Randomised Algorithms

Lecture 1: Introduction to Course & Introduction to Chernoff Bounds

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

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.

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

But sometimes: simple algorithm at the cost of a complicated analysis!

"... 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...

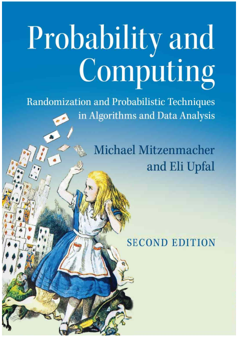
In this course we will assume some basic knowledge of probability:

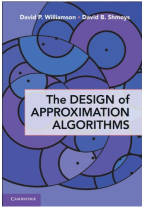
- random variable
- computing expectations and variances
- notions of independence
- "general" idea of how to compute probabilities (manipulating, counting and **estimating**)

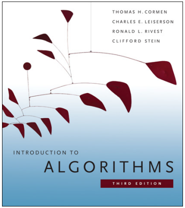
You should also be familiar with basic computer science, mathematics knowledge such as:

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

Textbooks







- **($\star$) 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.

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

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

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 $(\Omega, \Sigma, \mathbf{P})$.

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

- The Sample Space Ω contains all the possible outcomes $\omega_1, \omega_2, \ldots$ 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 $\mathcal{E}_1, \mathcal{E}_2, \ldots \in \Sigma$ 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 $X : \Omega \to \mathbb{R}$ 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 $X_1, X_2 \in \{1, 2, \ldots, 6\}$ is

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

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 $\mathcal{E}_1, \ldots, \mathcal{E}_n$ 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 $B \subseteq V$.

MAX-CUT Problem

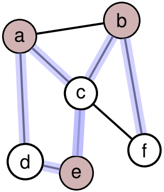
- Given: Undirected graph $G = (V, E)$
- Goal: Find $S \subseteq V$ such that $e(S, S^c) := |E(S, V \setminus S)|$ is maximised.

Applications:

- network design, VLSI design
- clustering, statistical physics

Comments:

- This example will appear again in the course
- 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)

```
RandMaxCut(G)
1: Start with $S \leftarrow \emptyset$
2: For each $v \in V$, add $v$ to $S$ with probability $1/2$
3: Return $S$
```

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

More details on approximation algorithms from Lecture 9 onwards!

Proposition

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

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

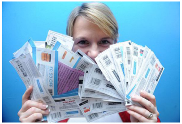
Proof:

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

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

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

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 (Supervision)

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

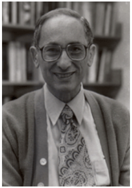
Concentration Inequalities

- 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
- It gives us the best of two worlds:
    1. **Randomised Algorithms:** Easy to Design and Implement
    2. **Deterministic Algorithms:** They do what they claim

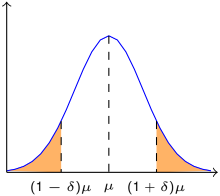
Chernoff Bounds: A Tool for Concentration

- Chernoffs 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
    - Statistics
    - Random Projections and Dimensionality Reduction
    - Learning Theory (e.g., PAC-learning)

    ⋮



Hermann Chernoff (1923-)



Recap: Markov and Chebyshev

Markov's Inequality

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

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

Chebyshev's Inequality

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

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

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

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

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

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

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

From Markov and Chebyshev to Chernoff

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

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

This implies that for any $t > \mu$,

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

While $(\bigstar)$ 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
- $X_i \in \{0, 1\}$, $X = \sum_{i=1}^{n} X_i$ 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)

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.

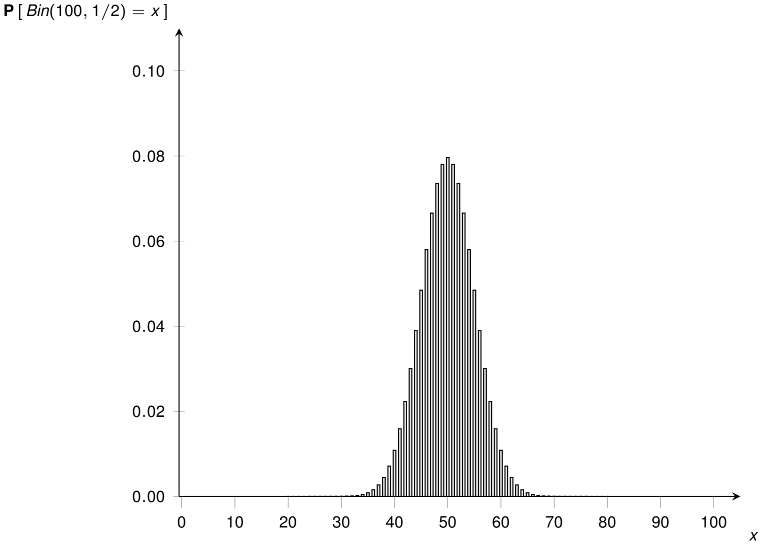
- Chernoff bound: setting $\delta = 1/2$ gives

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

- Remark: The exact probability is **0.00000028 . . .**

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

Example: Coin Flips (3/3)



Randomised Algorithms

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

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

Outline

How to Derive Chernoff Bounds

Application 1: Balls into Bins

General Recipe for Deriving Chernoff Bounds

Recipe

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

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

Chernoff Bound: Proof

Chernoff Bound (General Form, Upper Tail)

Suppose $X_1, \ldots, X_n$ are independent Bernoulli random variables with parameter p_i. Let $X = X_1 + \ldots + X_n$ 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 $\lambda > 0$,

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

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 $\lambda > 0$,

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

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

4. Putting all together

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

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 $X_1, \ldots, X_n$ are independent Bernoulli random variables with parameter p_i. Let $X = X_1 + \ldots + X_n$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} p_i$. Then, for any $\delta > 0$ 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 $X_1, \ldots, X_n$ are independent Bernoulli random variables with parameter p_i. Let $X = X_1 + \ldots + X_n$ 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 $x_1, x_2, \ldots, x_{i-1} \in \{0, 1\}$,

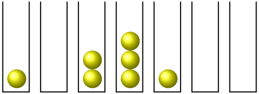
$$\mathbf{P}[X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq p_i.$$

Outline

How to Derive Chernoff Bounds

Application 1: Balls into Bins

Balls into Bins



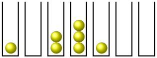
Balls into Bins Model

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.

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.
- 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,

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

- 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
$\mathbf{P}\left[\,\mathcal{E}\,\right] \to 1$ as $n \to \infty$.
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 $m = n$?

- Using the Chernoff Bound:

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

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

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

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

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

This inequality only works for large enough n.

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

We just proved that

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

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

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. $1 - 1/n$.

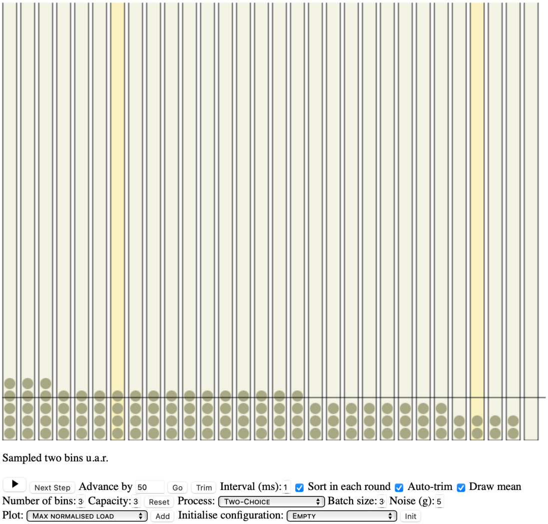
This is called the **power of two choices**: It is a common technique to improve the performance of randomised algorithms (covered in Chapter 17 of the textbook by Mitzenmacher and Upfal)

ACM Paris Kanellakis Theory and Practice Award 2020

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."

Simulation



Sampled two bins u.a.r.

Advance by 50

Interval (ms): 1

☑ Sort in each round

☑ Auto-trim

☑ Draw mean

Number of bins: 3

Capacity: 3

Process: TWO-CHOICE

Batch size: 3

Noise (g): 5

Plot: MAX NORMALISED LOAD

Initialise configuration: EMPTY

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

Randomised Algorithms

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

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: Moment Generating Functions

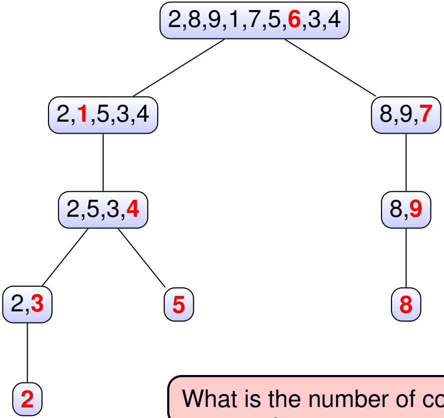
QuickSort

```
QUICKSORT (Input $A[1], A[2], \dots, A[n]$)
Pick an element from the array, the so-called pivot
If $|A| = 0$ or $|A| = 1$ then
    return $A$
else
    Create two subarrays $A_1$ and $A_2$ (without the pivot) such that:
        $A_1$ contains the elements that are smaller than the pivot
        $A_2$ contains the elements that are greater (or equal) than the pivot
    QUICKSORT($A_1$)
    QUICKSORT($A_2$)
    return $A$
```

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



What is the number of comparisons?

Note that the number of comparison by QUICKSORT is equivalent to the sum of the height 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.**

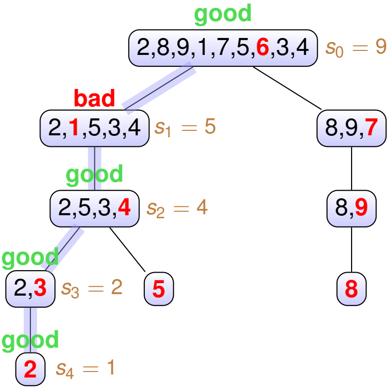
This should be your standard answer in this course ☺

Let us analyse QUICKSORT with random pivots.

1. Assume A consists of n different numbers, w.l.o.g., $\{1, 2, \ldots, n\}$
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 exists $C > 0$ such that
$$\mathbf{P}\left[H \leq Cn \log n \right] \geq 1 - n^{-1}.$$
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
  - 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: $(2,8,9,1,7,5,\mathbf{6},3,4) \to (2,\mathbf{1},5,3,4) \to (2,5,3,\mathbf{4}) \to (2,\mathbf{3}) \to (\mathbf{2})$

Randomised QuickSort: Analysis (3/4)

- Consider now any element $i \in \{1, 2, \ldots, n\}$ 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.

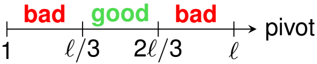
- Assume $|P| \geq C \log n$ for $C := 24$

  $\Rightarrow$ number of **bad** vertices 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 $24 \log n$ vertices of P to the deepest level of element i.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, 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**.
- $\mathbf{P}\left[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,\right] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

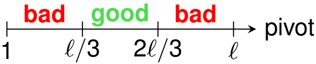


Question: But what if the path P does not reach level j?

Answer: We can then simply define X_j as the result of an independent coin flip with probability $2/3$.

Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ vertices of P to the deepest level of element i.
- For any level $j \in \{0, 1, \ldots, 24 \log n - 1\}$, 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**.



- $\mathbf{P}\left[\, X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \,\right] \leq \frac{2}{3}$
- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

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

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

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

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

$$= e^{-(50/24) \log n} \leq n^{-2}.$$

- 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: 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)

Supervision Exercise: 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!

Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: Moment Generating Functions

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

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

and

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

Proof Outline (skipped):

- Let $X_i' = X_i - \mu_i$ and $X' = X_1' + \ldots + X_n'$, then $\mathbf{P}\left[X \geq \mu + t\right] = \mathbf{P}\left[X' \geq t\right]$
- $\mathbf{P}\left[X' \geq t\right] \leq e^{-\lambda t} \prod_{i=1}^{n} \mathbf{E}\left[e^{\lambda X_i'}\right] \leq \exp\left[-\lambda t + \frac{\lambda^2}{8} \sum_{i=1}^{n}(b_i - a_i)^2\right]$
- 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 $X_1, \ldots, X_n$. We want to study the random variable:

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

Some examples:

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

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

Method of Bounded Differences

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

$$|f(x_1, x_2, \ldots, x_{i-1}, \mathbf{x_i}, x_{i+1}, \ldots, x_n) - f(x_1, x_2, \ldots, x_{i-1}, \widetilde{\mathbf{x_i}}, x_{i+1}, \ldots, x_n)| \leq c_i,$$

where x_i and $\widetilde{x_i}$ are in the domain of the i-th coordinate.

McDiarmid's inequality

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

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

and

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

- Notice the similarity with Hoeffding's inequality!
- The proof is omitted here (it requires the concept of martingales).

Outline

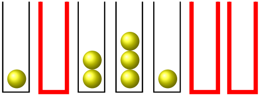
Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: Moment Generating Functions

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

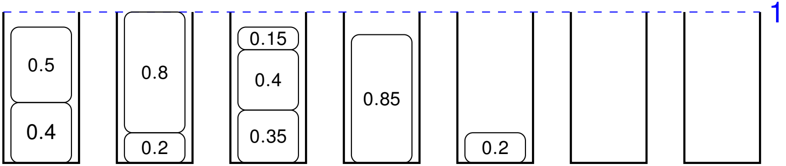


- Consider again m balls assigned uniformly at random into n bins.
- Enumerate the balls from 1 to m. Ball i is assigned to a random bin X_i
- Let Z be the number of empty bins (after assigning the m balls)
- $Z = Z(X_1, \ldots, X_m)$ and Z is Lipschitz with $\mathbf{c} = (1, \ldots, 1)$
  (If we move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

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

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 $[0, 1]$
- 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 $B = B(X_1, \ldots, X_n)$ be the optimal number of bins
- The Lipschitz conditions holds with $\boldsymbol{c} = (1, \ldots, 1)$. **Why?**
- Therefore

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

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

Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: Moment Generating Functions

Moment Generating Functions

Moment-Generating Function

The moment-generating function of a random variable X is

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

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:

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

Randomised Algorithms

Lecture 4: Markov Chains and Mixing Times

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

Outline

Recap of Markov Chain Basics

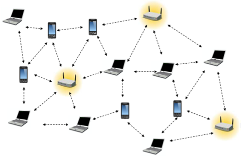
Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

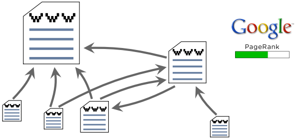
Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

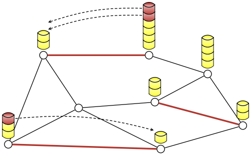
Applications of Markov Chains in Computer Science



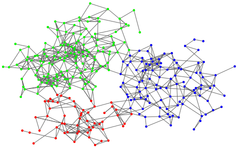
Broadcasting



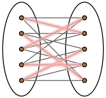
Ranking Websites



Load Balancing

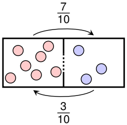


Clustering



$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Sampling and Optimisation



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:

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 $x_0, \ldots, x_{t+1} \in \Omega$,

$$\begin{aligned}\mathbf{P}\left[\, X_{t+1} = x_{t+1} \,\middle|\, X_t = x_t, \ldots, X_0 = x_0 \,\right] &= \mathbf{P}\left[\, X_{t+1} = x_{t+1} \,\middle|\, X_t = x_t \,\right] \\ &:= P(x_t, x_{t+1}).\end{aligned}$$

From the definition one can deduce that (check!)

- For all $t, x_0, x_1, \ldots, x_t \in \Omega$,

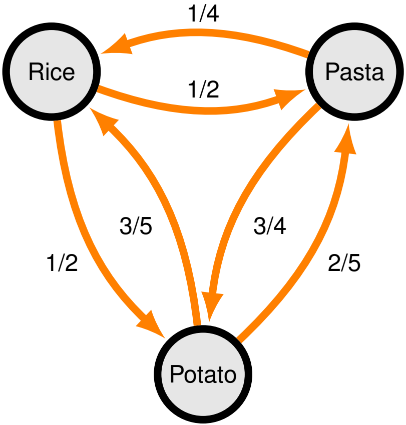
$$\begin{aligned}&\mathbf{P}[\, X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0 \,] \\ &= \mu(x_0) \cdot P(x_0, x_1) \cdot \ldots \cdot P(x_{t-2}, x_{t-1}) \cdot P(x_{t-1}, x_t).\end{aligned}$$

- 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:

$$P = \begin{array}{c@{}l} \begin{array}{ccc} \text{Rice} & \text{Pasta} & \text{Potato} \end{array} & \\ \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{bmatrix} & \begin{array}{l} \text{Rice} \\ \text{Pasta} \\ \text{Potato} \end{array} \end{array}$$

Transition Matrices and Distributions

The Transition Matrix P of a Markov chain (μ, P) on $\Omega = \{1, \ldots n\}$ is given by

$$P = \begin{pmatrix} P(1,1) & \ldots & P(1,n) \\ \vdots & \ddots & \vdots \\ P(n,1) & \ldots & P(n,n) \end{pmatrix}.$$

- $\rho^t = (\rho^t(1), \rho^t(2), \ldots, \rho^t(n))$: state vector at time t (row vector).
- Multiplying ρ^t by P corresponds to advancing the chain one step:

$$\rho^t(y) = \sum_{j \in \Omega} \rho^{t-1}(x) \cdot P(x,y) \qquad \text{and thus} \qquad \rho^t = \rho^{t-1} \cdot P.$$

- The Markov Property and line above imply that for any $t \geq 0$

$$\rho^t = \rho \cdot P^{t-1} \qquad \text{and thus} \qquad P^t(x,y) = \mathbf{P}\left[\, X_t = y \mid X_0 = x \,\right].$$

Thus $\rho^t(x) = (\mu P^t)(x)$ and so $\rho^t = \mu P^t = (\mu P^t(1), \mu P^t(2), \ldots, \mu P^t(n))$.

- 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 $X_0, \ldots, X_s$.

Example - College Carbs Stopping times:

✓ “We had rice yesterday” $\rightsquigarrow$ $\tau := \min\{t \geq 1 \colon X_{t-1} = \text{“rice”}\}$

× “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:

$$h(x, y) := \mathbf{E}_x[\tau_y] = \mathbf{E}\left[\, \tau_y \mid X_0 = x \,\right] \quad \text{where } \tau_y = \min\{t \geq 1 : X_t = y\}.$$

Some distinguish between $\tau_y^+ = \min\{t \geq 1 \colon X_t = y\}$ and $\tau_y = \min\{t \geq 0 \colon X_t = y\}$

A Useful Identity

Hitting times are the solution to a set of linear equations:

$$h(x, y) \overset{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \qquad \forall x \neq y \in \Omega.$$

Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

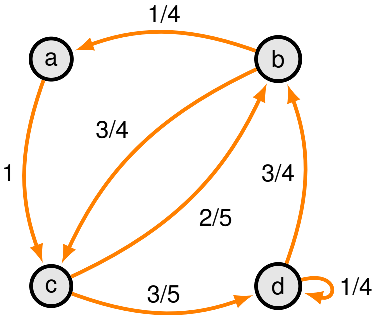
Total Variation Distance and Mixing Times

Application 1: Card Shuffling

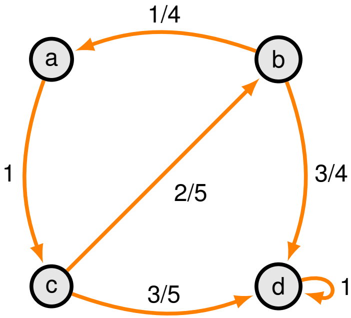
Application 2: Markov Chain Monte Carlo (non-examin.)

Irreducible Markov Chains

A Markov Chain is irreducible if for every state $x \in \Omega$ there is an integer $k \geq 0$ such that $P^k(x, x) > 0$.



✓ irreducible



× not-irreducible (thus reducible)

Finite Hitting Time Theorem

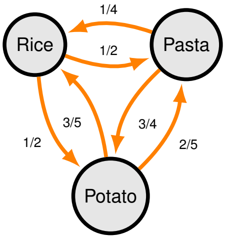
For any states x and y of a finite irreducible Markov Chain $h(x, y) < \infty$.

Stationary Distribution

A probability distribution $\pi = (\pi(1), \ldots, \pi(n))$ is the stationary distribution of a Markov Chain if $\pi P = \pi$ (π is a left eigenvector with eigenvalue 1)

College carbs example:

$$\underbrace{\left(\frac{4}{13}, \frac{4}{13}, \frac{5}{13}\right)}_{\pi} \cdot \underbrace{\begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 0 & 3/4 \\ 3/5 & 2/5 & 0 \end{pmatrix}}_{P} = \underbrace{\left(\frac{4}{13}, \frac{4}{13}, \frac{5}{13}\right)}_{\pi}$$



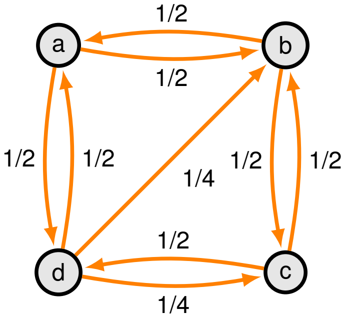
- A Markov Chain reaches stationary distribution if $\rho^t = \pi$ for some t.
- If reached, then it persists: If $\rho^t = \pi$ then $\rho^{t+k} = \pi$ for all $k \geq 0$.

Existence and Uniqueness of a Positive Stationary Distribution

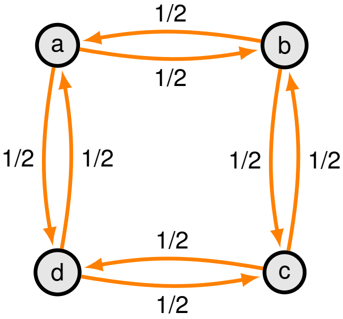
Let P be finite, irreducible M.C., then there exists a unique probability distribution π on Ω such that $\pi = \pi P$ and $\pi(x) = 1/h(x,x) > 0$, $\forall x \in \Omega$.

Periodicity

- A Markov Chain is aperiodic if for all $x \in \Omega$, $\mathrm{gcd}\{t \geq 1 : P^{t}_{x,x} > 0\} = 1$.
- Otherwise we say it is periodic.



✓ Aperiodic



× Periodic

Exercise: Which of the two chains (if any) are aperiodic?

Convergence Theorem

Ergodic = Irreducible + Aperiodic

Convergence Theorem

Let P be any finite, irreducible, aperiodic Markov Chain with stationary distribution π. Then for any $x, y \in \Omega$,

$$\lim_{t \to \infty} P^t_{x,y} = \pi_y.$$

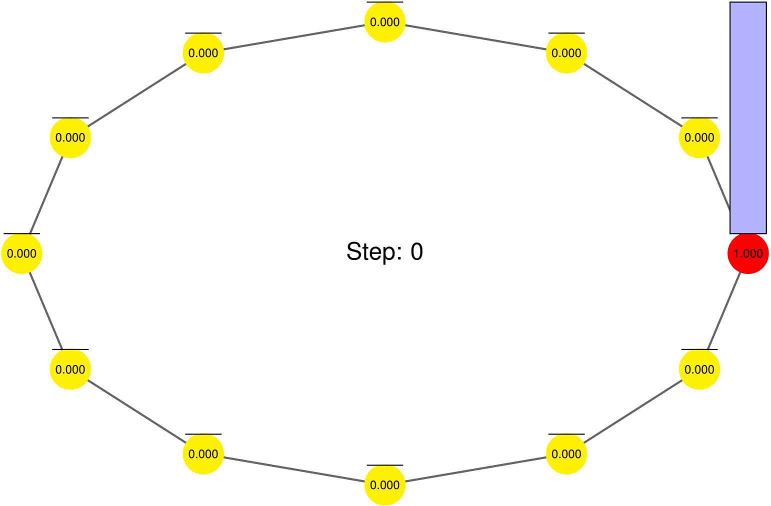
- mentioned before: For finite irreducible M.C.'s π exists, is unique and

$$\pi_y = \frac{1}{h(y,y)} > 0.$$

- We will prove a simpler version of the Convergence Theorem after introducing Spectral Graph Theory.

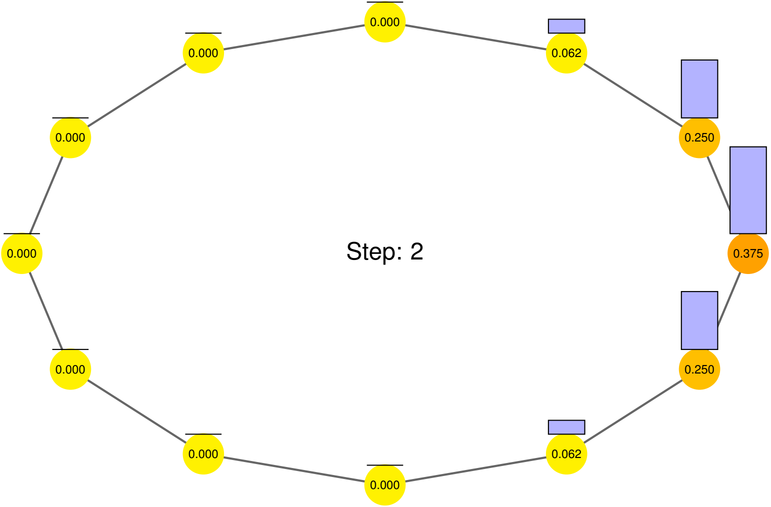
Convergence to Stationarity (Example)

- Markov Chain: stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \ldots, 12\}$ is $P^t(1, x)$.



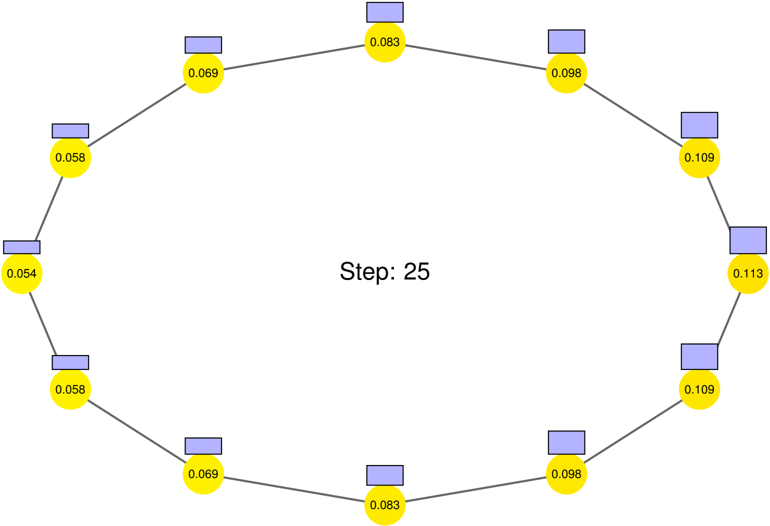
Convergence to Stationarity (Example)

- Markov Chain: stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \ldots, 12\}$ is $P^t(1, x)$.



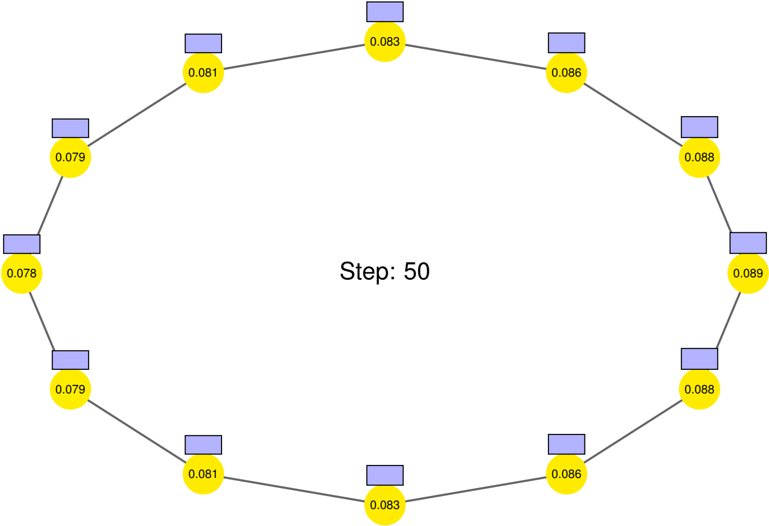
Convergence to Stationarity (Example)

- Markov Chain: stays put with $1/2$ and moves left (or right) w.p. $1/4$
- At step t the value at vertex $x \in \{1, 2, \ldots, 12\}$ 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 $x \in \{1, 2, \ldots, 12\}$ is $P^t(1, x)$.



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.)

How Similar are Two Probability Measures?

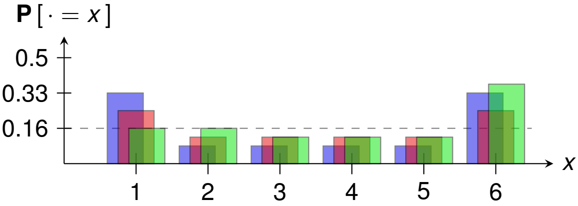
Loaded Dice

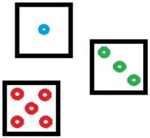
- You are presented three loaded (unfair) dice A, B, C:

x	1	2	3	4	5	6
$\mathbf{P}[A = x]$	1/3	1/12	1/12	1/12	1/12	1/3
$\mathbf{P}[B = x]$	1/4	1/8	1/8	1/8	1/8	1/4
$\mathbf{P}[C = x]$	1/6	1/6	1/8	1/8	1/8	9/24

- Question 1: Which dice is the least fair? Most of you 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

$$\|\mu - \eta\|_{tv} = \frac{1}{2} \sum_{\omega \in \Omega} |\mu(\omega) - \eta(\omega)|.$$

Loaded Dice: let $D = \mathit{Unif}\{1,2,3,4,5,6\}$ 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

$$\|D - B\|_{tv} = \|D - C\|_{tv} \qquad \text{and} \qquad \|D - B\|_{tv}, \|D - C\|_{tv} < \|D - A\|_{tv}.$$

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 μ be a prob. vector on Ω (might be just one vertex) and $t \geq 0$. Then

$$P_\mu^t := \mathbf{P}\left[X_t = \cdot \mid X_0 \sim \mu \right],$$

  is a probability measure on Ω.

- 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

$$\lim_{t \to \infty} \max_{x \in \Omega} \left\| P_x^t - \pi \right\|_{tv} = 0.$$

We will see a similar result later after introducing spectral techniques!

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

$$\tau_x(\epsilon) = \min \left\{ t \colon \left\| P_x^t - \pi \right\|_{tv} \leq \epsilon \right\},$$

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 $\varepsilon = 1/4$, indeed let $t_{mix} := \tau(1/4)$

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.)

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?

His research revealed beautiful connections between Markov Chains and Algebra.

Persi Diaconis (Professor of Statistics and former Magician)

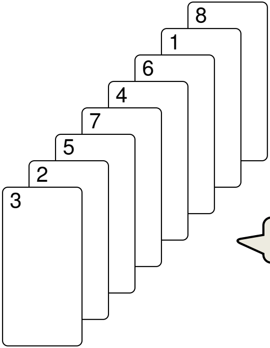
Source: `www.soundcloud.com`

The Card Shuffling Markov Chain

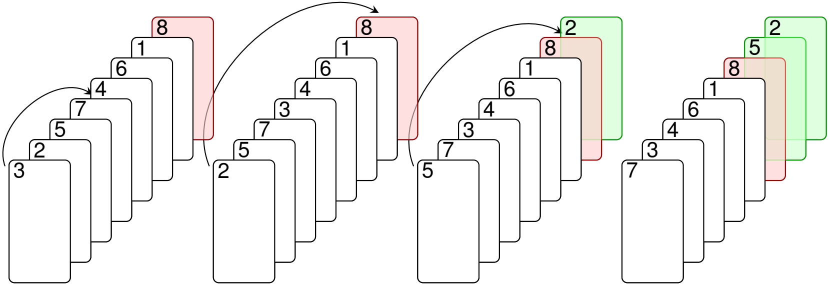
TOPTORANDOMSHUFFLE (Input: A pile of n cards)

```
For $t = 1, 2, \dots$
    Pick $i \in \{1, 2, \dots, n\}$ uniformly at random
    Take the top card and insert it behind the $i$-th card
```

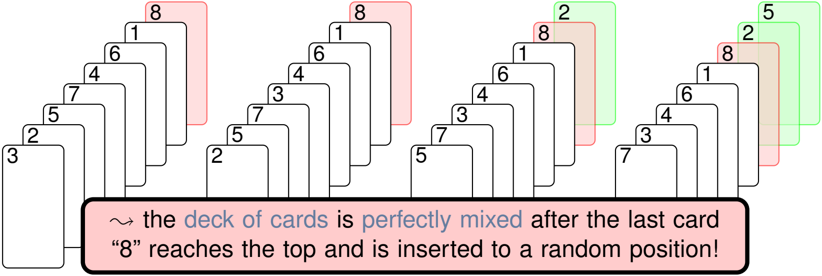
This is a slightly informal definition, so let us look at a small example...



We will focus on this “small” set of cards ($n = 8$)

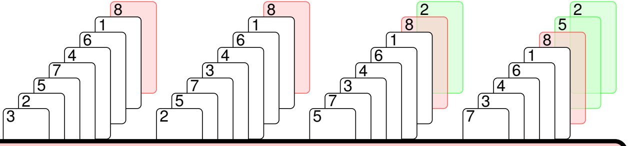


Even if we know which set of cards come after 8, every permutation is equally likely!



$\rightsquigarrow$ the 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)



$\rightsquigarrow$ 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)

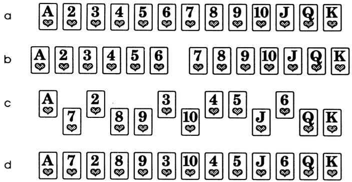
This is a **"reversed" coupon collector** process with n cards, which takes $n \log n$ in expectation.

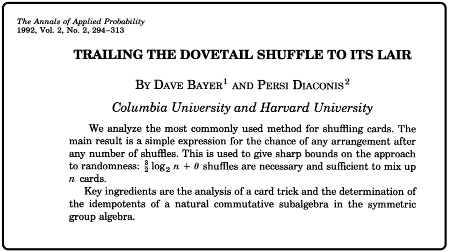
Using the so-called coupling method, one could prove $t_{mix} \leq n \log n$.

Analysis of 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 Bayer[1] and Persi Diaconis[2]

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

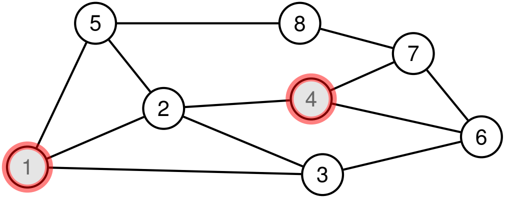
Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

Application 2: Markov Chain Monte Carlo (non-examin.)

A Markov Chain for Sampling Independent Sets (1/2)



$S = \{1, 4\}$ 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 $\{u, v\} \in E(G)$.

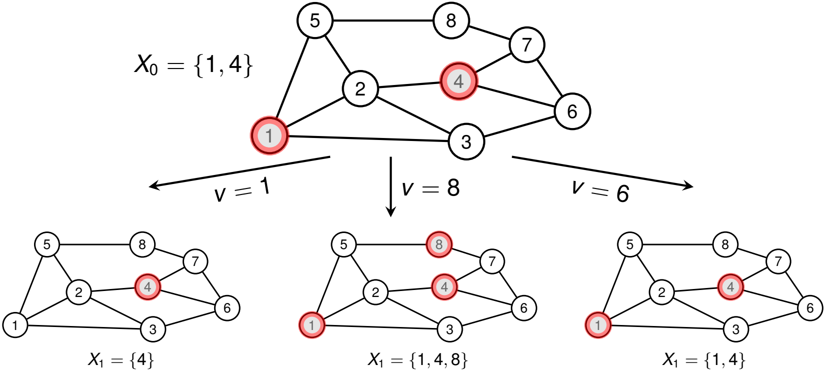
How can we take a sample from the space of all independent sets?

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!

A Markov Chain for Sampling Independent Sets (2/2)

```
INDEPENDENTSETSAMPLER
Let $X_0$ be an arbitrary independent set in $G$
For $t = 1, 2, \ldots$:
    Pick a vertex $v \in V(G)$ uniformly at random
    If $v \in X_t$ then $X_{t+1} \leftarrow X_t \setminus \{v\}$
    elif $v \notin X_t$ and $X_t \cup \{v\}$ is an independent set then $X_{t+1} \leftarrow X_t \cup \{v\}$
    else $X_{t+1} \leftarrow X_t$
```



A Markov Chain for Sampling Independent Sets (2/2)

```
IndependentSetSampler
Let $X_0$ be an arbitrary independent set in $G$
For $t = 1, 2, \ldots$:
    Pick a vertex $v \in V(G)$ uniformly at random
    If $v \in X_t$ then $X_{t+1} \leftarrow X_t \setminus \{v\}$
    elif $v \notin X_t$ and $X_t \cup \{v\}$ is an independent set then $X_{t+1} \leftarrow X_t \cup \{v\}$
    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

Randomised Algorithms

Lecture 5: Random Walks, Hitting Times and Application to 2-SAT

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

Outline

Application 2: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

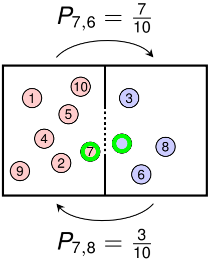
SAT and a Randomised Algorithm for 2-SAT

The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles labelled $1, 2, \ldots, d$
- At each step a particle is selected uniformly at random and switches to the other box
- If $\Omega = \{0, 1, \ldots, d\}$ denotes the number of particles in the red box, then:

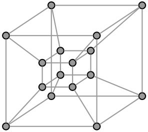
$$P_{x,x-1} = \frac{x}{d} \qquad \text{and} \qquad P_{x,x+1} = \frac{d-x}{d}.$$



Let us now enlarge the state space by looking at each particle individually!

Random Walk on the Hypercube

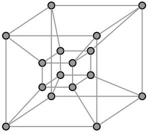
- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a random coordinate in $[d]$ and flip it



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



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)

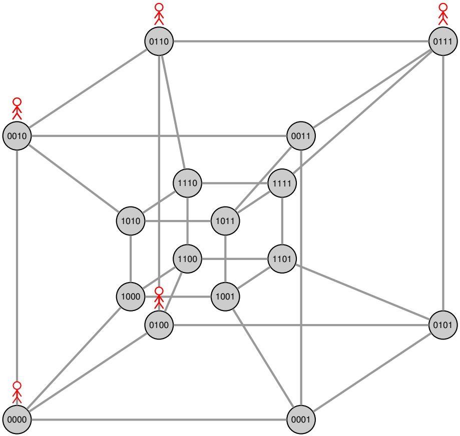
- At each step $t = 0, 1, 2 \ldots$
  - Pick a random coordinate in $[d]$
  - With prob. $1/2$ flip coordinate.

Lazy Random Walk (2nd Version)

- At each step $t = 0, 1, 2 \ldots$
  - 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



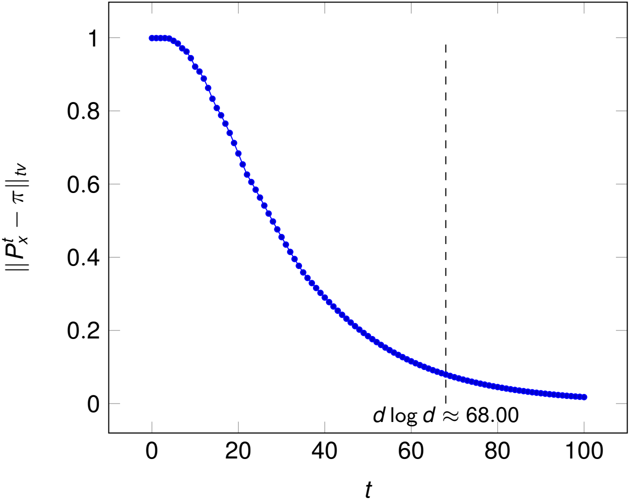
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9	1	0	0	1	0
10	done!	0	0	1	0

Once all coordinates have been picked at least once, the state is uniformly at random in $\{0,1\}^d$.

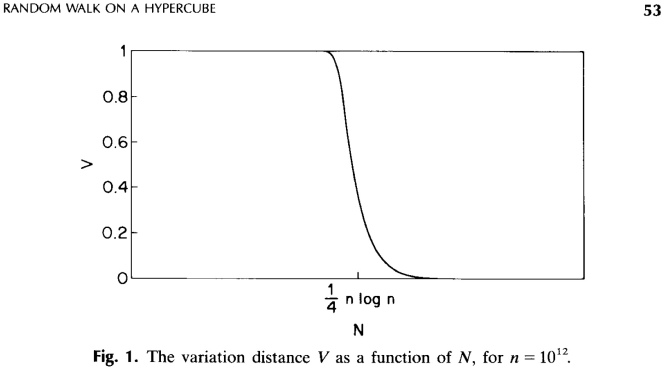
Coupon Collector $\rightsquigarrow$ mixing time should be $O(d \log d)$

We won't formalise this argument ($\exists$ related exercise question)

Total Variation Distance of Random Walk on Hypercube $(d = 22)$



Theoretical Results (by Diaconis, Graham and Morrison)



RANDOM WALK ON A HYPERCUBE 53

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 $\frac{1}{4} n \log n + \Theta(n)$

Outline

Application 2: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

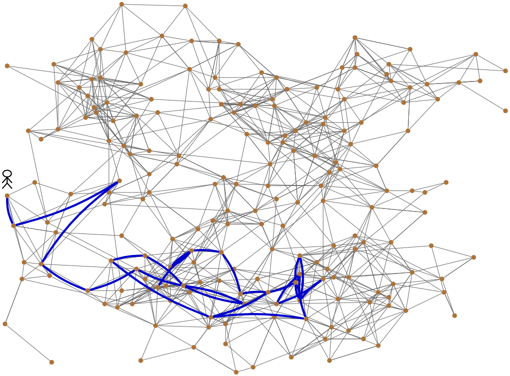
SAT and a Randomised Algorithm for 2-SAT

Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

$$P(u,v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u,v\} \in E, \\ 0 & \text{if } \{u,v\} \notin E. \end{cases}, \qquad \text{and} \qquad \pi(u) = \frac{\deg(u)}{2|E|}$$

Recall: $h(u,v) = \mathbf{E}_u[\min\{t \geq 1 : X_t = v\}]$ is the hitting time of v from u.



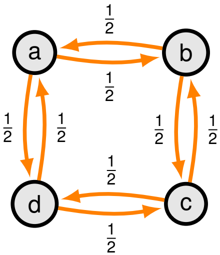
Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\widetilde{P} = (P + I)/2$,

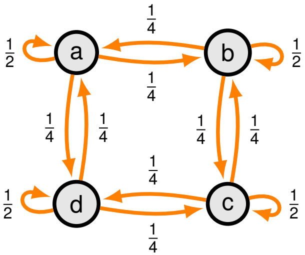
$$\widetilde{P}_{u,v} = \begin{cases} \frac{1}{2\deg(u)} & \text{if } \{u,v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise} \end{cases}.$$

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*

Outline

Application 2: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

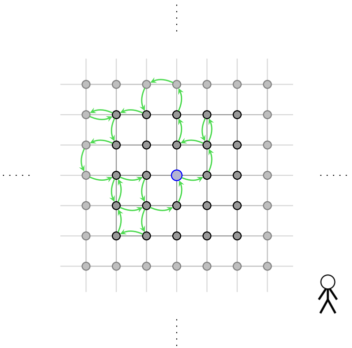
Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

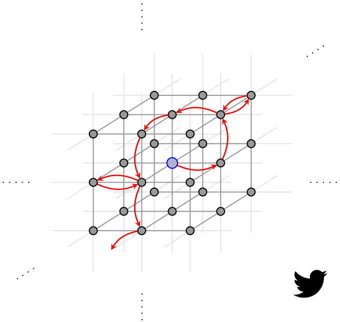
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

Infinite 2D Grid



Infinite 3D Grid



“A drunk man will find his way home, but a drunk bird may get lost forever.”

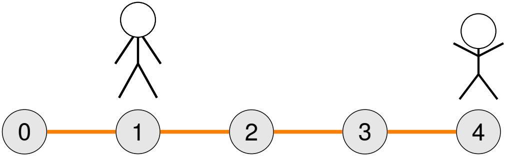
But for any regular (finite) graph, the expected return time to u is $1/\pi(u) = n$

SRW Random Walk on Two-Dimensional Grids: Animation

For animation, see full slides.

Random Walk on a Path (1/2)

The n-path P_n is the graph with $V(P_n) = [n]$ and $E(P_n) = \{\{i, j\} : j = i + 1\}$.



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 (2/2)

Proposition

For the SRW on P_n we have $h(k,n) = n^2 - k^2$, for any $0 \leq k \leq n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x,y) \overset{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} h(z,y) \cdot P(x,z) \qquad \forall x \neq y \in V.$$

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,

$$f(k) = 1 + \frac{n^2 - (k-1)^2}{2} + \frac{n^2 - (k+1)^2}{2} = n^2 - k^2. \qquad \square$$

Outline

Application 2: 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.

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:

$$\text{SAT:} \quad (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

$$\text{Solution:} \quad x_1 = \texttt{True}, \quad x_2 = \texttt{False}, \quad x_3 = \texttt{False} \quad \text{and} \quad x_4 = \texttt{True}.$$

- 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)

```
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"
```

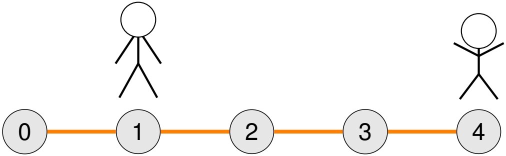
- Call each loop of (2) a step. Let A_i be the variable assignment at step i.
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

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

$$\alpha = (\mathtt{T}, \mathtt{T}, \mathtt{F}, \mathtt{T}).$$



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 clauses
4:     Choose a random literal and switch its value
5:     If formula is satisfied then return “Satisfiable”
6: return “Unsatisfiable”
```

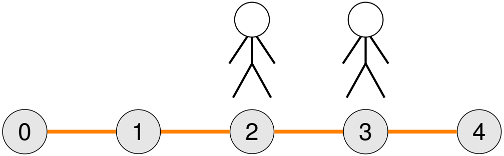
- Call each loop of (2) a step. Let A_i be the variable assignment at step i.
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

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

$$\alpha = (\mathtt{T}, \mathtt{F}, \mathtt{F}, \mathtt{T}).$$



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.

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 stochastic 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

$$\mathbf{E}[\text{time to find sol}] \leq \mathbf{E}_0[\min\{t : X_t = n\}] \leq \mathbf{E}_0[\min\{t : Y_t = n\}] = h(0,n) = n^2.$$

☐

Running for $2n^2$ time and using Markov's inequality yields:

Proposition

Provided a solution exists, RANDOMISED-2-SAT will return a valid solution in $O(n^2)$ time with probability at least $1/2$.

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 $O(n^2 \log n)$-time algorithm for 2-SAT which succeeds w.h.p.

Randomised Algorithms

Lecture 6: Linear Programming: Introduction

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

UNIVERSITY OF
CAMBRIDGE

Outline

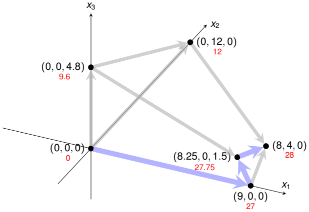
Introduction

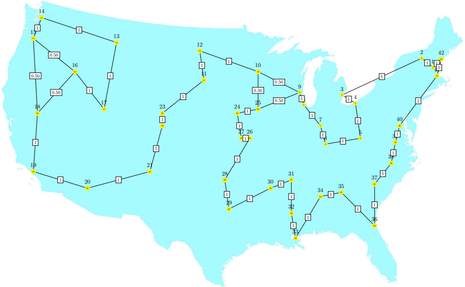
A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms

Introduction





- 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

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms

What are Linear Programs?

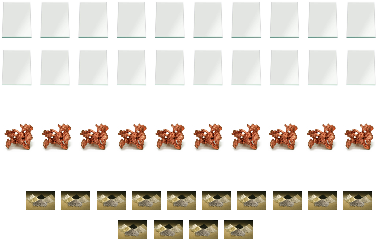
Linear Programming (informal definition)

- maximise or minimise an objective, given limited resources (competing constraint)
- constraints are specified as (in)equalities
- objective function and constraints are linear

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

$$\begin{array}{lrcrcr} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program

- Given $a_1, a_2, \ldots, a_n$ and a set of variables $x_1, x_2, \ldots, x_n$, a linear function f is defined by

$$f(x_1, x_2, \ldots, x_n) = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n.$$

- Linear Equality: $f(x_1, x_2, \ldots, x_n) = b$
- Linear Inequality: $f(x_1, x_2, \ldots, x_n) \overset{\geq}{\leq} b$

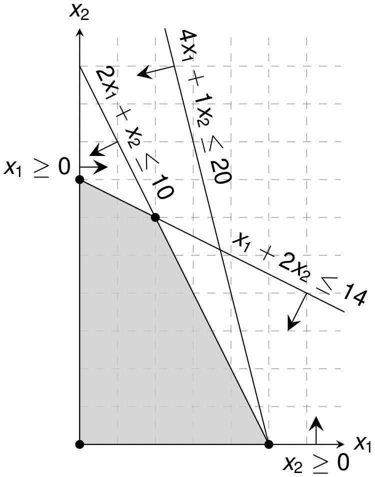
Linear Constraints

- Linear-Progamming Problem: either minimise or maximise a linear function subject to a set of linear constraints

Finding the Optimal Production Schedule

$$
\begin{array}{llcrcr}
\text{maximise} & x_1 & + & x_2 & & \\
\text{subject to} & & & & & \\
 & 4x_1 & + & x_2 & \leq & 20 \\
 & 2x_1 & + & x_2 & \leq & 10 \\
 & x_1 & + & 2x_2 & \leq & 14 \\
 & x_1\,, x_2 & & & \geq & 0
\end{array}
$$

Any setting of x_1 and x_2 satisfying all constraints is a feasible solution

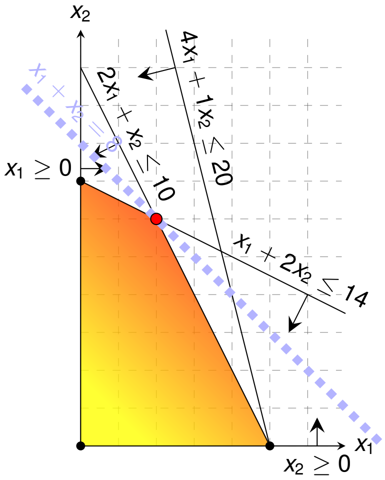


Question: Which aspect did we ignore in the formulation of the linear program?

Finding the Optimal Production Schedule

$$\begin{array}{lrcrcr} \text{maximise} & x_1 & + & x_2 & & \\ \text{subject to} & & & & & \\ & 4x_1 & + & x_2 & \leq & 20 \\ & 2x_1 & + & x_2 & \leq & 10 \\ & x_1 & + & 2x_2 & \leq & 14 \\ & x_1, x_2 & & & \geq & 0 \end{array}$$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.



While the same approach also works for higher-dimensions, we need to take a more systematic and algebraic procedure.

Outline

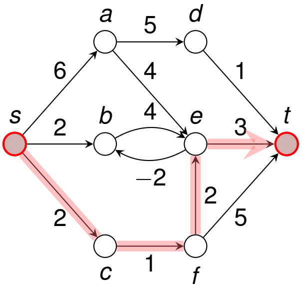
Introduction

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 $w : E \to \mathbb{R}$, pair of vertices $s, t \in V$
- Goal: Find a path of minimum weight from s to t in G

$p = (v_0 = s, v_1, \ldots, v_k = t)$ such that $w(p) = \sum_{i=1}^{k} w(v_{k-1}, v_k)$ is minimised.

Shortest Paths as LP

$$\begin{array}{lllll} \text{maximise} & d_t & & & \\ \text{subject to} & & & & \\ & d_v & \leq & d_u + w(u,v) & \text{for each edge } (u,v) \in E, \\ & d_s & = & 0. & \end{array}$$

Recall: When Bellman-Ford terminates, all these inequalities are satisfied.

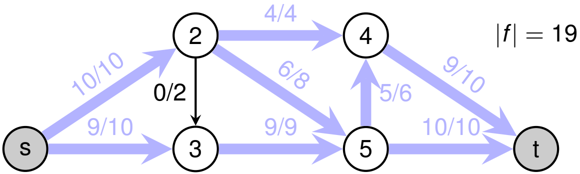
this is a maximisation problem!

Solution $\overline{d}$ satisfies $\overline{d}_v = \min_{u:\,(u,v)\in E} \left\{ \overline{d}_u + w(u,v) \right\}$

Maximum Flow

Maximum Flow Problem

- Given: directed graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$
- Goal: Find a maximum flow $f : V \times V \to \mathbb{R}$ from s to t which satisfies the capacity constraints and flow conservation



Maximum Flow as LP

$$\begin{array}{lrcll} \text{maximise} & \sum_{v \in V} f_{sv} & - & \sum_{v \in V} f_{vs} & \\ \text{subject to} & & & & \\ & f_{uv} & \leq & c(u, v) & \text{for each } u, v \in V, \\ & \sum_{v \in V} f_{vu} & = & \sum_{v \in V} f_{uv} & \text{for each } u \in V \setminus \{s, t\}, \\ & f_{uv} & \geq & 0 & \text{for each } u, v \in V. \end{array}$$

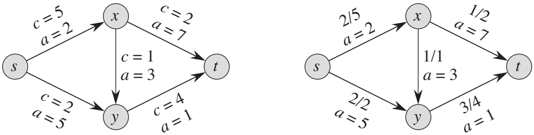
Minimum-Cost Flow

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem

- Given: directed graph $G = (V, E)$ with capacities $c : E \to \mathbb{R}^+$, pair of vertices $s, t \in V$, cost function $a : E \to \mathbb{R}^+$, flow demand of d units
- Goal: Find a flow $f : V \times V \to \mathbb{R}$ from s to t with $|f| = d$ while minimising the total cost $\sum_{(u,v) \in E} a(u,v) f_{uv}$ incurrred by the flow.

Optimal Solution with total cost:
$\sum_{(u,v) \in E} a(u,v) f_{uv} = (2\cdot2)+(5\cdot2)+(3\cdot1)+(7\cdot1)+(1\cdot3) = 27$



(a) (b)

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 a LP

Minimum Cost Flow as LP

minimise $\sum_{(u,v)\in E} a(u,v) f_{uv}$

subject to

$$\begin{array}{rcll} f_{uv} & \leq & c(u,v) & \text{for } u,v \in V, \\ \sum_{v\in V} f_{vu} - \sum_{v\in V} f_{uv} & = & 0 & \text{for } u \in V \setminus \{s,t\}, \\ \sum_{v\in V} f_{sv} - \sum_{v\in V} f_{vs} & = & d\ , & \\ f_{uv} & \geq & 0 & \text{for } u,v \in V. \end{array}$$

Real power of Linear Programming comes from the ability to solve **new problems**!

Outline

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms

Standard and Slack Forms

Standard Form

maximise $$\sum_{j=1}^{n} c_j x_j$$

Objective Function

subject to

$$\left\{ \begin{aligned} \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad & \text{for } i = 1, 2, \ldots, m \\ x_j \geq 0 \qquad & \text{for } j = 1, 2, \ldots, n \end{aligned} \right.$$

$n + m$ constraints

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 $\leq$).

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.

$$\begin{array}{lrcrcr}
\boxed{\text{minimise} \quad -2x_1 + 3x_2} & & & & & \\
\text{subject to} & & & & & \\
 & x_1 & + & x_2 & = & 7 \\
 & x_1 & - & 2x_2 & \leq & 4 \\
 & x_1 & & & \geq & 0
\end{array}$$

$$\Big\downarrow \quad \text{Negate objective function}$$

$$\begin{array}{lrcrcr}
\boxed{\text{maximise} \quad 2x_1 - 3x_2} & & & & & \\
\text{subject to} & & & & & \\
 & x_1 & + & x_2 & = & 7 \\
 & x_1 & - & 2x_2 & \leq & 4 \\
 & x_1 & & & \geq & 0
\end{array}$$

Converting into Standard Form (2/5)

Reasons for a LP not being in standard form:

2. There might be variables without nonnegativity constraints.

$$\begin{array}{llcccc}
\text{maximise} & 2x_1 & - & 3x_2 & & \\
\text{subject to} & & & & & \\
 & x_1 & + & x_2 & = & 7 \\
 & x_1 & - & 2x_2 & \leq & 4 \\
 & \boxed{x_1 \qquad\qquad \geq \quad 0} & & & &
\end{array}$$

Replace x_2 by two non-negative variables x_2' and x_2''

$$\begin{array}{llcccc}
\text{maximise} & 2x_1 & - & \boxed{3x_2' \quad + \quad 3x_2''} & & \\
\text{subject to} & & & & & \\
 & x_1 & + & \boxed{x_2' \quad - \quad x_2''} & = & 7 \\
 & x_1 & - & \boxed{2x_2' \quad + \quad 2x_2''} & \leq & 4 \\
 & \boxed{x_1, x_2', x_2'' \qquad\qquad \geq \quad 0} & & & &
\end{array}$$

Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

$$\begin{array}{ll} \text{maximise} & 2x_1 - 3x_2' + 3x_2'' \\ \text{subject to} & \\ & \boxed{x_1 + x_2' - x_2'' = 7} \\ & x_1 - 2x_2' + 2x_2'' \leq 4 \\ & x_1, x_2', x_2'' \geq 0 \end{array}$$

Replace each equality by two inequalities.

$$\begin{array}{ll} \text{maximise} & 2x_1 - 3x_2' + 3x_2'' \\ \text{subject to} & \\ & \boxed{\begin{array}{l} x_1 + x_2' - x_2'' \leq 7 \\ x_1 + x_2' - x_2'' \geq 7 \end{array}} \\ & x_1 - 2x_2' + 2x_2'' \leq 4 \\ & x_1, x_2', x_2'' \geq 0 \end{array}$$

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 $\leq$).

$$
\begin{array}{lrcrcrcr}
\text{maximise} & 2x_1 & - & 3x_2' & + & 3x_2'' & & \\
\text{subject to} & & & & & & & \\
 & x_1 & + & x_2' & - & x_2'' & \leq & 7 \\
 & \multicolumn{7}{c}{\boxed{x_1 + x_2' - x_2'' \geq 7}} \\
 & x_1 & - & 2x_2' & + & 2x_2'' & \leq & 4 \\
 & \multicolumn{5}{c}{x_1, x_2', x_2''} & \geq & 0
\end{array}
$$

$\downarrow$ Negate respective inequalities.

$$
\begin{array}{lrcrcrcr}
\text{maximise} & 2x_1 & - & 3x_2' & + & 3x_2'' & & \\
\text{subject to} & & & & & & & \\
 & x_1 & + & x_2' & - & x_2'' & \leq & 7 \\
 & \multicolumn{7}{c}{\boxed{-x_1 - x_2' + x_2'' \leq -7}} \\
 & x_1 & - & 2x_2' & + & 2x_2'' & \leq & 4 \\
 & \multicolumn{5}{c}{x_1, x_2', x_2''} & \geq & 0
\end{array}
$$

Converting into Standard Form (5/5)

Rename variable names (for consistency).

$$
\begin{array}{lrcrcrcr}
\text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & \\
\text{subject to} & & & & & & & \\
 & x_1 & + & x_2 & - & x_3 & \leq & 7 \\
 & -x_1 & - & x_2 & + & x_3 & \leq & -7 \\
 & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\
 & \multicolumn{5}{c}{x_1, x_2, x_3} & \geq & 0
\end{array}
$$

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.

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

$$s = b_i - \sum_{j=1}^{n} a_{ij}x_j$$

$$s \geq 0.$$

s measures the slack between the two sides of the inequality.

- Denote slack variable of the i-th inequality by x_{n+i}

Converting Standard Form into Slack Form (2/3)

$$\begin{array}{lrcrcrcr}
\text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & \\
\text{subject to} & & & & & & & \\
 & x_1 & + & x_2 & - & x_3 & \leq & 7 \\
 & -x_1 & - & x_2 & + & x_3 & \leq & -7 \\
 & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\
 & \multicolumn{5}{c}{x_1, x_2, x_3} & \geq & 0
\end{array}$$

Introduce slack variables

$$\begin{array}{lrcrcrcrcr}
\text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & & & \\
\text{subject to} & & & & & & & & & \\
 & x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
 & x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
 & x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \\
 & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0 & &
\end{array}$$

Converting Standard Form into Slack Form (3/3)

$$\begin{array}{llcrcrcrcr}
\text{maximise} & 2x_1 & - & 3x_2 & + & 3x_3 & & & & \\
\text{subject to} & & & & & & & & & \\
 & x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
 & x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
 & x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3 \\
 & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0 & &
\end{array}$$

Use variable z to denote objective function and omit the nonnegativity constraints.

$$\boxed{z = 2x_1 - 3x_2 + 3x_3}$$

$$\begin{array}{lcrcrcrcr}
x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3
\end{array}$$

This is called slack form.

Basic and Non-Basic Variables

$$\begin{array}{rcrcrcrcr}
z & = & & & 2x_1 & - & 3x_2 & + & 3x_3 \\
x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3
\end{array}$$

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

$$z = v + \sum_{j \in N} c_j x_j$$

$$x_i = b_i - \sum_{j \in N} a_{ij} x_j \qquad \text{for } i \in B,$$

and all variables are non-negative.

Variables/Coefficients on the right hand side are indexed by B and N.

Slack Form (Example)

$$\begin{array}{rcrcrcrcr} z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\ x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\ x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\ x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \end{array}$$

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 2023

Outline

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

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

$$\begin{array}{lrcrcrcr}
\text{maximise} & 3x_1 & + & x_2 & + & 2x_3 & & \\
\text{subject to} & & & & & & & \\
 & x_1 & + & x_2 & + & 3x_3 & \leq & 30 \\
 & 2x_1 & + & 2x_2 & + & 5x_3 & \leq & 24 \\
 & 4x_1 & + & x_2 & + & 2x_3 & \leq & 36 \\
 & & & x_1, x_2, x_3 & & & \geq & 0
\end{array}$$

Conversion into slack form

$$\begin{array}{rcrcrcrcr}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
\end{array}$$

Extended Example: Iteration 1

$$\begin{array}{rcrcrcrcr} z & = & & & 3x_1 & + & x_2 & + & 2x_3 \\ x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \end{array}$$

Basic solution: $(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_6}) = (0, 0, 0, 30, 24, 36)$

This basic solution is **feasible**

Objective value is 0.

.1

Extended Example: Iteration 1

Increasing the value of x_1 would increase the objective value.

$$\begin{array}{rcrcrcrcr}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
\end{array}$$

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

Increasing the value of x_3 would increase the objective value.

$$\begin{array}{rcrcrcrcr}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2}
\end{array}$$

Basic solution: $(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_6}) = (9, 0, 0, 21, 6, 0)$ with objective value 27

.3

Extended Example: Iteration 2

$$\begin{array}{rcrcccccc}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2}
\end{array}$$

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

Increasing the value of x_2 would increase the objective value.

$$\begin{array}{rclllllll}
z & = & \frac{111}{4} & + & \frac{x_2}{16} & - & \frac{x_5}{8} & - & \frac{11x_6}{16} \\
x_1 & = & \frac{33}{4} & - & \frac{x_2}{16} & + & \frac{x_5}{8} & - & \frac{5x_6}{16} \\
x_3 & = & \frac{3}{2} & - & \frac{3x_2}{8} & - & \frac{x_5}{4} & + & \frac{x_6}{8} \\
x_4 & = & \frac{69}{4} & + & \frac{3x_2}{16} & + & \frac{5x_5}{8} & - & \frac{x_6}{16}
\end{array}$$

Basic solution: $(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_6}) = (\frac{33}{4}, 0, \frac{3}{2}, \frac{69}{4}, 0, 0)$ with objective value $\frac{111}{4} = 27.75$

Extended Example: Iteration 3

$$z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16}$$

$$x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16}$$

$$\boxed{x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8}}$$

$$x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}$$

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

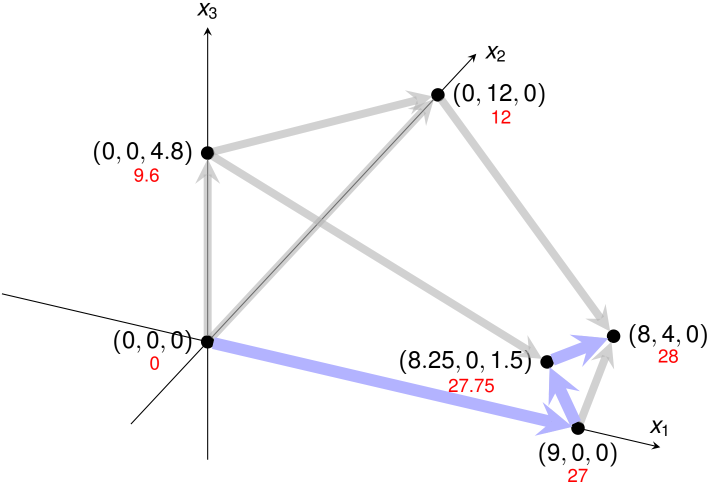
Extended Example: Iteration 4

All coefficients are negative, and hence this basic solution is **optimal**!

$$\begin{array}{rcrcrcrcr}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\
x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & &
\end{array}$$

Basic solution: $(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_6}) = (8, 4, 0, 18, 0, 0)$ with objective value 28

Extended Example: Visualization of SIMPLEX



Exercise: How many basic solutions (including non-feasible ones) are there?

Extended Example: Alternative Runs (1/2)

$$\begin{array}{rcrcrcrcr}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
\end{array}$$

Switch roles of x_2 and x_5

$$\begin{array}{rcrcrcrcr}
z & = & 12 & + & 2x_1 & - & \frac{x_3}{2} & - & \frac{x_5}{2} \\
x_2 & = & 12 & - & x_1 & - & \frac{5x_3}{2} & - & \frac{x_5}{2} \\
x_4 & = & 18 & - & x_2 & - & \frac{x_3}{2} & + & \frac{x_5}{2} \\
x_6 & = & 24 & - & 3x_1 & + & \frac{x_3}{2} & + & \frac{x_5}{2}
\end{array}$$

Switch roles of x_1 and x_6

$$\begin{array}{rcrcrcrcr}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \\
x_1 & = & 8 & + & \frac{x_3}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \\
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \\
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & &
\end{array}$$

Extended Example: Alternative Runs (2/2)

$$
\begin{aligned}
z &= && 3x_1 + x_2 + 2x_3 \\
x_4 &= 30 &&- x_1 - x_2 - 3x_3 \\
x_5 &= 24 &&- 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 &&- 4x_1 - x_2 - 2x_3
\end{aligned}
$$

Switch roles of x_3 and x_5

$$
\begin{aligned}
z &= \frac{48}{5} + \frac{11x_1}{5} + \frac{x_2}{5} - \frac{2x_5}{5} \\
x_4 &= \frac{78}{5} + \frac{x_1}{5} + \frac{x_2}{5} + \frac{3x_5}{5} \\
x_3 &= \frac{24}{5} - \frac{2x_1}{5} - \frac{2x_2}{5} - \frac{x_5}{5} \\
x_6 &= \frac{132}{5} - \frac{16x_1}{5} - \frac{x_2}{5} + \frac{2x_3}{5}
\end{aligned}
$$

Switch roles of x_1 and x_6

$$
\begin{aligned}
z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\
x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\
x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\
x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}
\end{aligned}
$$

Switch roles of x_2 and x_3

$$
\begin{aligned}
z &= 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \\
x_1 &= 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \\
x_2 &= 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \\
x_4 &= 18 - \frac{x_3}{2} + \frac{x_5}{2}
\end{aligned}
$$

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$(N, B, A, b, c, \nu, l, e)$
// Compute the coefficients of the equation for new basic variable $x_e$.
let $\widehat{A}$ be a new $m \times n$ matrix
$\widehat{b}_e = b_l / a_{le}$
for each $j \in N - \{e\}$
    $\widehat{a}_{ej} = a_{lj} / a_{le}$
$\widehat{a}_{el} = 1 / a_{le}$
// Compute the coefficients of the remaining constraints.
for each $i \in B - \{l\}$
    $\widehat{b}_i = b_i - a_{ie}\widehat{b}_e$
    for each $j \in N - \{e\}$
        $\widehat{a}_{ij} = a_{ij} - a_{ie}\widehat{a}_{ej}$
    $\widehat{a}_{il} = -a_{ie}\widehat{a}_{el}$
// Compute the objective function.
$\widehat{\nu} = \nu + c_e\widehat{b}_e$
for each $j \in N - \{e\}$
    $\widehat{c}_j = c_j - c_e\widehat{a}_{ej}$
$\widehat{c}_l = -c_e\widehat{a}_{el}$
// Compute new sets of basic and nonbasic variables.
$\widehat{N} = N - \{e\} \cup \{l\}$
$\widehat{B} = B - \{l\} \cup \{e\}$
return $(\widehat{N}, \widehat{B}, \widehat{A}, \widehat{b}, \widehat{c}, \widehat{\nu})$
```

Need that $a_{le} \neq 0$!

Rewrite “tight” equation for enterring 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 $\textsc{Pivot}(N, B, A, b, c, v, l, e)$ in which $a_{le} \neq 0$. Let the values returned from the call be $(\widehat{N}, \widehat{B}, \widehat{A}, \widehat{b}, \widehat{c}, \widehat{v})$, and let $\overline{x}$ denote the basic solution after the call. Then

1. $\overline{x}_j = 0$ for each $j \in \widehat{N}$.
2. $\overline{x}_e = b_l / a_{le}$.
3. $\overline{x}_i = b_i - a_{ie}\widehat{b}_e$ for each $i \in \widehat{B} \setminus \{e\}$.

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

$$x_i = \widehat{b}_i - \sum_{j \in \widehat{N}} \widehat{a}_{ij} x_j,$$

   we have $\overline{x}_i = \widehat{b}_i$ for each $i \in \widehat{B}$. Hence $\overline{x}_e = \widehat{b}_e = b_l / a_{le}$.
3. After substituting into the other constraints, we have

$$\overline{x}_i = \widehat{b}_i = b_i - a_{ie}\widehat{b}_e. \qquad \square$$

Formalizing the Simplex Algorithm: Questions

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 $(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$
```

Returns a slack form with a feasible basic solution (if it exists)

Main Loop:

- terminates if all coefficients in objective function are negative
- Line 4 picks enterring variable x_e with negative 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"
```

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

$$\begin{array}{llcrcr}
\text{maximise} & 2x_1 & - & x_2 & & \\
\text{subject to} & & & & & \\
& 2x_1 & - & x_2 & \leq & 2 \\
& x_1 & - & 5x_2 & \leq & -4 \\
& & x_1, x_2 & & \geq & 0
\end{array}$$

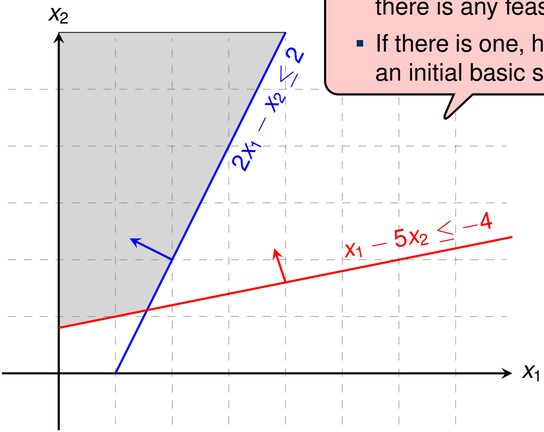
Conversion into slack form

$$\begin{array}{rcrcrcr}
z & = & & & 2x_1 & - & x_2 \\
x_3 & = & 2 & - & 2x_1 & + & x_2 \\
x_4 & = & -4 & - & x_1 & + & 5x_2
\end{array}$$

Basic solution $(x_1, x_2, x_3, x_4) = (0, 0, 2, -4)$ is not feasible!

Geometric Illustration

$$\begin{array}{llcrcr} \text{maximise} & 2x_1 & - & x_2 & & \\ \text{subject to} & & & & & \\ & 2x_1 & - & x_2 & \leq & 2 \\ & x_1 & - & 5x_2 & \leq & -4 \\ & & x_1, x_2 & & \geq & 0 \end{array}$$



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

$$\begin{array}{ll} \text{maximise} & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} & \\ & \begin{array}{rcll} \sum_{j=1}^{n} a_{ij} x_j & \leq & b_i & \text{for } i = 1, 2, \ldots, m, \\ x_j & \geq & 0 & \text{for } j = 1, 2, \ldots, n \end{array} \end{array}$$

↓ Formulating an Auxiliary Linear Program

$$\begin{array}{ll} \text{maximise} & -x_0 \\ \text{subject to} & \\ & \begin{array}{rcll} \sum_{j=1}^{n} a_{ij} x_j - x_0 & \leq & b_i & \text{for } i = 1, 2, \ldots, m, \\ x_j & \geq & 0 & \text{for } j = 0, 1, \ldots, n \end{array} \end{array}$$

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.

- "$\Rightarrow$": Suppose L has a feasible solution $\overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$
    - $\overline{x}_0 = 0$ combined with $\overline{x}$ is a feasible solution to L_{aux} with objective value 0.
    - Since $\overline{x}_0 \geq 0$ and the objective is to maximise $-x_0$, this is optimal for L_{aux}
- "$\Leftarrow$": Suppose that the optimal objective value of L_{aux} is 0
    - Then $\overline{x}_0 = 0$, and the remaining solution values $(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$ satisfy L. □

- 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

$$\begin{array}{ll} \text{maximise} & -x_0 \\ \text{subject to} & \\ & \begin{array}{rcrcrcr} 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\ x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\ & x_0, x_1, x_2 & & & & \geq & 0 \end{array} \end{array}$$

For the animation see the full slides.

Now the Feasible Region of the Auxiliary LP in 3D

- 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

$$\begin{array}{ll} \text{maximise} & -x_0 \\ \text{subject to} & \\ & \begin{array}{rcrcrcr} 2x_1 & - & x_2 & - & x_0 & \leq & -2 \\ -x_1 & + & 5x_2 & - & x_0 & \leq & 4 \\ & & x_0, x_1, x_2 & & & \geq & 0 \end{array} \end{array}$$

For the animation see the full slides.

INITIALIZE-SIMPLEX

```
INITIALIZE-SIMPLEX$(A, b, c)$
let $k$ be the index of the minimum $b_i$
if $b_k \geq 0$                    // is the initial basic solution feasible?
    return $(\{1, 2, \ldots, n\}, \{n+1, n+2, \ldots, n+m\}, A, b, c, 0)$
form $L_{\text{aux}}$ by adding $-x_0$ to the left-hand side of each constraint
    and setting the objective function to $-x_0$
let $(N, B, A, b, c, \nu)$ be the resulting slack form for $L_{\text{aux}}$
$l = n + k$
// $L_{\text{aux}}$ has $n + 1$ nonbasic variables and $m$ basic variables.
$(N, B, A, b, c, \nu) = $ PIVOT$(N, B, A, b, c, \nu, l, 0)$
// The basic solution is now feasible for $L_{\text{aux}}$.
iterate the while loop of lines 3–12 of SIMPLEX until an optimal solution
    to $L_{\text{aux}}$ is found
if the optimal solution to $L_{\text{aux}}$ sets $\bar{x}_0$ to 0
    if $\bar{x}_0$ is basic
        perform one (degenerate) pivot to make it nonbasic
    from the final slack form of $L_{\text{aux}}$, 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
    return the modified final slack form
else return "infeasible"
```

Test solution with $N = \{1, 2, \ldots, n\}$, $B = \{n+1, n+2, \ldots, n+m\}$, $\overline{x}_i = b_i$ for $i \in B$, $\overline{x}_i = 0$ 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)

$$\begin{array}{lrcrcr}
\text{maximise} & 2x_1 & - & x_2 & & \\
\text{subject to} & & & & & \\
 & 2x_1 & - & x_2 & \leq & 2 \\
 & x_1 & - & 5x_2 & \leq & -4 \\
 & & x_1, x_2 & & \geq & 0
\end{array}$$

Formulating the auxiliary linear program

$$\begin{array}{lrcrcrcr}
\text{maximise} & & & & - & x_0 & & \\
\text{subject to} & & & & & & & \\
 & 2x_1 & - & x_2 & - & x_0 & \leq & 2 \\
 & x_1 & - & 5x_2 & - & x_0 & \leq & -4 \\
 & & x_1, x_2, x_0 & & & & \geq & 0
\end{array}$$

Basic solution $(0, 0, 0, 2, -4)$ not feasible!

Converting into slack form

$$\begin{array}{rcrcrcrcr}
z & = & & & & & & - & x_0 \\
x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\
x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0
\end{array}$$

Example of INITIALIZE-SIMPLEX (2/3)

$$\begin{array}{rcrcrcrcr} z & = & & & & & & - & x_0 \\ x_3 & = & 2 & - & 2x_1 & + & x_2 & + & x_0 \\ x_4 & = & -4 & - & x_1 & + & 5x_2 & + & x_0 \end{array}$$

Pivot with x_0 entering and x_4 leaving

$$\begin{array}{rcrcrcrcr} z & = & -4 & - & x_1 & + & 5x_2 & - & x_4 \\ x_0 & = & 4 & + & x_1 & - & 5x_2 & + & x_4 \\ x_3 & = & 6 & - & x_1 & - & 4x_2 & + & x_4 \end{array}$$

Basic solution $(4, 0, 0, 6, 0)$ is feasible!

Pivot with x_2 entering and x_0 leaving

$$\begin{array}{rcrcrcrcr} z & = & & - & x_0 & & & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

Optimal solution has $x_0 = 0$, hence the initial problem was feasible!

Example of INITIALIZE-SIMPLEX (3/3)

$$\begin{array}{rclllll} z & = & & - & x_0 & & \\ x_2 & = & \frac{4}{5} & - & \frac{x_0}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & + & \frac{4x_0}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

$2x_1 - x_2 = 2x_1 - (\frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5})$

Set $x_0 = 0$ and express objective function by non-basic variables

$$\begin{array}{rclllll} z & = & -\frac{4}{5} & + & \frac{9x_1}{5} & - & \frac{x_4}{5} \\ x_2 & = & \frac{4}{5} & + & \frac{x_1}{5} & + & \frac{x_4}{5} \\ x_3 & = & \frac{14}{5} & - & \frac{9x_1}{5} & + & \frac{x_4}{5} \end{array}$$

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)

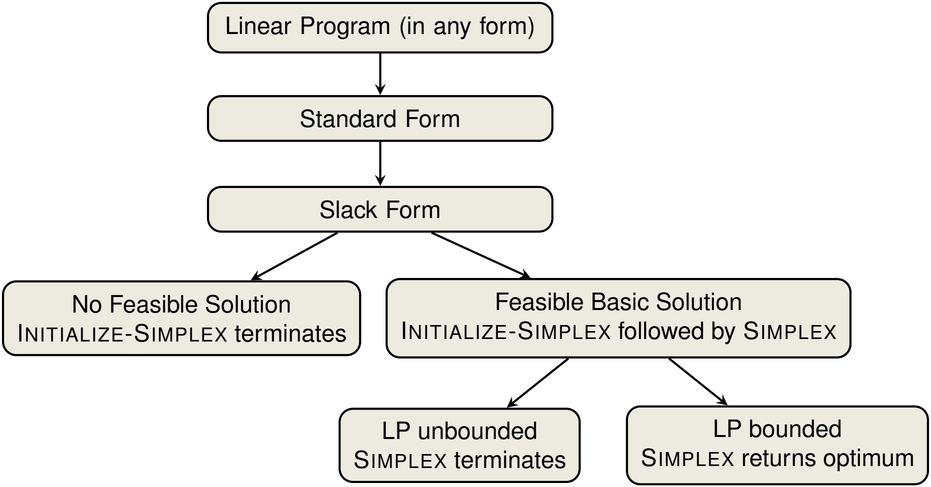
Any linear program L, given in standard form, either

1. has an optimal solution with a finite objective value,
2. is infeasible, or
3. is unbounded.

If L is infeasible, SIMPLEX returns "infeasible". If L is unbounded, SIMPLEX returns "unbounded". Otherwise, SIMPLEX returns an optimal solution with a finite objective value.

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 Program (in any form)

Standard Form

Slack Form

No Feasible Solution
INITIALIZE-SIMPLEX terminates

Feasible Basic Solution
INITIALIZE-SIMPLEX followed by SIMPLEX

LP unbounded
SIMPLEX terminates

LP bounded
SIMPLEX returns optimum

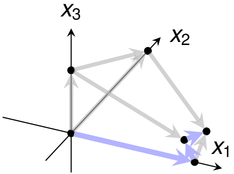
Linear Programming and Simplex: Summary and Outlook

Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of Integer Programming, to be discussed in later lectures

Simplex Algorithm

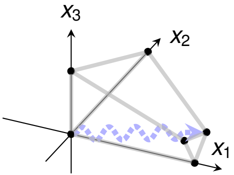
- In practice: usually terminates in polynomial time, i.e., $O(m + n)$
- In theory: even with anti-cycling may need exponential time



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!)



Outline

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.

$$\begin{array}{rcrcrcrcr} z & = & & & x_1 & + & x_2 & + & x_3 \\ x_4 & = & 8 & - & x_1 & - & x_2 & & \\ x_5 & = & & & & & x_2 & - & x_3 \end{array}$$

Pivot with x_1 entering and x_4 leaving

$$\begin{array}{rcrcrcrcr} z & = & 8 & & & + & x_3 & - & x_4 \\ x_1 & = & 8 & - & x_2 & & & - & x_4 \\ x_5 & = & & & x_2 & - & x_3 & & \end{array}$$

Pivot with x_3 entering and x_5 leaving

Cycling: If additionally slack form at two iterations are identical, SIMPLEX fails to terminate!

$$\begin{array}{rcrcrcrcr} z & = & 8 & + & x_2 & - & x_4 & - & x_5 \\ x_1 & = & 8 & - & x_2 & - & x_4 & & \\ x_3 & = & & & x_2 & & & - & x_5 \end{array}$$

Exercise: Execute one more step of the Simplex Algorithm on the tableau from the previous slide.

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 $\widehat{b}_i = b_i + \epsilon_i$, 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 2023

UNIVERSITY OF CAMBRIDGE

Outline

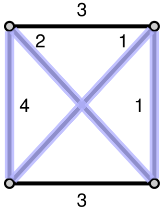
Introduction

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.



$2 + 4 + 1 + 1 = 8$

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

- Metric TSP: costs satisfy triangle inequality:

$$\forall u, v, w \in V: \qquad c(u, w) \leq c(u, v) + c(v, w).$$

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 Toody and Muldoon find the shortest round trip route to visit all 33 locations shown on the map.

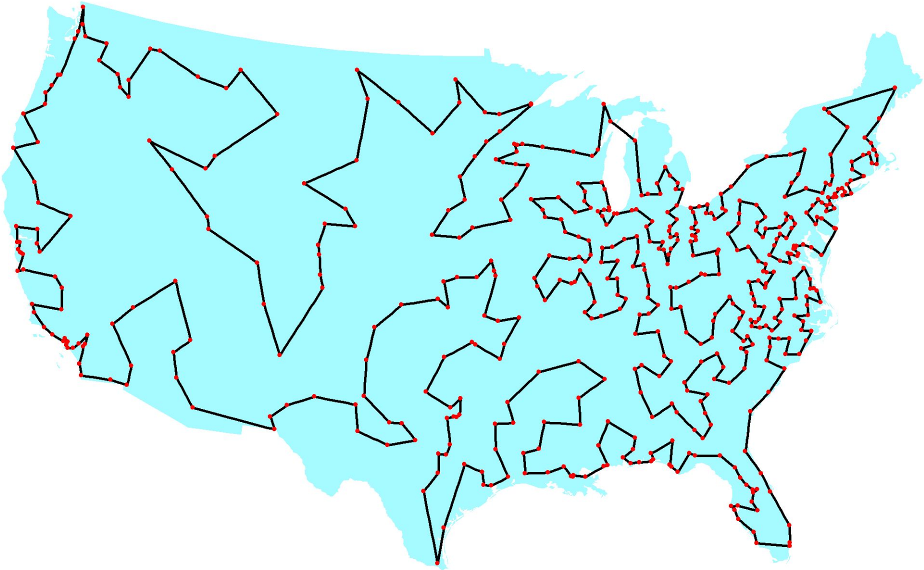
All you do is draw connecting straight lines from location to location to show the shortest round trip route.

HERE'S THE CORRECT START...

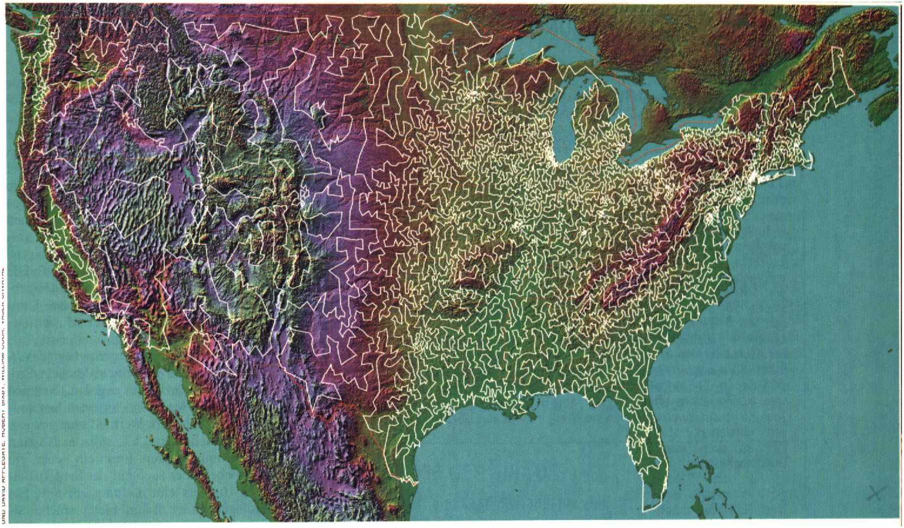
Begin at Chicago, Illinois. From there, lines show correct route as far as Erie, Pennsylvania. Next, do you go to Carlisle, Pennsylvania or Wana, West Virginia? Check the easy instructions on back of this entry blank for details.

OFFICIAL RULES ON REVERSE SIDE

532 cities (1987 [Padberg, Rinaldi])

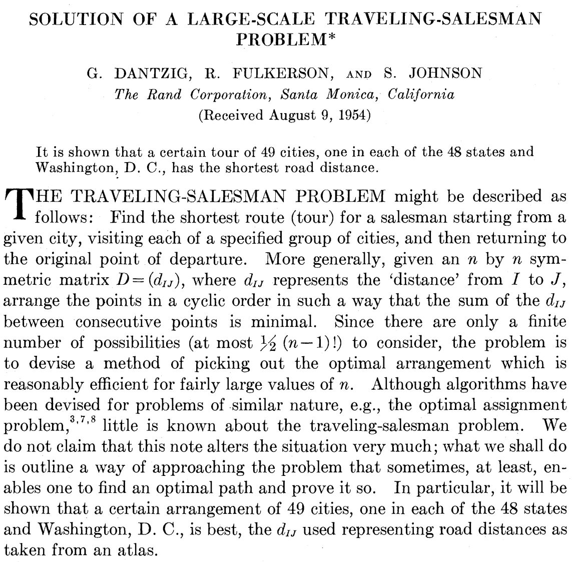


13,509 cities (1999 [Applegate, Bixby, Chavatal, Cook])



UND DAVID APPLEGATE, ROBERT BIXBY, WILLIAM COOK, VASEK CHVATAL

The Original Article (1954)



SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM*

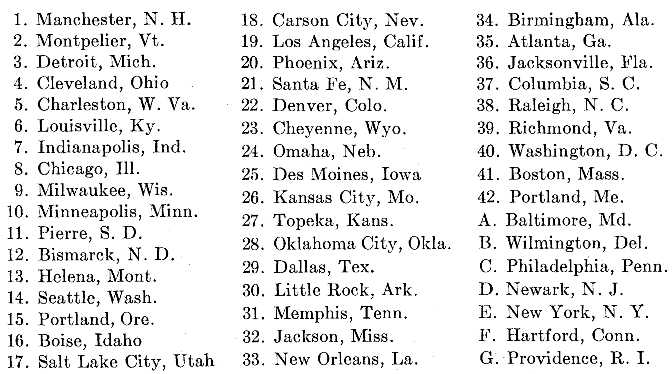
G. DANTZIG, R. FULKERSON, AND S. JOHNSON
The Rand Corporation, Santa Monica, California

(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.

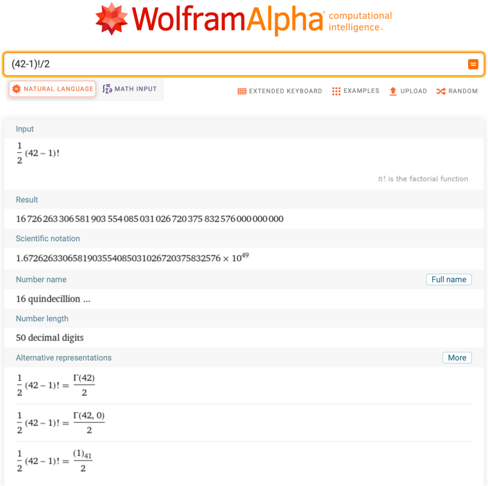
The 42 (49) Cities



1. Manchester, N. H.
2. Montpelier, Vt.
3. Detroit, Mich.
4. Cleveland, Ohio
5. Charleston, W. Va.
6. Louisville, Ky.
7. Indianapolis, Ind.
8. Chicago, Ill.
9. Milwaukee, Wis.
10. Minneapolis, Minn.
11. Pierre, S. D.
12. Bismarck, N. D.
13. Helena, Mont.
14. Seattle, Wash.
15. Portland, Ore.
16. Boise, Idaho
17. Salt Lake City, Utah
18. Carson City, Nev.
19. Los Angeles, Calif.
20. Phoenix, Ariz.
21. Santa Fe, N. M.
22. Denver, Colo.
23. Cheyenne, Wyo.
24. Omaha, Neb.
25. Des Moines, Iowa
26. Kansas City, Mo.
27. Topeka, Kans.
28. Oklahoma City, Okla.
29. Dallas, Tex.
30. Little Rock, Ark.
31. Memphis, Tenn.
32. Jackson, Miss.
33. New Orleans, La.
34. Birmingham, Ala.
35. Atlanta, Ga.
36. Jacksonville, Fla.
37. Columbia, S. C.
38. Raleigh, N. C.
39. Richmond, Va.
40. Washington, D. C.
41. Boston, Mass.
42. Portland, Me.

A. Baltimore, Md.
B. Wilmington, Del.
C. Philadelphia, Penn.
D. Newark, N. J.
E. New York, N. Y.
F. Hartford, Conn.
G. Providence, R. I.

Combinatorial Explosion



(42-1)!/2

Input

$\frac{1}{2}(42-1)!$

$n!$ is the factorial function

Result

16 726 263 306 581 903 554 085 031 026 720 375 832 576 000 000 000

Scientific notation

$1.6726263306581903554085031026720375832576 \times 10^{49}$

Number name

16 quindecillion ...

Number length

50 decimal digits

Alternative representations

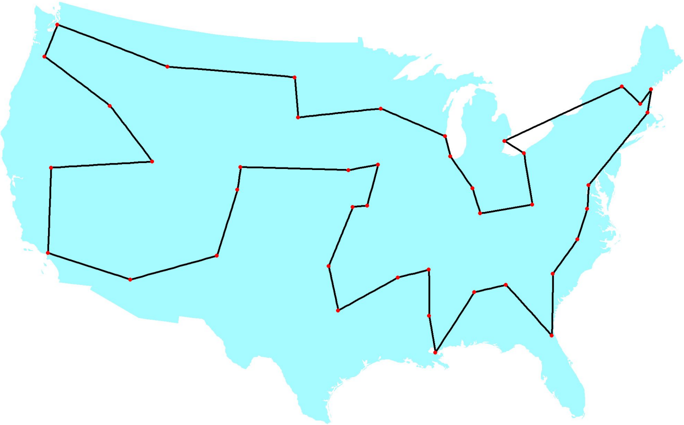
$$\frac{1}{2}(42-1)! = \frac{\Gamma(42)}{2}$$

$$\frac{1}{2}(42-1)! = \frac{\Gamma(42, 0)}{2}$$

$$\frac{1}{2}(42-1)! = \frac{(1)_{41}}{2}$$

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

Hence this is an instance of the Metric TSP, but not Euclidean TSP.

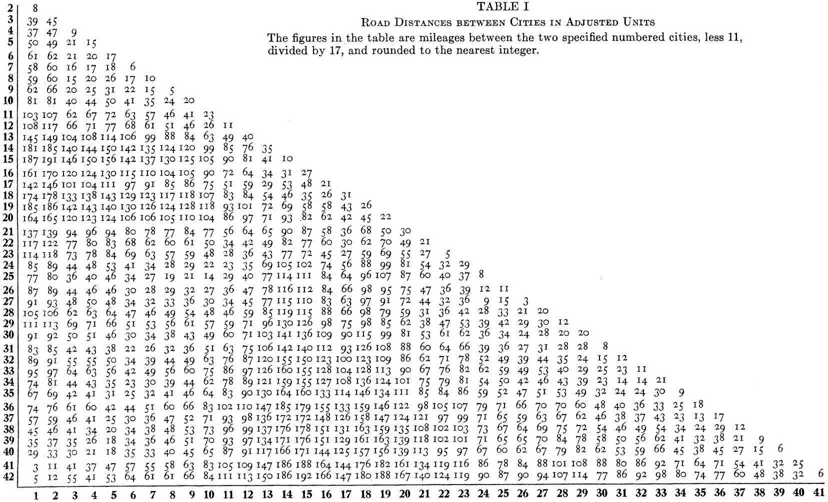


TABLE I

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.

2	8																																								
3	39	45																																							
4	37	47	9																																						
5	50	49	21	15																																					
6	61	62	21	20	17																																				
7	58	60	16	17	18	6																																			
8	59	60	15	20	26	17	10																																		
9	62	66	20	25	31	22	15	5																																	
10	81	81	40	44	50	41	35	24	20																																
11	103	107	62	67	72	63	57	46	41	23																															
12	108	117	66	71	77	68	61	51	46	26	11																														
13	145	149	104	108	114	106	99	88	84	63	49	40																													
14	181	185	140	144	150	142	135	124	120	99	85	76	35																												
15	187	191	146	150	156	142	137	130	125	105	90	81	41	10																											
16	161	170	120	124	130	115	110	104	105	90	72	64	34	31	27																										
17	142	146	101	104	111	97	91	85	86	75	51	59	29	53	48	21																									
18	174	178	133	138	143	129	123	117	118	107	83	84	54	46	35	26	31																								
19	185	186	142	143	140	130	126	124	128	118	93	101	72	69	58	58	43	26																							
20	164	165	120	123	124	106	106	105	110	104	86	97	71	93	82	62	42	45	22																						
21	137	139	94	96	94	80	78	77	84	77	56	64	65	90	87	58	36	68	50	30																					
22	117	122	77	80	83	68	62	60	61	50	34	42	49	82	77	60	30	62	70	49	21																				
23	114	118	73	78	84	69	63	57	59	48	28	36	43	77	72	45	27	59	69	55	27	5																			
24	85	89	44	48	53	41	34	28	29	22	23	35	69	105	102	74	56	88	99	81	54	32	29																		
25	77	80	36	40	46	34	27	19	21	14	29	40	77	114	111	84	64	96	107	87	60	40	37	8																	
26	87	89	44	46	46	30	28	29	32	27	36	47	78	116	112	84	66	98	95	75	47	36	39	12	11																
27	91	93	48	50	48	34	32	33	36	30	34	45	77	115	110	83	63	97	91	72	44	32	36	9	15	3															
28	105	106	62	63	64	47	46	49	54	48	46	59	85	119	115	88	66	98	79	59	31	36	42	28	33	21	20														
29	111	113	69	71	66	51	53	56	61	57	59	71	96	130	126	98	75	98	85	62	38	47	53	39	42	29	30	12													
30	91	92	50	51	46	30	34	38	43	49	60	71	103	141	136	109	90	115	99	81	53	61	62	36	34	24	28	20	20												
31	83	85	42	43	38	22	26	32	36	51	63	75	106	142	140	112	93	126	108	88	60	64	66	39	36	27	31	28	28	8											
32	89	91	55	55	50	34	39	44	49	63	76	87	120	155	150	123	100	123	109	86	62	71	78	52	49	39	44	35	24	15	12										
33	95	97	64	63	56	42	49	56	60	75	86	97	126	160	155	128	104	128	113	90	67	76	82	62	59	49	53	40	29	25	23	11									
34	74	81	44	43	35	23	30	39	44	62	78	89	121	159	155	127	108	136	124	101	75	79	81	54	50	42	46	43	39	23	14	14	21								
35	67	69	42	41	31	25	32	41	46	64	83	90	130	164	160	133	114	146	134	111	85	84	86	59	52	47	51	53	49	32	24	24	30	9							
36	74	76	61	60	42	44	51	60	66	83	102	110	147	185	179	155	133	159	146	122	98	105	107	79	71	66	70	70	60	48	40	36	33	25	18						
37	57	59	46	41	25	30	36	47	52	71	93	98	136	172	172	148	126	158	147	124	121	97	99	71	65	59	63	67	62	46	38	37	43	23	13	17					
38	45	46	41	34	20	34	38	48	53	73	96	99	137	176	178	151	131	163	159	135	108	102	103	73	67	64	69	75	72	54	46	49	54	34	24	29	12				
39	35	37	35	26	18	34	36	46	51	70	93	97	134	171	176	151	129	161	163	139	118	102	101	71	65	65	70	84	78	58	50	56	62	41	32	38	21	9			
40	29	33	30	21	18	35	33	40	45	65	87	91	117	166	171	144	125	157	156	139	113	95	97	67	60	62	67	79	82	62	53	59	66	45	38	45	27	15	6		
41	3	11	41	37	47	57	55	58	63	83	105	109	147	186	188	164	144	176	182	161	134	119	116	86	78	84	88	101	108	88	80	86	92	71	64	71	54	41	32	25	
42	5	12	55	41	53	64	61	61	66	84	111	113	150	186	192	166	147	180	188	167	140	124	119	90	87	90	94	107	114	77	86	92	98	80	74	77	60	48	38	32	6
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**

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

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

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

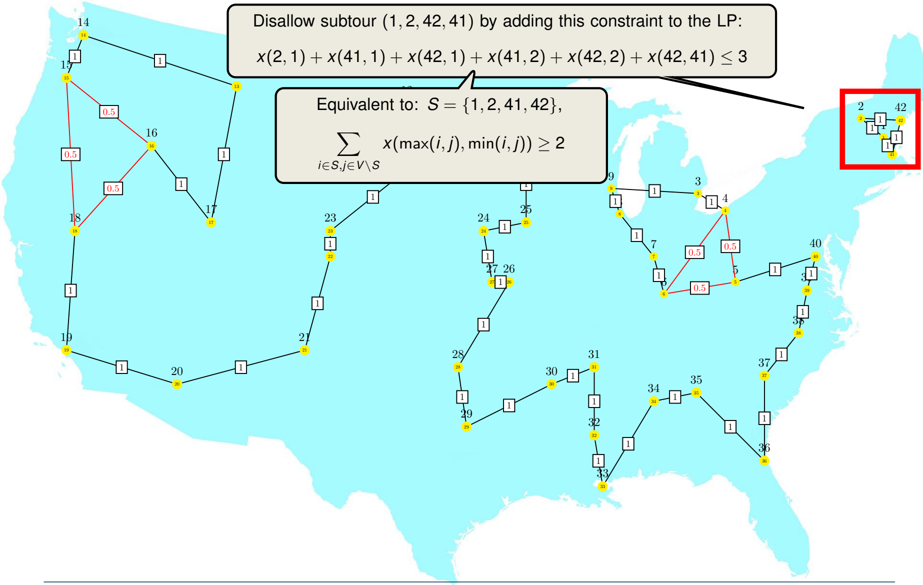
Examples of TSP Instances

Demonstration

In the following, there are a few different runs of the demo. In the example class, we choose a different branching variable in iteration 7 ($x_{16,17}$) and found the optimal very quickly.

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:

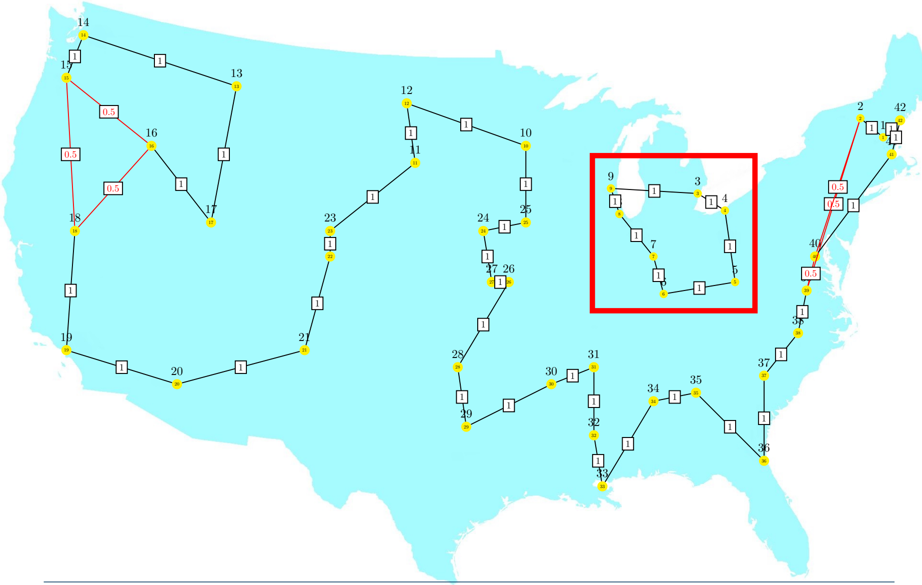
$$x(2,1) + x(41,1) + x(42,1) + x(41,2) + x(42,2) + x(42,41) \leq 3$$

Equivalent to: $S = \{1, 2, 41, 42\}$,

$$\sum_{i \in S, j \in V \setminus S} x(\max(i,j), \min(i,j)) \geq 2$$

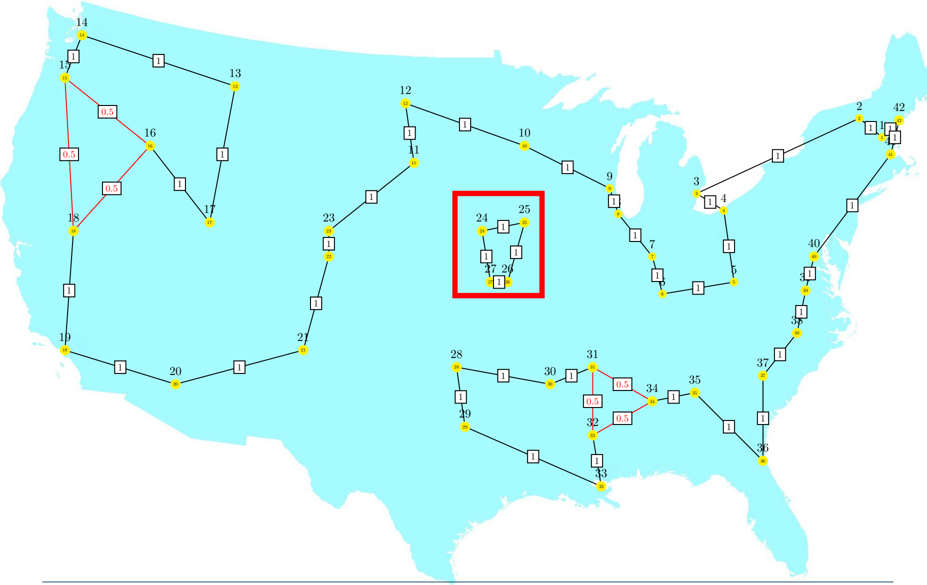
Iteration 2: Eliminate Subtour $3-9$

Objective value: -676.000000, 861 variables, 946 constraints, 1802 iterations



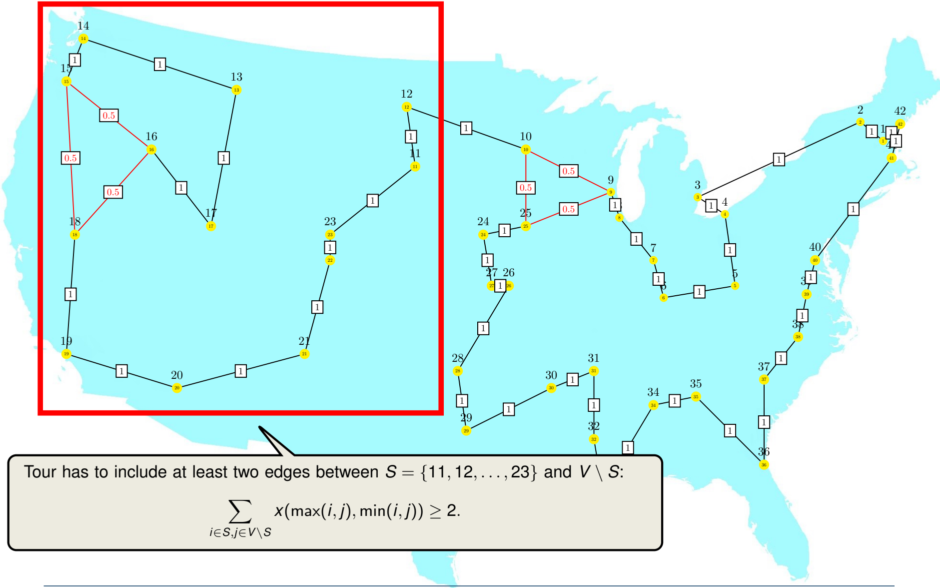
Iteration 3: Eliminate Subtour $24, 25, 26, 27$

Objective value: -681.000000, 861 variables, 947 constraints, 1984 iterations



Iteration 4: Eliminate Cut $11-23$

Objective value: -682.500000, 861 variables, 948 constraints, 1492 iterations

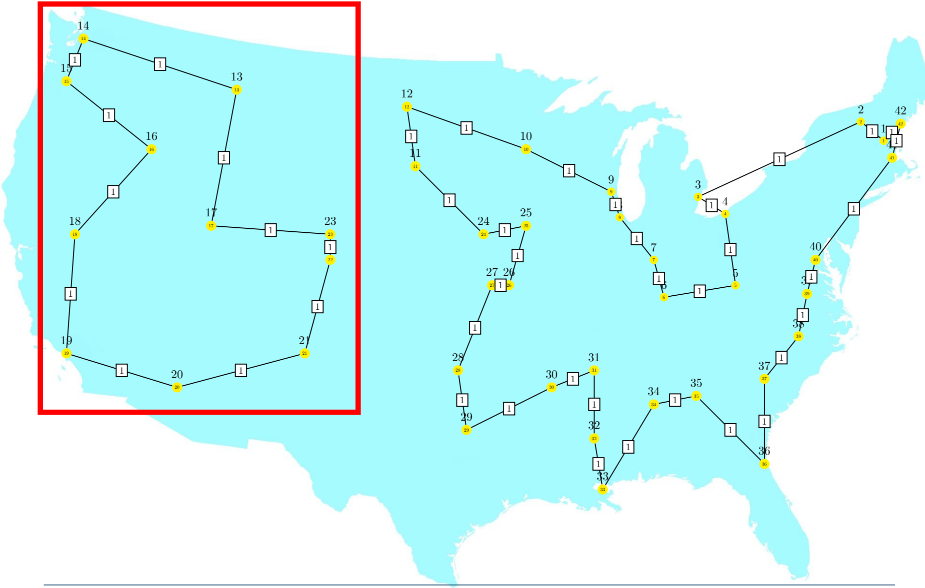


Tour has to include at least two edges between $S = \{11, 12, \dots, 23\}$ and $V \setminus S$:

$$\sum_{i \in S, j \in V \setminus S} x(\max(i,j), \min(i,j)) \geq 2.$$

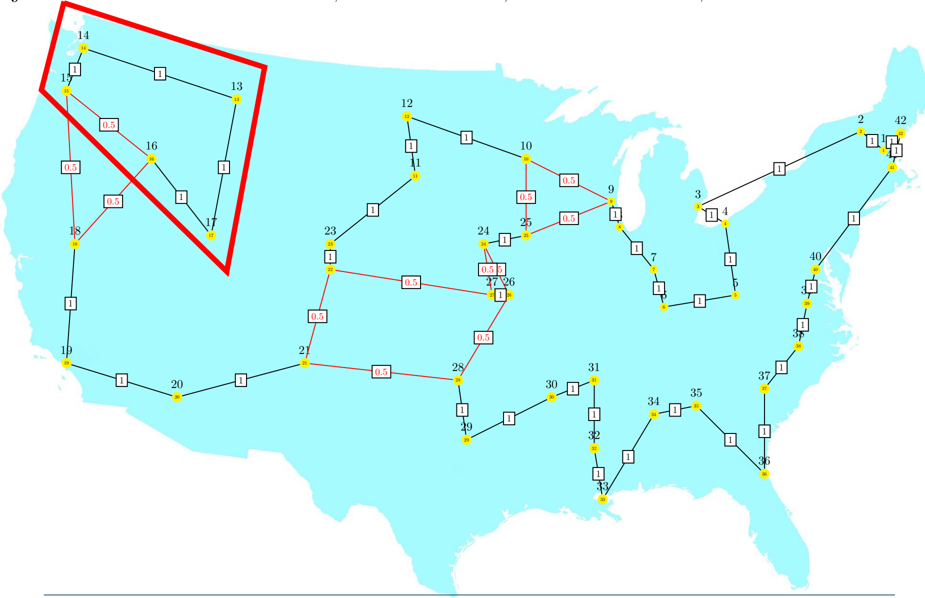
Iteration 5: Eliminate Subtour $13 - 23$

Objective value: -686.000000, 861 variables, 949 constraints, 2446 iterations



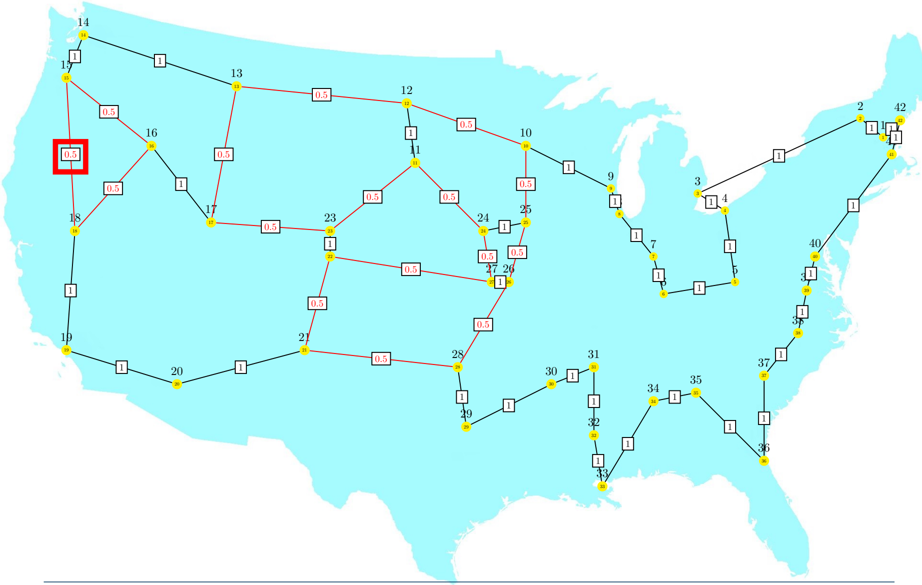
Iteration 6: Eliminate Cut 13 – 17

Objective value: −694.500000, 861 variables, 950 constraints, 1690 iterations



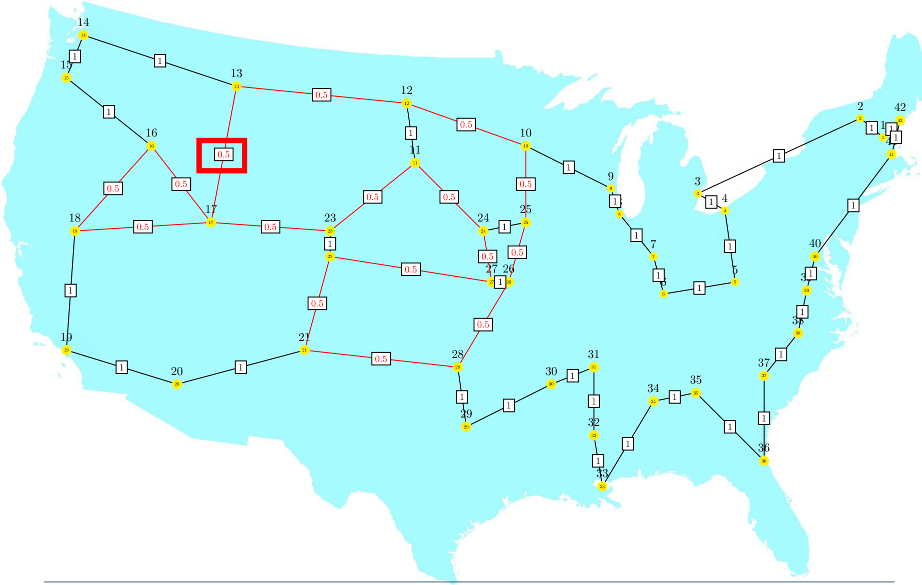
Iteration 7: Branch 1a $x_{18,15} = 0$

Objective value: -697.000000, 861 variables, 951 constraints, 2212 iterations



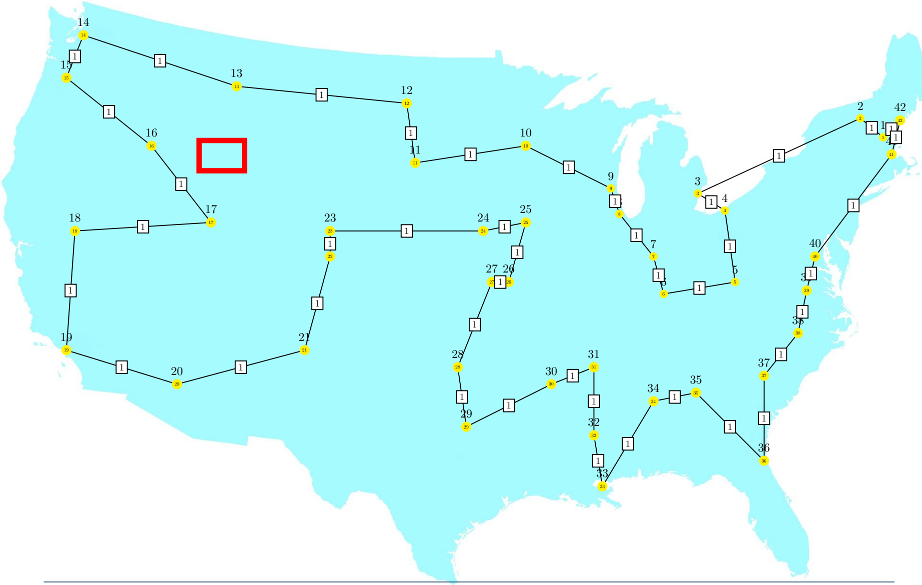
Iteration 8: Branch 2a $x_{17,13} = 0$

Objective value: −698.000000, 861 variables, 952 constraints, 1878 iterations



Iteration 9: Branch 2b $x_{17,13} = 1$

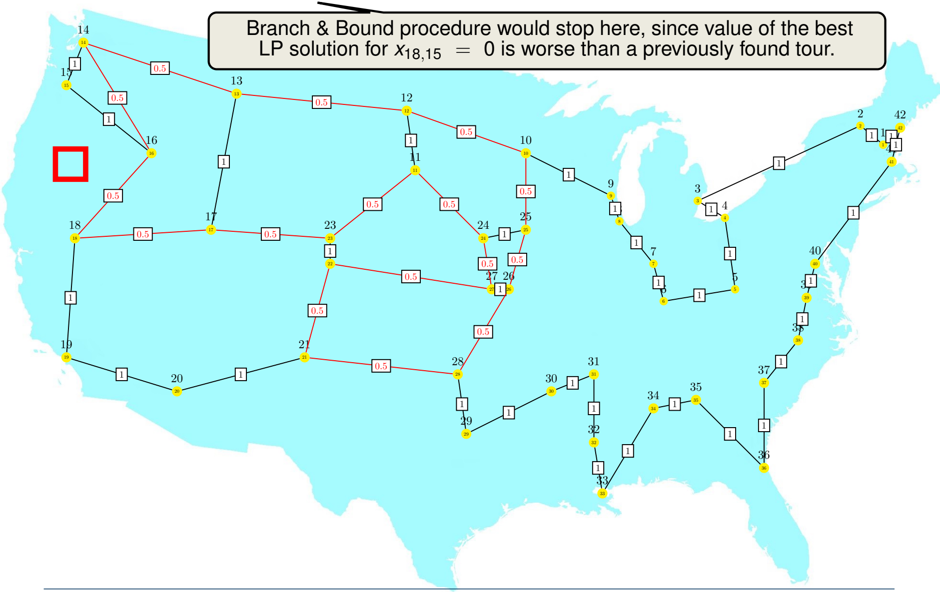
Objective value: -699.000000, 861 variables, 953 constraints, 2281 iterations



Iteration 10: Branch 1b $x_{18,15} = 1$

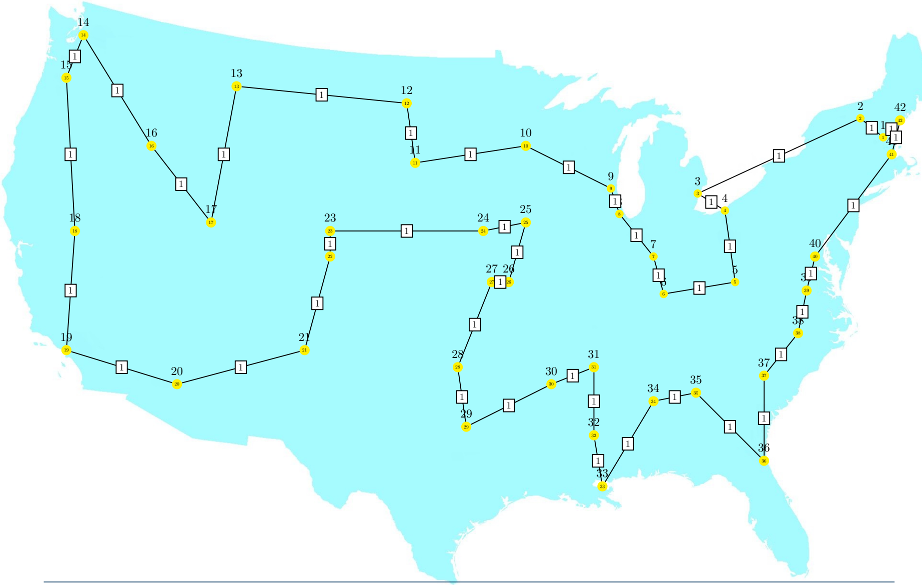
Objective value: -700.000000, 861 variables, 954 constraints, 2398 iterations

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.

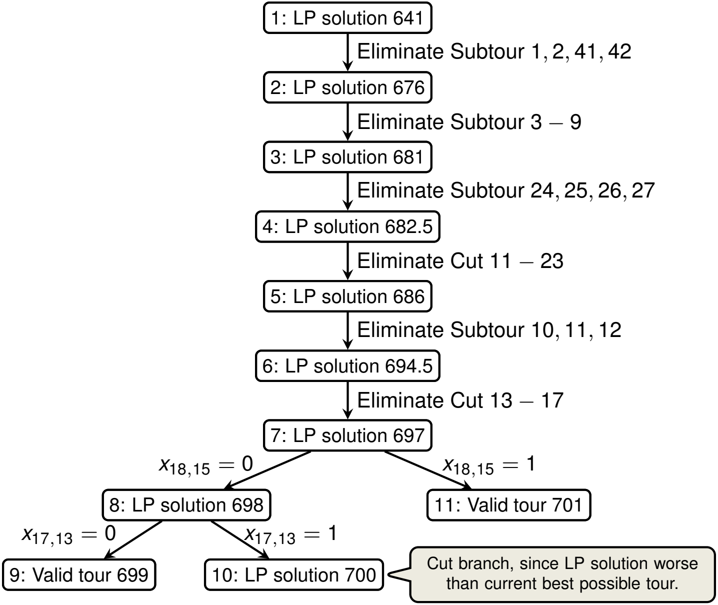


Iteration 11: Branch & Bound terminates

Objective value: -701.000000, 861 variables, 953 constraints, 2506 iterations



Branch & Bound Overview



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

Eliminate Cut $11 - 23$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 694.5

Eliminate Cut $13 - 17$

7: LP solution 697

$x_{18,15} = 0$

$x_{18,15} = 1$

8: LP solution 698

11: Valid tour 701

$x_{17,13} = 0$

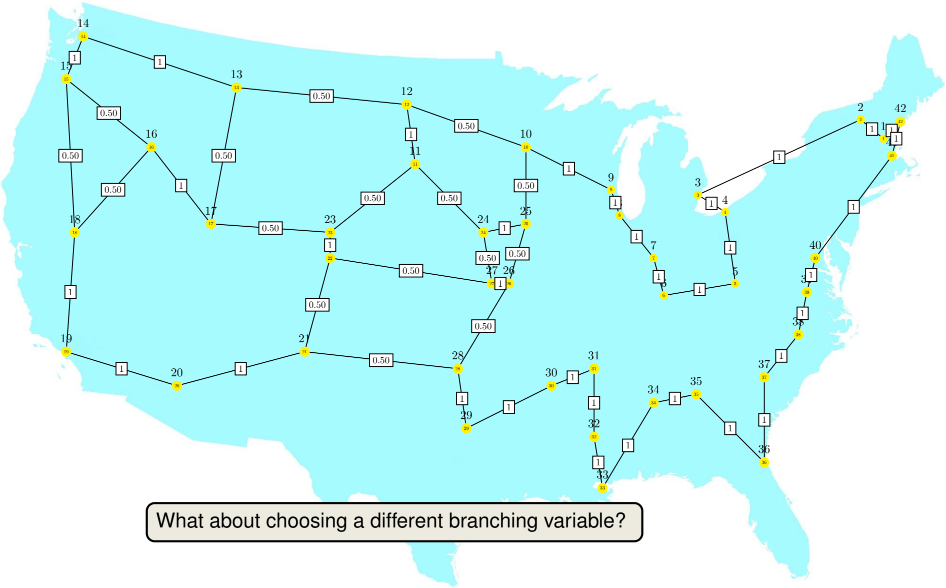
$x_{17,13} = 1$

9: Valid tour 699

10: LP solution 700

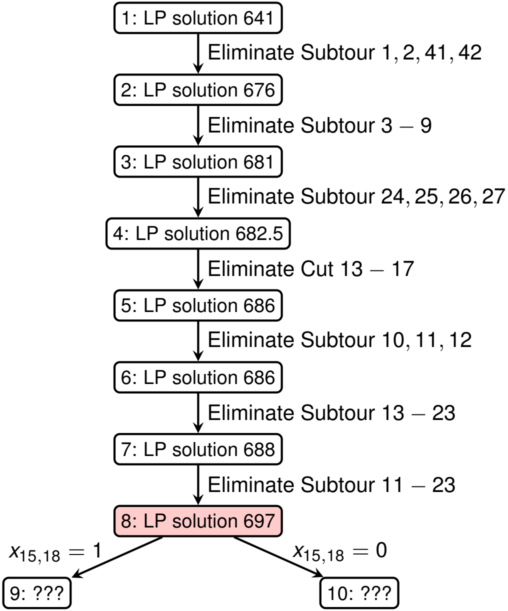
Cut branch, since LP solution worse than current best possible tour.

Iteration 8: Objective 697



What about choosing a different branching variable?

Solving Progress (Alternative Branch 1)



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

Eliminate Cut $13 - 17$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 686

Eliminate Subtour $13 - 23$

7: LP solution 688

Eliminate Subtour $11 - 23$

8: LP solution 697

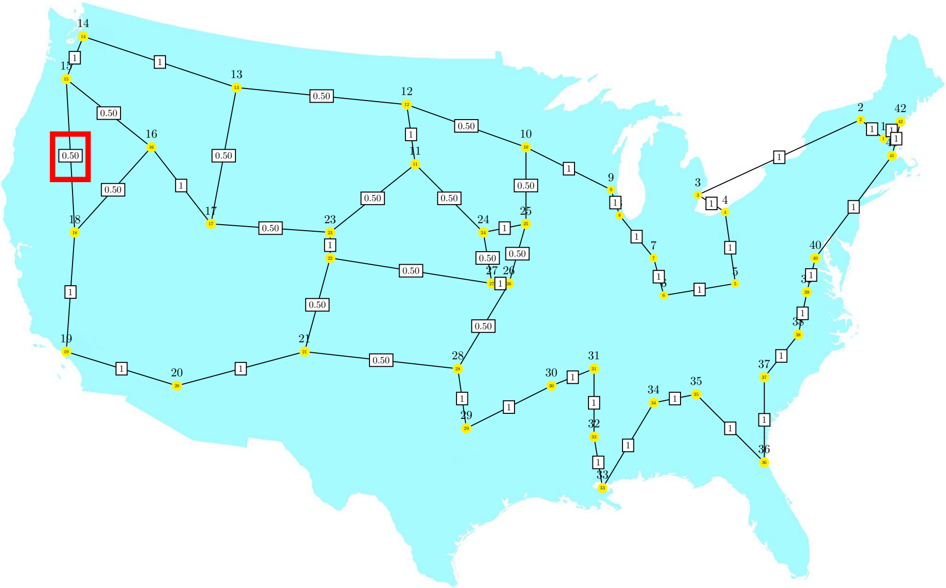
$x_{15,18} = 1$

$x_{15,18} = 0$

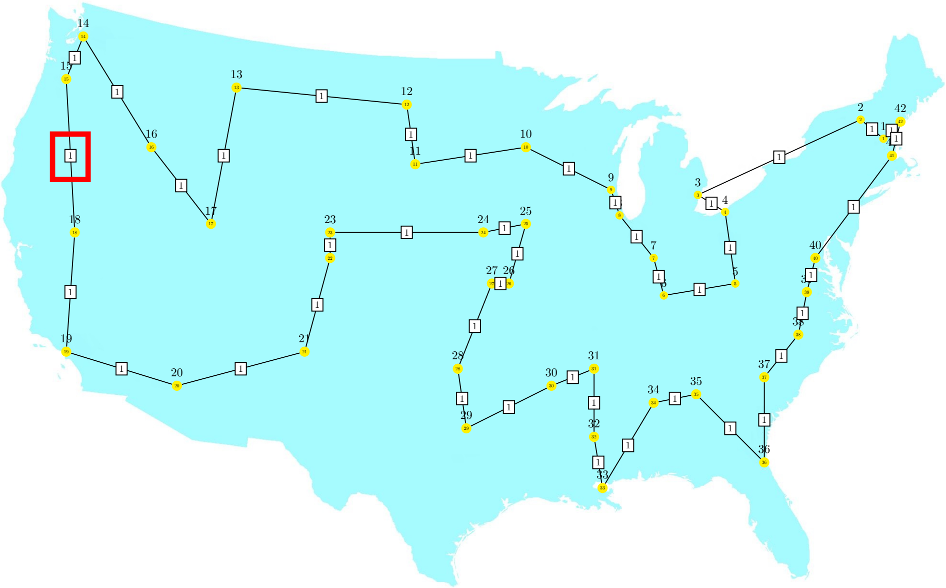
9: ???

10: ???

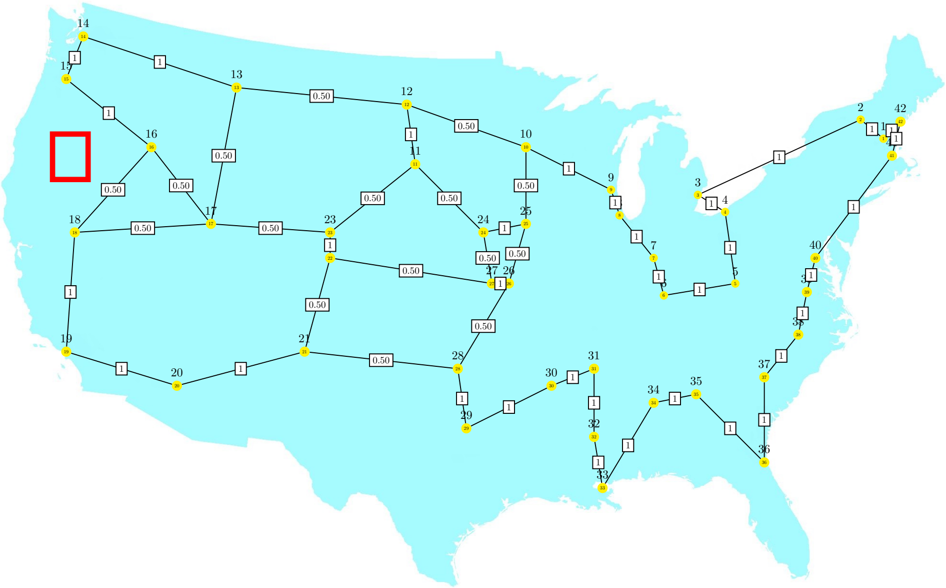
Alternative Branch 1: $x_{18,15}$, Objective 697



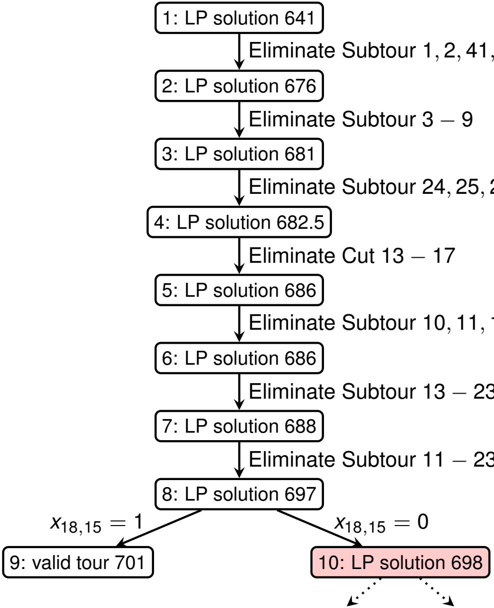
Alternative Branch 1a: $x_{18,15} = 1$, Objective 701 (Valid Tour)



Alternative Branch 1b: $x_{18,15} = 0$, Objective 698



Solving Progress (Alternative Branch 1)



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

Eliminate Cut $13 - 17$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 686

Eliminate Subtour $13 - 23$

7: LP solution 688

Eliminate Subtour $11 - 23$

8: LP solution 697

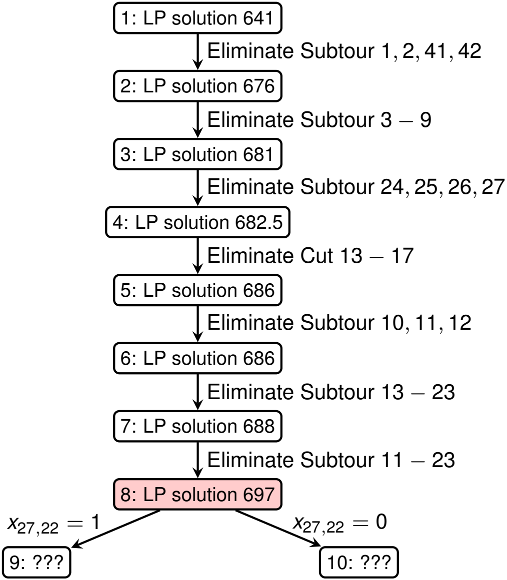
$x_{18,15} = 1$

9: valid tour 701

$x_{18,15} = 0$

10: LP solution 698

Solving Progress (Alternative Branch 2)



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

Eliminate Cut $13 - 17$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 686

Eliminate Subtour $13 - 23$

7: LP solution 688

Eliminate Subtour $11 - 23$

8: LP solution 697

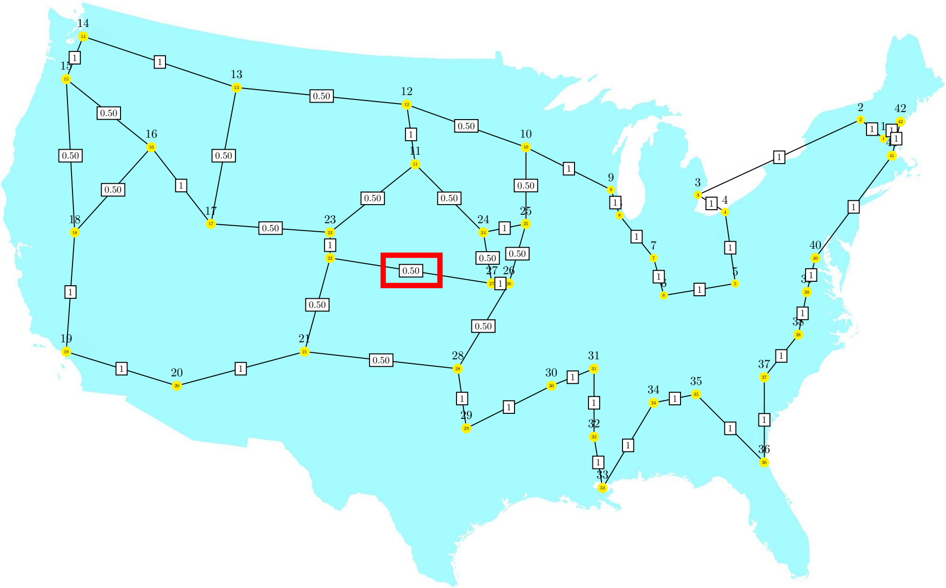
$x_{27,22} = 1$

9: ???

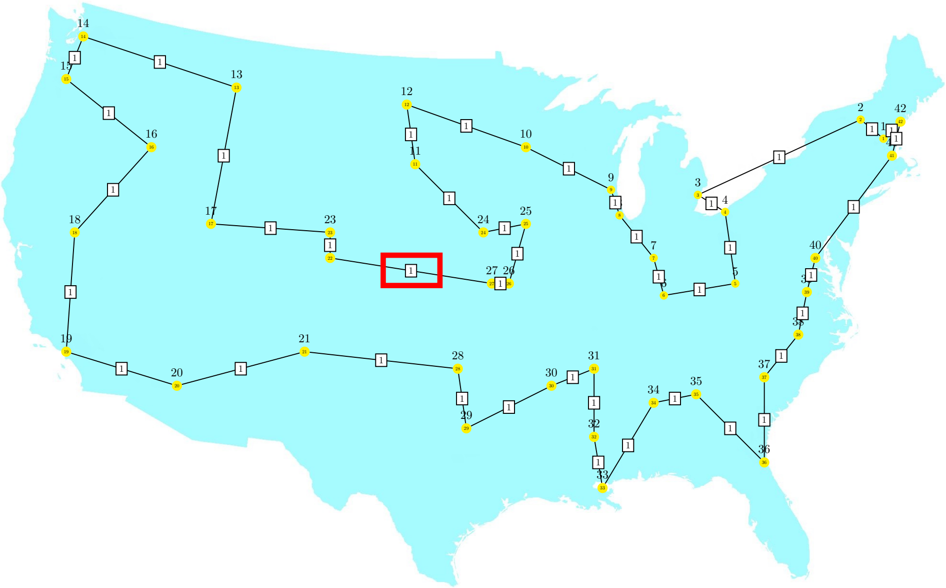
$x_{27,22} = 0$

10: ???

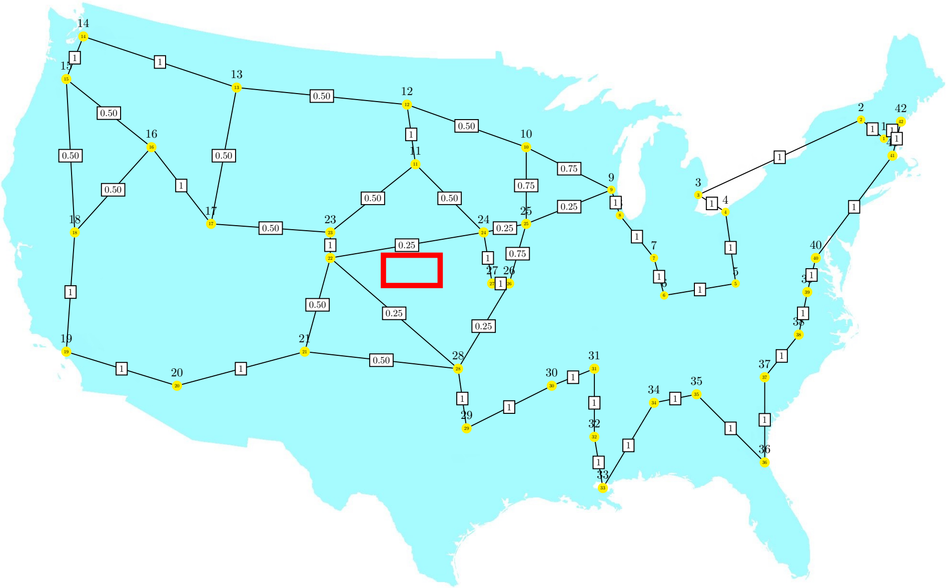
Alternative Branch 2: $x_{27,22}$, Objective 697



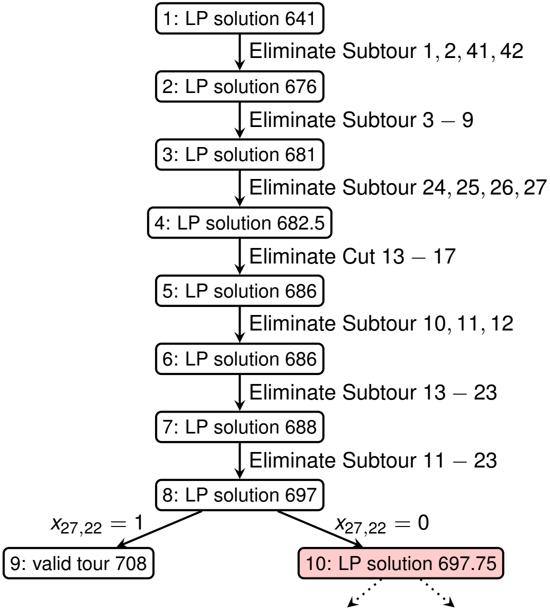
Alternative Branch 2a: $x_{27,22} = 1$, Objective 708 (Valid tour)



Alternative Branch 2b: $x_{27,22} = 0$, Objective 697.75



Solving Progress (Alternative Branch 2)



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

Eliminate Cut $13 - 17$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 686

Eliminate Subtour $13 - 23$

7: LP solution 688

Eliminate Subtour $11 - 23$

8: LP solution 697

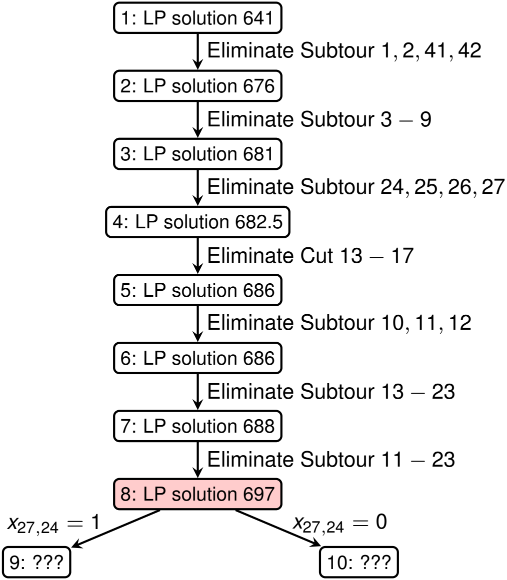
$x_{27,22} = 1$

$x_{27,22} = 0$

9: valid tour 708

10: LP solution 697.75

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

Eliminate Cut $13 - 17$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 686

Eliminate Subtour $13 - 23$

7: LP solution 688

Eliminate Subtour $11 - 23$

8: LP solution 697

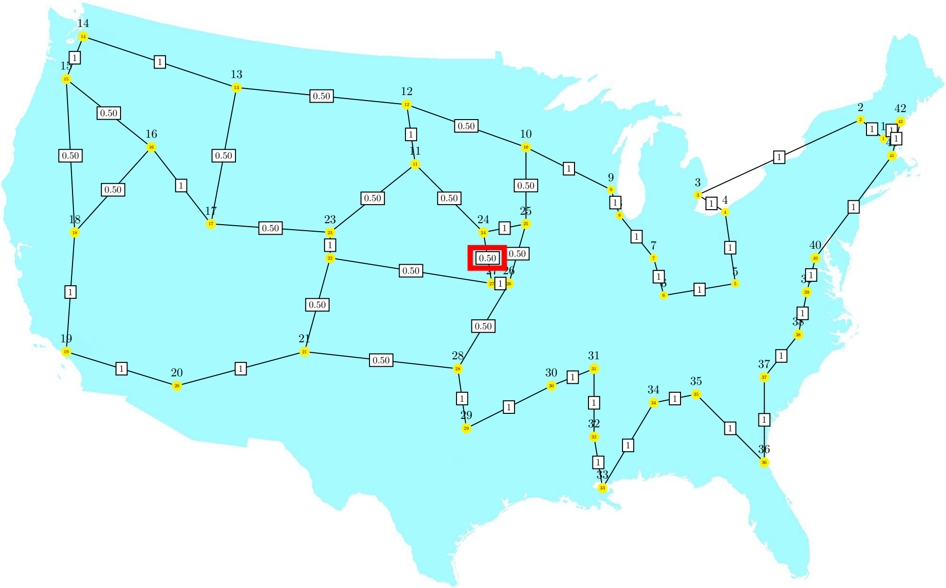
$x_{27,24} = 1$

9: ???

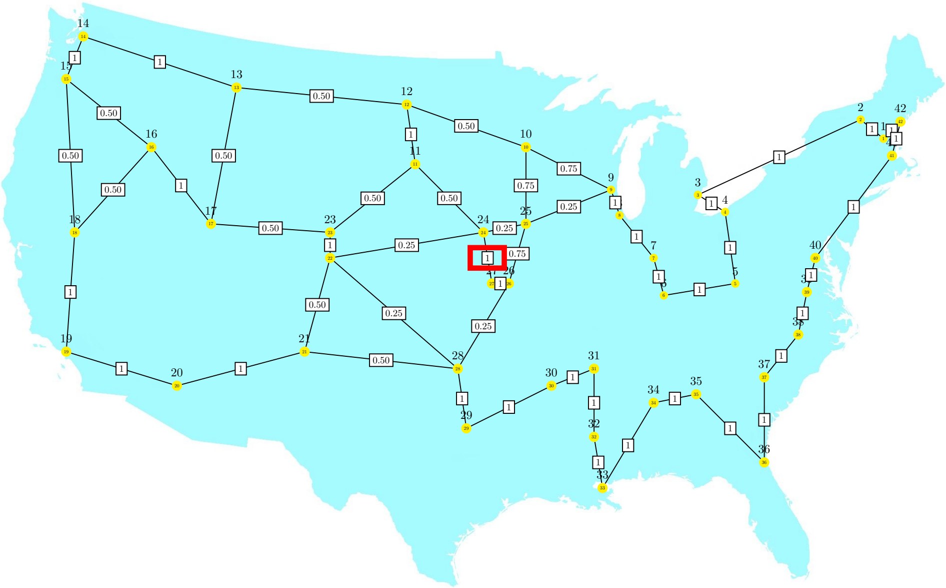
$x_{27,24} = 0$

10: ???

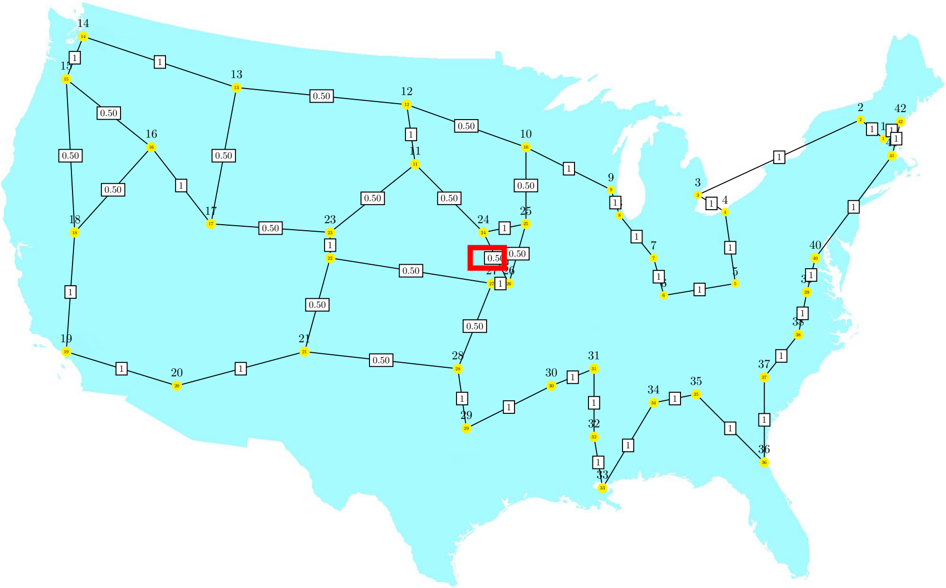
Alternative Branch 3: $x_{27,24}$, Objective 697



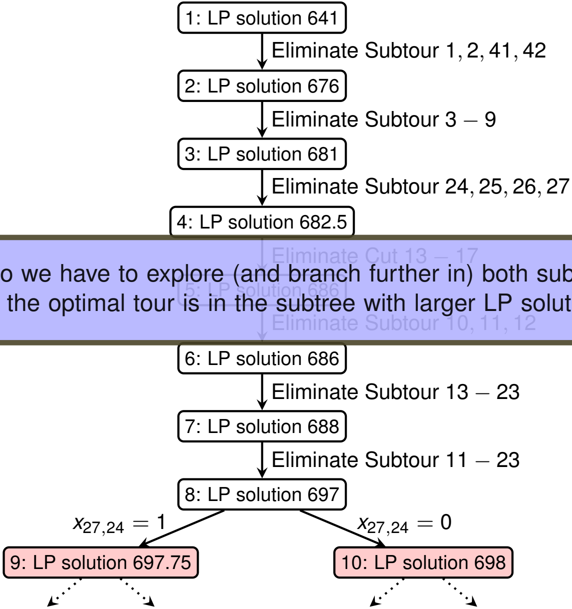
Alternative Branch 3a: $x_{27,24} = 1$, Objective 697.75



Alternative Branch 3b: $x_{27,24} = 0$, Objective 698



Solving Progress (Alternative Branch 3)



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

Eliminate Cut $13 - 17$

5: LP solution 686

Eliminate Subtour $10, 11, 12$

6: LP solution 686

Eliminate Subtour $13 - 23$

7: LP solution 688

Eliminate Subtour $11 - 23$

8: LP solution 697

$x_{27,24} = 1$

$x_{27,24} = 0$

9: LP solution 697.75

10: LP solution 698

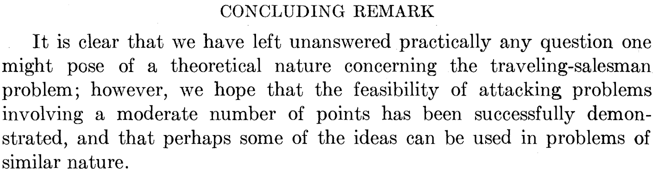
Not only do we have to explore (and branch further in) both subtrees, but also the optimal tour is in the subtree with larger LP solution!

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 Eliminiation 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.

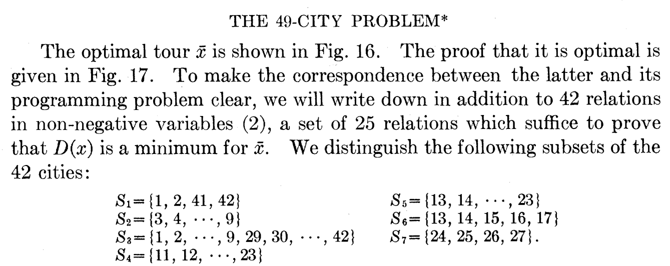


CONCLUDING REMARK

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 $24, 25, 26, 27$

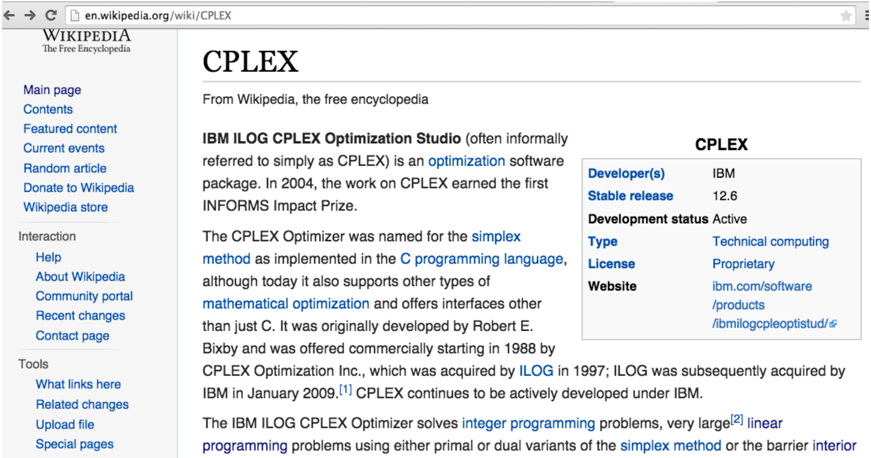


THE 49-CITY PROBLEM*

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:

$S_1 = \{1, 2, 41, 42\}$
$S_2 = \{3, 4, \cdots, 9\}$
$S_3 = \{1, 2, \cdots, 9, 29, 30, \cdots, 42\}$
$S_4 = \{11, 12, \cdots, 23\}$
$S_5 = \{13, 14, \cdots, 23\}$
$S_6 = \{13, 14, 15, 16, 17\}$
$S_7 = \{24, 25, 26, 27\}$.

CPLEX



CPLEX

From Wikipedia, the free encyclopedia

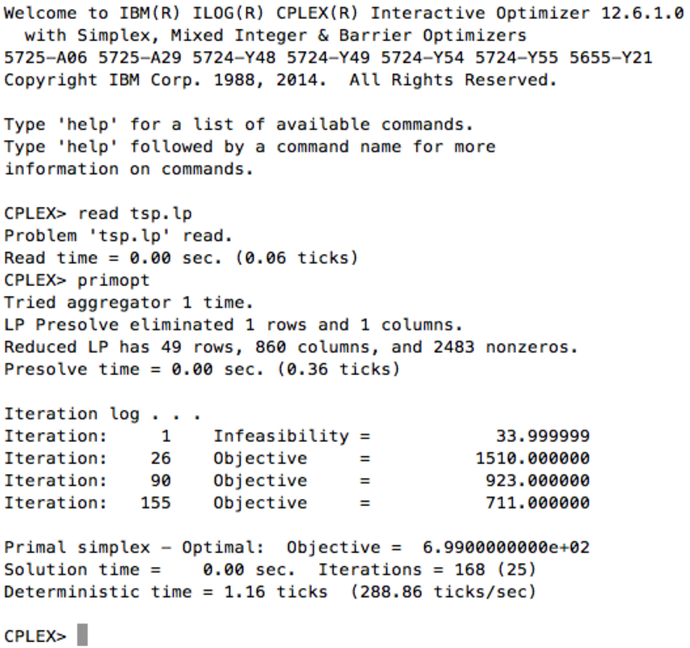
IBM ILOG CPLEX Optimization Studio (often informally referred to simply as CPLEX) is an optimization software package. In 2004, the work on CPLEX earned the first INFORMS Impact Prize.

The CPLEX Optimizer was named for the simplex method as implemented in the C programming language, although today it also supports other types of mathematical optimization and offers interfaces other than just C. It was originally developed by Robert E. Bixby and was offered commercially starting in 1988 by CPLEX Optimization Inc., which was acquired by ILOG in 1997; ILOG was subsequently acquired by IBM in January 2009.[1] CPLEX continues to be actively developed under IBM.

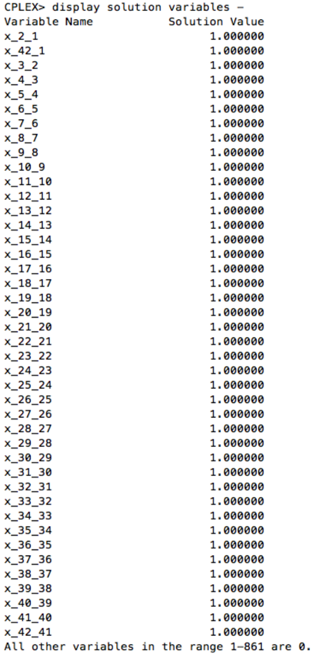
The IBM ILOG CPLEX Optimizer solves integer programming problems, very large[2] linear programming problems using either primal or dual variants of the simplex method or the barrier interior

CPLEX

Developer(s)	IBM
Stable release	12.6
Development status	Active
Type	Technical computing
License	Proprietary
Website	ibm.com/software /products /ibmilogcpleoptistud/



```
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-CNF and Vertex-Cover

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

UNIVERSITY OF CAMBRIDGE

Outline

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:

$$\max\left(\frac{\mathbf{E}[C]}{C^*}, \frac{C^*}{\mathbf{E}[C]}\right) \leq \rho(n).$$

not covered here...

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, $O((1/\epsilon)^2 \cdot n^3)$.

Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

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.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \cdots$
- 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?

Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with n variables $x_1, x_2, \ldots, x_n$ and m clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

Proof:

- For every clause $i = 1, 2, \ldots, m$, 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,

$$\begin{aligned} \mathbf{P}[\,\text{clause } i \text{ is not satisfied}\,] &= \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \\ \Rightarrow \qquad \mathbf{P}[\,\text{clause } i \text{ is satisfied}\,] &= 1 - \frac{1}{8} = \frac{7}{8} \\ \Rightarrow \qquad \mathbf{E}[\,Y_i\,] &= \mathbf{P}[\,Y_i = 1\,] \cdot 1 = \frac{7}{8}. \end{aligned}$$

- Let $Y := \sum_{i=1}^{m} Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[\,Y\,] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} \mathbf{E}[\,Y_i\,] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8} \cdot m. \qquad \square$$

Linearity of Expectations

maximum number of satisfiable clauses is m

Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with n variables $x_1, x_2, \ldots, x_n$ 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 assigment 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 $x_1, x_2, \ldots, x_n$ 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($\phi$, $n$, $m$)
for $j = 1, 2, \ldots, n$
    Compute $\mathbf{E}[\,Y \mid x_1 = v_1 \ldots, x_{j-1} = v_{j-1}, x_j = 1\,]$
    Compute $\mathbf{E}[\,Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 0\,]$
    Let $x_j = v_j$ so that the conditional expectation is maximized
return the assignment $v_1, v_2, \ldots, v_n$
```

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 $j = 1, 2, \ldots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
  - A smarter way is to use linearity of (conditional) expectations:

$$\mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \,\right] = \sum_{i=1}^{m} \mathbf{E}\left[\, Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \,\right]$$

computable in $O(1)$

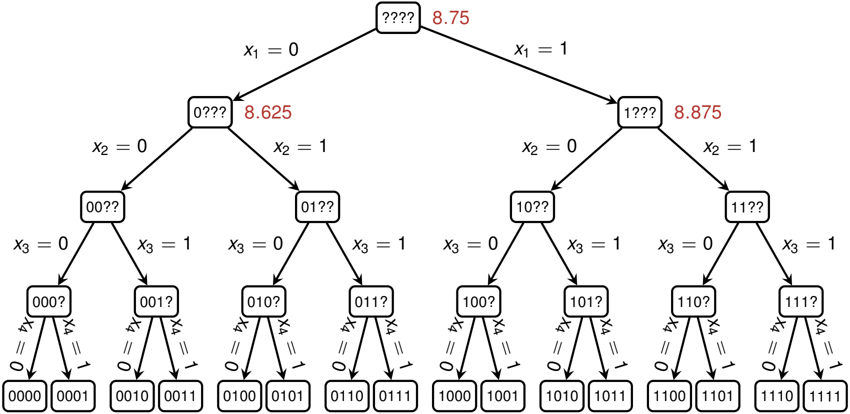
- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,

$$\begin{aligned}\mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \,\right] &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \,\right] \\ &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \,\right] \\ &\vdots \\ &\geq \mathbf{E}\left[\, Y \,\right] = \frac{7}{8} \cdot m. \end{aligned}$$

□

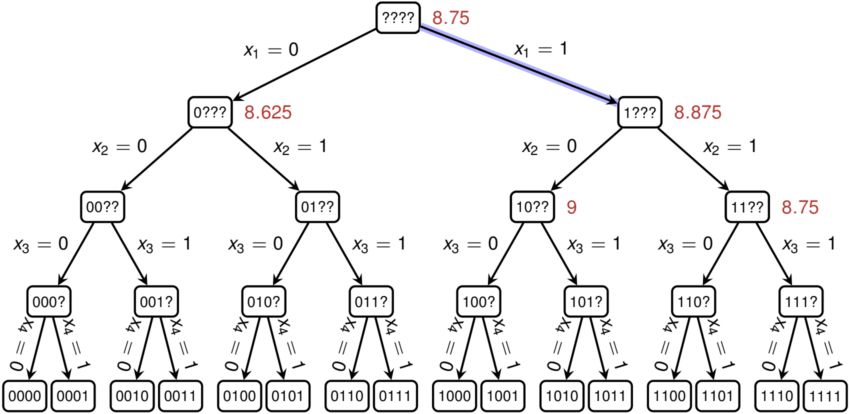
Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge$
$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



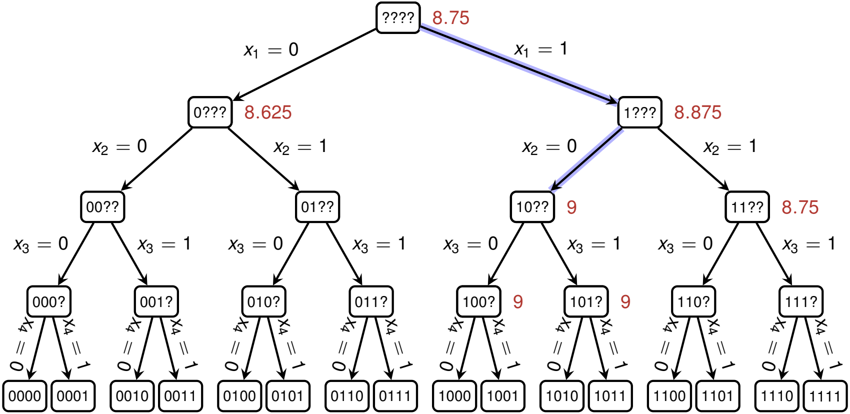
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



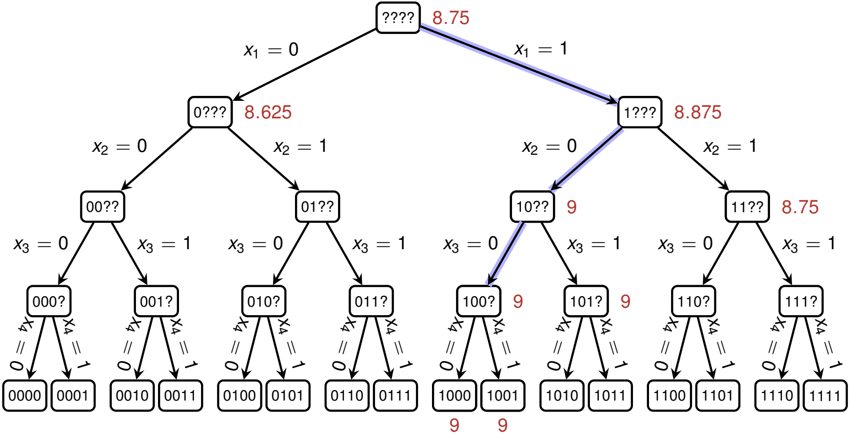
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



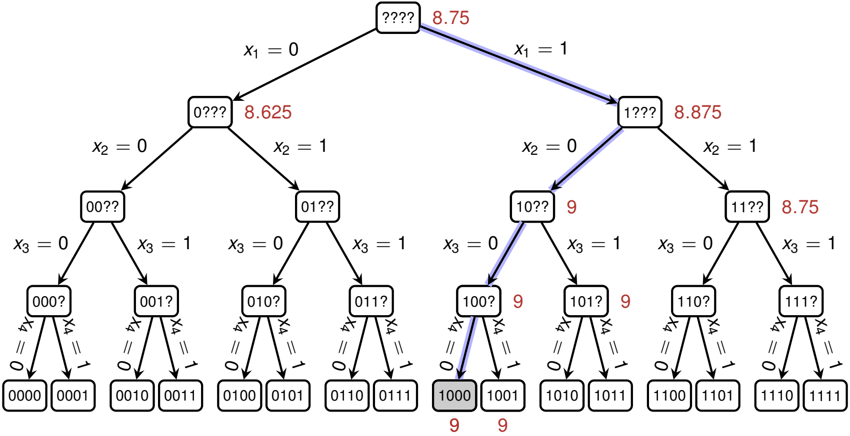
Run of **GREEDY-3-CNF**(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



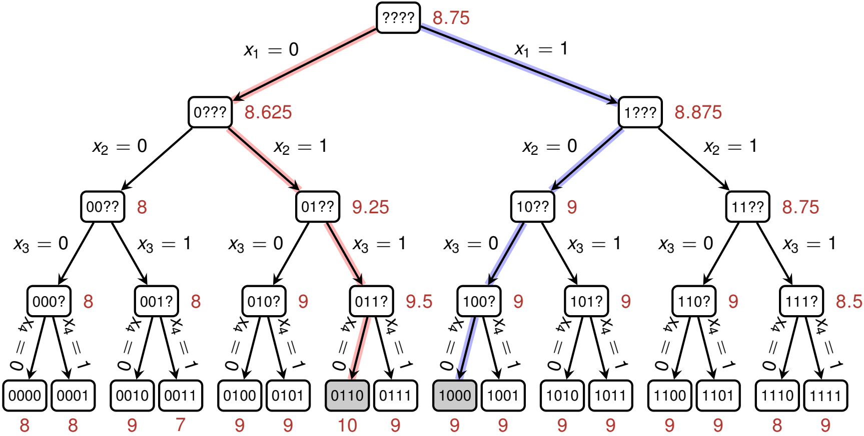
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge$
$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.

MAX-3-CNF: Concluding Remarks

Theorem 35.6

Given an instance of MAX-3-CNF with n variables $x_1, x_2, \ldots, x_n$ 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.

Essentially there is nothing smarter than just guessing!

Outline

Randomised Approximation

MAX-3-CNF

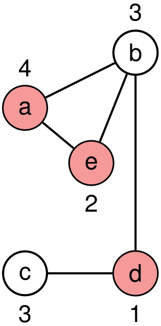
Weighted Vertex Cover

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 $v \in V'$.

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

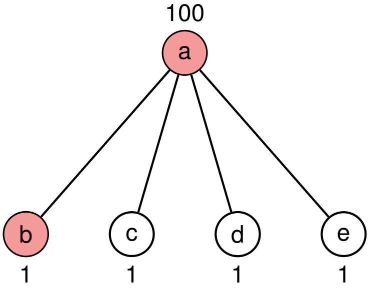
A Greedy Approach working for Unweighted Vertex Cover

```
APPROX-VERTEX-COVER($G$)
C = ∅
E' = G.E
while $E' \neq \emptyset$
    let $(u, v)$ be an arbitrary edge of $E'$
    $C = C \cup \{u, v\}$
    remove from $E'$ every edge incident on either $u$ or $v$
return $C$
```

This algorithm is a 2-approximation for **unweighted graphs**!

A Greedy Approach working for Unweighted Vertex Cover

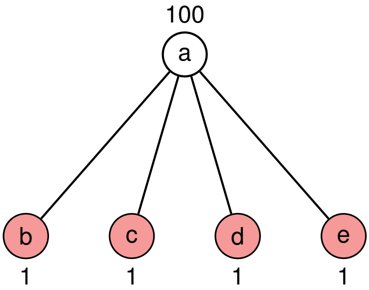
```
APPROX-VERTEX-COVER($G$)
$C = \emptyset$
$E' = G.E$
while $E' \neq \emptyset$
    let $(u, v)$ be an arbitrary edge of $E'$
    $C = C \cup \{u, v\}$
    remove from $E'$ every edge incident on either $u$ or $v$
return $C$
```



Computed solution has weight 101

A Greedy Approach working for Unweighted Vertex Cover

```
APPROX-VERTEX-COVER($G$)
$C = \emptyset$
$E' = G.E$
while $E' \neq \emptyset$
    let $(u, v)$ be an arbitrary edge of $E'$
    $C = C \cup \{u, v\}$
    remove from $E'$ every edge incident on either $u$ or $v$
return $C$
```



Optimal solution has weight 4

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{llcll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\ \text{subject to} & x(u)+x(v) & \geq & 1 & \text{for each } (u,v) \in E \\ & x(v) & \in & \{0,1\} & \text{for each } v \in V \end{array}$$

optimum is a lower bound on the optimal weight of a minimum weight-cover.

Linear Program

$$\begin{array}{llcll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\ \text{subject to} & x(u)+x(v) & \geq & 1 & \text{for each } (u,v) \in E \\ & x(v) & \in & [0,1] & \text{for each } v \in V \end{array}$$

Rounding Rule: if $x(v) \geq 1/2$ then round up, otherwise round down.

The Algorithm

```
APPROX-MIN-WEIGHT-VC$(G, w)$
C = ∅
compute $\bar{x}$, an optimal solution to the linear program
for each $v \in V$
    if $\bar{x}(v) \geq 1/2$
        $C = C \cup \{v\}$
return $C$
```

Theorem 35.7

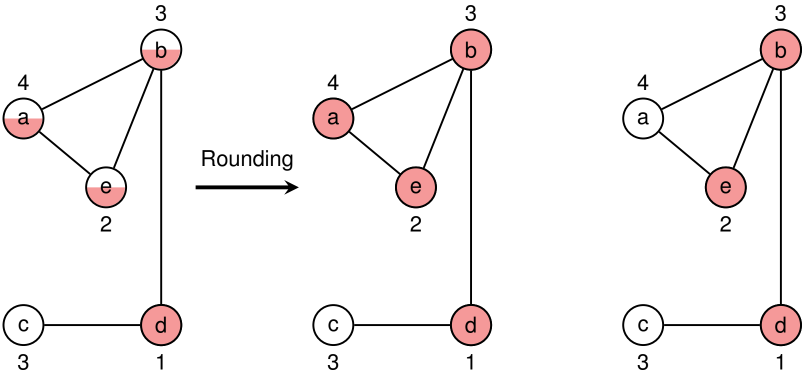
APPROX-MIN-WEIGHT-VC is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.

is polynomial-time because we can solve the linear program in polynomial time

Example of APPROX-MIN-WEIGHT-VC

$\overline{x}(a) = \overline{x}(b) = \overline{x}(e) = \frac{1}{2}, \overline{x}(d) = 1, \overline{x}(c) = 0$

$x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0$



fractional solution of LP
with weight $= 5.5$

rounded solution of LP
with weight $= 10$

optimal solution
with weight $= 6$

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

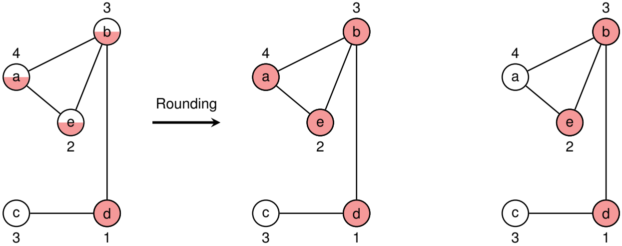
$$z^* \leq w(C^*)$$

- **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 $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)

- **Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v)\overline{x}(v) \geq \sum_{v \in V:\ \overline{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C). \quad \square$$



Randomised Algorithms

Lecture 10: Approximation Algorithms: Set-Cover and MAX-k-CNF

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

Outline

Weighted Set Cover

MAX-CNF

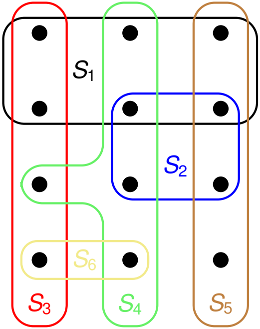
Appendix: An Approximation Algorithm of TSP (non-examin.)

The Weighted Set-Covering Problem

Set Cover Problem

- Given: set X and a family of subsets $\mathcal{F}$, and a cost function $c : \mathcal{F} \to \mathbb{R}^+$
- Goal: Find a minimum-cost subset $\mathcal{C} \subseteq \mathcal{F}$ s.t. $$X = \bigcup_{S \in \mathcal{C}} S.$$

Sum over the costs of all sets in $\mathcal{C}$



	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

Setting up an Integer Program

Exercise: Try to formulate the integer program and linear program of the weighted SET-COVER problem (solution on next slide!)

Setting up an Integer Program

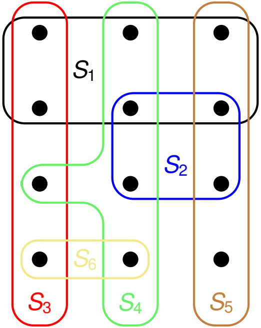
0-1 Integer Program

$$\begin{array}{llllll}
\text{minimize} & \sum_{S \in \mathcal{F}} c(S) y(S) & & & & \\
\text{subject to} & \sum_{S \in \mathcal{F}\colon\, x \in S} & y(S) & \geq & 1 & \text{for each } x \in X \\
& & y(S) & \in & \{0, 1\} & \text{for each } S \in \mathcal{F}
\end{array}$$

Linear Program

$$\begin{array}{llllll}
\text{minimize} & \sum_{S \in \mathcal{F}} c(S) y(S) & & & & \\
\text{subject to} & \sum_{S \in \mathcal{F}\colon\, x \in S} & y(S) & \geq & 1 & \text{for each } x \in X \\
& & y(S) & \in & [0, 1] & \text{for each } S \in \mathcal{F}
\end{array}$$

Back to the Example



	S_1	S_2	S_3	S_4	S_5	S_6
c :	2	3	3	5	1	2
$\overline{y}(.)$:	$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$

Cost equals 8.5

The strategy employed for Vertex-Cover would take all 6 sets!

Even worse: If all $\overline{y}$'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
$\overline{y}(.)$:	$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$

Idea: Interpret the $\overline{y}$-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 $\overline{y}(S)$.
- More precisely, if $\overline{y}$ denotes the optimal solution of the LP, then we compute an integral solution y by:

$$y(S) = \begin{cases} 1 & \text{with probability } \overline{y}(S) \\ 0 & \text{otherwise.} \end{cases} \qquad \text{for all } S \in \mathcal{F}.$$

- Therefore, $\mathbf{E}\left[\, y(S) \,\right] = \overline{y}(S)$.

Randomised Rounding

	S_1	S_2	S_3	S_4	S_5	S_6
c :	2	3	3	5	1	2
$\overline{y}(.)$:	$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$

Idea: Interpret the $\overline{y}$-values as probabilities for picking the respective set.

Lemma

- The expected cost satisfies

$$\mathbf{E}\left[\, c(\mathcal{C}) \,\right] = \sum_{S \in \mathcal{F}} c(S) \cdot \overline{y}(S)$$

- 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 $\overline{y}(S)$.

- The expected cost satisfies $\mathbf{E}\left[\, c(\mathcal{C}) \,\right] = \sum_{S \in \mathcal{F}} c(S) \cdot \overline{y}(S)$.
- The probability that x is covered satisfies $\mathbf{P}\left[\, x \in \cup_{S \in \mathcal{C}} S \,\right] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $\mathcal{C}$

$$\mathbf{E}\left[\, c(\mathcal{C}) \,\right] = \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S) \right] = \mathbf{E}\left[\sum_{S \in \mathcal{F}} \mathbf{1}_{S \in \mathcal{C}} \cdot c(S) \right]$$
$$= \sum_{S \in \mathcal{F}} \mathbf{P}\left[\, S \in \mathcal{C} \,\right] \cdot c(S) = \sum_{S \in \mathcal{F}} \overline{y}(S) \cdot c(S).$$

- **Step 2**: The probability for an element to be (not) covered

$$\mathbf{P}\left[\, x \notin \cup_{S \in \mathcal{C}} S \,\right] = \prod_{S \in \mathcal{F}\colon\, x \in S} \mathbf{P}\left[\, S \notin \mathcal{C} \,\right] = \prod_{S \in \mathcal{F}\colon\, x \in S} (1 - \overline{y}(S))$$
$$\leq \prod_{S \in \mathcal{F}\colon\, x \in S} e^{-\overline{y}(S)}$$
$$= e^{-\sum_{S \in \mathcal{F}\colon\, x \in S} \overline{y}(S)} \leq e^{-1} \qquad \square$$

$1 + x \leq e^x$ for any $x \in \mathbb{R}$

$\overline{y}$ solves the LP!

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($X, \mathcal{F}, c$)
compute $\overline{y}$, an optimal solution to the linear program
$\mathcal{C} = \emptyset$
repeat $2 \ln n$ times
    for each $S \in \mathcal{F}$
        let $\mathcal{C} = \mathcal{C} \cup \{S\}$ with probability $\overline{y}(S)$
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 $2\ln(n)$.

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

$$\mathbf{P}\left[\, x \not\in \cup_{S \in \mathcal{C}} S \,\right] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.$$

  - This implies for the event that all elements are covered:

$$\mathbf{P}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{P}\left[\bigcup_{x \in X} \{x \not\in \cup_{S \in \mathcal{C}} S\}\right]$$

$\mathbf{P}\left[\, A \cup B \,\right] \leq \mathbf{P}\left[\, A \,\right] + \mathbf{P}\left[\, B \,\right]$

$$\geq 1 - \sum_{x \in X} \mathbf{P}\left[\, x \not\in \cup_{S \in \mathcal{C}} S \,\right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$

- **Step 2**: The expected approximation ratio
  - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot \overline{y}(S)$.
  - Linearity $\Rightarrow$ $\mathbf{E}\left[\, c(\mathcal{C}) \,\right] \leq 2\ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot \overline{y}(S) \leq 2\ln(n) \cdot c(\mathcal{C}^*)$ ☐

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, $\mathbf{P}\left[\, c(\mathcal{C}) \leq 4\ln(n) \cdot c(\mathcal{C}^*) \,\right] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is 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

Outline

Weighted Set Cover

MAX-CNF

Appendix: An Approximation Algorithm of TSP (non-examin.)

MAX-CNF

Recall:

MAX-3-CNF Satisfiability

- Given: 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \cdots$
- Goal: Find an assignment of the variables that satisfies as many clauses as possible.

MAX-CNF Satisfiability (MAX-SAT)

- Given: CNF formula, e.g.: $(x_1 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4 \vee \overline{x_5}) \wedge \cdots$
- 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.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause i which has length ℓ,

$$\mathbf{P}\left[\text{clause } i \text{ is satisfied}\right] = 1 - 2^{-\ell} := \alpha_\ell.$$

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,

$$\mathbf{E}\left[\,Y\,\right] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} \mathbf{E}\left[\,Y_i\,\right] \geq \sum_{i=1}^{m} \frac{1}{2} = \frac{1}{2} \cdot m. \qquad \square$$

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

$$
\begin{array}{llll}
\text{maximize} & \displaystyle\sum_{i=1}^{m} z_i & & \\
\text{subject to} & \displaystyle\sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i & & \text{for each } i = 1, 2, \ldots, m \\
& z_i \in \{0, 1\} & & \text{for each } i = 1, 2, \ldots, m \\
& y_j \in \{0, 1\} & & \text{for each } j = 1, 2, \ldots, n
\end{array}
$$

These auxiliary variables are used to reflect whether a clause is satisfied or not

C_i^+ is the index set of the unnegated variables of clause i.

- In the corresponding LP each $\in \{0, 1\}$ is replaced by $\in [0, 1]$
- Let $(\overline{y}, \overline{z})$ be the optimal solution of the LP
- Obtain an integer solution y through randomised rounding of $\overline{y}$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

$$\mathbf{P}\left[\,\text{clause } i \text{ is satisfied}\,\right] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot \overline{z}_i.$$

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 $\overline{x_j}$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \vee \cdots \vee x_\ell)$

$$\Rightarrow \quad \mathbf{P}\left[\,\text{clause } i \text{ is satisfied}\,\right] = 1 - \prod_{j=1}^{\ell} \mathbf{P}\left[\ y_j \text{ is false }\ \right] = 1 - \prod_{j=1}^{\ell} \left(1 - \overline{y}_j\right)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \times \ldots \times a_k}.$$

$$\geq 1 - \left(\frac{\sum_{j=1}^{\ell}(1 - \overline{y}_j)}{\ell}\right)^{\ell}$$

$$= 1 - \left(1 - \frac{\sum_{j=1}^{\ell} \overline{y}_j}{\ell}\right)^{\ell} \geq 1 - \left(1 - \frac{\overline{z}_i}{\ell}\right)^{\ell}.$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

$$\mathbf{P}\left[\text{clause } i \text{ is satisfied}\right] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot \overline{z}_i.$$

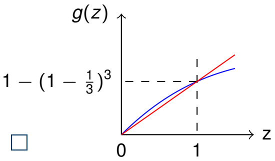
Proof of Lemma (2/2):

- So far we have shown:

$$\mathbf{P}\left[\text{clause } i \text{ is satisfied}\right] \geq 1 - \left(1 - \frac{\overline{z}_i}{\ell}\right)^{\ell}$$

- 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 $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^{\ell} =: \beta_\ell$.

$$\Rightarrow \quad g(z) \geq \beta_\ell \cdot z \qquad \text{for any } z \in [0,1]$$



- Therefore, $\mathbf{P}\left[\text{clause } i \text{ is satisfied}\right] \geq \beta_\ell \cdot \overline{z}_i$. □

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

$$\mathbf{P}\left[\,\text{clause } i \text{ is satisfied}\,\right] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot \overline{z}_i.$$

Theorem

Randomised Rounding yields a $1/(1-1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \ldots, m$, let ℓ_i be the corresponding length.
- Then the expected number of satisfied clauses is:

$$\mathbf{E}\left[\,Y\,\right] = \sum_{i=1}^{m} \mathbf{E}\left[\,Y_i\,\right] \geq \sum_{i=1}^{m} \left(1 - \left(1 - \frac{1}{\ell_i}\right)^{\ell_i}\right) \cdot \overline{z}_i \geq \sum_{i=1}^{m} \left(1 - \frac{1}{e}\right) \cdot \overline{z}_i \geq \left(1 - \frac{1}{e}\right) \cdot \text{OPT}$$

By Lemma

Since $(1 - 1/x)^x \leq 1/e$

LP solution at least as good as optimum

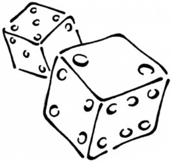
Approach 3: Hybrid Algorithm

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($\varphi$, $n$, $m$)
Let $b \in \{0, 1\}$ be the flip of a fair coin
If $b = 0$ then perform random guessing
If $b = 1$ then perform randomised rounding
return the computed solution
```



Algorithm sets each variable x_i to TRUE with prob. $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \overline{y}_i$.
Note, however, that variables are **not** independently assigned!

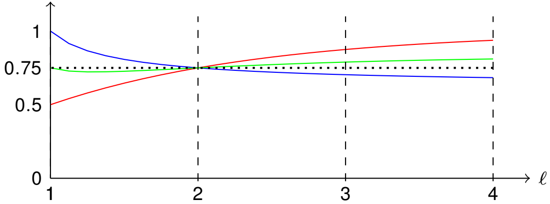
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, n, m) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause i is satisfied with probability at least $3/4 \cdot \overline{z}_i$
- For any clause i of length ℓ:
  - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \overline{z}_i$.
  - Algorithm 2 satisfies it with probability $\beta_\ell \cdot \overline{z}_i$.
  - HYBRID-MAX-CNF(φ, n, m) satisfies it with probability $\frac{1}{2} \cdot \alpha_\ell \cdot \overline{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \overline{z}_i$.
- 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 $3/4 \cdot \overline{z}_i$ ☐



MAX-CNF Conclusion

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!

Outline

Weighted Set Cover

MAX-CNF

Appendix: An Approximation Algorithm of TSP (non-examin.)

Metric TSP (TSP Problem with the Triangle Inequality)

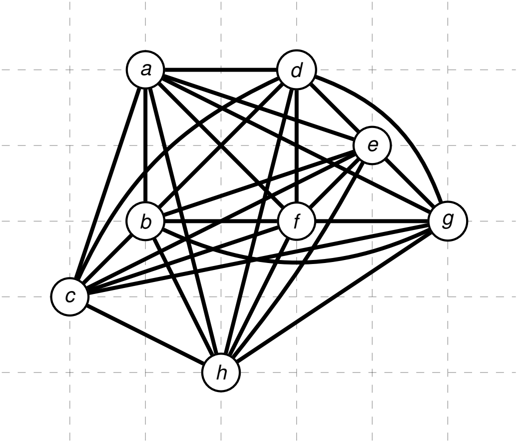
Idea: First compute an MST, and then create a tour based on the tree.

```
APPROX-TSP-TOUR($G, c$)
select a vertex $r \in G.V$ to be a “root” vertex
compute a minimum spanning tree $T_{\min}$ for $G$ from root $r$
        using MST-PRIM($G, c, r$)
let $H$ be a list of vertices, ordered according to when they are first visited
        in a preorder walk of $T_{\min}$
return the hamiltonian cycle $H$
```

Runtime is dominated by MST-PRIM, which is $\Theta(V^2)$.

Remember: In the Metric-TSP problem, G is a complete graph.

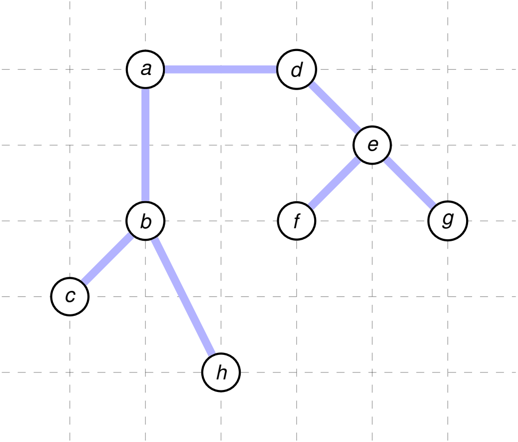
Run of APPROX-TSP-TOUR



1. Compute MST T_{min}

.1

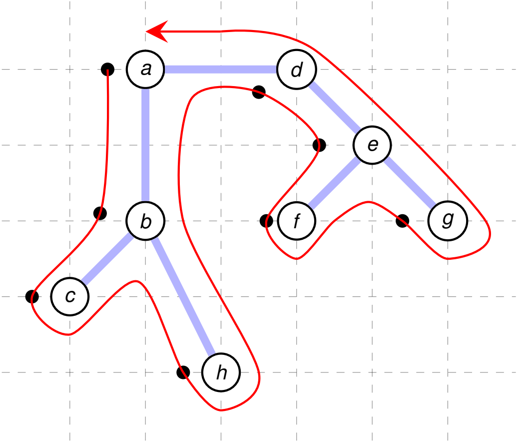
Run of APPROX-TSP-TOUR



1. Compute MST T_{min} ✓
2. Perform preorder walk on MST T_{min}

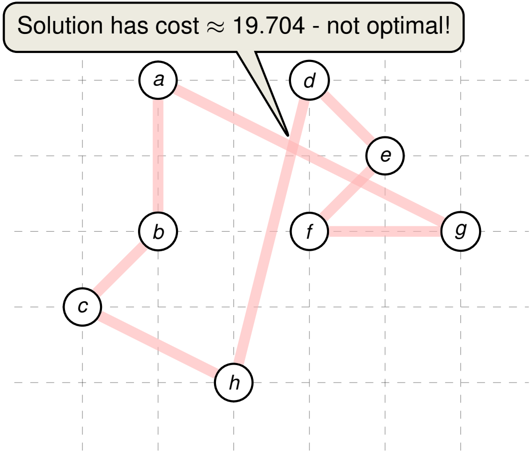
.2

Run of APPROX-TSP-TOUR



1. Compute MST $T_{\min}$ ✓
2. Perform preorder walk on MST $T_{\min}$ ✓
3. Return list of vertices according to the preorder tree walk

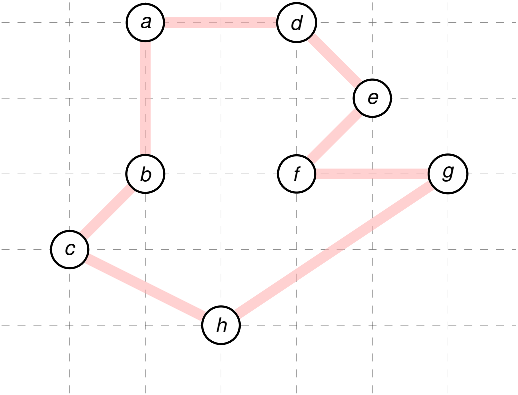
Run of APPROX-TSP-TOUR


Solution has cost ≈ 19.704 - not optimal!

1. Compute MST $T_{\min}$ ✓
2. Perform preorder walk on MST $T_{\min}$ ✓
3. Return list of vertices according to the preorder tree walk ✓

Run of APPROX-TSP-TOUR

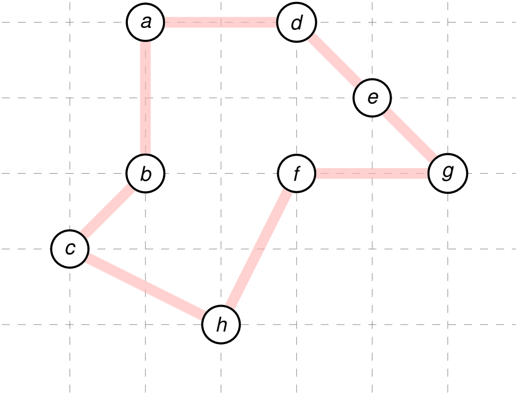
Better solution, yet still not optimal!



1. Compute MST $T_{\min}$ ✓
2. Perform preorder walk on MST $T_{\min}$ ✓
3. Return list of vertices according to the preorder tree walk ✓

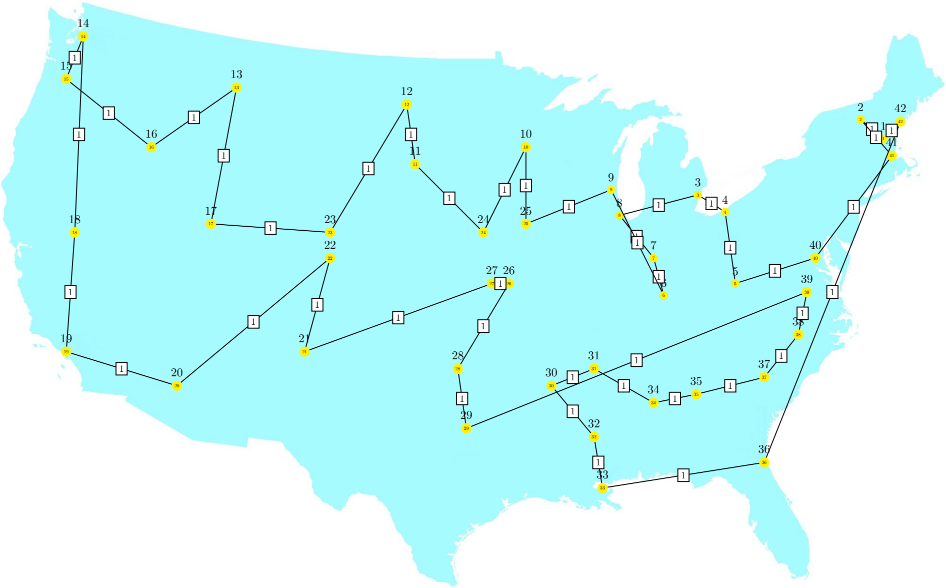
Run of APPROX-TSP-TOUR

This is the optimal solution (cost $\approx$ 14.715).

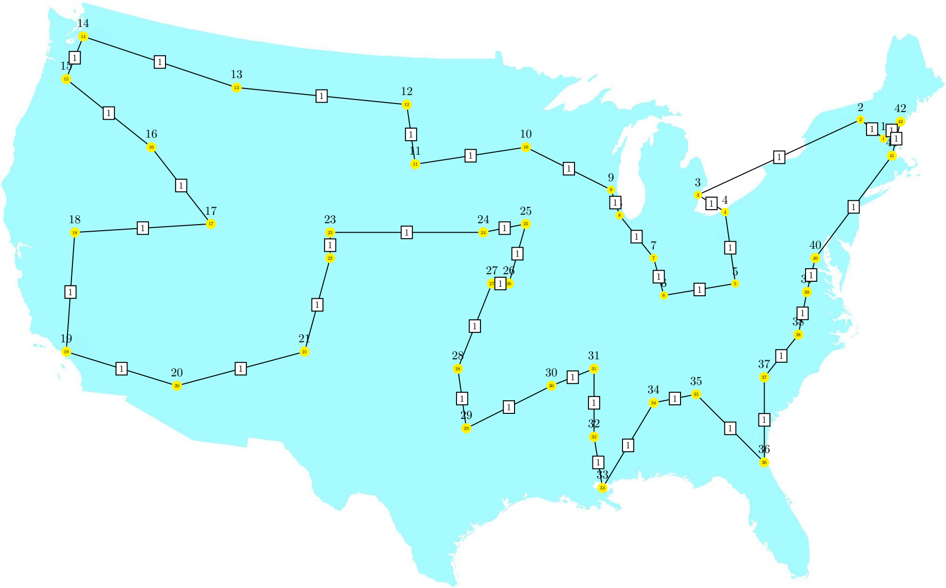


1. Compute MST $T_{\min}$ ✓
2. Perform preorder walk on MST $T_{\min}$ ✓
3. Return list of vertices according to the preorder tree walk ✓

Approximate Solution: Objective 921



Optimal Solution: Objective 699



Proof of the Approximation Ratio

Theorem 35.2

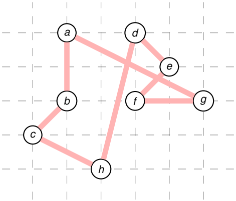
APPROX-TSP-TOUR is a polynomial-time 2-approximation for the traveling-salesman problem with the triangle inequality.

Proof:

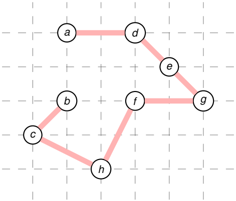
- Consider the optimal tour H^* and remove an arbitrary edge

$\Rightarrow$ yields a spanning tree T and $c(T_{\min}) \leq c(T) \leq c(H^*)$

exploiting that all edge costs are non-negative!



solution H of APPROX-TSP



spanning tree T as a subset of H^*

Proof of the Approximation Ratio

Theorem 35.2

APPROX-TSP-TOUR is a polynomial-time 2-approximation for the traveling-salesman problem with the triangle inequality.

Proof:

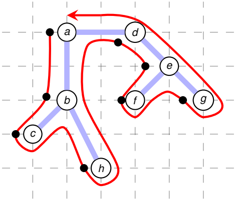
- Consider the optimal tour H^* and remove an arbitrary edge

$\Rightarrow$ yields a spanning tree T and $c(T_{\min}) \leq c(T) \leq c(H^*)$

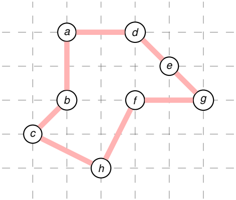
- Let W be the full walk of the minimum spanning tree $T_{\min}$ (including repeated visits)

$\Rightarrow$ Full walk traverses every edge exactly twice, so

$$c(W) = 2c(T_{\min}) \leq 2c(T) \leq 2c(H^*)$$



Walk $W = (a, b, c, b, h, b, a, d, e, f, e, g, e, d, a)$



optimal solution H^*

Proof of the Approximation Ratio

Theorem 35.2

APPROX-TSP-TOUR is a polynomial-time 2-approximation for the traveling-salesman problem with the triangle inequality.

Proof:

- Consider the optimal tour H^* and remove an arbitrary edge

$\Rightarrow$ yields a spanning tree T and $c(T_{\min}) \leq c(T) \leq c(H^*)$

- Let W be the full walk of the minimum spanning tree $T_{\min}$ (including repeated visits)

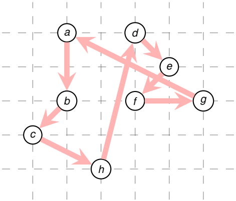
$\Rightarrow$ Full walk traverses every edge exactly twice, so

$$c(W) = 2c(T_{\min}) \leq 2c(T) \leq 2c(H^*)$$

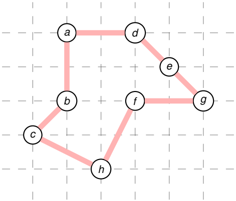
exploiting triangle inequality!

- Deleting duplicate vertices from W yields a tour H with smaller cost:

$$c(H) \leq c(W) \leq 2c(H^*) \qquad \square$$



Walk $W = (a, b, c, \cancel{b}, h, \cancel{b}, \cancel{a}, d, e, f, \cancel{e}, g, \cancel{e}, \cancel{d}, a)$



optimal solution H^*

Christofides Algorithm

Theorem 35.2

APPROX-TSP-TOUR is a polynomial-time 2-approximation for the traveling-salesman problem with the triangle inequality.

Can we get a better approximation ratio?

```
CHRISTOFIDES($G, c$)
select a vertex $r \in G.V$ to be a "root" vertex
compute a minimum spanning tree $T_{\min}$ for $G$ from root $r$
        using MST-PRIM($G, c, r$)
compute a perfect matching $M_{\min}$ with minimum weight in the complete graph
        over the odd-degree vertices in $T_{\min}$
let $H$ be a list of vertices, ordered according to when they are first visited
        in a Eulearian circuit of $T_{\min} \cup M_{\min}$
return the hamiltonian cycle $H$
```

Theorem (Christofides'76)

There is a polynomial-time $\frac{3}{2}$-approximation algorithm for the travelling salesman problem with the triangle inequality.

Randomised Algorithms

Lecture 11: Spectral Graph Theory

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

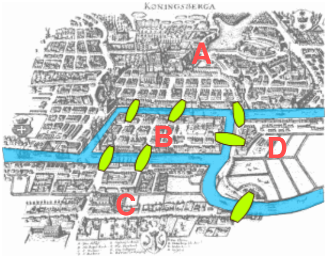
Outline

Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem

Origin of Graph Theory

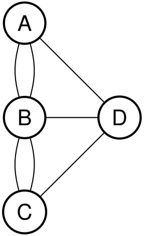


Source: Wikipedia

Seven Bridges at Königsberg 1737

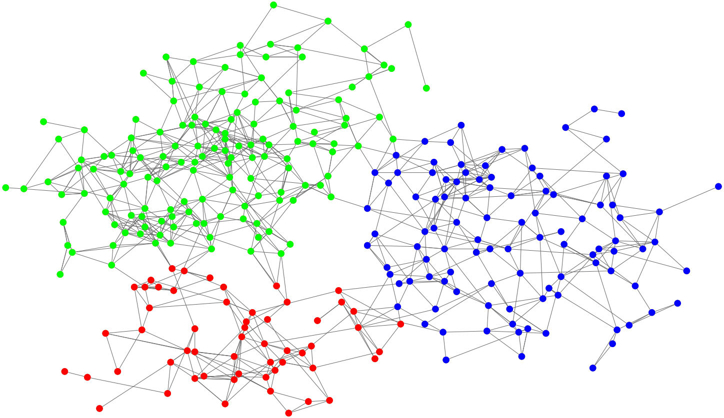
Source: Wikipedia

Leonhard Euler (1707-1783)



Is there a tour which crosses each bridge **exactly once**?

Graphs Nowadays: Clustering



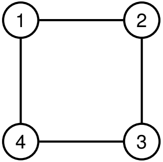
Goal: Use spectrum of graphs (unstructured data) to extract clustering (communitites) or other structural information.

Graph Clustering (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.
  - Graph Clustering: partition vertices in a graph
    - modularity, conductance, min-cut, etc.

Graphs and Matrices

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
- . . .

Outline

Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

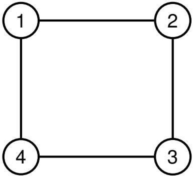
A Simplified Clustering Problem

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

$$\mathbf{A}_{u,v} = \begin{cases} 1 & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

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 $x \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ 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

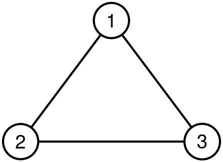
Let $\mathbf{A}$ be the adjacency matrix of a d-regular graph G with n vertices. Then, $\mathbf{A}$ has n real eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and n corresponding orthonormal eigenvectors $f_1, \ldots, f_n$. These eigenvalues associated with their multiplicities constitute the spectrum of G.

For symmetric matrices: algebraic multiplicity $=$ geometric multiplicity

Example 1

Bonus: Can you find a short-cut to $\det(\mathbf{A} - \lambda \cdot \mathbf{I})$?

Exercise: What are the Eigenvalues and Eigenvectors?

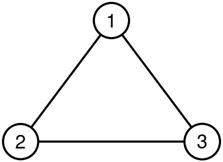


$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Example 1

Bonus: Can you find a short-cut to $\det(\mathbf{A} - \lambda \cdot \mathbf{I})$?

Exercise: 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):

$$f_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad f_2 = \begin{pmatrix} -\frac{1}{2} \\ 1 \\ -\frac{1}{2} \end{pmatrix}, \quad f_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

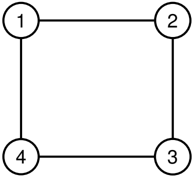
Laplacian Matrix

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.



$$\mathbf{L} = \begin{pmatrix} 1 & -1/2 & 0 & -1/2 \\ -1/2 & 1 & -1/2 & 0 \\ 0 & -1/2 & 1 & -1/2 \\ -1/2 & 0 & -1/2 & 1 \end{pmatrix}$$

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 eigenvectors.

Exercise: Proof this correspondence. Hint: Use that $\mathbf{L} = \mathbf{I} - \frac{1}{d}\mathbf{A}$.

Eigenvalues and Graph Spectrum of L

Eigenvalues and eigenvectors

Let **M** $\in \mathbb{R}^{n\times n}$, $\lambda \in \mathbb{C}$ is an eigenvalue of **M** if and only if there exists $x \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbf{M}x = \lambda x.$$

We call x an eigenvector of **M** corresponding to the eigenvalue λ.

Graph Spectrum

Let **L** be the Laplacian matrix of a d-regular graph G with n vertices. Then, **L** has n real eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and n corresponding orthonormal eigenvectors $f_1, \ldots, f_n$.

Useful Facts of Graph Spectrum

Lemma

Let $\mathbf{L}$ be the Laplacian matrix of an undirected, regular graph $G = (V, E)$ with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$.

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

Let **M** be an n by n symmetric matrix with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$. Then,

$$\lambda_k = \min_{\substack{x^{(1)},\ldots,x^{(k)} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \\ x^{(i)} \perp x^{(j)}}} \max_{i \in \{1,\ldots,k\}} \frac{{x^{(i)}}^T \mathbf{M} x^{(i)}}{{x^{(i)}}^T x^{(i)}}.$$

The eigenvectors corresponding to $\lambda_1, \ldots, \lambda_k$ minimise such expression.

$$\lambda_1 = \min_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{x^T \mathbf{M} x}{x^T x}$$

minimised by an eigenvector f_1 for λ_1

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ x \perp f_1}} \frac{x^T \mathbf{M} x}{x^T x}$$

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 $x \in \mathbb{R}^n$,

$$x^T \mathbf{L} x = \sum_{\{u,v\} \in E} \frac{(x_u - x_v)^2}{d}.$$

Proof:

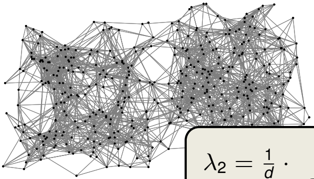
$$\begin{aligned} x^T \mathbf{L} x = x^T \left(\mathbf{I} - \frac{1}{d} \mathbf{A} \right) x &= x^T x - \frac{1}{d} x^T \mathbf{A} x \\ &= \sum_{u \in V} x_u^2 - \frac{2}{d} \sum_{\{u,v\} \in E} x_u x_v \\ &= \frac{1}{d} \sum_{\{u,v\} \in E} (x_u^2 + x_v^2 - 2 x_u x_v) \\ &= \sum_{\{u,v\} \in E} \frac{(x_u - x_v)^2}{d}. \end{aligned}$$

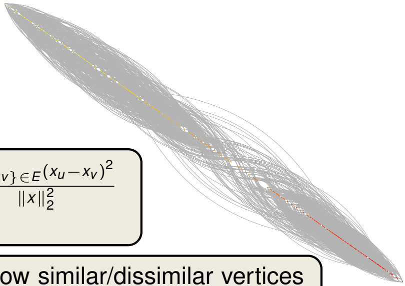
Visualising a Graph

Question: How can we visualize a complicated object like an unknown graph with many vertices in low-dimensional space?

Embedding onto Line

Coordinates given by x





$$\lambda_2 = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ x \perp f_1}} \frac{\sum_{\{u,v\} \in E} (x_u - x_v)^2}{\|x\|_2^2}$$

The coordinates in the vector **x** indicate how similar/dissimilar vertices are. Edges between dissimilar vertices are penalised quadratically.

Outline

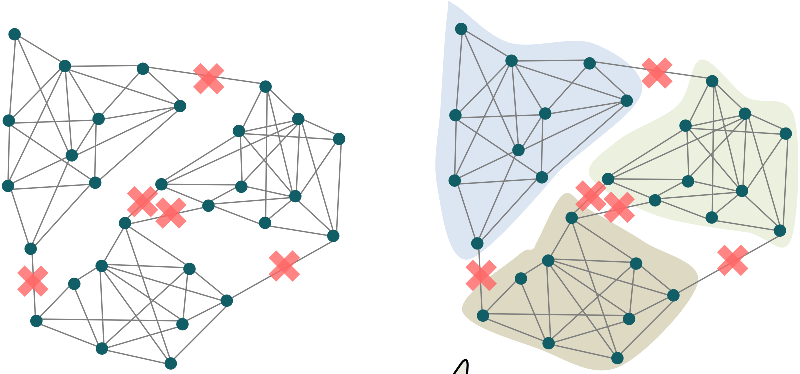
Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem

A Simplified Clustering Problem

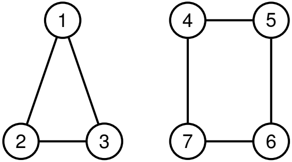
Partition the graph into ***connected components*** *so that any pair of vertices in the same component is connected, but vertices in different components are not.*



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

Exercise: What are the Eigenvectors with Eigenvalue 0 of $\mathbf{L}$?

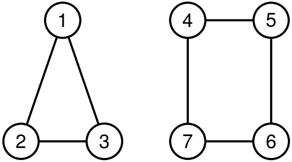


$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Example 2

Exercise: 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:

- The two smallest eigenvalues are $\lambda_1 = \lambda_2 = 0$.
- The corresponding two eigenvectors are:

Thus we can easily solve the simplified clustering pro
lem by computing the eigenvectors with eigenvalue

$$f_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad f_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \Big(\text{or} \quad f_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad f_2 = \begin{pmatrix} -1/3 \\ -1/3 \\ -1/3 \\ 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} \Big)$$

Next Lecture: A fine-grained
approach works even if the clus-
ters are **sparsely** connected!

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. ("$\Longrightarrow$" $cc(G) \leq \mathrm{mult}(0)$). We will show:
   G has exactly k connected comp. $C_1, \ldots, C_k \quad \Rightarrow \quad \lambda_1 = \cdots = \lambda_k = 0$
   - 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
   - $\chi_{C_i}^T \mathbf{L} \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u,v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0 \quad \Rightarrow \quad \lambda_1 = \cdots = \lambda_k = 0$

2. ("$\Longleftarrow$" $cc(G) \geq \mathrm{mult}(0)$). We will show:
   $\lambda_1 = \cdots = \lambda_k = 0 \quad \Rightarrow \quad G$ has at least k connected comp. $C_1, \ldots, C_k$
   - there exist $f_1, \ldots, f_k$ orthonormal such that $\sum_{\{u,v\} \in E} (f_i(u) - f_i(v))^2 = 0$
   - $\Rightarrow$ $f_1, \ldots, f_k$ constant on connected components
   - as $f_1, \ldots, f_k$ are pairwise orthogonal, G must have k different connected components.

$\square$

Randomised Algorithms

Lecture 12: Spectral Graph Clustering

Thomas Sauerwald (`tms41@cam.ac.uk`)

Lent 2023

Outline

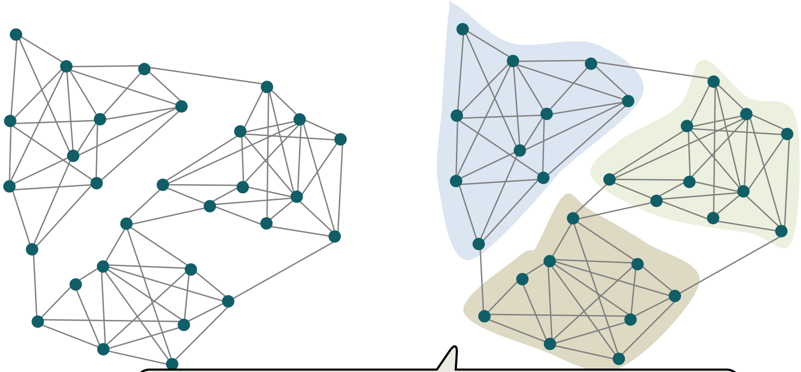
Conductance, Cheeger’s Inequality and Spectral Clustering

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*



Let us for simplicity focus on the case of **two clusters**!

Conductance

Conductance

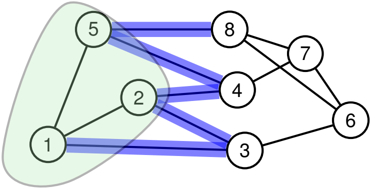
Let $G = (V, E)$ be a d-regular and undirected graph and $\emptyset \neq S \subsetneq V$. The conductance (edge expansion) of S is

$$\phi(S) := \frac{e(S, S^c)}{d \cdot |S|}$$

Moreover, the conductance (edge expansion) of the graph G is

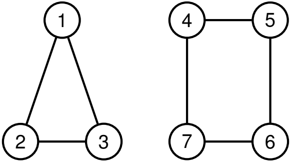
$$\phi(G) := \min_{S \subseteq V\colon\, 1 \leq |S| \leq n/2} \phi(S)$$

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 $\phi(G) \approx 1/2$.

λ_2 versus Conductance (1/2)



$$\phi(G) = 0 \quad \Leftrightarrow \quad G \text{ is disconnected} \quad \Leftrightarrow \quad \lambda_2(G) = 0$$

What is the relationship between $\phi(G)$ and $\lambda_2(G)$ for **connected** graphs?

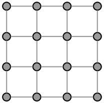
λ_2 versus Conductance (2/2)

1D Grid (Path)

$\lambda_2 \sim n^{-2}$

$\phi \sim n^{-1}$

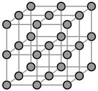
2D Grid



$\lambda_2 \sim n^{-1}$

$\phi \sim n^{-1/2}$

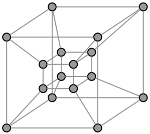
3D Grid



$\lambda_2 \sim n^{-2/3}$

$\phi \sim n^{-1/3}$

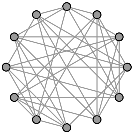
Hypercube



$\lambda_2 \sim (\log n)^{-1}$

$\phi \sim (\log n)^{-1}$

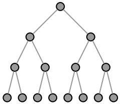
Random Graph (Expanders)



$\lambda_2 = \Theta(1)$

$\phi = \Theta(1)$

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 $\lambda_1 \leq \cdots \leq \lambda_n$ 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 $x_1 \leq x_2 \leq \cdots \leq x_n$ (embed V on $\mathbb{R}$)
3. Try all $n-1$ sweep cuts of the form $(\{1,2,\ldots,k\},\{k+1,\ldots,n\})$ 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):

Optimisation Problem: Embed vertices on a line such that sum of squared distances is minimised

- By the Courant-Fischer Formula,

$$\lambda_2 = \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{x^T \mathbf{L} x}{x^T x} = \frac{1}{d} \cdot \min_{\substack{x \in \mathbb{R}^n \\ x \neq 0, x \perp 1}} \frac{\sum_{u \sim v} (x_u - x_v)^2}{\sum_u x_u^2}.$$

- 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

$$\lambda_2 \leq \frac{1}{d} \cdot \frac{\sum_{u \sim v} (y_u - y_v)^2}{\sum_u y_u^2} = \frac{1}{d} \cdot \frac{|E(S, V \setminus S)| \cdot (\frac{1}{|S|} + \frac{1}{|V \setminus S|})^2}{\frac{1}{|S|} + \frac{1}{|V \setminus S|}}$$

$$= \frac{1}{d} \cdot |E(S, V \setminus S)| \cdot \left(\frac{1}{|S|} + \frac{1}{|V \setminus S|} \right)$$

$$\leq \frac{1}{d} \cdot \frac{2 \cdot |E(S, V \setminus S)|}{|S|} = 2 \cdot \phi(G). \qquad \square$$

Outline

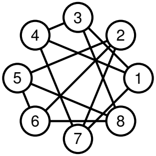
Conductance, Cheeger’s Inequality and Spectral Clustering

Illustrations of Spectral Clustering and Extension to Non-Regular Graphs

Appendix: Relating Spectrum to Mixing Times (non-examinable)

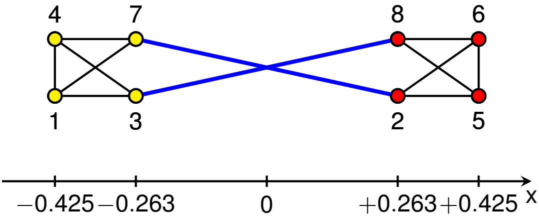
Illustration on a small Example

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{L} = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\ -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1 \end{pmatrix}$$



$$\lambda_2 = 1 - \sqrt{5}/3 \approx 0.25$$

$$v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T$$



Sweep: 4

Conductance: 0.166

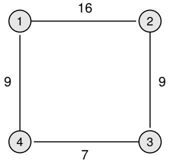
Let us now look at an example of a non-regular graph!

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 s.t. $\mathbf{D}_{uu} = deg(u) = \sum_{\{u,v\} \in E} w(u,v)$, 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

$$\phi(S) := \frac{w(S, S^c)}{\min\{\mathrm{vol}(S), \mathrm{vol}(S^c)\}},$$

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\mathrm{vol}(S) := \sum_{u \in S} d(u)$. 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 $y_1 \leq y_2 \leq \cdots \leq y_n$ (embed V on $\mathbb{R}$)
3. Try all $n-1$ sweep cuts of the form $(\{1, 2, \ldots, k\}, \{k+1, \ldots, n\})$ and return the one with smallest conductance

Stochastic Block Model and 1D-Embedding

Stochastic Block Model

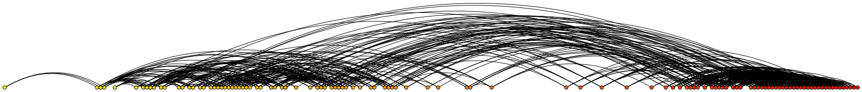
$G = (V, E)$ with clusters $S_1, S_2 \subseteq V$, $0 \leq q < p \leq 1$

$$\mathbf{P}[\{u, v\} \in E] = \begin{cases} p & \text{if } u, v \in S_i, \\ q & \text{if } u \in S_i, v \in S_j, i \neq j. \end{cases}$$

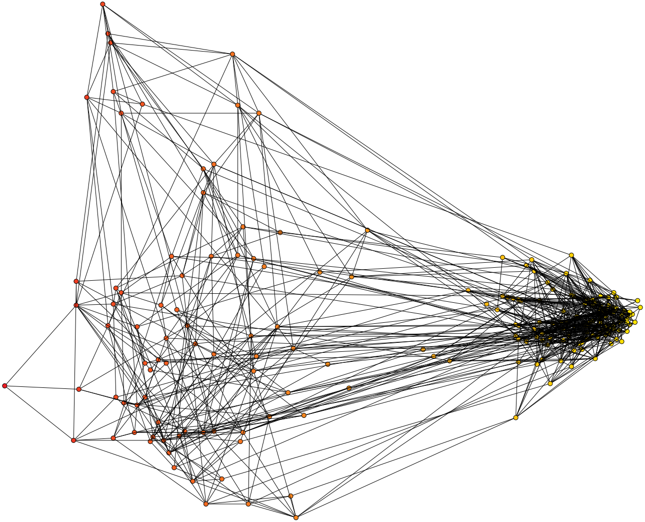
Here:

- $|S_1| = 80$, $|S_2| = 120$
- $p = 0.08$
- $q = 0.01$

```
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
```



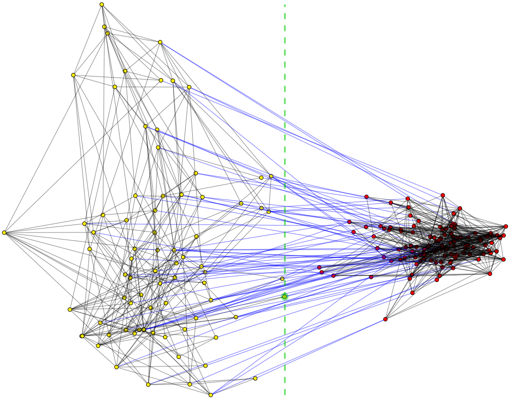
Drawing the 2D-Embedding



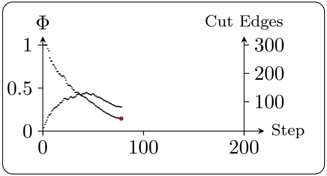
Spectral Clustering

For the complete animation, see the full slides.

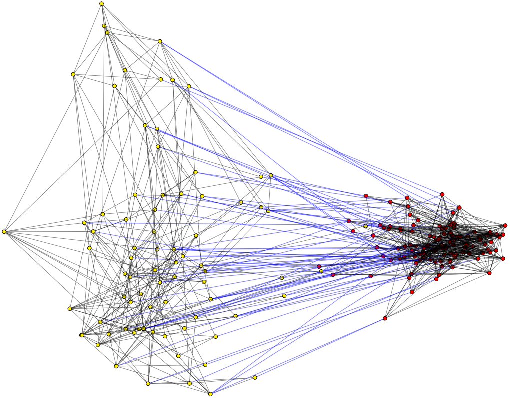
Best Solution found by Spectral Clustering



- Step: 78
- Threshold: −0.0268
- Partition Sizes: 78/122
- Cut Edges: 84
- Conductance: 0.1448



Clustering induced by Blocks



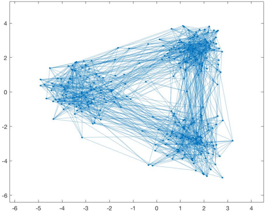
- Step: 1
- Threshold: 0
- Partition Sizes: 80/120
- Cut Edges: 88
- Conductance: 0.1486

Additional Example: Stochastic Block Models with 3 Clusters

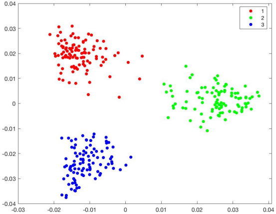
Graph $G = (V, E)$ with clusters
$S_1, S_2, S_3 \subseteq V; \quad 0 \leq q < p \leq 1$

$$\mathbf{P}\left[\,\{u, v\} \in E\,\right] = \begin{cases} p & u, v \in S_i \\ q & u \in S_i, v \in S_j, i \neq j \end{cases}$$

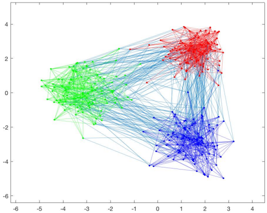
$|V| = 300, |S_i| = 100$
$p = 0.08, q = 0.01.$



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: $\lambda_1 = \ldots = \lambda_k = 0$ means there are k connected components)
  - large λ_{k+1} means all these k subsets have "good" inner-connectivity properties

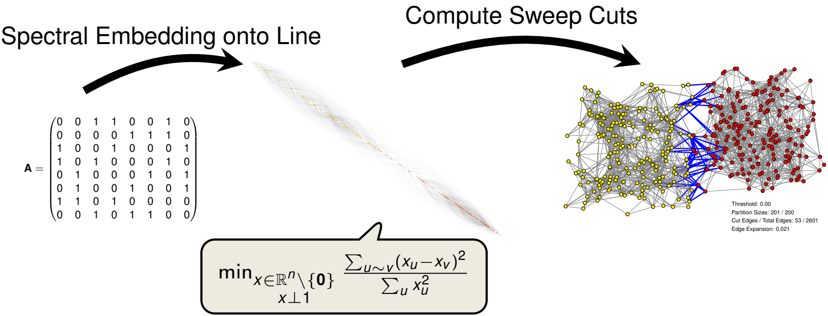
  $\Rightarrow$ choose smallest $k \geq 2$ so that the spectral gap $\lambda_{k+1} - \lambda_k$ is "large"

- In the latter example $\lambda = \{0, 0.20, 0.22, 0.43, 0.45, \ldots\} \implies k = 3$.

- In the former example $\lambda = \{0, 0.15, 0.37, 0.40, 0.43, \ldots\} \implies k = 2$.

- For $k = 2$ use sweep-cut extract clusters. For $k \geq 3$ use embedding in k-dimensional space and apply k-means (geometric clustering)

Summary: Spectral Clustering



Spectral Embedding onto Line

Compute Sweep Cuts

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$\min_{\substack{x\in\mathbb{R}^n\setminus\{\mathbf{0}\} \\ x\perp 1}} \frac{\sum_{u\sim v}(x_u-x_v)^2}{\sum_u x_u^2}$$

- 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

Outline

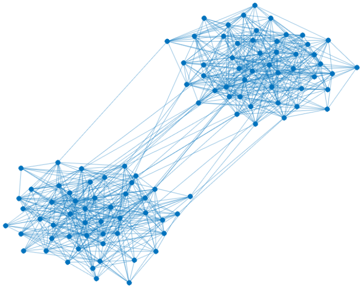
Conductance, Cheeger's Inequality and Spectral Clustering

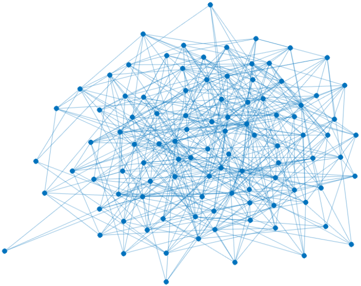
Illustrations of Spectral Clustering and Extension to Non-Regular Graphs

Appendix: Relating Spectrum to Mixing Times (non-examinable)

Relation between Clustering and Mixing

- Which graph has a “cluster-structure”?
- Which graph mixes faster?





Convergence of Random Walk

Recall: If the underlying graph G is connected, undirected and d-regular, then the random walk converges towards the stationary distribution $\pi = (1/n, \ldots, 1/n)$, 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,

$$\left\| x\mathbf{P}^t - \pi \right\|_2 \leq \lambda^t,$$

with $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$ as eigenvalues and $\lambda := \max\{|\lambda_2|, |\lambda_n|\}$.

$\Rightarrow$ This implies for $t = \mathcal{O}\left(\frac{\log n}{\log(1/\lambda)}\right) = \mathcal{O}\left(\frac{\log n}{1-\lambda}\right)$,

$$\left\| x\mathbf{P}^t - \pi \right\|_{tv} \leq \frac{1}{4}.$$

due to laziness, $\lambda_n \geq 0$

Proof of Lemma

- Express x in terms of the orthonormal basis of $\mathbf{P}$, $v_1 = \pi, v_2, \ldots, v_n$:

$$x = \sum_{i=1}^{n} \alpha_i v_i.$$

- Since x is a probability vector and all $v_i \geq 2$ are orthogonal to π, $\alpha_1 = 1$.

$\Rightarrow$

$$\begin{aligned}
\|x\mathbf{P} - \pi\|_2^2 &= \left\| \Big(\sum_{i=1}^{n} \alpha_i v_i \Big) \mathbf{P} - \pi \right\|_2^2 \\
&= \left\| \pi + \sum_{i=2}^{n} \alpha_i \lambda_i v_i - \pi \right\|_2^2 \\
&= \left\| \sum_{i=2}^{n} \alpha_i \lambda_i v_i \right\|_2^2 \\
&= \sum_{i=2}^{n} \|\alpha_i \lambda_i v_i\|_2^2 \\
&\leq \lambda^2 \sum_{i=2}^{n} \|\alpha_i v_i\|_2^2 = \lambda^2 \left\| \sum_{i=2}^{n} \alpha_i v_i \right\|_2^2 = \lambda^2 \|x - \pi\|_2^2
\end{aligned}$$

since the v_i's are orthogonal

since the v_i's are orthogonal

- Hence $\|x\mathbf{P}^t - \pi\|_2^2 \leq \lambda^{2t} \cdot \|x - \pi\|_2^2 \leq \lambda^{2t} \cdot 1$.

$\|x - \pi\|_2^2 + \|\pi\|_2^2 = \|x\|_2^2 \leq 1$

References

- 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. Widgerson.
  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`

The End...

Thank you and Best Wishes for the Exam!