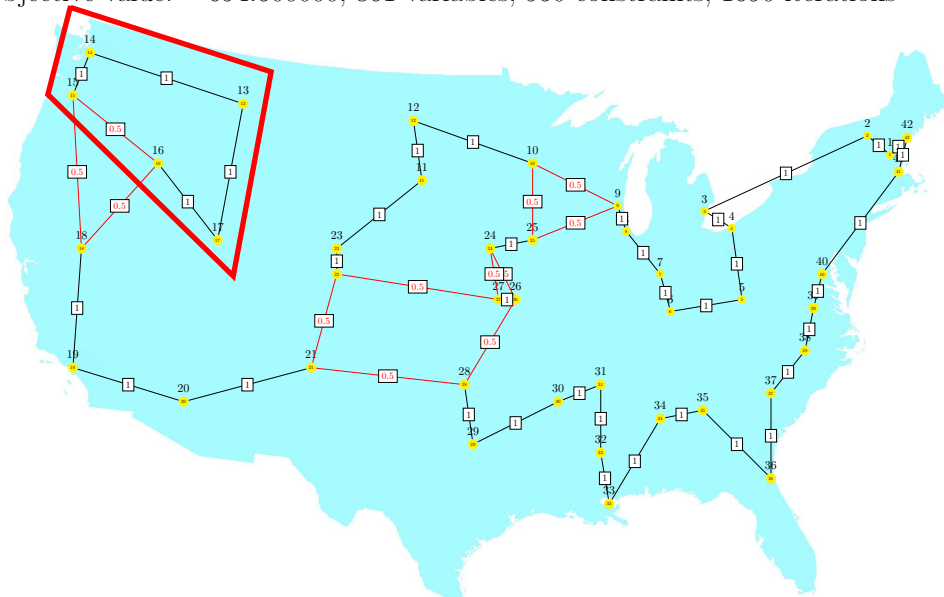
Iteration 6:

Objective value: -694.500000 , 861 variables, 950 constraints, 1690 iterations



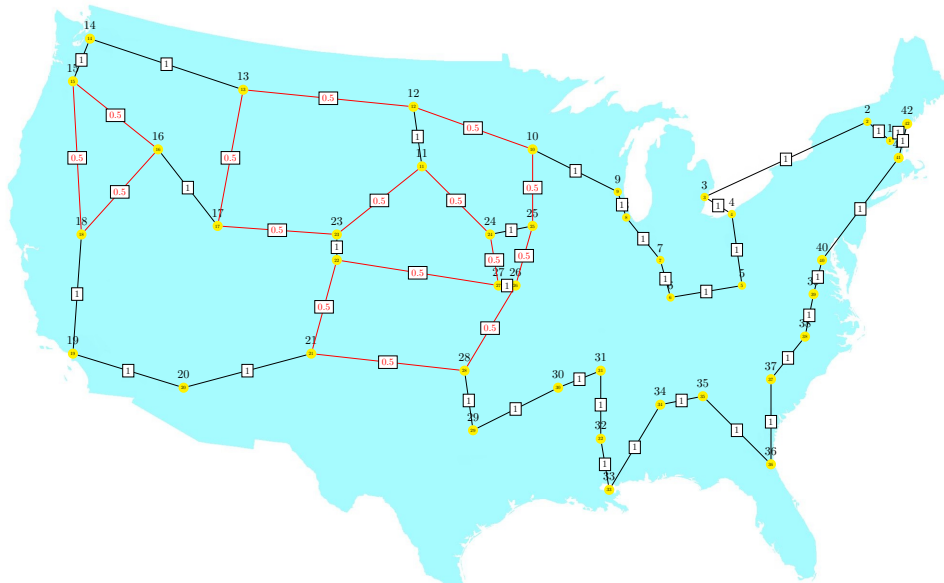
Iteration 6: Eliminate Cut 13 – 17

Objective value: -694.500000, 861 variables, 950 constraints, 1690 iterations



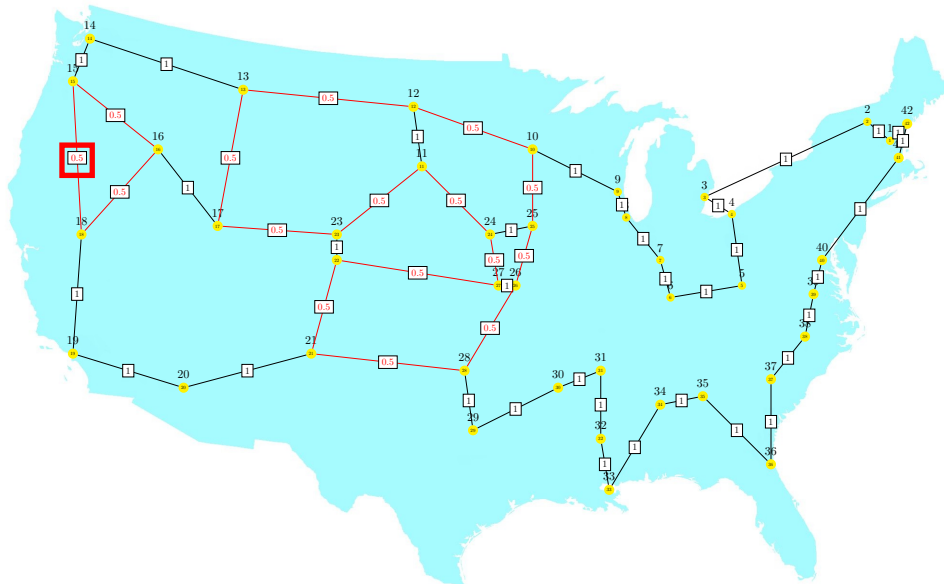
Iteration 7:

Objective value: -697.000000 , 861 variables, 951 constraints, 2212 iterations



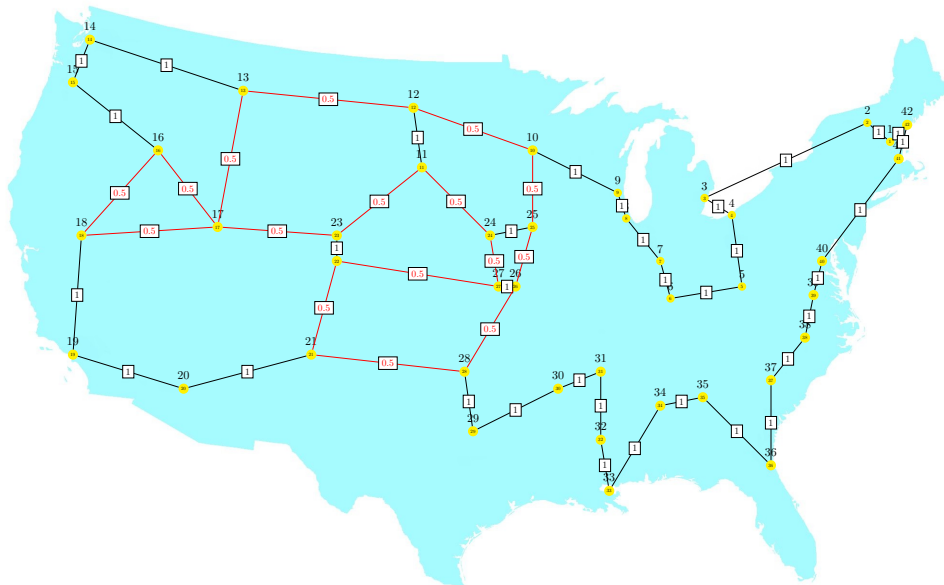
Iteration 7: Branch 1a $x_{18,15} = 0$

Objective value: -697.000000 , 861 variables, 951 constraints, 2212 iterations



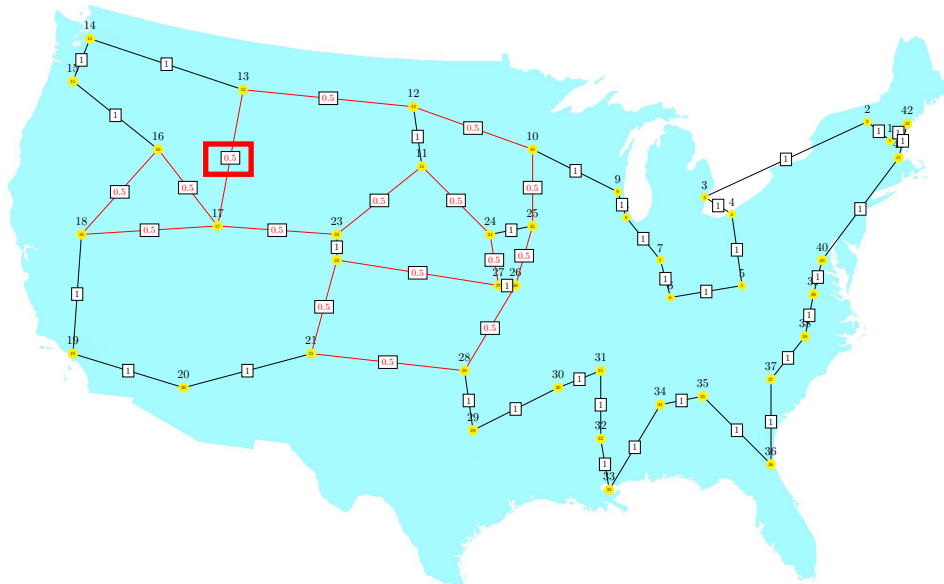
Iteration 8:

Objective value: -698.000000 , 861 variables, 952 constraints, 1878 iterations



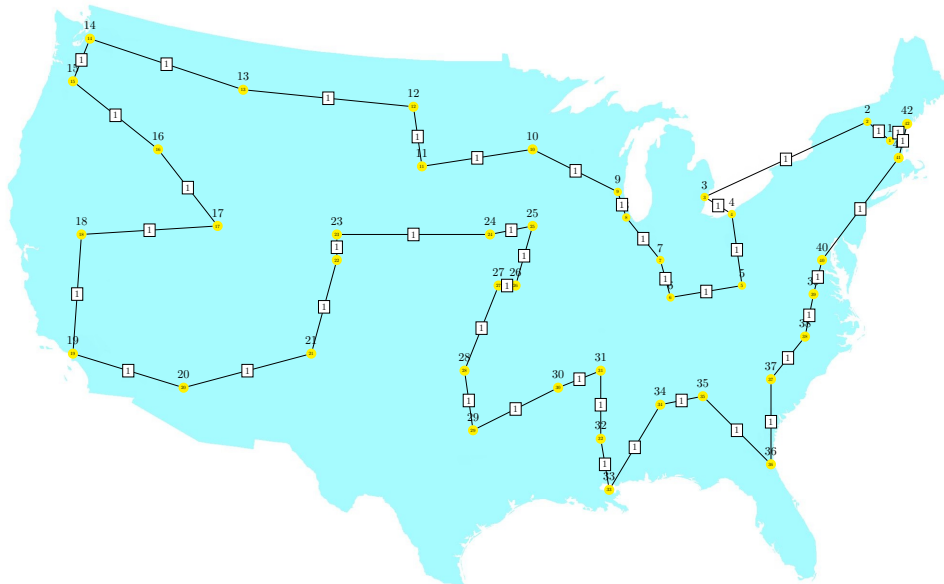
Iteration 8: Branch 2a $x_{17,13} = 0$

Objective value: -698.000000 , 861 variables, 952 constraints, 1878 iterations



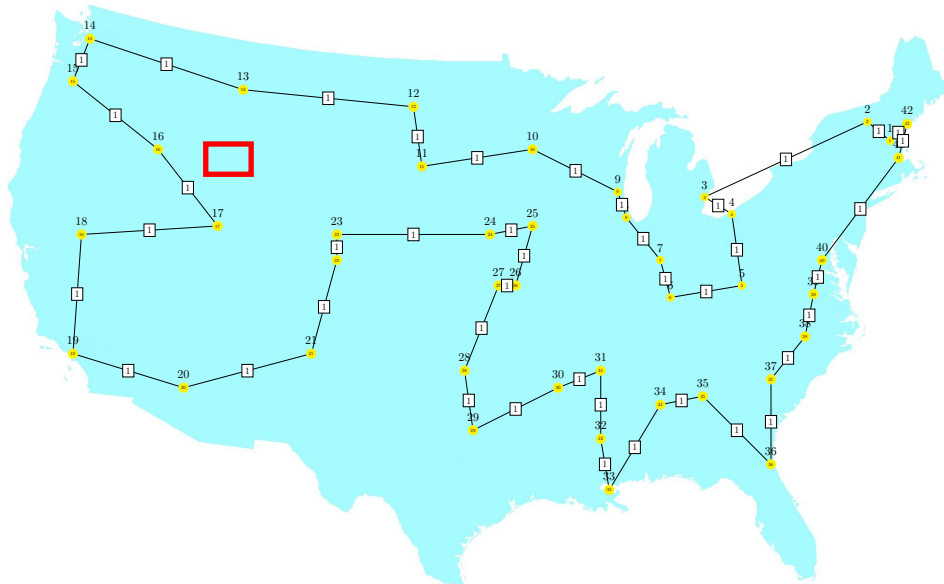
Iteration 9:

Objective value: -699.000000 , 861 variables, 953 constraints, 2281 iterations



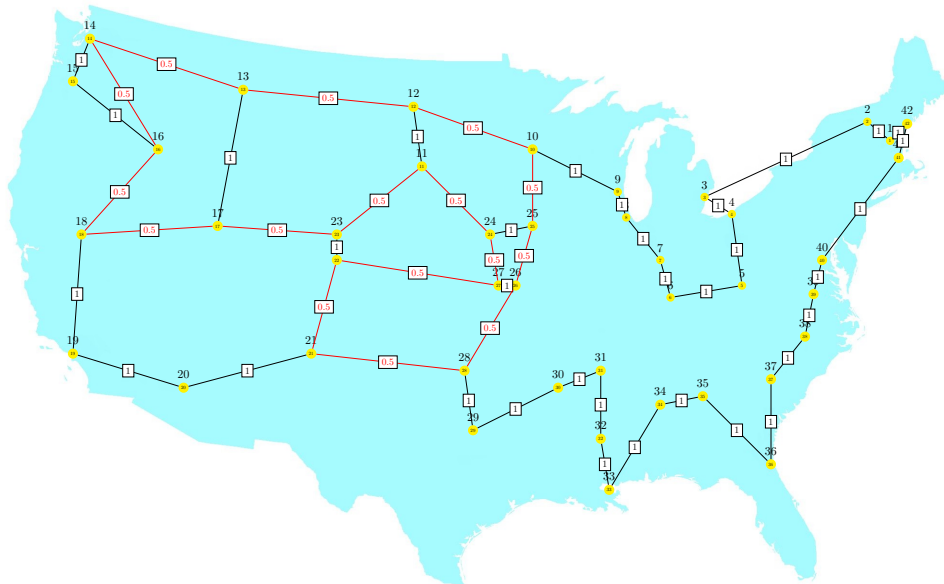
Iteration 9: Branch 2b $x_{17,13} = 1$

Objective value: -699.000000 , 861 variables, 953 constraints, 2281 iterations



Iteration 10:

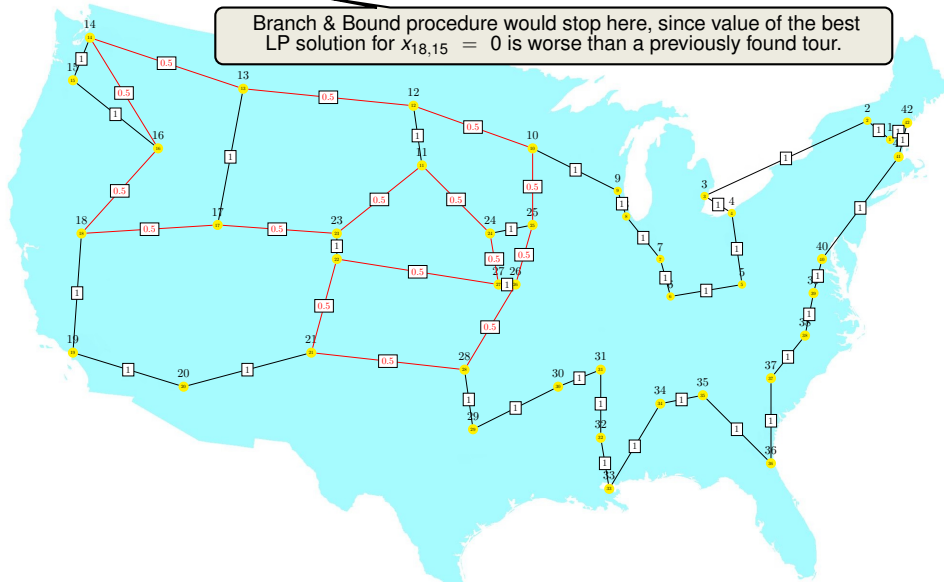
Objective value: -700.000000 , 861 variables, 954 constraints, 2398 iterations



Iteration 10:

Objective value: -700.000000 , 861 variables, 954 constraints, 2398 iterations

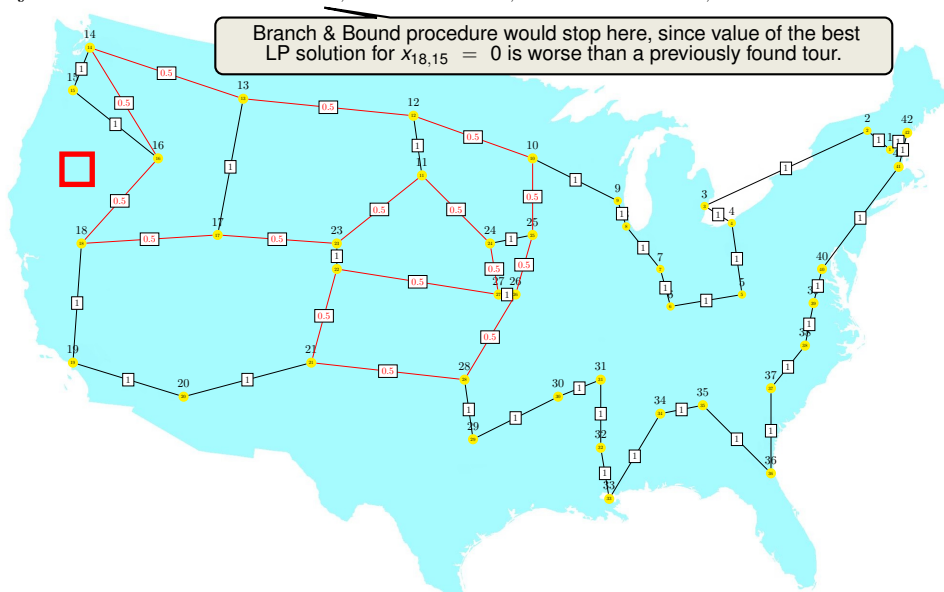
Branch & Bound procedure would stop here, since value of the best LP solution for $x_{18,15} = 0$ is worse than a previously found tour.



Iteration 10: Branch 1b $x_{18,15} = 1$

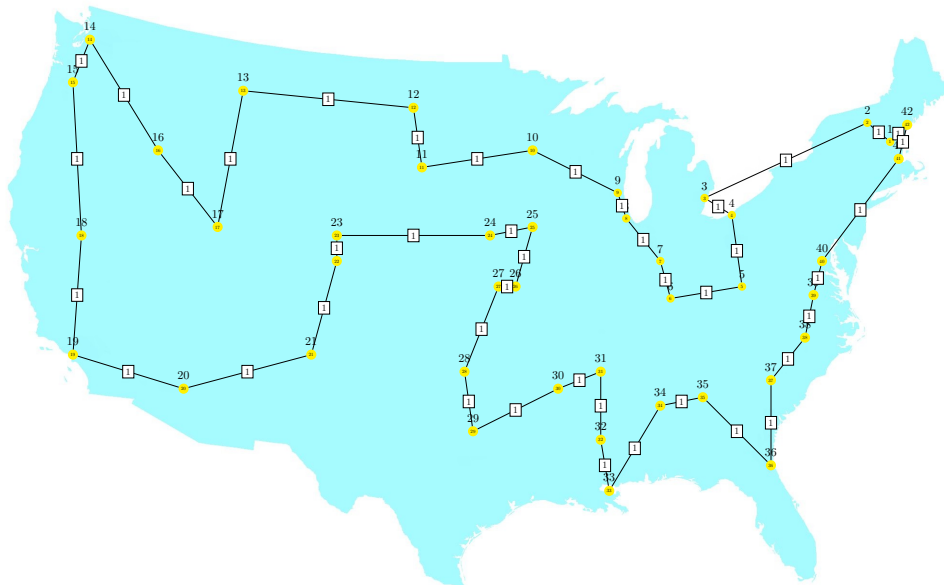
Objective value: -700.000000 , 861 variables, 954 constraints, 2398 iterations

Branch & Bound procedure would stop here, since value of the best LP solution for $x_{18,15} = 0$ is worse than a previously found tour.



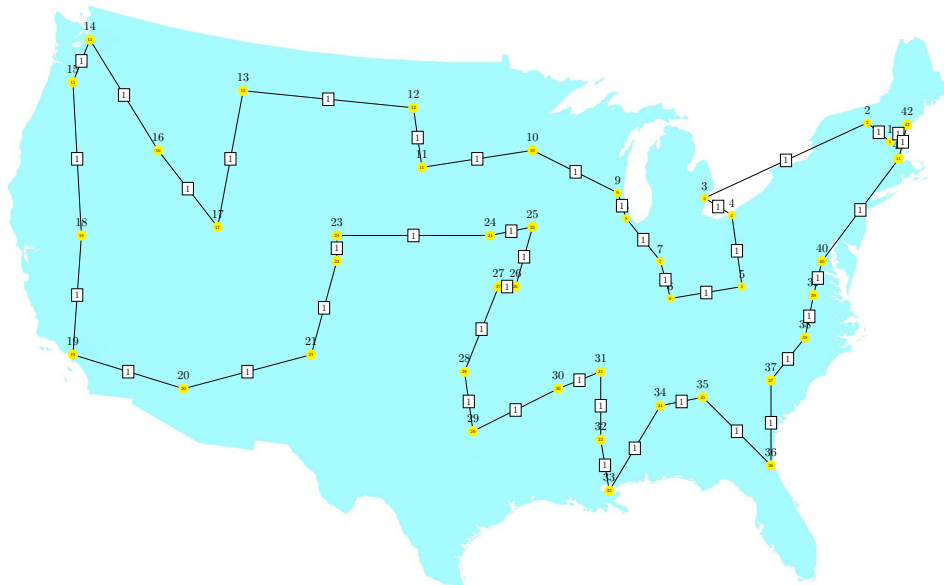
Iteration 11:

Objective value: -701.000000 , 861 variables, 953 constraints, 2506 iterations



Iteration 11: Branch & Bound terminates

Objective value: -701.000000 , 861 variables, 953 constraints, 2506 iterations



Branch & Bound Overview

1: LP solution 641

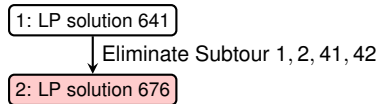
Branch & Bound Overview

1: LP solution 641

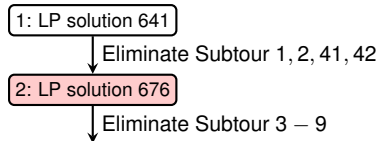


Eliminate Subtour 1, 2, 41, 42

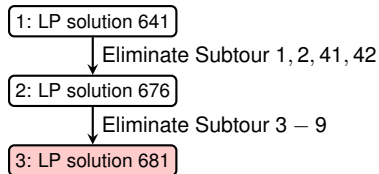
Branch & Bound Overview



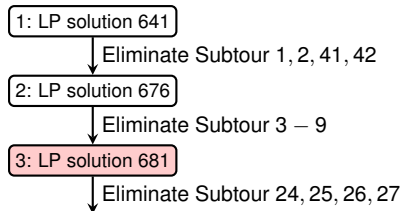
Branch & Bound Overview



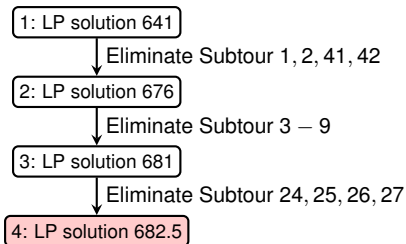
Branch & Bound Overview



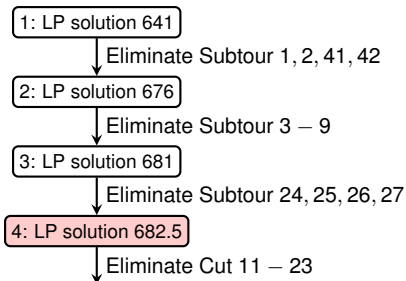
Branch & Bound Overview



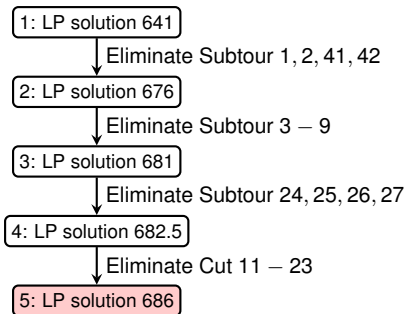
Branch & Bound Overview



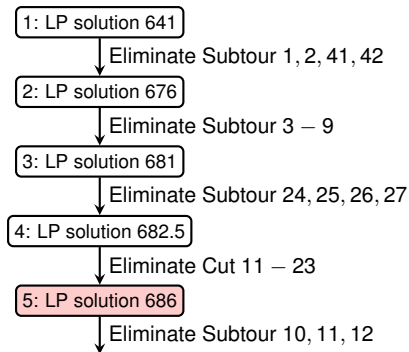
Branch & Bound Overview



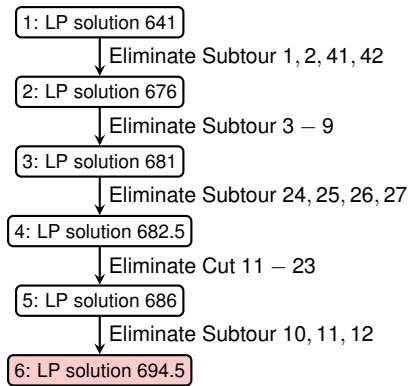
Branch & Bound Overview



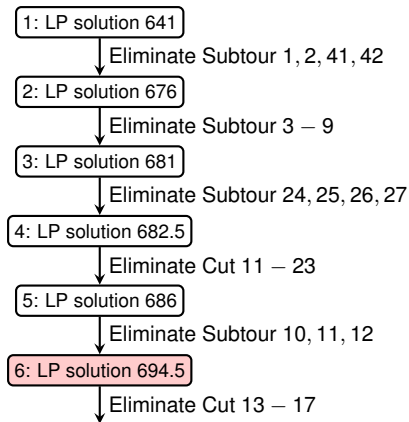
Branch & Bound Overview



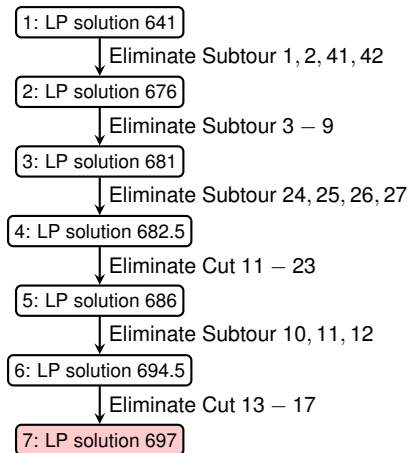
Branch & Bound Overview



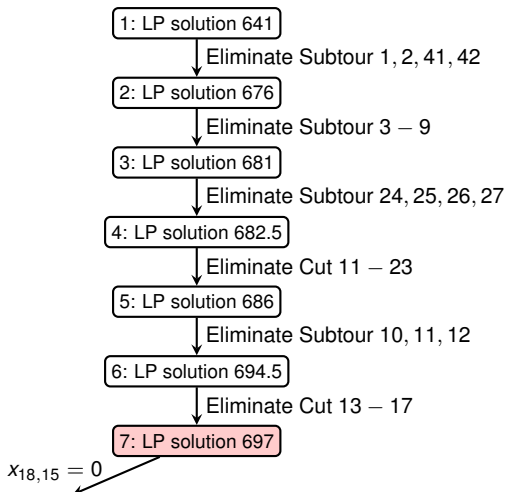
Branch & Bound Overview



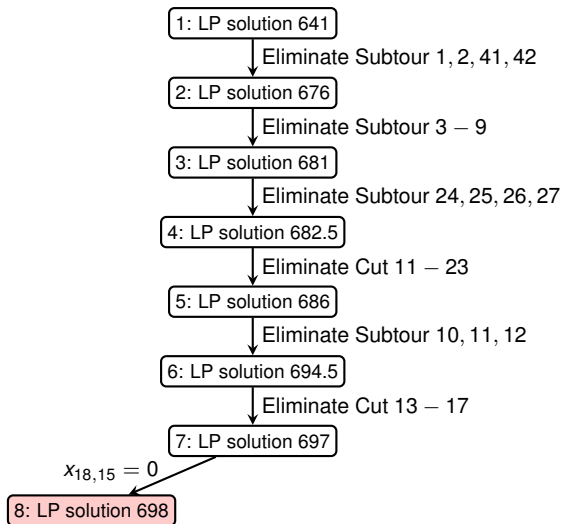
Branch & Bound Overview



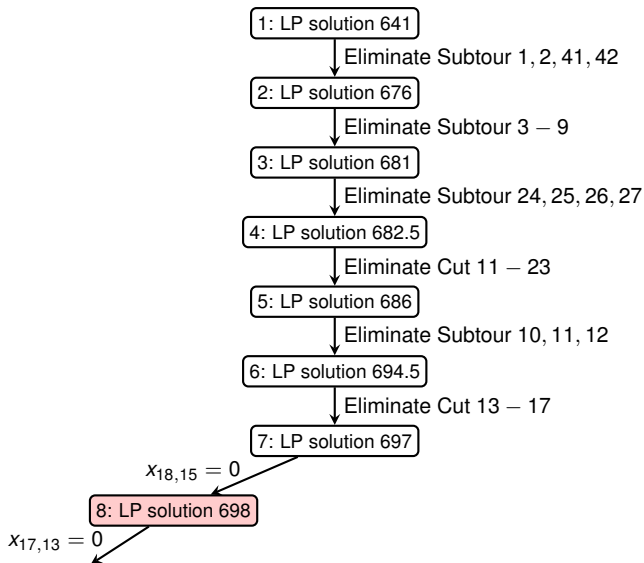
Branch & Bound Overview



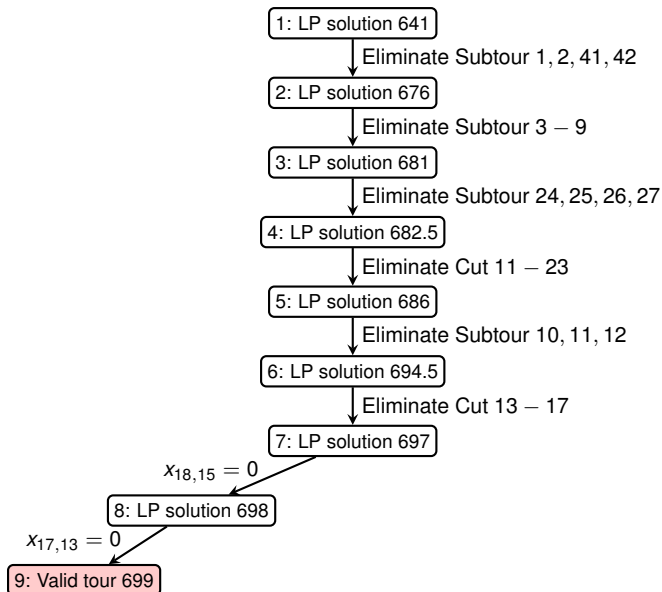
Branch & Bound Overview



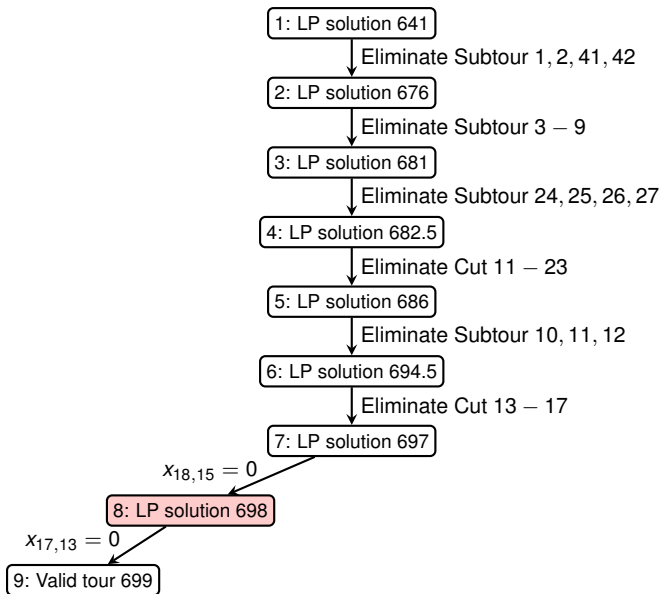
Branch & Bound Overview



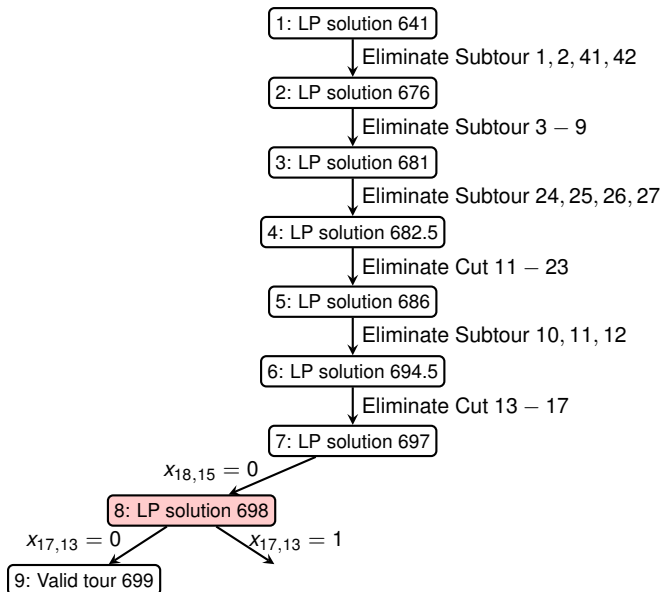
Branch & Bound Overview



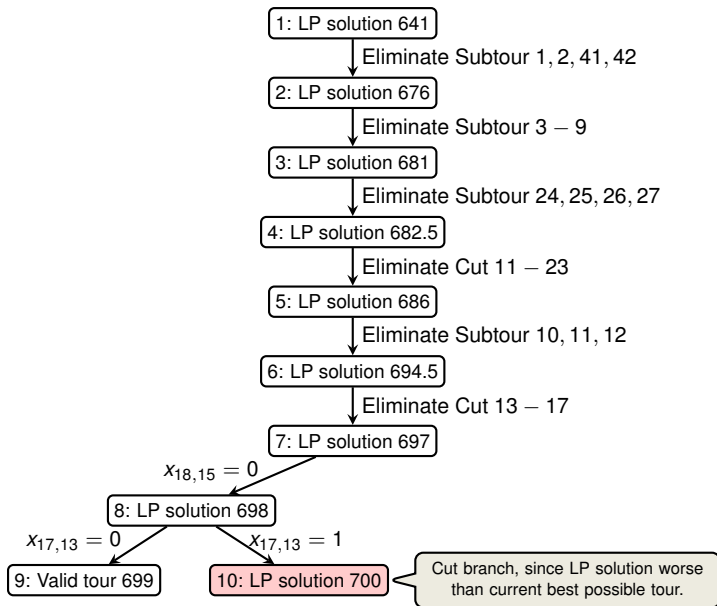
Branch & Bound Overview



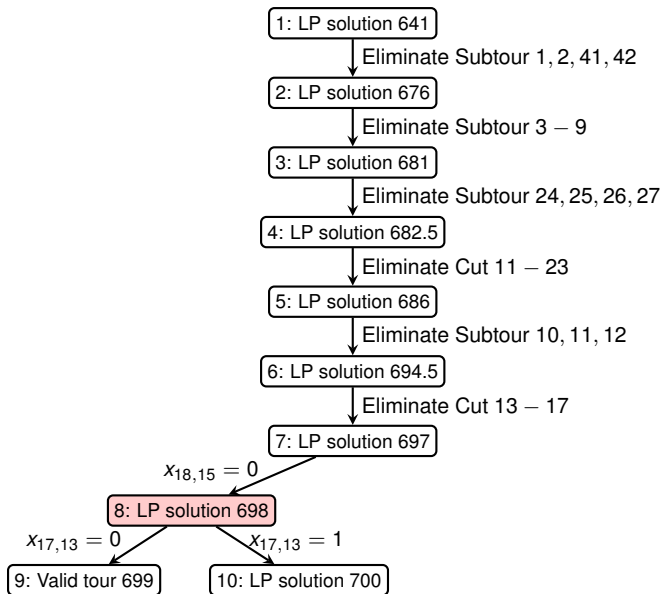
Branch & Bound Overview



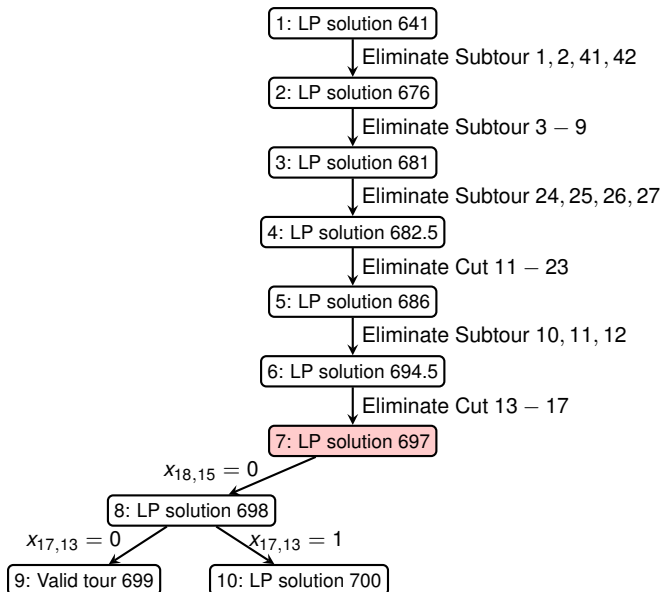
Branch & Bound Overview



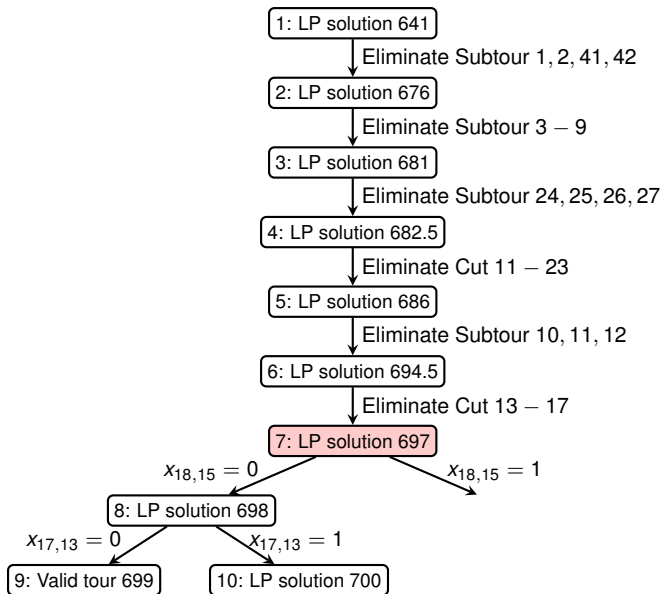
Branch & Bound Overview



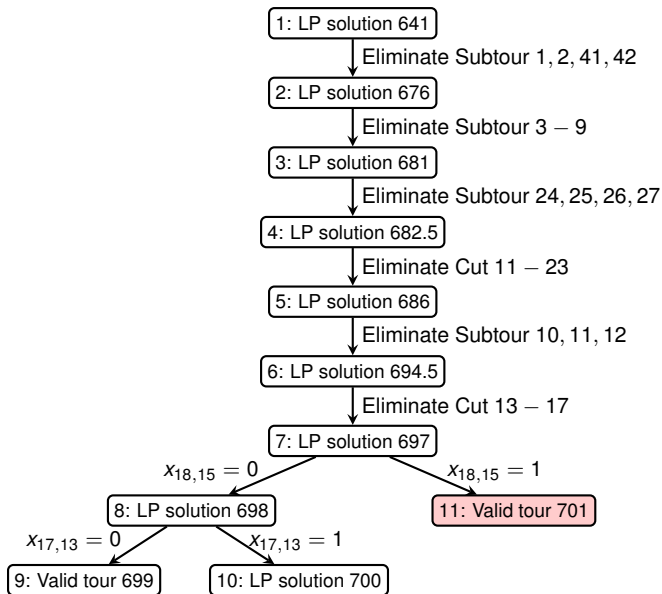
Branch & Bound Overview



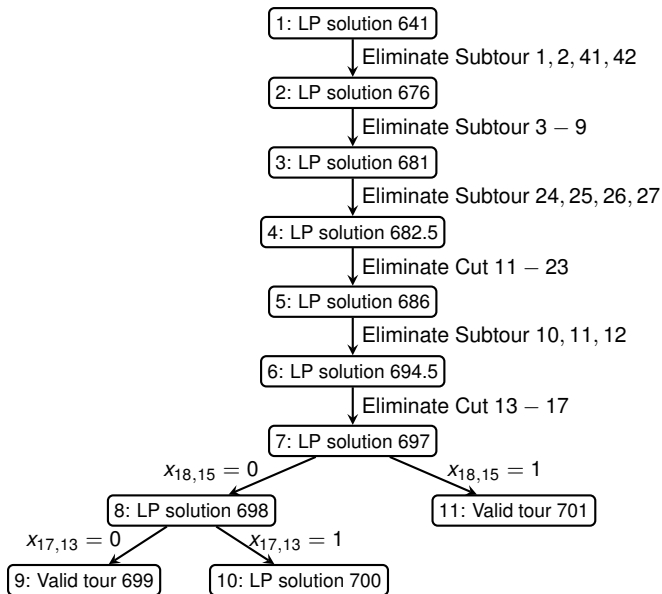
Branch & Bound Overview



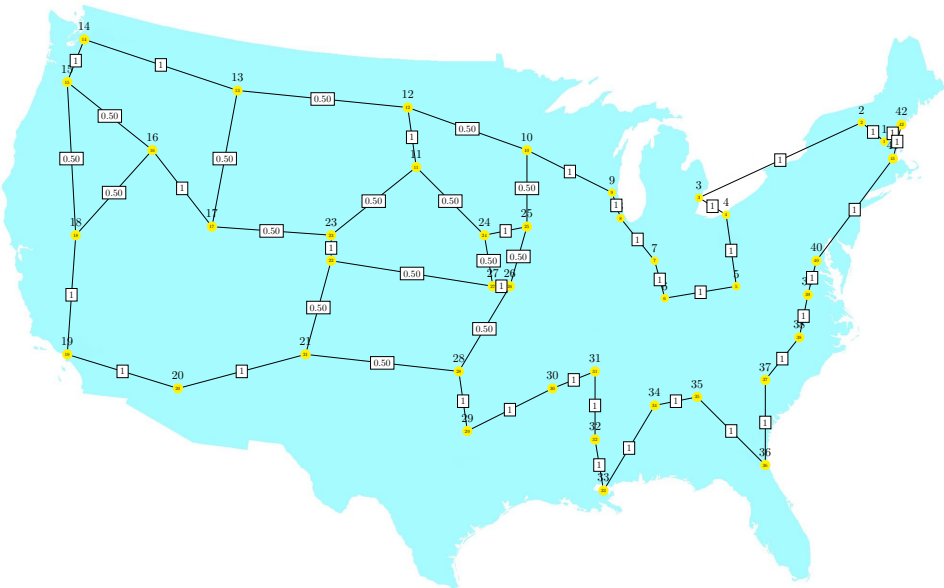
Branch & Bound Overview



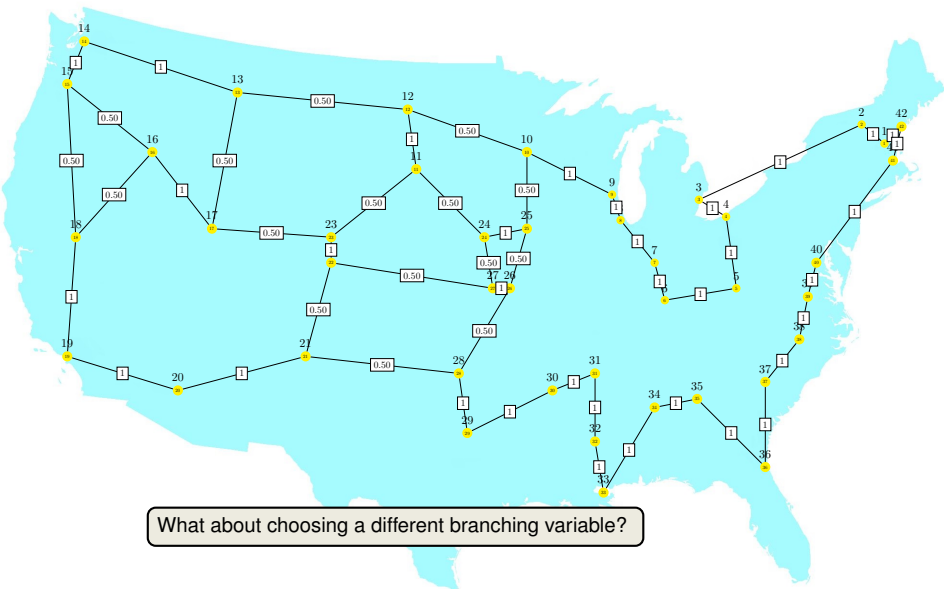
Branch & Bound Overview



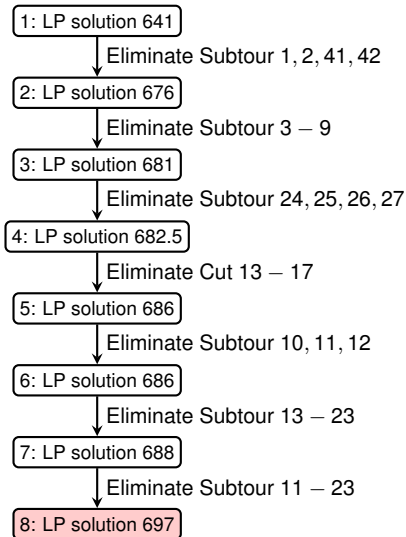
Iteration 7: Objective 697



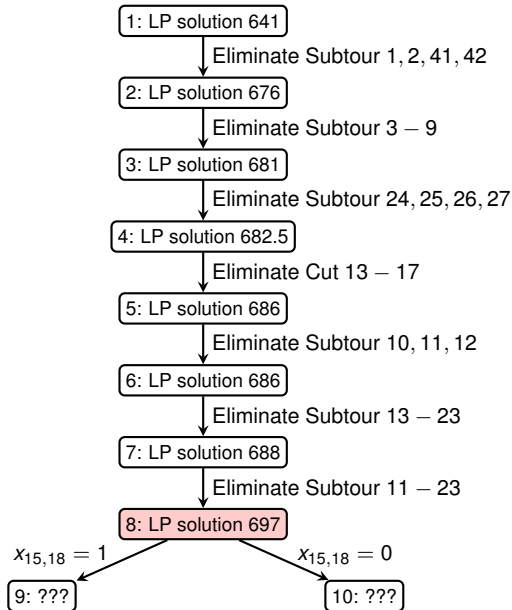
Iteration 7: Objective 697



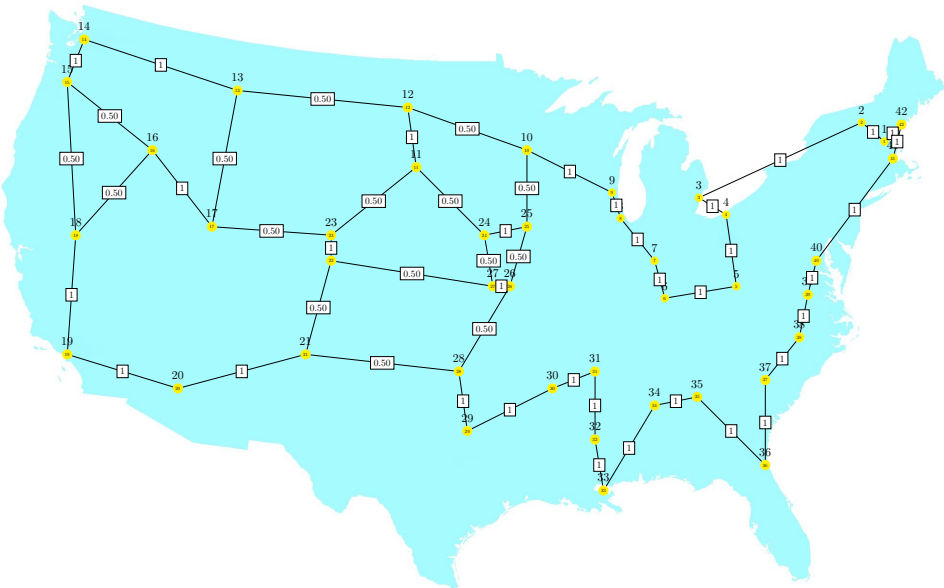
Solving Progress (Alternative Branch 1)



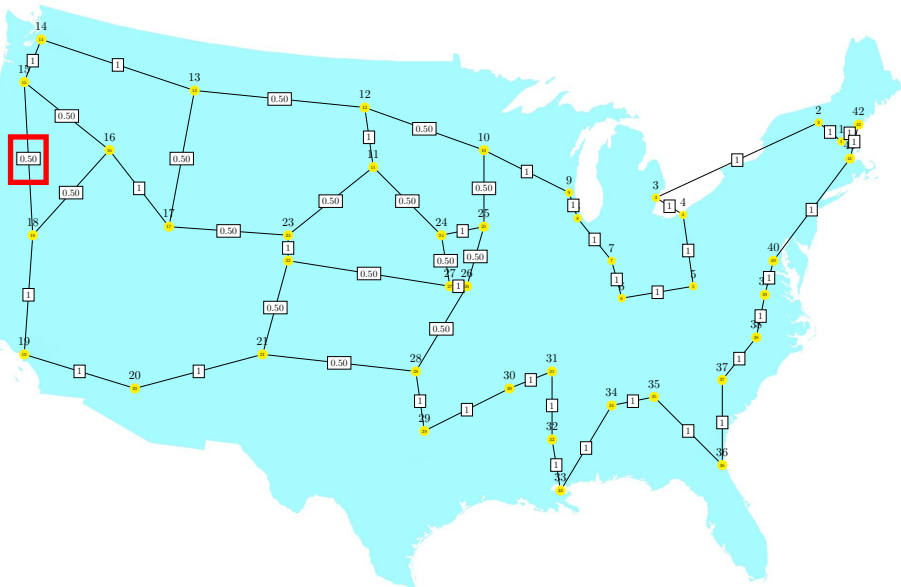
Solving Progress (Alternative Branch 1)



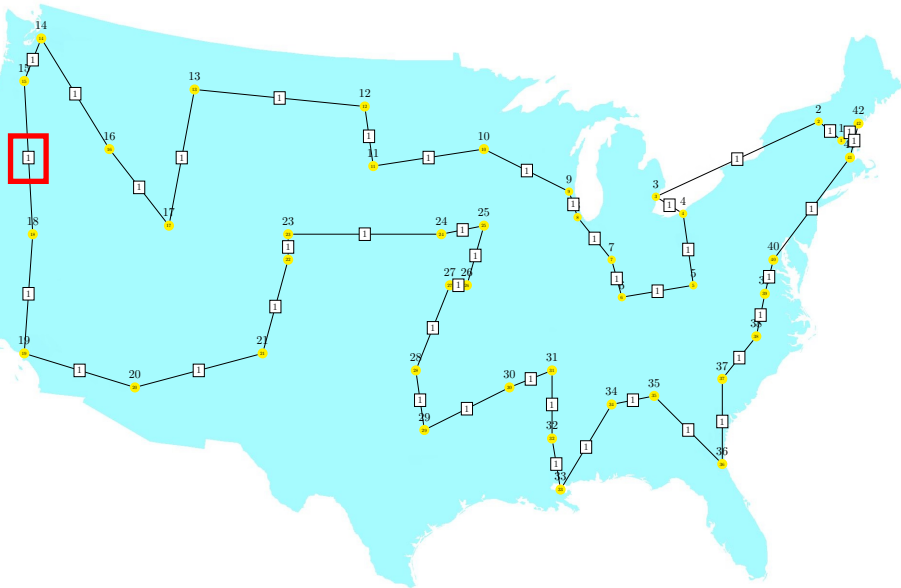
Alternative Branch 1: $x_{18,15}$, Objective 697



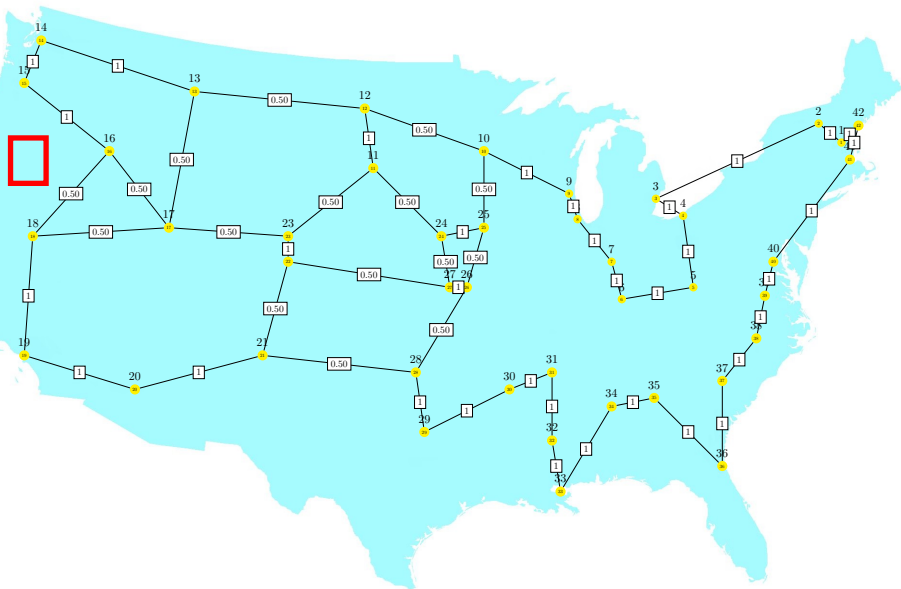
Alternative Branch 1: $x_{18,15}$, Objective 697



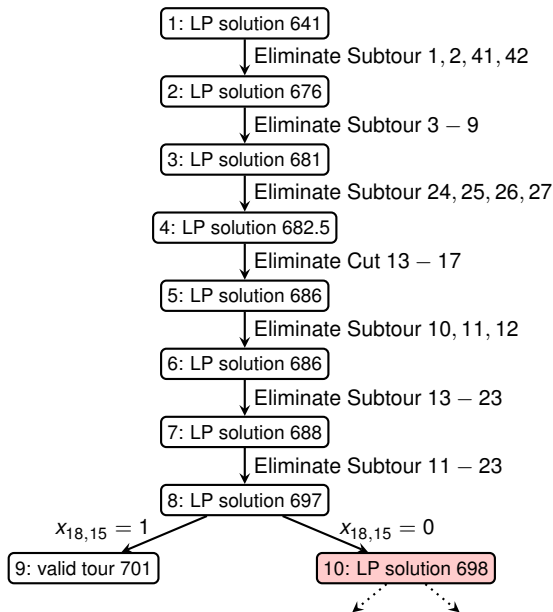
Alternative Branch 1a: $x_{18,15} = 1$, Objective 701 (Valid Tour)



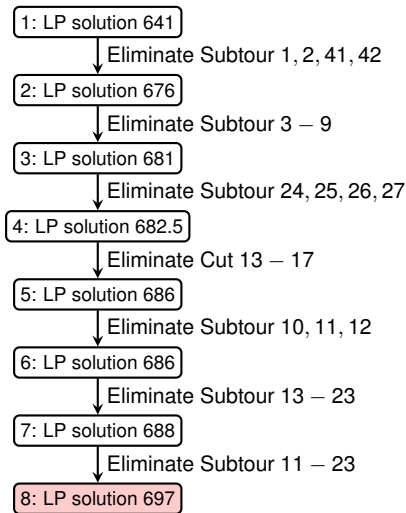
Alternative Branch 1b: $x_{18,15} = 0$, Objective 698



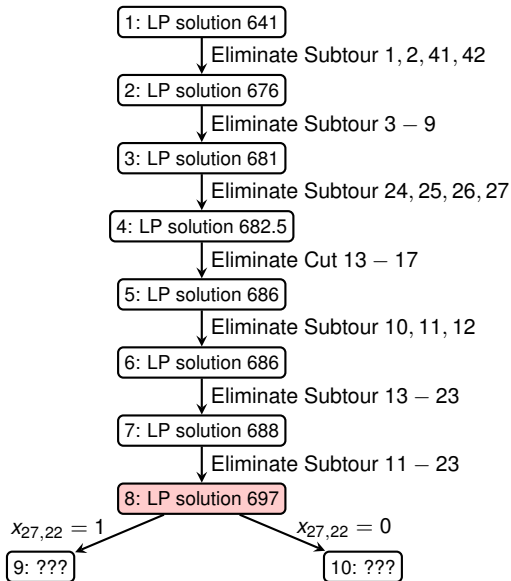
Solving Progress (Alternative Branch 1)



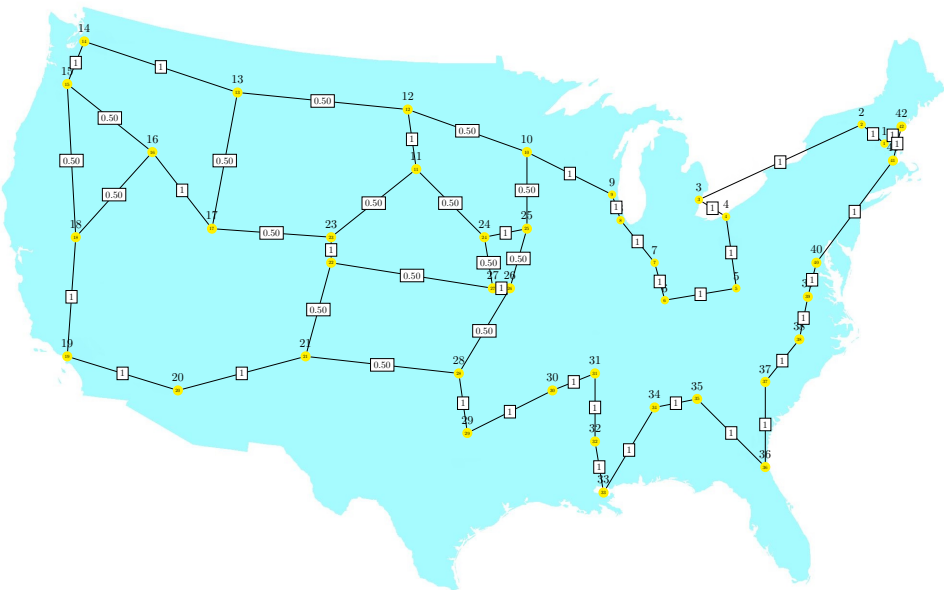
Solving Progress (Alternative Branch 2)



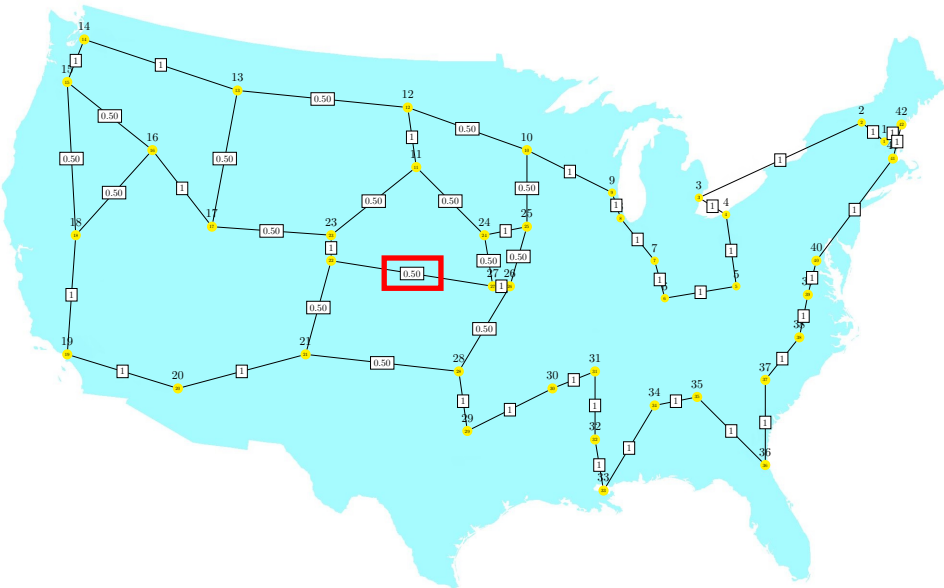
Solving Progress (Alternative Branch 2)



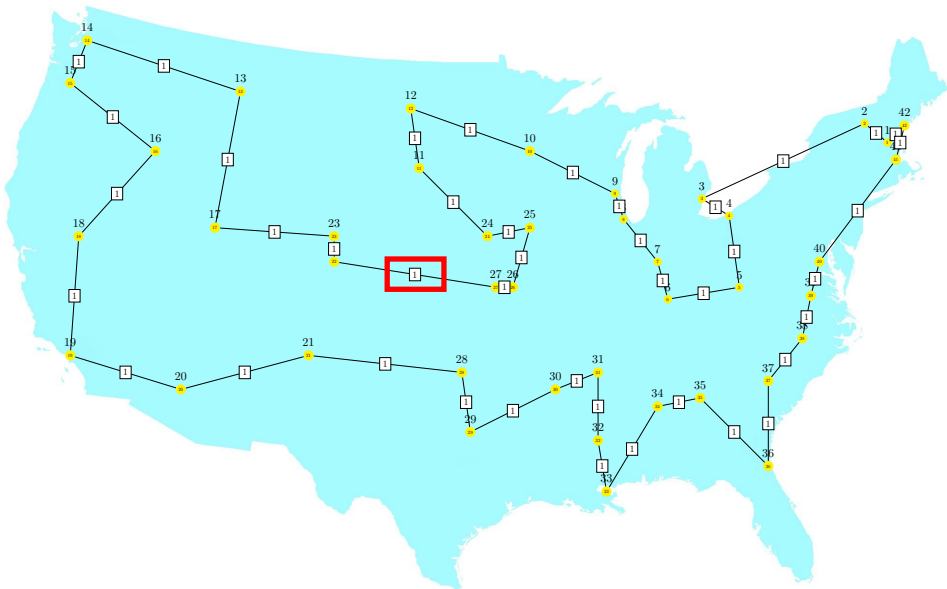
Alternative Branch 2: $x_{27,22}$, Objective 697



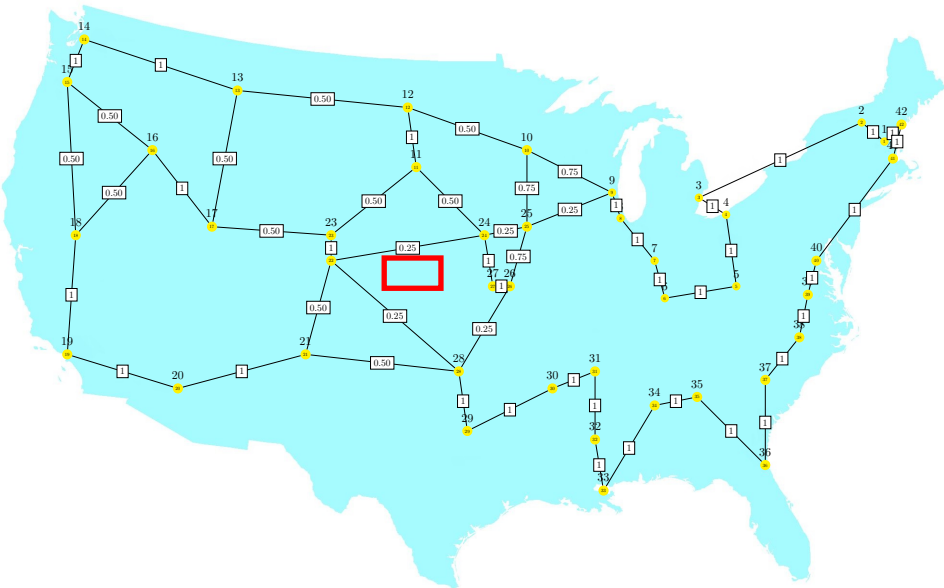
Alternative Branch 2: $x_{27,22}$, Objective 697



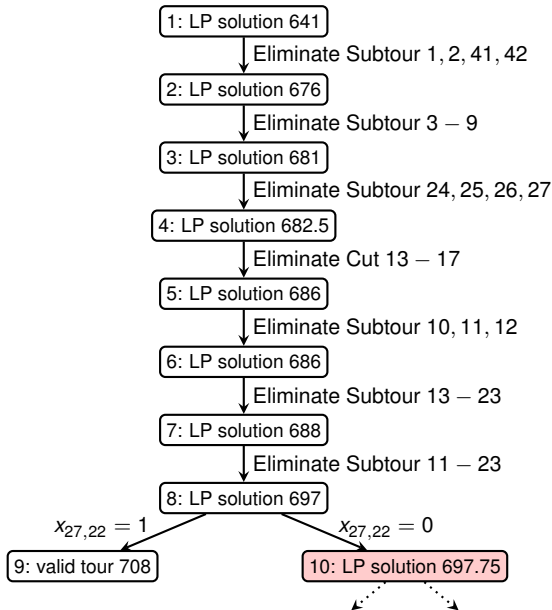
Alternative Branch 2a: $x_{27,22} = 1$, Objective 708 (Valid tour)



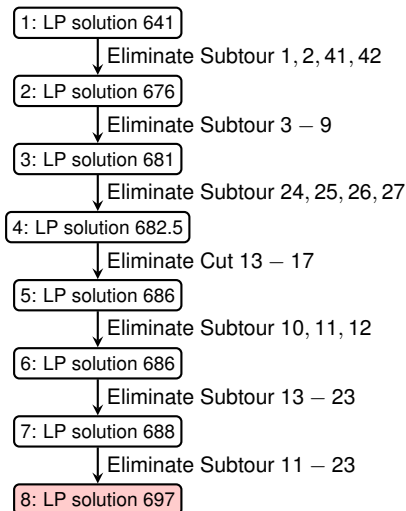
Alternative Branch 2b: $x_{27,22} = 0$, Objective 697.75



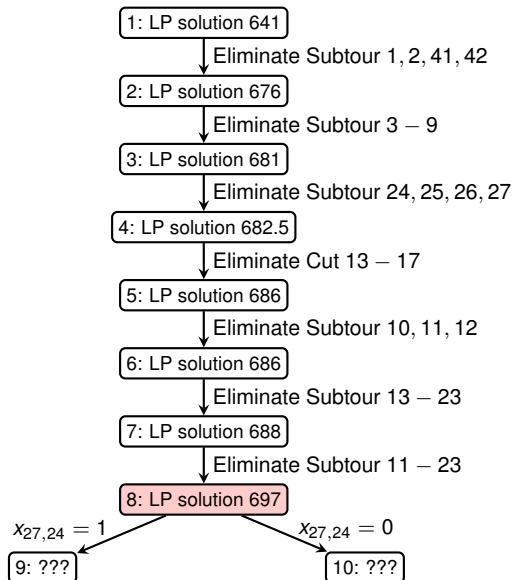
Solving Progress (Alternative Branch 2)



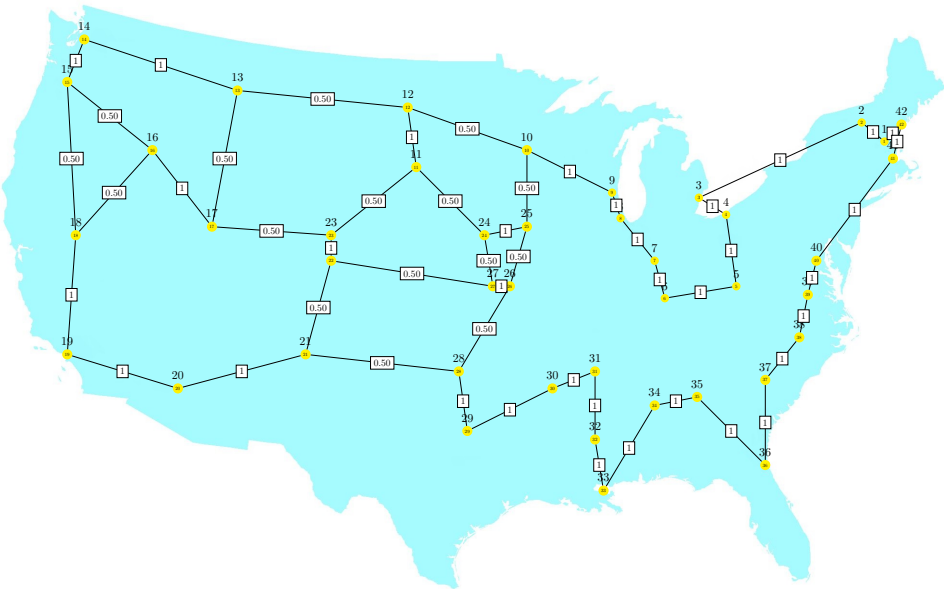
Solving Progress (Alternative Branch 3)



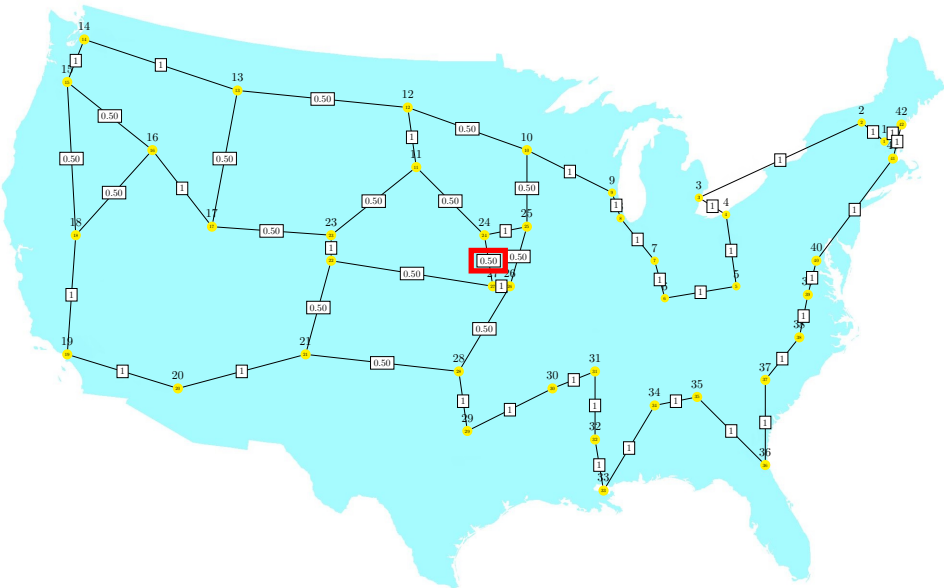
Solving Progress (Alternative Branch 3)



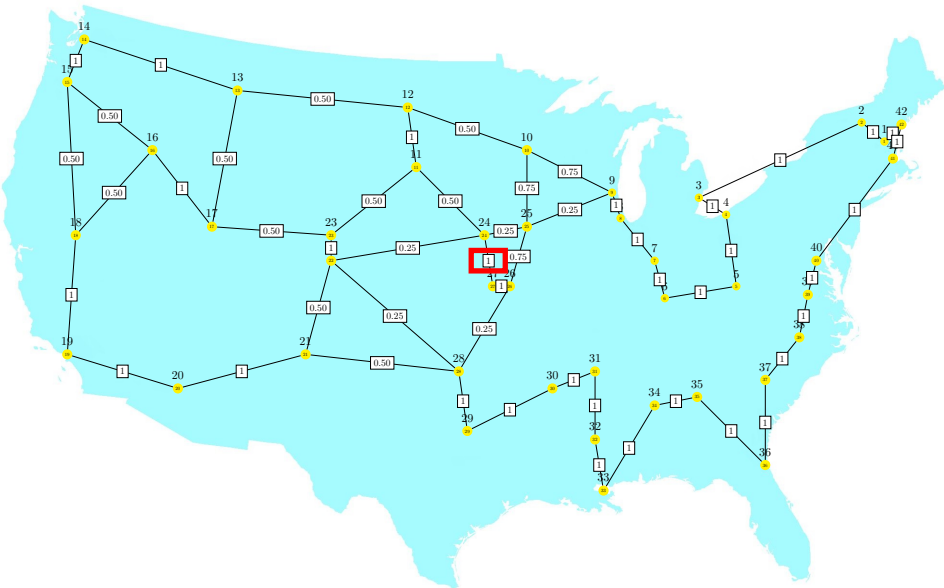
Alternative Branch 3: $x_{27,24}$, Objective 697



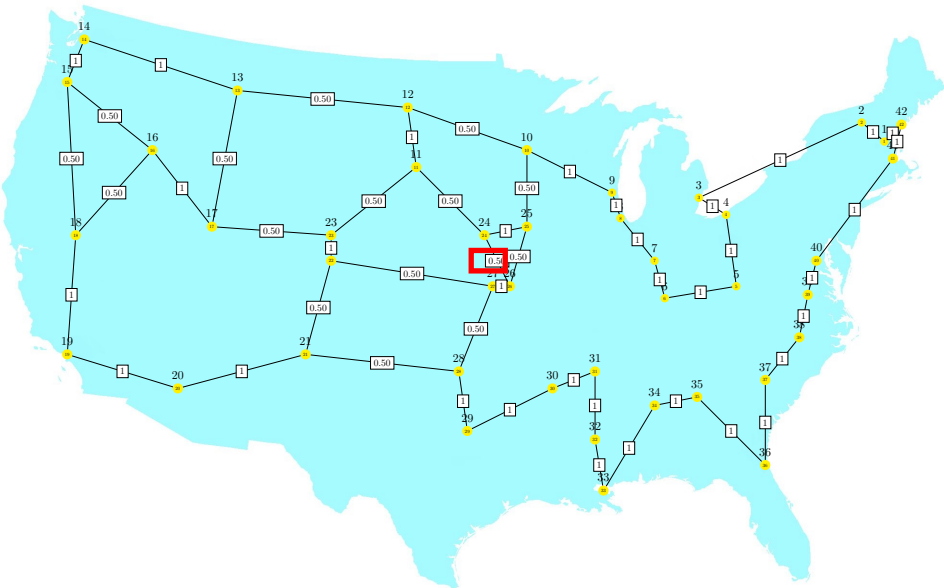
Alternative Branch 3: $x_{27,24}$, Objective 697



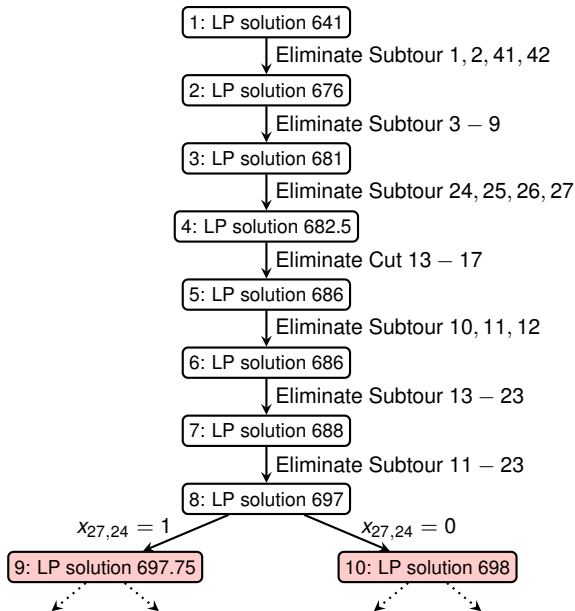
Alternative Branch 3a: $x_{27,24} = 1$, Objective 697.75



Alternative Branch 3b: $x_{27,24} = 0$, Objective 698



Solving Progress (Alternative Branch 3)



Solving Progress (Alternative Branch 3)

1: LP solution 641

Eliminate Subtour 1, 2, 41, 42

2: LP solution 676

Eliminate Subtour 3 – 9

3: LP solution 681

Eliminate Subtour 24, 25, 26, 27

4: LP solution 682.5

Not only do we have to explore (and branch further in) both subtrees, but also the optimal tour is in the subtree with larger LP solution!

6: LP solution 686

Eliminate Subtour 13 – 23

7: LP solution 688

Eliminate Subtour 11 – 23

8: LP solution 697

$x_{27,24} = 1$

9: LP solution 697.75

$x_{27,24} = 0$

10: LP solution 698

Conclusion (1/2)

- How can one generate these constraints automatically?

Conclusion (1/2)

- How can one generate these constraints automatically?
Subtour Elimination: Finding Connected Components
Small Cuts: Finding the Minimum Cut in Weighted Graphs

Conclusion (1/2)

- How can one generate these constraints automatically?
Subtour Elimination: Finding Connected Components
Small Cuts: Finding the Minimum Cut in Weighted Graphs
- Why don't we add all possible Subtour Elimination constraints to the LP?

Conclusion (1/2)

- How can one generate these constraints automatically?
Subtour Elimination: Finding Connected Components
Small Cuts: Finding the Minimum Cut in Weighted Graphs
- Why don't we add all possible Subtour Elimination constraints to the LP?
There are exponentially many of them!

Conclusion (1/2)

- How can one generate these constraints automatically?
Subtour Elimination: Finding Connected Components
Small Cuts: Finding the Minimum Cut in Weighted Graphs
- Why don't we add all possible Subtour Elimination constraints to the LP?
There are exponentially many of them!
- Should the search tree be explored by BFS or DFS?

Conclusion (1/2)

- How can one generate these constraints automatically?
Subtour Elimination: Finding Connected Components
Small Cuts: Finding the Minimum Cut in Weighted Graphs
- Why don't we add all possible Subtour Elimination constraints to the LP?
There are exponentially many of them!
- Should the search tree be explored by BFS or DFS?
BFS may be more attractive, even though it might need more memory.

- How can one generate these constraints automatically?
Subtour Elimination: Finding Connected Components
Small Cuts: Finding the Minimum Cut in Weighted Graphs
- Why don't we add all possible Subtour Elimination constraints to the LP?
There are exponentially many of them!
- Should the search tree be explored by BFS or DFS?
BFS may be more attractive, even though it might need more memory.

CONCLUDING REMARK

It is clear that we have left unanswered practically any question one might pose of a theoretical nature concerning the traveling-salesman problem; however, we hope that the feasibility of attacking problems involving a moderate number of points has been successfully demonstrated, and that perhaps some of the ideas can be used in problems of similar nature.

Conclusion (2/2)

- Eliminate Subtour 1, 2, 41, 42
- Eliminate Subtour 3 – 9
- **Eliminate Subtour 10, 11, 12**
- Eliminate Subtour 11 – 23
- Eliminate Subtour 13 – 23
- Eliminate Cut 13 – 17
- Eliminate Subtour 24, 25, 26, 27

Conclusion (2/2)

- Eliminate Subtour 1, 2, 41, 42
- Eliminate Subtour 3 – 9
- **Eliminate Subtour 10, 11, 12**
- Eliminate Subtour 11 – 23
- Eliminate Subtour 13 – 23
- Eliminate Cut 13 – 17
- Eliminate Subtour 24, 25, 26, 27

THE 49-CITY PROBLEM*

The optimal tour \bar{x} is shown in Fig. 16. The proof that it is optimal is given in Fig. 17. To make the correspondence between the latter and its programming problem clear, we will write down in addition to 42 relations in non-negative variables (2), a set of 25 relations which suffice to prove that $D(x)$ is a minimum for \bar{x} . We distinguish the following subsets of the 42 cities:

$$S_1 = \{1, 2, 41, 42\}$$

$$S_2 = \{3, 4, \dots, 9\}$$

$$S_3 = \{1, 2, \dots, 9, 29, 30, \dots, 42\}$$

$$S_4 = \{11, 12, \dots, 23\}$$

$$S_5 = \{13, 14, \dots, 23\}$$

$$S_6 = \{13, 14, 15, 16, 17\}$$

$$S_7 = \{24, 25, 26, 27\}.$$

Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.1.0
with Simplex, Mixed Integer & Barrier Optimizers
5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
Copyright IBM Corp. 1988, 2014. All Rights Reserved.

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

```
CPLEX> read tsp.lp
Problem 'tsp.lp' read.
Read time = 0.00 sec. (0.06 ticks)
CPLEX> primopt
Tried aggregator 1 time.
LP Presolve eliminated 1 rows and 1 columns.
Reduced LP has 49 rows, 860 columns, and 2483 nonzeros.
Presolve time = 0.00 sec. (0.36 ticks)
```

```
Iteration log . . .
Iteration:    1   Infeasibility =          33.999999
Iteration:   26   Objective      =        1510.000000
Iteration:   90   Objective      =          923.000000
Iteration:  155   Objective      =          711.000000
```

```
Primal simplex - Optimal: Objective = 6.9900000000e+02
Solution time =    0.00 sec. Iterations = 168 (25)
Deterministic time = 1.16 ticks (288.86 ticks/sec)
```

```
CPLEX> █
```

```

CPLEX> display solution variables -
Variable Name      Solution Value
x_2_1              1.000000
x_42_1             1.000000
x_3_2              1.000000
x_4_3              1.000000
x_5_4              1.000000
x_6_5              1.000000
x_7_6              1.000000
x_8_7              1.000000
x_9_8              1.000000
x_10_9             1.000000
x_11_10            1.000000
x_12_11            1.000000
x_13_12            1.000000
x_14_13            1.000000
x_15_14            1.000000
x_16_15            1.000000
x_17_16            1.000000
x_18_17            1.000000
x_19_18            1.000000
x_20_19            1.000000
x_21_20            1.000000
x_22_21            1.000000
x_23_22            1.000000
x_24_23            1.000000
x_25_24            1.000000
x_26_25            1.000000
x_27_26            1.000000
x_28_27            1.000000
x_29_28            1.000000
x_30_29            1.000000
x_31_30            1.000000
x_32_31            1.000000
x_33_32            1.000000
x_34_33            1.000000
x_35_34            1.000000
x_36_35            1.000000
x_37_36            1.000000
x_38_37            1.000000
x_39_38            1.000000
x_40_39            1.000000
x_41_40            1.000000
x_42_41            1.000000

```

All other variables in the range 1-861 are 0.

Randomised Algorithms

Lecture 9: Approximation Algorithms: MAX-3-CNF and Vertex-Cover

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Approximation Ratio for Randomised Approximation Algorithms

Approximation Ratio

A **randomised** algorithm for a problem has **approximation ratio** $\rho(n)$, if for any input of size n , the **expected** cost (value) $\mathbf{E}[C]$ of the returned solution and optimal cost C^* satisfy:

$$\max \left(\frac{\mathbf{E}[C]}{C^*}, \frac{C^*}{\mathbf{E}[C]} \right) \leq \rho(n).$$

Approximation Ratio for Randomised Approximation Algorithms

Approximation Ratio

A **randomised** algorithm for a problem has **approximation ratio** $\rho(n)$, if for any input of size n , the **expected** cost (value) $\mathbf{E}[C]$ of the returned solution and optimal cost C^* satisfy:

$$\max \left(\frac{\mathbf{E}[C]}{C^*}, \frac{C^*}{\mathbf{E}[C]} \right) \leq \rho(n).$$

- **Maximisation** problem: $\frac{C^*}{\mathbf{E}[C]} \geq 1$
- **Minimisation** problem: $\frac{\mathbf{E}[C]}{C^*} \geq 1$

Approximation Ratio for Randomised Approximation Algorithms

Approximation Ratio

A **randomised** algorithm for a problem has **approximation ratio** $\rho(n)$, if for any input of size n , the **expected** cost (value) $\mathbf{E}[C]$ of the returned solution and optimal cost C^* satisfy:

$$\max \left(\frac{\mathbf{E}[C]}{C^*}, \frac{C^*}{\mathbf{E}[C]} \right) \leq \rho(n).$$

not covered here (non-examinable)

Randomised Approximation Schemes

An **approximation scheme** is an approximation algorithm, which given any input and $\epsilon > 0$, is a $(1 + \epsilon)$ -approximation algorithm.

Approximation Ratio for Randomised Approximation Algorithms

Approximation Ratio

A **randomised** algorithm for a problem has **approximation ratio** $\rho(n)$, if for any input of size n , the **expected** cost (value) $\mathbf{E}[C]$ of the returned solution and optimal cost C^* satisfy:

$$\max \left(\frac{\mathbf{E}[C]}{C^*}, \frac{C^*}{\mathbf{E}[C]} \right) \leq \rho(n).$$

not covered here (non-examinable)

Randomised Approximation Schemes

An **approximation scheme** is an approximation algorithm, which given any input and $\epsilon > 0$, is a $(1 + \epsilon)$ -approximation algorithm.

- It is a **polynomial-time approximation scheme** (PTAS) if for any fixed $\epsilon > 0$, the runtime is polynomial in n . (For example, $O(n^{2/\epsilon})$.)
- It is a **fully polynomial-time approximation scheme** (FPTAS) if the runtime is polynomial in both $1/\epsilon$ and n . (For example, $O((1/\epsilon)^2 \cdot n^3)$.)

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$

MAX-3-CNF Satisfiability

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

MAX-3-CNF Satisfiability

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the **satisfiability** problem. Want to compute how “close” the formula to being satisfiable is.

MAX-3-CNF Satisfiability

Assume that no literal (including its negation) appears more than once in the same clause.

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_5) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the **satisfiability** problem. Want to compute how “close” the formula to being satisfiable is.

MAX-3-CNF Satisfiability

Assume that no literal (including its negation) appears more than once in the same clause.

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_5) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the **satisfiability** problem. Want to compute how “close” the formula to being satisfiable is.

Example:

$$(x_1 \vee x_3 \vee \overline{x}_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_5) \wedge (x_2 \vee \overline{x}_4 \vee x_5) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$$

MAX-3-CNF Satisfiability

Assume that no literal (including its negation) appears more than once in the same clause.

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_5) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the **satisfiability** problem. Want to compute how “close” the formula to being satisfiable is.

Example:

$$(x_1 \vee x_3 \vee \overline{x}_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_5) \wedge (x_2 \vee \overline{x}_4 \vee x_5) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$$

$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ and $x_5 = 1$ satisfies 3 (out of 4 clauses)

MAX-3-CNF Satisfiability

Assume that no literal (including its negation) appears more than once in the same clause.

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_5) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the **satisfiability** problem. Want to compute how “close” the formula to being satisfiable is.

Example:

$$(x_1 \vee x_3 \vee \overline{x}_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_5) \wedge (x_2 \vee \overline{x}_4 \vee x_5) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$$

$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ and $x_5 = 1$ satisfies 3 (out of 4 clauses)

Idea: What about assigning each variable uniformly and independently at random?

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

Proof:

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \quad \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 7/8-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y]$$

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right]$$

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right]$$

Linearity of Expectations

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i]$$

Linearity of Expectations

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i] = \sum_{i=1}^m \frac{7}{8}$$

Linearity of Expectations

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i] = \sum_{i=1}^m \frac{7}{8} = \frac{7}{8} \cdot m.$$

Linearity of Expectations

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i] = \sum_{i=1}^m \frac{7}{8} = \frac{7}{8} \cdot m.$$

Linearity of Expectations

maximum number of satisfiable clauses is m

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised 8/7-approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i] = \sum_{i=1}^m \frac{7}{8} = \frac{7}{8} \cdot m. \quad \square$$

Linearity of Expectations

maximum number of satisfiable clauses is m

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Corollary

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.

Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Corollary

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.

There is $\omega \in \Omega$ such that $Y(\omega) \geq \mathbf{E}[Y]$

Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Corollary

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.

There is $\omega \in \Omega$ such that $Y(\omega) \geq \mathbf{E}[Y]$

Probabilistic Method: powerful tool to show existence of a non-obvious property.

Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Corollary

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.

There is $\omega \in \Omega$ such that $Y(\omega) \geq \mathbf{E}[Y]$

Probabilistic Method: powerful tool to show existence of a non-obvious property.

Corollary

Any instance of MAX-3-CNF with at most 7 clauses is satisfiable.

Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Corollary

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.

There is $\omega \in \Omega$ such that $Y(\omega) \geq \mathbf{E}[Y]$

Probabilistic Method: powerful tool to show existence of a non-obvious property.

Corollary

Any instance of MAX-3-CNF with at most 7 clauses is satisfiable.

Follows from the previous Corollary.

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

$$\mathbf{E}[Y] = \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 1] + \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 0].$$

Y is defined as in the previous proof.

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time **randomised $8/7$ -approximation algorithm**.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

$$\mathbf{E}[Y] = \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 1] + \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 0].$$

Y is defined as in the previous proof.

One of the two conditional expectations is at least $\mathbf{E}[Y]$

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time **randomised $8/7$ -approximation algorithm**.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

$$\mathbf{E}[Y] = \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 1] + \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 0].$$

Y is defined as in the previous proof.

One of the two conditional expectations is at least $\mathbf{E}[Y]$

Algorithm: Assign x_1 so that the conditional expectation is maximised and recurse.

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time **randomised 8/7-approximation algorithm**.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

$$\mathbf{E}[Y] = \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 1] + \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 0].$$

Y is defined as in the previous proof.

One of the two conditional expectations is at least $\mathbf{E}[Y]$

GREEDY-3-CNF(ϕ, n, m)

- 1: **for** $j = 1, 2, \dots, n$
- 2: Compute $\mathbf{E}[Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$
- 3: Compute $\mathbf{E}[Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 0]$
- 4: Let $x_j = v_j$ so that the conditional expectation is maximised
- 5: **return** the assignment v_1, v_2, \dots, v_n

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time **randomised $8/7$ -approximation algorithm**.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

$$\mathbf{E}[Y] = \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 1] + \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 0].$$

Y is defined as in the previous proof.

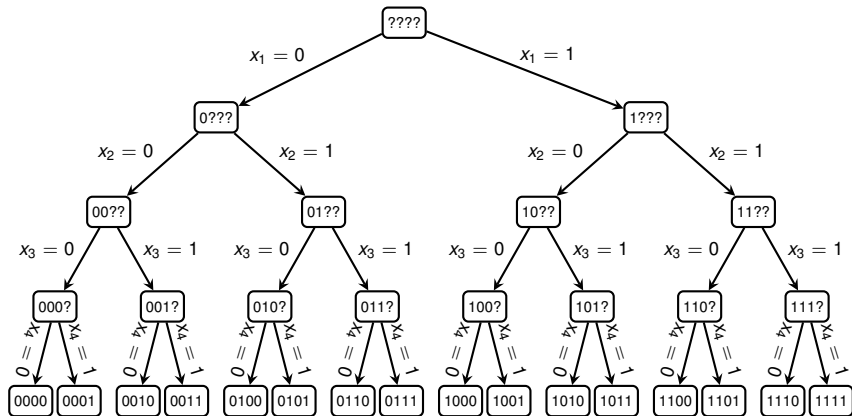
One of the two conditional expectations is at least $\mathbf{E}[Y]$

GREEDY-3-CNF(ϕ, n, m)

- 1: **for** $j = 1, 2, \dots, n$
- 2: Compute $\mathbf{E}[Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$
- 3: Compute $\mathbf{E}[Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 0]$
- 4: Let $x_j = v_j$ so that the conditional expectation is maximised
- 5: **return** the assignment v_1, v_2, \dots, v_n

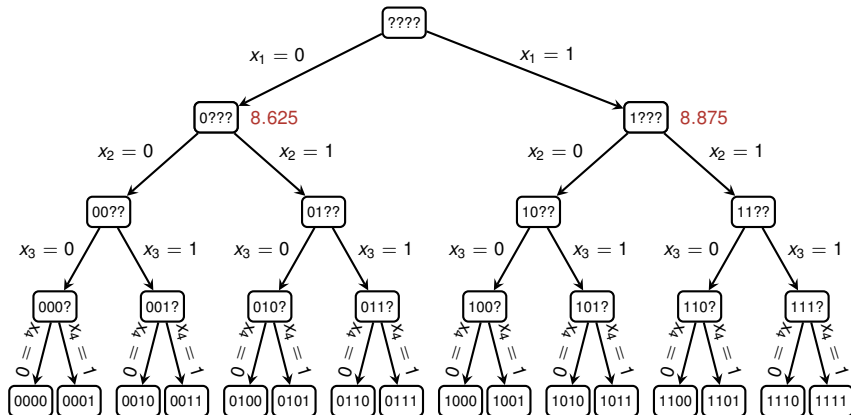
Run of GREEDY-3-CNF(φ, n, m)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



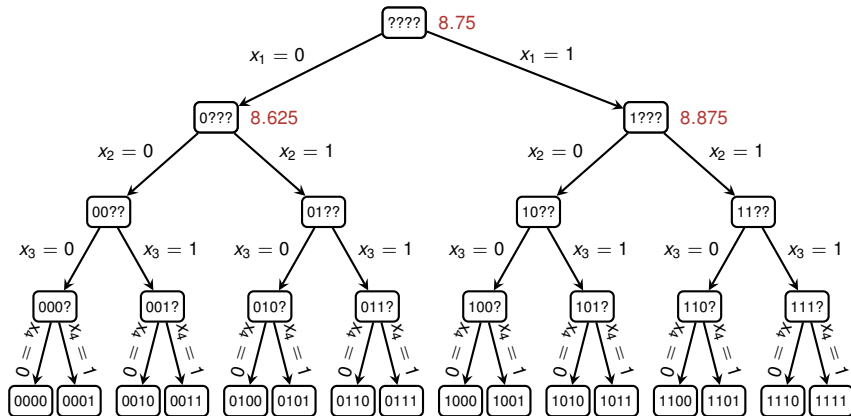
Run of GREEDY-3-CNF(φ, n, m)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



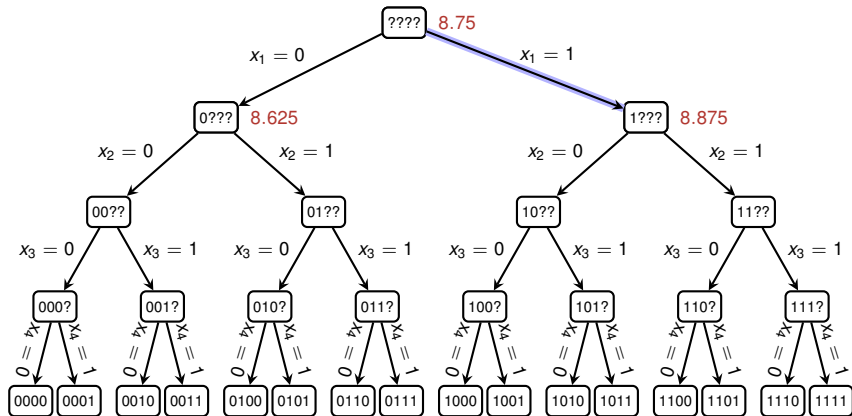
Run of GREEDY-3-CNF(φ, n, m)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



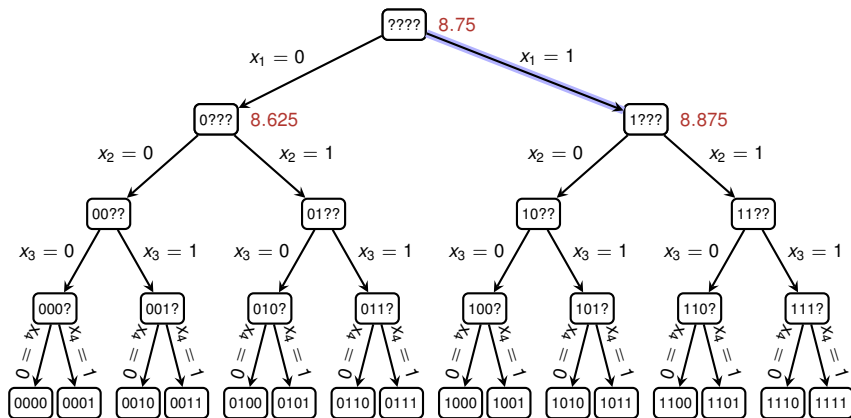
Run of GREEDY-3-CNF(φ, n, m)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



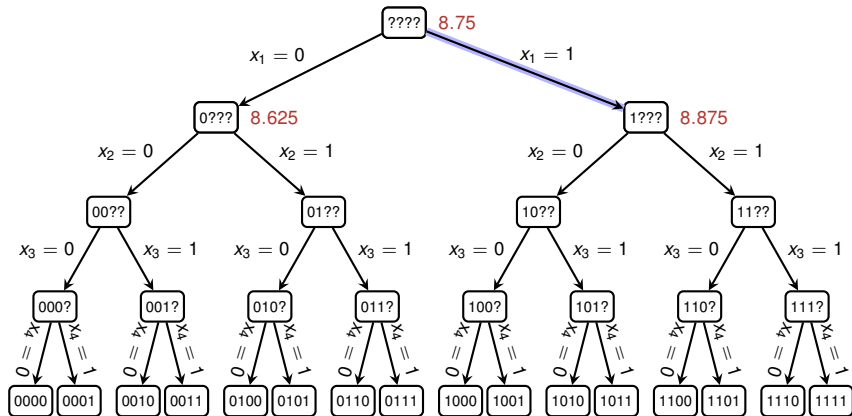
Run of GREEDY-3-CNF(φ, n, m)

$$\cancel{(x_1 \vee x_2 \vee x_3)} \wedge \cancel{(x_1 \vee \bar{x}_2 \vee \bar{x}_4)} \wedge \cancel{(x_1 \vee x_2 \vee \bar{x}_4)} \wedge \cancel{(\bar{x}_1 \vee \bar{x}_3 \vee x_4)} \wedge \cancel{(x_1 \vee x_2 \vee \bar{x}_4)} \wedge \cancel{(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)} \wedge \cancel{(\bar{x}_1 \vee x_2 \vee x_3)} \wedge \cancel{(\bar{x}_1 \vee \bar{x}_2 \vee x_3)} \wedge \cancel{(x_1 \vee x_3 \vee \bar{x}_4)} \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$



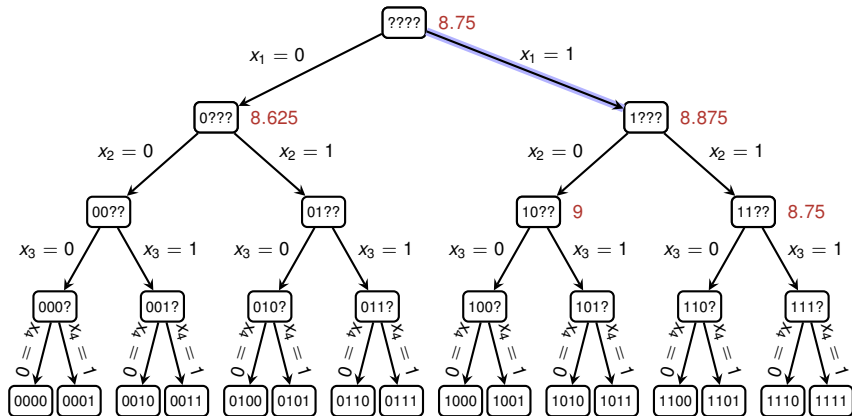
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



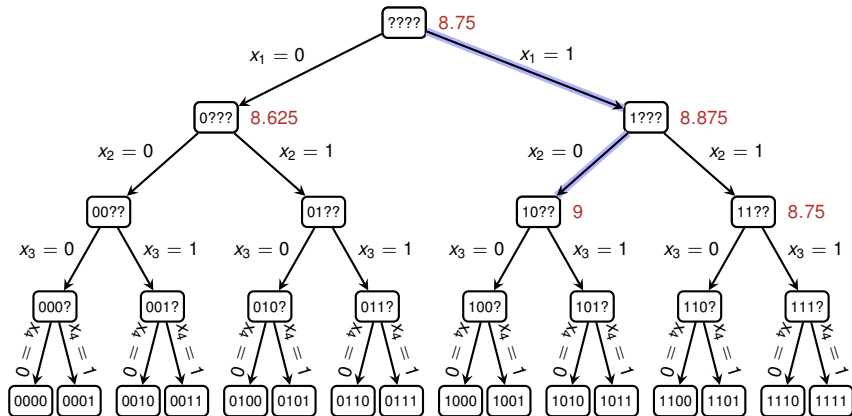
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



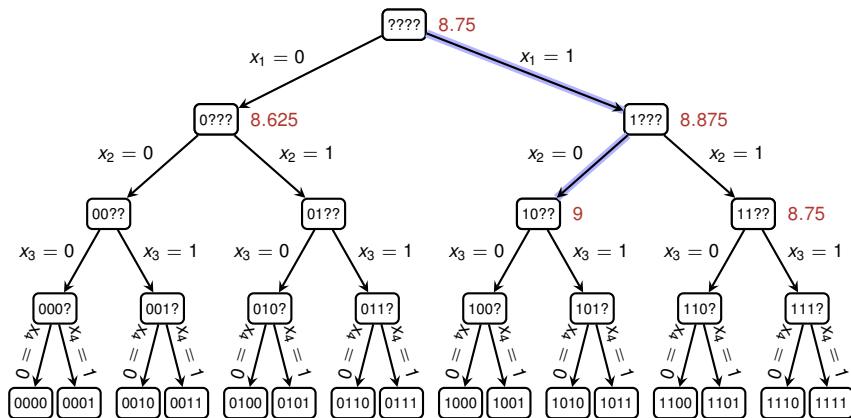
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



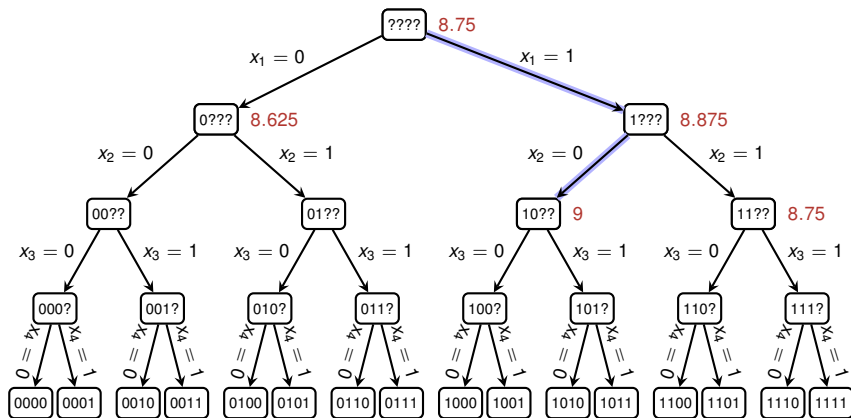
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$$



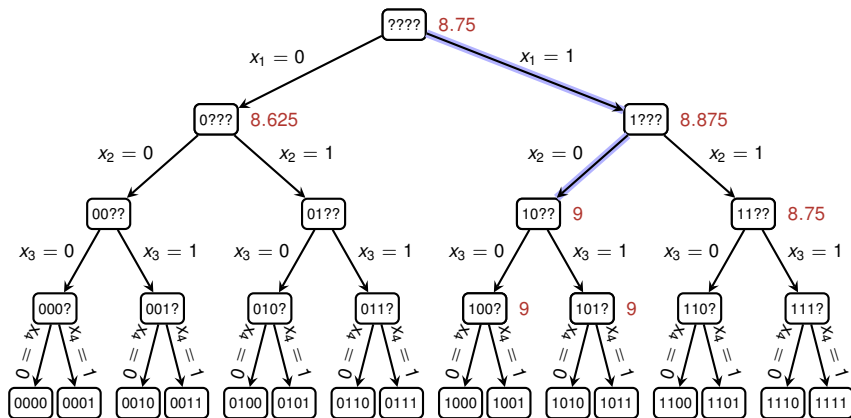
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



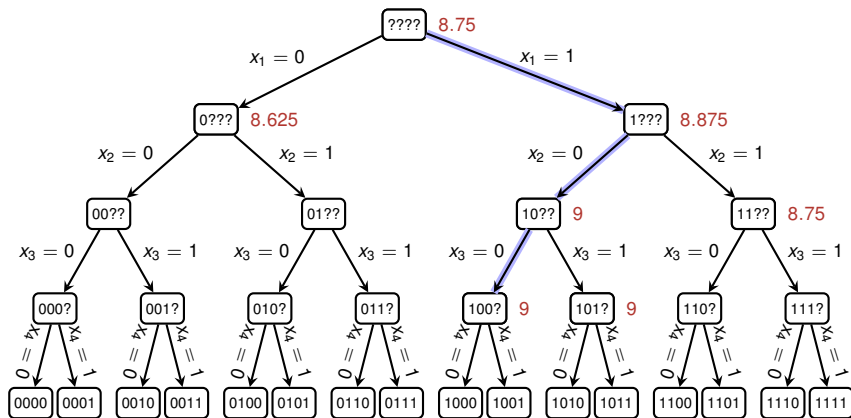
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



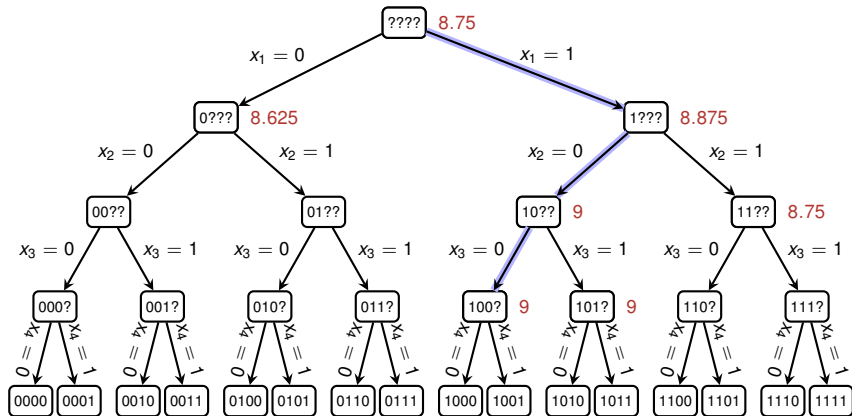
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



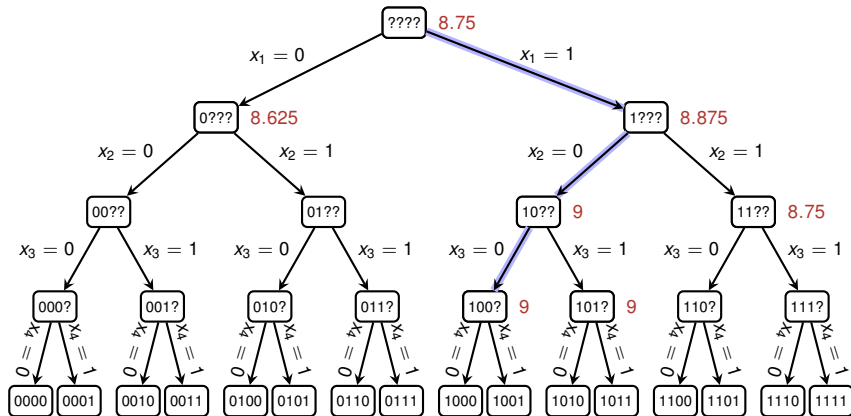
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (\overline{x_3}) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



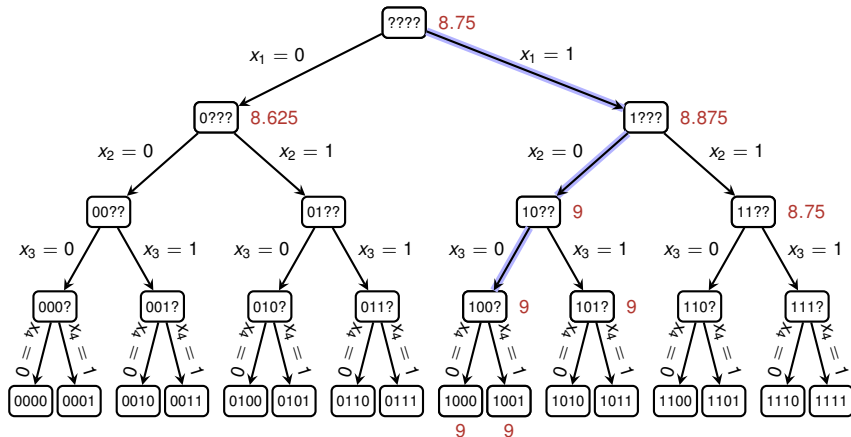
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$$



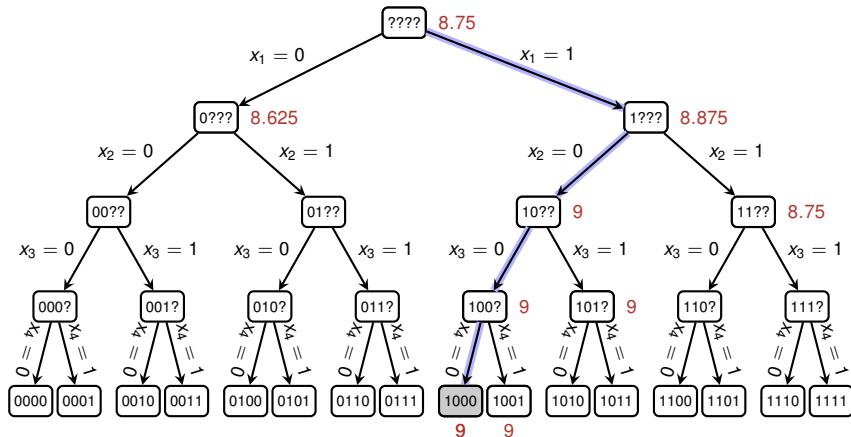
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$$



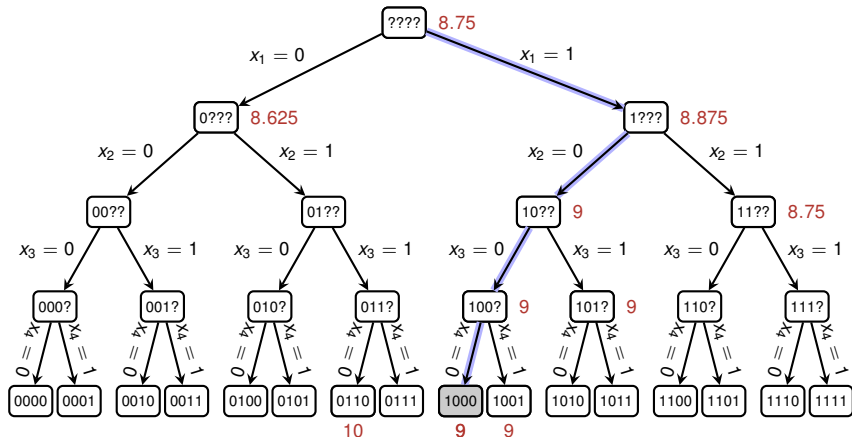
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$$



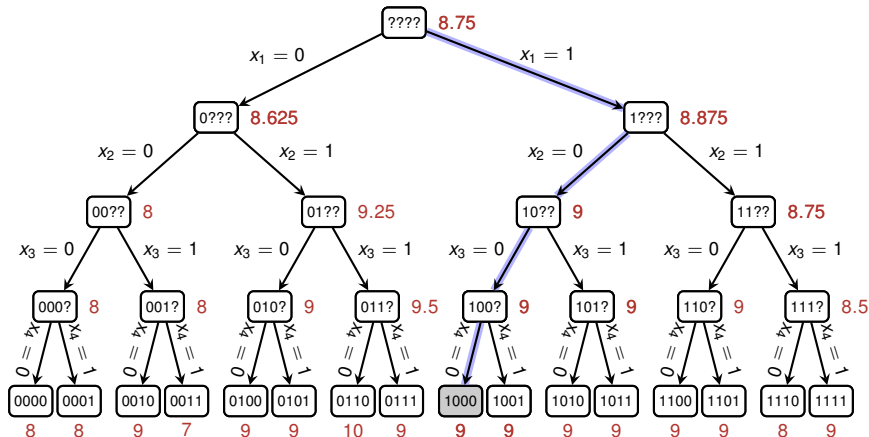
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$$



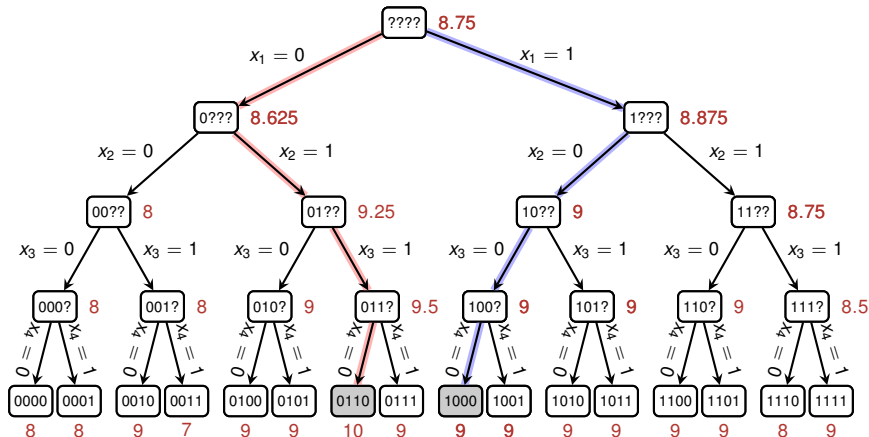
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



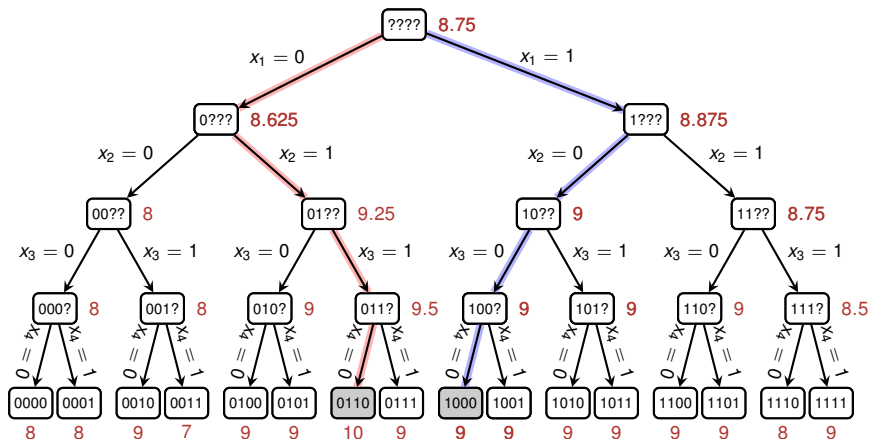
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



Run of GREEDY-3-CNF(φ, n, m)

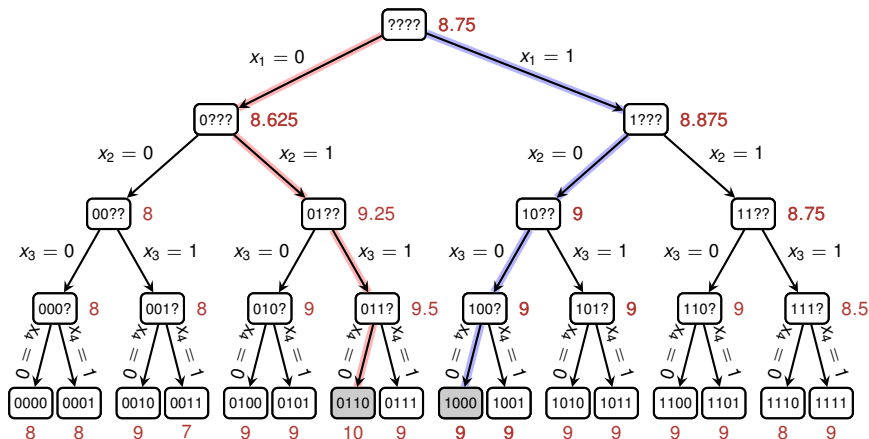
$$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$$



Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.

Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



Analysis of GREEDY-3-CNF(ϕ, n, m)

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

computable in $O(1)$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

computable in $O(1)$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,
$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] \geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}]$$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,
$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \end{aligned}$$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,
$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \\ &\vdots \\ &\geq \mathbf{E} [Y] \end{aligned}$$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses

- Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,

$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \\ &\vdots \\ &\geq \mathbf{E} [Y] = \frac{7}{8} \cdot m. \end{aligned}$$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses ✓
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,
$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \\ &\vdots \\ &\geq \mathbf{E} [Y] = \frac{7}{8} \cdot m. \end{aligned}$$

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses ✓
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,
$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \\ &\vdots \\ &\geq \mathbf{E} [Y] = \frac{7}{8} \cdot m. \end{aligned}$$

□

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

- **Step 1:** polynomial-time algorithm ✓
 - In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
 - A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses ✓
 - Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,
$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \\ &\vdots \\ &\geq \mathbf{E} [Y] = \frac{7}{8} \cdot m. \end{aligned}$$

□

MAX-3-CNF: Concluding Remarks

— Theorem 35.6 —

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

MAX-3-CNF: Concluding Remarks

— Theorem 35.6 —

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

— Theorem —

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

MAX-3-CNF: Concluding Remarks

— Theorem 35.6 —

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

— Theorem —

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

— Theorem (Hastad'97) —

For any $\epsilon > 0$, there is **no** polynomial time $8/7 - \epsilon$ **approximation algorithm** of MAX3-CNF unless $P=NP$.

MAX-3-CNF: Concluding Remarks

— Theorem 35.6 —

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

— Theorem —

$\text{GREEDY-3-CNF}(\phi, n, m)$ is a polynomial-time $8/7$ -approximation.

— Theorem (Hastad'97) —

For any $\epsilon > 0$, there is **no** polynomial time $8/7 - \epsilon$ **approximation algorithm** of MAX3-CNF unless $P=NP$.

Essentially there is nothing smarter than just guessing!



Source of Image: Stefan Szeider, TU Vienna



Source of Image: Stefan Szeider, TU Vienna

Yes, my research has finally concluded...

So you said you have been studying the field of algorithms for MAX-3-SAT?

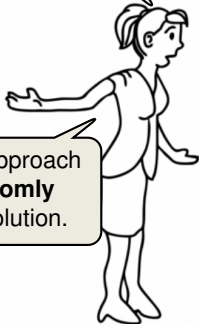


Source of Image: Stefan Szeider, TU Vienna

Yes, my research has finally concluded...

So you said you have been studying the field of algorithms for MAX-3-SAT?

...the best approach is to **randomly guess** a solution.



Source of Image: Stefan Szeider, TU Vienna

Outline

Randomised Approximation

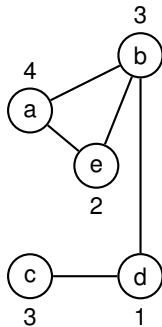
MAX-3-CNF

Weighted Vertex Cover

The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.



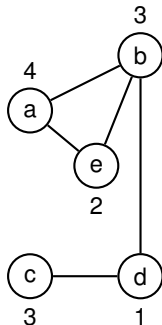
The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.



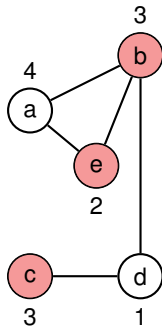
Question: How can we deal with graphs that have **negative** weights?



The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

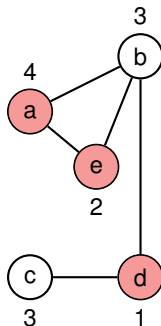
- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.



The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

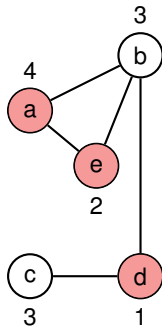


The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.

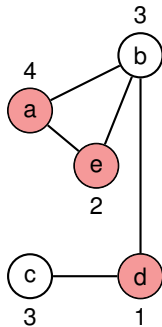


The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.



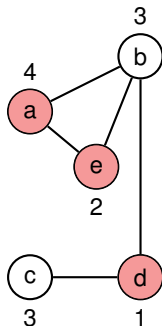
Applications:

The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.



Applications:

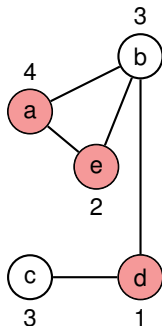
- Every **edge** forms a **task**, and every **vertex** represents a **person/machine** which can execute that task

The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.



Applications:

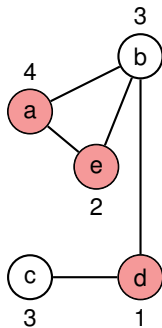
- Every **edge** forms a **task**, and every **vertex** represents a **person/machine** which can execute that task
- **Weight** of a vertex could be **salary** of a person

The **Weighted** Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.



Applications:

- Every **edge** forms a **task**, and every **vertex** represents a **person/machine** which can execute that task
- **Weight** of a vertex could be **salary** of a person
- Perform all tasks with the **minimal amount of resources**

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

```
1   $C = \emptyset$ 
2   $E' = G.E$ 
3  while  $E' \neq \emptyset$ 
4      let  $(u, v)$  be an arbitrary edge of  $E'$ 
5       $C = C \cup \{u, v\}$ 
6      remove from  $E'$  every edge incident on either  $u$  or  $v$ 
7  return  $C$ 
```

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

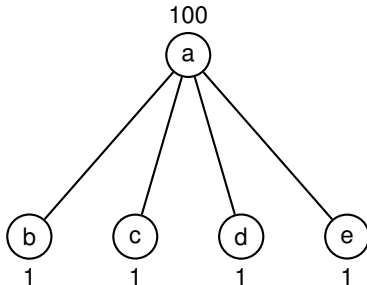
```
1   $C = \emptyset$ 
2   $E' = G.E$ 
3  while  $E' \neq \emptyset$ 
4      let  $(u, v)$  be an arbitrary edge of  $E'$ 
5       $C = C \cup \{u, v\}$ 
6      remove from  $E'$  every edge incident on either  $u$  or  $v$ 
7  return  $C$ 
```

This algorithm is a 2-approximation for unweighted graphs!

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

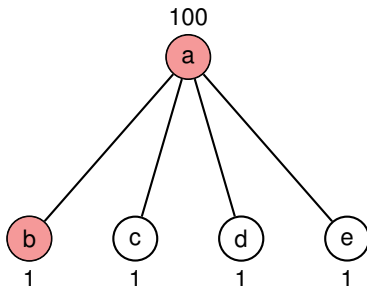
```
1  $C = \emptyset$ 
2  $E' = G.E$ 
3 while  $E' \neq \emptyset$ 
4   let  $(u, v)$  be an arbitrary edge of  $E'$ 
5    $C = C \cup \{u, v\}$ 
6   remove from  $E'$  every edge incident on either  $u$  or  $v$ 
7 return  $C$ 
```



A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

```
1  $C = \emptyset$   
2  $E' = G.E$   
3 while  $E' \neq \emptyset$   
4   let  $(u, v)$  be an arbitrary edge of  $E'$   
5    $C = C \cup \{u, v\}$   
6   remove from  $E'$  every edge incident on either  $u$  or  $v$   
7 return  $C$ 
```

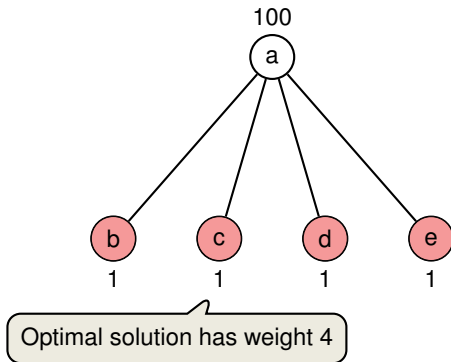


Computed solution has weight 101

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

```
1  $C = \emptyset$ 
2  $E' = G.E$ 
3 while  $E' \neq \emptyset$ 
4   let  $(u, v)$  be an arbitrary edge of  $E'$ 
5    $C = C \cup \{u, v\}$ 
6   remove from  $E'$  every edge incident on either  $u$  or  $v$ 
7 return  $C$ 
```



Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in \{0, 1\} \quad \text{for each } v \in V \end{array}$$

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in \{0, 1\} \quad \text{for each } v \in V \end{array}$$

Linear Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in [0, 1] \quad \text{for each } v \in V \end{array}$$

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in \{0, 1\} \quad \text{for each } v \in V \end{array}$$

optimum is a lower bound on the optimal weight of a minimum weight-cover.

Linear Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in [0, 1] \quad \text{for each } v \in V \end{array}$$

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in \{0, 1\} \quad \text{for each } v \in V \end{array}$$

optimum is a lower bound on the optimal weight of a minimum weight-cover.

Linear Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in [0, 1] \quad \text{for each } v \in V \end{array}$$

Rounding Rule: if $x(v) \geq 1/2$ then round up, otherwise round down.

The Algorithm

APPROX-MIN-WEIGHT-VC(G, w)

```
1   $C = \emptyset$ 
2  compute  $\bar{x}$ , an optimal solution to the linear program
3  for each  $v \in V$ 
4      if  $\bar{x}(v) \geq 1/2$ 
5           $C = C \cup \{v\}$ 
6  return  $C$ 
```

The Algorithm

APPROX-MIN-WEIGHT-VC(G, w)

```
1   $C = \emptyset$ 
2  compute  $\bar{x}$ , an optimal solution to the linear program
3  for each  $v \in V$ 
4      if  $\bar{x}(v) \geq 1/2$ 
5           $C = C \cup \{v\}$ 
6  return  $C$ 
```

Theorem 35.7

APPROX-MIN-WEIGHT-VC is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.

The Algorithm

APPROX-MIN-WEIGHT-VC(G, w)

```
1   $C = \emptyset$ 
2  compute  $\bar{x}$ , an optimal solution to the linear program
3  for each  $v \in V$ 
4      if  $\bar{x}(v) \geq 1/2$ 
5           $C = C \cup \{v\}$ 
6  return  $C$ 
```

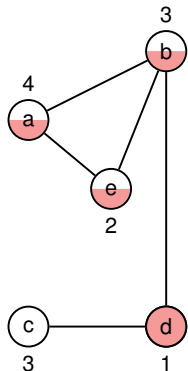
Theorem 35.7

APPROX-MIN-WEIGHT-VC is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.

is polynomial-time because we can solve the linear program in polynomial time

Example of APPROX-MIN-WEIGHT-VC

$$\bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \bar{x}(d) = 1, \bar{x}(c) = 0$$

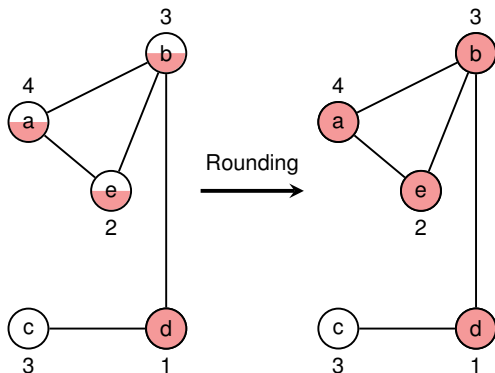


fractional solution of LP
with weight = 5.5

Example of APPROX-MIN-WEIGHT-VC

$$\bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \bar{x}(d) = 1, \bar{x}(c) = 0$$

$$x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0$$

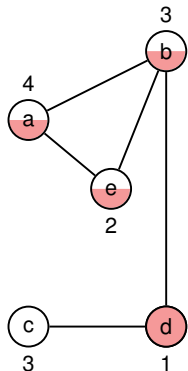


fractional solution of LP
with weight = 5.5

rounded solution of LP
with weight = 10

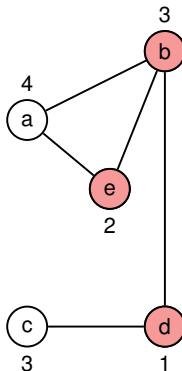
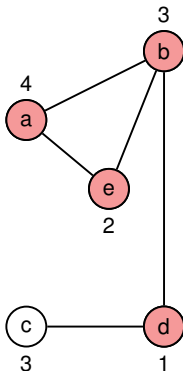
Example of APPROX-MIN-WEIGHT-VC

$$\bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \bar{x}(d) = 1, \bar{x}(c) = 0$$



Rounding
→

$$x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0$$



fractional solution of LP
with weight = 5.5

rounded solution of LP
with weight = 10

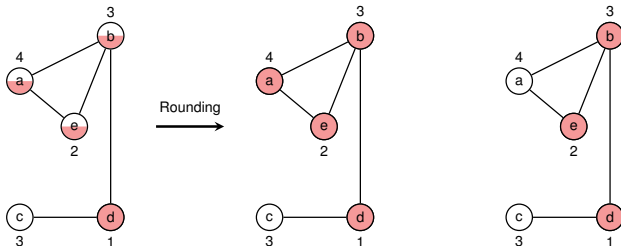
optimal solution
with weight = 6

Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

Approximation Ratio

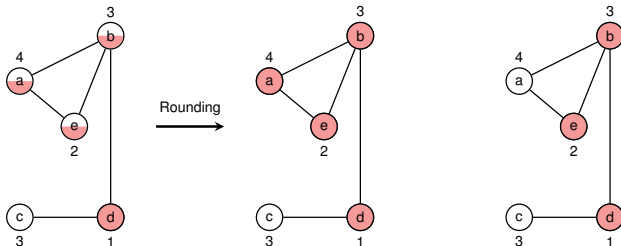
Proof (Approximation Ratio is 2 and Correctness):



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

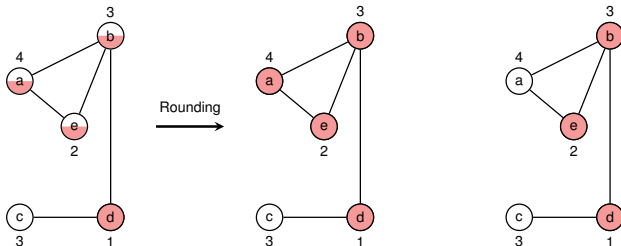
- Let C^* be an optimal solution to the minimum-weight vertex cover problem



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

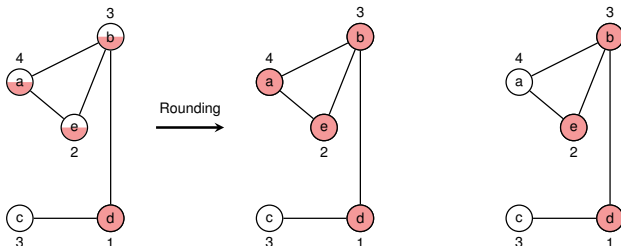


Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$



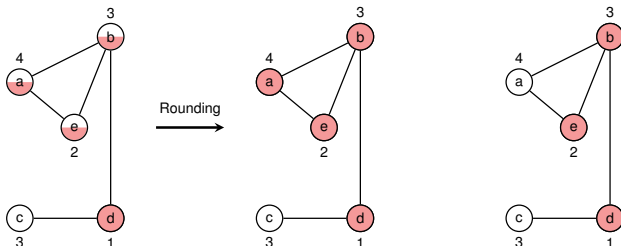
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:



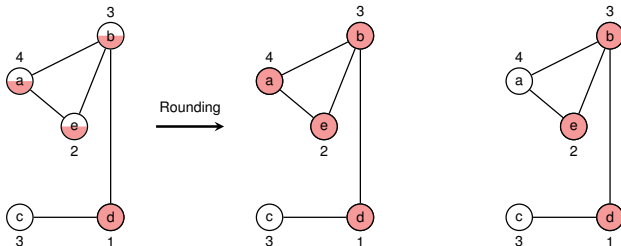
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$



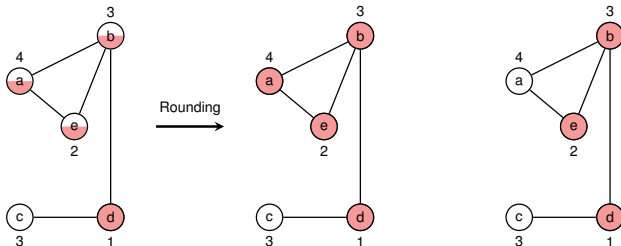
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2$



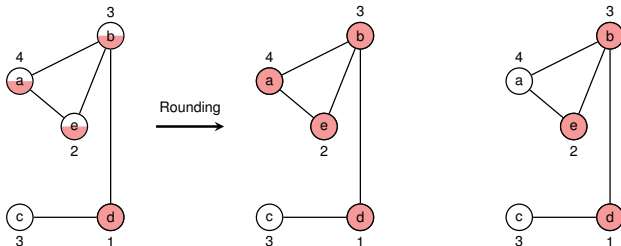
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)



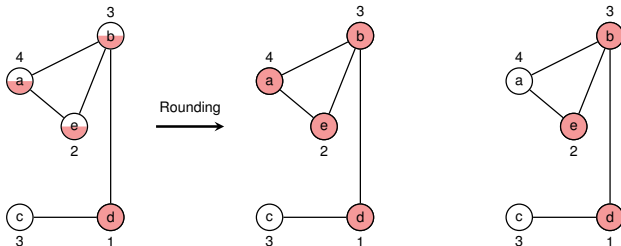
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:



Approximation Ratio

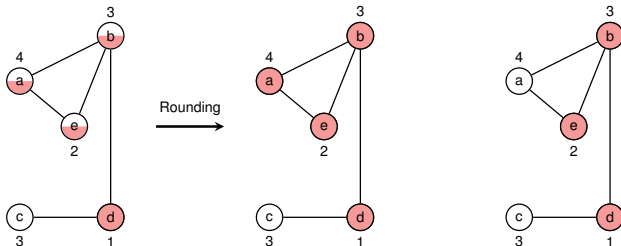
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

z^*



Approximation Ratio

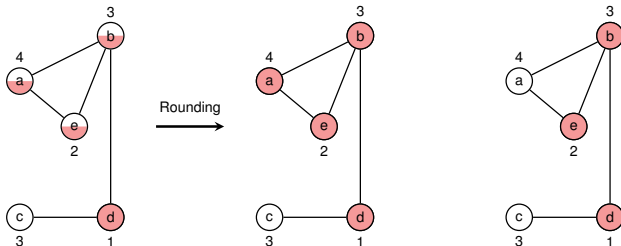
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^*$$



Approximation Ratio

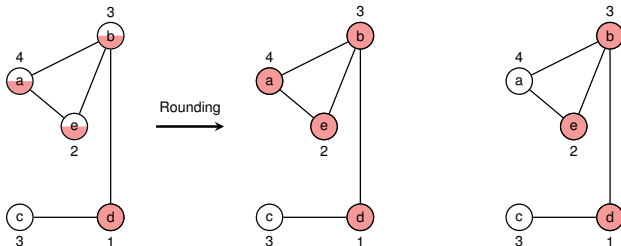
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v) \bar{x}(v)$$



Approximation Ratio

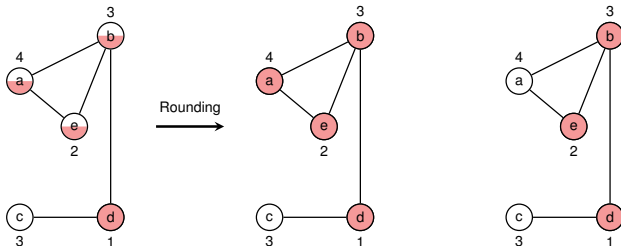
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v) \bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2}$$



Approximation Ratio

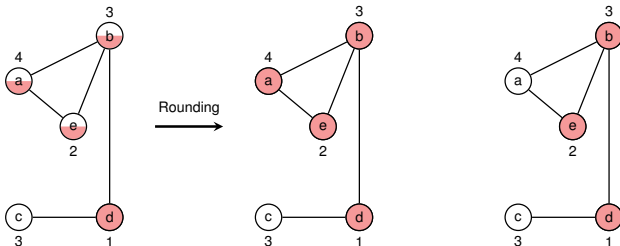
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v) \bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C).$$



Approximation Ratio

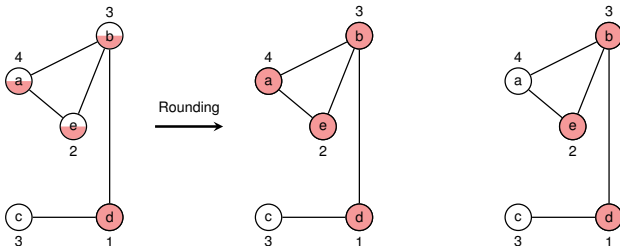
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v) \bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C).$$



Approximation Ratio

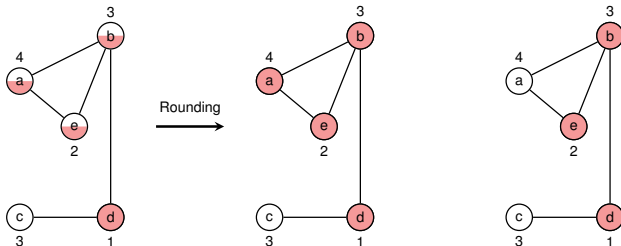
Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v) \bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C). \quad \square$$



Randomised Algorithms

Lecture 10: Approximation Algorithms: Set-Cover and MAX-CNF

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Weighted Set Cover

MAX-CNF

The **Weighted** Set-Cover Problem

Set Cover Problem

- **Given:** set X , $|X| = n$, a family of subsets \mathcal{F} , and **cost function** $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a **minimum-cost** subset $\mathcal{C} \subseteq \mathcal{F}$

$$\text{s.t.} \quad X = \bigcup_{S \in \mathcal{C}} S.$$

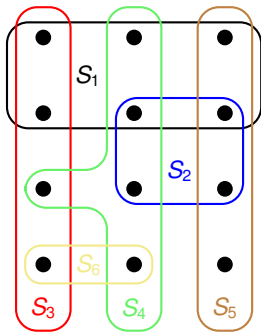
The **Weighted** Set-Cover Problem

Set Cover Problem

- **Given:** set X , $|X| = n$, a family of subsets \mathcal{F} , and **cost function** $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a **minimum-cost** subset $\mathcal{C} \subseteq \mathcal{F}$

Sum over the costs
of all **sets** in \mathcal{C}

$$\text{s.t.} \quad X = \bigcup_{S \in \mathcal{C}} S.$$



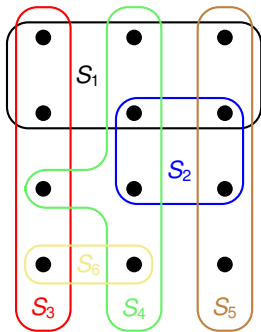
The **Weighted** Set-Cover Problem

Set Cover Problem

- **Given:** set X , $|X| = n$, a family of subsets \mathcal{F} , and **cost function** $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a **minimum-cost** subset $\mathcal{C} \subseteq \mathcal{F}$

Sum over the costs
of all **sets** in \mathcal{C}

$$\text{s.t.} \quad X = \bigcup_{S \in \mathcal{C}} S.$$



	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2

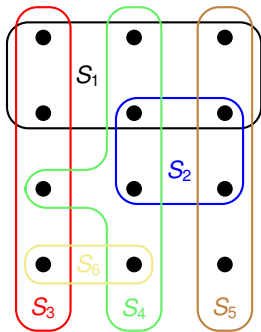
The **Weighted** Set-Cover Problem

Set Cover Problem

- **Given:** set X , $|X| = n$, a family of subsets \mathcal{F} , and **cost function** $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a **minimum-cost** subset $\mathcal{C} \subseteq \mathcal{F}$

Sum over the costs
of all **sets** in \mathcal{C}

$$\text{s.t.} \quad X = \bigcup_{S \in \mathcal{C}} S.$$



	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2

Remarks:

- generalisation of the weighted Vertex-Cover problem
- models resource allocation problems

The **Weighted** Set-Cover Problem

Set Cover Problem

- **Given:** set X , $|X| = n$, a family of subsets \mathcal{F} , and **cost function** $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a **minimum-cost** subset $\mathcal{C} \subseteq \mathcal{F}$

Sum over the costs
of all **sets** in \mathcal{C}

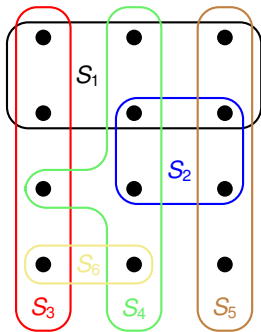
$$\text{s.t.} \quad X = \bigcup_{S \in \mathcal{C}} S.$$



Question: How can we reduce the
Vertex-Cover problem to the
Set-Cover problem?

Remarks:

- generalisation of the weighted Vertex-Cover problem
- models resource allocation problems



	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2



Question: Try to formulate the integer program and linear program of the weighted SET-COVER problem (solution on next slide!)

Setting up an Integer Program

0-1 Integer Program

$$\begin{array}{ll} \text{minimize} & \sum_{S \in \mathcal{F}} c(S)y(S) \\ \text{subject to} & \sum_{S \in \mathcal{F}: x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\ & y(S) \in \{0, 1\} \quad \text{for each } S \in \mathcal{F} \end{array}$$

Setting up an Integer Program

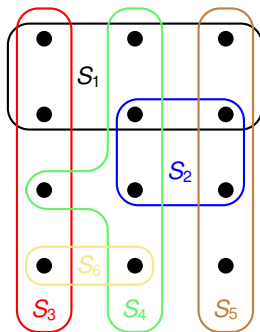
0-1 Integer Program

$$\begin{array}{ll}\text{minimize} & \sum_{S \in \mathcal{F}} c(S)y(S) \\ \text{subject to} & \sum_{S \in \mathcal{F}: x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\ & y(S) \in \{0, 1\} \quad \text{for each } S \in \mathcal{F}\end{array}$$

Linear Program

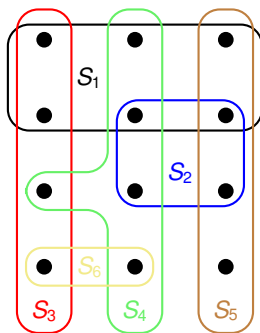
$$\begin{array}{ll}\text{minimize} & \sum_{S \in \mathcal{F}} c(S)y(S) \\ \text{subject to} & \sum_{S \in \mathcal{F}: x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\ & y(S) \in [0, 1] \quad \text{for each } S \in \mathcal{F}\end{array}$$

Back to the Example



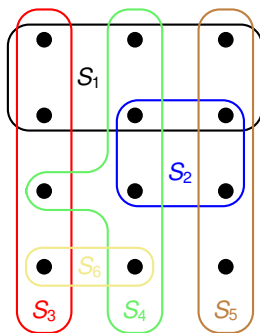
	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2

Back to the Example



	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$

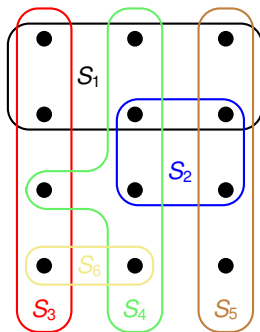
Back to the Example



	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$

Cost equals 8.5

Back to the Example

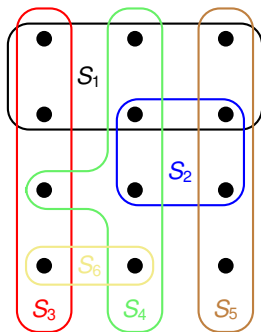


	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Cost equals 8.5

The strategy employed for Vertex-Cover would take all 6 sets!

Back to the Example



	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Cost equals 8.5

The strategy employed for Vertex-Cover would take all 6 sets!

Even worse: If all \bar{y} 's were below 1/2, we would not even return a valid cover!

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Idea: Interpret the \bar{y} -values as **probabilities** for picking the respective set.

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Idea: Interpret the \bar{y} -values as **probabilities** for picking the respective set.

Randomised Rounding

- Let $\mathcal{C} \subseteq \mathcal{F}$ be a **random set** with each set S being included independently with probability $\bar{y}(S)$.
- More precisely, if \bar{y} denotes the optimal solution of the LP, then we compute an integral solution y by:

$$y(S) = \begin{cases} 1 & \text{with probability } \bar{y}(S) \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } S \in \mathcal{F}.$$

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(\cdot) :$	1/2	1/2	1/2	1/2	1	1/2

Idea: Interpret the \bar{y} -values as **probabilities** for picking the respective set.

Randomised Rounding

- Let $\mathcal{C} \subseteq \mathcal{F}$ be a **random set** with each set S being included independently with probability $\bar{y}(S)$.
- More precisely, if \bar{y} denotes the optimal solution of the LP, then we compute an integral solution y by:

$$y(S) = \begin{cases} 1 & \text{with probability } \bar{y}(S) \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } S \in \mathcal{F}.$$

- Therefore, $\mathbf{E}[y(S)] = \bar{y}(S)$.

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Idea: Interpret the \bar{y} -values as **probabilities** for picking the respective set.

Lemma

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(.):$	1/2	1/2	1/2	1/2	1	1/2

Idea: Interpret the \bar{y} -values as **probabilities** for picking the respective set.

Lemma

- The **expected cost** satisfies

$$\mathbf{E} [c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S).$$

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
$c :$	2	3	3	5	1	2
$\bar{y}(\cdot) :$	1/2	1/2	1/2	1/2	1	1/2

Idea: Interpret the \bar{y} -values as **probabilities** for picking the respective set.

Lemma

- The **expected cost** satisfies

$$\mathbf{E} [c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S).$$

- The **probability** that an element $x \in X$ is **covered** satisfies

$$\mathbf{P} \left[x \in \bigcup_{S \in \mathcal{C}} S \right] \geq 1 - \frac{1}{e}.$$

Proof of Lemma

— Lemma —

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C}

Proof of Lemma

— Lemma —

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C}

$$\mathbf{E}[c(\mathcal{C})]$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C}

$$\mathbf{E}[c(\mathcal{C})] = \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right]$$

Proof of Lemma

— Lemma —

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C}

$$\mathbf{E}[c(\mathcal{C})] = \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right]$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C}

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S)\end{aligned}$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C}

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S]$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}]$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.


Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$


$$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$

$$\leq \prod_{S \in \mathcal{F}: x \in S} e^{-\bar{y}(S)}$$

$$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\begin{aligned}\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] &= \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S)) \\ &\leq \prod_{S \in \mathcal{F}: x \in S} e^{-\bar{y}(S)} \\ &= e^{-\sum_{S \in \mathcal{F}: x \in S} \bar{y}(S)}\end{aligned}$$

$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$

$$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$$

$$\begin{aligned}&\leq \prod_{S \in \mathcal{F}: x \in S} e^{-\bar{y}(S)} \\ &= e^{-\sum_{S \in \mathcal{F}: x \in S} \bar{y}(S)}\end{aligned}$$

\bar{y} solves the LP!

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$

$$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$$

$$\leq \prod_{S \in \mathcal{F}: x \in S} e^{-\bar{y}(S)}$$

$$= e^{-\sum_{S \in \mathcal{F}: x \in S} \bar{y}(S)} \leq e^{-1}$$

\bar{y} solves the LP!

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered ✓

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$

$$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$$

$$\leq \prod_{S \in \mathcal{F}: x \in S} e^{-\bar{y}(S)}$$

$$= e^{-\sum_{S \in \mathcal{F}: x \in S} \bar{y}(S)} \leq e^{-1}$$

\bar{y} solves the LP!

Proof of Lemma

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $\bar{y}(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set \mathcal{C} ✓

$$\begin{aligned}\mathbf{E}[c(\mathcal{C})] &= \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S)\right] \\ &= \sum_{S \in \mathcal{F}} \mathbf{P}[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} \bar{y}(S) \cdot c(S).\end{aligned}$$

- **Step 2:** The probability for an element to be (not) covered ✓

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F}: x \in S} \mathbf{P}[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F}: x \in S} (1 - \bar{y}(S))$$

$$1 + x \leq e^x \text{ for any } x \in \mathbb{R}$$

$$\leq \prod_{S \in \mathcal{F}: x \in S} e^{-\bar{y}(S)}$$

\bar{y} solves the LP!

$$= e^{-\sum_{S \in \mathcal{F}: x \in S} \bar{y}(S)} \leq e^{-1} \quad \square$$

The Final Step

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

The Final Step

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Problem: Need to make sure that every element is covered!

The Final Step

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Problem: Need to make sure that every element is covered!

Idea: Amplify this probability by taking the union of $\Omega(\log n)$ random sets \mathcal{C} .

The Final Step

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set S being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that x is covered satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Problem: Need to make sure that every element is covered!

Idea: Amplify this probability by taking the union of $\Omega(\log n)$ random sets \mathcal{C} .

WEIGHTED SET COVER-LP(X, \mathcal{F}, c)

- 1: compute \bar{y} , an optimal solution to the linear program
- 2: $\mathcal{C} = \emptyset$
- 3: **repeat** $2 \ln n$ times
- 4: **for each** $S \in \mathcal{F}$
- 5: let $\mathcal{C} = \mathcal{C} \cup \{S\}$ with probability $\bar{y}(S)$
- 6: **return** \mathcal{C}

The Final Step

Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a **random subset** with each set S being included independently with probability $y(S)$.

- The **expected cost** satisfies $\mathbf{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The **probability** that x is **covered** satisfies $\mathbf{P}[x \in \cup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Problem: Need to make sure that every element is covered!

Idea: Amplify this probability by taking the union of $\Omega(\log n)$ random sets \mathcal{C} .

WEIGHTED SET COVER-LP(X, \mathcal{F}, c)

- 1: compute \bar{y} , an optimal solution to the linear program
- 2: $\mathcal{C} = \emptyset$
- 3: **repeat** $2 \ln n$ times
- 4: **for each** $S \in \mathcal{F}$
- 5: let $\mathcal{C} = \mathcal{C} \cup \{S\}$ with probability $\bar{y}(S)$
- 6: **return** \mathcal{C}

clearly runs in **polynomial-time**!

Analysis of WEIGHTED SET COVER-LP

— Theorem —

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Analysis of WEIGHTED SET COVER-LP

— Theorem —

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

Analysis of WEIGHTED SET COVER-LP

— Theorem —

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n}$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] =$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B]$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S]$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2}$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that **all** elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that **all** elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2:** The expected approximation ratio

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2:** The expected approximation ratio
 - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that all elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2:** The expected approximation ratio
 - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
 - Linearity $\Rightarrow \mathbf{E}[c(\mathcal{C})] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that **all** elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2:** The expected approximation ratio
 - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
 - Linearity $\Rightarrow \mathbf{E}[c(\mathcal{C})] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S) \leq 2 \ln(n) \cdot c(\mathcal{C}^*)$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that **all** elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2:** The expected approximation ratio ✓
 - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
 - Linearity $\Rightarrow \mathbf{E}[c(\mathcal{C})] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S) \leq 2 \ln(n) \cdot c(\mathcal{C}^*)$

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The **probability** that \mathcal{C} is a cover ✓
 - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

- This implies for the event that **all** elements are covered:

$$\mathbf{P}[X = \cup_{S \in \mathcal{C}} S] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B] \quad \geq 1 - \sum_{x \in X} \mathbf{P}[x \notin \cup_{S \in \mathcal{C}} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2:** The **expected approximation ratio** ✓
 - By previous lemma, the **expected cost** of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S)$.
 - Linearity $\Rightarrow \mathbf{E}[c(\mathcal{C})] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot \bar{y}(S) \leq 2 \ln(n) \cdot c(\mathcal{C}^*)$ □

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

By Markov's inequality, $\mathbf{P}[c(\mathcal{C}) \leq 4 \ln(n) \cdot c(\mathcal{C}^*)] \geq 1/2$.

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

By Markov's inequality, $\mathbf{P}[c(\mathcal{C}) \leq 4 \ln(n) \cdot c(\mathcal{C}^*)] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is **valid** and within a factor of $4 \ln(n)$ of the optimum.

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

By Markov's inequality, $\mathbf{P}[c(\mathcal{C}) \leq 4 \ln(n) \cdot c(\mathcal{C}^*)] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is **valid** and within a factor of $4 \ln(n)$ of the optimum.

probability could be further increased by repeating

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

By Markov's inequality, $\mathbf{P}[c(\mathcal{C}) \leq 4 \ln(n) \cdot c(\mathcal{C}^*)] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is **valid** and within a factor of $4 \ln(n)$ of the optimum.

probability could be further increased by repeating

Typical Approach for Designing Approximation Algorithms based on LPs

Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set \mathcal{C} is a valid cover of X .
- The expected approximation ratio is $2 \ln(n)$.

By Markov's inequality, $\mathbf{P}[c(\mathcal{C}) \leq 4 \ln(n) \cdot c(\mathcal{C}^*)] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is **valid** and within a factor of $4 \ln(n)$ of the optimum.

probability could be further increased by repeating

Typical Approach for Designing Approximation Algorithms based on LPs

[Exercise Question (9/10).10] gives a different perspective on the amplification procedure through **non-linear randomised rounding**.

Weighted Set Cover

MAX-CNF

MAX-CNF

Recall:

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

MAX-CNF Satisfiability (MAX-SAT)

Recall:

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

MAX-CNF Satisfiability (MAX-SAT)

- **Given:** CNF formula, e.g.: $(x_1 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4 \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

MAX-CNF

Recall:

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

MAX-CNF Satisfiability (MAX-SAT)

- **Given:** CNF formula, e.g.: $(x_1 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4 \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Why study this generalised problem?

Recall:

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

MAX-CNF Satisfiability (MAX-SAT)

- **Given:** CNF formula, e.g.: $(x_1 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4 \vee \overline{x_5}) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Why study this generalised problem?

- Allowing arbitrary clause lengths makes the problem more interesting (we will see that simply guessing is not the best!)
- a nice concluding example where we can practice previously learned approaches

Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

— Analysis —

For any clause i which has length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_{\ell}.$$

In particular, the guessing algorithm is a **randomised 2-approximation**.

Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

— Analysis —

For any clause i which has length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_{\ell}.$$

In particular, the guessing algorithm is a randomised 2-approximation.

Proof:

Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause i which has length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_{\ell}.$$

In particular, the guessing algorithm is a **randomised 2-approximation**.

Proof:

- First statement as in the proof of Theorem 35.6. For clause i not to be satisfied, all ℓ occurring variables must be set to a specific value.

Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause i which has length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_{\ell}.$$

In particular, the guessing algorithm is a **randomised 2-approximation**.

Proof:

- First statement as in the proof of Theorem 35.6. For clause i not to be satisfied, all ℓ occurring variables must be set to a specific value.
- As before, let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i] \geq \sum_{i=1}^m \frac{1}{2} = \frac{1}{2} \cdot m. \quad \square$$

Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!

Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!

0-1 Integer Program

$$\begin{array}{ll}\text{maximize} & \sum_{i=1}^m z_i \\ \text{subject to} & \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \quad \text{for each } i = 1, 2, \dots, m \\ & z_i \in \{0, 1\} \quad \text{for each } i = 1, 2, \dots, m \\ & y_j \in \{0, 1\} \quad \text{for each } j = 1, 2, \dots, n\end{array}$$

Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!

0-1 Integer Program

$$\text{maximize } \sum_{i=1}^m z_i$$

$$\text{subject to } \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \quad \text{for each } i = 1, 2, \dots, m$$

C_i^+ is the index set of the un-negated variables of clause i .

$$z_i \in \{0, 1\} \quad \text{for each } i = 1, 2, \dots, m$$

$$y_j \in \{0, 1\} \quad \text{for each } j = 1, 2, \dots, n$$

Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!

0-1 Integer Program

$$\text{maximize} \quad \sum_{i=1}^m z_i$$

$$\text{subject to} \quad \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \quad \text{for each } i = 1, 2, \dots, m$$

C_i^+ is the index set of the un-negated variables of clause i .

These **auxiliary** variables are used to reflect whether a clause is satisfied or not

$$z_i \in \{0, 1\} \quad \text{for each } i = 1, 2, \dots, m$$

$$y_j \in \{0, 1\} \quad \text{for each } j = 1, 2, \dots, n$$

Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!

0-1 Integer Program

$$\text{maximize} \quad \sum_{i=1}^m z_i$$

$$\text{subject to} \quad \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \quad \text{for each } i = 1, 2, \dots, m$$

C_i^+ is the index set of the un-negated variables of clause i .

These **auxiliary** variables are used to reflect whether a clause is satisfied or not

$$z_i \in \{0, 1\} \quad \text{for each } i = 1, 2, \dots, m$$

$$y_j \in \{0, 1\} \quad \text{for each } j = 1, 2, \dots, n$$

- In the **corresponding LP** each $\in \{0, 1\}$ is replaced by $\in [0, 1]$
- Let (\bar{y}, \bar{z}) be the optimal solution of the LP
- Obtain an integer solution y through randomised rounding of \bar{y}

Analysis of Randomised Rounding

— Lemma —

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] =$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \mathbf{P}[x_j \text{ is false}]$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \mathbf{P}[x_j \text{ is false}] = 1 - \prod_{j=1}^{\ell} (1 - \bar{y}_j)$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \mathbf{P}[x_j \text{ is false}] = 1 - \prod_{j=1}^{\ell} (1 - \bar{y}_j)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \dots + a_k}{k} \geq \sqrt[k]{a_1 \times \dots \times a_k}.$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \mathbf{P}[x_j \text{ is false}] = 1 - \prod_{j=1}^{\ell} (1 - \bar{y}_j)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \dots + a_k}{k} \geq \sqrt[k]{a_1 \times \dots \times a_k}.$$

$$\geq 1 - \left(\frac{\sum_{j=1}^{\ell} (1 - \bar{y}_j)}{\ell} \right)^\ell$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \mathbf{P}[x_j \text{ is false}] = 1 - \prod_{j=1}^{\ell} (1 - \bar{y}_j)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \dots + a_k}{k} \geq \sqrt[k]{a_1 \times \dots \times a_k}.$$

$$\begin{aligned} &\geq 1 - \left(\frac{\sum_{j=1}^{\ell} (1 - \bar{y}_j)}{\ell} \right)^\ell \\ &= 1 - \left(1 - \frac{\sum_{j=1}^{\ell} \bar{y}_j}{\ell} \right)^\ell \end{aligned}$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause i appear non-negated (otherwise replace every occurrence of x_j by \bar{x}_j in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \dots \vee x_\ell)$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \mathbf{P}[x_j \text{ is false}] = 1 - \prod_{j=1}^{\ell} (1 - \bar{y}_j)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \dots + a_k}{k} \geq \sqrt[k]{a_1 \times \dots \times a_k}.$$

$$\begin{aligned} &\geq 1 - \left(\frac{\sum_{j=1}^{\ell} (1 - \bar{y}_j)}{\ell} \right)^\ell \\ &= 1 - \left(1 - \frac{\sum_{j=1}^{\ell} \bar{y}_j}{\ell} \right)^\ell \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell} \right)^\ell. \end{aligned}$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (2/2):

- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (2/2):

- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$.

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (2/2):

- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a **concave** function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

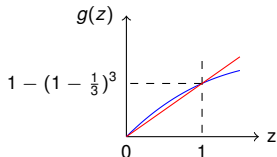
$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (2/2):

- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a **concave** function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.



Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

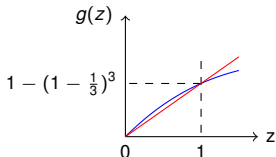
Proof of Lemma (2/2):

- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a **concave** function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.

$$\Rightarrow \quad g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$



Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (2/2):

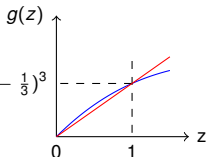
- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a **concave** function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.

$$\Rightarrow \quad g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$

- Therefore, $\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \beta_\ell \cdot \bar{z}_i$.



Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Proof of Lemma (2/2):

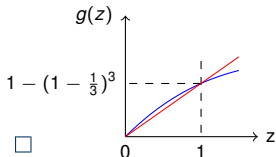
- So far we have shown:

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{\bar{z}_i}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a **concave** function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.

$$\Rightarrow \quad g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$

- Therefore, $\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \beta_\ell \cdot \bar{z}_i$. \square



Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \dots, m$, let ℓ_i be the corresponding length.

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \dots, m$, let ℓ_i be the corresponding length.
- Then the **expected number** of satisfied clauses is:

$$\mathbf{E}[Y] = \sum_{i=1}^m \mathbf{E}[Y_i] \geq$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \dots, m$, let ℓ_i be the corresponding length.
- Then the **expected number** of satisfied clauses is:

$$\mathbf{E}[Y] = \sum_{i=1}^m \mathbf{E}[Y_i] \geq \sum_{i=1}^m \left(1 - \left(1 - \frac{1}{\ell_i}\right)^{\ell_i}\right) \cdot \bar{z}_i$$

By Lemma

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \dots, m$, let ℓ_i be the corresponding length.
- Then the **expected number** of satisfied clauses is:

$$\mathbf{E}[Y] = \sum_{i=1}^m \mathbf{E}[Y_i] \geq \sum_{i=1}^m \left(1 - \left(1 - \frac{1}{\ell_i}\right)^{\ell_i}\right) \cdot \bar{z}_i \geq \sum_{i=1}^m \left(1 - \frac{1}{e}\right) \cdot \bar{z}_i$$

By Lemma

Since $(1 - 1/x)^x \leq 1/e$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ ,

$$\mathbf{P}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot \bar{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \dots, m$, let ℓ_i be the corresponding length.
- Then the **expected number** of satisfied clauses is:

$$\mathbf{E}[Y] = \sum_{i=1}^m \mathbf{E}[Y_i] \geq \sum_{i=1}^m \left(1 - \left(1 - \frac{1}{\ell_i}\right)^{\ell_i}\right) \cdot \bar{z}_i \geq \sum_{i=1}^m \left(1 - \frac{1}{e}\right) \cdot \bar{z}_i \geq \left(1 - \frac{1}{e}\right) \cdot \text{OPT}$$

By Lemma

Since $(1 - 1/x)^x \leq 1/e$

LP solution at least as good as optimum

Summary

- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Approach 3: Hybrid Algorithm

Summary

- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Idea: Consider a hybrid algorithm which interpolates between the two approaches

Approach 3: Hybrid Algorithm

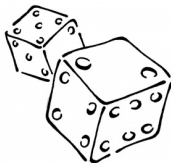
Summary

- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Idea: Consider a hybrid algorithm which interpolates between the two approaches

HYBRID-MAX-CNF(φ, n, m)

- 1: Let $b \in \{0, 1\}$ be the flip of a fair coin
- 2: **If** $b = 0$ **then** perform random guessing
- 3: **If** $b = 1$ **then** perform randomised rounding
- 4: **return** the computed solution



Approach 3: Hybrid Algorithm

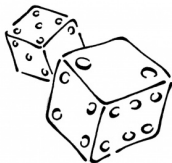
Summary

- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Idea: Consider a hybrid algorithm which interpolates between the two approaches

HYBRID-MAX-CNF(φ, n, m)

- 1: Let $b \in \{0, 1\}$ be the flip of a fair coin
- 2: **If** $b = 0$ **then** perform random guessing
- 3: **If** $b = 1$ **then** perform randomised rounding
- 4: **return** the computed solution



Algorithm sets each variable x_i to TRUE with prob. $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \bar{y}_i$.
Note, however, that variables are **not** independently assigned!

Analysis of Hybrid Algorithm

— Theorem —

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Analysis of Hybrid Algorithm

— Theorem —

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

Analysis of Hybrid Algorithm

— Theorem —

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$

Analysis of Hybrid Algorithm

— Theorem —

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :

Analysis of Hybrid Algorithm

— Theorem —

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - **Algorithm 1** satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.

Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.

Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - **Algorithm 1** satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - **Algorithm 2** satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - **HYBRID-MAX-CNF(φ, n, m)** satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.

Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - **Algorithm 1** satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - **Algorithm 2** satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - **HYBRID-MAX-CNF(φ, n, m)** satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$,

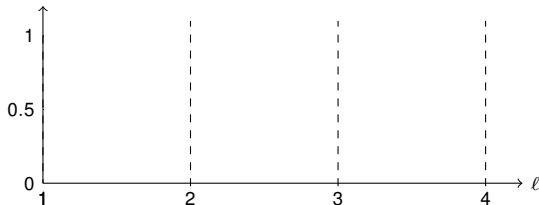
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - HYBRID-MAX-CNF(φ, n, m) satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$, and for $\ell \geq 3$, $\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4$ (see figure)



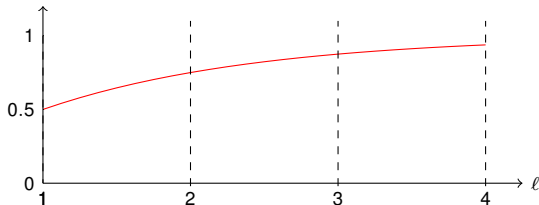
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - HYBRID-MAX-CNF(φ, n, m) satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$, and for $\ell \geq 3$, $\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4$ (see figure)



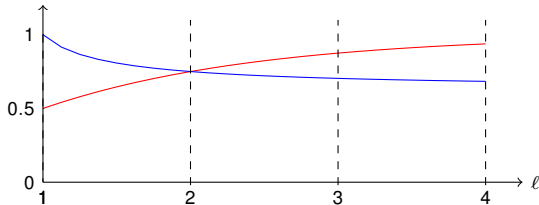
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - HYBRID-MAX-CNF(φ, n, m) satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$, and for $\ell \geq 3$, $\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4$ (see figure)



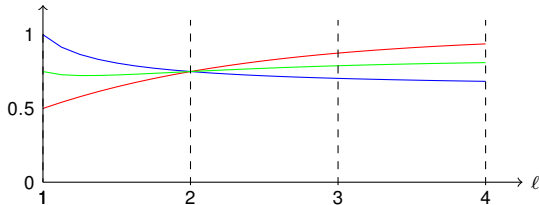
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - HYBRID-MAX-CNF(φ, n, m) satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$, and for $\ell \geq 3$, $\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4$ (see figure)



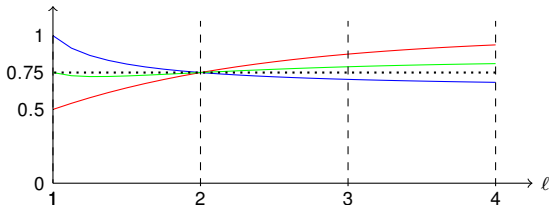
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1** satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2** satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - HYBRID-MAX-CNF(φ, n, m)** satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$, and for $\ell \geq 3$, $\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4$ (see figure)



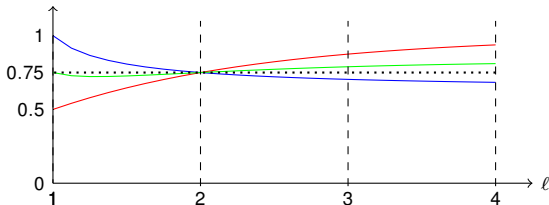
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised $4/3$ -approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \bar{z}_i$
- For any clause i of length ℓ :
 - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i$.
 - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \bar{z}_i$.
 - HYBRID-MAX-CNF(φ, n, m) satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i$.
- Note $\frac{\alpha_\ell + \beta_\ell}{2} = 3/4$ for $\ell \in \{1, 2\}$, and for $\ell \geq 3$, $\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4$ (see figure)
- \Rightarrow HYBRID-MAX-CNF(φ, n, m) satisfies it with prob. at least $3/4 \cdot \bar{z}_i$ \square



Summary

- Since $\alpha_2 = \beta_2 = 3/4$, we cannot achieve a better approximation ratio than $4/3$ by combining Algorithm 1 & 2 in a different way
- The $4/3$ -approximation algorithm can be easily derandomised
 - Idea: use the conditional expectation trick for both Algorithm 1 & 2 and output the better solution
- The $4/3$ -approximation algorithm applies unchanged to a weighted version of MAX-CNF, where each clause has a non-negative weight
- Even MAX-2-CNF (every clause has length 2) is NP-hard!

Randomised Algorithms

Lecture 11: Spectral Graph Theory

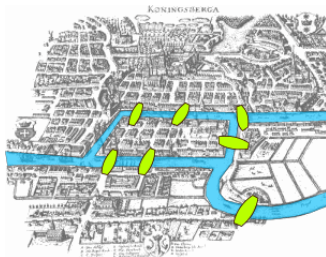
Thomas Sauerwald (tms41@cam.ac.uk)

Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem

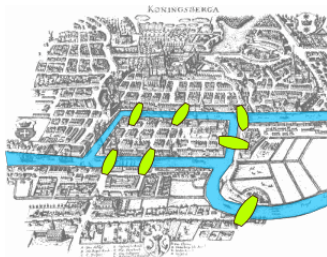
Origin of Graph Theory



Source: Wikipedia

Seven Bridges at Königsberg 1737

Origin of Graph Theory



Source: Wikipedia

Seven Bridges at Königsberg 1737

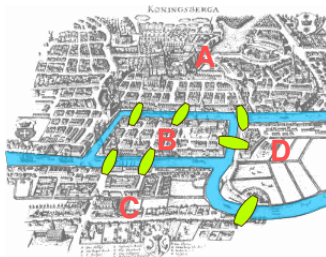


Source: Wikipedia

Leonhard Euler (1707-1783)

Is there a tour which crosses
each bridge **exactly once**?

Origin of Graph Theory



Source: Wikipedia

Seven Bridges at Königsberg 1737

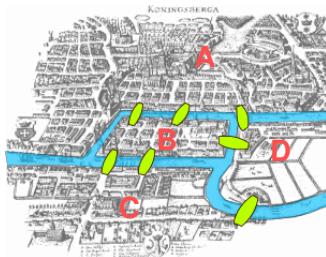


Source: Wikipedia

Leonhard Euler (1707-1783)

Is there a tour which crosses
each bridge **exactly once**?

Origin of Graph Theory



Source: Wikipedia



Source: Wikipedia

Seven Bridges at Königsberg 1737

Leonhard Euler (1707-1783)

A

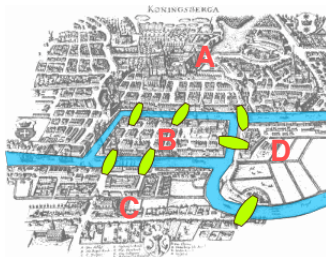
B

C

D

Is there a tour which crosses
each bridge **exactly once**?

Origin of Graph Theory



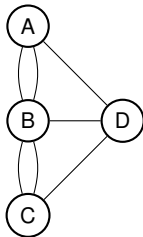
Source: Wikipedia



Source: Wikipedia

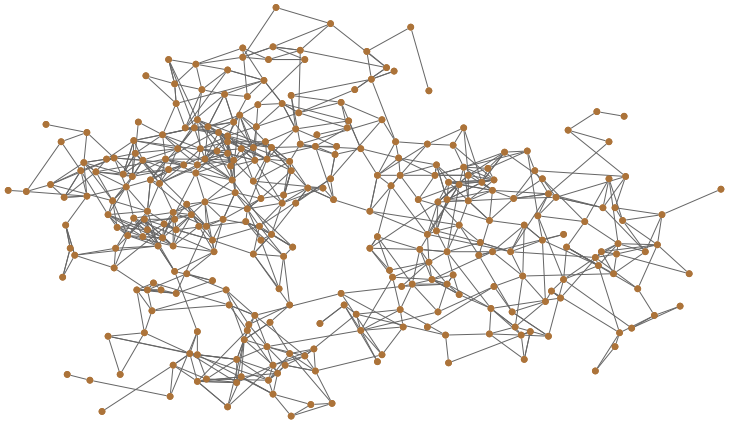
Seven Bridges at Königsberg 1737

Leonhard Euler (1707-1783)

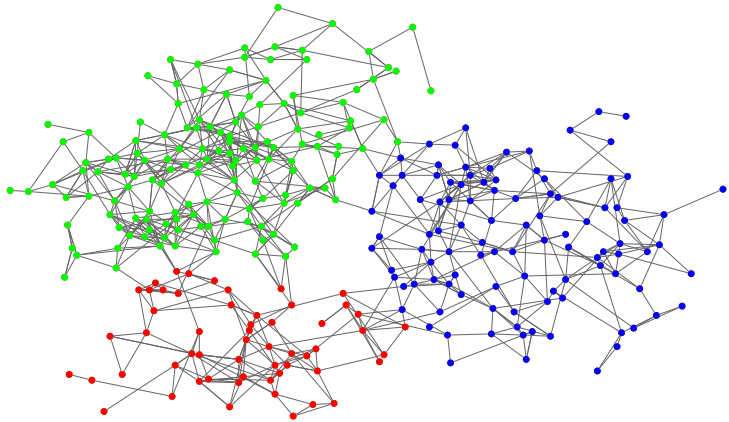


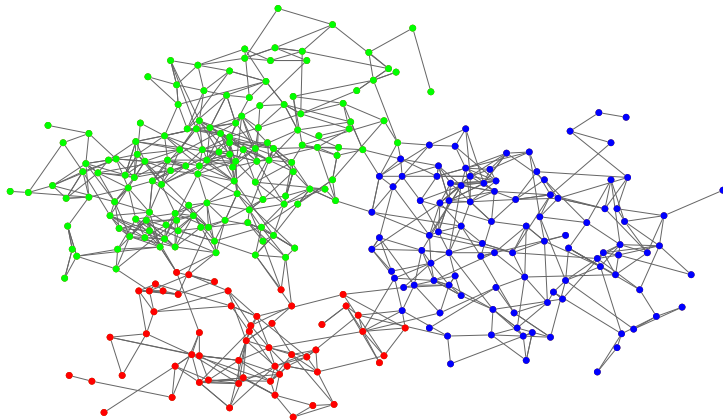
Is there a tour which crosses each bridge **exactly once**?

Graphs Nowadays: Clustering



Graphs Nowadays: Clustering





Goal: Use spectrum of graphs (unstructured data) to extract clustering (communities) or other structural information.

- Applications of Graph Clustering
 - Community detection
 - Group webpages according to their topics
 - Find proteins performing the same function within a cell
 - Image segmentation
 - Identify bottlenecks in a network
 - ...

- Applications of Graph Clustering
 - Community detection
 - Group webpages according to their topics
 - Find proteins performing the same function within a cell
 - Image segmentation
 - Identify bottlenecks in a network
 - ...
- Unsupervised learning method
(there is no ground truth (usually), and we cannot learn from mistakes!)

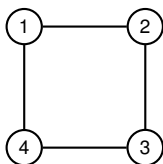
- Applications of Graph Clustering
 - Community detection
 - Group webpages according to their topics
 - Find proteins performing the same function within a cell
 - Image segmentation
 - Identify bottlenecks in a network
 - ...
- Unsupervised learning method
(there is no ground truth (usually), and we cannot learn from mistakes!)
- Different formalisations for different applications

- Applications of Graph Clustering
 - Community detection
 - Group webpages according to their topics
 - Find proteins performing the same function within a cell
 - Image segmentation
 - Identify bottlenecks in a network
 - ...
- Unsupervised learning method
(there is no ground truth (usually), and we cannot learn from mistakes!)
- Different formalisations for different applications
 - Geometric Clustering: partition points in a Euclidean space
 - k -means, k -medians, k -centres, etc.

- Applications of Graph Clustering
 - Community detection
 - Group webpages according to their topics
 - Find proteins performing the same function within a cell
 - Image segmentation
 - Identify bottlenecks in a network
 - ...
- Unsupervised learning method
(there is no ground truth (usually), and we cannot learn from mistakes!)
- Different formalisations for different applications
 - Geometric Clustering: partition points in a Euclidean space
 - k -means, k -medians, k -centres, etc.
 - Graph Clustering: partition vertices in a graph
 - modularity, conductance, min-cut, etc.

- Applications of Graph Clustering
 - Community detection
 - Group webpages according to their topics
 - Find proteins performing the same function within a cell
 - Image segmentation
 - Identify bottlenecks in a network
 - ...
- Unsupervised learning method
(there is no ground truth (usually), and we cannot learn from mistakes!)
- Different formalisations for different applications
 - Geometric Clustering: partition points in a Euclidean space
 - k -means, k -medians, k -centres, etc.
 - Graph Clustering: partition vertices in a graph
 - modularity, conductance, min-cut, etc.

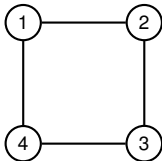
Graphs



Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Graphs



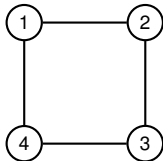
- Connectivity
- Bipartiteness
- Number of triangles
- Graph Clustering
- Graph isomorphism
- Maximum Flow
- Shortest Paths
- ...

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

- Eigenvalues
- Eigenvectors
- Inverse
- Determinant
- Matrix-powers
- ...

Graphs



- Connectivity
- Bipartiteness
- Number of triangles
- Graph Clustering
- Graph isomorphism
- Maximum Flow
- Shortest Paths
- ...

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

- Eigenvalues
- Eigenvectors
- Inverse
- Determinant
- Matrix-powers
- ...

Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem

Adjacency Matrix

Adjacency matrix

Let $G = (V, E)$ be an **undirected** graph. The **adjacency matrix** of G is the n by n matrix \mathbf{A} defined as

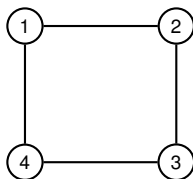
$$\mathbf{A}_{u,v} = \begin{cases} 1 & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

Adjacency Matrix

Adjacency matrix

Let $G = (V, E)$ be an undirected graph. The adjacency matrix of G is the n by n matrix \mathbf{A} defined as

$$\mathbf{A}_{u,v} = \begin{cases} 1 & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$



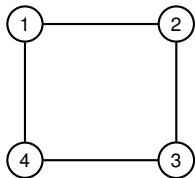
$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Adjacency Matrix

Adjacency matrix

Let $G = (V, E)$ be an **undirected** graph. The **adjacency matrix** of G is the n by n matrix \mathbf{A} defined as

$$\mathbf{A}_{u,v} = \begin{cases} 1 & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Properties of \mathbf{A} :

- The sum of elements in each row/column i equals the **degree** of the corresponding vertex i , $\deg(i)$
- Since G is **undirected**, \mathbf{A} is **symmetric**

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

Graph Spectrum

Let \mathbf{A} be the adjacency matrix of a **d -regular** graph G with n vertices.

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

An **undirected** graph G is **d -regular** if every degree is d , i.e., every vertex has exactly d connections.

Graph Spectrum

Let \mathbf{A} be the adjacency matrix of a **d -regular** graph G with n vertices.

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

An **undirected** graph G is **d -regular** if every degree is d , i.e., every vertex has exactly d connections.

Graph Spectrum

Let \mathbf{A} be the adjacency matrix of a **d -regular** graph G with n vertices. Then, \mathbf{A} has n **real eigenvalues** $\lambda_1 \leq \dots \leq \lambda_n$ and n corresponding **orthonormal eigenvectors** f_1, \dots, f_n .

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

An **undirected** graph G is **d -regular** if every degree is d , i.e., every vertex has exactly d connections.

Graph Spectrum

Let \mathbf{A} be the adjacency matrix of a **d -regular** graph G with n vertices. Then, \mathbf{A} has n **real eigenvalues** $\lambda_1 \leq \dots \leq \lambda_n$ and n corresponding **orthonormal eigenvectors** f_1, \dots, f_n .

= **orthogonal** and **normalised**

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

An **undirected** graph G is **d -regular** if every degree is d , i.e., every vertex has exactly d connections.

Graph Spectrum

Let \mathbf{A} be the adjacency matrix of a **d -regular** graph G with n vertices. Then, \mathbf{A} has n **real eigenvalues** $\lambda_1 \leq \dots \leq \lambda_n$ and n corresponding **orthonormal eigenvectors** f_1, \dots, f_n . These eigenvalues associated with their **multiplicities** constitute the **spectrum** of G .

Eigenvalues and Graph Spectrum of A

Eigenvalues and Eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

An **undirected** graph G is **d -regular** if every degree is d , i.e., every vertex has exactly d connections.

Graph Spectrum

Let \mathbf{A} be the adjacency matrix of a **d -regular** graph G with n vertices. Then, \mathbf{A} has n **real eigenvalues** $\lambda_1 \leq \dots \leq \lambda_n$ and n corresponding **orthonormal eigenvectors** f_1, \dots, f_n . These eigenvalues associated with their **multiplicities** constitute the **spectrum** of G .

Remark: For **symmetric** matrices we have **algebraic multiplicity** = **geometric multiplicity** (otherwise \geq)

Example 1

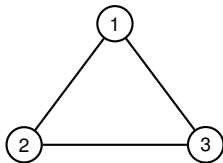


Question: What are the Eigenvalues and Eigenvectors?

Example 1



Question: What are the Eigenvalues and Eigenvectors?



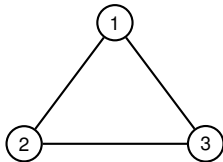
$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Example 1



Bonus: Can you find a short-cut to $\det(\mathbf{A} - \lambda \cdot \mathbf{I})$?

Question: What are the Eigenvalues and Eigenvectors?



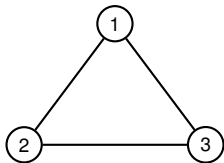
$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Example 1



Bonus: Can you find a short-cut to $\det(\mathbf{A} - \lambda \cdot \mathbf{I})$?

Question: What are the Eigenvalues and Eigenvectors?



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Solution:

- The three eigenvalues are $\lambda_1 = \lambda_2 = -1, \lambda_3 = 2$.
- The three eigenvectors are (for example):

$$f_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad f_2 = \begin{pmatrix} -\frac{1}{2} \\ 1 \\ -\frac{1}{2} \end{pmatrix}, \quad f_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Laplacian Matrix

Laplacian Matrix

Let $G = (V, E)$ be a d -regular undirected graph. The (normalised) Laplacian matrix of G is the n by n matrix \mathbf{L} defined as

$$\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A},$$

where \mathbf{I} is the $n \times n$ identity matrix.

Laplacian Matrix

Laplacian Matrix

Let $G = (V, E)$ be a d -regular undirected graph. The (normalised) Laplacian matrix of G is the n by n matrix \mathbf{L} defined as

$$\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A},$$

where \mathbf{I} is the $n \times n$ identity matrix.



Question: What is the matrix $\frac{1}{d} \cdot \mathbf{A}$?

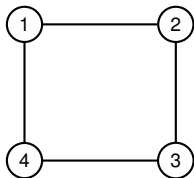
Laplacian Matrix

Laplacian Matrix

Let $G = (V, E)$ be a d -regular undirected graph. The (normalised) Laplacian matrix of G is the n by n matrix \mathbf{L} defined as

$$\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A},$$

where \mathbf{I} is the $n \times n$ identity matrix.



$$\mathbf{L} = \begin{pmatrix} 1 & -1/2 & 0 & -1/2 \\ -1/2 & 1 & -1/2 & 0 \\ 0 & -1/2 & 1 & -1/2 \\ -1/2 & 0 & -1/2 & 1 \end{pmatrix}$$

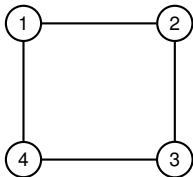
Laplacian Matrix

Laplacian Matrix

Let $G = (V, E)$ be a d -regular undirected graph. The (normalised) Laplacian matrix of G is the n by n matrix \mathbf{L} defined as

$$\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A},$$

where \mathbf{I} is the $n \times n$ identity matrix.



$$\mathbf{L} = \begin{pmatrix} 1 & -1/2 & 0 & -1/2 \\ -1/2 & 1 & -1/2 & 0 \\ 0 & -1/2 & 1 & -1/2 \\ -1/2 & 0 & -1/2 & 1 \end{pmatrix}$$

Properties of \mathbf{L} :

- The sum of elements in each row/column equals zero

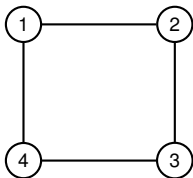
Laplacian Matrix

Laplacian Matrix

Let $G = (V, E)$ be a d -regular undirected graph. The (normalised) Laplacian matrix of G is the n by n matrix \mathbf{L} defined as

$$\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A},$$

where \mathbf{I} is the $n \times n$ identity matrix.



$$\mathbf{L} = \begin{pmatrix} 1 & -1/2 & 0 & -1/2 \\ -1/2 & 1 & -1/2 & 0 \\ 0 & -1/2 & 1 & -1/2 \\ -1/2 & 0 & -1/2 & 1 \end{pmatrix}$$

Properties of \mathbf{L} :

- The sum of elements in each row/column equals zero
- \mathbf{L} is symmetric

Relating Spectrum of Adjacency Matrix and Laplacian Matrix

Correspondence between Adjacency and Laplacian Matrix

A and **L** have the same set of eigenvectors.



Exercise: Prove this correspondence. Hint: Use that $\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A}$.

[Exercise 11/12.1]

Eigenvalues and Graph Spectrum of L

Eigenvalues and eigenvectors

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue** of \mathbf{M} if and only if there exists $x \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an **eigenvector** of \mathbf{M} corresponding to the eigenvalue λ .

Graph Spectrum

Let \mathbf{L} be the **Laplacian matrix** of a d -regular graph G with n vertices. Then, \mathbf{L} has n real **eigenvalues** $\lambda_1 \leq \dots \leq \lambda_n$ and n corresponding **orthonormal eigenvectors** f_1, \dots, f_n . These eigenvalues associated with their multiplicities constitute the **spectrum** of G .

— Lemma —

Let \mathbf{L} be the Laplacian matrix of an undirected, regular graph $G = (V, E)$ with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$.

1. $\lambda_1 = 0$ with eigenvector $\mathbf{1}$
2. the multiplicity of the eigenvalue 0 is equal to the number of connected components in G

— Lemma —

Let \mathbf{L} be the Laplacian matrix of an undirected, regular graph $G = (V, E)$ with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$.

1. $\lambda_1 = 0$ with eigenvector $\mathbf{1}$
2. the multiplicity of the eigenvalue 0 is equal to the number of connected components in G
3. $\lambda_n \leq 2$
4. $\lambda_n = 2$ iff there exists a bipartite connected component.

Lemma

Let \mathbf{L} be the Laplacian matrix of an undirected, regular graph $G = (V, E)$ with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$.

1. $\lambda_1 = 0$ with eigenvector $\mathbf{1}$
2. the multiplicity of the eigenvalue 0 is equal to the number of connected components in G
3. $\lambda_n \leq 2$
4. $\lambda_n = 2$ iff there exists a bipartite connected component.

The proof of these properties is based on a powerful characterisation of eigenvalues/vectors!

A Min-Max Characterisation of Eigenvalues and Eigenvectors

Courant-Fischer Min-Max Formula (non-examinable)

Let \mathbf{M} be an n by n symmetric matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Then,

$$\lambda_k = \min_{S: \dim(S)=k} \max_{x \in S, x \neq 0} \frac{x^T \mathbf{M} x}{x^T x},$$

where S is a subspace of \mathbb{R}^n . The eigenvectors corresponding to $\lambda_1, \dots, \lambda_k$ minimise such expression.

A Min-Max Characterisation of Eigenvalues and Eigenvectors

Courant-Fischer Min-Max Formula (non-examinable)

Let \mathbf{M} be an n by n symmetric matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Then,

$$\lambda_k = \min_{S: \dim(S)=k} \max_{x \in S, x \neq 0} \frac{x^T \mathbf{M} x}{x^T x},$$

where S is a subspace of \mathbb{R}^n . The eigenvectors corresponding to $\lambda_1, \dots, \lambda_k$ minimise such expression.

$$\lambda_1 = \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by an eigenvector f_1 for λ_1

A Min-Max Characterisation of Eigenvalues and Eigenvectors

Courant-Fischer Min-Max Formula (non-examinable)

Let \mathbf{M} be an n by n symmetric matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Then,

$$\lambda_k = \min_{S: \dim(S)=k} \max_{x \in S, x \neq 0} \frac{x^T \mathbf{M} x}{x^T x},$$

where S is a subspace of \mathbb{R}^n . The eigenvectors corresponding to $\lambda_1, \dots, \lambda_k$ minimise such expression.

$$\lambda_1 = \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by an eigenvector f_1 for λ_1

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp f_1}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by f_2

A Min-Max Characterisation of Eigenvalues and Eigenvectors

Courant-Fischer Min-Max Formula (non-examinable)

Let \mathbf{M} be an n by n symmetric matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Then,

$$\lambda_k = \min_{S: \dim(S)=k} \max_{x \in S, x \neq 0} \frac{x^T \mathbf{M} x}{x^T x},$$

where S is a subspace of \mathbb{R}^n . The eigenvectors corresponding to $\lambda_1, \dots, \lambda_k$ minimise such expression.

$$\lambda_1 = \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by an eigenvector f_1 for λ_1

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp f_1}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by f_2

A Min-Max Characterisation of Eigenvalues and Eigenvectors

Courant-Fischer Min-Max Formula (non-examinable)

Let \mathbf{M} be an n by n symmetric matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Then,

$$\lambda_k = \min_{S: \dim(S)=k} \max_{x \in S, x \neq 0} \frac{x^T \mathbf{M} x}{x^T x},$$

where S is a subspace of \mathbb{R}^n . The eigenvectors corresponding to $\lambda_1, \dots, \lambda_k$ minimise such expression.

$$\lambda_1 = \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by an eigenvector f_1 for λ_1

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp f_1}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by f_2

Quadratic Forms of the Laplacian

— Lemma —

Let \mathbf{L} be the Laplacian matrix of a d -regular graph $G = (V, E)$ with n vertices. For any $x \in \mathbb{R}^n$,

$$x^T \mathbf{L} x = \sum_{\{u,v\} \in E} \frac{(x_u - x_v)^2}{d}.$$

Quadratic Forms of the Laplacian

Lemma

Let \mathbf{L} be the Laplacian matrix of a d -regular graph $G = (V, E)$ with n vertices. For any $x \in \mathbb{R}^n$,

$$x^T \mathbf{L} x = \sum_{\{u,v\} \in E} \frac{(x_u - x_v)^2}{d}.$$

Proof:

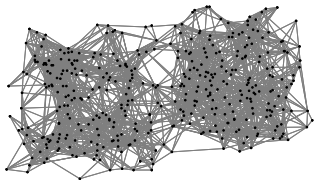
$$\begin{aligned} x^T \mathbf{L} x &= x^T \left(\mathbf{I} - \frac{1}{d} \mathbf{A} \right) x = x^T x - \frac{1}{d} x^T \mathbf{A} x \\ &= \sum_{u \in V} x_u^2 - \frac{2}{d} \sum_{\{u,v\} \in E} x_u x_v \\ &= \frac{1}{d} \sum_{\{u,v\} \in E} (x_u^2 + x_v^2 - 2x_u x_v) \\ &= \sum_{\{u,v\} \in E} \frac{(x_u - x_v)^2}{d}. \end{aligned}$$

Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?

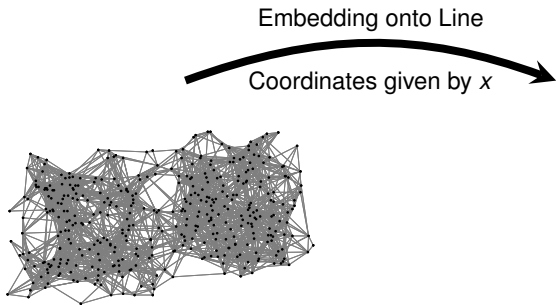
Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?



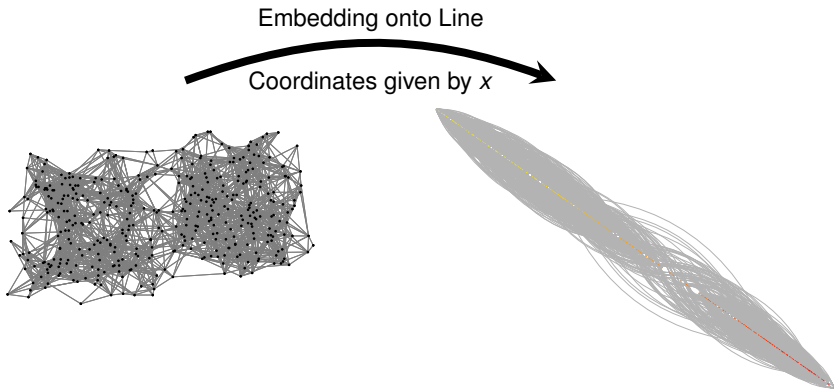
Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?



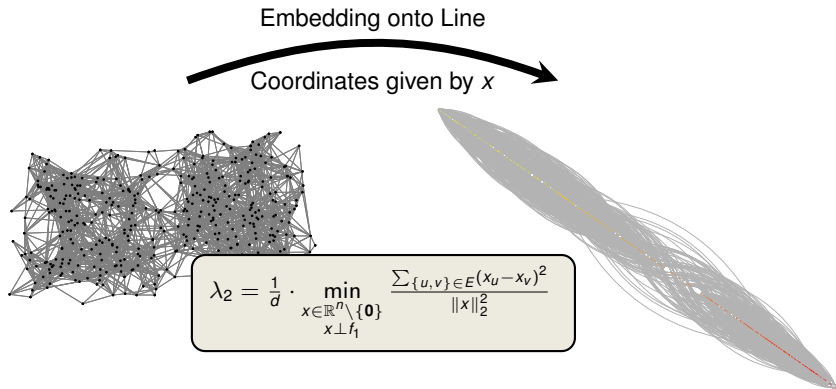
Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?



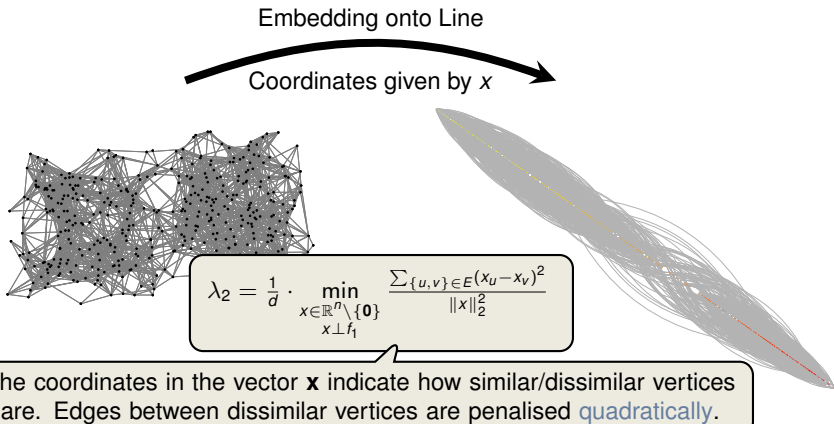
Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?



Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?



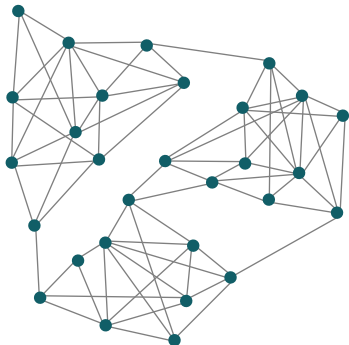
Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem

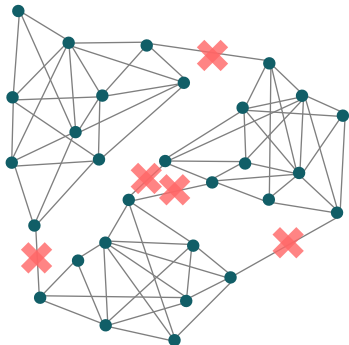
A Simplified Clustering Problem

*Partition the graph into **connected components** so that any pair of vertices in the same component is connected, but vertices in different components are not.*



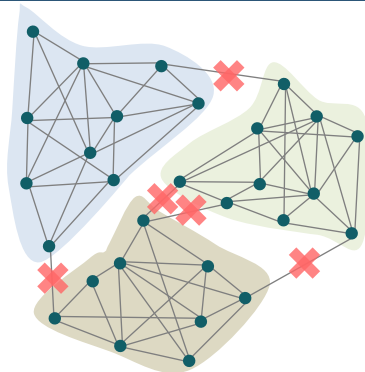
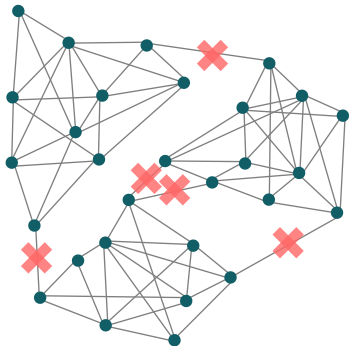
A Simplified Clustering Problem

Partition the graph into **connected components** so that any pair of vertices in the same component is connected, but vertices in different components are not.



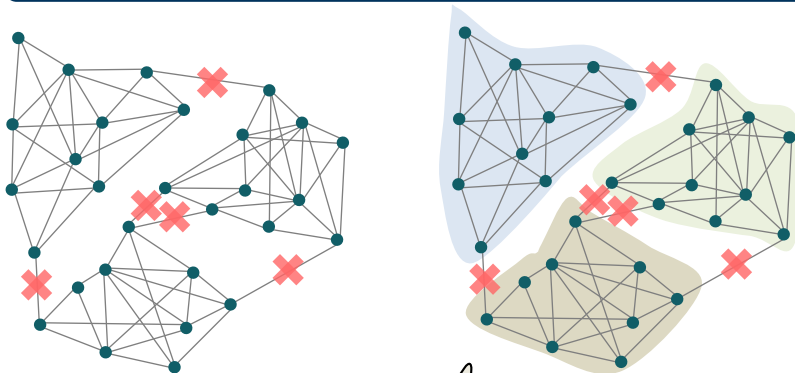
A Simplified Clustering Problem

Partition the graph into **connected components** so that any pair of vertices in the same component is connected, but vertices in different components are not.



A Simplified Clustering Problem

Partition the graph into **connected components** so that any pair of vertices in the same component is connected, but vertices in different components are not.



We could obviously solve this easily using DFS/BFS, but let's see how we can tackle this using the **spectrum of L** !

Example 2

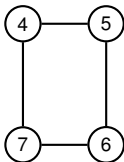
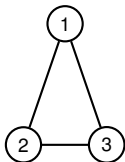


Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?

Example 2



Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?

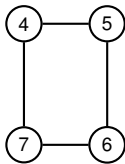
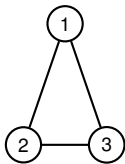


$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Example 2



Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?



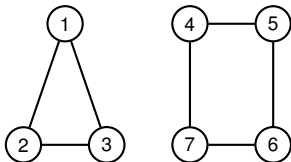
$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Example 2



Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Solution:

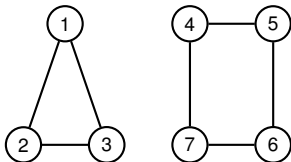
- Two smallest eigenvalues are $\lambda_1 = \lambda_2 = 0$.
- The corresponding two eigenvectors are:

$$f_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad f_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Example 2



Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Solution:

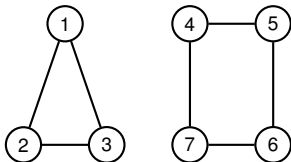
- Two smallest eigenvalues are $\lambda_1 = \lambda_2 = 0$.
- The corresponding two eigenvectors are:

$$\mathbf{f}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{f}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad (\text{or } \mathbf{f}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{f}_2 = \begin{pmatrix} -1/3 \\ -1/3 \\ -1/3 \\ 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix})$$

Example 2



Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Solution:

- Two smallest eigenvalues are $\lambda_1 = \lambda_2 = 0$.
- The corresponding two eigenvectors are:

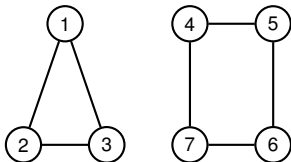
$$f_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad f_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Thus we can easily solve the simplified clustering problem by computing the eigenvectors with eigenvalue 0

Example 2



Question: What are the Eigenvectors with Eigenvalue 0 of \mathbf{L} ?



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Solution:

- Two smallest eigenvalues are $\lambda_1 = \lambda_2 = 0$.
- The corresponding two eigenvectors are:

$$\mathbf{f}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{f}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Thus we can easily solve the simplified clustering problem by computing the eigenvectors with eigenvalue 0

Next Lecture: A fine-grained approach works even if the clusters are **sparsely** connected!

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (" \implies " $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (" \implies " $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \implies ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} ’s are orthogonal

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \Rightarrow ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \Rightarrow \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are **orthogonal**
- $\chi_{C_i}^T \mathbf{L} \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \Rightarrow \lambda_1 = \dots = \lambda_k = 0$

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \Rightarrow ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \Rightarrow \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are **orthogonal**
- $\chi_{C_i}^T L \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \Rightarrow \lambda_1 = \dots = \lambda_k = 0$

2. (“ \Leftarrow ” $cc(G) \geq \text{mult}(0)$). We will show:

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \implies ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are orthogonal
- $\chi_{C_i}^T L \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \implies \lambda_1 = \dots = \lambda_k = 0$

2. (“ \impliedby ” $cc(G) \geq \text{mult}(0)$). We will show:

$\lambda_1 = \dots = \lambda_k = 0 \implies G$ has at least k connected comp. C_1, \dots, C_k

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \implies ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are orthogonal
- $\chi_{C_i}^T \mathbf{L} \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \implies \lambda_1 = \dots = \lambda_k = 0$

2. (“ \impliedby ” $cc(G) \geq \text{mult}(0)$). We will show:

$\lambda_1 = \dots = \lambda_k = 0 \implies G$ has at least k connected comp. C_1, \dots, C_k

- there exist f_1, \dots, f_k orthonormal such that $\sum_{\{u,v\} \in E} (f_i(u) - f_i(v))^2 = 0$

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \implies ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are orthogonal
- $\chi_{C_i}^T \mathbf{L} \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \implies \lambda_1 = \dots = \lambda_k = 0$

2. (“ \impliedby ” $cc(G) \geq \text{mult}(0)$). We will show:

$\lambda_1 = \dots = \lambda_k = 0 \implies G$ has at least k connected comp. C_1, \dots, C_k

- there exist f_1, \dots, f_k orthonormal such that $\sum_{\{u,v\} \in E} (f_i(u) - f_i(v))^2 = 0$
- $\implies f_1, \dots, f_k$ constant on connected components

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \implies ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are orthogonal
- $\chi_{C_i}^T L \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \implies \lambda_1 = \dots = \lambda_k = 0$

2. (“ \impliedby ” $cc(G) \geq \text{mult}(0)$). We will show:

$\lambda_1 = \dots = \lambda_k = 0 \implies G$ has at least k connected comp. C_1, \dots, C_k

- there exist f_1, \dots, f_k orthonormal such that $\sum_{\{u,v\} \in E} (f_i(u) - f_i(v))^2 = 0$
- $\implies f_1, \dots, f_k$ constant on connected components
- as f_1, \dots, f_k are pairwise orthogonal, G must have k different connected components.

Proof of Lemma, 2nd statement (non-examinable)

Let us generalise and formalise the previous example!

Proof (multiplicity of 0 equals the no. of connected components):

1. (“ \implies ” $cc(G) \leq \text{mult}(0)$). We will show:

G has exactly k connected comp. $C_1, \dots, C_k \implies \lambda_1 = \dots = \lambda_k = 0$

- Take $\chi_{C_i} \in \{0, 1\}^n$ such that $\chi_{C_i}(u) = \mathbf{1}_{u \in C_i}$ for all $u \in V$
- Clearly, the χ_{C_i} 's are orthogonal
- $\chi_{C_i}^T L \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \implies \lambda_1 = \dots = \lambda_k = 0$

2. (“ \impliedby ” $cc(G) \geq \text{mult}(0)$). We will show:

$\lambda_1 = \dots = \lambda_k = 0 \implies G$ has at least k connected comp. C_1, \dots, C_k

- there exist f_1, \dots, f_k orthonormal such that $\sum_{\{u,v\} \in E} (f_i(u) - f_i(v))^2 = 0$
- $\implies f_1, \dots, f_k$ constant on connected components
- as f_1, \dots, f_k are pairwise orthogonal, G must have k different connected components.



Randomised Algorithms

Lecture 12: Spectral Graph Clustering

Thomas Sauerwald (tms41@cam.ac.uk)

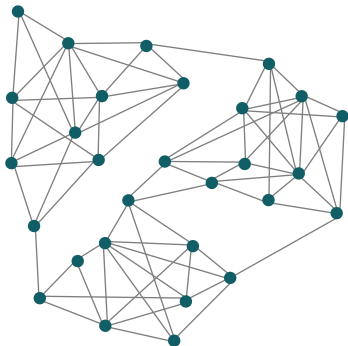
Conductance, Cheeger's Inequality and Spectral Clustering

Illustrations of Spectral Clustering and Extension to Non-Regular Graphs

Appendix: Relating Spectrum to Mixing Times (non-examinable)

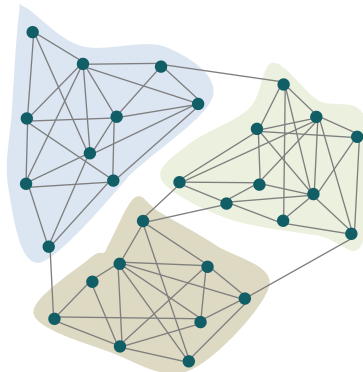
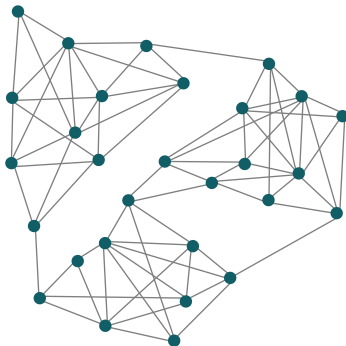
Graph Clustering

*Partition the graph into **pieces (clusters)** so that vertices in the same piece have, on average, more connections among each other than with vertices in other clusters*



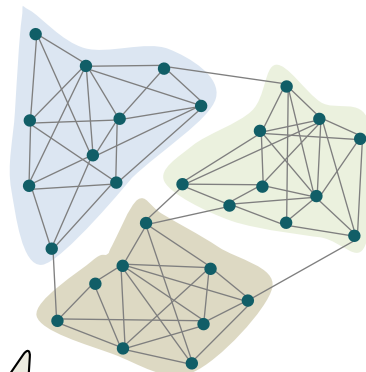
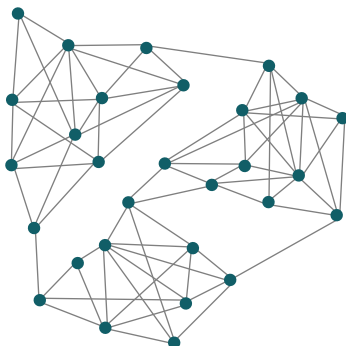
Graph Clustering

*Partition the graph into **pieces (clusters)** so that vertices in the same piece have, on average, more connections among each other than with vertices in other clusters*



Graph Clustering

Partition the graph into **pieces (clusters)** so that vertices in the same piece have, on average, more connections among each other than with vertices in other clusters



Let us for simplicity focus on the case of **two clusters**!

Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$.
The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$.
The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$

Conductance

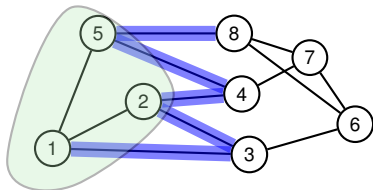
Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$



Conductance

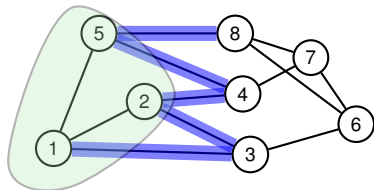
Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$.
The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$



▪ $\phi(S) = ??$

Conductance

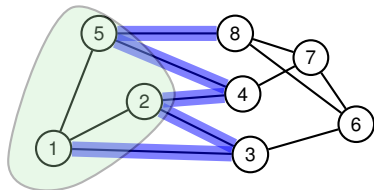
Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$



▪ $\phi(S) = \frac{5}{9}$

Conductance

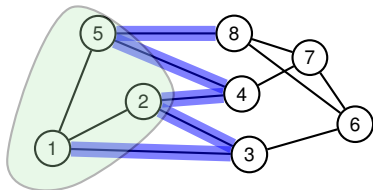
Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$



- $\phi(S) = \frac{5}{9}$
- $\phi(G) \in [0, 1]$ and $\phi(G) = 0$ iff G is disconnected

Conductance

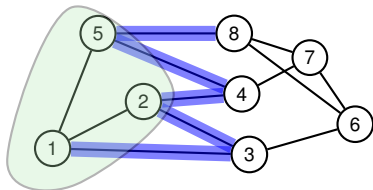
Conductance

Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$



- $\phi(S) = \frac{5}{9}$
- $\phi(G) \in [0, 1]$ and $\phi(G) = 0$ iff G is disconnected
- If G is a **complete graph**, then $e(S, V \setminus S) = |S| \cdot (n - |S|)$ and $\phi(G) \approx 1/2$.

Conductance

Conductance

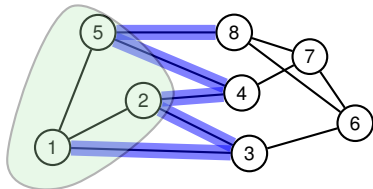
Let $G = (V, E)$ be a d -regular and undirected graph and $\emptyset \neq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the **conductance** (edge expansion) of the graph G is

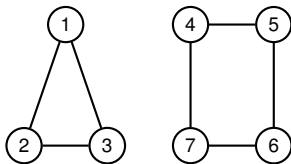
$$\phi(G) := \min_{S \subseteq V: 1 \leq |S| \leq n/2} \phi(S)$$

NP-hard to compute!

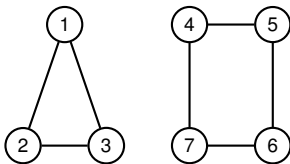


- $\phi(S) = \frac{5}{9}$
- $\phi(G) \in [0, 1]$ and $\phi(G) = 0$ iff G is disconnected
- If G is a **complete graph**, then $e(S, V \setminus S) = |S| \cdot (n - |S|)$ and $\phi(G) \approx 1/2$.

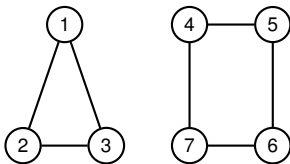
G is disconnected



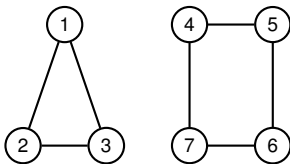
G is disconnected



$$\phi(G) = 0 \iff G \text{ is disconnected}$$



$$\phi(G) = 0 \iff G \text{ is disconnected} \iff \lambda_2(G) = 0$$



$$\phi(G) = 0 \iff G \text{ is disconnected} \iff \lambda_2(G) = 0$$

What is the relationship between $\phi(G)$ and $\lambda_2(G)$ for **connected** graphs?

λ_2 versus Conductance (2/2)

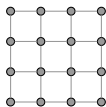
1D Grid (Path)



$$\lambda_2 \sim n^{-2}$$

$$\phi \sim n^{-1}$$

2D Grid



$$\lambda_2 \sim n^{-1}$$

$$\phi \sim n^{-1/2}$$

3D Grid



$$\lambda_2 \sim n^{-2/3}$$

$$\phi \sim n^{-1/3}$$

λ_2 versus Conductance (2/2)

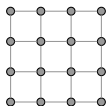
1D Grid (Path)



$$\lambda_2 \sim n^{-2}$$

$$\phi \sim n^{-1}$$

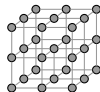
2D Grid



$$\lambda_2 \sim n^{-1}$$

$$\phi \sim n^{-1/2}$$

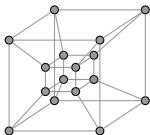
3D Grid



$$\lambda_2 \sim n^{-2/3}$$

$$\phi \sim n^{-1/3}$$

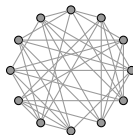
Hypercube



$$\lambda_2 \sim (\log n)^{-1}$$

$$\phi \sim (\log n)^{-1}$$

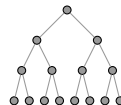
Random Graph (Expanders)



$$\lambda_2 = \Theta(1)$$

$$\phi = \Theta(1)$$

Binary Tree



$$\lambda_2 \sim n^{-1}$$

$$\phi \sim n^{-1}$$

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2
2. Order the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (embed V on \mathbb{R})

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2
2. Order the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (embed V on \mathbb{R})
3. Try all $n - 1$ sweep cuts of the form $(\{1, 2, \dots, k\}, \{k + 1, \dots, n\})$ and return the one with smallest conductance

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2
2. Order the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (embed V on \mathbb{R})
3. Try all $n - 1$ sweep cuts of the form $(\{1, 2, \dots, k\}, \{k + 1, \dots, n\})$ and return the one with smallest conductance

- It returns cluster $S \subseteq V$ such that $\phi(S) \leq \sqrt{2\lambda_2}$

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2
2. Order the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (embed V on \mathbb{R})
3. Try all $n - 1$ sweep cuts of the form $(\{1, 2, \dots, k\}, \{k + 1, \dots, n\})$ and return the one with smallest conductance

- It returns cluster $S \subseteq V$ such that $\phi(S) \leq \sqrt{2\lambda_2} \leq 2\sqrt{\phi(G)}$

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2
2. Order the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (embed V on \mathbb{R})
3. Try all $n - 1$ sweep cuts of the form $(\{1, 2, \dots, k\}, \{k + 1, \dots, n\})$ and return the one with smallest conductance

- It returns cluster $S \subseteq V$ such that $\phi(S) \leq \sqrt{2\lambda_2} \leq 2\sqrt{\phi(G)}$
- no constant factor worst-case guarantee, but usually works well in practice (see examples later!)

Relating λ_2 and Conductance

Cheeger's inequality

Let G be a d -regular undirected graph and $\lambda_1 \leq \dots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

Spectral Clustering:

1. Compute the eigenvector x corresponding to λ_2
2. Order the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (embed V on \mathbb{R})
3. Try all $n - 1$ sweep cuts of the form $(\{1, 2, \dots, k\}, \{k + 1, \dots, n\})$ and return the one with smallest conductance

- It returns cluster $S \subseteq V$ such that $\phi(S) \leq \sqrt{2\lambda_2} \leq 2\sqrt{\phi(G)}$
- no constant factor worst-case guarantee, but usually works well in practice (see examples later!)
- **very fast**: can be implemented in $O(|E| \log |E|)$ time

Proof of Cheeger's Inequality (non-examinable)

Proof (of the easy direction):

Proof of Cheeger's Inequality (non-examinable)

Proof (of the easy direction):

- By the Courant-Fischer Formula,

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{x^T \mathbf{L} x}{x^T x}$$

Proof of Cheeger's Inequality (non-examinable)

Proof (of the easy direction):

- By the Courant-Fischer Formula,

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{\sum_{u \sim v} (x_u - x_v)^2}{\sum_u x_u^2}.$$

Proof of Cheeger's Inequality (non-examinable)

Proof (of the easy direction):

- By the Courant-Fischer Formula,

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{\sum_{u \sim v} (x_u - x_v)^2}{\sum_u x_u^2}.$$

Optimisation Problem: Embed vertices on a line such that sum of squared distances is minimised

Proof of Cheeger's Inequality (non-examinable)

Proof (of the easy direction):

- By the Courant-Fischer Formula,

Optimisation Problem: Embed vertices on a line such that sum of squared distances is minimised

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{\sum_{u \sim v} (x_u - x_v)^2}{\sum_u x_u^2}.$$

- Let $S \subseteq V$ be the subset for which $\phi(G)$ is minimised. Define $y \in \mathbb{R}^n$ by:

$$y_u = \begin{cases} \frac{1}{|S|} & \text{if } u \in S, \\ -\frac{1}{|V \setminus S|} & \text{if } u \in V \setminus S. \end{cases}$$

Proof of Cheeger's Inequality (non-examinable)

Proof (of the easy direction):

- By the Courant-Fischer Formula,

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{x^T L x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{\sum_{u \sim v} (x_u - x_v)^2}{\sum_u x_u^2}.$$

Optimisation Problem: Embed vertices on a line such that sum of squared distances is minimised

- Let $S \subseteq V$ be the subset for which $\phi(G)$ is minimised. Define $y \in \mathbb{R}^n$ by:

$$y_u = \begin{cases} \frac{1}{|S|} & \text{if } u \in S, \\ -\frac{1}{|V \setminus S|} & \text{if } u \in V \setminus S. \end{cases}$$

- Since $y \perp 1$, it follows that

$$\begin{aligned} \lambda_2 &\leq \frac{1}{d} \cdot \frac{\sum_{u \sim v} (y_u - y_v)^2}{\sum_u y_u^2} = \frac{1}{d} \cdot \frac{|E(S, V \setminus S)| \cdot \left(\frac{1}{|S|} + \frac{1}{|V \setminus S|}\right)^2}{\frac{1}{|S|} + \frac{1}{|V \setminus S|}} \\ &= \frac{1}{d} \cdot |E(S, V \setminus S)| \cdot \left(\frac{1}{|S|} + \frac{1}{|V \setminus S|}\right) \\ &\leq \frac{1}{d} \cdot \frac{2 \cdot |E(S, V \setminus S)|}{|S|} = 2 \cdot \phi(G). \quad \square \end{aligned}$$

Conductance, Cheeger's Inequality and Spectral Clustering

Illustrations of Spectral Clustering and Extension to Non-Regular Graphs

Appendix: Relating Spectrum to Mixing Times (non-examinable)

Illustration on a small Example

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & -\frac{1}{\omega_3} & -\frac{1}{\omega_3} & 0 & 0 & -\frac{1}{\omega_3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{\omega_3} & -\frac{1}{\omega_3} & -\frac{1}{\omega_3} & 0 \\ -\frac{1}{\omega_3} & 0 & 1 & -\frac{1}{\omega_3} & 0 & 0 & 0 & -\frac{1}{\omega_3} \\ -\frac{1}{\omega_3} & 0 & -\frac{1}{\omega_3} & 1 & 0 & 0 & -\frac{1}{\omega_3} & 0 \\ 0 & -\frac{1}{\omega_3} & 0 & 0 & 1 & -\frac{1}{\omega_3} & 0 & -\frac{1}{\omega_3} \\ 0 & -\frac{1}{\omega_3} & 0 & 0 & -\frac{1}{\omega_3} & 1 & 0 & -\frac{1}{\omega_3} \\ -\frac{1}{\omega_3} & -\frac{1}{\omega_3} & 0 & -\frac{1}{\omega_3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\omega_3} & 0 & -\frac{1}{\omega_3} & -\frac{1}{\omega_3} & 0 & 1 \end{pmatrix}$$

Illustration on a small Example

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$

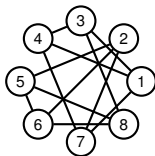
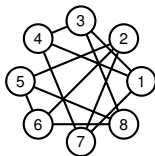


Illustration on a small Example

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & -\frac{1}{\omega_2} & -\frac{1}{\omega_2} & 0 & 0 & -\frac{1}{\omega_2} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{\omega_2} & -\frac{1}{\omega_2} & 0 & 0 \\ -\frac{1}{\omega_2} & 0 & 1 & -\frac{1}{\omega_2} & 0 & 0 & 0 & -\frac{1}{\omega_2} \\ -\frac{1}{\omega_2} & 0 & -\frac{1}{\omega_2} & 1 & 0 & 0 & -\frac{1}{\omega_2} & 0 \\ 0 & -\frac{1}{\omega_2} & 0 & 0 & 1 & -\frac{1}{\omega_2} & 0 & -\frac{1}{\omega_2} \\ 0 & -\frac{1}{\omega_2} & 0 & 0 & -\frac{1}{\omega_2} & 1 & 0 & -\frac{1}{\omega_2} \\ -\frac{1}{\omega_2} & -\frac{1}{\omega_2} & 0 & -\frac{1}{\omega_2} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\omega_2} & 0 & -\frac{1}{\omega_2} & -\frac{1}{\omega_2} & 0 & 1 \end{pmatrix}$$

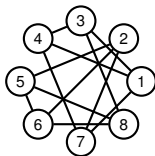


$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$\mathbf{v} = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

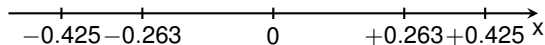
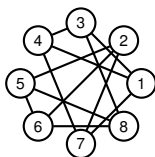


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

○
1

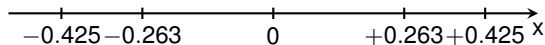
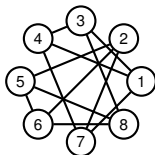


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

○
1

○
2

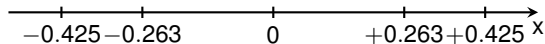
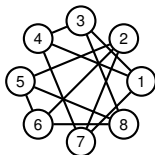


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

○
1

○
3

○
2

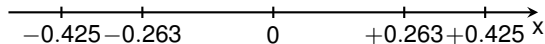
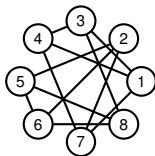


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

4
○

○
1

○
3

○
2

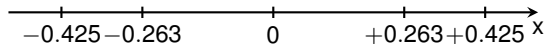
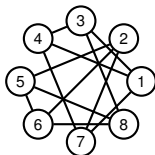


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

4
○

○
1

○
3

○
2

○
5

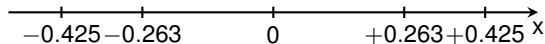
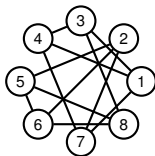


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

4
○

6
○

○
1

○
3

○
2

○
5

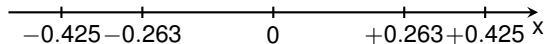
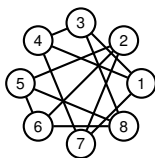


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

4 7
○ ○

6
○

○ ○
1 3

○ ○
2 5

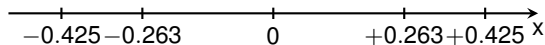
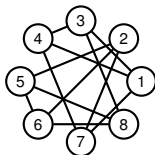


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

4 7
○ ○

8 6
○ ○

○ ○
1 3

○ ○
2 5

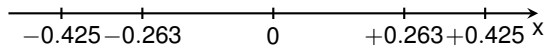
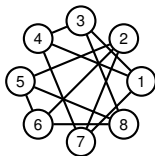


Illustration on a small Example

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$

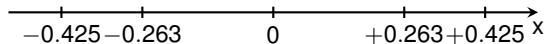
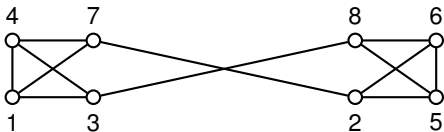
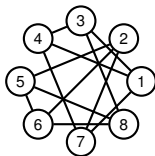


Illustration on a small Example

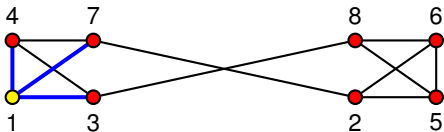
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 1

Conductance: 1

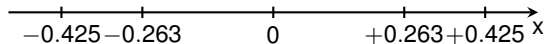
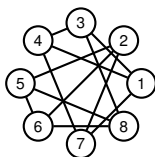


Illustration on a small Example

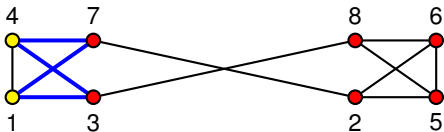
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 2

Conductance: 0.666

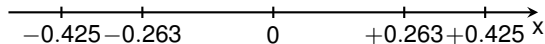
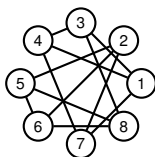


Illustration on a small Example

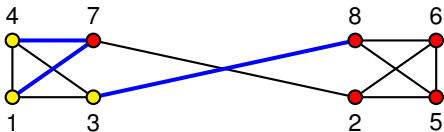
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 3

Conductance: 0.333

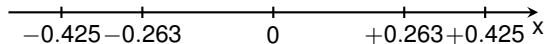
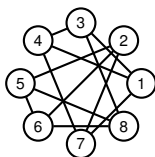


Illustration on a small Example

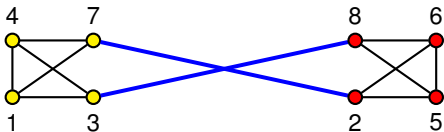
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 4

Conductance: 0.166

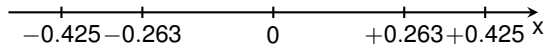
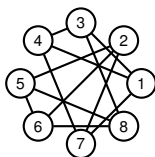


Illustration on a small Example

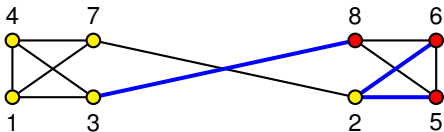
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 5

Conductance: 0.333

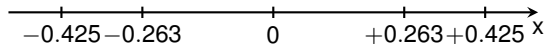
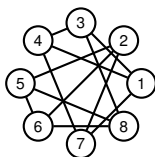


Illustration on a small Example

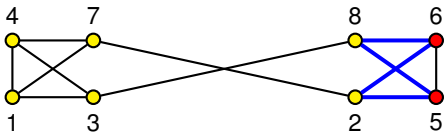
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 6

Conductance: 0.666

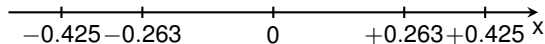
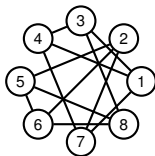


Illustration on a small Example

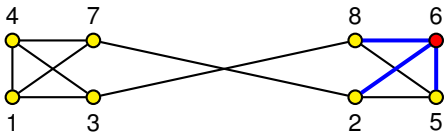
$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



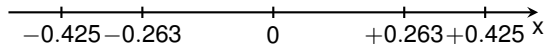
$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 7

Conductance: 1



Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v

Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v



Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$



Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$
- Courant-Fisher characterisation:

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp \mathbf{1}}} \frac{x^T \mathbf{L} x}{x^T x}$$

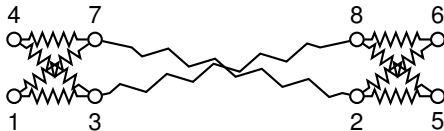


Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$
- Courant-Fisher characterisation:

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp \mathbf{1}}} \frac{x^T \mathbf{L} x}{x^T x}$$

- In our example, we found out that $\lambda_2 \approx 0.25$

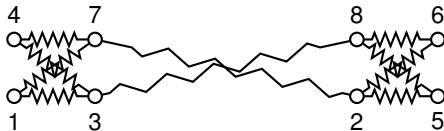


Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$
- Courant-Fisher characterisation:

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp \mathbf{1}}} \frac{x^T \mathbf{L} x}{x^T x}$$

- In our example, we found out that $\lambda_2 \approx 0.25$
- The eigenvector x on the last slide is normalised (i.e., $\|x\|_2^2 = 1$). Hence,



Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$
- Courant-Fisher characterisation:

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp \mathbf{1}}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ \|x\|_2^2 = 1, x \perp \mathbf{1}}} (x_u - x_v)^2$$

- In our example, we found out that $\lambda_2 \approx 0.25$
- The eigenvector x on the last slide is normalised (i.e., $\|x\|_2^2 = 1$). Hence,



Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$
- Courant-Fisher characterisation:

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp \mathbf{1}}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ \|x\|_2^2 = 1, x \perp \mathbf{1}}} (x_u - x_v)^2$$

- In our example, we found out that $\lambda_2 \approx 0.25$
- The eigenvector x on the last slide is normalised (i.e., $\|x\|_2^2 = 1$). Hence,

$$\lambda_2 = \frac{1}{3} \cdot \left((x_1 - x_3)^2 + (x_1 - x_4)^2 + (x_1 - x_7)^2 + \dots + (x_6 - x_8)^2 \right)$$



Physical Interpretation of the Minimisation Problem

- For each edge $\{u, v\} \in E(G)$, add spring between pins at x_u and x_v
- The potential energy at each spring is $(x_u - x_v)^2$
- Courant-Fisher characterisation:

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{0\} \\ x \perp \mathbf{1}}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ \|x\|_2^2 = 1, x \perp \mathbf{1}}} (x_u - x_v)^2$$

- In our example, we found out that $\lambda_2 \approx 0.25$
- The eigenvector x on the last slide is normalised (i.e., $\|x\|_2^2 = 1$). Hence,

$$\lambda_2 = \frac{1}{3} \cdot \left((x_1 - x_3)^2 + (x_1 - x_4)^2 + (x_1 - x_7)^2 + \dots + (x_6 - x_8)^2 \right) \approx 0.25$$



Let us now look at an example of a **non-regular** graph!

The Laplacian Matrix (General Version)

The (normalised) Laplacian matrix of $G = (V, E, w)$ is the n by n matrix

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

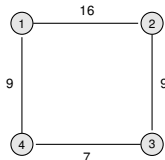
where \mathbf{D} is a diagonal $n \times n$ matrix such that $\mathbf{D}_{uu} = \deg(u) = \sum_{v: \{u,v\} \in E} w(u, v)$, and \mathbf{A} is the weighted adjacency matrix of G .

The Laplacian Matrix (General Version)

The (normalised) Laplacian matrix of $G = (V, E, w)$ is the n by n matrix

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

where \mathbf{D} is a diagonal $n \times n$ matrix such that $\mathbf{D}_{uu} = \deg(u) = \sum_{v: \{u,v\} \in E} w(u, v)$, and \mathbf{A} is the weighted adjacency matrix of G .



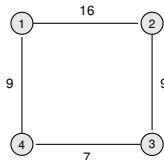
$$\mathbf{L} = \begin{pmatrix} 1 & -16/25 & 0 & -9/20 \\ -16/25 & 1 & -9/20 & 0 \\ 0 & -9/20 & 1 & -7/16 \\ -9/20 & 0 & -7/16 & 1 \end{pmatrix}$$

The Laplacian Matrix (General Version)

The (normalised) Laplacian matrix of $G = (V, E, w)$ is the n by n matrix

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$$

where \mathbf{D} is a diagonal $n \times n$ matrix such that $\mathbf{D}_{uu} = \deg(u) = \sum_{v: \{u,v\} \in E} w(u, v)$, and \mathbf{A} is the weighted adjacency matrix of G .



$$\mathbf{L} = \begin{pmatrix} 1 & -16/25 & 0 & -9/20 \\ -16/25 & 1 & -9/20 & 0 \\ 0 & -9/20 & 1 & -7/16 \\ -9/20 & 0 & -7/16 & 1 \end{pmatrix}$$

- $\mathbf{L}_{uv} = -\frac{w(u,v)}{\sqrt{d_u d_v}}$ for $u \neq v$
- \mathbf{L} is symmetric
- If G is d -regular, $\mathbf{L} = \mathbf{I} - \frac{1}{d} \cdot \mathbf{A}$.

Conductance and Spectral Clustering (General Version)

Conductance (General Version)

Let $G = (V, E, w)$ and $\emptyset \subsetneq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{w(S, S^c)}{\min\{\text{vol}(S), \text{vol}(S^c)\}},$$

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\text{vol}(S) := \sum_{u \in S} d(u)$. Moreover, the **conductance** (edge expansion) of G is

$$\phi(G) := \min_{\emptyset \neq S \subsetneq V} \phi(S).$$

Conductance and Spectral Clustering (General Version)

Conductance (General Version)

Let $G = (V, E, w)$ and $\emptyset \subsetneq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{w(S, S^c)}{\min\{\text{vol}(S), \text{vol}(S^c)\}},$$

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\text{vol}(S) := \sum_{u \in S} d(u)$. Moreover, the **conductance** (edge expansion) of G is

$$\phi(G) := \min_{\emptyset \neq S \subsetneq V} \phi(S).$$

Spectral Clustering (General Version):

Conductance and Spectral Clustering (General Version)

Conductance (General Version)

Let $G = (V, E, w)$ and $\emptyset \subsetneq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{w(S, S^c)}{\min\{\text{vol}(S), \text{vol}(S^c)\}},$$

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\text{vol}(S) := \sum_{u \in S} d(u)$. Moreover, the **conductance** (edge expansion) of G is

$$\phi(G) := \min_{\emptyset \neq S \subsetneq V} \phi(S).$$

Spectral Clustering (General Version):

1. Compute the eigenvector x corresponding to λ_2 **and** $y = \mathbf{D}^{-1/2}x$.

Conductance and Spectral Clustering (General Version)

Conductance (General Version)

Let $G = (V, E, w)$ and $\emptyset \subsetneq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{w(S, S^c)}{\min\{\text{vol}(S), \text{vol}(S^c)\}},$$

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\text{vol}(S) := \sum_{u \in S} d(u)$. Moreover, the **conductance** (edge expansion) of G is

$$\phi(G) := \min_{\emptyset \neq S \subsetneq V} \phi(S).$$

Spectral Clustering (General Version):

1. Compute the eigenvector x corresponding to λ_2 **and** $y = \mathbf{D}^{-1/2}x$.
2. Order the vertices so that $y_1 \leq y_2 \leq \dots \leq y_n$ (embed V on \mathbb{R})

Conductance and Spectral Clustering (General Version)

Conductance (General Version)

Let $G = (V, E, w)$ and $\emptyset \subsetneq S \subsetneq V$. The **conductance** (edge expansion) of S is

$$\phi(S) := \frac{w(S, S^c)}{\min\{\text{vol}(S), \text{vol}(S^c)\}},$$

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\text{vol}(S) := \sum_{u \in S} d(u)$. Moreover, the **conductance** (edge expansion) of G is

$$\phi(G) := \min_{\emptyset \neq S \subsetneq V} \phi(S).$$

Spectral Clustering (General Version):

1. Compute the eigenvector x corresponding to λ_2 **and** $y = D^{-1/2}x$.
2. Order the vertices so that $y_1 \leq y_2 \leq \dots \leq y_n$ (**embed** V on \mathbb{R})
3. Try all $n - 1$ **sweep cuts** of the form $(\{1, 2, \dots, k\}, \{k + 1, \dots, n\})$ and return the one with smallest conductance

Stochastic Block Model and 1D-Embedding

Stochastic Block Model

$G = (V, E)$ with clusters $S_1, S_2 \subseteq V$, $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & \text{if } u, v \in S_i, \\ q & \text{if } u \in S_i, v \in S_j, i \neq j. \end{cases}$$

Stochastic Block Model and 1D-Embedding

Stochastic Block Model

$G = (V, E)$ with clusters $S_1, S_2 \subseteq V$, $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & \text{if } u, v \in S_i, \\ q & \text{if } u \in S_i, v \in S_j, i \neq j. \end{cases}$$

Here:

- $|S_1| = 80$,
 $|S_2| = 120$
- $p = 0.08$
- $q = 0.01$

Stochastic Block Model and 1D-Embedding

Stochastic Block Model

$G = (V, E)$ with clusters $S_1, S_2 \subseteq V$, $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & \text{if } u, v \in S_i, \\ q & \text{if } u \in S_i, v \in S_j, i \neq j. \end{cases}$$

Here:

- $|S_1| = 80$,
 $|S_2| = 120$
- $p = 0.08$
- $q = 0.01$

Number of Vertices: 200

Number of Edges: 919

Eigenvalue 1 : -1.1968431479565368e-16

Eigenvalue 2 : 0.1543784937248489

Eigenvalue 3 : 0.37049909753568877

Eigenvalue 4 : 0.39770640242147404

Eigenvalue 5 : 0.4316114413430584

Eigenvalue 6 : 0.44379221120189777

Eigenvalue 7 : 0.4564011652684181

Eigenvalue 8 : 0.4632911204500282

Eigenvalue 9 : 0.474638606357877

Eigenvalue 10 : 0.4814019607292904

Stochastic Block Model and 1D-Embedding

Stochastic Block Model

$G = (V, E)$ with clusters $S_1, S_2 \subseteq V$, $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & \text{if } u, v \in S_i, \\ q & \text{if } u \in S_i, v \in S_j, i \neq j. \end{cases}$$

Here:

- $|S_1| = 80$,
 $|S_2| = 120$
- $p = 0.08$
- $q = 0.01$

Number of Vertices: 200

Number of Edges: 919

Eigenvalue 1 : -1.1968431479565368e-16

Eigenvalue 2 : 0.1543784937248489

Eigenvalue 3 : 0.37049909753568877

Eigenvalue 4 : 0.39770640242147404

Eigenvalue 5 : 0.4316114413430584

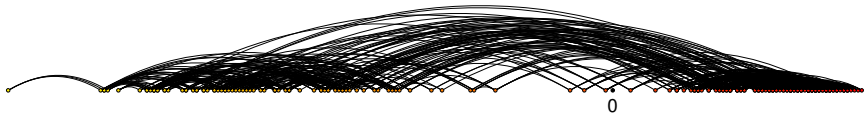
Eigenvalue 6 : 0.44379221120189777

Eigenvalue 7 : 0.4564011652684181

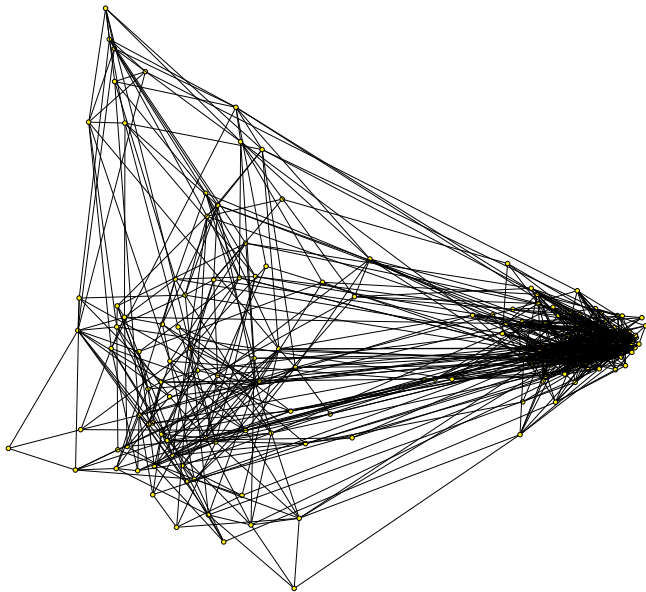
Eigenvalue 8 : 0.4632911204500282

Eigenvalue 9 : 0.474638606357877

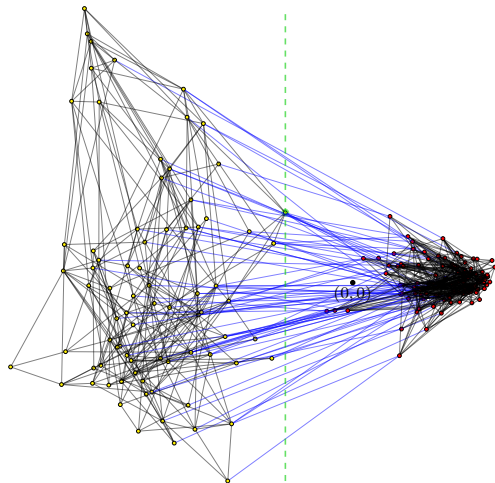
Eigenvalue 10 : 0.4814019607292904



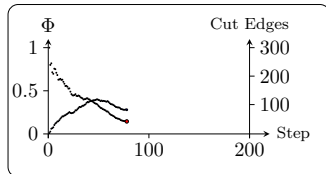
Drawing the 2D-Embedding



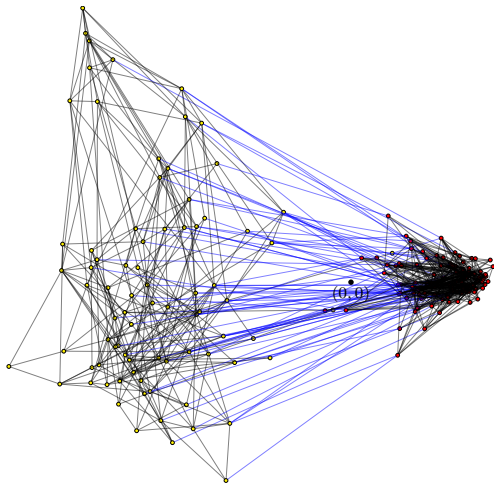
Best Solution found by Spectral Clustering



- Step: 78
- Threshold: -0.0336
- Partition Sizes: 78/122
- Cut Edges: 84
- Conductance: 0.1448



Clustering induced by Blocks



- Step: –
- Threshold: –
- Partition Sizes: 80/120
- Cut Edges: 88
- Conductance: 0.1486

Additional Example: Stochastic Block Models with 3 Clusters

Graph $G = (V, E)$ with clusters
 $S_1, S_2, S_3 \subseteq V$; $0 \leq q < p \leq 1$

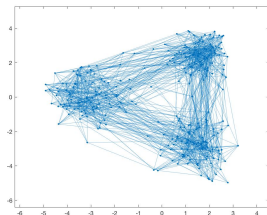
$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & u, v \in S_i \\ q & u \in S_i, v \in S_j, i \neq j \end{cases}$$

Additional Example: Stochastic Block Models with 3 Clusters

Graph $G = (V, E)$ with clusters
 $S_1, S_2, S_3 \subseteq V$; $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & u, v \in S_i \\ q & u \in S_i, v \in S_j, i \neq j \end{cases}$$

$|V| = 300, |S_i| = 100$
 $p = 0.08, q = 0.01$.

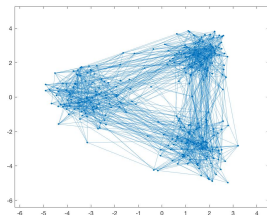


Additional Example: Stochastic Block Models with 3 Clusters

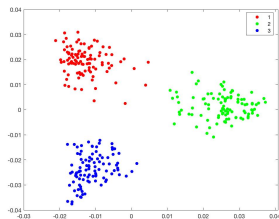
Graph $G = (V, E)$ with clusters
 $S_1, S_2, S_3 \subseteq V$; $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & u, v \in S_i \\ q & u \in S_i, v \in S_j, i \neq j \end{cases}$$

$|V| = 300, |S_i| = 100$
 $p = 0.08, q = 0.01$.



Spectral embedding

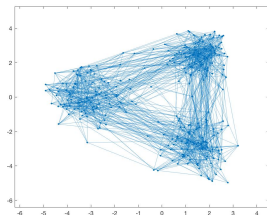


Additional Example: Stochastic Block Models with 3 Clusters

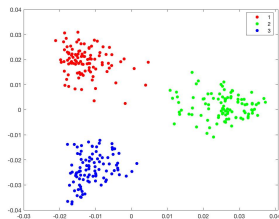
Graph $G = (V, E)$ with clusters
 $S_1, S_2, S_3 \subseteq V$; $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & u, v \in S_i \\ q & u \in S_i, v \in S_j, i \neq j \end{cases}$$

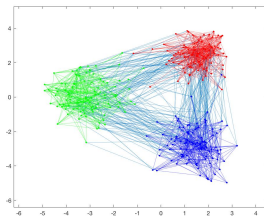
$|V| = 300, |S_i| = 100$
 $p = 0.08, q = 0.01$.



Spectral embedding



Output of Spectral Clustering



How to Choose the Cluster Number k

- If k is unknown:
 - small λ_k means there exist k sparsely connected subsets in the graph
(recall: $\lambda_1 = \dots = \lambda_k = 0$ means there are k connected components)

How to Choose the Cluster Number k

- If k is unknown:
 - small λ_k means there exist k sparsely connected subsets in the graph (recall: $\lambda_1 = \dots = \lambda_k = 0$ means there are k connected components)
 - large λ_{k+1} means all these k subsets have “good” inner-connectivity properties (cannot be divided further)

How to Choose the Cluster Number k

- If k is unknown:
 - small λ_k means there exist k sparsely connected subsets in the graph (recall: $\lambda_1 = \dots = \lambda_k = 0$ means there are k connected components)
 - large λ_{k+1} means all these k subsets have “good” inner-connectivity properties (cannot be divided further)
- ⇒ choose smallest $k \geq 2$ so that the spectral gap $\lambda_{k+1} - \lambda_k$ is “large”

How to Choose the Cluster Number k

- If k is unknown:
 - small λ_k means there exist k sparsely connected subsets in the graph (recall: $\lambda_1 = \dots = \lambda_k = 0$ means there are k connected components)
 - large λ_{k+1} means all these k subsets have “good” inner-connectivity properties (cannot be divided further)

\Rightarrow choose smallest $k \geq 2$ so that the spectral gap $\lambda_{k+1} - \lambda_k$ is “large”
- In the latter example $\lambda = \{0, 0.20, 0.22, 0.43, 0.45, \dots\} \implies k = 3$.

How to Choose the Cluster Number k

- If k is unknown:
 - small λ_k means there exist k sparsely connected subsets in the graph (recall: $\lambda_1 = \dots = \lambda_k = 0$ means there are k connected components)
 - large λ_{k+1} means all these k subsets have “good” inner-connectivity properties (cannot be divided further)

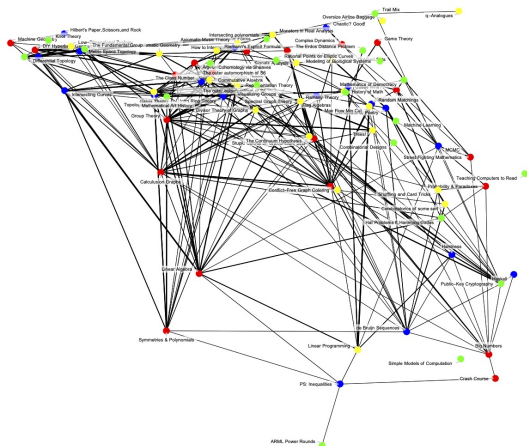
\Rightarrow choose smallest $k \geq 2$ so that the spectral gap $\lambda_{k+1} - \lambda_k$ is “large”
- In the latter example $\lambda = \{0, 0.20, 0.22, 0.43, 0.45, \dots\} \implies k = 3$.
- In the former example $\lambda = \{0, 0.15, 0.37, 0.40, 0.43, \dots\} \implies k = 2$.

How to Choose the Cluster Number k

- If k is unknown:
 - small λ_k means there exist k sparsely connected subsets in the graph (recall: $\lambda_1 = \dots = \lambda_k = 0$ means there are k connected components)
 - large λ_{k+1} means all these k subsets have “good” inner-connectivity properties (cannot be divided further)

\Rightarrow choose smallest $k \geq 2$ so that the spectral gap $\lambda_{k+1} - \lambda_k$ is “large”
- In the latter example $\lambda = \{0, 0.20, 0.22, 0.43, 0.45, \dots\} \implies k = 3$.
- In the former example $\lambda = \{0, 0.15, 0.37, 0.40, 0.43, \dots\} \implies k = 2$.
- For $k = 2$ use sweep-cut extract clusters. For $k \geq 3$ use embedding in k -dimensional space and apply k -means (geometric clustering)

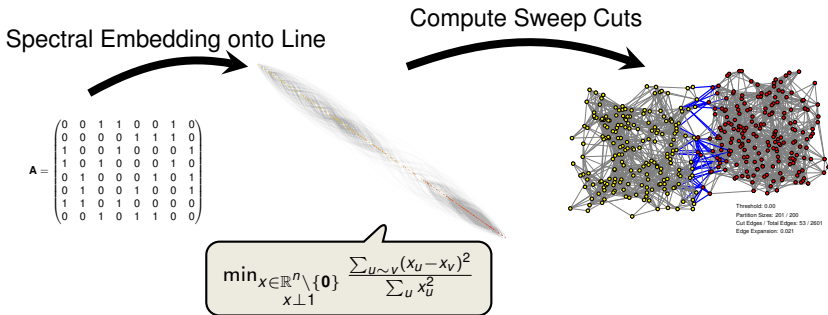
Another Example



(many thanks to Kalina Jasinska)

- nodes represent math topics taught within 4 weeks of a Mathcamp
- node colours represent to the week in which they thought
- teachers were asked to assign weights in 0 – 10 indicating how closely related two classes are

Summary: Spectral Clustering



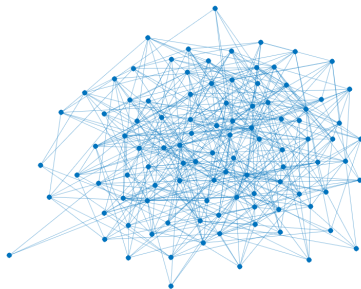
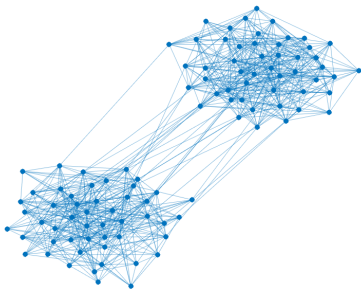
- Given any graph (adjacency matrix)
- Graph Spectrum (computable in poly-time)
 - λ_2 (relates to connectivity)
 - λ_n (relates to bipartiteness)
 - ...
- Cheeger's Inequality
 - relates λ_2 to conductance
 - unbounded approximation ratio
 - effective in practice

Conductance, Cheeger's Inequality and Spectral Clustering

Illustrations of Spectral Clustering and Extension to Non-Regular Graphs

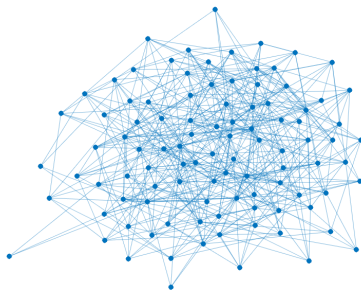
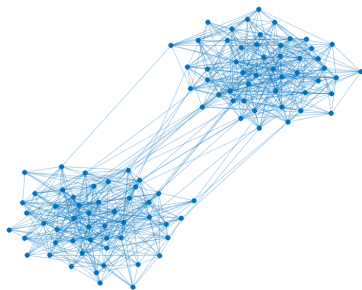
Appendix: Relating Spectrum to Mixing Times (non-examinable)

- Which graph has a “cluster-structure”?



Relation between Clustering and Mixing (non-examinable)

- Which graph has a “cluster-structure”?
- Which graph mixes faster?



Convergence of Random Walk (non-examinable)

Recall: If the underlying graph G is **connected**, **undirected** and **d -regular**, then the random walk converges towards the **stationary distribution** $\pi = (1/n, \dots, 1/n)$, which satisfies $\pi \mathbf{P} = \pi$.

Convergence of Random Walk (non-examinable)

Recall: If the underlying graph G is **connected**, **undirected** and **d -regular**, then the random walk converges towards the **stationary distribution** $\pi = (1/n, \dots, 1/n)$, which satisfies $\pi \mathbf{P} = \pi$.

Here all vector multiplications (including eigenvectors) will always be from the **left**!

Convergence of Random Walk (non-examinable)

Recall: If the underlying graph G is **connected, undirected and d -regular**, then the random walk converges towards the **stationary distribution** $\pi = (1/n, \dots, 1/n)$, which satisfies $\pi \mathbf{P} = \pi$.

Here all vector multiplications (including eigenvectors) will always be from the **left!**

— Lemma —

Consider a **lazy** random walk on a **connected, undirected and d -regular** graph. Then for any initial distribution x ,

$$\|x\mathbf{P}^t - \pi\|_2 \leq \lambda^t,$$

with $1 = \lambda_1 > \lambda_2 \geq \dots \geq \lambda_n$ as eigenvalues and $\lambda := \max\{|\lambda_2|, |\lambda_n|\}$.

Convergence of Random Walk (non-examinable)

Recall: If the underlying graph G is **connected, undirected and d -regular**, then the random walk converges towards the **stationary distribution** $\pi = (1/n, \dots, 1/n)$, which satisfies $\pi \mathbf{P} = \pi$.

Here all vector multiplications (including eigenvectors) will always be from the **left!**

— Lemma —

Consider a **lazy** random walk on a **connected, undirected and d -regular** graph. Then for any initial distribution x ,

$$\|x\mathbf{P}^t - \pi\|_2 \leq \lambda^t,$$

with $1 = \lambda_1 > \lambda_2 \geq \dots \geq \lambda_n$ as eigenvalues and $\lambda := \max\{|\lambda_2|, |\lambda_n|\}$.

due to laziness, $\lambda_n \geq 0$

Convergence of Random Walk (non-examinable)

Recall: If the underlying graph G is **connected, undirected and d -regular**, then the random walk converges towards the **stationary distribution** $\pi = (1/n, \dots, 1/n)$, which satisfies $\pi \mathbf{P} = \pi$.

Here all vector multiplications (including eigenvectors) will always be from the **left**!

— Lemma —

Consider a **lazy** random walk on a **connected, undirected and d -regular** graph. Then for any initial distribution x ,

$$\|x\mathbf{P}^t - \pi\|_2 \leq \lambda^t,$$

with $1 = \lambda_1 > \lambda_2 \geq \dots \geq \lambda_n$ as eigenvalues and $\lambda := \max\{|\lambda_2|, |\lambda_n|\}$.
 \Rightarrow This implies for $t = \mathcal{O}\left(\frac{\log n}{\log(1/\lambda)}\right) = \mathcal{O}\left(\frac{\log n}{1-\lambda}\right)$,

$$\|x\mathbf{P}^t - \pi\|_{tv} \leq \frac{1}{4}.$$

due to laziness, $\lambda_n \geq 0$

Proof of Lemma (non-examinable)

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\begin{aligned} \|x\mathbf{P} - \pi\|_2^2 &= \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2 \\ &= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2 \end{aligned}$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\begin{aligned}\|x\mathbf{P} - \pi\|_2^2 &= \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2 \\ &= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2 \\ &= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2\end{aligned}$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

$$= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2$$

since the v_i 's
are orthogonal

$$= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2$$

$$= \sum_{i=2}^n \|\alpha_i \lambda_i v_i\|_2^2$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

$$= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2$$

since the v_i 's are orthogonal

$$= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2$$

$$= \sum_{i=2}^n \|\alpha_i \lambda_i v_i\|_2^2$$

since the v_i 's are orthogonal

$$\leq \lambda^2 \sum_{i=2}^n \|\alpha_i v_i\|_2^2 = \lambda^2 \left\| \sum_{i=2}^n \alpha_i v_i \right\|_2^2$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

$$= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2$$

since the v_i 's are orthogonal

$$= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2$$

$$= \sum_{i=2}^n \|\alpha_i \lambda_i v_i\|_2^2$$

since the v_i 's are orthogonal

$$\leq \lambda^2 \sum_{i=2}^n \|\alpha_i v_i\|_2^2 = \lambda^2 \left\| \sum_{i=2}^n \alpha_i v_i \right\|_2^2 = \lambda^2 \|x - \pi\|_2^2$$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

$$= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2$$

since the v_i 's
are orthogonal

$$= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2$$

$$= \sum_{i=2}^n \|\alpha_i \lambda_i v_i\|_2^2$$

since the v_i 's
are orthogonal

$$\leq \lambda^2 \sum_{i=2}^n \|\alpha_i v_i\|_2^2 = \lambda^2 \left\| \sum_{i=2}^n \alpha_i v_i \right\|_2^2 = \lambda^2 \|x - \pi\|_2^2$$

- Hence $\|x\mathbf{P}^t - \pi\|_2^2$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

$$= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2$$

since the v_i 's
are orthogonal

$$= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2$$

$$= \sum_{i=2}^n \|\alpha_i \lambda_i v_i\|_2^2$$

since the v_i 's
are orthogonal

$$\leq \lambda^2 \sum_{i=2}^n \|\alpha_i v_i\|_2^2 = \lambda^2 \left\| \sum_{i=2}^n \alpha_i v_i \right\|_2^2 = \lambda^2 \|x - \pi\|_2^2$$

- Hence $\|x\mathbf{P}^t - \pi\|_2^2 \leq \lambda^{2t} \cdot \|x - \pi\|_2^2$

Proof of Lemma (non-examinable)

- Express x in terms of the orthonormal basis of \mathbf{P} , $v_1 = \pi, v_2, \dots, v_n$:

$$x = \sum_{i=1}^n \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π , $\alpha_1 = 1$.

\Rightarrow

$$\|x\mathbf{P} - \pi\|_2^2 = \left\| \left(\sum_{i=1}^n \alpha_i v_i \right) \mathbf{P} - \pi \right\|_2^2$$

$$= \left\| \pi + \sum_{i=2}^n \alpha_i \lambda_i v_i - \pi \right\|_2^2$$

since the v_i 's are orthogonal

$$= \left\| \sum_{i=2}^n \alpha_i \lambda_i v_i \right\|_2^2$$

$$= \sum_{i=2}^n \|\alpha_i \lambda_i v_i\|_2^2$$

since the v_i 's are orthogonal

$$\leq \lambda^2 \sum_{i=2}^n \|\alpha_i v_i\|_2^2 = \lambda^2 \left\| \sum_{i=2}^n \alpha_i v_i \right\|_2^2 = \lambda^2 \|x - \pi\|_2^2$$

- Hence $\|x\mathbf{P}^t - \pi\|_2^2 \leq \lambda^{2t} \cdot \|x - \pi\|_2^2 \leq \lambda^{2t} \cdot 1$.

$$\|x - \pi\|_2^2 + \|\pi\|_2^2 = \|x\|_2^2 \leq 1$$

Some References on Spectral Graph Theory and Clustering



Fan R.K. Chung.

Graph Theory in the Information Age.

Notices of the AMS, vol. 57, no. 6, pages 726–732, 2010.



Fan R.K. Chung.

Spectral Graph Theory.

Volume 92 of CBMS Regional Conference Series in Mathematics, 1997.



S. Hoory, N. Linial and A. Wigderson.

Expander Graphs and their Applications.

Bulletin of the AMS, vol. 43, no. 4, pages 439–561, 2006.



Daniel Spielman.

Chapter 16, Spectral Graph Theory

Combinatorial Scientific Computing, 2010.



Luca Trevisan.

Lectures Notes on Graph Partitioning, Expanders and Spectral Methods,
2017.

<https://lucatrevisan.github.io/books/expanders-2016.pdf>

Thank you and Best Wishes for the Exam!

I'm very interested to hear your feedback about the slides and the course more generally. You can use the student feedback form or send me an email during or after the course (tms41@cam.ac.uk).