Randomised Algorithms
Lecture 1: Introduction to Course & Introduction to Chernoff Bounds

Thomas Sauerwald (tms41@cam.ac.uk)
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 often: simple algorithm at the cost of a sophisticated 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.
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


(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 \((\Omega, \Sigma, P)\).

Components of the Probability Space \((\Omega, \Sigma, P)\)

- **The Sample Space** \(\Omega\) contains all the possible outcomes \(\omega_1, \omega_2, \ldots\) of the experiment.
- **The Event Space** \(\Sigma\) is the power-set of \(\Omega\) containing events, which are combinations of outcomes (subsets of \(\Omega\) including \(\emptyset\) and \(\Omega\)).
- **The Probability Measure** \(P\) is a function from \(\Sigma\) to \(\mathbb{R}\) satisfying
  
  (i) \(0 \leq P[\mathcal{E}] \leq 1\), for all \(\mathcal{E} \in \Sigma\)
  
  (ii) \(P[\Omega] = 1\)
  
  (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

  \[
  P \left[ \bigcup_{i=1}^{\infty} \mathcal{E}_i \right] = \sum_{i=1}^{\infty} P[\mathcal{E}_i].
  \]
Recap: Random Variables

A random variable \( X \) on \((\Omega, \Sigma, P)\) 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** \( 1_E \) of an event \( E \in \Sigma \) given by
  
  \[
  1_E(\omega) = \begin{cases} 
  1 & \text{if } \omega \in E \\ 
  0 & \text{otherwise}. 
  \end{cases}
  \]

  For the indicator random variable \( 1_E \) we have \( E[1_E] = P[E] \).

- **The number of sixes** of two dice throws \( X_1, X_2 \in \{1, 2, \ldots, 6\} \) is
  
  \[
  1_{X_1=6} + 1_{X_2=6}
  \]
Recap: Boole’s Inequality (Union Bound)

Let \( E_1, \ldots, E_n \) be a collection of events in \( \Sigma \). Then

\[
P \left[ \bigcup_{i=1}^{n} E_i \right] \leq \sum_{i=1}^{n} P[ E_i ].
\]

A Proof using Indicator Random Variables:
1. Let \( 1_{E_i} \) be the random variable that takes value 1 if \( E_i \) holds, 0 otherwise
2. \( E \left[ 1_{E_i} \right] = P \left[ E_i \right] \) (Check this)
3. It is clear that \( 1_{\bigcup_{i=1}^{n} E_i} \leq \sum_{i=1}^{n} 1_{E_i} \) (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 $e(S, S^c) := |E(S, S^c)|$ is maximised.

Applications:

- network design, VLSI design
- clustering, statistical physics

Comments:

- This problem 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!
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!

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

More details on approximation algorithms from Lecture 9 onwards!

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

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

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

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

$$= \sum_{\{u,v\} \in E} P\left[\{u \in S, v \in S^c\} \cup \{u \in S^c, v \in S\}\right]$$

$$= 2 \sum_{\{u,v\} \in E} P\left[u \in S, v \in S^c\right] = 2 \sum_{\{u,v\} \in E} P\left[u \in S\right] \cdot P\left[v \in S^c\right] = |E|/2.$$  

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

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, 2, 5, 4, 2, 4, 1, 4, 2, 1, 4, 3, 1, 4, 8 ✓

Exercise ([Ex. 1.11])

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 $x$.
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 (1952)

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

\[
(1 + \delta)\mu (1 - \delta)\mu \mu
\]

Hermann Chernoff (1923-)

1. Introduction © T. Sauerwald

Introduction to Chernoff Bounds
Recap: Markov and Chebyshev

---

**Markov’s Inequality**

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

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

---

**Chebyshev’s Inequality**

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

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

---

- Let $f : \mathbb{R} \rightarrow [0, \infty)$ and increasing, then $f(X) \geq 0$, and thus
  $$
P [ X \geq a ] \leq P [ f(X) \geq f(a) ] \leq E [ f(X) ] / f(a).
$$

- Similarly, if $g : \mathbb{R} \rightarrow [0, \infty)$ and decreasing, then $g(X) \geq 0$, and thus
  $$
P [ X \leq a ] \leq P [ g(X) \geq g(a) ] \leq E [ g(X) ] / g(a).
$$

Chebyshev’s inequality (or Markov) can be obtained by choosing $f(X) := (X - \mu)^2$ (or $f(X) := X$, respectively).
Markov and Chebyshev use the first and second moment of the random variable. Can we keep going?

- Yes!

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

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

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

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

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

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 $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then, for any $\delta > 0$ it holds that

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

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

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

While (\star) is one of the easiest (and most generic) Chernoff bounds to derive, the bound is complicated and hard to apply...
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 \( E[X] = n \cdot 1/2 = n/2 \)
- The Chernoff Bound gives for any \( \delta > 0 \),

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

The above expression equals 1 only for \( \delta = 0 \), and then it gives a value strictly less than 1 (check this!)

\( \Rightarrow \) The inequality is exponential in \( n \), (for fixed \( \delta \)) which is much better than Chebyshev’s inequality.

What about a concrete value of \( n \), say \( n = 100 \)?
Example: Coin Flips (2/3)

\[ P[Bin(100, 1/2) = x] \]
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:**
  
  \[
P \left[ X \geq 3/2 \cdot E \left[ X \right] \right] \leq 2/3 = 0.666.
  \]

- **Chebyshev’s inequality:**
  
  \[
P \left[ |X - \mu| \geq t \right] \leq \frac{V \left[ X \right]}{t^2},
  \]

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

- **Chernoff bound:**

  setting $\delta = 1/2$ gives

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

- **Remark:** The exact probability is $0.00000028 \ldots$

Chernoff bound yields a much better result (but needs independence!)
Randomised Algorithms
Lecture 2: Concentration Inequalities, Application to Balls-into-Bins

Thomas Sauerwald (tms41@cam.ac.uk)
How to Derive Chernoff Bounds

Application 1: Balls into Bins
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.

2. Compute an upper bound for $\mathbb{E} [ e^{\lambda X} ]$ (using independence).

3. Optimise value of $\lambda$ to obtain best tail bound.
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 $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then, for any $\delta > 0$ it holds that

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

Proof:

1. For $\lambda > 0$,

$$
\Pr[X \geq (1 + \delta)\mu] \leq \Pr[e^{\lambda X} \geq e^{\lambda(1+\delta)\mu}] \leq e^{-\lambda(1+\delta)\mu} \mathbb{E}[e^{\lambda X}]
$$

2. $\mathbb{E}[e^{\lambda X}] = \mathbb{E}[e^{\lambda \sum_{i=1}^{n} X_i}] = \prod_{i=1}^{n} \mathbb{E}[e^{\lambda X_i}]$

3. $\mathbb{E}[e^{\lambda X_i}] = e^{\lambda p_i + (1 - p_i)} = 1 + p_i(e^{\lambda} - 1) \leq e^{p_i(e^{\lambda} - 1)}$ for $1 + x \leq e^x$. 

Chernoff Bound: Proof
1. For $\lambda > 0$,

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

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

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

4. Putting all together

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

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 $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then, for any $0 < \delta < 1$ it holds that

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

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

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

Exercise on Supervision Sheet

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

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

- For all $t > 0$,
  \[
  \Pr[X \geq \mathbb{E}[X] + t] \leq e^{-\frac{2t^2}{n}}
  \]
  \[
  \Pr[X \leq \mathbb{E}[X] - t] \leq e^{-\frac{2t^2}{n}}
  \]

- For $0 < \delta < 1$,
  \[
  \Pr[X \geq (1 + \delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbb{E}[X]}{3}\right)
  \]
  \[
  \Pr[X \leq (1 - \delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbb{E}[X]}{2}\right)
  \]

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

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

How to Derive Chernoff Bounds

Application 1: Balls into Bins
Balls into Bins

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)

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 \( p_i = \Pr[X_i = 1] = 1/n \).
- The total balls in the bin is given by \( X := \sum_{i=1}^{n} X_i \).
- Since \( m = 2n \log n \), then \( \mu = \mathbb{E}[X] = 2 \log n \).
- By the Chernoff Bound,
  \[
  \Pr[X \geq 6 \log n] \leq e^{-\mu} \left(\frac{e \mu}{t}\right)^t \leq e^{-2 \log n} = n^{-2}
  \]
Let \( E_j := \{ X(j) \geq 6 \log n \} \), 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 \( \bigcup_{j=1}^{n} E_j \)

By the Union Bound,

\[
P \left[ \bigcup_{j=1}^{n} E_j \right] \leq \sum_{j=1}^{n} P[E_j] \leq n \cdot n^{-2} = n^{-1}.
\]

Therefore \textit{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.

\[\text{whp stands for with high probability:}\]
An event \( \mathcal{E} \) (that implicitly depends on an input parameter \( n \)) occurs \textit{whp} if
\[
P[\mathcal{E}] \to 1 \text{ as } n \to \infty.
\]
This is a very standard notation in randomised algorithms but it may vary from author to author. \textbf{Be careful!}
Question 2: How large is the maximum load if \( m = n \)?

- Using the Chernoff Bound:
  \[
  \Pr[X \geq t] \leq e^{-\mu}(e^{\mu}/t)^t
  \]

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

- Indeed:
  \[
  \left(\frac{e \log \log n}{4 \log n}\right)^{4 \log n/ \log \log n} = \exp\left(\frac{4 \log n}{\log \log n} \cdot \log \left(\frac{e \log \log n}{4 \log n}\right)\right)
  \]

- The term inside the exponential is
  \[
  \frac{4 \log n}{\log \log n} \cdot (\log(e/4) + \log \log \log n - \log \log n) \leq \frac{4 \log n}{\log \log n} \left(-\frac{1}{2} \log \log n\right),
  \]

obtaining that \( \Pr[X \geq t] \leq n^{-4/2} = n^{-2} \).

This inequality only works for large enough \( n \).
We just proved that
\[ P \left[ X \geq 4 \log n / \log \log n \right] \leq n^{-2}, \]
thus by the Union Bound, no bin receives more than \( \Omega \left( \log n / \log \log n \right) \) balls with probability at least \( 1 - 1/n \).

- As mentioned on the to prove that whp at least one bin receives at least \( c \log n / \log \log n \) balls, for some constant \( c > 0 \).
Conclusions

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

**A Better Load Balancing Approach**

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

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

This is called the **power of two choices**: It is a common technique to improve the performance of randomised algorithms (covered in Chapter 17 of the textbook by Mitzenmacher and Upfal)
For “the discovery and analysis of balanced allocations, known as the power of two choices, and their extensive applications to practice.”

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

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)
Outline

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: More on Moment Generating Functions (non-examinable)
QUICKSORT (Input \(A[1], A[2], \ldots, A[n]\))

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

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

- **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 depths of all nodes in the tree (why?). In this case:

\[0 + 1 + 1 + 2 + 2 + 3 + 3 + 3 + 4 = 19.\]
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 there exists $C > 0$ such that
   
   $$P \left[ H \leq Cn \log n \right] \geq 1 - n^{-1}.$$
4. Actually, we will prove something slightly stronger:
   
   $$P \left[ \bigcap_{i=1}^{n} \{ H_i \leq C \log n \} \right] \geq 1 - n^{-1}.$$
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:

$\begin{align*}
(2, 8, 9, 1, 7, 5, 6, 3, 4) &\rightarrow (2, 1, 5, 3, 4) \\
&\rightarrow (2, 5, 3, 4) \\
&\rightarrow (2, 3) \\
&\rightarrow (2)
\end{align*}$
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: $s_{k+1} \leq s_k$.

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

**Question:** Edge Case: What if the path $P$ does not reach level $j$?
Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ 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.

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

- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

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

**Answer:** We can then simply define $X_j$ as 0 (deterministically).
Randomised QuickSort: Analysis (4/4)

- Consider the first $24 \log n$ 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.

\[ P \left[ X_j = 1 \mid X_0 = x_0, \ldots, X_{j-1} = x_{j-1} \right] \leq \frac{2}{3} \]

- $X := \sum_{j=0}^{24 \log n - 1} X_j$ satisfies relaxed independence assumption (Lecture 2)

We can now apply the “nicer” Chernoff Bound!

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

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

\[ P \left[ X > 21 \log n \right] \leq P \left[ X > E[X] + 5 \log n \right] \]

- Hence $P$ has more than $24 \log n$ nodes with probability at most $n^{-2}$.
- As there are in total $n$ paths, by the union bound, the probability that at least one of them has more than $24 \log n$ nodes is at most $n^{-1}$.
- This implies $P \left[ \bigcap_{i=1}^{n} \{ H_i \leq 24 \log n \} \right] \geq 1 - n^{-1}$, as needed.
Randomised QuickSort: Final Remarks

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

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

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

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

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

**Hoeffding’s Extension Lemma**

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

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

You can always consider

$$
X' = X - E[X]
$$

We omit the proof of this lemma!
Hoeffding Bounds

Hoeffding’s Inequality

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

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

and

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

Proof Outline (skipped):

- Let \( X'_i = X_i - \mu_i \) and \( X' = X'_1 + \ldots + X'_n \), then \( P[X \geq \mu + t] = P[X' \geq t] \)
- \( P[X' \geq t] \leq e^{-\lambda t} \prod_{i=1}^{n} E\left[ e^{\lambda X'_i} \right] \leq \exp \left[ -\lambda t + \frac{\lambda^2}{8} \sum_{i=1}^{n} (b_i - a_i)^2 \right] \)
- 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 \( \lambda \)!
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. $X = X_1 + \ldots + X_n$ (our setting earlier)
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. In a randomly generated graph, $X_i$ indicates if the $i$-th edge is present 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.
A function $f$ is called **Lipschitz with parameters** $c = (c_1, \ldots, c_n)$ if for all $i = 1, 2, \ldots, n,$

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

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

---

**McDiarmid’s inequality**

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

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

and

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

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

Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

Appendix: More on Moment Generating Functions (non-examinable)
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 $c = (1, \ldots, 1)$ (If we move one ball to another bin, number of empty bins changes by $\leq 1$.)

By McDiarmid’s inequality, for any $t \geq 0$,

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

This is a decent bound, but for some values of $m$ it is far from tight and stronger bounds are possible through a refined analysis.
Application 4: Bin Packing

- We are given \( n \) items of sizes in the unit interval \([0, 1]\)
- 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 \( c = (1, \ldots, 1) \). Why?
- Therefore

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

This is a typical example where proving concentration is much easier than calculating (or estimating) the expectation!
Application 2: Randomised QuickSort

Extensions of Chernoff Bounds

Applications of Method of Bounded Differences

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

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

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

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

**Lemma**

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

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

**Proof of 2:**

$$M_{X+Y}(t) = \mathbb{E}[e^{t(X+Y)}] = \mathbb{E}[e^{tX} \cdot e^{tY}] \overset{(!)}{=} \mathbb{E}[e^{tX}] \cdot \mathbb{E}[e^{tY}] = M_X(t)M_Y(t) \quad \square$$
Randomised Algorithms
Lecture 4: Markov Chains and Mixing Times

Thomas Sauerwald (tms41@cam.ac.uk)
Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

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

Appendix: Remarks on Mixing Time (non-examin.)
Applications of Markov Chains in Computer Science

Broadcasting

Ranking Websites

Load Balancing

Clustering

Sampling and Optimisation

Particle Processes

A =

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
We say that \((X_t)_{t=0}^{\infty}\) is a Markov Chain on State Space \(\Omega\) with Initial Distribution \(\mu\) and Transition Matrix \(P\) if:

1. For any \(x \in \Omega\), \(P[ X_0 = x ] = \mu(x)\).
2. The Markov Property holds: for all \(t \geq 0\) and any \(x_0, \ldots, x_{t+1} \in \Omega\),

\[
P \left[ \frac{X_{t+1} = x_{t+1}}{X_t = x_t, \ldots, X_0 = x_0} \right] = P \left[ \frac{X_{t+1} = x_{t+1}}{X_t = x_t} \right] := P(x_t, x_{t+1}).
\]

From the definition one can deduce that (check!)

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

\[
P \left[ X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0 \right] = \mu(x_0) \cdot P(x_0, x_1) \cdot \ldots \cdot P(x_{t-2}, x_{t-1}) \cdot P(x_{t-1}, x_t).
\]

- For all \(0 \leq t_1 < t_2, x \in \Omega\),

\[
P \left[ X_{t_2} = x \right] = \sum_{y \in \Omega} P \left[ X_{t_2} = x \mid X_{t_1} = y \right] \cdot P \left[ X_{t_1} = y \right].
\]
What does a Markov Chain Look Like?

Example: the carbohydrate served with lunch in the college cafeteria.

This has transition matrix:

\[
P = \begin{bmatrix}
0 & 1/2 & 1/2 \\
1/4 & 0 & 3/4 \\
3/5 & 2/5 & 0
\end{bmatrix}
\]
The Transition Matrix $P$ of a Markov chain $(\mu, P)$ 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 $\rho^t$ by $P$ corresponds to advancing the chain one step:
  $$\rho^t(y) = \sum_{x \in \Omega} \rho^{t-1}(x) \cdot P(x, y) \quad \text{and thus} \quad \rho^t = \rho^{t-1} \cdot P.$$
- The Markov Property and line above imply that for any $t \geq 0$
  $$\rho^t = \rho \cdot P^{t-1} \quad \text{and thus} \quad P^t(x, y) = \mathbb{P} [ X_t = y \mid X_0 = x ].$$
  Thus $\rho^t(x) = (\mu P^t)(x)$ and so $\rho^t = \mu P^t = (\mu P^t(1), \mu P^t(2), \ldots, \mu P^t(n))$.

- Everything boils down to deterministic vector/matrix computations
  \[\Rightarrow\] can replace $\rho$ by any (load) vector and view $P$ as a balancing matrix!
Stopping and Hitting Times

A non-negative integer random variable \( \tau \) 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” \( \leadsto \tau := \min \{ t \geq 1 : X_{t-1} = “rice” \} \)

× “We are having pasta next Thursday”

For two states \( x, y \in \Omega \) we call \( h(x, y) \) the hitting time of \( y \) from \( x \):

\[
h(x, y) := \mathbb{E}_x[\tau_y] = \mathbb{E}[\tau_y \mid X_0 = x]
\]

where \( \tau_y = \min \{ t \geq 1 : X_t = y \} \).

Some distinguish between \( \tau_y^+ = \min \{ t \geq 1 : X_t = y \} \) and \( \tau_y = \min \{ t \geq 0 : X_t = y \} \)

A Useful Identity

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

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

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

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

Appendix: Remarks on Mixing Time (non-examin.)
Irreducible Markov Chains

A Markov Chain is **irreducible** if for every pair of states \( x, y \in \Omega \) there is an integer \( k \geq 0 \) such that \( P^k(x, y) > 0 \).

![Diagrams of irreducible and reducible Markov Chains]

**✓ irreducible**

**× not irreducible (thus reducible)**

---

**Finite Hitting Time Theorem**

For any states \( x \) and \( y \) of a **finite irreducible** Markov Chain \( h(x, y) < \infty \).
A probability distribution \( \pi = (\pi(1), \ldots, \pi(n)) \) is the **stationary distribution** of a Markov Chain if \( \pi P = \pi \) (\( \pi \) is a left eigenvector with eigenvalue 1).

**College carbs example:**

\[
\begin{pmatrix}
\frac{4}{13}, & \frac{4}{13}, & \frac{5}{13}
\end{pmatrix} \cdot \begin{pmatrix}
0 & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{4} & 0 & \frac{3}{4} \\
\frac{3}{5} & \frac{2}{5} & 0
\end{pmatrix} = \begin{pmatrix}
\frac{4}{13}, & \frac{4}{13}, & \frac{5}{13}
\end{pmatrix}
\]

- 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 \( \pi \) on \( \Omega \) such that \( \pi = \pi P \) and \( \pi(x) = 1/h(x, x) > 0, \forall x \in \Omega. \)
Periodicity

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

**Question:** Which of the two chains (if any) are aperiodic?
Convergence Theorem

Let $P$ be any finite, irreducible, aperiodic Markov Chain with stationary distribution $\pi$. 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 $\pi$ 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 $P^t(1, x)$. 

![Diagram of a Markov Chain with vertex values](chart.png)

Step: 0
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 $P^t(1, x)$.
- **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 $P^t(1, x)$.

Step: 25
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.)

Appendix: Remarks on Mixing Time (non-examin.)
How Similar are Two Probability Measures?

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

\[
\begin{array}{ccccccc}
 x & 1 & 2 & 3 & 4 & 5 & 6 \\
P[A = x] & 1/3 & 1/12 & 1/12 & 1/12 & 1/12 & 1/3 \\
P[B = x] & 1/4 & 1/8 & 1/8 & 1/8 & 1/8 & 1/4 \\
P[C = x] & 1/6 & 1/6 & 1/8 & 1/8 & 1/8 & 9/24 \\
\end{array}
\]

**Question 1:** Which dice is the least fair? Most choose \( A \). Why?

**Question 2:** Which dice is the most fair? Dice \( B \) and \( C \) seem “fairer” than \( A \) but which is fairest?

We need a formal “fairness measure” to compare probability distributions!
### Total Variation Distance

The **Total Variation Distance** between two probability distributions $\mu$ and $\eta$ on a countable state space $\Omega$ is given by

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

**Loaded Dice:** let $D = \text{Unif}\{1, 2, 3, 4, 5, 6\}$ be the law of a fair dice:

$$\|D - A\|_{tv} = \frac{1}{2} \left( 2 \left| \frac{1}{6} - \frac{1}{3} \right| + 4 \left| \frac{1}{6} - \frac{1}{12} \right| \right) = \frac{1}{3}$$

$$\|D - B\|_{tv} = \frac{1}{2} \left( 2 \left| \frac{1}{6} - \frac{1}{4} \right| + 4 \left| \frac{1}{6} - \frac{1}{8} \right| \right) = \frac{1}{6}$$

$$\|D - C\|_{tv} = \frac{1}{2} \left( 3 \left| \frac{1}{6} - \frac{1}{8} \right| + \left| \frac{1}{6} - \frac{9}{24} \right| \right) = \frac{1}{6}.$$

Thus

$$\|D - B\|_{tv} = \|D - C\|_{tv}$$

and

$$\|D - B\|_{tv}, \|D - C\|_{tv} < \|D - A\|_{tv}.$$

So $A$ is the least “fair”, however $B$ and $C$ are equally “fair” (in TV distance).
Let $P$ be a finite Markov Chain with stationary distribution $\pi$.

- Let $\mu$ be a prob. vector on $\Omega$ (might be just one vertex) and $t \geq 0$. Then

$$P^t_\mu := P \left[ X_t = \cdot \mid X_0 \sim \mu \right],$$

is a probability measure on $\Omega$.

- [Exercise 4/5.5] For any $\mu$,

$$\| P^t_\mu - \pi \|_{tv} \leq \max_{x \in \Omega} \| P^t_x - \pi \|_{tv}. $$

---

**Convergence Theorem (Implication for TV Distance)**

For any finite, irreducible, aperiodic Markov Chain

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

We will see a similar result later after introducing spectral techniques (Lecture 12)!
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 $\pi$ is defined as

$$
\tau_x(\epsilon) = \min \left\{ t \geq 0 : \left\| P_x^t - \pi \right\|_{t\text{v}} \leq \epsilon \right\},
$$

and,

$$
\tau(\epsilon) = \max_x \tau_x(\epsilon).
$$

- This is how long we need to wait until we are “$\epsilon$-close” to stationarity
- We often take $\epsilon = 1/4$, indeed let $t_{\text{mix}} := \tau(1/4)$

See final slides for some comments on why we choose $1/4$. 
Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

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

Appendix: Remarks on Mixing Time (non-examin.)
Experiment Gone Wrong...

Thanks to Krzysztof Onak (pointer) and Eric Price (graph)

Source: Slides by Ronitt Rubinfeld

4. Markov Chains and Mixing Times © T. Sauerwald

Application 1: Card Shuffling
What is Card Shuffling?

Here we will focus on one shuffling scheme which is easy to analyse.

How long does it take to shuffle a deck of 52 cards?

How quickly do we converge to the uniform distribution over all \( n! \) permutations?

One of the leading experts in the field who has related card shuffling to many other mathematical problems.

Persi Diaconis (Professor of Statistics and former Magician)

Source: www.soundcloud.com
The Card Shuffling Markov Chain

**TOPTORANDOMSHUFFLE** (Input: A pile of $n$ cards)

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

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

We will focus on this “small” set of cards ($n = 8$)
Even if we know which set of cards come after 8, every permutation is equally likely!

The 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 $\frac{2}{n}$.
- : 
- 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 $t_{mix} \leq n \log n$. 
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.

---

Figure: Total Variation Distance for \( t \) riffle shuffles of 52 cards.
Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

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

Appendix: Remarks on Mixing Time (non-examin.)
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!
INDEPENDENT SET SAMPLER

1: Let $X_0$ be an arbitrary independent set in $G$
2: For $t = 0, 1, 2, \ldots$
3: Pick a vertex $v \in V(G)$ uniformly at random
4: If $v \in X_t$ then $X_{t+1} \leftarrow X_t \setminus \{v\}$
5: elif $v \notin X_t$ and $X_t \cup \{v\}$ is an independent set then $X_{t+1} \leftarrow X_t \cup \{v\}$
6: else $X_{t+1} \leftarrow X_t$

$X_0 = \{1, 4\}$

$v = 1$

$v = 8$

$v = 6$

$X_1 = \{4\}$

$X_1 = \{1, 4, 8\}$

$X_1 = \{1, 4\}$
Markov Chain for Sampling Independent Sets (2/2) (non-examin.)

INDEPENDENT SET SAMPLER

1: Let $X_0$ be an arbitrary independent set in $G$
2: For $t = 0, 1, 2, \ldots$:
3: Pick a vertex $v \in V(G)$ uniformly at random
4: If $v \in X_t$ then $X_{t+1} \leftarrow X_t \setminus \{v\}$
5: elif $v \not\in X_t$ and $X_t \cup \{v\}$ is an independent set then $X_{t+1} \leftarrow X_t \cup \{v\}$
6: else $X_{t+1} \leftarrow X_t$

Remark

- This is a local definition (no explicit definition of $P$!)
- This chain is irreducible (every independent set is reachable)
- This chain is aperiodic (Check!)
- The stationary distribution is uniform, since $P_{u,v} = P_{v,u}$ (Check!)

Key Question: What is the mixing time of this Markov Chain?

not covered here, see the textbook by Mitzenmacher and Upfal
Outline

Recap of Markov Chain Basics

Irreducibility, Periodicity and Convergence

Total Variation Distance and Mixing Times

Application 1: Card Shuffling

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

Appendix: Remarks on Mixing Time (non-examin.)
Further Remarks on the Mixing Time (non-examin.)

- One can prove \( \max_x \| P_t^x - \pi \|_{tv} \) is non-increasing in \( t \) (this means if the chain is “\( \epsilon \)-mixed” at step \( t \), then this also holds in future steps) \([\text{Mitzenmacher, Upfal, 12.3}]\).

- We chose \( t_{mix} := \tau(1/4) \), but other choices of \( \epsilon \) are perfectly fine too (e.g., \( t_{mix} := \tau(1/e) \) is often used); in fact, any constant \( \epsilon \in (0, 1/2) \) is possible.

Remark: This freedom on how to pick \( \epsilon \) relies on the sub-multiplicative property of a (version) of the variation distance. First, let

\[
d(t) := \max_x \| P_t^x - \pi \|_{tv}
\]

be the variation distance after \( t \) steps when starting from the worst state. Further, define

\[
\overline{d}(t) := \max_{\mu, \nu} \| P_t^\mu - P_t^\nu \|_{tv}.
\]

These quantities are related by the following double inequality

\[
d(t) \leq \overline{d}(t) \leq 2d(t).
\]

Further, \( \overline{d}(t) \) is sub-multiplicative, that is for any \( s, t \geq 1 \),

\[
\overline{d}(s + t) \leq \overline{d}(s) \cdot \overline{d}(t).
\]

Hence for any fixed \( 0 < \epsilon < \delta < 1/2 \) it follows from the above that

\[
\tau(\epsilon) \leq \left\lfloor \frac{\ln \epsilon}{\ln(2\delta)} \right\rfloor \tau(\delta).
\]

In particular, for any \( \epsilon < 1/4 \)

\[
\tau(\epsilon) \leq \left\lfloor \log_2 \epsilon^{-1} \right\rfloor \tau(1/4).
\]

Hence smaller constants \( \epsilon < 1/4 \) only increase the mixing time by some constant factor.
Outline

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT
The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- 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:

\[
P_{x, x-1} = \frac{x}{d} \quad \text{and} \quad P_{x, x+1} = \frac{d - x}{d}.
\]

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

Random Walk on the Hypercube

- For each particle an indicator variable \(\Rightarrow \Omega = \{0, 1\}^d\)
- At each step: pick a random coordinate in \([d]\) and flip it
Analysis of the Mixing Time

For each particle an indicator variable $\Omega = \{0, 1\}^d$

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

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

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

We won’t formalise this argument here (see [Ex. 4/5.11])
Total Variation Distance of Random Walk on Hypercube \((d = 22)\)

\[
\|P_t x - \pi\|_TV \\
\]

\[
d \log d \approx 68.00
\]

\(t\)

\(\|P_t x - \pi\|_TV\)
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 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT
A Simple Random Walk (SRW) on a graph $G$ is a Markov chain on $V(G)$ with

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

and

$$\pi(u) = \frac{\deg(u)}{2|E|}.$$

Recall: $h(u, v) = E_u[\min\{t \geq 1 : X_t = v\}]$ is the hitting time of $v$ from $u$. 
Lazy Random Walks and Periodicity

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

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

Fact: For any graph $G$ the LRW on $G$ is aperiodic.

SRW on $C_4$, Periodic

LRW on $C_4$, Aperiodic
Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT
Will a random walk always return to the origin?

“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.
The $n$-path $P_n$ is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.

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

**Exercise:** [Exercise 4/5.15] What happens for the LRW on $P_n$?
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:

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

Proof: Let $f(k) = h(k, n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k - 1)}{2} + \frac{f(k + 1)}{2} \quad \text{for } 1 \leq k \leq n - 1.$$  

System of $n$ independent equations in $n$ unknowns, so has a unique solution. Thus it suffices to check that $f(k) = n^2 - k^2$ satisfies the above. Indeed

$$f(0) = 1 + f(1) = 1 + n^2 - 1^2 = n^2,$$

and for any $1 \leq k \leq n - 1$ we have,

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

$\square$
Outline

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT
A Satisfiability (SAT) formula is a logical expression that’s the conjunction (AND) of a set of Clauses, where a clause is the disjunction (OR) of Literals.

A Solution to a SAT formula is an assignment of the variables to the values True and False so that all the clauses are satisfied.

Example:

\[
\text{SAT: } (x_1 \lor \overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_3}) \land (x_1 \lor x_2 \lor x_4) \land (x_4 \lor \overline{x_3}) \land (\overline{x_4} \lor x_1)
\]

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

- If each clause has \(k\) literals we call the problem \(k\)-SAT.
- In general, determining if a SAT formula has a solution is NP-hard
- In practice solvers are fast and used to great effect
- A huge amount of problems can be posed as a SAT:
  - Model checking and hardware/software verification
  - Design of experiments
  - Classical planning
  - …
**2-SAT**

**RANDOMISED-2-SAT** (Input: a 2-SAT-Formula)

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 $\alpha$ be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

**Example 1 : Solution Found**

\[(x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_3) \land (x_1 \lor x_2) \land (x_4 \lor \overline{x}_3) \land (x_4 \lor \overline{x}_1)\]

<table>
<thead>
<tr>
<th>t</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

\[\alpha = (T, T, F, T).\]
2-SAT

**RANDOMISED-2-SAT** (Input: a 2-SAT-Formula)

1: Start with an arbitrary truth assignment
2: **Repeat up to** $2n^2$ **times**
3: Pick an arbitrary unsatisfied clause
4: Choose a random literal and switch its value
5: If formula is satisfied **then return** "Satisfiable"
6: **return** "Unsatisfiable"

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

**Example 2:** (Another) Solution Found

$$(x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_3) \land (x_1 \lor x_2) \land (x_4 \lor x_3) \land (x_4 \lor \overline{x_1})$$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

$\alpha = (T, F, F, T)$. 

\[\begin{array}{cccc}
0 & 1 & 2 & 3 & 4
\end{array}\]
If the formula is satisfiable, then the expected number of steps before \textsc{Randomised-2-SAT} outputs a valid solution is at most $n^2$.

**Proof:** Fix any solution $\alpha$, then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

(i) $\Pr[ X_{i+1} = 1 \mid X_i = 0 ] = 1$

(ii) $\Pr[ X_{i+1} = k + 1 \mid X_i = k ] \geq 1/2$

(iii) $\Pr[ X_{i+1} = k - 1 \mid X_i = k ] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus solution found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process $X_i$ is complicated to describe in full; however by (i) – (iii) we can bound it by $Y_i$ (SRW on the $n$-path from 0). This gives (see also \cite{Ex 4/5.16})

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

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

Proposition

Provided a solution exists, \textsc{Randomised-2-SAT} will return a valid solution in $O(n^2)$ steps with probability at least $1/2$. 

5. Hitting Times © T. Sauerwald

SAT and a Randomised Algorithm for 2-SAT

20
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

\[
P \left[ \text{t runs all fail} \right] \leq (1 - p)^t \\ \leq e^{-pt} \\ \leq n^{-C},
\]

thus the probability one of the runs succeeds is at least \( 1 - n^{-C} \). \( \square \)

Randomised-2-SAT

There is a \( O(n^2 \log n) \)-step algorithm for 2-SAT which succeeds w.h.p.
Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
- linear programming is a powerful tool in optimisation
- inspired more sophisticated techniques such as quadratic optimisation, convex optimisation, integer programming and semi-definite programming
- we will later use the connection between linear and integer programming to tackle several problems (Vertex-Cover, Set-Cover, TSP, satisfiability)
Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
What are Linear Programs?

- 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

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]

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
  \[
  f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n.
  \]
- Linear Equality: \( f(x_1, x_2, \ldots, x_n) = b \)
- Linear Inequality: \( f(x_1, x_2, \ldots, x_n) \geq b \)
- Linear Programming Problem: either minimise or maximise a linear function subject to a set of linear constraints
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)
subject to

\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]

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

maximise $x_1 + x_2$
subject to
$4x_1 + x_2 \leq 20$
$2x_1 + x_2 \leq 10$
$x_1 + 2x_2 \leq 14$
$x_1, x_2 \geq 0$

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

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

- **Exercise**: Translate the SPSP problem into a linear program!
Shortest Paths

Given: directed graph \( G = (V, E) \) with edge weights \( w : E \rightarrow \mathbb{R} \), pair of vertices \( s, t \in V \)

Goal: Find a path of minimum weight from \( s \) to \( t \) in \( G \)

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

Shortest Paths as LP

maximise

\[ d_t \]

subject to

\[ d_v \leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \]

\[ d_s = 0. \]

this is a maximisation problem!

Recall: When BELLMAN-FORD terminates, all these inequalities are satisfied.

Solution \( \bar{d} \) satisfies

\[ \overline{d}_v = \min_{u : (u, v) \in E} \{ \overline{d}_u + w(u, v) \} \]
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

maximise \[ \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} \]

subject to

- \[ f_{uv} \leq c(u, v) \] for each \( u, v \in V \),
- \[ \sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv} \] for each \( u \in V \setminus \{s, t\} \),
- \[ f_{uv} \geq 0 \] for each \( u, v \in V \).
Minimum-Cost Flow

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 \( \sum_{(u,v) \in E} a(u, v)f_{uv} \) incurred by the flow.

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

Figure 29.3  (a) An example of a minimum-cost-flow problem. We denote the capacities by \( c \) and the costs by \( a \). Vertex \( s \) is the source and vertex \( t \) is the sink, and we wish to send 4 units of flow from \( s \) to \( t \). (b) A solution to the minimum-cost flow problem in which 4 units of flow are sent from \( s \) to \( t \). For each edge, the flow and capacity are written as flow/capacity.
Minimum Cost Flow as a LP

minimise \[ \sum_{(u,v) \in E} a(u, v)f_{uv} \]
subject to
- \[ f_{uv} \leq c(u, v) \] for \( u, v \in V \),
- \[ \sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} = 0 \] for \( u \in V \setminus \{s, t\} \),
- \[ \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} = d \),
- \[ f_{uv} \geq 0 \] for \( u, v \in V \).

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

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
Standard and Slack Forms

**Standard Form**

\[
\begin{align*}
\text{maximise} & \quad \sum_{j=1}^{n} c_j x_j & \text{Objective Function} \\
\text{subject to} & \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \text{for } i = 1, 2, \ldots, m \\
& \quad x_j \geq 0 \quad \text{for } j = 1, 2, \ldots, n
\end{align*}
\]

**n + m constraints**

**Standard Form (Matrix-Vector-Notation)**

\[
\begin{align*}
\text{maximise} & \quad c^T x & \text{Inner product of two vectors} \\
\text{subject to} & \quad Ax \leq b & \text{Matrix-vector product} \\
& \quad x \geq 0
\end{align*}
\]
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.
Reasons for a LP not being in standard form:
1. The objective might be a minimisation rather than maximisation.

\[
\begin{align*}
\text{minimise} & \quad -2x_1 + 3x_2 \\
\text{subject to} & \\
& x_1 + x_2 = 7 \\
& x_1 - 2x_2 \leq 4 \\
& x_1 \geq 0 \\
\end{align*}
\]

Negate objective function

\[
\begin{align*}
\text{maximise} & \quad 2x_1 - 3x_2 \\
\text{subject to} & \\
& x_1 + x_2 = 7 \\
& x_1 - 2x_2 \leq 4 \\
& x_1 \geq 0 \\
\end{align*}
\]
Reasons for a LP not being in standard form:

2. There might be variables without nonnegativity constraints.

maximise \( 2x_1 - 3x_2 \)

subject to

\[
\begin{align*}
    x_1 + x_2 &= 7 \\
    x_1 - 2x_2 &\leq 4 \\
    x_1 &\geq 0
\end{align*}
\]

Replace \( x_2 \) by two non-negative variables \( x'_2 \) and \( x''_2 \):

maximise \( 2x_1 - 3x'_2 + 3x''_2 \)

subject to

\[
\begin{align*}
    x_1 + x'_2 - x''_2 &= 7 \\
    x_1 - 2x'_2 + 2x''_2 &\leq 4 \\
    x_1, x'_2, x''_2 &\geq 0
\end{align*}
\]
Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

maximise $2x_1 - 3x_2 + 3x_2''$

subject to

\[
\begin{align*}
x_1 + x_2' - x_2'' &= 7 \\
x_1 - 2x_2' + 2x_2'' &\leq 4 \\
x_1, x_2', x_2'' &\geq 0
\end{align*}
\]

Replace each equality by two inequalities.

maximise $2x_1 - 3x_2' + 3x_2''$

subject to

\[
\begin{align*}
x_1 + x_2' - x_2'' &\leq 7 \\
x_1 + x_2' - x_2'' &\geq 7 \\
x_1 - 2x_2' + 2x_2'' &\leq 4 \\
x_1, x_2', x_2'' &\geq 0
\end{align*}
\]
Reasons for a LP not being in standard form:

4. There might be inequality constraints (with $\geq$ instead of $\leq$).

\[
\begin{align*}
\text{maximise} & \quad 2x_1 - 3x'_2 + 3x''_2 \\
\text{subject to} & \\
& x_1 + x'_2 - x''_2 \leq 7 \\
& x_1 + x'_2 - x''_2 \geq 7 \\
& x_1 - 2x'_2 + 2x''_2 \leq 4 \\
& x_1, x'_2, x''_2 \geq 0
\end{align*}
\]

Negate respective inequalities.

\[
\begin{align*}
\text{maximise} & \quad 2x_1 - 3x'_2 + 3x''_2 \\
\text{subject to} & \\
& -x_1 - x'_2 + x''_2 \leq -7 \\
& x_1 - 2x'_2 + 2x''_2 \leq 4 \\
& x_1, x'_2, x''_2 \geq 0
\end{align*}
\]
Converting into Standard Form (5/5)

It is always possible to convert a linear program into standard form.
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

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

$s \geq 0$.

- Denote slack variable of the $i$-th inequality by $x_{n+i}$
Converting Standard Form into Slack Form (2/3)

maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_1 + x_2 - x_3 \leq 7 \]
\[ -x_1 - x_2 + x_3 \leq -7 \]
\[ x_1 - 2x_2 + 2x_3 \leq 4 \]
\[ x_1, x_2, x_3 \geq 0 \]

Introduce slack variables

maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_4 = 7 - x_1 - x_2 + x_3 \]
\[ x_5 = -7 + x_1 + x_2 - x_3 \]
\[ x_6 = 4 - x_1 + 2x_2 - 2x_3 \]
\[ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \]
maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_4 = 7 - x_1 - x_2 + x_3 \]
\[ x_5 = -7 + x_1 + x_2 - x_3 \]
\[ x_6 = 4 - x_1 + 2x_2 - 2x_3 \]
\[ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \]
Use variable \( z \) to denote objective function and omit the nonnegativity constraints.

\( z = 2x_1 - 3x_2 + 3x_3 \)
\( x_4 = 7 - x_1 - x_2 + x_3 \)
\( x_5 = -7 + x_1 + x_2 - x_3 \)
\( x_6 = 4 - x_1 + 2x_2 - 2x_3 \)

This is called \textit{slack form}. 
Basic and Non-Basic Variables

\[
\begin{align*}
  z &= 2x_1 - 3x_2 + 3x_3 \\
  x_4 &= 7 - x_1 - x_2 + x_3 \\
  x_5 &= -7 + x_1 + x_2 - x_3 \\
  x_6 &= 4 - x_1 + 2x_2 - 2x_3
\end{align*}
\]

Basic Variables: \( B = \{4, 5, 6\} \)
Non-Basic Variables: \( N = \{1, 2, 3\} \)

Slack Form (Formal Definition)

Slack form is given by a tuple \((N, B, A, b, c, v)\) so that

\[
\begin{align*}
  z &= v + \sum_{j \in N} c_j x_j \\
  x_i &= b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B,
\end{align*}
\]

and all variables are non-negative.

Variables/Coefficients on the right hand side are indexed by \(B\) and \(N\).
Slack Form (Example)

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

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)
Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)
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

maximise $3x_1 + x_2 + 2x_3$

subject to

$x_1 + x_2 + 3x_3 \leq 30$
$2x_1 + 2x_2 + 5x_3 \leq 24$
$4x_1 + x_2 + 2x_3 \leq 36$

$x_1, x_2, x_3 \geq 0$

Conversion into slack form

$z = 3x_1 + x_2 + 2x_3$
$x_4 = 30 - x_1 - x_2 - 3x_3$
$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$
$x_6 = 36 - 4x_1 - x_2 - 2x_3$
Extended Example: Iteration 1

\[
\begin{align*}
    z &= 3x_1 + x_2 + 2x_3 \\
    x_4 &= 30 - x_1 - x_2 - 3x_3 \\
    x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
    x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{align*}
\]

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

This basic solution is **feasible**

Objective value is 0.

Increasing the value of \(x_1\) would increase the objective value.

Increasing the value of \(x_3\) would increase the objective value.

Increasing the value of \(x_2\) would increase the objective value.

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

The third constraint is the tightest and limits how much we can increase \(x_1\).

The third constraint is the tightest and limits how much we can increase \(x_3\).

The second constraint is the tightest and limits how much we can increase \(x_2\).

Switch roles of \(x_1\) and \(x_6\):

Solving for \(x_1\) yields:
\[
x_1 = 9 - x_2 - x_3 - 4x_6
\]

Substitute this into \(x_1\) in the other three equations

Switch roles of \(x_3\) and \(x_5\):

Solving for \(x_3\) yields:
\[
x_3 = 3 - 2x_2 - 8x_5 + 2x_6
\]

Substitute this into \(x_3\) in the other three equations

Switch roles of \(x_2\) and \(x_3\):

Solving for \(x_2\) yields:
\[
x_2 = 4 - 2x_3 - 3x_5 + 2x_6 - x_3 - 3x_6
\]

Substitute this into \(x_2\) in the other three equations

7. Linear Programming © T. Sauerwald

Simplex Algorithm by Example
Extended Example: Iteration 1

Increasing the value of $x_1$ would increase the objective value.

\[
    z = 3x_1 + x_2 + 2x_3
\]

\[
    x_4 = 30 - x_1 - x_2 - 3x_3
\]

\[
    x_5 = 24 - 2x_1 - 2x_2 - 5x_3
\]

\[
    x_6 = 36 - 4x_1 - x_2 - 2x_3
\]

The third constraint is the tightest and limits how much we can increase $x_1$.

**Switch roles of $x_1$ and $x_6$:**

- Solving for $x_1$ yields:

  \[
  x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}.
  \]

- Substitute this into $x_1$ in the other three equations.
Extended Example: Iteration 2

Increasing the value of \( x_3 \) would increase the objective value.

\[
\begin{align*}
z &= 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \\
x_1 &= 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \\
x_4 &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \\
x_5 &= 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2}
\end{align*}
\]

Basic solution: \((x_1, x_2, \ldots, x_6) = (9, 0, 0, 21, 6, 0)\) with objective value 27
Extended Example: Iteration 2

\[ z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \]

\[ x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \]

\[ x_4 = 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \]

\[ x_5 = 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2} \]

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.

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

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

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

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 \textbf{optimal}!

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

Basic solution: \((x_1, x_2, \ldots, x_6) = (8, 4, 0, 18, 0, 0)\) with objective value 28
Exercise: [Ex. 6/7.6] How many basic solutions (including non-feasible ones) are there?
Extended Example: Alternative Runs (1/2)

\[
\begin{align*}
z & = 3x_1 + x_2 + 2x_3 \\
x_4 & = 30 - x_1 - x_2 - 3x_3 \\
x_5 & = 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 & = 36 - 4x_1 - x_2 - 2x_3 \\
\end{align*}
\]

Switch roles of \(x_2\) and \(x_5\)

\[
\begin{align*}
z & = 12 + 2x_1 - \frac{x_3}{2} - \frac{x_5}{2} \\
x_2 & = 12 - x_1 - \frac{5x_3}{2} - \frac{x_5}{2} \\
x_4 & = 18 - x_2 - \frac{x_3}{2} + \frac{x_5}{2} \\
x_6 & = 24 - 3x_1 + \frac{x_3}{2} + \frac{x_5}{2} \\
\end{align*}
\]

Switch roles of \(x_1\) and \(x_6\)

\[
\begin{align*}
z & = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \\
x_1 & = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \\
x_2 & = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \\
x_4 & = 18 - \frac{x_3}{2} + \frac{x_5}{2} \\
\end{align*}
\]
Extended Example: Alternative Runs (2/2)

\[ z = 3x_1 + x_2 + 2x_3 \]
\[ x_4 = 30 - x_1 - x_2 - 3x_3 \]
\[ x_5 = 24 - 2x_1 - 2x_2 - 5x_3 \]
\[ x_6 = 36 - 4x_1 - x_2 - 2x_3 \]

Switch roles of \( x_3 \) and \( x_5 \)

\[ z = \frac{48}{5} + \frac{11x_1}{5} + \frac{x_2}{5} - \frac{2x_5}{5} \]
\[ x_4 = \frac{78}{5} + \frac{x_1}{5} + \frac{x_2}{5} + \frac{3x_5}{5} \]
\[ x_3 = \frac{24}{5} - \frac{2x_1}{5} - \frac{2x_2}{5} - \frac{x_5}{5} \]
\[ x_6 = \frac{132}{5} - \frac{16x_1}{5} - \frac{x_2}{5} + \frac{2x_3}{5} \]

Switch roles of \( x_1 \) and \( x_6 \)

\[ z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \]
\[ x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \]
\[ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \]
\[ x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_6}{8} - \frac{x_6}{16} \]

Switch roles of \( x_2 \) and \( x_3 \)

\[ z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \]
\[ x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \]
\[ x_2 = 4 - \frac{8x_3}{3} - \frac{2x_6}{3} + \frac{x_6}{3} \]
\[ x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} \]
Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)
The Pivot Step Formally

\( \text{Pivot}(N, B, A, b, c, v, l, e) \)

1. // Compute the coefficients of the equation for new basic variable \( x_e \).
2. let \( \hat{A} \) be a new \( m \times n \) matrix
3. \( \hat{b}_e = b_l/a_{le} \)
4. for each \( j \in N - \{e\} \)
   5. \( \hat{a}_{ej} = a_{lj}/a_{le} \)
   6. \( \hat{a}_{el} = 1/a_{le} \)
7. // Compute the coefficients of the remaining constraints.
8. for each \( i \in B - \{l\} \)
   9. \( \hat{b}_i = b_i - a_{ie}\hat{b}_e \)
10. for each \( j \in N - \{e\} \)
    11. \( \hat{a}_{ij} = a_{ij} - a_{ie}\hat{a}_{ej} \)
    12. \( \hat{a}_{il} = -a_{ie}\hat{a}_{el} \)
13. // Compute the objective function.
14. \( \hat{\nu} = v + c_e\hat{b}_e \)
15. for each \( j \in N - \{e\} \)
    16. \( \hat{c}_j = c_j - c_e\hat{a}_{ej} \)
    17. \( \hat{c}_l = -c_e\hat{a}_{el} \)
18. // Compute new sets of basic and nonbasic variables.
19. \( \hat{N} = N - \{e\} \cup \{l\} \)
20. \( \hat{B} = B - \{l\} \cup \{e\} \)
21. return (\( \hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{\nu} \))

Rewrite “tight” equation for entering variable \( x_e \).

Substituting \( x_e \) into other equations.

Substituting \( x_e \) into objective function.

Update non-basic and basic variables.
Consider a call to PIVOT($N, B, A, b, c, v, l, e$) in which $a_{le} \neq 0$. Let the values returned from the call be ($\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v}$), and let $\bar{x}$ denote the basic solution after the call. Then

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

Proof:

1. holds since the basic solution always sets all non-basic variables to zero.
2. When we set each non-basic variable to 0 in a constraint

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

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

$$\bar{x}_i = \hat{b}_i = b_i - a_{ie}\hat{b}_e.$$
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. \((N, B, A, b, c, v) = \text{INITIALIZE-SIMPLEX}(A, b, c)\)
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 \((N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, e)\)
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 \((\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)\)

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

Main Loop:
- terminates if all coefficients in objective function are non-positive
- Line 4 picks entering variable \(x_e\) with positive coefficient
- Lines 6 – 9 pick the tightest constraint, associated with \(x_l\)
- Line 11 returns “unbounded” if there are no constraints
- Line 12 calls PIVOT, switching roles of \(x_l\) and \(x_e\)

Return corresponding solution.
The formal procedure SIMPLEX

SIMPLEX\((A, b, c)\)
1. \((N, B, A, b, c, v) = \text{INITIALIZE-SIMPLEX}(A, b, c)\)
2. let \(\Delta\) be a new vector of length \(m\)
3. \(\text{while some index } j \in N \text{ has } c_j > 0\)
4. choose an index \(e \in N\) for which \(c_e > 0\)
5. \(\text{for each index } i \in B\)
6. \(\text{if } a_{ie} > 0\)
7. \(\Delta_i = b_i / a_{ie}\)
8. \(\text{else } \Delta_i = \infty\)
9. choose an index \(l \in B\) that minimizes \(\Delta_i\)
10. \(\text{if } \Delta_l = \infty\)
11. \(\text{return } \text{“unbounded”}\)

Proof is based on the following three-part loop invariant:

1. the slack form is always equivalent to the one returned by \text{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 \text{INITIALIZE-SIMPLEX} in line 1 returns a slack form for which the basic solution is feasible. Then if \text{SIMPLEX} returns a solution, it is a feasible solution. If \text{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)
maximise \( 2x_1 - x_2 \)
subject to
\[
\begin{align*}
2x_1 - x_2 & \leq 2 \\
x_1 - 5x_2 & \leq -4 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Conversion into slack form

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

maximise \( 2x_1 - x_2 \)

subject to

\[
\begin{align*}
2x_1 - x_2 &\leq 2 \\
x_1 - 5x_2 &\leq -4 \\
x_1, x_2 &\geq 0
\end{align*}
\]

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

Let $L_{aux}$ be the auxiliary LP of a linear program $L$ in standard form. Then $L$ is feasible if and only if the optimal objective value of $L_{aux}$ is 0.

Proof. Exercise!
Let us illustrate the role of $x_0$ as “distance from feasibility”. We’ll also see that increasing $x_0$ enlarges the feasible region.
maximise \(-x_0\)
subject to
\[\begin{align*}
2x_1 - x_2 - x_0 & \leq 2 \\
x_1 - 5x_2 - x_0 & \leq -4 \\
x_0, x_1, x_2 & \geq 0
\end{align*}\]

For the animation see the full slides.
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 $x_0 > 0$!
maximise $-x_0$
subject to

$$
\begin{align*}
2x_1 - x_2 - x_0 & \leq -2 \\
-x_1 + 5x_2 - x_0 & \leq 4 \\
x_0, x_1, x_2 & \geq 0
\end{align*}
$$

For the animation see the full slides.
INITIALIZE-SIMPLEX ($A, b, c$)

1. Let $k$ be the index of the minimum $b_i$
2. If $b_k \geq 0$  // is the initial basic solution feasible?
3. Return ($\{1, 2, \ldots, n\}, \{n + 1, n + 2, \ldots, n + m\}, A, b, c, 0$)
4. Form $L_{aux}$ by adding $-x_0$ to the left-hand side of each constraint
   and setting the objective function to $-x_0$.
5. Let $(N, B, A, b, c, v)$ be the resulting slack form for $L_{aux}$
6. $l = n + k$
7. // $L_{aux}$ has $n + 1$ nonbasic variables and $m$ basic variables.
8. $(N, B, A, b, c, v) = \text{Pivot}(N, B, A, b, c, v, l, 0)$
9. // The basic solution is now feasible for $L_{aux}$.
10. Iterate the while loop of lines 3–12 of SIMPLEX until an optimal solution
    to $L_{aux}$ is found
11. If the optimal solution to $L_{aux}$ sets $\bar{x}_0$ to 0
12.   If $\bar{x}_0$ is basic
13.      Perform one (degenerate) pivot to make it nonbasic
14.      From the final slack form of $L_{aux}$, remove $x_0$ from the constraints and
        restore the original objective function of $L$, but replace each basic
        variable in this objective function by the right-hand side of its
        associated constraint
15.    Return the modified final slack form
16. Else return “infeasible”

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

$\ell$ will be the leaving variable so that $x_\ell$ has the most negative value.

Pivot step with $x_\ell$ leaving and $x_0$ entering.

This pivot step does not change the value of any variable.
Example of INITIALIZE-SIMPLEX (1/3)

maximise \(2x_1 - x_2\)
subject to
\[
\begin{align*}
2x_1 & - x_2 \leq 2 \\
x_1 & - 5x_2 \leq -4 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Formulating the auxiliary linear program

maximise \(-x_0\)
subject to
\[
\begin{align*}
2x_1 & - x_2 - x_0 \leq 2 \\
x_1 & - 5x_2 - x_0 \leq -4 \\
x_1, x_2, x_0 & \geq 0
\end{align*}
\]

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

Converting into slack form
\[
\begin{align*}
z & = 2 - 2x_1 + x_2 + x_0 \\
x_3 & = 2 - 2x_1 + x_2 + x_0 \\
x_4 & = -4 - x_1 + 5x_2 + x_0
\end{align*}
\]
Example of INITIALIZE-SIMPLEX (2/3)

\[
\begin{align*}
 z &= -x_0 \\
 x_3 &= 2 - 2x_1 + x_2 + x_0 \\
 x_4 &= -4 - x_1 + 5x_2 + x_0 \\
\end{align*}
\]

Pivot with \( x_0 \) entering and \( x_4 \) leaving

\[
\begin{align*}
 z &= -4 - x_1 + 5x_2 - x_4 \\
 x_0 &= 4 + x_1 - 5x_2 + x_4 \\
 x_3 &= 6 - x_1 - 4x_2 + x_4 \\
\end{align*}
\]

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

Pivot with \( x_2 \) entering and \( x_0 \) leaving

\[
\begin{align*}
 z &= \frac{4}{5} - x_0 \\
 x_2 &= \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\
 x_3 &= \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5} \\
\end{align*}
\]

Optimal solution has \( x_0 = 0 \), hence the initial problem was feasible!
Example of **INITIALIZE-SIMPLEX** (3/3)

\[
\begin{align*}
  z &= -x_0, \\
  x_2 &= \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5}, \\
  x_3 &= \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5}.
\end{align*}
\]

Set \( x_0 = 0 \) and express objective function by non-basic variables:

\[
\begin{align*}
  z &= -\frac{4}{5} + \frac{9x_1}{5} - \frac{x_4}{5}, \\
  x_2 &= \frac{4}{5} + \frac{x_1}{5} + \frac{x_4}{5}, \\
  x_3 &= \frac{14}{5} - \frac{9x_1}{5} + \frac{x_4}{5}.
\end{align*}
\]

Basic solution \((0, \frac{4}{5}, \frac{14}{5}, 0)\), which is feasible!

**Lemma 29.12**

If a linear program \( L \) has no feasible solution, then **INITIALIZE-SIMPLEX** returns “infeasible”. Otherwise, it returns a valid slack form for which the basic solution is feasible.
Fundamental Theorem of Linear Programming

Theorem 29.13 (Fundamental Theorem of Linear Programming)

For any linear program $L$, given in standard form, either:

1. $L$ is infeasible $\Rightarrow$ SIMPLEX returns “infeasible”.
2. $L$ is unbounded $\Rightarrow$ SIMPLEX returns “unbounded”.
3. $L$ has an optimal solution with a finite objective value $\Rightarrow$ SIMPLEX returns an optimal solution with a finite objective value.

Small Technicality: need to equip SIMPLEX with an “anti-cycling strategy” (see extra slides)

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
\[ \text{INITIALIZE-SIMPLEX terminates} \]

Feasible Basic Solution
\[ \text{INITIALIZE-SIMPLEX followed by SIMPLEX} \]

LP unbounded
\[ \text{SIMPLEX terminates} \]

LP bounded
\[ \text{SIMPLEX returns optimum} \]
Linear Programming

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

Simplex Algorithm

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

Research Problem: Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?

Polynomial-Time Algorithms

- Interior-Point Methods: traverses the interior of the feasible set of solutions (not just vertices!)
1.2 Famous Failures and the Need for Alternatives

For many problems a bit beyond the scope of an undergraduate course, the downside of worst-case analysis rears its ugly head. This section reviews four famous examples in which worst-case analysis gives misleading or useless advice about how to solve a problem. These examples motivate the alternatives to worst-case analysis that are surveyed in Section 1.4 and described in detail in later chapters of the book.

1.2.1 The Simplex Method for Linear Programming

Perhaps the most famous failure of worst-case analysis concerns linear programming, the problem of optimizing a linear function subject to linear constraints (Figure 1.1). Dantzig proposed in the 1940s an algorithm for solving linear programs called the simplex method. The simplex method solves linear programs using greedy local

Source: “Beyond the Worst-Case Analysis of Algorithms” by Tim Roughgarden, 2020
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.

\[
\begin{align*}
  z &= x_1 + x_2 + x_3 \\
  x_4 &= 8 - x_1 - x_2 \\
  x_5 &=
\end{align*}
\]

Pivot with \(x_1\) entering and \(x_4\) leaving

\[
\begin{align*}
  z &= 8 + x_3 - x_4 \\
  x_1 &= 8 - x_2 - x_4 \\
  x_5 &= x_2 - x_3
\end{align*}
\]

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

\[
\begin{align*}
  z &= 8 + x_2 - x_4 - x_5 \\
  x_1 &= 8 - x_2 - x_4 \\
  x_3 &= x_2 - x_5
\end{align*}
\]
Exercise: Execute one more step of the Simplex Algorithm on the tableau from the previous slide.
Termination and Running Time

**Cycling**: SIMPLEX may fail to terminate.

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

Anti-Cycling Strategies

- **Bland’s rule**: Choose entering variable with smallest index
- **Random rule**: Choose entering variable uniformly at random
- **Perturbation**: Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Replace each \( b_i \) by \( \hat{b}_i = b_i + \epsilon_i \), where \( \epsilon_i \gg \epsilon_{i+1} \) are all small.

**Lemma 29.7**

Assuming INITIALIZE-SIMPLEX returns a slack form for which the basic solution is feasible, SIMPLEX either reports that the program is unbounded or returns a feasible solution in at most \( \binom{n+m}{m} \) iterations.

Every set \( B \) of basic variables uniquely determines a slack form, and there are at most \( \binom{n+m}{m} \) unique slack forms.
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.

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

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
The traveling salesman problem recently achieved national prominence when a soap company used it as the basis of a promotional contest. Prizes up to $10,000 were offered. Thus Flood realized that the Nearest Neighbor method is not a good estimate of the TSP but it created a decent first solution. In 1962 a contest brought the TSP national recognition through a contest given by Proctor and Gamble. A flyer of the contest is pictured below.
532 cities (1987 [Padberg, Rinaldi])
13,509 cities (1999 [Applegate, Bixby, Chavatal, Cook])
SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM*

G. DANTZIG, R. FULKERSON, AND S. JOHNSON

The Rand Corporation, Santa Monica, California

(Received August 9, 1954)

It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington, D. C., has the shortest road distance.

THE TRAVELING-SALESMAN PROBLEM might be described as follows: Find the shortest route (tour) for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the original point of departure. More generally, given an \( n \) by \( n \) symmetric matrix \( D = (d_{ij}) \), where \( d_{ij} \) represents the 'distance' from \( I \) to \( J \), arrange the points in a cyclic order in such a way that the sum of the \( d_{ij} \) between consecutive points is minimal. Since there are only a finite number of possibilities (at most \( \frac{1}{2} (n-1)! \)) to consider, the problem is to devise a method of picking out the optimal arrangement which is reasonably efficient for fairly large values of \( n \). Although algorithms have been devised for problems of similar nature, e.g., the optimal assignment problem,\(^{3,7,8}\) little is known about the traveling-salesman problem. We do not claim that this note alters the situation very much; what we shall do is outline a way of approaching the problem that sometimes, at least, enables one to find an optimal path and prove it so. In particular, it will be shown that a certain arrangement of 49 cities, one in each of the 48 states and Washington, D. C., is best, the \( d_{ij} \) used representing road distances as taken from an atlas.
The 42 (49) Cities

1. Manchester, N. H.  
4. Cleveland, Ohio  
7. Indianapolis, Ind.  
8. Chicago, Ill.  
9. Milwaukee, Wis.  
10. Minneapolis, Minn.  
11. Pierre, S. D.  
12. Bismarck, N. D.  
13. Helena, Mont.  
15. Portland, Ore.  
16. Boise, Idaho  
17. Salt Lake City, Utah  
18. Carson City, Nev.  
19. Los Angeles, Calif.  
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.  
40. Washington, D. C.  
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

\[ \frac{1}{2} (42 - 1)! \]

Result

\[ 16,726,263,306,581,903,554,085,031,026,720,375,832,576,000,000,000 \]

Scientific notation

\[ 1.6726263306581903554085031026720375832576 \times 10^{69} \]

Number name

16 quadricillion ...

Number length

50 decimal digits

Alternative representations

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

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

\[ \frac{1}{2} (42 - 1)! = \frac{(1)_{41}}{2} \]
Dantzig, Fulkerson and Johnson found an optimal tour through 42 cities.

http://www.math.uwaterloo.ca/tsp/history/img/dantzig_big.html
Hence this is an instance of the Metric TSP, but not Euclidean TSP.

| Road Distances |

| TABLE I |

| ROAD DISTANCES BETWEEN CITIES IN ADJUSTED UNITS |

| The figures in the table are mileages between the two specified numbered cities, less 11, divided by 17, and rounded to the nearest integer. | 8. Solving TSP via Linear Programming © T. Sauerwald | Examples of TSP Instances | 12 |
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)

\[
\text{minimize} \quad \sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i, j) x(i, j)
\]

subject to

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

\[
0 \leq x(i, j) \leq 1 \quad \text{for each } 1 \leq j < i \leq 42
\]

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.
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

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 \(S = \{11, 12, \ldots, 23\}\) 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
   \[ x_{18,15} = 0 \]
   \[ x_{17,13} = 0 \]

8: LP solution 698
   \[ x_{18,15} = 1 \]
   \[ x_{17,13} = 0 \]

9: Valid tour 699

10: LP solution 700
   \[ x_{18,15} = 1 \]
   \[ x_{17,13} = 1 \]

11: Valid tour 701

Cut branch, since LP solution worse than current best possible tour.
What about choosing a different branching variable?
Solving Progress (Alternative Branch 1)

1: LP solution 641
   \[ \text{Eliminate Subtour } 1, 2, 41, 42 \]

2: LP solution 676
   \[ \text{Eliminate Subtour } 3 - 9 \]

3: LP solution 681
   \[ \text{Eliminate Subtour } 24, 25, 26, 27 \]

4: LP solution 682.5
   \[ \text{Eliminate Cut } 13 - 17 \]

5: LP solution 686
   \[ \text{Eliminate Subtour } 10, 11, 12 \]

6: LP solution 686
   \[ \text{Eliminate Subtour } 13 - 23 \]

7: LP solution 688
   \[ \text{Eliminate Subtour } 11 - 23 \]

8: LP solution 697
   \[ x_{15,18} = 1 \]
   \[ x_{15,18} = 0 \]

9: ???

10: ???

8. Solving TSP via Linear Programming © T. Sauerwald
Alternative Branch 1: $x_{18,15}$, Objective 697

8. Solving TSP via Linear Programming © T. Sauerwald
Alternative Branch 1a: $x_{18,15} = 1$, Objective 701 (Valid Tour)

8. Solving TSP via Linear Programming © T. Sauerwald
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

9: valid tour 701

10: LP solution 698

\[ x_{18,15} = 1 \]
\[ x_{18,15} = 0 \]
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: ???

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

$x_{27,22} = 1$

9: valid tour 708

$x_{27,22} = 0$

10: LP solution 697.75

8. Solving TSP via Linear Programming © T. Sauerwald
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: ???

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
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 Elimination constraints to the LP?
  There are exponentially many of them!

- Should the search tree be explored by BFS or DFS?
  BFS may be more attractive, even though it might need more memory.

CONCLUDING REMARK

It is clear that we have left unanswered practically any question one might pose of a theoretical nature concerning the traveling-salesman problem; however, we hope that the feasibility of attacking problems involving a moderate number of points has been successfully demonstrated, and that perhaps some of the ideas can be used in problems of similar nature.
Conclusion (2/2)

- Eliminate Subtour 1, 2, 41, 42
- Eliminate Subtour 3 – 9
- Eliminate Subtour 10, 11, 12
- Eliminate Subtour 11 – 23
- Eliminate Subtour 13 – 23
- Eliminate Cut 13 – 17
- Eliminate Subtour 24, 25, 26, 27

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:

\[
S_1 = \{1, 2, 41, 42\} \quad S_5 = \{13, 14, \ldots, 23\}
\]
\[
S_2 = \{3, 4, \ldots, 9\} \quad S_6 = \{13, 14, 15, 16, 17\}
\]
\[
S_3 = \{1, 2, \ldots, 9, 29, 30, \ldots, 42\} \quad S_7 = \{24, 25, 26, 27\}
\]
\[
S_4 = \{11, 12, \ldots, 23\}
\]
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 linear programming problems using either primal or dual variants of the simplex method or the barrier interior
Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.1.0
with Simplex, Mixed Integer & Barrier Optimizers
5725-A06 5725-A29 5725-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>
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Solution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_2_1</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_42_1</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_3_2</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_4_3</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_5_4</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_6_5</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_7_6</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_8_7</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_9_8</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_10_9</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_11_10</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_12_11</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_13_12</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_14_13</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_15_14</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_16_15</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_17_16</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_18_17</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_19_18</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_20_19</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_21_20</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_22_21</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_23_22</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_24_23</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_25_24</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_26_25</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_27_26</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_28_27</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_29_28</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_30_29</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_31_30</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_32_31</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_33_32</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_34_33</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_35_34</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_36_35</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_37_36</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_38_37</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_39_38</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_40_39</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_41_40</td>
<td>1.000000</td>
</tr>
<tr>
<td>x_42_41</td>
<td>1.000000</td>
</tr>
</tbody>
</table>

All other variables in the range 1-861 are 0.
Randomised Algorithms
Lecture 9: Approximation Algorithms: MAX-3-CNF and Vertex-Cover

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

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover
Approximation Ratio

A randomised algorithm for a problem has approximation ratio $\rho(n)$, if for any input of size $n$, the expected cost (value) $E[C]$ of the returned solution and optimal cost $C^*$ satisfy:

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

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

Not covered here (non-examinable)
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover
MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
- **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:**

\[(x_1 \lor x_3 \lor \overline{x_4}) \land (x_1 \lor \overline{x_3} \lor \overline{x_5}) \land (x_2 \lor \overline{x_4} \lor x_5) \land (\overline{x_1} \lor x_2 \lor \overline{x_3})\]

\(x_1 = 1, \ x_2 = 0, \ x_3 = 1, \ x_4 = 0\) 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:
  \[
  Y_i = 1 \{ \text{clause } i \text{ is satisfied} \}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \mathbb{P} [ \text{clause } i \text{ is not satisfied} ] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{P} [ \text{clause } i \text{ is satisfied} ] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E} [ Y_i ] = \mathbb{P} [ Y_i = 1 ] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  \mathbb{E} [ Y ] = \mathbb{E} \left[ \sum_{i=1}^{m} Y_i \right] = \sum_{i=1}^{m} \mathbb{E} [ Y_i ] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8} \cdot m.
  \]

Linearity of Expectations

maximum number of satisfiable clauses is \( m \)
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 \( \frac{8}{7} \)-approximation algorithm.

**Corollary**

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least \( \frac{7}{8} \) of all clauses.

There is \( \omega \in \Omega \) such that \( Y(\omega) \geq \mathbb{E}[Y] \)

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 \( \frac{8}{7} \)-approximation algorithm.

One could prove that the probability to satisfy \( \left( \frac{7}{8} \right) \cdot m \) clauses is at least \( \frac{1}{8m} \).

\[
E[ Y ] = \frac{1}{2} \cdot E[ Y | x_1 = 1 ] + \frac{1}{2} \cdot E[ Y | x_1 = 0 ].
\]

\( Y \) is defined as in the previous proof.

One of the two conditional expectations is at least \( E[ Y ] \).

Algorithm: \textsc{Greedy-3-CNF}(\( \phi, n, m \))
\begin{enumerate}
\item \textbf{for} \( j = 1, 2, \ldots, n \)
\item Compute \( E[ Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 ] \)
\item Compute \( E[ Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 0 ] \)
\item Let \( x_j = v_j \) so that the conditional expectation is maximised
\item \textbf{return} the assignment \( v_1, v_2, \ldots, v_n \)
\end{enumerate}
Run of GREEDY-3-CNF(ϕ, n, m)

\[(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor \overline{x_3} \lor \overline{x_4})\]

\[
\begin{array}{c}
\text{x}_1 = 0 \\
\text{????} \quad 8.75 \\
\text{????} \quad 8.625 \\
\text{0???} \quad \text{x}_2 = 0 \\
\text{00??} \quad \text{x}_3 = 0 \\
\text{000?} \quad \text{0000} \\
\text{001?} \quad \text{0010} \\
\text{010?} \quad \text{0100} \\
\text{011?} \quad \text{0110} \\
\text{10??} \quad \text{x}_3 = 1 \\
\text{100?} \quad \text{1000} \\
\text{101?} \quad \text{1010} \\
\text{11??} \quad \text{x}_3 = 1 \\
\text{110?} \quad \text{1100} \\
\text{111?} \quad \text{1110} \\
\end{array}
\]
Run of **GREEDY-3-CNF**\((\varphi, n, m)\)

\[
1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land (\overline{x_2} \lor \overline{x_3}) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})
\]

---

**Go to Analysis**
Run of \textsc{Greedy-3-CNF}(\(\varphi, n, m\))

\[
1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x_3} \lor \overline{x_4})
\]
Run of GREEDY-3-CNF(\(\varphi, n, m\))

\[1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1\]
Run of GREEDY-3-CNF($\varphi, n, m$)

$$1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1$$

$\text{Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.}$
Run of GREEDY-3-CNF($\varphi, n, m$)

\[
(x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_4}) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_3} \lor x_4) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor \overline{x_3} \lor \overline{x_4})
\]

Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.
Analysis of GREEDY-3-CNF($\phi, n, m$)

**Theorem**

GREEDY-3-CNF($\phi, n, m$) 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:
    \[
    E\left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} E\left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
    \]

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,
    \[
    E\left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \right] \geq E\left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \right] \\
    \geq E\left[ Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \right] \quad \vdots \\
    \geq E\left[ Y \right] = \frac{7}{8} \cdot m.
    \]

This algorithm is deterministic.
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 35.6**

**Theorem**

$\text{GREEDY-3-CNF}(\phi, n, m)$ is a polynomial-time $8/7$-approximation.

**Theorem (Hastad'97)**

For any $\epsilon > 0$, there is no polynomial time $8/7 - \epsilon$ approximation algorithm of MAX3-CNF unless P=NP.

Essentially there is nothing smarter than just guessing!
Yes, my research has finally concluded...

So you said you have been studying the field of algorithms for MAX-3-SAT?

...the best approach is to **randomly guess** a solution.

Source of Image: Stefan Szeider, TU Vienna
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover
The Weighted Vertex-Cover Problem

**Vertex Cover Problem**

- **Given:** Undirected, vertex-weighted graph $G = (V, E)$
- **Goal:** Find a minimum-weight subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.

**Question:** How can we deal with graphs that have **negative** weights?

**Applications:**

- Every edge forms a **task**, and every vertex represents a person/machine which can execute that task
- **Weight** of a vertex could be salary of a person
- Perform all tasks with the **minimal amount of resources**
### A Greedy Approach working for Unweighted Vertex Cover

**APPROX-VERTEX-COVER**($G$)

1. $C = \emptyset$
2. $E' = G. E$
3. **while** $E' \neq \emptyset$
   4. let $(u, v)$ be an arbitrary edge of $E'$
   5. $C = C \cup \{u, v\}$
   6. remove from $E'$ every edge incident on either $u$ or $v$
7. **return** $C$

This algorithm is a **2-approximation** for unweighted graphs!
A Greedy Approach working for Unweighted Vertex Cover

**APPROX-VERTEX-COVER** \((G)\)

1. \( C = \emptyset \)
2. \( E' = G, E \)
3. **while** \( E' \neq \emptyset \)
   4. let \((u, v)\) be an arbitrary edge of \( E' \)
   5. \( C = C \cup \{u, v\} \)
   6. remove from \( E' \) every edge incident on either \( u \) or \( v \)
4. **return** \( C \)

---

**Figure 35.1** illustrates how **APPROX-VERTEX-COVER** operates on an example graph. The variable \( C \) contains the vertex cover being constructed. Line 1 initializes \( C \) to the empty set. Line 2 sets \( E' \) to be a copy of the edge set \( G, E \) of the graph. The loop of lines 3–6 repeatedly picks an edge \((u, v)\) from \( E' \), adds it to \( C \) using line 5, and removes from \( E' \) every edge incident on either \( u \) or \( v \). Line 7 **return** the set \( C \).

**Computed solution has weight 101**

Optimal solution has weight 4

This algorithm is a 2-approximation for unweighted graphs!
A Greedy Approach working for Unweighted Vertex Cover

\textsc{Approx-Vertex-Cover}(G)

1 \quad C = \emptyset

2 \quad E' = G.E

3 \quad \textbf{while} \ E' \neq \emptyset

4 \quad \textbf{let} (u, v) \text{ be an arbitrary edge of } E'

5 \quad C = C \cup \{u, v\}

6 \quad \text{remove from } E' \text{ every edge incident on either } u \text{ or } v

7 \quad \textbf{return} \ C

**Figure 35.1** The operation of \textsc{Approx-Vertex-Cover}.

- **(a)** The input graph \( G \), which has 7 vertices and 8 edges.
- **(b)** The edge \((b, c)\), shown heavy, is the first edge chosen by \textsc{Approx-Vertex-Cover}. Vertices \( b \) and \( c \), shown lightly shaded, are added to the set \( C \) containing the vertex cover being created. Edges \((a, b)\), \((c, e)\), and \((c, d)\), shown as dashed, are removed since they are no longer covered by some vertex in \( C \).
- **(c)** Edge \((e, f)\) is chosen; vertices \( e \) and \( f \) are added to \( C \).
- **(d)** Edge \((d, g)\) is chosen; vertices \( d \) and \( g \) are added to \( C \).
- **(e)** The set \( C \), which is the vertex cover produced by \textsc{Approx-Vertex-Cover}, contains the vertices \( b, c, d, e, f, g \).
- **(f)** The optimal vertex cover for this problem contains only three vertices: \( b, d, \) and \( e \).

This algorithm is a 2-approximation for unweighted graphs!
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

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

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

Linear Program

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

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

**APPROX-MIN-WEIGHT-VC** \((G, w)\)

1. \( C = \emptyset \)
2. compute \( \tilde{x} \), an optimal solution to the linear program
3. for each \( \nu \in V \)
4.     if \( \tilde{x}(\nu) \geq 1/2 \)
5.         \( C = C \cup \{\nu\} \)
6. return \( C \)

**Theorem 35.7**

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

is polynomial-time because we can solve the linear program in polynomial time
Example of **APPROX-MIN-WEIGHT-VC**

\[
\bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \quad \bar{x}(d) = 1, \quad \bar{x}(c) = 0
\]

\[
x(a) = x(b) = x(e) = 1, \quad x(d) = 1, \quad x(c) = 0
\]

Rounding

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 $x(u) + x(v) \geq 1 \implies$ at least one of $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2 \implies C$ 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-CNF

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

Weighted Set Cover

MAX-CNF
The Weighted Set-Cover 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 $C \subseteq \mathcal{F}$ s.t. $X = \bigcup_{S \in C} S$.

Sum over the costs of all sets in $C$

Remarks:
- generalisation of the weighted Vertex-Cover problem
- models resource allocation problems
Setting up an Integer Program

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

$$\begin{align*}
\text{minimize} & \quad \sum_{S \in F} c(S) y(S) \\
\text{subject to} & \quad \sum_{S \in F : x \in S} y(S) \geq 1 \text{ for each } x \in X \\
& \quad y(S) \in \{0, 1\} \text{ for each } S \in F
\end{align*}$$

**Linear Program**

$$\begin{align*}
\text{minimize} & \quad \sum_{S \in F} c(S) y(S) \\
\text{subject to} & \quad \sum_{S \in F : x \in S} y(S) \geq 1 \text{ for each } x \in X \\
& \quad y(S) \in [0, 1] \text{ for each } S \in F
\end{align*}$$
Setting up an Integer Program

0-1 Integer Program

\[
\begin{align*}
\text{minimize} & \quad \sum_{S \in \mathcal{F}} c(S)y(S) \\
\text{subject to} & \quad \sum_{S \in \mathcal{F} : x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\
& \quad y(S) \in \{0, 1\} \quad \text{for each } S \in \mathcal{F}
\end{align*}
\]

Linear Program

\[
\begin{align*}
\text{minimize} & \quad \sum_{S \in \mathcal{F}} c(S)y(S) \\
\text{subject to} & \quad \sum_{S \in \mathcal{F} : x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\
& \quad y(S) \in [0, 1] \quad \text{for each } S \in \mathcal{F}
\end{align*}
\]
Back to the Example

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!

Cost equals 8.5
Randomised Rounding

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

$\bar{y}(.)$: $1/2$ $1/2$ $1/2$ $1/2$ 1 $1/2$

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

- Let $C \subseteq F$ be a random set with each set $S$ being included independently with probability $\bar{y}(S)$.
- More precisely, if $\bar{y}$ denotes the optimal solution of the LP, then we compute an integral solution $y$ by:

$$y(S) = \begin{cases} 
1 & \text{with probability } \bar{y}(S) \\
0 & \text{otherwise.}
\end{cases}$$

for all $S \in F$.

- Therefore, $E[y(S)] = \bar{y}(S)$. 
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(\cdot)$</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Idea: Interpret the $y$-values as probabilities for picking the respective set.

Lemma

- The expected cost satisfies
  \[
  E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)
  \]

- The probability that an element $x \in X$ is covered satisfies
  \[
  P\left[ x \in \bigcup_{S \in c} S \right] \geq 1 - \frac{1}{e}.
  \]
Proof of Lemma

**Lemma**

Let $C \subseteq F$ be a random subset with each set $S$ being included independently with probability $y(S)$.

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

**Proof:**

- **Step 1:** The expected cost of the random set $C$

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

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

  $$P[x \notin \bigcup_{S \in C} S] = \prod_{S \in F : x \in S} P[S \notin C] = \prod_{S \in F : x \in S} (1 - y(S))$$

  $$\leq \prod_{S \in F : x \in S} e^{-y(S)} \quad \text{for any } x \in \mathbb{R}$$

  $$\bar{y} \text{ solves the LP!}$$
The Final Step

Let $C \subseteq F$ be a random subset with each set $S$ being included independently with probability $y(S)$.

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

Problem: Need to make sure that every element is covered!

Idea: Amplify this probability by taking the union of $\Omega(\log n)$ random sets $C$.

**Weighted Set Cover-LP** $(X, F, c)$

1: compute $\bar{y}$, an optimal solution to the linear program
2: $C = \emptyset$
3: repeat $2 \ln n$ times
4: for each $S \in F$
5: let $C = C \cup \{S\}$ with probability $\bar{y}(S)$
6: return $C$

clearly runs in polynomial-time!
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that $C$ 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
    \[
    P \left[ x \notin \bigcup_{S \in C} S \right] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    P \left[ X = \bigcup_{S \in C} S \right] = 1 - \mathbb{P} \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right] \geq 1 - \sum_{x \in X} P \left[ x \notin \bigcup_{S \in C} S \right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

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

\[\text{(Exercise Question (9/10).10)} \text{ gives a different perspective on the amplification procedure through non-linear randomised rounding.}\]
Analysis of \textbf{WEIGHTED SET COVER-LP}

- Theorem

  - With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
  - The expected approximation ratio is $2 \ln(n)$.

  By Markov's inequality, $P\left[ c(C) \leq 4 \ln(n) \cdot c(C^*) \right] \geq 1/2$.

  Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is valid and within a factor of $4 \ln(n)$ of the optimum.

  Probability could be further increased by repeating.

---

Typical Approach for Designing Approximation Algorithms based on LPs

[Exercise Question (9/10).10] gives a different perspective on the amplification procedure through non-linear randomised rounding.
Outline

Weighted Set Cover

MAX-CNF
MAX-CNF

Recall:

MAX-3-CNF Satisfiability

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

MAX-CNF Satisfiability (MAX-SAT)

- **Given:** CNF formula, e.g.: \((x_1 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor x_4 \lor \overline{x_5}) \land \cdots\)
- **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 $\ell$,

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

In particular, the guessing algorithm is a randomised 2-approximation.

Proof:

- First statement as in the proof of Theorem 35.6. For clause $i$ not to be satisfied, all $\ell$ 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,

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

maximize \[ \sum_{i=1}^{m} Z_i \]

subject to \[ \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \quad \text{for each } i = 1, 2, \ldots, m \]

- \( z_i \in \{0, 1\} \) for each \( i = 1, 2, \ldots, m \)
- \( y_j \in \{0, 1\} \) for each \( j = 1, 2, \ldots, n \)

- \( C_i^+ \) is the index set of the un-negated 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 $\ell$,\[ P[\text{clause } i \text{ is satisfied}] \geq \left( 1 - \left( 1 - \frac{1}{\ell} \right)^\ell \right) \cdot z_i. \]

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 $C_i = (x_1 \lor \cdots \lor x_\ell)$.

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

Arithmetic vs. geometric mean:\[ \frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \times \ldots \times a_k}. \]

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

\[ = 1 - \left( 1 - \frac{\sum_{j=1}^{\ell} \overline{y}_j}{\ell} \right)^\ell \geq 1 - \left( 1 - \frac{z_i}{\ell} \right)^\ell. \]
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

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

Proof of Lemma (2/2):

- So far we have shown:

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

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

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

- Therefore, $P \left[ \text{clause } i \text{ is satisfied} \right] \geq \beta_\ell \cdot \overline{Z}_i. \quad \Box$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

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

Theorem

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

Proof of Theorem:

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

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

By Lemma

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

LP solution at least as good as optimum
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

\[
\text{HYBRID-MAX-CNF}(\varphi, n, m) \\
1: \text{Let } b \in \{0, 1\} \text{ be the flip of a fair coin} \\
2: \text{If } b = 0 \text{ then perform random guessing} \\
3: \text{If } b = 1 \text{ then perform randomised rounding} \\
4: \text{return the computed solution}
\]

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(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot \bar{z}_i\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot \bar{z}_i\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot \bar{z}_i\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot \bar{z}_i + \frac{1}{2} \cdot \beta_\ell \cdot \bar{z}_i\).
- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure).
- \(\Rightarrow\) HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with prob. at least \(3/4 \cdot \bar{z}_i\). \(\square\)
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-CNDF, where each clause has a non-negative weight.

- Even MAX-2-CNDF (every clause has length 2) is NP-hard!
Outline

Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem
Origin of Graph Theory

Leonhard Euler (1707-1783)

Seven Bridges at Königsberg 1737

Is there a tour which crosses each bridge exactly once?

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 (communities) 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

\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0
\end{bmatrix}
\]

- Eigenvalues
- Eigenvectors
- Inverse
- Determinant
- Matrix-powers
- ...

---

11. Spectral Graph Theory © T. Sauerwald
Introduction to (Spectral) Graph Theory and Clustering
Outline

Introduction to (Spectral) Graph Theory and Clustering

Matrices, Spectrum and Structure

A Simplified Clustering Problem
Adjacency Matrix

Let $G = (V, E)$ be an undirected graph. The adjacency matrix of $G$ is the $n$ by $n$ matrix $A$ defined as

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

Properties of $A$:

- The sum of elements in each row/column $i$ equals the degree of the corresponding vertex $i$, $\deg(i)$
- Since $G$ is undirected, $A$ is symmetric
Eigenvalues and Graph Spectrum of A

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

$$Mx = \lambda x.$$ 

We call $x$ an eigenvector of $M$ corresponding to the eigenvalue $\lambda$.

Graph Spectrum

An undirected graph $G$ is $d$-regular if every degree is $d$, i.e., every vertex has exactly $d$ connections.

Let $A$ be the adjacency matrix of a $d$-regular graph $G$ with $n$ vertices. Then, $A$ 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 $G$.

$= \text{orthogonal and normalised}$

Remark: For symmetric matrices we have algebraic multiplicity $=$ geometric multiplicity (otherwise $\geq$)
Example 1

**Question:** What are the Eigenvalues and Eigenvectors?

**Bonus:** Can you find a short-cut to \( \det(A - \lambda \cdot I) \)?

**Solution:**

The three eigenvalues are \( \lambda_1 = \lambda_2 = -1, \lambda_3 = 2 \).

The three eigenvectors are (for example):

\[ f_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \\
\begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}, \\
\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \]

\[
A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}
\]
Example 1

**Question:** What are the Eigenvalues and Eigenvectors?

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

**Solution:**

- The three eigenvalues are $\lambda_1 = \lambda_2 = -1, \lambda_3 = 2$.
- The three eigenvectors are (for example):

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

\[
A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}
\]
Let $G = (V, E)$ be a $d$-regular undirected graph. The (normalised) Laplacian matrix of $G$ is the $n$ by $n$ matrix $L$ defined as

$$L = I - \frac{1}{d} A,$$

where $I$ is the $n \times n$ identity matrix.

**Question:** What is the matrix $\frac{1}{d} \cdot A$?
Laplacian Matrix

Let $G = (V, E)$ be a $d$-regular undirected graph. The (normalised) Laplacian matrix of $G$ is the $n$ by $n$ matrix $L$ defined as

$$L = I - \frac{1}{d}A,$$

where $I$ is the $n \times n$ identity matrix.

Properties of $L$:
- The sum of elements in each row/column equals zero
- $L$ is symmetric
Correspondence between Adjacency and Laplacian Matrix

\( A \) and \( L \) have the same set of eigenvectors.

**Exercise:** Prove this correspondence. Hint: Use that \( L = I - \frac{1}{d} A \).

*[Exercise 11/12.1]*
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 $x \in \mathbb{C}^n \setminus \{0\}$ such that

$$Mx = \lambda x.$$ 

We call $x$ an eigenvector of $M$ corresponding to the eigenvalue $\lambda$.

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 $f_1, \ldots, f_n$. These eigenvalues associated with their multiplicities constitute the spectrum of $G$. 
Lemma

Let $L$ 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 $1$
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,

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

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

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

minimised by an eigenvector $f_1$ for $\lambda_1$

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

minimised by $f_2$
Quadratic Forms of the Laplacian

Let $L$ be the Laplacian matrix of a $d$-regular graph $G = (V, E)$ with $n$ vertices. For any $x \in \mathbb{R}^n$,

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

**Proof:**

$$x^T L x = x^T \left( I - \frac{1}{d} A \right) x = x^T x - \frac{1}{d} x^T Ax$$

$$= \sum_{u \in V} x_u^2 - \frac{2}{d} \sum_{\{u,v\} \in E} x_u x_v$$

$$= \frac{1}{d} \sum_{\{u,v\} \in E} (x_u^2 + x_v^2 - 2x_u x_v)$$

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

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

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

Question: What are the Eigenvectors with Eigenvalue 0 of \( L \)?

\[
A = \begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\end{pmatrix}
\]

\[
L = \begin{pmatrix}
1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
-\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
-\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\
0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\
0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \\
\end{pmatrix}
\]

Solution:
Two smallest eigenvalues are \( \lambda_1 = \lambda_2 = 0 \).

The corresponding two eigenvectors are:
\[
f_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \\
f_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}
\]

Thus we can easily solve the simplified clustering problem by computing the eigenvectors with eigenvalue 0.
Example 2

Question: What are the Eigenvectors with Eigenvalue 0 of \( L \)?

\[
A = \begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 
\end{pmatrix}
\]

\[
L = \begin{pmatrix}
1 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
-\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
-\frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\
0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \n\end{pmatrix}
\]

Solution:

- Two smallest eigenvalues are \( \lambda_1 = \lambda_2 = 0 \).
- The corresponding two eigenvectors are:

\[ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix} \]

Thus we can easily solve the simplified clustering problem by computing the eigenvectors with eigenvalue 0.

Next Lecture: A fine-grained approach works even if the clusters are \textit{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. (\(\implies\) \(\text{cc}(G) \leq \text{mult}(0)\)). We will show:
   \(G\) has exactly \(k\) connected comp. \(C_1, \ldots, C_k\) \(\Rightarrow\) \(\lambda_1 = \cdots = \lambda_k = 0\)
   - 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 \(\chi_{C_i}\)'s are orthogonal
   - \(\chi_{C_i}^T L \chi_{C_i} = \frac{1}{d} \cdot \sum_{\{u, v\} \in E} (\chi_{C_i}(u) - \chi_{C_i}(v))^2 = 0\) \(\Rightarrow\) \(\lambda_1 = \cdots = \lambda_k = 0\)

2. (\(\iff\) \(\text{cc}(G) \geq \text{mult}(0)\)). We will show:
   \(\lambda_1 = \cdots = \lambda_k = 0\) \(\Rightarrow\) \(G\) 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.
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

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

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

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

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

- $\phi(S) = \frac{5}{9}$
- $\phi(G) \in [0, 1]$ and $\phi(G) = 0$ iff $G$ is disconnected
- If $G$ is a complete graph, then $e(S, V \setminus S) = |S| \cdot (n - |S|)$ and $\phi(G) \approx 1/2$.
What is the relationship between $\phi(G)$ and $\lambda_2(G)$ for connected graphs?
$\lambda_2$ 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 $\lambda_2$ and Conductance

Let $G$ be a $d$-regular undirected graph and $\lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of its Laplacian matrix. Then,

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$ 

Cheeger’s inequality

Spectral Clustering:

1. Compute the eigenvector $x$ corresponding to $\lambda_2$
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):

- By the Courant-Fischer Formula,

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

- 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

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

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

\[ \leq \frac{1}{d} \cdot 2 \cdot \frac{|E(S, V \setminus S)|}{|S|} = 2 \cdot \phi(G). \]
Illustration on a small Example

\[
A = \begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{pmatrix}
\]

\[
L = \begin{pmatrix}
1 & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 0 \\
0 & 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\
-\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 & 0 & 0 & -\frac{1}{3} \\
-\frac{1}{3} & 0 & -\frac{1}{3} & 1 & 0 & 0 & -\frac{1}{3} & 0 \\
0 & -\frac{1}{3} & 0 & 0 & 1 & -\frac{1}{3} & 0 & -\frac{1}{3} \\
0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} \\
-\frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 & 1 & 0 \\
0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} & -\frac{1}{3} & 0 & 1
\end{pmatrix}
\]

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

\[v = (-0.425, +0.263, -0.263, -0.425, +0.425, +0.425, -0.263, +0.263)^T\]

\[\text{Sweep: 4}\]

\[\text{Conductance: 0.166}\]
Physical Interpretation of the Minimisation Problem

- For each edge \{u, v\} ∈ E(G), add spring between pins at \(x_u\) and \(x_v\)
- The potential energy at each spring is \((x_u - x_v)^2\)
- Courant-Fisher characterisation:

\[
\lambda_2 = \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T L x}{x^T x} = \frac{1}{d} \cdot \min_{x \in \mathbb{R}^n \atop \|x\|_2^2 = 1, x \perp 1} (x_u - x_v)^2
\]

- In our example, we found out that \(\lambda_2 \approx 0.25\)
- The eigenvector \(x\) on the last slide is normalised (i.e., \(\|x\|^2 = 1\)). Hence,

\[
\lambda_2 = \frac{1}{3} \cdot \left( (x_1 - x_3)^2 + (x_1 - x_4)^2 + (x_1 - x_7)^2 + \cdots + (x_6 - x_8)^2 \right) \approx 0.25
\]
Let us now look at an example of a non-regular graph!
The (normalised) Laplacian matrix of $G = (V, E, w)$ is the $n$ by $n$ matrix

$$L = I - D^{-1/2}AD^{-1/2}$$

where $D$ is a diagonal $n \times n$ matrix such that $D_{uu} = \text{deg}(u) = \sum_{v: \{u, v\} \in E} w(u, v)$, and $A$ is the weighted adjacency matrix of $G$.

- $L_{uv} = -\frac{w(u,v)}{\sqrt{d_u d_v}}$ for $u \neq v$
- $L$ is symmetric
- If $G$ is $d$-regular, $L = I - \frac{1}{d} \cdot A$. 

### Example

$$L = \begin{pmatrix}
1 & -16/25 & 0 & -9/20 \\
-16/25 & 1 & -9/20 & 0 \\
0 & -9/20 & 1 & -7/16 \\
-9/20 & 0 & -7/16 & 1
\end{pmatrix}$$
Conductance and Spectral Clustering (General Version)

Let $G = (V, E, w)$ and $\emptyset \subset S \subset V$. The conductance (edge expansion) of $S$ is

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

where $w(S, S^c) := \sum_{u \in S, v \in S^c} w(u, v)$ and $\text{vol}(S) := \sum_{u \in S} d(u)$. Moreover, the conductance (edge expansion) of $G$ is

$$\phi(G) := \min_{\emptyset \neq S \subset V} \phi(S).$$

Spectral Clustering (General Version):

1. Compute the eigenvector $x$ corresponding to $\lambda_2$ and $y = D^{-1/2}x$.
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

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

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

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
Best Solution found by Spectral Clustering

For the complete animation, see the full slides.

- Step: 78
- Threshold: $-0.027$
- Partition Sizes: $78/122$
- Cut Edges: 84
- Conductance: 0.145
Clustering induced by Blocks

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

(0, 0)
Additional Example: Stochastic Block Models with 3 Clusters

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

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

$|V| = 300, |S_i| = 100$

$p = 0.08, q = 0.01.$

Spectral embedding

Output of Spectral Clustering
How to Choose the Cluster Number $k$

- If $k$ is unknown:
  - small $\lambda_k$ means there exist $k$ sparsely connected subsets in the graph
    (recall: $\lambda_1 = \ldots = \lambda_k = 0$ means there are $k$ connected components)
  - large $\lambda_{k+1}$ means all these $k$ subsets have “good” inner-connectivity properties (cannot be divided further)

  \[ \Rightarrow \text{choose smallest } k \geq 2 \text{ so that the spectral gap } \lambda_{k+1} - \lambda_k \text{ is “large”} \]

- 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)
Another Example

- nodes represent math topics taught within 4 weeks of a Mathcamp
- node colours represent to the week in which they thought
- teachers were asked to assign weights in 0 – 10 indicating how closely related two classes are

(many thanks to Kalina Jasinska)
### Summary: Spectral Clustering

#### Illustration on a (very) small Example

Given any graph (adjacency matrix) and graph spectrum (computable in poly-time):
- \( \lambda_2 \) (relates to connectivity)
- \( \lambda_n \) (relates to bipartiteness)

#### Cheeger’s Inequality
- relates \( \lambda_2 \) to conductance
- unbounded approximation ratio
- effective in practice

\[
\begin{align*}
\min_{x \in \mathbb{R}^n \setminus \{0\}} & \quad \frac{\sum_{u \sim v} (x_u - x_v)^2}{\sum_u x_u^2} \\
\text{s.t.} & \quad x \perp 1
\end{align*}
\]
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 (non-examinable)

- Which graph has a “cluster-structure”?
- Which graph mixes faster?
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 $\pi P = \pi$.

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 $x$,

$$\left\| xP^t - \pi \right\|_2 \leq \lambda^t,$$

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

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

$$\left\| xP^t - \pi \right\|_{tv} \leq \frac{1}{4}.$$
Proof of Lemma (non-examinable)

- Express $x$ in terms of the orthonormal basis of $P$, $v_1 = \pi, v_2, \ldots, v_n$:
  
  $$x = \sum_{i=1}^{n} \alpha_i v_i.$$ 

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

  $\Rightarrow$

  $$\|xP - \pi\|^2 = \left\| \left( \sum_{i=1}^{n} \alpha_i v_i \right) P - \pi \right\|^2$$

  $$= \left\| \pi + \sum_{i=2}^{n} \alpha_1 v_i - \pi \right\|^2$$

  $$= \left\| \sum_{i=2}^{n} \alpha_i v_i \right\|^2$$

  since the $v_i$'s are orthogonal

  $$\leq \lambda^2 \sum_{i=2}^{n} \|\alpha_i v_i\|^2 = \lambda^2 \left\| \sum_{i=2}^{n} \alpha_i v_i \right\|^2 = \lambda^2 \|x - \pi\|^2$$

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

$$\|x - \pi\|^2 + \|\pi\|^2 = \|x\|^2 \leq 1$$
Some References on Spectral Graph Theory and Clustering

Thank you and Best Wishes for the Exam!