Approximation Algorithms

Thomas Rothvoß

Institute of Mathematics
EPFL, Lausanne
Spring 2010
• Introduction (page 3)
• Steiner tree (page 8)
• $k$-Center (page 13)
• Traveling Salesman Problem (page 22)
• The Capacitated Vehicle Routing Problem (page 31)
• Set Cover (page 35)
• Set Cover via LPs (page 38)
• Insertion: Linear Programming (page 44)
• Weighted Vertex Cover (page 60)
• Insertion: Algorithmic probability theory (page 66)
• Minimizing Congestion (page 75)
• Knapsack (page 91)
• Multi Constraint Knapsack (page 96)
• Bin Packing (page 102)
  • The algorithm of Karmarkar & Karp (page 109)
• Minimum Makespan Scheduling (page 128)
• Scheduling on Unrelated Parallel Machines (page 134)
• Multiprocessor Scheduling with Precedence Constraints (page 143)
• Euclidean TSP (page 148)
• Tree Embeddings (page 174)
• Introduction into Primal dual algorithms (page 199)
• Steiner Forest (page 204)
• Facility Location (page 227)
• Insertion: Semidefinite Programming (page 244)
• MaxCut (page 254)
• Max2Sat (page 263)
• Budgeted Spanning Tree (page 270)
• $k$-Median (page 281)
Part 1
Introduction

Source: Approximation Algorithms (Vazirani, Springer Press)
Why approximation algorithms?

**Task:** Solve \textbf{NP}-hard optimization problem \( A \)
\( \rightarrow \) no efficient algorithm (unless \( \textbf{NP} = \textbf{P} \))

**Possible approaches:**

- exponential time algorithms \( \rightarrow \) some theory but too slow and no lower bounds
- heuristic \( \rightarrow \) fast, easy but no guarantee, not much theory
- approximation algorithms \( \rightarrow \) rich theory in many cases good lower bounds

**Running times:** \( n = \) number of objects in instance, \( B \) biggest appearing number, \( \varepsilon > 0 \) constant

- \textbf{exponential:} \( 2^n, n \cdot B \)
- \textbf{polynomial:} \( n^2, n^{100}, n \cdot \log B, n \cdot 2^{1/\varepsilon}, n^{O(1/\varepsilon)}^{O(1/\varepsilon)} \)
Basic definitions

Definition
Let $\Pi$ be an optimization problem and $I$ is instance for $A$. Then $OPT_\Pi(I)$ is the value of the optimum solution.

Definition
Let $\alpha \geq 1$. $A$ is an $\alpha$-approximation algorithm for a minimization problem $\Pi$ if

$$A(I) \leq \alpha \cdot OPT_\Pi(I) \quad \forall \text{ instances } I$$

where $A(I)$ is the value of the solution, that $A$ returns for $I$.

- Typical values for $\alpha$: $1.5, 2, O(1), O(\log n)$
- Usually we omit $\Pi$ and $I$ in $OPT_\Pi(I)$
- For a maximization problem: $A(I) \geq \frac{1}{\alpha} \cdot OPT_\Pi(I)$
- **Attention:** Sometimes in literature $\alpha < 1$ for maximization problems. For example $\frac{1}{2}$-apx means $A(I) \geq \frac{1}{2}OPT_\Pi(I)$
Definition PTAS

A_\varepsilon is a polynomial time approximation scheme (PTAS) for a minimization problem \Pi if

$$A_\varepsilon(I) \leq (1 + \varepsilon) \cdot OPT(I) \quad \forall \text{ instances } I$$

and for every fixed \(\varepsilon > 0\), the running time of \(A_\varepsilon\) is polynomial in the input size.

Typical running times: \(O(n/\varepsilon), 2^{1/\varepsilon} n^2 \log^2(B), n^{O(1/\varepsilon)}\)
Definition FPTAS

**Definition**

$A_\varepsilon$ is a fully polynomial time approximation scheme (FPTAS) for a minimization problem $\Pi$ if for every $\varepsilon > 0$

$$A_\varepsilon(I) \leq (1 + \varepsilon) \cdot OPT(I) \quad \forall \text{ instances } I$$

and the running time of $A_\varepsilon$ is polynomial in the input size and $1/\varepsilon$.

- Typical running time: $O(n^3/\varepsilon^2)$
Part 2
Steiner tree

Source: Approximation Algorithms (Vazirani, Springer Press)
Problem: **Steiner tree**

- **Given:** Undirected graph $G = (V, E)$, metric cost function $c : E \to \mathbb{Q}_+$, terminals $R \subseteq V$
- **Find:** Minimum cost tree $T$ connecting all terminals $R$:

$$OPT = \min\{c(T) \mid T \text{ spans } R\}$$

- $c(T) := \sum_{e \in T} c_e$
- **metric:** $\forall u, v, w \in V : c_{uw} \leq c_{uv} + c_{vw}$ (triangle inequality)
Steiner tree (2)

Fact
If $R = V$, then Steiner Tree is just the Minimum Spanning Tree Problem which can be solved optimally by picking greedily the cheapest edges (without closing a cycle).

Algorithm:
1. Compute the minimum spanning tree $T$ on $R$
2. Return $T$

Theorem
The algorithm gives a 2-approximation.
Proof of approximation guarantee

- **Claim:** \( \exists \) spanning tree of cost \( \leq 2 \cdot \text{OPT} \)
- Let \( T^* \) be optimum Steiner tree
- Double the edges of \( T^* \)
- Observe: Degrees now even \( \Rightarrow \exists \) Euler tour \( \mathcal{E} \) visiting each terminal

**Theorem (Euler)**

*Given an undirected, connected graph \( G = (V, E) \). Then \( G \) has an Euler tour (tour containing each edge exactly once) if and only if \( |\delta(v)| \) is even for all \( v \in V \).*

- Shortcut \( \mathcal{E} \) such that each terminal is visited once
- Remove an edge \( \Rightarrow \) spanning tree of cost \( \leq 2 \cdot c(T^*) \)  

\[ \begin{array}{ccc}
\square & \square & \square \\
\square & \square & \square \\
\square & \square & \square \\
\end{array} \quad \Rightarrow \quad \begin{array}{ccc}
\square & \square & \square \\
\square & \square & \square \\
\square & \square & \square \\
\end{array} \quad \Rightarrow \quad \begin{array}{ccc}
\square & \square & \square \\
\square & \square & \square \\
\square & \square & \square \\
\end{array} \]
State of the art

Known results:

- There is a 1.39-approximation.
- For quasi-bipartite graphs (no Steiner nodes incident):
  1.22-apx
- No $< \frac{96}{95}$-apx unless $\text{NP} = \text{P}$. 
Part 3

$k$-Center

Source: Approximation Algorithms (Vazirani, Springer Press)
**Problem: \( k\text{-Center} \)**

- **Given:** Undirected, metric graph \( G = (V, E) \), \( k \in \mathbb{N} \). Define

\[
\ell(v, F) := \min_{u \in F} c_{uv}
\]

- **Find:** \( k \) many centers \( F \subseteq V \) that minimize the maximum distance from any \( v \in V \) to the nearest center:

\[
OPT = \min_{F \subseteq V, |F| = k} \max_{v \in V} \{ \ell(v, F) \}
\]
The algorithm

Algorithm:

1. Guess $OPT \in \{c_{uv} \mid u, v \in V\}$
2. $F := \emptyset$
3. REPEAT
   4. IF $\exists v \in V : \ell(v, F) > 2 \cdot OPT$ THEN $F := F \cup \{v\}$
   ELSE RETURN $F$
The algorithm

**Algorithm:**

1. Guess $OPT \in \{c_{uv} | u, v \in V\}$
2. $F := \emptyset$
3. **REPEAT**
   
   **(4) IF** $\exists v \in V : \ell(v, F) > 2 \cdot OPT$ **THEN** $F := F \cup \{v\}$
   
   **ELSE** RETURN $F$
The algorithm

Algorithm:

1. Guess $OPT \in \{c_{uv} \mid u, v \in V\}$
2. $F := \emptyset$
3. Repeat
   4. If $\exists v \in V: \ell(v, F) > 2 \cdot OPT$ then $F := F \cup \{v\}$
      Else return $F$
Guessing

For simplicity we sometimes guess parameters:

**Algorithm with guessing:**

1. Guess a parameter $m$
2. ... compute a solution $S$ using $m$ ...
3. return $S$

**Algorithm without guessing:**

1. FOR all choices of $m$ DO
   
   2. ... compute a solution $S(m)$ ...
3. return the best found solution $S(m)$

- Still polynomial if the domain of $m$ is polynomial
- Typical guesses: $OPT$, $O(1)$ many nodes in a graph
The analysis

**Theorem**

One has $|F| \leq k$ and $\ell(v, F) \leq 2 \cdot OPT$ for all $v \in V$.

- $\ell(v, F) \leq 2 \cdot OPT$, otherwise algo would not have stopped.
- Remains to show $|F| \leq k$.
- Let $F^* \subseteq V, |F^*| = k$ be optimum solution.
- Observe: $c_{uv} > 2 \cdot OPT \forall u, v \in F : u \neq v$
- Hence the centers in $F^*$ that serve $u$ and $v$ must be different $\Rightarrow |F| \leq |F^*| \leq k$. 

![Diagram](image-url)
Dominating Set

Problem: **DOMINATING SET**

- **Given:** Undirected graph $G = (V, E)$
- **Find:** Dominating set $U \subseteq V$ of minimum size

$$OPT_{DS} = \min\{ |U| \mid U \subseteq V, U \cup \bigcup_{u \in U} \delta(u) = V \}$$

![Diagram of a graph with a dominating set highlighted]

Theorem

*Given* $(G, k)$, it is **NP**-hard to decide, whether $OPT_{DS} \leq k$. 
Hardness of $k$-Center

Theorem

Unless $\text{NP} = \text{P}$, for all $\epsilon > 0$, there is no $(2 - \epsilon)$-approximation algorithm for $k$-Center.

- Let $(G, k)$ be DOMINATINGSET instance.
- Suppose $A$ is a $(2 - \epsilon)$-algorithm for $k$-Center
- Define complete graph $G'$ on nodes $V$ with
  \[ c(u, v) := \begin{cases} 
  1 & \text{if } (u, v) \in E \\
  2 & \text{otherwise} 
  \end{cases} \]

- $\exists$ DS of size $\leq k \Rightarrow k$-Center solution with value 1
- $\exists$ $k$-CENTER solution with value $\leq 1 \Rightarrow \exists$ DS of size $\leq k$
- Run $A$ on $G'$:
  - $A(G') < 2 \Rightarrow A(G') = 1 \Rightarrow$ answer to DS instance is YES
  - $A(G') \geq 2 \Rightarrow$ answer is NO
PART 4

TRAVELING SALESMAN PROBLEM

**Problem:** Traveling Salesman Problem (TSP)

- **Given:** Undirected graph \( G = (V, E) \) with metric cost \( c : E \to \mathbb{Q}_+ \)
- **Find:** Minimum cost tour visiting all nodes

\[
\min_{\text{tour } \pi : V \to V} \left\{ \sum_{v \in V} c(v, \pi(v)) \right\}
\]
A 2-approximation for TSP

Algorithm:

1. Compute an MST $T$ on $G$
2. Double the edges in $T$
3. Compute Euler tour $\mathcal{E}$ using edges in $T$
4. Shortcut to obtain a tour $\pi$

Theorem

Algorithm yields a 2-apx.

- Let $\pi^*$ be optimum tour
- $\exists$ a spanning tree on $G$ of cost $c(T) \leq OPT$ (just delete an arbitrary edge from $\pi^*$)
- Degrees are even after doubling, hence $\mathcal{E}$ exists and $c(\mathcal{E}) \leq 2 \cdot OPT$
- $c(\pi) \leq 2 \cdot OPT$ ($G$ is metric, hence shortcutting does not increase the cost)

□
A 3/2-approximation for TSP

Algorithm (Christofides): 
1. Compute an MST $T$
2. Find min cost perfect matching $M$ on nodes $V^{\text{odd}} \subseteq V$ with odd degree in $T$
3. Find Euler tour in $T \cup M$.
4. Return $\pi$ obtained by shortcutting the Euler tour

Reminder

A perfect matching in an undirected graph $G' = (V', E')$ is an edge set $M \subseteq E'$ with $|\delta_M(v)| = 1 \ \forall v \in V'$. The cheapest perfect matching can be found in poly-time.
A 3/2-approximation for TSP (2)

**Theorem**

The algorithm gives a 3/2-apx.

- Again $c(T) \leq OPT$
- $V^{\text{odd}} := \{v \in V \mid |\delta_T(v)| \text{ odd}\}$.
- **Claim:** $|V^{\text{odd}}|$ is even because

$$\left| V^{\text{odd}} \right| \equiv_2 \sum_{v \in V^{\text{odd}}} |\delta_T(v)| \equiv_2 \sum_{v \in V} |\delta_T(v)| \equiv_2 0$$

![Diagram of a graph with a tree $T$ and set $V^{\text{odd}}$]
A 3/2-approximation for TSP (3)

- Let \( \pi^* \) be optimum tour. Obtain shortcutted tour \( \pi^{\text{odd}} \) on \( V^{\text{odd}} \): \( c(\pi^{\text{odd}}) \leq OPT \).
- Partition \( \pi^{\text{odd}} \) into 2 matchings \( M_1, M_2 \) on \( V^{\text{odd}} \).
- Let \( M \in \{M_1, M_2\} \) be the cheaper of both matchings.
- \( c(M) \leq \frac{1}{2} c(\pi^{\text{odd}}) \leq \frac{1}{2} OPT \).
- In \( T \cup M \) all nodes have even degree, hence \( T \cup M \) contains an Euler tour of cost \( \leq c(T) + c(M) \leq \frac{3}{2} OPT \).
A $3/2$-approximation for TSP (3)

- Let $\pi^*$ be optimum tour. Obtain shortcutted tour $\pi^{\text{odd}}$ on $V^{\text{odd}}$: $c(\pi^{\text{odd}}) \leq \text{OPT}$.
- Partition $\pi^{\text{odd}}$ into 2 matchings $M_1, M_2$ on $V^{\text{odd}}$
- Let $M \in \{M_1, M_2\}$ be the cheaper of both matchings
- $c(M) \leq \frac{1}{2}c(\pi^{\text{odd}}) \leq \frac{1}{2}\text{OPT}$
- In $T \cup M$ all nodes have even degree, hence $T \cup M$ contains an Euler tour of cost $\leq c(T) + c(M) \leq \frac{3}{2}\text{OPT}$.
A 3/2-approximation for TSP (3)

- Let \( \pi^* \) be optimum tour. Obtain shortcutted tour \( \pi^{\text{odd}} \) on \( V^{\text{odd}} \): \( c(\pi^{\text{odd}}) \leq \text{OPT} \).
- Partition \( \pi^{\text{odd}} \) into 2 matchings \( M_1, M_2 \) on \( V^{\text{odd}} \).
- Let \( M \in \{ M_1, M_2 \} \) be the cheaper of both matchings.
- \( c(M) \leq \frac{1}{2} c(\pi^{\text{odd}}) \leq \frac{1}{2} \text{OPT} \).
- In \( T \cup M \) all nodes have even degree, hence \( T \cup M \) contains an Euler tour of cost \( \leq c(T) + c(M) \leq \frac{3}{2} \text{OPT} \).
Open Problems on TSP

Open Problem

- Is there a $< 3/2$-apx for TSP?
- Held-Karp LP relaxation is conjectured to have integrality gap $4/3$.
- No $\left(\frac{5381}{5380} - \varepsilon\right)$-apx even if $c_e \in \{1, 2\}$
Part 5
The Capacitated Vehicle Routing Problem

Source: Bounds and Heuristics for capacitated routing problems
(Haimovich, Rinnooy Kan)
http://www.jstor.org/stable/3689422
The Capacitated Vehicle Routing Problem

**Problem:** CVRP

- **Given:** Undirected graph $G = (C \cup \{r\}, E)$ with metric costs $c : E \to \mathbb{Q}_+$, depot $r$, clients $C$ and vehicle capacity $k$
- **Find:** A tour $\pi$ of minimal cost which visits all clients at least once, but must revisit the depot after each $\leq k$ client visits

**Assume:** $|C| = \mathbb{Z} \cdot k$ (otherwise add clients at the depot)
A 5/2-apx for CVRP

Algorithm:

(1) Compute a 3/2-approximate TSP tour $\pi$ on clients
(2) Let $v_0, \ldots, v_{n-1}$ be clients in visiting order
(3) Choose randomly a starting node $v_i^*$
(4) Starting from $v_i^*$ revisit $r$ every $k$ many clients (i.e. augment the tour with edges $r \rightarrow v_i, v_{i-1} \rightarrow r$ if $i \equiv_k i^*$) to obtain a CVRP solution $\pi'$
The analysis

**Lemma**

\[ E[APX] \leq \frac{5}{2} OPT \]

- Opt. TSP tour costs \( OPT_{TSP} \leq OPT \) hence \( c(\pi) \leq \frac{3}{2}OPT \)
- \( \Pr[\text{need edge } (r, v_i)] = \frac{2}{k} \)
- \( E[APX] \leq c(\pi) + \frac{2}{k} \sum_{v \in C} c(r, v) \)

- Look at a subtour in optimum CVRP solution. Send \( k/2 \) clients [counter-]clockwise to \( r \): edges in subtour used \( \leq k/2 \) times
  \( \Rightarrow \sum_{v \in C} c(v, r) \leq \frac{k}{2} OPT \)

\[
E[APX] \leq c(\pi) + \frac{2}{k} \sum_{v \in C} c(r, v) \leq \frac{3}{2} OPT + \frac{2}{k} \cdot \frac{k}{2} OPT = \frac{5}{2} OPT
\]
Part 6
Set Cover

Source: Approximation Algorithms (Vazirani, Springer Press)
Set Cover

**Problem:** Set Cover

- **Given:** Elements $U := \{1, \ldots, n\}$, sets $S_1, \ldots, S_m \subseteq U$ with cost $c(S_i)$
- **Find:**

$$OPT = \min_{I \subseteq \{1, \ldots, m\}} \left\{ \sum_{i \in I} c(S_i) \mid \bigcup_{i \in I} S_i = U \right\}$$

**Greedy algorithm:**

1. $I := \emptyset$
2. WHILE not yet all elements covered DO
   3. $price(S) := \frac{c(S)}{|S \setminus \bigcup_{i \in I} S_i|}$
   4. $I := I \cup \{ \text{ set } S \text{ with minimum } price(S) \}$

**Theorem**

The greedy algorithm yields a $O(\log n)$-approximation.
Analysis

- Let \( e_1, \ldots, e_n \) be elements in the order of covering.
- Suppose \( S \ (S \in I) \) newly covered \( e_k, \ldots, e_\ell \)

\[
\begin{align*}
&\text{\( n-k+1 \) elements} \\
&\{ e_1, e_2, e_3, \ldots, e_k, \ldots, e_j, \ldots, e_\ell, \ldots, e_n \} \\
&\text{covered by } S
\end{align*}
\]

- Define \( \text{price}(e_j) := \text{price}(S) \) for \( j \in \{k, \ldots, \ell\} \).
- Consider the iteration, when \( S \) was chosen: Still \( n - k + 1 \) elements where uncovered and it was still possible to cover them all at cost \( \text{OPT} \). Since \( S \) minimizes the price:

\[
\text{price}(e_j) = \text{price}(e_k) \leq \frac{\text{OPT}}{n - k + 1} \leq \frac{\text{OPT}}{n - j + 1}
\]

- Finally

\[
\text{APX} = \sum_{j=1}^{n} \text{price}(e_j) \leq \sum_{j=1}^{n} \frac{\text{OPT}}{n - j + 1} = \text{OPT} \cdot \sum_{j=1}^{n} \frac{1}{j} = O(\log n) \cdot \text{OPT}
\]
**Part 7**

**Set Cover via LPs**

**Source:** *Approximation Algorithms* (Vazirani, Springer Press)
A linear program for **SetCover**

Introduce decision variables

\[ x_i = \begin{cases} 
1 & \text{take set } S_i \\
0 & \text{otherwise}
\end{cases} \]

Formulate **SetCover** as integer linear program:

\[
\min \sum_{i=1}^{m} c(S_i)x_i \quad \text{(ILP)}
\]

\[
\sum_{i: j \in S_i} x_i \geq 1 \quad \forall j \in U
\]

\[ x_i \in \{0, 1\} \quad \forall i \]

- Cheapest **Set Cover** solution = best (ILP) solution
The LP relaxation

We relax this to a linear program

\[
\min \sum_{i=1}^{m} c(S_i)x_i \quad (LP)
\]

\[
\sum_{i: j \in S_i} x_i \geq 1 \quad \forall j \in U
\]

\[
0 \leq x_i \leq 1 \quad \forall i
\]

- \( (LP) \) can be solved in polynomial time (see next chapter)
- Let \( OPT_f \) be value of optimum solution
- Of course \( OPT_f \leq OPT \)
- Integrality gap

\[
\alpha(n) := \sup_{\text{instances } |\mathcal{I}| = n} \frac{OPT(\mathcal{I})}{OPT_f(\mathcal{I})}
\]
The algorithm

Algorithm:

1. Solve $(LP) \rightarrow x^*$ opt. fractional solution
2. (Randomized rounding:) FOR $i = 1, \ldots, m$ DO
   3. Pick $S_i$ with probability $\min\{\ln(n) \cdot x_i^*, 1\}$
4. (Repairing:) FOR every not covered element $j \in U$ pick the cheapest set containing $j$
Analysis

**Theorem**

\[ E[APX] \leq (\ln(n) + 1) \cdot OPT_f \]

Consider an element \( j \in U \):

\[
\Pr[j \text{ not covered in (2)}] = \prod_{i:j \in S_i} \Pr[S_i \text{ not picked in (2)}] \\
\leq \prod_{i:j \in S_i} (1 - \ln(n) \cdot x_i^*) \\
\leq e^{1+y} \prod_{i:j \in S_i} e^{-\ln(n) \cdot x_i^*} \\
\leq e^{-\ln(n) \cdot \sum_{i:j \in S_i} x_i^*} \\
\leq e^{-\ln(n)} = \frac{1}{n}
\]
Analysis (2)

- Cost of randomized rounding:

\[ E[\text{cost in (2)}] = \sum_{i=1}^{m} \Pr[S_i \text{ picked in (2)}] \cdot c(S_i) \leq \sum_{i=1}^{m} \ln(n)x^*_i c(S_i) = \ln(n) \cdot OPT_f \]

- Cost of repairing step: In step (3), we pick \( n \) times with prob. \( \leq \frac{1}{n} \) a set of cost \( \leq OPT_f \). Hence

\[ E[\text{cost of step (3)}] \leq n \cdot \frac{1}{n} \cdot OPT_f = OPT_f \]

- By linearity of expectation

\[ E[APX] = E[\text{cost in (2)}] + E[\text{cost in (3)}] \leq (\ln(n)+1) \cdot OPT_f \]
PART 8
INSERTION: LINEAR PROGRAMMING

SOURCE: Geometric Algorithms and Combinatorial Optimization
(Grötschel, Lovász, Schrijver)
Linear programs

Let \( A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n \) then

\[
\begin{align*}
\text{max } & \quad c^T x \\
\text{Ax} & \quad \leq \quad b \\
x_i & \quad \geq \quad 0 \quad \forall i
\end{align*}
\]

is called a linear program. Alternatively one might have

- min instead of max
- no non-negativity \( x_i \geq 0 \)
- \( Ax = b \)

More terminology

- \( \text{conv}(\{x, y\}) := \{\lambda x + (1 - \lambda)y \mid \lambda \in [0, 1]\} \)
- Set \( Q \subseteq \mathbb{R}^n \) convex if \( \forall x, y \in Q : \text{conv}(\{x, y\}) \subseteq Q \)
- A set \( P \) is called a polyhedron if \( P = \{x \in \mathbb{R}^n \mid Ax \leq b\} \)
- If \( P \) bounded (\( \exists M : P \subseteq [-M, M]^n \)) then \( P \) is a polytope.
Vertices

Let \( P = \{ x \in \mathbb{R}^n \mid Ax \leq b \} \) be a polyhedron.

**Definition**

A point \( x^* \in P \) is called a **vertex** if there is a \( c \in \mathbb{R}^n \) such that \( x^* \) is the unique optimum solution of \( \max\{ c^T x \mid x \in P \} \).

Alternative names: basic solution, extreme point.
Alternative characterisations

Lemma

Let $x^* \in P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. The following statements are equivalent:

- $x^*$ is a vertex

- There are no $y, z \in P$ with $(x^*, y, z$ pairwise different) and $x^* \in \text{conv}\{y, z\}$

- There is a linear independent subsystem $A'x \leq b'$ (with $n$ constraints) of $Ax \leq b$ s.t. $\{x^*\} = \{x \in \mathbb{R}^n \mid A'x = b'\}$. 
Not every polyhedron has vertices

**Example:** The polyhedron $P = \{ x \in \mathbb{R}^2 \mid -x_1 + x_2 \leq 1 \}$ does not have any vertices.

![Diagram of the polyhedron](image)

**Lemma**

*Any polytope has vertices.*

**Lemma**

*Any polyhedron $P \subseteq \mathbb{R}^n$ with non-negativity constraints $x_i \geq 0 \ \forall i = 1, \ldots, n$ has vertices.*
Support of vertex solutions

Lemma

Let \( x^* \) be a vertex of

\[
P = \{ x \in \mathbb{R}^n \mid a_j^T x \leq b_j \ \forall j = 1, \ldots, m; x_i \geq 0 \ \forall i \}
\]

Then \( |\{ i \mid x^*_i > 0 \}| \leq m \) (\#non-zero entries \leq \#constraints).

Proof: There is a subsystem \( I, J \) with \( |J| + |I| = n \) and
\[
\{ x^* \} = \{ x \mid a_j^T x = b_j \ \forall j \in J; x_i = 0 \ \forall i \in I \}. \text{ Hence } |I| = n - |J| \geq n - m.
\]
Linear programming is doable in polytime

**Theorem**

Given $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, there is an algorithm which solves

$$\max\{c^T x \mid Ax \leq b\}$$

in time polynomial in $n, m$ and the encoding length of $A, b, c$. The algorithm returns an optimum vertex solution if there is any.

- **Polynomial** here means that the number of bit operations is bounded by a polynomial (Turing model).
- **Encoding length** (= #bits used to encode an object) for
  - integer $\alpha \in \mathbb{Z}$: $\langle \alpha \rangle := \lceil \log_2(|\alpha| + 1) \rceil + 1$.
  - rational number $\alpha = \frac{p}{q} \in \mathbb{Q}$: $\langle \alpha \rangle := \langle p \rangle + \langle q \rangle$
  - vector $c \in \mathbb{Q}^n$: $\langle c \rangle := \sum_{i=1}^n \langle c_i \rangle$
  - inequality $a^T x \leq \delta$: $\langle a \rangle + \langle \delta \rangle$
  - matrix $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$: $\langle A \rangle := \sum_{i=1}^m \sum_{j=1}^n \langle a_{ij} \rangle$
The ellipsoid method

**Input:** Fulldimensional polytope $P \subseteq \mathbb{R}^n$

**Output:** Point in $P$

1. Find ellipsoid $E_1 \supseteq P$ with center $z_1$
2. FOR $t = 1, \ldots, \infty$ DO
   3. IF $z_t \in P$ THEN RETURN $z_t$
   4. Find hyperplane $a^T x = \delta$ through $z_t$ such that $P \subseteq \{ x \mid a^T x < \delta \}$
   5. Compute ellipsoid $E_{t+1} \supseteq E_t \cap \{ x \mid a^T x \leq \delta \}$ with $\text{vol}(E_{t+1}) = (1 - \frac{\Theta(1)}{n}) \text{vol}(E_t)$
The ellipsoid method (2)

Problem: Separation Problem for $P$:
- **Given:** $y \in \mathcal{Q}^n$
- **Find:** $a \in \mathcal{Q}^n$ with $a^Ty > a^Tx \forall x \in P$ (or assert $y \in P$).

Rule of thumb
If one can solve the Separation Problem for $P \subseteq \mathbb{R}^n$ in poly-time, then one can solve $\max\{c^Tx \mid x \in P\}$ efficiently.

**Important:** The number of inequalities does not play a role. Especially we can optimize in many cases even if the number of inequalities is exponential.
Theorem

Let $P \subseteq \mathbb{R}^n$ be a polyhedron that can be described as $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ with $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m$, and let $c \in \mathbb{Q}^n$ be an objective function. Let $\varphi$ be an upper bound on

- the encoding length of each single inequality in $Ax \leq b$.
- the dimension $n$
- the encoding length of $c$.

Suppose one can solve the following problem in time $\text{poly}(\varphi)$:

**Separation problem:** Given $y \in \mathbb{Q}^n$ with encoding length $\text{poly}(\varphi)$ as input. Decide, whether $y \in P$. If not find an $a \in \mathbb{Q}^n$ with $a^T y > a^T x \ \forall x \in P$.

Then there is an algorithm that yields in time $\text{poly}(\varphi)$ either

- $x^* \in \mathbb{Q}^n$ attaining $\max\{c^T x \mid x \in P\}$ ($x^*$ will be a vertex if $P$ has vertices)
- $P$ empty
- Vectors $x, y \in \mathbb{Q}^n$ with $x + \lambda y \in P \ \forall \lambda \geq 0$ and $c^T y \geq 1$.

Here running times are w.r.t. the Turing machine model.
Weak duality

**Observation**

Consider the LP \( \max \{ c^T x \mid x \in P \} \) with \( P = \{ x \in \mathbb{R}^n \mid Ax \leq b \} \). Let \( y \geq 0 \). Then \((y^T A)x \leq y^T b\) is a feasible inequality for \( P \) (i.e. \((y^T A)x \leq y^T b \ \forall x \in P \)). In fact, if \( y^T A = c^T \), then
\[
c^T x = (y^T A)x \leq y^T b \ \forall x \in P
\]

**Example:** \( \max \{ x_1 + x_2 \mid x_1 + 2x_2 \leq 6, \ x_1 \leq 2, \ x_1 - x_2 \leq 1 \} \)

Optimum solution: \( x^* = (2, 2) \) with \( c^T x^* = 4 \).
Weak duality (2)

Theorem (Weak duality)

Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$. Then

$$
\max \{ c^T x \mid Ax \leq b \} \leq \min \{ b^T y \mid y^T A = c^T; \ y \geq 0 \}
$$

(given that both systems are feasible.

- If $(P)$ is the primal program, then $(D)$ is the dual program to $(P)$.
- Note: The dual of the dual is the primal.
Strong duality

Theorem (Strong duality I)

Let $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^{m}, c \in \mathbb{R}^{n}$. Then

$$\max\{c^T x \mid Ax \leq b\} = \min\{b^T y \mid y^T A = c^T; \ y \geq 0\}$$

given that both systems are feasible.

Theorem (Strong duality II)

Let $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^{m}, c \in \mathbb{R}^{n}$. Then

$$\max\{c^T x \mid Ax \leq b, x \geq 0\} = \min\{b^T y \mid y^T A \geq c^T, y \geq 0\}$$

given that both systems are feasible.
Hand-waving proof of strong duality

Claim

Let $x^*$ be optimum solution of $\max\{c^T x \mid Ax \leq b\}$. Then there is a $y \geq 0$ with $y^T A = c^T$ and $y^T b = c^T x^*$.

- Let $a_1, \ldots, a_m$ be rows of $A$.
- Let $I := \{i \mid a_i^T x^* = b_i\}$ be the tight inequalities.
- Suppose for contradiction $c \notin \{\sum_i a_i y_i \mid y_i \geq 0, i \in I\} =: C$
- Then there is a $\lambda \in \mathbb{R}^n$ with $c^T \lambda > 0$, $a_i^T \lambda \leq 0 \forall i \in I$.
- Walking in direction $\lambda$ improves objective function. But $x^*$ was optimal. **Contradiction!**
Hand-waving proof of strong duality

Claim

Let $x^*$ be optimum solution of $\max \{ c^T x \mid Ax \leq b \}$. Then there is a $y \geq 0$ with $y^T A = c^T$ and $y^T b = c^T x^*$.

- Let $a_1, \ldots, a_m$ be rows of $A$.
- Let $I := \{ i \mid a_i^T x^* = b_i \}$ be the tight inequalities.

- $\exists y \geq 0 : y^T A = c^T$ and $y_i = 0 \ \forall i \notin I$ (we only use tight inequalities)

$$y^T b - c^T x^* = y^T b - y^T A x^* = y^T (b - A x^*) = \sum_{i=1}^{m} y_i \cdot \begin{cases} b_i - a_i^T x^* & \text{if } i \notin I \\ 0 & \text{if } i \in I \end{cases} = 0$$
Complementary Slackness

**Warning:** Primal and dual are switched here.

**Theorem (Complementary slackness)**

Let $x^*$ be a solution for

\[(P) : \min \{ c^T x \mid Ax \geq b, x \geq 0 \} \]

and $y^*$ a solution for

\[(D) : \max \{ b^T y \mid A^T y \leq c, y \geq 0 \}. \]

Let $a_i$ be the $i$th row of $A$ and $a^j$ be its $j$th column. Then $x^*$ and $y^*$ are both optimal $\iff$ both following conditions are true

- **Primal complementary slackness:** $x_j > 0 \Rightarrow (a^j)^T y = c_j$
- **Dual complementary slackness:** $y_i > 0 \Rightarrow a_i^T x = b_i$
Part 9

Weighted Vertex Cover

Source: Approximation Algorithms (Vazirani, Springer Press)
**Vertex Cover**

**Problem:** Weighted Vertex Cover

- **Given:** Undirected graph $G = (V, E)$, node weights $c : V \rightarrow \mathbb{Q}_+$
- **Find:** Subset $U \subseteq V$ such that every edge is incident to at least one node in $U$ and $\sum_{v \in U} c(v)$ is minimized.

Consider the LP

$$
\min \sum_{v \in V} c(v)x_v
$$

$$
x_u + x_v \geq 1 \quad \forall (u, v) \in E
$$

$$
x_v \geq 0 \quad \forall v \in V
$$
Half-integrality

Lemma

Let \( x^* \) be a basic solution of \((LP)\). Then \( x^*_v \in \{0, \frac{1}{2}, 1\} \) for all \( v \in V \), i.e. \( x^* \) is half-integral.

- Suppose \( x^* \) is not half-integral, i.e. not both sets are empty:

  \[
  V_+ := \{ v \mid \frac{1}{2} < x^*_v < 1 \}, \quad V_- := \{ v \mid 0 < x^*_v < \frac{1}{2} \}
  \]

- It suffices to show that \( x^* \) can be written as convex combination \( x^* = \frac{1}{2}y + \frac{1}{2}z \) for 2 different feasible \((LP)\) solutions \( y, z \).

\[
\begin{align*}
V_- &\ni v_1 & v_2 &\in V_+ \\
 x^*_v &\in [0.3, 0.7]
\end{align*}
\]
Half-integrality (2)

- Define

\[
y_v := \begin{cases} 
  x_v^* + \varepsilon & x_v^* \in V_+ \\
  x_v^* - \varepsilon & x_v^* \in V_- \\
  x_v^* & \text{otherwise}
\end{cases}
\quad \text{and} \quad
z_v := \begin{cases} 
  x_v^* - \varepsilon & x_v^* \in V_+ \\
  x_v^* + \varepsilon & x_v^* \in V_- \\
  x_v^* & \text{otherwise}
\end{cases}
\]

- Tight edges \((u, v) \in E : x_v^* + x_u^* = 1\) drawn solid

- Constraints satisfied by \(y, z\) for \(\varepsilon > 0\) small enough.
The Algorithm

Algorithm:

1. Compute an optimum basic solution \( x^* \) to \((LP)\)
2. Choose vertex cover \( U := \{v \mid x^*_v > 0\} \)

Theorem

\( U \) is a vertex cover of cost \( \leq 2 \cdot OPT_f \).

Proof.

Clearly \( U \) is feasible. Furthermore

\[
\sum_{v \in U} c(v) = \sum_{v \in V} \lfloor x^*_v \rfloor c(v) \leq 2 \sum_{v \in V} x^*_v c(v) = 2 \cdot OPT_f.
\]
Inapproximability

Theorem (Khot & Regev ’03)

There is no polynomial time \((2 - \varepsilon)\)-apx unless Unique Games Conjecture is false.

Unique Games Conjecture

For all \(\varepsilon > 0\), there is a prime \(p := p(\varepsilon)\) such that the following problem is \(\text{NP}\)-hard:

- **Given:** Equations \(x_i \equiv_p a_{ij}x_j\) for some \((i, j)\) pairs
- **Distinguish:**
  - Yes: max satisfiable fraction \(\geq 1 - \varepsilon\)
  - No: max satisfiable fraction \(\leq \varepsilon\)

Example:

\[
x_1 \equiv_{13} 4 \cdot x_3 \\
x_2 \equiv_{13} 9 \cdot x_1 \\
\ldots
\]
PART 10

INSERTION: ALGORITHMIC PROBABILITY THEORY

Probability theory

Definition

A (discrete) probability space consists of
  ▶ A (countable) sample space \( \Omega \) modelling all possible outcomes of a random process.
  ▶ A probability function \( \Pr : 2^\Omega \rightarrow \mathbb{R} \) such that
    (a) \( 0 \leq \Pr[E] \leq 1 \ \forall E \subseteq \Omega \)
    (b) \( \Pr[\Omega] = 1 \)
    (c) For any (countable) sequence of pairwise disjoint events \( E_1, E_2, \ldots \subseteq \Omega \)

\[
\Pr \left[ \bigcup_{i \geq 1} E_i \right] = \sum_{i \geq 1} \Pr[E_i]
\]

Definition (Random variable)

A function \( X : \Omega \rightarrow \mathbb{R} \) is called a random variable.
Probability theory (2)

Definition (Expectation)
Let \( X : \Omega \rightarrow \mathbb{R} \) be a random variable. Then

\[
E[X] = \sum_{i} i \cdot \text{Pr}[X = i]
\]

Lemma (Linearity of expectation)
Let \( X_1, \ldots, X_n : \Omega \rightarrow \mathbb{R} \) random variables with finite expectations. Then

\[
E \left[ \sum_{i=1}^{n} X_i \right] = \sum_{i=1}^{n} E[X_i]
\]
Probability theory (3)

Lemma (Independence)

Random variables $X_1, \ldots, X_n$ are called independent if

$$\forall I \subseteq \{1, \ldots, n\} : \forall x_i : \Pr \left[ \bigcap_{i \in I} (X_i = x_i) \right] = \prod_{i \in I} \Pr[X_i = x_i]$$

Lemma

Let $X_1, \ldots, X_n$ independent random variables. Then

$$E \left[ \prod_{i=1}^{n} X_i \right] = \prod_{i=1}^{n} E[X_i]$$
Lemma (Union bound)

Let $E_1, \ldots, E_n \subseteq \Omega$ be events

\[
\Pr \left[ \bigcup_{i=1}^{n} E_i \right] \leq \sum_{i=1}^{n} \Pr[E_i]
\]
Probability theory (5)

Lemma (Markov bound)

Let $X \geq 0$ be a random variable. Then

$$\Pr[X \geq a] \leq \frac{E[X]}{a}$$

Proof.

The value $E[X]$ is

$$E[X] = \underbrace{E[X \mid X \geq a]}_{\geq a} \cdot \Pr[X \geq a] + \underbrace{E[X \mid X < a]}_{\geq 0} \cdot \Pr[X < a]$$

$$\geq a \cdot \Pr[X \geq a]$$
Probability theory (6)

**Theorem (Chernov bound)**

Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in \{0, 1\}$ and $X := X_1 + \ldots + X_n$. For any $\delta > 0$ one has

$$
\Pr[X \geq (1 + \delta)E[X]] \leq \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^{E[X]}
$$
Let $t := \ln(1 + \delta) > 0$, $p_i := \Pr[X_i = 1]$. Note that $E[X_i] = p_i$.

$$\Pr[X \geq (1 + \delta)E[X]] \overset{e^{tx}\text{ mon.inc.}}{=} \Pr[e^{tX} \geq e^{t(1+\delta)E[X]}]$$

Markov

$$\leq \frac{E[e^{tX}]}{e^{t(1+\delta)E[X]}}$$

$$\leq \frac{E[\prod_{i=1}^n e^{tX_i}]}{e^{t(1+\delta)E[X]}}$$

$X_1, \ldots, X_n$ indep

$$\overset{=} {=} \frac{\prod_{i=1}^n E[e^{tX_i}]}{e^{t(1+\delta)E[X]}}$$

$\overset{(*)}{\leq} \frac{\prod_{i=1}^n e^{\delta p_i}}{e^{t(1+\delta)E[X]}}$

$$= \frac{e^{\delta \sum_{i=1}^n p_i}}{e^{t(1+\delta)E[X]}}$$

$$E[X] = \sum_{i=1}^n p_i \left( \frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right)^{E[X]}$$

$\overset{(*)}{\leq} E[e^{tX_i}] = p_i \cdot e^{t \cdot \frac{1}{1+\delta}} + (1 - p_i) \cdot e^{t \cdot \frac{0}{1}} = 1 + \delta p_i \leq e^{\delta p_i}$
Probability theory (7)

Theorem (Variants of Chernov bound)

Let $X_1, \ldots, X_n \in \{0, 1\}$ be independent random variables with and $X := X_1 + \ldots + X_n$ and $0 < \delta \leq 1$. Then

- Let $\mu \geq E[X]$, then
  \[ \Pr[X \geq (1 + \delta)\mu] \leq e^{-\mu \cdot \delta^2 / 2} \]

- Let $\mu \leq E[X]$, then
  \[ \Pr[X \leq (1 - \delta)\mu] \leq e^{-\mu \cdot \delta^2 / 2} \]
PART 11
MINIMIZING CONGESTION

SOURCE: Randomized rounding: A technique for provably good algorithms and algorithmic proofs (Raghavan, Tompson)
http://www.springerlink.com/content/n16347864k45367w/fulltext.pdf
Minimizing Congestion

**Problem:** \textsc{MinCongestion}

- **Given:** Directed graph $G = (V, E)$ with demand pairs $(s_i, t_i)$ \( s_i, t_i \in V, i = 1, \ldots, k \)
- **Find:** $s_i$-$t_i$ paths $P_i$ that minimize the congestion

\[
\max_{e \in E} \left| \{i : e \in P_i\} \right|
\]
Minimizing Congestion

**Problem:** \textbf{MinCongestion}

- **Given:** Directed graph $G = (V, E)$ with demand pairs $(s_i, t_i)$ $s_i, t_i \in V$, $i = 1, \ldots, k$
- **Find:** $s_i$-$t_i$ paths $P_i$ that minimize the congestion

$$\max_{e \in E} |\{i : e \in P_i\}|$$

![Diagram](image)
A flow-based LP formulation of MinCongestion

\[ \min C \quad \text{(LP)} \]

\[ \sum_{e \in \delta^+(v)} f_i(e) - \sum_{e \in \delta^-(v)} f_i(e) = \begin{cases} 
1 & v = s_i \\
-1 & v = t_i \\
0 & \text{otherwise}
\end{cases} \]

\[ \sum_{i=1}^{k} f_i(e) \leq C \quad \forall e \in E \]

\[ C \geq 1 \]

\[ f_i(e) \geq 0 \quad \forall i \forall e \in E \]

\[ f_1(e) = \frac{1}{2} \text{ on red } e \]

\[ f_2(e) = \frac{1}{2} \text{ on blue } e \]

\[ f_3(e) = \frac{1}{2} \text{ on green } e \]
Path Decomposition

- **Input:** \( s-t \) flow \( f : E \to \mathbb{Q}_+ \) (without directed cycles)
- **Output:** Paths \( p_1, \ldots, p_m \) with values \( v_1, \ldots, v_m \geq 0 \)

(1) \( i := 1 \)

(2) **WHILE** \( f \neq 0 \) **DO**

(3) Let \( p_i \) be any \( s-t \) path in \( \{ e \mid f(e) > 0 \} \)

(4) \( v_i := \min\{ f(e) \mid e \in p_i \} \)

(5) \( f(e) := f(e) - v_i \ \forall e \in p_i \)

(6) \( i := i + 1 \)
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \rightarrow \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. **WHILE** $f \neq 0$ **DO**
   3. Let $p_i$ be any $s$-$t$ path in $\{e | f(e) > 0\}$
   4. $v_i := \min\{f(e) | e \in p_i\}$
   5. $f(e) := f(e) - v_i \ \forall e \in p_i$
   6. $i := i + 1$
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \rightarrow \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. WHILE $f \neq 0$ DO
   3. Let $p_i$ be any $s$-$t$ path in $\{ e \mid f(e) > 0 \}$
   4. $v_i := \min \{ f(e) \mid e \in p_i \}$
   5. $f(e) := f(e) - v_i \ \forall e \in p_i$
   6. $i := i + 1$
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \rightarrow \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. WHILE $f \neq \mathbf{0}$ DO
3. Let $p_i$ be any $s$-$t$ path in \{e | f(e) > 0\}
4. $v_i := \min\{f(e) | e \in p_i\}$
5. $f(e) := f(e) - v_i \ \forall e \in p_i$
6. $i := i + 1$
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \rightarrow \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. **WHILE** $f \neq 0$ **DO**
   3. Let $p_i$ be any $s$-$t$ path in $\{e \mid f(e) > 0\}$
   4. $v_i := \min\{f(e) \mid e \in p_i\}$
   5. $f(e) := f(e) - v_i \ \forall e \in p_i$
   6. $i := i + 1$
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \to \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. WHILE $f \neq 0$ DO
   3. Let $p_i$ be any $s$-$t$ path in $\{e | f(e) > 0\}$
   4. $v_i := \min\{f(e) | e \in p_i\}$
   5. $f(e) := f(e) - v_i \quad \forall e \in p_i$
   6. $i := i + 1$
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \to \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. WHILE $f \neq 0$ DO
3. Let $p_i$ be any $s$-$t$ path in $\{e \mid f(e) > 0\}$
4. $v_i := \min\{f(e) \mid e \in p_i\}$
5. $f(e) := f(e) - v_i \ \forall e \in p_i$
6. $i := i + 1$
Path Decomposition

- **Input:** $s$-$t$ flow $f : E \rightarrow \mathbb{Q}_+$ (without directed cycles)
- **Output:** Paths $p_1, \ldots, p_m$ with values $v_1, \ldots, v_m \geq 0$

1. $i := 1$
2. **WHILE** $f \neq 0$ **DO**
   3. Let $p_i$ be any $s$-$t$ path in $\{e \mid f(e) > 0\}$
   4. $v_i := \min\{f(e) \mid e \in p_i\}$
   5. $f(e) := f(e) - v_i \ \forall e \in p_i$
   6. $i := i + 1$
Path Decomposition

Lemma

The algorithm decomposes the flow in s-t paths $p_1, \ldots, p_m$ with $m \leq |E|$.

$$
\sum_{e \in \delta^+(s)} f(e) = \sum_{i=1}^m v_i \quad \text{and} \quad \sum_{i: e \in p_i} v_i = f(e) \quad \forall e \in E
$$

- $f$ remains a flow throughout the algorithm.
- In each iteration there is an edge, where the flow drops down to 0.
An approximation algorithm for MinCONGESTION

Algorithm

1. Solve $(LP) \rightarrow \text{flows } f_1, \ldots, f_k \text{ frac. congestion } OPT_f$
2. FOR $i = 1, \ldots, k$ DO
   3. apply path decomposition to $f_i \rightarrow (p^i_j, v^i_j)$ ($\sum_j v^i_j = 1 \forall i$)
4. Choose $P_i$ among $p^i_j$’s with $\Pr[P_i = p^i_j] = v^i_j$

Theorem

With probability $\geq 1 - \frac{1}{n} \text{ the congestion is } \leq O\left(\frac{\ln n}{\ln \ln n}\right) \cdot OPT_f.$

- Consider any edge $e \in E$.
- Let $X^e_i \in \{0, 1\}$ be the random variable, saying whether the $s_i-t_i$ path uses $e$. $X^e_1, \ldots, X^e_k$ are independent!
- Let $X^e := \sum_{i=1}^k X^e_i$ be the number of paths, crossing $e$.
- $E[X^e] = \sum_{i=1}^k \Pr[X^e_i] = \sum_{i=1}^k f_i(e) \leq OPT_f$. 

\[= f_i(e) \]
Proof (2)

\[
\Pr \left[ X^e > \left( \frac{\log n}{\log \log n} + 1 \right) \frac{\geq E[X^e]}{\text{OPT}_f} \right] \leq \left( \frac{e^\delta}{\delta \delta} \right)^{\geq 1 \text{OPT}_f} \\
\leq \left( \frac{e}{\ln n} \right)^{\frac{\ln n}{\ln \ln n}} \\
\leq \left( \frac{\ln \ln n}{\ln n} \right)^{\frac{\ln n}{\ln \ln n}} \\
= \left( \exp \left( \ln \ln \ln n - \ln \ln n \right) \right)^{\frac{\ln n}{\ln \ln n}} \\
\leq \exp \left( - \frac{1}{2} \ln \ln n \cdot \frac{\ln n}{\ln \ln n} \right) \\
= \frac{1}{n^{c/2}}
\]

\[
\Pr \left[ \bigvee_{e \in E} \left( X^e > 6 \frac{\ln n}{\ln \ln n} \text{OPT}_f \right) \right] \leq |E| \cdot \frac{1}{n^3} \leq \frac{1}{n} \quad \square
\]
Inapproximability

**Theorem (Andrews & Zhang - JACM’08)**

There is no $\log^{1-\varepsilon} n$-apx unless $\text{NP} \subseteq \text{ZPTIME}(n^{\text{polylog}(n)})$. 
PART 12
KnapSack

Source: Approximation Algorithms (Vazirani, Springer Press)
Knapsack

Problem: **Knapsack**

- **Given:** \( n \) objects with weight \( w_i \in \mathbb{Q}_+ \) and profit \( p_i \in \mathbb{Q}_+ \), size \( G \in \mathbb{Q}_+ \)

- **Find:** Subset of objects, maximizing the profit and not exceeding the weight bound:

\[
OPT = \max_{I \subseteq \{1, \ldots, n\}} \left\{ \sum_{i \in I} p_i \mid \sum_{i \in I} w_i \leq G \right\}
\]
A dynamic program for **KNAPSACK**

**Dynamic program:**

1. Assume restricted profits \( p_i \in \{0, \ldots, B\} \)
2. Compute table entries

\[
T(i, b) = \min_{I \subseteq \{1, \ldots, i\}} \left\{ \sum_{j \in I} w_j \mid \sum_{j \in I} p_j \geq b \right\}
\]

\[
= \text{minimum weight needed for a subset of the first } i \text{ objects to obtain a profit of at least } b
\]

using dynamic programming

\[
T(i, b) = \min \left\{ T(i - 1, b), T(i - 1, b - p_i) + w_i \right\} \quad \forall i \quad \forall p = 0, \ldots, B
\]

\( \text{don’t take } i \quad \text{take } i \)

3. Reconstruct \( I \) leading to \( \max \{b \in \mathbb{N}_0 \mid T(n, b) \leq G\} \)

**Observation**

The algorithm finds optimum solutions in time \( O(n \cdot B) \).
The FPTAS

Algorithm:

1. Scale profits s.t. $p_{\text{max}} = n/\varepsilon$
2. Round $p'_i := \lfloor p_i \rfloor$
3. Compute and return optimum solution $I$ for weights $p'_i$
Analysis of FPTAS

Theorem

Let $0 < \varepsilon \leq \frac{1}{2}$. The algo gives a $(1 + 2\varepsilon)$-apx in time $O(n^2/\varepsilon)$.

- W.l.o.g. $OPT \geq p_{\text{max}} = n/\varepsilon$ (we can delete objects that even alone do not fit into the knapsack)
- Let $I^*$ be optimum solution for original profits. Let $OPT'$ be optimum value for profits $p'$. Then

$$OPT' \geq \sum_{i \in I^*} p'_i = \sum_{i \in I^*} |p_i| \geq \sum_{i \in I^*} p_i - |I^*| \geq OPT - n$$

$$\geq (1 - \varepsilon)OPT \geq \frac{OPT}{1 + 2\varepsilon}$$

- Let $I$ be solution found by dynamic program:

$$\sum_{i \in I} p_i \geq \sum_{i \in I} p'_i = OPT' \geq \frac{OPT}{1 + 2\varepsilon}$$

- $B = \max\{p'_i\} \leq n/\varepsilon$ hence the running time is $O(n^2/\varepsilon)$
Part 13
Multi Constraint Knapsack

Source: Folklore
Multi Constraint Knapsack

Problem: Multi Constraint Knapsack (MCK)

- **Given:** \( n \) objects with profits \( p_i \in \mathbb{Q}_+ \) and \( k \) many budgets \( B_j \). Object \( i \) has requirement \( a_{ij} \in \mathbb{Q}_+ \) w.r.t. budget \( j \).
- **Find:** Subset of objects, maximizing the profit and not exceeding any budget:

\[
OPT = \max_{I \subseteq \{1, \ldots, n\}} \left\{ \sum_{i \in I} p_i \mid \sum_{i \in I} a_{ij} \leq B_j \forall j = 1, \ldots, k \right\}
\]

- For arbitrary \( k \) there is no \( n^{1-\varepsilon} \)-apx: Take an Independent Set instance \( G = (V, E) \). For each edge \( e = (u, v) \) add an “edge budget constraint” \( a_u^{e} = a_v^{e} = 1, B_e = 1 \). Then \( OPT = OPT_{IS} \).

\[
\begin{align*}
&\max x_1 + x_2 + x_3 \\
&1x_1 + 1x_2 + 0x_3 \leq 1 \\
&0x_1 + 1x_2 + 1x_3 \leq 1 \\
x_i &\in \{0, 1\}
\end{align*}
\]
A PTAS for $k = O(1)$

Algorithm:

1. Guess the $\left\lceil \frac{k}{\epsilon} \right\rceil$ items $I_{\text{large}}$ in the optimum solution with maximum profit

2. Let $x^*$ be optimum basic solution to the following LP

$$\max \sum_{i=1}^{n} x_i p_i$$

$$\sum_{i=1}^{n} a_i^j x_i \leq B_j \quad \forall j = 1, \ldots, k$$

$$x_i = 1 \quad \forall i \in I_{\text{large}}$$

$$x_i = 0 \quad \forall i \notin I_{\text{large}} : p_i > \min \{ p_j \mid j \in I_{\text{large}} \}$$

$$0 \leq x_i \leq 1 \quad \forall i = 1, \ldots, n$$

3. Output $I := \{ i \mid x_i^* = 1 \}$. 
The Analysis

**Theorem**

*For constant $k$ the algorithm has polynomial running time. Furthermore $APX \geq (1 - \varepsilon)OPT$.***

- The produced solution is clearly feasible
- $LP \geq OPT$ (since we guess elements from $OPT$)
- Observation: $|\{i \mid 0 < x_i^* < 1\}| \leq k$ since $x^*$ is a basic solution and apart from $0 \leq \ldots \leq 1$ there are only $k$ constraints.
- For $i$ with $0 < x_i^* < 1$ one has $p_i \leq \frac{\varepsilon}{k}OPT$

\[
APX \geq \sum_{i=1}^{n} |x_i^*| p_i \geq LP - \sum_{i:0 < x_i^* < 1} p_i \leq k \cdot \frac{\varepsilon}{k}OPT \\
\geq OPT - k \cdot \frac{\varepsilon}{k}OPT = (1 - \varepsilon)OPT
\]
Hardness of $\text{MultiConstraintKnapsack}$

**Theorem**

There is no FPTAS for $\text{MultiConstraintKnapsack}$ even for 2 budgets, unless $\text{NP} = \text{P}$.

**Problem: ** $\text{Partition}$

- **Given:** Numbers $a_1, \ldots, a_n \in \mathbb{N}$, $S := \sum_{i=1}^{n} a_i$, $m \in \{1, \ldots, n\}$
- **Find:** $I \subseteq \{1, \ldots, n\}$ : $|I| = m$, $\sum_{i \in I} a_i = S/2$

- Recall: $\text{Partition}$ is $\text{NP}$-hard.

- Define MCK instance with 2 constraints:

\[
\begin{align*}
\max & \sum_{i=1}^{n} x_i \\
\sum_{i=1}^{n} x_i a_i & \leq S/2 \\
\sum_{i=1}^{n} x_i (S - a_i) & \leq S(m - \frac{1}{2}) \\
x_i & \in \{0, 1\} \quad \forall i = 1, \ldots, n
\end{align*}
\]
Proof

- **Claim:** \( \exists \) PARTITION solution \( \iff \text{OPT}_{\text{MCK}} \geq m \)

\( \Rightarrow \) Suppose \( \exists I : |I| = m, \sum_{i \in I} a_i = S/2 \). Then this is a MCK solution of value \( m \) since

\[
\sum_{i \in I} (S - a_i) = mS - \sum_{i \in I} a_i = S(m - \frac{1}{2})
\]

\( \Leftarrow \) Let \( I \) be MCK solution of value \( \geq m \).

\[
|I| \cdot S - \frac{S}{2} \overset{1.\text{ constr.}}{\leq} |I| \cdot S - \sum_{i \in I} a_i = \sum_{i \in I} (S - a_i) \overset{2.\text{ const.}}{\leq} m \cdot S - \frac{S}{2}
\]

\[
\leq \frac{S}{2}
\]

- Hence \( |I| = m \). Then ineq. holds with "="
- Thus \( \sum_{i \in I} a_i = S/2 \).

\[ \square \]

- Now suppose for contradiction we would have an FPTAS for MCK: Then choose \( \varepsilon := \frac{1}{n+1} \). Then the FPTAS would give an optimum solution for the instance resulting from the PARTITION reduction.
Part 14
Bin Packing

Source: Combinatorial Optimization: Theory and Algorithms
(Korte, Vygen)
Bin Packing

**Problem:** \textsc{BinPacking}

- **Given:** Items with sizes $a_1, \ldots, a_n \in [0, 1]$
- **Find:** Assign items to minimum number of bins of size 1.

\[
OPT = \min \left\{ k \mid \exists I_1 \cup \ldots \cup I_k = \{1, \ldots, n\} : \forall j : \sum_{i \in I_j} a_i \leq 1 \right\}
\]

- Define $size(I) = \sum_{i \in I} a_i$
First Fit

First Fit algorithm:
(1) Start with empty bins
(2) FOR $i = 1, \ldots, n$ DO
    (3) Assign item $i$ to the bin $B$ with least index such that
        $a_i + \sum_{j \in B} a_j \leq 1$

Lemma

Let $m$ be the number of used bins. Then
$m \leq 2 \sum_{i=1}^{n} a_i + 1 \leq 2 \cdot OPT + 1.$

- All but $m - 1$ bins must be filled with $\geq \frac{1}{2}$ (otherwise we would not have opened a new bin):
  $\sum_{i=1}^{n} a_i \geq \frac{1}{2}(m - 1)$
- Hence $m \leq 2 \sum_{i=1}^{n} a_i + 1.$
Linear Grouping

- **INPUT**: Instance $I = (a_1, \ldots, a_n)$, $k \in \mathbb{N}$
- **OUTPUT**: Instance $I' = (a'_1, \ldots, a'_n)$ with $a'_i \geq a_i$ and $\leq k$ different item sizes

1. Sort $a_1 \leq a_2 \leq \ldots \leq a_n$
2. Partition items into $k$ consecutive groups of $\lceil n/k \rceil$ items (the last group might have less items)
3. Let $a'_i$ be the size of the largest item in $i$’s group

![Diagram showing the grouping process with arrows connecting items from $I$ to $I'$, illustrating the process of sorting and partitioning.](image)
Linear Grouping (2)

Lemma

\[ OPT(I') \leq OPT(I) + \lceil n/k \rceil. \]

- Consider solution \( OPT(I) \). Assign item \( a'_i \) of group \( j \) to a space for item in group \( j + 1 \)
- Assign largest \( \lceil n/k \rceil \) items to their own bin
An asymptotic PTAS

Algorithm of Fernandez de la Vega & Lueker:
(1) Let \( I = \{i \mid a_i > \varepsilon\} \) be set of large items (other items are small)
(2) Apply linear grouping with \( k = 1/\varepsilon^2 \) groups to \( I \rightarrow I' \)
(3) Compute an optimum distribution of \( I' \)
(4) Distribute the small items over the used bins using First Fit

Lemma

The algorithm runs in polynomial time and uses at most \( (1 + 2\varepsilon)\text{OPT} + 1 \) bins.

- Let \( b_1, \ldots, b_{1/\varepsilon^2} \) different item sizes in \( I' \).
- Possible bin configurations
  \[ \mathcal{P} = \{p \in \{0, \ldots, 1/\varepsilon\}^{1/\varepsilon^2} \mid b^T p \leq 1\}. \quad |\mathcal{P}| \leq (1/\varepsilon^2)^{1/\varepsilon}. \]
- Solution is described by \((n_p)_{p \in \mathcal{P}}\) \((n_p = \text{how many times shall I pack a bin with configuration } p?)\), \( n_p \in \{0, \ldots, n\} \)
- \( \leq n^{(1/\varepsilon^2)^{1/\varepsilon}} \) possibilities for \((n_p)_{p \in \mathcal{P}}\).
An asymptotic PTAS (2)

- We need $OPT(I') + \# \text{ of bins additionally opened for the small items}$
- Note that

$$OPT(I') \leq OPT(I) + \lfloor |I| \cdot \varepsilon^2 \rfloor \leq OPT(I) + \lfloor \varepsilon \cdot OPT(I) \rfloor = (1+2\varepsilon) \cdot OPT$$

using $OPT(I) \geq \sum_{i \in I} a_i \geq \varepsilon \cdot |I|$ and $OPT \geq OPT(I)$.

- Suppose we need to open an additional bin for small items. Let $m$ be total number of used bins. Then all but one bin are filled to $\geq 1 - \varepsilon$. Hence

$$OPT \geq \sum_{i=1}^{m} a_i \geq (1 - \varepsilon) \cdot (m - 1)$$

and

$$m \leq \frac{OPT}{1 - \varepsilon} + 1 \leq (1 + 2\varepsilon)OPT + 1$$
Section 14.1

The algorithm of Karmarkar & Karp
The Algorithm of Karmarkar & Karp

Theorem (Karmarkar, Karp ’82)

One can compute a BinPacking solution with \( OPT + O(\log^2 n) \) many bins in polynomial time.

- Assume \( a_i \geq \delta := \frac{1}{n} \) (again one can distribute items that are smaller than \( \frac{1}{n} \) after distributing the large items.)
The Gilmore-Gomory LP-relaxation

- Let $b_i \in \mathbb{N}$ now the number of items of size $a_i$
- $n =$ number of different item sizes
- $m := \sum_{i=1}^{n} b_i =$ total number of items
- $\mathcal{P} = \{p \in \mathbb{Z}_+^n \mid a^T p \leq 1\}$ set of feasible patterns
- Variable $x_p =$ # of bins packed with pattern $p$

\begin{align*}
\text{Primal} & & \text{Dual} \\
\min 1^T x & (P(\mathcal{P})) & \max y^T b & (D(\mathcal{P})) \\
\sum_{p \in \mathcal{P}} x_p p & \geq b & p^T y & \leq 1 \quad \forall p \in \mathcal{P} \\
x & \geq 0 & y & \geq 0
\end{align*}

- # var. exponential
- # constr. polynomial
- # var. polynomial
- # constr. exponential

**Idea:** Solve the dual with Ellipsoid!
Example

- Item sizes \(a_1 = 0.3, a_2 = 0.4\)
- \# of items \(b_1 = 31, b_2 = 7\)
- Set of patterns \(\mathcal{P} = \{(0_1), (0_2), (1_1), (2_1), (1_0), (2_0), (3_0)\}\)

**Primal**

\[
\begin{align*}
\min & \; 1^T x \\
\begin{pmatrix} 0 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} x & \geq \begin{pmatrix} 31 \\ 7 \end{pmatrix} \\
x & \geq 0
\end{align*}
\]

- Opt basic solution is \(x = (0, 0, 0, 7, 0, 0, \frac{17}{3})\)

**Dual**

\[
\begin{align*}
\max & \; 31 y_1 + 7 y_2 \\
\begin{pmatrix} 0 & 1 \\ 0 & 2 \\ 1 & 1 \\ 2 & 1 \\ 0 & 1 \\ 2 & 0 \\ 3 & 0 \end{pmatrix} y & \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\
y & \geq 0
\end{align*}
\]
Weak Separation Problem

\(\varepsilon\)-Weak Separation Oracle for \(P \subseteq \mathbb{R}^n\), obj.fct. \(c \in \mathbb{Q}^n\)

**Input:** Vector \(z \in \mathbb{Q}^n\)

**Output:** One of the following

- *Case (A):* Vector \(a\) with \(a^Tx \leq a^Tz\ \forall x \in P\)
- *Case (B):* Point \(y \in P\) with \(c^Ty \geq c^Tz - \frac{\varepsilon}{2}\)

**Case (A):**

- If \(z \in P\), just return \(z\) (→ case (B)).
Grötschel-Lovász-Schrijver Algorithm

- **INPUT:** $c \in \mathbb{Q}^n, x_0 \in \mathbb{Q}^n, \varepsilon, r, R \in \mathbb{Q}_+$
  
  $B(x_0, r) \subseteq P \subseteq B(x_0, R)$

- **OUTPUT:** $y^* \in P$ with $c^T y^* \geq OPT_f - \varepsilon$

1. Ellipsod $E_0 := B(x_0, R)$ with center $z_0 := x_0$, $y^* := x_0$
2. FOR $t = 0, \ldots, \text{poly}$ DO
   
   4. Submit $z_t$ to $\varepsilon$-weak separation oracle
   5. Case (A) $\rightarrow a$: Compute $E_{t+1} \supseteq E_t \cap \{x | a^T x \leq a^T z_t\}$
   6. Case (B) $\rightarrow y \in P$:
      
      7. IF $c^T y > c^T y^*$ THEN $y^* := y$
   8. Compute $E_{t+1} \supseteq E_t \cap \{x | c^T x \geq c^T z_t\}$

**Input**

**Output**:

![Diagram](image-url)
Grötschel-Lovász-Schrijver Algorithm

- **INPUT**: $c \in \mathbb{Q}^n$, $x_0 \in \mathbb{Q}^n$, $\varepsilon, r, R \in \mathbb{Q}_+$:
  $B(x_0, r) \subseteq P \subseteq B(x_0, R)$

- **OUTPUT**: $y^* \in P$ with $c^T y^* \geq OPT_f - \varepsilon$

1. Ellipsoïd $E_0 := B(x_0, R)$ with center $z_0 := x_0$, $y^* := x_0$
2. FOR $t = 0, \ldots, poly$ DO
3. (4) Submit $z_t$ to $\varepsilon$-weak separation oracle
4. (5) Case (A) $\rightarrow a$: Compute $E_{t+1} \supseteq E_t \cap \{x \mid a^T x \leq a^T z_t\}$
5. (6) Case (B) $\rightarrow y \in P$:
6. (7) IF $c^T y > c^T y^*$ THEN $y^* := y$
7. (8) Compute $E_{t+1} \supseteq E_t \cap \{x \mid c^T x \geq c^T z_t\}$

**Case (A):**
Grötschel-Lovász-Schrijver Algorithm

- **Input:** $c \in \mathbb{Q}^n$, $x_0 \in \mathbb{Q}^n$, $\varepsilon, r, R \in \mathbb{Q}_+$:
  
  \[ B(x_0, r) \subseteq P \subseteq B(x_0, R) \]

- **Output:** $y^* \in P$ with $c^T y^* \geq OPT_f - \varepsilon$

1. Ellipsoid $E_0 := B(x_0, R)$ with center $z_0 := x_0$, $y^* := x_0$
2. FOR $t = 0, \ldots, \text{poly}$ DO
   
   4. Submit $z_t$ to $\varepsilon$-weak separation oracle
   5. **Case (A) → a:** Compute $E_{t+1} \supseteq E_t \cap \{x \mid a^T x \leq a^T z_t\}$
   6. **Case (B) → y ∈ P:**
      
      7. IF $c^T y > c^T y^*$ THEN $y^* := y$
      
      8. Compute $E_{t+1} \supseteq E_t \cap \{x \mid c^T x \geq c^T z_t\}$

**Case (A):**
Grötschel-Lovász-Schrijver Algorithm

- **INPUT:** \( c \in \mathbb{Q}^n, x_0 \in \mathbb{Q}^n, \varepsilon, r, R \in \mathbb{Q}_+ : \)
  \( B(x_0, r) \subseteq P \subseteq B(x_0, R) \)
- **OUTPUT:** \( y^* \in P \) with \( c^T y^* \geq OPT_f - \varepsilon \)

(1) Ellipsoid \( E_0 := B(x_0, R) \) with center \( z_0 := x_0, y^* := x_0 \)
(2) FOR \( t = 0, \ldots, \text{poly} \) DO
  (4) Submit \( z_t \) to \( \varepsilon \)-weak separation oracle
  (5) **Case (A) \( \rightarrow a \):** Compute \( E_{t+1} \supseteq E_t \cap \{ x \mid a^T x \leq a^T z_t \} \)
  (6) **Case (B) \( \rightarrow y \):**
    (7) IF \( c^T y > c^T y^* \) THEN \( y^* := y \)
    (8) Compute \( E_{t+1} \supseteq E_t \cap \{ x \mid c^T x \geq c^T z_t \} \)

**Case (B):**
Grötschel-Lovász-Schrijver Algorithm

- **INPUT**: \( c \in \mathbb{Q}^n, x_0 \in \mathbb{Q}^n, \varepsilon, r, R \in \mathbb{Q}_+ : \\
B(x_0, r) \subseteq P \subseteq B(x_0, R) \\
**OUTPUT**: \( y^* \in P \text{ with } c^T y^* \geq OPT_f - \varepsilon \)

(1) Ellipsod \( E_0 := B(x_0, R) \) with center \( z_0 := x_0, y^* := x_0 \)

(2) **FOR** \( t = 0, \ldots, \text{poly} \) **DO**

(4) Submit \( z_t \) to \( \varepsilon \)-weak separation oracle

(5) **Case** (A) \( \rightarrow a \): Compute \( E_{t+1} \supseteq E_t \cap \{ x \mid a^T x \leq a^T z_t \} \)

(6) **Case** (B) \( \rightarrow y \in P \):

(7) **IF** \( c^T y > c^T y^* \) **THEN** \( y^* := y \)

(8) Compute \( E_{t+1} \supseteq E_t \cap \{ x \mid c^T x \geq c^T z_t \} \)

**Case** (B):

![Diagram of the Grötschel-Lovász-Schrijver Algorithm](image-url)
Analysis

Theorem

Let \( OPT_f = \max \{ c^T x \mid x \in P \} \). The GLS algorithm finds a \( y^* \in P \) with \( c^T y^* \geq OPT_f - \varepsilon \).

- Suppose for contradiction this is false.
- Let \( x^* \in P \) be opt. sol.; \( \varphi \) input size.
- Inequalities from case (A) never cut points from \( P \)
- Ineq. from case (B) never cut points better than \( OPT_f - \frac{\varepsilon}{2} \) (otherwise we would have found a suitable \( y^* \))
- Let \( U := \text{conv}\{B(x_0, r), x^*\} \) and \( U' = \{ x \in U \mid c^T x \geq OPT_f - \frac{\varepsilon}{2} \} \). By standard volume bounds: \( \text{vol}(U') \geq \left( \frac{1}{2} \right)^{\text{poly}(\varphi)} \). But \( U' \subseteq E_t \ \forall t \). After \( \text{poly}(\varphi) \) many it. \( \text{vol}(E_t) = (1 - \frac{\Theta(1)}{n})^t \cdot \text{vol}(E_0) < \text{vol}(U') \). Contradiction!
A useful observation

Observation

Consider a run of the GLS algorithm for $P \subseteq \mathbb{R}^n$ which yields $y^* \in P$. Let $a_1^T x \leq b_1, \ldots, a_N^T x \leq b_N$ be the inequalities which the oracle are returned for Case (A).

- Each $a_i^T x \leq b_i$ is feasible for $P$
- $c^T y^* \geq \max\{c^T x \mid a_i^T x \leq b_i \ \forall i = 1, \ldots, N\} - \varepsilon$

[Diagram showing a polytope $P$ and a point $y^*$ with inequalities $a_i^T x \leq b_i$ and $c^T y^* \geq \max\{c^T x \mid a_i^T x \leq b_i \ \forall i = 1, \ldots, N\} - \varepsilon$.]

[Diagram notes: $<$ symbol indicating a relationship between $y^*$ and $P$, $\varepsilon$ indicating a tolerance, $c$ indicating the direction.]
Solving $D(\mathcal{P})$

**Lemma**

Suppose $a_i \geq \delta$. Then we can find a feasible solution $y^*$ to $D(\mathcal{P})$ of value $\geq \text{OPT}_f - 1$ in time polynomial in $n, m, \frac{1}{\delta}$.

- Apply GLS algo for $\varepsilon := 1$. Choose $y_0 = (\frac{\delta}{2}, \ldots, \frac{\delta}{2})$.

\[
B\left(y_0, \frac{\delta}{2}\right) \quad \overset{(\delta,\ldots,\delta)^T p \leq 1}{\subseteq} \quad D(\mathcal{P}) \subseteq B(y_0, n)
\]

- We use $\sum_{i=1}^{n} p_i \leq \frac{1}{\delta}$ for any feasible pattern $p \in \mathcal{P}$ since $a_i \geq \delta$
Solving $D(\mathcal{P})$ (2)

- We solve $\varepsilon$-weak separation problem for $z \in \mathbb{Q}^n$.
- If $z_i < 0 \rightarrow$ Case (A) (inequality $z_i \geq 0$ violated)
- If $z_i > 1 \rightarrow$ Case (A) (inequality $z^T e_i \leq 1$ violated)
- Round $z$ down to nearest multiple of $\frac{1}{2m}$ and term this vector $y$. Solve $p^* = \operatorname{argmax}\{y^T p \mid p \in \mathcal{P}\}$ (KNAPSACK with profits from $0, 1 \cdot \frac{1}{2m}, 2 \cdot \frac{1}{2m}, \ldots, 1$)

**Case $y^T p^* > 1$:**

- Then $z^T p^* \geq y^T p^* > 1$ \rightarrow Case (A).

**Case $y^T p^* \leq 1$:**

- Then $y \in D(\mathcal{P})$. And $z^T b - y^T b \leq m \cdot \frac{1}{2m} = \frac{1}{2} = \frac{\varepsilon}{2}$.
  \rightarrow Case (B)

- GLS yields a solution $y^*$ mit $b^T y^* \geq \text{OPT}_f - 1$. 

Finding a near optimal basic solution for \( P(\mathcal{P}) \)

**Theorem**

Suppose \( a_i \geq \delta \). Then we can find a basic solution \( x^* \) for \( P(\mathcal{P}) \) of value \( \leq OPT_f + 1 \) in time polynomial in \( n, m, \frac{1}{\delta} \).

- Run GLS to obtain sol. \( y^* \) to \( D(\mathcal{P}) \) with \( b^T y^* \geq OPT_f - 1 \)
- Let \( y^T p \leq 1, p \in \mathcal{P}' \) be inequalities returned by oracle for case (A). \( \mathcal{P}' \subseteq \mathcal{P} \) has polynomial size and

\[
D(\mathcal{P}) \quad y^* \text{ valid for } D(\mathcal{P}) \quad b^T y^* \geq D(\mathcal{P}') - 1 \quad (1)
\]

- Compute optimum basic solution \( x^* \) for \( P(\mathcal{P}') \) in poly-time.

\[
1^T x^* = P(\mathcal{P}') \stackrel{\text{duality}}{=} D(\mathcal{P}') \stackrel{(1)}{=} D(\mathcal{P}) + 1 \stackrel{\text{duality}}{=} P(\mathcal{P}) + 1
\]

- \( x^* \) is also a (non-optimal) basic solution for \( P(\mathcal{P}) \)
Geometric Grouping

- **Input:** Instance $I = (a_1, \ldots, a_n)$, $size(I) = \sum_{i=1}^{n} a_i b_i \leq n$, $a_i \geq \delta$

- **Output:** Rounded up instance $I'$ with $n/2$ different item sizes $OPT_f(I') \leq OPT_f(I)$ plus waste of $O(\log \frac{1}{\delta})$

1. Sort items w.r.t. sizes $e_1 \leq e_2 \leq \ldots \leq e_m$ ($a_i$ appears $b_i$ times)

2. Let $G_1 = \{e_1, \ldots, e_{\ell_1}\}$ be minimal set of items with $\sum_{i \in G_1} e_i \geq 2$, then continue with $G_2, \ldots$. Let $\ell_i := |G_i|$ be number of items in $G_i$

3. Remove first and last group $\rightarrow$ waste

4. From $G_i$ throw away smallest $\ell_i - \ell_{i+1}$ items $\rightarrow$ waste

5. Round up items in $G_i$ to largest item $\rightarrow I'$
Geometric Grouping (2)

Lemma

Size of waste is $O(\log \frac{1}{\delta})$.

- Size of 1st and last group is $O(1)$
- Consider group $G_i$. Total size of items in $G_i$ is $\leq 3$.
- Num of groups is $\leq n/2$. Clearly $\frac{2}{\delta} \geq \ell_1 \geq \ell_2 \geq \ldots$
- The $n_i := \ell_i - \ell_{i+1}$ smallest items in $G_i$ have size $\leq 3\frac{n_i}{\ell_i}$.

$$
\text{waste} \leq 3 \sum_i \frac{n_i}{\ell_i} \leq 3 \sum_{j=1}^{\ell_1} \frac{1}{j} \ell_1 \leq 2/\delta \quad O(\log \frac{1}{\delta})
$$

$\ell_i$ items of total size $\leq 3$

$n_i$ items of total size $\leq 3\frac{n_i}{\ell_i}$
The algorithm

Algorithm:
1. Compute a basic solution \( x \) to \( P(\mathcal{P}) \) with \( 1^T x \leq OPT_f + 1 \)
2. Buy \( \lfloor x_p \rfloor \) times pattern \( p \), let \( I \) be remaining instance
3. Apply geometric grouping to \( I \) (with \( n \) different item sizes) → \( I' \) (with \( n/2 \) different item sizes)
4. Recurse

Theorem

One has \( APX \leq OPT_f + O(\log^2 n) \).

- Since \( x \) is basic solution, \( |\{p \mid x_p > 0\}| \leq n \).
- After (2) \( size(I) \leq \sum_p (x_p - \lfloor x_p \rfloor) \leq n \).
- Let \( x^t \) be solution \( x \) in iteration \( t \). We buy \( \sum_p \lfloor x_p^t \rfloor \) bins, but \( OPT_f \) decreases by the same quantity.
- We pay in total \( OPT_f + \) total waste. We have \( O(\log n) \) recursions; in each recursion we have a waste of \( O(\log \frac{1}{\delta}) = O(\log n) \).
State of the art

- Computing $OPT$ exactly is $\textbf{NP}$-hard even if the numbers $a_i$ are unary encoded (i.e. $\text{BINPACKING}$ is strongly $\textbf{NP}$-hard).

### Open question
One can compute a $\text{BIN PACKING}$ solution with $\leq OPT + 1$ bins in poly-time?

### Mixed Integer Roundup Conjecture
One has $OPT \leq \lfloor OPT_f \rfloor + 1$. 
Part 15
Minimum Makespan Scheduling

Source: Approximation Algorithms (Vazirani, Springer Press)
Minimum Makespan

**Problem:** Minimum Makespan Scheduling

- **Given:** \( n \) jobs, job \( j \) has processing time \( p_j \). Number \( m \) of machines.
- **Find:** Assign jobs to machines to minimize the makespan.

\[
OPT = \min_{I_1 \cup \ldots \cup I_m = \{1, \ldots, n\}} \left\{ \max_{i = 1, \ldots, m} \left\{ \sum_{j \in I_i} p_j \right\} \right\}
\]
A PTAS for Minimum Makespan Scheduling

Algorithm:
(1) Guess $OPT$
(2) Call job with $p_j > \varepsilon \cdot OPT$ large and small otherwise $\rightarrow$ sub-instance $I$ of large jobs
(3) Round processing times $p_j$ for large jobs down to multiple of $OPT \cdot \varepsilon^2 \rightarrow$ instance $I'$ with processing times $p'_j$
(4) Distribute rounded large jobs $I'$ such that makespan is $\leq OPT$
(5) Distribute small jobs consecutively on least loaded machine
Analysis

Lemma

The algorithm runs in polynomial time and produces a makespan of at most \((1 + \varepsilon)OPT\).

- Large jobs with rounded processing times can be distributed optimally in polynomial time since: \(1/\varepsilon^2\) different job sizes, at most \(1/\varepsilon\) large jobs per machine, hence \(O((1/\varepsilon^2)^{1/\varepsilon})\) many ways how to pack a machine, hence \(\leq n^{O((1/\varepsilon^2)^{1/\varepsilon})}\) possible solutions.

- Clearly \(OPT(I') \leq OPT(I) \leq OPT\). Let \(I_i\) set of jobs on most loaded machine (attaining the makespan).

- Case: Small jobs don’t inc. makespan. No small job in \(I_i\).

\[
\sum_{j \in I_i} p_j \leq \sum_{j \in I_i} (p_j' + \varepsilon \cdot \varepsilon OPT) \leq p_j' \leq (1 + \varepsilon)OPT
\]
Analysis (2)

- \( \text{OPT} \geq \frac{1}{m} \sum_{j=1}^{n} p_j = \text{average load} \)
- **Case: Small jobs do inc. makespan.** Then all machines are filled up to makespan \(- \varepsilon \cdot \text{OPT} \leq \text{OPT} \). Hence makespan \( \leq (1 + \varepsilon)\text{OPT} \)

![Diagram showing small job and makespan relation](image-url)
Hardness

Lemma

There is no FPTAS for Minimum Makespan Scheduling unless \( \text{NP} = \text{P} \).

- Recall that given a BinPacking instance \( I = (a_1, \ldots, a_n), a_i \in \mathbb{N} \) unary encoded and \( m, B \in \mathbb{N} \), it is \( \text{NP} \)-hard to decide, whether \( m \) bins of size \( B \) suffice to pack the items.

- Suppose there is an FPTAS for Minimum Makespan Scheduling. Take items as jobs, \( m \) as number of machines and \( \varepsilon := \frac{\sum_{i=1}^{n} a_i + 1}{\sum_{i=1}^{n} a_i} \). Then the FPTAS would give an exact answer.

\[ \text{opt. makespan} \leq B \iff \exists \text{ Bin Packing solution with } m \text{ bins.} \]
Part 16
Scheduling on Unrelated Parallel Machines

Source: Approximation Algorithms (Vazirani, Springer Press)
Scheduling on Unrelated Parallel Machines

**Problem:** Unrelated Machine Scheduling

- **Given:** Jobs $J = \{1, \ldots, n\}$, machines $M = \{1, \ldots, m\}$. Running job $j$ on machine $i$ takes a processing time $p_{ij}$.
- **Find:** Assign jobs to machine to minimize the makespan.

$$OPT = \min_{I_1 \cup \ldots \cup I_m = \{1, \ldots, n\}} \left\{ \max_{i=1, \ldots, m} \left\{ \sum_{j \in I_i} p_{ij} \right\} \right\}$$
How NOT to solve the problem

LP:
\[
\begin{align*}
\min T \\
\sum_{i \in M} x_{ij} &= 1 \quad \forall j \in J \\
\sum_{j \in J} p_{ij} x_{ij} &\leq T \quad \forall i \in M \\
x_{ij} &\geq 0 \quad \forall i \forall j
\end{align*}
\]

Variables:
\[
x_{ij} = \begin{cases} 
1 & \text{job } j \text{ is assigned to machine } i \\
0 & \text{otherwise}
\end{cases}
\]
\[
T = \text{makespan}
\]

Example: 1 job with execution time \(p_{i1} = m, \forall i = 1, \ldots, m\)

Fractional solution: \(x_{i1} = \frac{1}{m}\)

Integer solution: \(x_{11} = 1\)

\(T = m\)

▶ Integrality gap of \(\geq m\)
A 2-approximation

Algorithm:

(1) Guess $OPT$

(2) Compute basic solution $x^*$ to

$$\sum_{i \in M} x_{ij} = 1 \quad \forall j \in J$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq OPT \quad \forall i \in M$$

$$x_{ij} = 0 \quad \text{for } i, j \text{ with } p_{ij} > OPT$$

$$x_{ij} \geq 0 \quad \forall i \in M \forall j \in J$$

(3) $x_{ij} = 1 \Rightarrow$ assign job $j$ to machine $i$

(4) For not yet assigned jobs: Assign $j$ to a machine $i$ with $0 < x^*_{ij} < 1$ s.t. every machine receives at most 1 extra job
The algorithm runs in polynomial time and the makespan is at most $OPT + \max\{p_{ij} \mid x^*_{ij} > 0\} \leq 2 \cdot OPT$.

- Running time is clearly polynomial:
  We solve a poly size LP in (2) and solve a maximum matching problem in (4).
- Let $H = (J \cup M, E)$ with $E := \{(j, i) \mid 0 < x^*_{ij} < 1\}$. For claim on makespan we need to show that $E$ contains a \{j not assigned in (3)\}-perfect matching.
Theorem

The algorithm runs in polynomial time and the makespan is at most $OPT + \max \{ p_{ij} \mid x_{ij}^* > 0 \} \leq 2 \cdot OPT$.

- Running time is clearly polynomial: We solve a poly size LP in (2) and solve a maximum matching problem in (4).

- Let $H = (J \cup M, E)$ with $E := \{(j, i) \mid 0 < x_{ij}^* < 1\}$. For claim on makespan we need to show that $E$ contains a
  \{$j$ not assigned in (3)$\}$-perfect matching.
Assigning the fractional jobs (1)

Claim

Consider a connected component \((\bar{\mathcal{J}} \cup \bar{\mathcal{M}}, \bar{\mathcal{E}})\) of \(H\). Then \(\bar{x}^* = (x^*_{ij})_{(j,i) \in \bar{\mathcal{E}}}\) is still a basic solution of the subsystem \(LP(\bar{\mathcal{E}})\).

\[
\sum_{i \in \bar{\mathcal{M}}} x_{ij} = 1 \quad \forall j \in \bar{\mathcal{J}} \quad (LP(\bar{\mathcal{E}}))
\]

\[
\sum_{j \in \bar{\mathcal{J}}} p_{ij} x_{ij} \leq T - \sum_{j \notin \bar{\mathcal{J}}} p_{ij} x^*_{ij} \quad \forall i \in \bar{\mathcal{M}}
\]

\[
0 \leq x_{ij} \leq 1 \quad \forall (j, i) \in \bar{\mathcal{E}}
\]

Reason: If \(\bar{x}^* \in \text{conv}(\{y^{(1)}, y^{(2)}\})\) then \(x^* = (\bar{x}^*, \hat{x}^*) \in \text{conv}(\{(y^{(1)}, \hat{x}^*), (y^{(2)}, \hat{x}^*)\})\).

Contradiction.
Assigning the fractional jobs (2)

- $\bar{x}$ is basic solution, hence $|\bar{E}| = |\{(j, i) \mid 0 < \bar{x}_{ij}^* < 1\}| \leq |\bar{J}| + |\bar{M}| \leq \#\text{nodes in } \bar{E}$

- But $\bar{E}$ is connected, thus $\bar{E}$ is a tree + $\leq 1$ extra edge.
- Jobs have degree $\geq 2$, hence leaves must be machines. As long as there are machine-leaves $i$, assign a $j$ with $x_{ij} > 0$ to $i$ and remove both, $i$ and $j$.
- A single even length job-machine cycle (potentially) remains. Extract a matching and we are done.

$\bar{E}$:
State of the art

Exercise

There is no \((3/2 - \varepsilon)\)-apx for **Unrelated Machine Scheduling** unless **NP = P**.

Open Problem 1

Is there a \(3/2\)-apx?

Open Problem 2

A \((2 - \varepsilon)\)-apx is still unknown even for the **Restricted Assignment Problem** where \(p_{ij} \in \{p_j, \infty\}\).

Theorem (Ebenlendr, Krčal, Sgall ’08)

There is a 1.75-apx for the **Restricted Assignment Problem** if each job \(j\) is admissible on \(\leq 2\) machines.
Part 17
Multiprocessor Scheduling with Precedence Constraints

Source:

- Lecture notes of Chandra Chekuri
  http://www.cs.illinois.edu/class/sp09/cs598csc/Lectures/lecture6.pdf
Multiprocessor Scheduling with Precedence Constraints

**Problem:** \textsc{precScheduling} \( (P \mid p_i, \text{prec} \mid C_{\text{max}}) \)

- **Given:** Jobs \( J_1, \ldots, J_n \), job \( J_i \) has processing time \( p_i \), precedence relation \( \prec \), \# of machines \( m \)
- **Find:** (Non-preemptive) schedule of the jobs on \( m \) machines respecting the precedence order and minimizing the makespan

- \( J_i \prec J_\ell \) means that \( J_i \) has to be finished, before \( J_\ell \) is allowed to start.

**Input:**

**Solution:**

\[
\begin{array}{c}
1 \\
m \\
makespan
\end{array}
\]
The algorithm

Graham’s List Scheduling:

1. FOR \( t = 1, \ldots \) DO

2. IF a machine \( j \in \{1, \ldots, m\} \) is idle at \( t \)

AND all predecessors of some (not yet processed) job \( J_i \) are already finished

THEN schedule \( J_i \) on machine \( j \) starting from \( t \)

- In other words: At any time, just start a job whenever possible.
The analysis

Theorem

The makespan of the produced schedule is at most $2 \cdot OPT$

- Find a sequence (w.l.o.g. after reordering) $J_1, \ldots, J_k$ s.t.
  - $J_k$ is the last job of the whole schedule that finishes
  - $J_1 \prec J_2 \prec \ldots \prec J_k$ (chain in the partial order $\prec$)
  - $J_i$ is the predecessor of $J_{i+1}$ that is finished last

- After $J_i$ finished $J_{i+1}$ is started as soon as a machine is available. Hence between $J_i$ is finished and $J_{i+1}$ begins, all machines must be fully busy.

- length of all busy periods $\leq OPT$
- Length of chain $J_1, \ldots, J_k$ is $\leq OPT$
- Makespan $\leq$ length chain + busy period $\leq 2 \cdot OPT$
Hardness

**Theorem (Svensson - STOC’10)**

For every fixed $\varepsilon > 0$, there is no $(2 - \varepsilon)$-apx unless a variant of the **Unique Games Conjecture** is false.

**Open Problem**

What is the complexity status of $P3 \mid p_i = 1, \text{prec} \mid C_{\text{max}}$ (i.e. **PESCHEDULING** with unit processing times and 3 machines)?

**Known:**

- 4/3-apx.
- $P2 \mid p_i = 1, \text{prec} \mid C_{\text{max}}$ is poly-time solvable
Part 18
Euclidean TSP

Source: Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems (Arora ’98, Link)
Euclidean Travelling Salesman Problem

**Problem:** \textsc{EuclideanTSP}

- **Given:** Points \( v_1, \ldots, v_n \in \mathbb{Q}^2 \) in the plane.
- **Find:** Minimum cost tour visiting all nodes

\[
\min_{\text{tour } \pi:V \rightarrow V} \left\{ \sum_{i=1}^{n} \| v_i - v_{\pi(i)} \|_2 \right\}
\]

\[ v_j = (x_j, y_j) \quad \| v_i - v_j \|_2 = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \]

**Goal:** Find a PTAS!
A random bounding box

- Choose a minimal square $S$ containing all points.
- W.l.o.g. this square is $[\frac{L}{2}, L]^2$ with $L = n/\varepsilon \in 2^N$ after scaling. Hence $\text{OPT} \geq L = n/\varepsilon$.
- Choose $a, b \in \{1, \ldots, L/2\}$ randomly.
- Let $R = [a, a + L] \times [b, b + L] \supseteq S$ be the randomly shifted bounding box.
Discretization

- Move all points \( v \) to nearest point in \( \mathbb{Z}^2 \).
- Changes the cost of any tour by \( \leq 2n \leq 2\varepsilon \cdot OPT \) (since \( OPT \geq L = n/\varepsilon \))
The dissection

- Divide the $L \times L$ bounding box into 4 squares of size $\frac{L}{2} \times \frac{L}{2}$
- Divide each $\frac{L}{2} \times \frac{L}{2}$ square into 4 squares of size $\frac{L}{4} \times \frac{L}{4}$
- Recurse, until unit size squares are reached
- Size $\frac{L}{2^i} \times \frac{L}{2^i}$ squares are level $i$ squares
- A line segment is on level $i$, if it is the boundary of a level $i$ square but not of a level $i - 1$ square
- A grid line is on level $i$, if it consists of level $i$ segments
The dissection

- Divide the $L \times L$ bounding box into 4 squares of size $\frac{L}{2} \times \frac{L}{2}$
- Divide each $\frac{L}{2} \times \frac{L}{2}$ square into 4 squares of size $\frac{L}{4} \times \frac{L}{4}$
- Recurse, until unit size squares are reached
- Size $\frac{L}{2^i} \times \frac{L}{2^i}$ squares are level $i$ squares
- A line segment is on level $i$, if it is the boundary of a level $i$ square but not of a level $i - 1$ square
- A grid line is on level $i$, if it consists of level $i$ segments
The dissection

- Divide the $L \times L$ bounding box into 4 squares of size $\frac{L}{2} \times \frac{L}{2}$
- Divide each $\frac{L}{2} \times \frac{L}{2}$ square into 4 squares of size $\frac{L}{4} \times \frac{L}{4}$
- Recurse, until unit size squares are reached
- Size $\frac{L}{2^i} \times \frac{L}{2^i}$ squares are level $i$ squares
- A line segment is on level $i$, if it is the boundary of a level $i$ square but not of a level $i - 1$ square
- A grid line is on level $i$, if it consists of level $i$ segments
Basic idea

- **Method:** Use dynamic programming.
- **Idea:** Consider a level $i$ square $Q$ in the dissection. For all ways how $OPT$ can intersect $Q$, compute the cheapest extension inside $Q$ that visits all nodes in $Q$ (using that we computed similar information already for all smaller squares).

- **Difficulty:** The number of possibilities how $OPT$ can cross $Q$ might be exponential/infinite.
- **Solution:** Limit this number.
Basic idea

- **Method:** Use dynamic programming.
- **Idea:** Consider a level $i$ square $Q$ in the dissection. For all ways how $OPT$ can intersect $Q$, compute the cheapest extension inside $Q$ that visits all nodes in $Q$ (using that we computed similar information already for all smaller squares).

- **Difficulty:** The number of possibilities how $OPT$ can cross $Q$ might be exponential/infinite.
- **Solution:** Limit this number.
Portals

- On any level $i$ line segment, place $\frac{1}{\varepsilon} \log L$ many level $i$ portals (plus one per corner)
- Distance of consecutive level $i$ portals is $\leq \frac{L}{2^i} \cdot \frac{\log L}{\varepsilon}$
Well rounded tours

**Definition**

A tour $\pi$ is called well-rounded tour if:

- It leaves and enters squares only at portals.
- Each square is entered at most $\frac{4}{\varepsilon}$ times.

- Each square has $\leq \frac{4}{\varepsilon} \log L + 4$ many portals. The number of times that a well-rounded tour can leave/enter a square is bounded by $\leq \left(\frac{4}{\varepsilon} \log L + 4\right)^{O(1/\varepsilon)}$ (which is polynomial).

**Theorem (Structure Theorem)**

*There is always a well-rounded tour of cost $\leq (1 + O(\varepsilon))OPT.$*
Well rounded tours

**Definition**

A tour $\pi$ is called well-rounded tour if:

- It leaves and enters squares only at portals.
- Each square is entered at most $\frac{4}{\varepsilon}$ times.

- Each square has $\leq \frac{4}{\varepsilon} \log L + 4$ many portals. The number of times that a well-rounded tour can leave/enter a square is bounded by $\leq (\frac{4}{\varepsilon} \log L + 4)^{O(1/\varepsilon)}$ (which is polynomial).

**Theorem (Structure Theorem)**

There is always a well-rounded tour of cost $\leq (1 + O(\varepsilon))OPT$. 
Relation $OPT$ vs. number of crossings

- For the optimum tour $\pi$ and a grid line $\ell$, let $t(\pi, \ell)$ be the number of times that $\pi$ crosses $\ell$.

\[
\frac{1}{3} \cdot \sum_{\text{grid lines } \ell} t(\pi, \ell) \leq OPT \leq \sqrt{2} \cdot \sum_{\text{grid lines } \ell} t(\pi, \ell)
\]

- $OPT = \Theta(1) \cdot \# \text{crossings}$
- **Goal:** Turn opt. tour $\pi$ into a well-rounded tour, such that the expected cost increase is $O(\varepsilon) \cdot \sum_{\ell} t(\pi, \ell)$
- **Alternatively:** Average cost increase per crossing must be $O(1) \cdot \varepsilon$

$t(\pi, \ell) = 4$
Bending edges through portals

- Consider a crossing of the optimum tour \( \pi \) at a grid line \( \ell \)
- \( \Pr[\text{line } \ell \text{ is at level } i] = \frac{2^i}{L} \)
- If line \( \ell \) is at level \( i \), we have to bend edge through the nearest portal and loose \( \leq \frac{L}{2^i} \cdot \frac{\varepsilon}{\log L} \)
- The expected length increase is

\[
\log L \sum_{i=0}^{\log L} \Pr[\ell \text{ at level } i] \cdot \text{portal distance at level } i \\
= \sum_{i=0}^{\log L} \frac{2^i}{L} \cdot \frac{L}{2^i} \cdot \frac{\varepsilon}{\log L} \leq 2\varepsilon
\]
Patching Lemma

Lemma

Given a TSP tour $\pi$, crossing a line segment $\ell$ of length $s$ an arbitrary number of times. $\exists$ tour $\pi'$ crossing $\ell$ at most 2 times which can be obtained by adding segments of length $\leq 6s$.

- Cut $\pi$ at $\ell$. Let $L_1, \ldots, L_t$ be endpoints on the left side, $R_1, \ldots, R_t$ end points on the right. Imagine their distance to $\ell$ as 0. Say $t$ is even (other case is similar).
- Add tours on $L_i$’s and on $R_i$’s of cost $\leq 2s$ each.
- Add matchings $(L_{2i-1}, L_{2i}), (R_{2i-1}, R_{2i})$ for $2i < t$ and 2 edges $(L_{t-1}, R_{t-1}), (L_t, R_t)$ of total cost $\leq 2s$.
- Degree of $V \cup \{L_i, R_i \mid i = 1, \ldots, t\}$ is even. Graph is again connected. Hence there is a tour visiting all nodes (at least once).
Patching Lemma

Lemma

Given a TSP tour $\pi$, crossing a line segment $\ell$ of length $s$ an arbitrary number of times. $\exists$ tour $\pi'$ crossing $\ell$ at most 2 times which can be obtained by adding segments of length $\leq 6s$.

- Cut $\pi$ at $\ell$. Let $L_1, \ldots, L_t$ be endpoints on the left side, $R_1, \ldots, R_t$ end points on the right. Imagine their distance to $\ell$ as 0. Say $t$ is even (other case is similar).
- Add tours on $L_i$’s and on $R_i$’s of cost $\leq 2s$ each.
- Add matchings $(L_{2i-1}, L_{2i}), (R_{2i-1}, R_{2i})$ for $2i < t$ and 2 edges $(L_{t-1}, R_{t-1}), (L_t, R_t)$ of total cost $\leq 2s$.
- Degree of $V \cup \{L_i, R_i \mid i = 1, \ldots, t\}$ is even. Graph is again connected. Hence there is a tour visiting all nodes (at least once).
Patching Lemma

Lemma

Given a TSP tour $\pi$, crossing a line segment $\ell$ of length $s$ an arbitrary number of times. $\exists$ tour $\pi'$ crossing $\ell$ at most 2 times which can be obtained by adding segments of length $\leq 6s$.

- Cut $\pi$ at $\ell$. Let $L_1, \ldots, L_t$ be endpoints on the left side, $R_1, \ldots, R_t$ end points on the right. Imagine their distance to $\ell$ as 0. Say $t$ is even (other case is similar).
- Add tours on $L_i$’s and on $R_i$’s of cost $\leq 2s$ each.
- Add matchings $(L_{2i-1}, L_{2i}), (R_{2i-1}, R_{2i})$ for $2i < t$ and 2 edges $(L_{t-1}, R_{t-1}), (L_t, R_t)$ of total cost $\leq 2s$.
- Degree of $V \cup \{L_i, R_i \mid i = 1, \ldots, t\}$ is even. Graph is again connected. Hence there is a tour visiting all nodes (at least once).
Patching Lemma

Lemma

Given a TSP tour $\pi$, crossing a line segment $\ell$ of length $s$ an arbitrary number of times. $\exists$ tour $\pi'$ crossing $\ell$ at most 2 times which can be obtained by adding segments of length $\leq 6s$.

- Cut $\pi$ at $\ell$. Let $L_1, \ldots, L_t$ be endpoints on the left side, $R_1, \ldots, R_t$ end points on the right. Imagine their distance to $\ell$ as 0. Say $t$ is even (other case is similar).
- Add tours on $L_i$’s and on $R_i$’s of cost $\leq 2s$ each.
- Add matchings $(L_{2i-1}, L_{2i}), (R_{2i-1}, R_{2i})$ for $2i < t$ and 2 edges $(L_{t-1}, R_{t-1}), (L_t, R_t)$ of total cost $\leq 2s$.
- Degree of $V \cup \{L_i, R_i \mid i = 1, \ldots, t\}$ is even. Graph is again connected. Hence there is a tour visiting all nodes (at least once).
Reducing the number of crossings (1)

**MODIFY Procedure:**

- **Input:** Grid line \( \ell \) on level \( i \)
- **Output:** Tour \( \pi' \) crossing each segment of \( \ell \) at most \( 1/\varepsilon \) times

1. FOR \( j = \log L \) downto \( i \) DO
   2. FOR all level \( j \) segments DO
   3. IF segment is crossed \( > 1/\varepsilon \) times THEN reduce \# crossings to 2 via Patching Lemma

\[
j = \log L \quad \cdots \quad j = i + 1 \quad j = i \quad \text{Output:}
\]

```
\ell \quad \ell \quad \ell \quad \ell \quad \ell
```

166 / 292
Reducing the number of crossings (2)

- Starting from optimum tour, we apply MODIFY to all horizontal and vertical grid lines.
- Now consider a fixed grid line $\ell$. Want to show:
  \[
  E[\text{cost for crossing reduction at } \ell] \leq O(\varepsilon) \cdot t(\pi, \ell)
  \]
- Let $c_{\ell,j}$ be number of times that MODIFY is applied to level $j$ segments of grid line $\ell$
- Each application of MODIFY reduces the number of crossings of $\ell$ by $1/\varepsilon - 2 \geq \frac{1}{2\varepsilon}$ (assuming $\varepsilon \leq 1/4$). Hence
  \[
  \sum_{j \geq 0} c_{\ell,j} \leq \frac{t(\pi, \ell)}{1/(2\varepsilon)} = 2\varepsilon \cdot t(\pi, \ell)
  \]
- The cost increase of a single crossing reduction on level $j$ is $\leq 6 \cdot \frac{L}{2^j}$ (by Patching Lemma).
- Thus
  \[
  E[\text{cost increase at } \ell \mid \ell \text{ at level } i] \cdot \leq \sum_{j \geq i} c_{\ell,j} \cdot 6 \cdot \frac{L}{2^j}
  \]
Reducing the number of crossings (3)

\[
\begin{align*}
E[\text{cost for crossing reduction at } \ell] & = \sum_{i \geq 0} \Pr[\ell \text{ at level } i] \cdot E[\text{cost increase at } \ell \mid \ell \text{ at level } i] \\
& \leq \sum_{i \geq 0} \frac{2^i}{L} \cdot \sum_{j \geq i} c_{\ell,j} \cdot \frac{L}{2j} \\
& \overset{\text{reordering}}{=} 6 \sum_{j \geq 0} \frac{c_{\ell,j}}{2^j} \cdot \sum_{i \leq j} 2^i \\
& \leq 12 \cdot \sum_{j \geq 0} c_{\ell,j} \\
\sum_{j \geq 0} c_{\ell,j} & \leq 2\varepsilon \cdot t(\pi, \ell) \\
& \leq 24\varepsilon \cdot t(\pi, \ell)
\end{align*}
\]

\[\exists \text{ well-rounded tour of cost } (1 + O(\varepsilon)) \cdot OPT \]
The dynamic program (1)

- **Table entries:**
  
  \[ A(Q, (s_1, t_1), \ldots, (s_q, t_q)) \]
  
  = cost of cheapest extension of \( q \) subtours to well-rounded tour visiting all nodes in \( Q \) such that subtour \( i \) goes from \( s_i \) to \( t_i \)
  
  \( \forall \) squares \( Q \) \( \forall q \in \{0, \ldots, 4/\varepsilon\} \) \( \forall \) portals \( s_i, t_i \) of \( Q \)

- **Number of table entries:**
  
  - \( O(n \cdot \log L) \) many non-empty squares \( Q \)
  
  - There are \( O\left(\frac{1}{\varepsilon} \log n\right)^{O(1/\varepsilon)} \) many ways to choose \( O(1/\varepsilon) \) portals out of \( O\left(\frac{1}{\varepsilon} \log L\right) \) portals
  
  - Total number of entries:
  
  \[ O(n(\log n)^{O(1/\varepsilon)}) \]
The dynamic program (1)

- **Table entries:**

\[ A(Q, (s_1, t_1), \ldots, (s_q, t_q)) \]

= cost of cheapest extension of \( q \) subtours to well-rounded tour visiting all nodes in \( Q \) such that subtour \( i \) goes from \( s_i \) to \( t_i \)

\( \forall \) squares \( Q \) \( \forall q \in \{0, \ldots, 4/\varepsilon\} \) \( \forall \) portals \( s_i, t_i \) of \( Q \)

- **Number of table entries:**

  - \( O(n \cdot \log L) \) many non-empty squares \( Q \)
  - There are \( O\left(\frac{1}{\varepsilon} \log n\right)^{O(1/\varepsilon)} \) many ways to choose \( O(1/\varepsilon) \) portals out of \( O\left(\frac{1}{\varepsilon} \log L\right) \) portals
  - Total number of entries:

\[ O(n \log n)^{O(1/\varepsilon)} \]
The dynamic program (2)

Lemma

*The best well rounded tour can be computed in* \(O(n(\log n)^{O(1/\varepsilon)})\)

- Compute table entries **bottom-up** (starting with smallest squares)
- For entry \(A(Q, (s_1, t_1), \ldots, (s_q, t_q))\):
  - Let \(Q_1, \ldots, Q_4\) be the subsquares of \(Q\).
  - Guess (i.e. try out all combinations) the visited portals of \(Q_1, \ldots, Q_4\) and their order → \(O(\frac{1}{\varepsilon} \log n)^{O(1/\varepsilon)}\) combinations
- Look up table entries for \(Q_1, \ldots, Q_4\) to determine cost.
The dynamic program (2)

Lemma

The best well rounded tour can be computed in $O(n(\log n)^{O(1/\varepsilon)})$

- Compute table entries bottom-up (starting with smallest squares)
- For entry $A(Q, (s_1, t_1), \ldots, (s_q, t_q))$: Let $Q_1, \ldots, Q_4$ be the subsquares of $Q$. Guess (i.e. try out all combinations) the visited portals of $Q_1, \ldots, Q_4$ and their order $\rightarrow O\left(\frac{1}{\varepsilon} \log n\right)^{O(1/\varepsilon)}$ combinations
- Look up table entries for $Q_1, \ldots, Q_4$ to determine cost.
Generalizations

Advantages of this approach:

- Applicable for many graph optimization problems, when nodes are points in the Euclidean plane (like Steiner Tree, $k$-Median, Steiner Forest, $k$-TSP, $k$-Mst).
- Works for general $\ell_p$-metrics (like maximum-norm).
- Extends to any constant dimension.
- (Theoretically) nice dependence on $\varepsilon$.

**Theorem (Arora '98)**

Let $d \in \mathbb{N}$, $\varepsilon > 0$, $p \in \mathbb{N} \cup \{\infty\}$ be fixed constants. Then there is an expected $(1 + \varepsilon)$-approximation for TSP if the nodes are points in $\mathbb{R}^d$ and distances are measured as $\|v - u\|_p := (\sum_{i=1}^{d} |v_i - u_i|^p)^{1/p}$ in time $n(O(\log n))O(\sqrt{d} \cdot 1/\varepsilon)^{d-1}$. This can be derandomized by increasing the running time by a factor of $O(n/\varepsilon)$.
Part 19
Tree Embeddings

Source: A tight bound on approximating arbitrary metrics by tree metrics (Fakcharoenphol, Rao, Talwar: Link)
Tree metric

Definition (Tree metric)

Given nodes $V$, spanning tree $T$, edge costs $c(e) \ \forall e \in T$. Then $d^T : V \times V \to \mathbb{Q}_+$ with

$$d^T(u, v) := \text{length of } u - v \text{ path in } T$$

is called a tree metric.
**Motivation**: Many optimization problems are easy on trees: **Steiner tree**, Tsp, \( k \)-Tsp, **Steiner Forest**, \( \ldots \)

**Question**: Can we for any node set \( V \) and metric \( d : V \times V \to \mathbb{Q}_+ \), find a tree metric \( d^T \) such that

\[
d(u, v) \leq d^T(u, v) \leq \alpha \cdot d(u, v) \quad \forall u, v \in V
\]

for a small distortion \( \alpha \)?

**Possible approach**: For some graph optimization problem, compute tree \( T \). Then solve problem on tree optimally (or get \( O(1) \)-apx). Obtain a \( \alpha \)-apx (or \( O(\alpha) \)-apx) for original problem.
Motivation

- **Motivation:** Many optimization problems are easy on trees: Steiner tree, Tsp, k-Tsp, Steiner Forest, ... 
- **Question:** Can we for any node set $V$ and metric $d : V \times V \rightarrow \mathbb{Q}_+$, find a tree metric $d^T$ such that
  \[ d(u, v) \leq d^T(u, v) \leq \alpha \cdot d(u, v) \quad \forall u, v \in V \]
  for a small distortion $\alpha$?

- **Possible approach:** For some graph optimization problem, compute tree $T$. Then solve problem on tree optimally (or get $O(1)$-apx). Obtain a $\alpha$-apx (or $O(\alpha)$-apx) for original problem.
One good, one bad news

Bad news:

**Theorem (Rabinovitch, Raz ’95)**

Any tree embedding for an $n$-cycle must have distortion $\Omega(n)$.

Good news:

- Delete a random edge.
- For $u, v \in V$ with $d(u, v) = k$ one has $d^T(u, v) = n - k$ with probability $\frac{k}{n}$ and $d^T(u, v) = k$ with probability $1 - \frac{k}{n}$.
- Expected distortion is at most 2 since:

$$E[d^T(u, v)] = \frac{k}{n}(n - k) + \left(1 - \frac{k}{n}\right) \cdot k \leq 2 \cdot k$$
The Theorem

**Theorem (Fakcharoenphol, Rao, Talwar ’03)**

*Given any metric \((V, d)\), one can find randomly (in time \(O(n^2)\)) a tree metric \((V \cup U, d^T)\) such that*

- \(d(u, v) \leq d^T(u, v)\) \(\forall u, v \in V\) (i.e. \(d^T\) dominates \(d\))
- \(E[d^T(u, v)] \leq O(\log n) \cdot d(u, v)\) \(\forall u, v \in V\)

*That means the tree metric has an expected \(O(\log n)\) distortion.*

**Remark:** The tree will contain extra nodes \(U\), which were not contained in the original nodeset.
Preliminaries

Assumptions:

- $2^\delta = \max_{u,v \in V} \{d(u, v)\}$ is diameter
- $d(u, v) > 1 \ \forall u \neq v$

Definition

A set system $S$ is called laminar if for every $S_1, S_2 \in S$ one has either $S_1 \cap S_2 = \emptyset$ or $S_1 \subseteq S_2$ or $S_2 \subseteq S_1$.

Idea: Obtain a random laminar family.
Clustering

**Algorithm:**

1. Choose a random permutation $\pi$ on nodes $V$
2. Choose $\beta \in [0, 1]$ uniformly at random
3. $D_\delta := \{V\}$
4. FOR $i = \delta - 1$ DOWNTO 0 DO
   5. Assign every node to first node (w.r.t. order $\pi$) that has distance $\leq 2\beta \cdot 2^{i-1}$
   6. All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_i$

$D_\delta$:

- $\pi(1)$
- $\pi(2)$
- $\pi(3)$
- $\pi(4)$
- $\pi(5)$
- $\pi(6)$
- $\pi(7)$
- $\pi(8)$
Clustering

Algorithm:
(1) Choose a random permutation $\pi$ on nodes $V$
(2) Choose $\beta \in [0, 1]$ uniformly at random
(3) $D_\delta := \{V\}$
(4) FOR $i = \delta - 1$ DOWNTO 0 DO
   (5) Assign every node to first node (w.r.t. order $\pi$) that has distance $\leq 2^\beta \cdot 2^{i-1}$
   (6) All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_i$
Clustering

Algorithm:

1. Choose a random permutation \( \pi \) on nodes \( V \)
2. Choose \( \beta \in [0, 1] \) uniformly at random
3. \( D_\delta := \{V\} \)
4. FOR \( i = \delta - 1 \) DOWNTO 0 DO
   5. Assign every node to first node (w.r.t. order \( \pi \)) that has distance \( \leq 2^\beta \cdot 2^{i-1} \)
   6. All nodes that are assigned to the same node and are in the same cluster (in \( D_{i+1} \)) form a new cluster of \( D_i \)

\[ D_i : \]

- \( \pi(1) \)
- \( \pi(2) \)
- \( \pi(3) \)
- \( \pi(4) \)
- \( \pi(5) \)
- \( \pi(6) \)
- \( \pi(8) \)
Clustering

Algorithm:

1. Choose a random permutation $\pi$ on nodes $V$
2. Choose $\beta \in [0, 1]$ uniformly at random
3. $D_{\delta} := \{V\}$
4. FOR $i = \delta - 1$ DOWNTO 0 DO
   5. Assign every node to first node (w.r.t. order $\pi$) that has distance $\leq 2^\beta \cdot 2^{i-1}$
   6. All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_{i}$
Clustering

Algorithm:

1. Choose a random permutation $\pi$ on nodes $V$
2. Choose $\beta \in [0, 1]$ uniformly at random
3. $D_\delta := \{V\}$
4. FOR $i = \delta - 1$ DOWNTO 0 DO
   5. Assign every node to first node (w.r.t. order $\pi$) that has distance $\leq 2^\beta \cdot 2^{i-1}$
6. All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_i$
Clustering

Algorithm:
1. Choose a random permutation $\pi$ on nodes $V$
2. Choose $\beta \in [0, 1]$ uniformly at random
3. $D_\delta := \{V\}$
4. FOR $i = \delta - 1$ DOWNTO 0 DO
   5. Assign every node to first node (w.r.t. order $\pi$) that has distance $\leq 2^\beta \cdot 2^{i-1}$
   6. All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_i$
Clustering

Algorithm:

(1) Choose a random permutation $\pi$ on nodes $V$

(2) Choose $\beta \in [0, 1]$ uniformly at random

(3) $D_\delta := \{V\}$

(4) FOR $i = \delta - 1$ DOWNTO 0 DO

(5) Assign every node to first node (w.r.t. order $\pi$) that has distance $\leq 2^\beta \cdot 2^{i-1}$

(6) All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_i$

$D_0 :$

$\bullet \pi(2)$

$\bullet \pi(4)$

$\bullet \pi(3)$

$\bullet \pi(1)$

$\bullet \pi(6)$

$\bullet \pi(5)$

$\bullet \pi(8)$

$\bullet \pi(7)$
Clustering

Algorithm:

1. Choose a random permutation $\pi$ on nodes $V$
2. Choose $\beta \in [0, 1]$ uniformly at random
3. $D_\delta := \{V\}$
4. FOR $i = \delta - 1$ DOWNTO 0 DO
   5. Assign every node to the first node (w.r.t. order $\pi$) that has distance $\leq 2^\beta \cdot 2^{i-1}$
   6. All nodes that are assigned to the same node and are in the same cluster (in $D_{i+1}$) form a new cluster of $D_i$
Defining the tree metric

- Each cluster becomes an extra node
- Insert edge of cost $2^i$ between $S \in D_i, S' \in D_{i-1}$ if $S' \subseteq S$

\[
\begin{align*}
D_0 & \quad \cdots \quad D_i \quad \cdots \quad D_{\delta} \\
D_0 & \quad \cdots \quad D_{i-1} \quad \cdots \quad D_0
\end{align*}
\]

- Note that in the last iteration ($i = 0$) we assign each node to a cluster center at distance $\leq 2^\beta \cdot 2^{0-1} \leq 1$. Hence the clusters of $D_0$ are indeed singletons (since $d(u, v) > 1 \ \forall u \neq v$).
$d^T$ dominates $d$

Lemma

The tree metric $d^T$ dominates $d$, i.e. $d(u, v) \leq d^T(u, v) \forall u, v \in V$

- Suppose $u, v$ are in the same $D_i$ cluster, but separated by $D_{i-1}$
- Cluster in $D_i$ have diameter $\leq 2 \cdot 2^\beta \cdot 2^{i-1} \leq 2^{i+1}$
- On the other hand $d^T(u, v) \geq 2 \cdot 2^i$.
- Hence $d(u, v) \leq 2^{i+1} \leq d^T(u, v)$  \[ \square \]
Proof of $O(\log n)$ average distortion

Lemma

For any $u, v \in V$: $E[d^T(u, v)] = O(\log n) \cdot d(u, v)$

- If only one of the nodes $u, v$ is assigned to center $w$ in an iteration $i$, then we say $w$ cuts edge $(u, v)$ at level $i$.
- We want to charge the $u$-$v$ distance to that cluster center that cuts the $u$-$v$ edge

$$d^T_w(u, v) := \sum_{i: w \text{ cuts } (u, v) \text{ at level } i} 2^{i+2}$$

- Then

$$d^T(u, v) \leq \sum_{w \in V} d^T_w(u, v)$$

since: Suppose $u, v$ are separated by $D_i$ (i.e. they are in the same $D_{i+1}$ cluster). Then $d^T(u, v) \leq \sum_{j=0}^{i+1} 2 \cdot 2^j \leq 2 \cdot 2^{i+2}$. But in iteration $i$, we find 2 cluster centers $w, w'$ that cut edge $(u, v)$, for both $d^T_w(u, v), d^T_{w'}(u, v) \geq 2^{i+2}$. 
Proof of $O(\log n)$ average distortion (2)

- Assume w.l.o.g. that $d(u, w_s) < d(v, w_s)$.
- Let $w_1, w_2, \ldots$ be nodes in increasing distance from $u$.
- $w_s$ can cut $(u, v)$ only if
  - (A) There exists level $i$, where $d(u, w_s) \leq 2^\beta \cdot 2^{i-1} < d(v, w_s)$
  - (B) $u$ is assigned to $w_s$
Proof of $O(\log n)$ average distortion (3)

- Assume for a second: $\exists i : 2^{i-1} \leq d(u, w_s) < d(v, w_s) < 2^i$.
- Then there is only one level $i$ at which $w_s$ might cut $(u, v)$.
- By triangle inequality, the length of the interval $[d(u, w_s), d(v, w_s)]$ is
  
  $$d(v, w_s) - d(u, w_s) \leq d(u, v).$$

- Logscale length of interval is at most $\log_2 \left( \frac{2^{i-1} + d(u, v)}{2^{i-1}} \right)$.

$$\Pr[(A)] \leq \log_2 \left( \frac{2^{i-1} + d(u, v)}{2^{i-1}} \right)^{\log_2(1+x) \leq 2x} \leq 2 \cdot \frac{d(u, v)}{2^{i-1}}$$

Standard:

$2^0 \quad 2^1 \quad \ldots \quad 2^{i-1} \quad d(u, w_s) \quad d(v, w_s) \quad 2^i$

Logscale:
Proof of $O(\log n)$ average distortion (3)

- Assume for a second: $\exists i : 2^{i-1} \leq d(u, w_s) < d(v, w_s) < 2^i$.
- Then there is only one level $i$ at which $w_s$ might cut $(u, v)$.
- By triangle inequality, the length of the interval $[d(u, w_s), d(v, w_s)]$ is

$$d(v, w_s) - d(u, w_s) \leq d(u, v).$$

- Logscale length of interval is at most $\log_2 \left(\frac{2^{i-1} + d(u, v)}{2^{i-1}}\right)$.

$$\Pr[(A)] \leq \log_2 \left(\frac{2^{i-1} + d(u, v)}{2^{i-1}}\right)^{\log_2(1+x)\leq 2x} \leq 2 \cdot \frac{d(u, v)}{2^{i-1}}$$

**Standard:**

- $2^0$ to $2^1$ to $2^{i-1}$

**Logscale:**

- $\beta$ to $1$ to $1$ to $2^{i-1}$

Cluster sizes: $2^\beta$ and $2^{i-1}$
Proof of $O(\log n)$ average distortion (4)

- Next, condition on $(A)$.

\[
\Pr[u \text{ assigned to } w_s|(A)] \leq \Pr[w_s \text{ 1st of } w_1, \ldots, w_s \text{ w.r.t. } \pi] = \frac{1}{s}
\]

- If $(A) \& (B)$ happen, this incurs cost of $2^{i+2}$.

- Hence

\[
E[d^T_{w_s}(u, v)] \leq 2^{i+2} \cdot 2 \cdot \frac{d(u, v)}{2^{i-1}} \cdot \frac{1}{s} = O\left(\frac{d(u, v)}{s}\right)
\]

- For general case: Let $\delta_i$ be length of

\[
[d(u, w_s), d(v, w_s)] \cap [2^{i-1}, 2^i]
\]

Then applying the arguments for each $\delta_i$: \[E[d^T_{w_s}(u, v)] \leq \sum_i \delta_i \cdot O\left(\frac{1}{s}\right) \leq O\left(\frac{d(u, v)}{s}\right).

- Then

\[
E[d^T(u, v)] \leq \sum_{s=1}^{n-2} E[d^T_{w_s}(u, v)] = \sum_{s=1}^{n-2} O\left(\frac{d(u, v)}{s}\right) = O(\log n) \cdot d(u, v)
\]
Distortion must be $\Omega(\log n)$

**Definition (Expander graph)**

An undirected graph $G = (V, E)$ is called an $(n, d, \alpha)$-expander graph if

- $|V| = n$
- constant degree: $\text{deg}(v) = d \ \forall v \in V$
- edge expansion
  \[
  \alpha = \min_{1 \leq |S| \leq n/2} \frac{\delta(S)}{|S|}
  \]

- Random $d$-regular graphs are good expanders w.h.p.
- The diameter of expanders is $\Theta(\log n)$.

**Theorem (Bartal ’96)**

A randomized tree embedding of any $(n, d, \alpha)$-expander graph $(d, \alpha$ constants) must have an edge with expected distortion of $\Omega(\log n)$. 
Steiner nodes are not really necessary

**Theorem (Gupta ’01)**

Given a weighted tree $T = (V, E, c)$, where the node set $V = R \cup S$ consists of required vertices $R$ and Steiner nodes $S$. Then in linear time, one can find a weighted tree $T^* = (R, E^*, c^*)$ such that

$$d^T(u, v) \leq d^{T^*}(u, v) \leq 8 \cdot d^T(u, v)$$

where $d^T$ and $d^{T^*}$ are the induced tree metrics.
Theorem (FRT + Gupta + Charikar et al.)

Given a complete graph $G = (V, E)$ with metric cost function $c : E \rightarrow \mathbb{Q}_+$. One can find deterministically, in polynomial time: spanning trees $T_1, \ldots, T_q$ on $V$, costs $d_i : T_i \rightarrow \mathbb{Q}_+$ and probabilities $\lambda_i > 0$, $\lambda_1 + \ldots + \lambda_q = 1$ where $q = \text{poly}(n)$. Then

- For $u, v \in V$ and $i = 1, \ldots, q$ one has $c(u, v) \leq d^{T_i}(u, v)$
- For any $u, v \in V$ one has

$$
\sum_{i=1}^{q} \lambda_i \cdot d^{T_i}(u, v) \leq O(\log n) \cdot c(u, v).
$$

Here $d^{T_i} : V \times V \rightarrow \mathbb{Q}_+$ is the tree metric induced by $T_i$ and $d_i$. 

Derandomization
PART 20
INTRODUCTION INTO PRIMAL DUAL ALGORITHMS

A generic problem

**Situation:** We want to approximate a problem, which (in many cases) is of the form

\[
\begin{align*}
\min & \sum_{j=1}^{n} c_j x_j \\
\sum_{j=1}^{n} a_{ij} x_j & \geq b_i \, \forall i = 1, \ldots, m \\
x_j & \in \{0, 1\} \, \forall j = 1, \ldots, n
\end{align*}
\]

**Examples so far:** Set Cover, Steiner tree, Vertex Cover, \ldots
A primal-dual pair

Primal "covering" LP:

\[
\min \sum_{j=1}^{n} c_j x_j \quad (P)
\]

\[
\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad \forall i = 1, \ldots, m
\]

\[
x_j \geq 0 \quad \forall j = 1, \ldots, n
\]

Dual "packing" LP:

\[
\max \sum_{i=1}^{m} b_i y_i \quad (D)
\]

\[
\sum_{i=1}^{m} a_{ij} y_i \leq c_j \quad \forall j = 1, \ldots, n
\]

\[
y_i \geq 0 \quad \forall i = 1, \ldots, m
\]
A generic Approximation algorithm

Generic primal-dual algorithm:

(1) $x := 0$, $y = 0$

(2) WHILE $x$ not feasible DO

(3) Increase dual variables in a suitable way until some dual constraint $j$ becomes tight

(4) Set $x_j := 1$

(5) RETURN $x$

Generic analysis:

- Show: At the end $x$ is integer and feasible for primal
- Show: At the end $y$ is feasible for dual
- Show: $\sum_{j=1}^{n} c_j x_j \leq \alpha \cdot \sum_{i=1}^{m} b_i y_i$ ($\alpha$ is the apx factor)

\[ \text{dual solutions} \quad \text{primal solutions} \]

\[ 0 \quad \sum_{i=1}^{m} b_i y_i \quad OPT_f \quad OPT \quad \sum_{j=1}^{n} c_j x_j \leq \text{factor of } \alpha \]
Relaxed complementary slackness

Lemma

Let $\alpha, \beta \geq 1$. Let $x, y$ be primal/dual feasible solutions obtained by the algorithm. If

(A) Relaxed primal compl. slack.: $x_j > 0 \Rightarrow c_j \leq \alpha \sum_{i=1}^{m} a_{ij} y_i$

(B) Relaxed dual compl. slack.: $y_i > 0 \Rightarrow \sum_{j=1}^{n} a_{ij} x_j \leq \beta \cdot b_i$

Then $APX \leq \alpha \cdot \beta \cdot OPT_f$.

- Let $APX$ be the cost of the produced solution. Then

$$APX = \sum_{j=1}^{n} c_j x_j \overset{(A)}{\leq} \sum_{j=1}^{n} x_j \left( \alpha \sum_{i=1}^{m} a_{ij} y_i \right) = \alpha \sum_{i=1}^{m} y_i \sum_{j=1}^{n} a_{ij} x_j$$

$$\overset{(B)}{\leq} \alpha \beta \sum_{i=1}^{m} y_i b_i \leq \alpha \beta \cdot OPT_f \quad \square$$
Part 21  
Steiner Forest  

Steiner Forest

**Problem:** Steiner Forest

- **Given:** Undirected graph $G = (V, E)$, edge cost $c : E \rightarrow \mathbb{Q}_+$, terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$
- **Find:** Minimum cost subgraph $F$ connecting all terminal pairs:

$$OPT = \min_{F \subseteq E} \left\{ \sum_{e \in F} c(e) \mid \forall i = 1, \ldots, k : F \text{ connects } s_i \text{ and } t_i \right\}$$
Steiner Forest

**Problem:** Steiner Forest

- **Given:** Undirected graph $G = (V, E)$, edge cost $c : E \to \mathbb{Q}_+$, terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$
- **Find:** Minimum cost subgraph $F$ connecting all terminal pairs:

$$OPT = \min_{F \subseteq E} \left\{ \sum_{e \in F} c(e) \mid \forall i = 1, \ldots, k : F \text{ connects } s_i \text{ and } t_i \right\}$$
The LP relaxation

- For any $S \subseteq V$ define cut requirement

$$f(S) = \begin{cases} 
1 & \text{if } \exists i : |S \cap \{s_i, t_i\}| = 1 \\
0 & \text{otherwise}
\end{cases}$$

Primal LP relaxation:

$$\min \sum_{e \in E} c_e x_e \quad (P)$$

$$\sum_{e \in \delta(S)} x_e \geq f(S) \quad \forall S \subseteq V$$

$$x_e \geq 0 \quad \forall e \in E$$

Dual LP:

$$\max \sum_{S \subseteq V} f(S)y_S \quad (D)$$

$$\sum_{S : e \in \delta(S)} y_S \leq c_e \quad \forall e \in E$$

$$y_S \geq 0 \quad \forall S \subseteq V$$
Preliminaries

- For $F \subseteq E, S \subseteq V$: $\delta_F(S) = \{\{u, v\} \in F \mid u \in S, v \notin S\}$
- A cut $S \subseteq V$ is violated by $F \subseteq E$, if there is a terminal pair $(s_i, t_i)$ with $|\{s_i, t_i\} \cap S| = 1$ but $\delta_F(S) = \emptyset$
- A cut $S$ is active w.r.t. $F$, if $S$ is violated and minimal (i.e. there is no subset $S' \subset S$ that is also violated).
- An edge $e$ is tight w.r.t. a dual solution $(y_S)_S$ if $\sum_{S : e \in \delta(S)} y_S = c_e$
  (i.e. if the dual constraint of $c_e$ satisfied with equality).
The algorithm

(1) $F := \emptyset$, $y := 0$

(2) WHILE $\exists$ violated cut DO

(3) Increase simultaneously $y_S$ for all active cuts $S$, until some edge $e$ gets tight

(4) Add the tight edge $e$ to $F$

(5) Compute an arbitrary minimal feasible solution $F' \subseteq F$
The active cuts

**Lemma**

The active cuts w.r.t. $F \subseteq E$ are connected components of $F$.

- Consider active cut $S$ ($S$ minimal, $f(S) = 1$, $\delta_F(S) = \emptyset$).
- $\delta_F(S) = \emptyset \Rightarrow$ connected components of $F$ are either fully contained in $S$ or fully outside
- $S$ is violated, hence there is a pair $|\{s_i, t_i\} \cap S| = 1$
- The connected component of $F$ inside $S$ that contains $s_i$ is also violated. Hence, $S$ is a single connected component (or we would have a contradiction).

\[\square\]
Example
Example

\begin{itemize}
\item \textbf{active set}
\end{itemize}
Example

$y_S = 6$ for $S = \{s_2\}$

edges added to $F$
Example
Example
Example

$F$ at the end of WHILE loop
Example

Solution $F'$
Feasibility

Lemma

\( F' \) is a feasible solution.

- Let \( F \) be the solution at the end of the WHILE loop.
- \( F \) is feasible, because there is no violated cut.
- We do not delete necessary edges, hence \( F' \) is also feasible.

Lemma

\( y \) is dual feasible, i.e. \( \sum_{S:e \in \delta(S)} y_S \leq c_e \) for all \( e \in E \).

- Each time that an edge \( e \) gets tight (i.e. \( \sum_{S:e \in \delta(S)} y_S = c_e \)), we add it to \( F \).
- We increase \( y_S \) only for violated cuts – not for cuts containing edges of \( F \).
The main analysis (1)

**Lemma**

Let \( y \) be the dual solution at the end of the algorithm. Then

\[
APX = \sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \leq 2 \cdot \text{OPT}_f.
\]

\[
\sum_{e \in F'} c_e^{\text{tight}} = \sum_{e \in F'} \left( \sum_{S: e \in \delta(S)} y_S \right) = \sum_{S \subseteq V} |\delta_{F'}(S)| \cdot y_S \overset{(*)}{\leq} \sum_{S \subseteq V} 2y_S
\]

- Consider any iteration \( i \). Let \( \alpha \) be the amount by which the dual variables \( y_S \) were increased. We show \((*)\) by proving

\[
\alpha \cdot \sum_{S \text{ active in it.}i} |\delta_{F'}(S)| \leq 2 \cdot \alpha \cdot \#\text{active sets in it.}i
\]
The main analysis (2)

- Consider an intermediate iteration $i$ with intermediate $F$.
- **Remark:** $F' \setminus F$ might contain edges that are added later $F \setminus F'$ might contain edges that are deleted at the end.
- **Claim:**
  \[
  \sum_{S \text{ active in it.}i} |\delta_{F'}(S)| \leq 2 \cdot \#\text{active sets in iteration } i
  \]
- Shrink connected components of $F \rightarrow H'$ ($S$ becomes node $v_S$). Nodes $v_S$ stemming from active cuts $S$ are active nodes, others are inactive nodes
- $H'$ is a forest. Degrees are preserved.
The main analysis (2)

- Consider an intermediate iteration $i$ with intermediate $F$.
- **Remark:** $F' \setminus F$ might contain edges that are added later $F \setminus F'$ might contain edges that are deleted at the end.
- **Claim:**
  \[
  \sum_{S \text{ active in } i} |\delta_{F'}(S)| \leq 2 \cdot \#\text{active sets in iteration } i
  \]

- Shrink connected components of $F \rightarrow H' \ (S$ becomes node $v_S)$. Nodes $v_S$ stemming from active cuts $S$ are active nodes, others are inactive nodes.

- $H'$ is a forest. Degrees are preserved.
The main analysis (2)

- Consider an intermediate iteration $i$ with intermediate $F$.
- **Remark:** $F' \setminus F$ might contain edges that are added later. $F \setminus F'$ might contain edges that are deleted at the end.
- **Claim:**

\[
\sum_{S \text{ active in it.}i} |\delta_{F'}(S)| \leq 2 \cdot \# \text{active sets in iteration } i
\]

- Shrink connected components of $F \rightarrow H'$ ($S$ becomes node $v_S$). Nodes $v_S$ stemming from active cuts $S$ are active nodes, others are inactive nodes.

**$H'$:**

- $H'$ is a forest. Degrees are preserved.
The main analysis (2)

\[ H' : \]

Consider non-singleton leaf \( v_S \). Edge to \( v_S \) was not deleted. Hence \( f(S) = 1 \). But then \( S \) was active (since \( S \) is a connected component of \( F \) at iteration \( i \)).

Average degree over all nodes in a forest is \( \leq 2 \) (since \# edges \( \leq \# \) nodes) and each edge contributes at most 2 to the degrees.

Inactive nodes are inner nodes of degree \( \geq 2 \), hence average degree of active nodes \( \leq \) average degree \( \leq 2 \). \( \square \)
Deleting redundant edges is crucial

**Observation:** Without the pruning step at the end of the algorithm, the solution would cost $n + 4$ instead of 4.
Conclusion

Theorem

The primal dual algorithm produces a 2-approximation in time $O(n^2 \log n)$.

Remark: The algorithm works whenever the requirement function $f : 2^V \rightarrow \{0, 1\}$ is proper, that means

- $f(V) = 0$
- $f(S) = f(V \setminus S)$ (symmetry)
- If $A, B \subseteq V$ are disjoint and $f(A \cup B) = 1$ then $f(A) = 1$ or $f(B) = 1$.

Note: Function $f$ for STEINER FOREST is proper.
State of the art

- There is no $\frac{96}{95}$-approximation algorithm unless $\text{NP} = \text{P}$ (same ratio as for the special case of $\text{STEINER TREE}$).
- There is still no better than 2-approximation known.
- The integrality gap of the considered LP is in fact exactly 2.
- There is also no other LP formulation known, which might have a smaller gap.
PART 22

FACILITY LOCATION

Facility Location

**Problem: Facility Location**

- **Given:** Facilities $F$, cities $C$, opening cost $f_i$ for every facility $i$. Metric cost $c_{ij}$ for connecting city $j$ to facility $i$.
- **Find:** Set of facilities $I$ and an assignment $\phi : C \rightarrow I$ of cities to opened facilities, minimizing the total cost:

$$OPT = \min_{I \subseteq F, \phi:C \rightarrow I} \left\{ \sum_{i \in I} f_i + \sum_{j \in C} c_{\phi(j),j} \right\}$$

- **Remark:** Without the metric assumption, the problem becomes $\Theta(\log n)$-hard.
- We assume w.l.o.g. $c_{ij}, f_i \in \mathbb{Z}_+$
Facility Location

**Problem:** Facility Location

- **Given:** Facilities $F$, cities $C$, opening cost $f_i$ for every facility $i$. Metric cost $c_{ij}$ for connecting city $j$ to facility $i$.
- **Find:** Set of facilities $I$ and an assignment $\phi : C \to I$ of cities to opened facilities, minimizing the total cost:

\[
OPT = \min_{I \subseteq F, \phi : C \to I} \left\{ \sum_{i \in I} f_i + \sum_{j \in C} c_{\phi(j),j} \right\}
\]

- **Remark:** Without the metric assumption, the problem becomes $\Theta(\log n)$-hard.
- We assume w.l.o.g. $c_{ij}, f_i \in \mathbb{Z}_+$
The primal dual pair

Primal LP:

\[
\begin{align*}
\text{min} & \quad \sum_{i,j} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \\
\sum_{i \in F} x_{ij} & \geq 1 \quad \forall j \in C \\
x_{ij} & \leq y_i \quad \forall i \in F \ \forall j \in C \\
x_{ij} & \geq 0 \quad \forall i \in F \ \forall j \in C \\
y_i & \geq 0 \quad \forall i \in F
\end{align*}
\]

Dual LP:

\[
\begin{align*}
\text{max} & \quad \sum_{j \in C} \alpha_j \\
\alpha_j & \leq c_{ij} + \beta_{ij} \quad \forall i \in F \ \forall j \in C \\
\sum_{j \in C} \beta_{ij} & \leq f_i \quad \forall i \in F \\
\alpha_j & \geq 0 \quad \forall j \in C \\
\beta_{ij} & \geq 0 \quad \forall i \in F \ \forall j \in C
\end{align*}
\]

Intuition:

- \(\alpha_j\) is the amount that city \(j\) ”pays” in total.
- \(\beta_{ij}\) is what city \(j\) ”pays” to open facility \(i\).
The algorithm - Phase 1:

(1) Initially all cities are unconnected
(2) $\alpha := 0, \beta := 0, F_t := \emptyset$
(3) WHILE not all cities are connected DO
(4) FOR ALL unconnected cities $j$ DO
(5) Increase $\alpha_j$ (by 1 per time unit)
(6) For tight edges $\alpha_j = c_{ij} + \beta_{ij}$ increase also $\beta_{ij}$
(7) IF $\sum_j \beta_{ij} = f_i$ (new) THEN
(8) open facility $i$ temporarily ($F_t := F_t \cup \{i\}$)
(9) FOR ALL cities $j$ where edge $(i, j)$ is tight DO
(10) connect city to facility $i$
(11) facility $i$ is connection witness of $j$: $w(j) := i$

Phase 2:

(1) Let $H = (F_t, E')$ with $(i, i') \in E'$ if $\exists j \in C : \beta_{ij}, \beta_{i'j} > 0$
(2) Open a maximal independent set $I \subseteq F_t$
(3) FOR ALL $j \in C$ DO
(4) IF $\exists j \in I : \beta_{ij} > 0$ THEN $\varphi(j) := i$ ($j$ directly conn.)
(5) ELSE IF $w(j) \in I$ THEN $\varphi(j) := w(j)$ ($j$ directly conn.)
(6) ELSE $\varphi(j) :=$ a neighbour of $w(j)$ in $H$ ($j$ indir. conn.)
Example:

Phase 1 - Time: 0

\[ \begin{align*}
\alpha_1 &= 0 \\
\alpha_2 &= 0 \\
\alpha_3 &= 0 \\
\alpha_4 &= 0
\end{align*} \]

The diagram shows a network with nodes labeled \( C \) and \( F \) and edges with weights indicated. The values \( f_1 = 4 \), \( f_2 = 5 \), and \( f_3 = 2 \) are given, possibly indicating flow values through the network.
Example:

Phase 1 - Time: 1

\[
\begin{align*}
\alpha_1 &= 1 & C & 1 & \beta = 0 & F & f_1 = 4 \\
& & 1 & \beta = 0 & & \\
& 10 & & & & \\
\alpha_2 &= 1 & 1 & \beta = 0 & f_2 = 5 \\
& 3 & & & & \\
& 5 & & & & \\
& 3 & & & & \\
\alpha_3 &= 1 & 1 & \beta = 0 & f_3 = 2 \\
& 10 & & & & \\
& 5 & & & & \\
& 3 & & & & \\
\alpha_4 &= 1 & 3 & & & \\
& 3 & & & & \\
\end{align*}
\]
Example:

Phase 1 - Time: 2
Example:

Phase 1 - Time: 3

conn.: \( w(1) = 1, \ \alpha_1 = 3 \)

conn.: \( w(2) = 1, \ \alpha_2 = 3 \)

conn.: \( w(3) = 1, \ \alpha_3 = 3 \)

\( \alpha_4 = 3 \)

\( f_1 = 4 \) temp. opened

\( f_2 = 5 \)

\( f_3 = 2 \)
Example:

Phase 1 - Time: 4

conn.: $w(1) = 1$, $\alpha_1 = 3$

conn.: $w(2) = 1$, $\alpha_2 = 3$

conn.: $w(3) = 1$, $\alpha_3 = 3$

conn.: $w(4) = 2$, $\alpha_4 = 4$
Example:

**Phase 2: Graph $H$**

$$f_1 = 4 \text{ temp. opened}$$

$$f_2 = 5 \text{ temp. opened}$$

$$f_3 = 2$$
Example:

Phase 2: The solution

\[ C \rightarrow F \in I \text{ (facility opened)} \]

\[ f_3 = 2 \]
Analysis

Theorem

One has $\sum_{j \in C} c_{\varphi(j), j} + \sum_{i \in I} f_i \leq 3 \sum_{j \in C} \alpha_j$.

We account the dual "payments"

$$\alpha^f_j := \text{payment for opening} := \begin{cases} \beta_{\varphi(j), j} & \text{if } j \text{ directly connected} \\ 0 & \text{if } j \text{ is indirectly conn.} \end{cases}$$

$$\alpha^c_j := \text{payment for connection} := \begin{cases} c_{\varphi(j), j} & \text{if } j \text{ directly connected} \\ \alpha_j & \text{if } j \text{ is indirectly conn.} \end{cases}$$

Claim: $\alpha_j = \alpha^f_j + \alpha^c_j$.

- For indirectly connected cities: clear
- For directly connected cities: $\alpha_j = c_{\varphi(j), j} + \beta_{\varphi(j), j}$ because edge $(\varphi(j), j)$ was tight.
Bounding the opening costs

Lemma

The dual prices pay for the opening cost, i.e.

$$\sum_{i \in I} f_i = \sum_{j \in C} \alpha_j^f.$$ 

- A facility $i \in I$ was temporarily opened because $\sum_j \beta_{ij} = f_i$
- All $j$ with $\beta_{ij} > 0$ must be directly connected to $i$ because: We opened an independent set in $H$ in Phase 2, hence any $i' \in F_t$ with $\beta_{i'i} > 0$ is not in $I$
- Thus all $j$ with $\beta_{ij} > 0$

$$\sum_{j: \phi(j) = i} \alpha_j^f = \sum_{j: \beta_{ij} > 0} \beta_{ij}^{\text{temp opened}} = f_i$$

- The claim follows from

$$\sum_{j \in C} \alpha_j^f = \sum_{i \in I} \sum_{j: \phi(j) = i} \alpha_j^f = \sum_{i \in I} f_i$$

$\blacksquare$
Bounding the connection cost

Lemma

For any city \( j \in C \) one has \( c_{\varphi(j),j} \leq 3\alpha_j^c \).

- If \( j \) directly connected, then even \( \alpha_j^c = c_{\varphi(j),j} \). Next, suppose \( j \) is indirectly connected.
- Then there is an edge \((w(j), \phi(j)) \in H\) (since \( j \) was indirectly connected).
- This edge implies that there is a \( j' \in C \) with \( \beta_{\varphi(j),j'} > 0, \beta_{w(j),j'} > 0 \).

\[
\begin{align*}
j & \quad \text{tight: } \alpha_j \geq c_{w(j),j} \\
j' & \quad \beta_{w(j),j'} > 0 \\
w(j) & \notin I \\
j' & \in H \\
& \beta_{\phi(i),j'} > 0 \\
\phi(j) & \in I
\end{align*}
\]
Bounding the connection cost (2)

- Event $\beta_{w(j),j} > 0$ only happened if $\alpha_j \geq c_{w(j),j}$. For the same reason: $\alpha_{j'} \geq c_{w(j),j'}$ and $\alpha_{j'} \geq c_{\phi(j),j'}$.

**Claim** $\alpha_j \geq \alpha_{j'}$: Consider the time $t$, when $w(j)$ was temporarily opened. Since $w(j)$ is connection witness of $j$, $\alpha_j \geq t$. At this time $t$, it was $\beta_{w(j),j'} > 0$ (since if $\beta_{w(j),j'} = 0$ at that time, then $\beta_{w(j),j'} = 0$ forever). At the latest at this time $t$, also $j'$ was connected and $\alpha_{j'}$ stopped growing. Hence $\alpha_j \geq t \geq \alpha_{j'}$.

- Then

  \[
  c_{\phi(j),j} \leq c_{w(j),j} + c_{w(j),j'} + c_{\phi(j),j'} \leq 3\alpha_j = 3\alpha_j^c
  \]

  \[\square\]
Conclusion

Theorem

The algorithm produces a 3-approximation in time $O(m \cdot \log(m))$, where $m = |C| \cdot |F|$ is the number of edges.

State of the art:

Theorem (Byrka ’07)

There is a 1.499-apx for Facility Location.

- The integrality gap for the considered LP lies in $[1.463, 1.499]$.

Theorem

There is no polynomial time 1.463-apx for Facility Location unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$. 
Part 23
Insertion: Semidefinite Programming

Source: Approximation Algorithms (Vazirani, Springer Press)
Positive definite matrices

Definition (positive semidefinite Matrix)

A matrix $A \in \mathbb{R}^{n \times n}$ is called positive semi-definite if

$$\forall x \in \mathbb{R}^n : x^T A x \geq 0.$$ 

Theorem (Diagonalization)

Let $A \in \mathbb{R}^{n \times n}$ be symmetric (i.e. $a_{ij} = a_{ji}$), then $A$ is diagonalizable, i.e. one can write

$$A = \left( \begin{array}{ccc} & & \vdots \\ \vdots & \ddots & \vdots \\ v_1 & \cdots & v_n \end{array} \right) = L \left( \begin{array}{cccc} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{array} \right) = D \left( \begin{array}{ccc} & & \vdots \\ \vdots & v_1 & \vdots \\ \vdots & \vdots & \ddots \\ \vdots & v_n & \vdots \end{array} \right) = L^T,$$

where $v_i \in \mathbb{R}^n$ is orthonormal Eigenvector for Eigenvalue $\lambda_i$, i.e $Av_i = \lambda_i v_i$, $\|v_i\|_2 = 1$, $v_i^T v_j = 0 \ \forall i \neq j$. 
Some useful results

Lemma

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix ($v_i$ orthonormal Eigenvector for $\lambda_i$). Then the following statements are equivalent

(1) $\forall x \in \mathbb{R}^n : x^T Ax \geq 0$
(2) $\lambda_i \geq 0 \forall i$
(3) There is $W \in \mathbb{R}^{n \times n}$ with $A = W^T W$

\begin{itemize}
  \item (1) $\Rightarrow$ (2). $0 \leq v_i^T Av_i = v_i^T (\lambda_i v_i) = \lambda_i v_i^T v_i = \lambda_i^{1}$
  \item (2) $\Rightarrow$ (3). $A = LDL^T = L\sqrt{D} \sqrt{D} L^T = (\sqrt{D} L^T)^T (\sqrt{D} L^T) =: W$
  \item (3) $\Rightarrow$ (1). For any $x \in \mathbb{R}^n$:
    \[ x^T Ax = x^T (W^T W)x = (Wx)^T \cdot (Wx) \geq 0 \]
\end{itemize}

Remark: Matrix $W$ can be found by Cholesky decomposition in $O(n^3)$ arithmetic operations (if $\sqrt{}$ counts as 1 operation).
The semidefinite cone

- **Def.:** Write $Y \succeq 0$ if $Y$ is positive semidefinite.
- **Fact:** The set
  \[
  \{ Y \in \mathbb{R}^{n \times n} \mid Y \succeq 0, Y \text{ symmetric} \} = \text{cone}\{xx^T \mid x \in \mathbb{R}^n \}
  \]
  is a convex, non-polyhedral cone.
A semidefinite program

Given:

- Obj. function vector $C = (c_{ij})_{1 \leq i, j \leq n} \in \mathbb{Q}^{n \times n}$
- Linear constraints $A_k = (a_{ij}^k)_{1 \leq i, j \leq n} \in \mathbb{Q}^{n \times n}, \ b_k \in \mathbb{Q}$

$$\max \sum_{i,j} c_{ij} y_{ij}$$

$$\sum_{i,j} a_{ij}^k y_{ij} \leq b_k \quad \forall k = 1, \ldots, m$$

$Y$ symmetric

$Y \succeq 0$

- Frobenius inner product: $C \bullet Y := \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot y_{ij}$
A semidefinite program

Given:

- Obj. function vector $C = (c_{ij})_{1 \leq i, j \leq n} \in \mathbb{Q}^{n \times n}$
- Linear constraints $A_k = (a_{ij}^k)_{1 \leq i, j \leq n} \in \mathbb{Q}^{n \times n}$, $b_k \in \mathbb{Q}$

\[
\begin{align*}
\text{max} \quad & C \cdot Y \\
A_k \cdot Y & \leq b_k \quad \forall k = 1, \ldots, m \\
Y & \text{ symmetric} \\
Y & \geq 0
\end{align*}
\]

- Frobenius inner product: $C \cdot Y := \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot y_{ij}$
Pathological situations

- **Case: All solutions might be irrational.** $x = \sqrt{2}$ is the unique solution of
  \[
  \begin{pmatrix}
  1 & x & 0 & 0 \\
  x & 2 & 0 & 0 \\
  0 & 0 & 2x & 2 \\
  0 & 0 & 2 & x \\
  \end{pmatrix} \succeq 0
  \]

- **Case: All sol. might have exponential encoding length.** Let $Q_1(x) = x_1 - 2, Q_i(x) := \begin{pmatrix} 1 & x_{i-1} \\ x_{i-1} & x_i \end{pmatrix}$. Then
  \[
  Q(x) := \begin{pmatrix}
  Q_1(x) & 0 & \ldots & 0 \\
  0 & Q_2(x) & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \ldots & Q_n(x) \\
  \end{pmatrix} \succeq 0
  \]
  if and only if $Q_1(x), \ldots, Q_n(x) \succeq 0$. I.e. $x_1 - 2 \geq 0$ and $x_i \geq x_{i-1}^2$, hence $x_n \geq 2^{2^{n-1}}$. 

\[ \]

\[250 / 292\]
Solvability of Semidefinite Programs

Theorem

Given rational input $A_1, \ldots, A_m, b_1, \ldots, b_m, C, R$ and $\varepsilon > 0$, suppose

$$SDP = \max\{C \cdot Y \mid A_k \cdot Y \leq b_k \forall k; \ Y \text{ symmetric}; \ Y \succeq 0\}$$

is feasible and all feasible points are contained in $B(0, R)$. Then one can find a $Y^*$ with

$$A_k \cdot Y^* \leq b_k + \varepsilon, \ Y^* \text{ symmetric, } Y^* \succeq 0$$

such that $C \cdot Y^* \geq SDP - \varepsilon$. The running time is polynomial in the input length, $\log(R)$ and $\log(1/\varepsilon)$ (in the Turing machine model).
Solving the separation problem

- **Remark:** We show that we can solve the separation problem, ignore numerical inaccuracies.
- Let infeasible $Y$ be given, we have to find a separating hyperplane.

1. **Case** $A_k \cdot Y < b_k$: return "$A_k \cdot Y \geq b_k$ violated".
2. **Case** $Y$ not symmetric: Find the $i, j$ with $y_{ij} < y_{ji}$. Return "$y_{ij} \geq y_{ji}$ violated".
3. **Case** $Y$ not positive semidefinite. Find eigenvector $v$ with Eigenvalue $\lambda < 0$, i.e. $Yv = \lambda v$. Then

   \[ \sum_{i,j} v_i^T v_j \cdot y_{ij} = v^T Y v < 0 \]

   hence return "$\sum_{i,j} v_i^T v_j \cdot y_{ij} \geq 0$ violated".
Vector programs

Idea:

\[ Y \text{ symmetric and } Y \succeq 0 \]
\[ \iff \exists W = (v_1, \ldots, v_n) \in \mathbb{R}^{n \times n} : W^T W = Y \]
\[ \iff \exists v_1, \ldots, v_n \in \mathbb{R}^n : y_{ij} = v_i^T v_j \]

SDP:

\[
\begin{align*}
\max & \sum_{i,j} c_{ij} y_{ij} \\
\sum_{i,j} a_{ij}^k \cdot y_{ij} & \leq b_k \quad \forall k \\
Y & \text{ sym.} \\
Y & \succeq 0
\end{align*}
\]

Vector program:

\[
\begin{align*}
\max & \sum_{i,j} c_{ij} v_i^T v_j \\
\sum_{i,j} a_{ij}^k \cdot v_i^T v_j & \leq b_k \quad \forall k \\
v_i & \in \mathbb{R}^n \quad \forall i
\end{align*}
\]

Observation

The SDP and the vector program are equivalent.
Part 24
MaxCut

Source:

- *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming* (Goemans, Williamson) ([link](#))
Problem definition

**Problem: MaxCut**

- **Given:** Complete undirected graph $G = (V, E)$, edge weights $w : E \rightarrow \mathbb{Q}_+$
- **Find:** Cut maximizing the weight of separated edges

$$OPT = \max_{S \subseteq V} \left\{ \sum_{e \in \delta(S)} w(e) \right\}$$

![Diagram of a graph with a cut set $S$ and weight $w_{ij} = 1$]
A vector program

- Choose decision variable for any node \( i \in V \):

\[
v_i = \begin{cases} 
(1, 0, \ldots, 0) & i \in S \\
(-1, 0, \ldots, 0) & i \notin S 
\end{cases}
\]

- An exact MAXCUT vector program:

\[
\max \sum_{(i,j) \in E} \frac{w_{ij}}{2} (1 - v_i^T v_j)
\]

\[
v_i^T v_i = 1 \quad \forall i = 1, \ldots, n
\]

\[
v_i \in \mathbb{R}^n \quad \forall i = 1, \ldots, n
\]

\[
v_i = (\pm 1, 0, \ldots, 0) \quad \forall i = 1, \ldots, n
\]

- Then

\[
\sum_{(i,j) \in E} w_{ij} \cdot \frac{1}{2} (1 - v_i^T v_j) = \sum_{(i,j) \in \delta(S)} w_{ij} = \text{1 if } (i,j) \in \delta(S), \ 0 \text{ o.w.}
\]

\[
\sum_{(i,j) \in E} w_{ij} \cdot \frac{1}{2} (1 - v_i^T v_j) = \sum_{(i,j) \in \delta(S)} w_{ij} = -1 \text{ if } (i,j) \in \delta(S) \quad \text{1 o.w.}
\]
A vector program (2)

The relaxed vector program:

$$\max \sum_{(i,j) \in E} \frac{w_{ij}}{2} (1 - v_i^T v_j)$$

$$v_i^T v_i = 1 \quad \forall i = 1, \ldots, n$$

$$v_i \in \mathbb{R}^n \quad \forall i = 1, \ldots, n$$
A physical interpretation

- $n$ vectors on $n$-dim unit ball.
- Repulsion force of $w_{ij}$ between $v_i$ and $v_j$

Example:

**Graph $G$**

**SDP solution:**

- $OPT = 4$
- For SDP solution, place $v_1, \ldots, v_5$ equidistantly on 2-dim. subspace. $SDP = 5 \cdot \frac{1}{2} (1 - \cos(\frac{4}{5}\pi)) \approx 4.52$
- Hence integrality gap $\geq 1.13$. 
The algorithm

Algorithm:

(1) Solve $\text{MAXCUT}$ vector program $\rightarrow v_1, \ldots, v_n \in \mathbb{Q}^n$
   (More precisely: Solve the equivalent SDP, obtain a matrix $Y \in \mathbb{Q}^{n \times n}$. Apply Cholesky decomposition to $Y$ to obtain $v_1, \ldots, v_n$)

(2) Choose randomly a vector $r$ from $n$-dimensional unit ball

(3) Choose cut $S := \{i \mid v_i \cdot r \geq 0\}$

Theorem

$$E\left[\sum_{(i,j) \in \delta(S)} w_{ij}\right] \geq 0.87 \cdot \text{OPT} \text{ (i.e. the algorithm gives an expected 1.13-apx).}$$
Proof

- Consider 2 vectors $v_i, v_j$ with angle $\theta \in [0, \pi]$. Let $\mathbb{R} \cdot a$ be the 1-dim. intersection of the $n - 1$-dim. hyperplane $x \cdot r = 0$ with the plane spanned by $v_i, v_j$
- $a$ has a random direction
- $v_i, v_j$ are separated
  $\iff$ they lie on different sides of line $a\mathbb{R}$
  $\iff$ $a$ lies in one of the 2 gray arcs of angle $\theta$
- $\Pr[v_i$ and $v_j$ separated$] = 2 \cdot \frac{\theta}{2\pi} = \frac{\theta}{\pi}$
- Expected contribution to $APX$ is $w_{ij} \cdot \frac{\theta}{\pi}$

$(i, j) \in \delta(S) :$

$(i, j) \notin \delta(S) :$
Proof (2)

- Expected contribution of edge \((i, j)\) to \(APX\) is \(w_{ij} \cdot \frac{\theta}{\pi}\)
- Contribution of edge \((i, j)\) to \(SDP\) is \(w_{ij} \cdot \frac{1}{2}(1 - \cos(\theta))\)

\[
\frac{E[APX]}{SDP} \geq \min_{0 \leq \theta \leq \pi} \frac{\theta/\pi}{\frac{1}{2}(1 - \cos(\theta))} \approx 0.878. \quad \square
\]
State of the art

Theorem (Khot, Kindler, Mossel, O’Donnell ’05)

There is no polynomial time < 1.138-approximation algorithm (unless the Unique Games Conjecture is false).

- That means the presented approximation is the best possible.
Part 25
Max2Sat

Source: Approximation Algorithms (Vazirani, Springer Press)
Problem definition

**Problem:** Max2Sat

- **Given:** SAT formula $\bigwedge_{C \in C} C$ on variables $x_1, \ldots, x_n$. Each clause $C$ contains at most 2 literals.
- **Find:** Truth assignment maximizing the number of satisfied clauses

$$OPT = \max_{a=(a_1, \ldots, a_n) \in \{0,1\}^n} \left| \{ C \in C \mid C \text{ true for assignment } a \} \right|$$

**Example:**

$$\underbrace{(\bar{x}_1 \lor x_2) \land (x_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (x_1 \lor x_2) \land \bar{x}_1}_{\text{clause}}$$

Optimal assignment: $a = (0,1)$ with 4 satisfied clauses.

**Remark:** Problem is NP-hard though testing whether all clauses can be satisfied is easy.
A quadratic program

Goal: Write MAX2SAT as quadratic program

\[ \max \sum_{i,j} a_{ij} (1 + y_i y_j) + b_{ij} (1 - y_i y_j) \]

\[ y_i^2 = 1 \]
\[ y_i \in \mathbb{Z} \]

for suitable coefficients \( a_{ij}, b_{ij} \).

Here \( y_i = 1 \equiv x_i \) true, \( y_i = -1 \equiv x_i \) false.

Let \( y_0 := 1 \) be auxiliary variable.

Write

\[ v(C') = \begin{cases} 1 & \text{if clause } C \text{ true for } y \\ 0 & \text{otherwise} \end{cases} \]

For clauses with 1 literal

\[ v(x_i) = \frac{1 + y_0 y_i}{2}, \quad v(\bar{x}_i) = \frac{1 - y_0 y_i}{2} \]
A quadratic program (2)

- For clause \( x_i \lor x_j \)

\[
v(x_i \lor x_j) = 1 - v(\bar{x}_i) \cdot v(\bar{x}_j) = 1 - \frac{1 - y_0y_i}{2} \cdot \frac{1 - y_0y_j}{2}
\]

\[
= \frac{1}{4} (3 + y_0y_i + y_0y_j - y_0^2 y_iy_j) = 1
\]

\[
= \frac{1}{4} + \frac{1}{4} + \frac{1 - y_iy_j}{4}
\]

- Similar for \( \bar{x}_i \lor x_j \) and \( \bar{x}_i \lor \bar{x}_j \).

- We obtain promised coefficients \( a_{ij}, b_{ij} \) by summing up  
\[
\sum_{C \in \mathcal{C}} v(C).
\]

- Now: Relax the quadratic program to a (solvable) vector program.
The algorithm

Algorithm:

1. Solve MAXCUT vector program

\[
\max \sum_{0 \leq i < j \leq n} \left( a_{ij}(1 + v_i v_j) + b_{ij}(1 - v_i v_j) \right)
\]

\[v_i^2 = 1 \ \forall i = 0, \ldots, n\]

\[v_i \in \mathbb{R}^{n+1}\]

2. Choose randomly a vector \( r \) from \( n \)-dimensional unit ball

3. Let \( y_i := 1 \) for all \( i \) that are on the same side of the hyperplane \( x \cdot r = 0 \) as \( v_0 \) (the "truth" vector)

---

**Theorem**

Let \( APX := \# \text{satisfied clauses} \). Then \( E[APX] \geq 0.87 \cdot SDP \).
Analysis

Case: Term \( b_{ij}(1 - v_i v_j) \) with angle \( \theta \) between \( v_i, v_j \)
- Contribution to \( E[APX] \): \( 2b_{ij} \cdot \Pr[y_i \neq y_j] = 2b_{ij} \frac{\theta}{\pi} \)
- Contribution to Vector program: \( b_{ij}(1 - \cos(\theta)) \)
- Gap: \( \min_{0 \leq \theta \leq \pi} \frac{2\theta/\pi}{1-\cos(\theta)} \approx 0.878 \)

Case: Term \( a_{ij}(1 + v_i v_j) \) with angle \( \theta \) between \( v_i, v_j \)
- Contribution to \( E[APX] \): \( 2a_{ij} \cdot \Pr[y_i = y_j] = 2a_{ij}(1 - \frac{\theta}{\pi}) \)
- Contribution to Vector program: \( a_{ij}(1 + \cos(\theta)) \)
- Gap: \( \min_{0 \leq \theta \leq \pi} \frac{2(1-\theta/\pi)}{1+\cos(\theta)} \approx 0.878 \)
State of the art

**Theorem (Feige, Goemans ’95)**

*There is a 1.0741-axp for MAX2Sat.*

**Theorem (Lewin, Livnat, Zwick ’02)**

*There is a 1.064-axp for MAX2Sat.*

**Theorem (Hastad ’97)**

*There is no 1.0476-axp for MAX2Sat (unless $\text{NP} = \text{P}$).*

**Theorem (Khot, Kindler, Mossel, O’Donnell ’05)**

*There is no polynomial time 1.063-axp for MAX2Sat (unless the Unique Games Conjecture is false).*
Part 26
Budgeted Spanning Tree

Source: The Constrained Minimum Spanning Tree Problem
(Goemans, Ravi) (link)
The Budgeted Spanning Tree problem

**Problem:** \textbf{Budgeted Spanning Tree}

- **Given:** Undirected graph $G = (V, E)$ with edge costs $c : E \rightarrow \mathbb{Q}_+$ and edge lengths $\ell : E \rightarrow \mathbb{Q}_+$. Budget $B$.
- **Find:** Spanning tree $T$ minimizing the cost, while not exceeding the budget

$$OPT = \max_{\text{spanning tree } T} \left\{ \sum_{e \in T} c_e \mid \sum_{e \in T} \ell_e \leq B \right\}$$

\[
\begin{array}{cccc}
& (3, 1) & & \\
\downarrow & & \downarrow & \\
(0, 2) & & (2, 1) & \\
& & & \\
& \downarrow & & \\
& (1, 2) & & (2, 1) \\
\end{array}
\]

$B = 4$

**cost** $c(e)$  **length** $\ell(e)$
The Budgeted Spanning Tree problem

**Problem:** \textbf{BUDGETED SPANNING TREE}

- \textbf{Given}: Undirected graph $G = (V, E)$ with edge costs $c : E \rightarrow \mathbb{Q}_+$ and edge lengths $\ell : E \rightarrow \mathbb{Q}_+$. Budget $B$.
- \textbf{Find}: Spanning tree $T$ minimizing the cost, while not exceeding the budget

$$OPT = \max_{\text{spanning tree } T} \left\{ \sum_{e \in T} c_e \mid \sum_{e \in T} \ell_e \leq B \right\}$$

![Graph example with edges labeled by cost $c(e)$ and length $\ell(e)$, and a tree $T$ labeled with edge $(0, 2)$. Budget $B = 4$.](graph.png)
Budgeted Spanning Tree is NP-hard

Recall that PARTITION is (weakly) NP-hard:

**Problem:** PARTITION

- **Given:** Numbers $a_1, \ldots, a_n \in \mathbb{N}$, $S := \sum_{i=1}^{n} a_i$
- **Find:** $I \subseteq \{1, \ldots, n\} : \sum_{i \in I} a_i = S/2$

**Reduction to Budgeted Spanning Tree:**

![Graph diagram](image)

- Budget $B := S/2$. There is a feasible tree $T$ of cost $c(T) \leq B$, $\ell(T) \leq B$ if and only if there is a PARTITION solution.
- Problem also NP-hard for simple graphs (our algorithm will also work for multigraphs).
- Recall: The Spanning Tree problem without a budget is easy.
Lagrangian Relaxation

Original problem:
\[
\min_T c(T) \\
T \text{ spanning tree} \\
\ell(T) \leq B
\]
\[:= OPT\]

Lagrangian Relaxation:
\[
\min_T c(T) + z \cdot (\ell(T) - B) \\
T \text{ spanning tree}
\]
\[:= OPT_{LR}(z)\]

Lemma

For any Lagrange multiplier \(z \geq 0\): \(OPT_{LR}(z) \leq OPT\).

- Let \(T\) be the optimum solution: \(c(T) = OPT, \ell(T) \leq B\). Then

\[
OPT = c(T) \geq c(T) + \begin{cases} 
\leq 0 & \text{if } z \geq 0 \\
\geq 0 & \text{if } z \leq 0 
\end{cases} \cdot (\ell(T) - B) \geq OPT_{LR}(z)
\]
Solving the Lagrangian relaxation

Lemma

A sol. $z^*, T_1, T_2$ can be computed in poly-time where $OPT_{LR} = OPT_{LR}(z^*)$ is attained by $T_1, T_2$, $\ell(T_1) \geq B \geq \ell(T_2)$.

- Assume w.l.o.g. $c(e), \ell(e) \in \mathbb{Z}$. $a_{\text{max}} := \max\{c(e), \ell(e)\}$
- For any spanning tree $T$, let $g_T(z) := c(T) + z \cdot (\ell(T) - B)$
- $OPT_{LR}(z) = \min_T\{g_T(z)\}$. Hence $OPT_{LR}(z)$ is concave.
Solving the Lagrangian relaxation (2)

- For a given $z$, choose $c'(e) := c(e) + z \cdot \ell(e)$, then

$$OPT_{LR}(z) = \min_{\text{sp. tree } T} \{c(T) + z \cdot (\ell(T) - B)\} = \min_{\text{sp. tree } T} \{c'(T)\} - z \cdot B$$

- $OPT_{LR}(0) \geq OPT_{LR}(z)$

- $OPT_{LR}(n \cdot a_{\text{max}}) \leq 0$ (if there is no tree with budget < $B$, then MST w.r.t. $c'(e) := \ell(e) + \frac{1}{n \cdot a_{\text{max}}} c(e)$ is optimal).

- Perform binary search (needs $O(\log(n \cdot a_{\text{max}}))$ iterations):
  1. $L := 0, R := n \cdot a_{\text{max}}$
  2. WHILE $|L - R| \geq \frac{1}{4n^2a_{\text{max}}^2}$ DO
     3. $z := \frac{L+R}{2}$
     4. $T := \text{MST}$ for cost function $c'(e) := c(e) + z \cdot \ell(e)$
     5. IF $\ell(T) > B$ THEN $L := z$ ELSE $R := z$
  6. $z^* :=$ rational number in $[L, R]$ with min. denominator
  7. $T_1 := \text{argmin}_T \{g_T(z^* - \varepsilon)\}$
  8. $T_2 := \text{argmin}_T \{g_T(z^* + \varepsilon)\}$ ($\varepsilon := \frac{1}{8n^2a_{\text{max}}^2}$ should suffice)

- Use: $z^* \in \frac{\mathbb{Z}}{q}$ for some $q \in \{1, \ldots, 4n^2a_{\text{max}}^2\}$
An example

$e_1 : (2, 0)$

$e_2 : (0, 2)$

$e_3 : (3/2, 1)$

$B = 1$

$OPT_{LR} = 1$

$g_{e_2}(z) = 0 + (2 - 1) \cdot z$

$g_{e_3}(z)$

$OPT_{LR}(z)$

$g_{e_1}(z) = 2 + (0 - 1) \cdot z$

In this example $OPT = \frac{3}{2}, OPT_{LR} = 1$
Obtaining 2 trees differing in 2 edges

\[ \text{Lemma} \]

One can find opt. Lagrange solutions \( T_1, T_2 \) with \( \ell(T_1) \geq B, \ell(T_2) \leq B \) which differ in exactly 2 edges.

- Let \( S_0, S_k \) the trees returned by the algorithm with \( \ell(S_0) \geq B, \ell(S_k) \leq B \) that differ in \( |S_k \Delta S_0| := |S_k \setminus S_0| + |S_0 \setminus S_k| = 2k \) edges.

- Let \( e_0 \in S_0 \) be edge maximizing \( c'(e) := c(e) + z^* \cdot \ell(e) \). There is an edge \( e_1 \in S_k \setminus S_0 \) such that \( S_1 := S_0 \setminus \{e_0\} \cup \{e_1\} \) is a spanning tree. Since \( c'(S_0) = c'(S_k) \), \( c'(e_0) \geq c'(e_1) \). On the other hand \( c'(S_1) \geq c'(S_0) \) since \( S_0 \) has minimal \( c' \)-cost. Hence \( c'(S_1) = c'(S_0) \) and \( |S_1 \Delta S_0| = 2(k - 1) \).

- We iterate this to obtain \( S_0, \ldots, S_k \) with \( c'(S_0) = c'(S_1) = \ldots = c'(S_k) \) and \( |S_i \Delta S_{i+1}| = 2 \forall i \).

- Since \( \ell(S_0) \geq B, \ell(S_k) \leq B \) there must be a pair \( (T_1, T_2) := (S_i, S_{i+1}) \) with \( \ell(S_i) \geq B, \ell(S_{i+1}) \leq B \).
$T_2$ is not that bad

Lemma

Let $z^*, T_1, T_2$ be opt. Lagrange solutions, $\ell(T_1) \geq B, \ell(T_2) \leq B$ s.t. $|T_1 \Delta T_2| = 2$. Then $c(T_2) \leq OPT + c_{\text{max}}$.

- Recall that

\[
c(T_1) \leq c(T_1) + \underbrace{z^* \cdot (\ell(T_1) - B)}_{\geq 0} = OPT_{LR}(z^*) \leq OPT
\]

- Let $e_1, e_2$ be edges with $T_2 = (T_1 \setminus \{e_1\}) \cup \{e_2\}$. Then

\[
c(T_2) = c(T_1) - c(e_1) + c(e_2) \leq OPT + c_{\text{max}} \quad \square
\]
A PTAS

Lemma

There is a PTAS for Budgeted Spanning Tree.

- Guess the $1/\varepsilon$ many edges of maximum cost in the optimum solution.
- Contract them. Now $c_{\text{max}} \leq \varepsilon \cdot OPT$ in the remaining instance.

State of the art:

- It is not know, whether there is an FPTAS for Budgeted Spanning Tree.
- [Hong et al.] can find a tree $T$ with $c(T) \leq (1 + \varepsilon)OPT$, $\ell(T) \leq (1 + \varepsilon)B$ in $\text{poly}(n, 1/\varepsilon)$ (i.e. a bicriteria FPTAS).
Part 27
$k$-Median

Source: Approximation Algorithms (Vazirani, Springer Press)
**Problem: $k$-Median**

- **Given:** Facilities $F$, cities $C$, parameter $k \in \mathbb{N}$. Metric cost $c_{ij}$ for connecting city $j$ to facility $i$.
- **Find:** Set of at most $k$ facilities $I$ and an assignment $\phi : C \rightarrow I$ of cities to opened facilities, minimizing the connection cost:

$$OPT := \min_{I \subseteq F, |I| \leq k, \phi : C \rightarrow I} \sum_{i \in I} c_{\phi(j),i}$$
$k$-Median

**Problem: $k$-Median**

- **Given:** Facilities $F$, cities $C$, parameter $k \in \mathbb{N}$. Metric cost $c_{ij}$ for connecting city $j$ to facility $i$.
- **Find:** Set of at most $k$ facilities $I$ and an assignment $\phi : C \rightarrow I$ of cities to opened facilities, minimizing the connection cost:

\[
OPT := \min_{I \subseteq F, |I| \leq k, \phi : C \rightarrow I} \sum_{i \in I} c_{\phi(j),i}
\]
Integer program:

\[
\begin{align*}
\text{min} & \quad \sum_{i \in F} \sum_{j \in C} x_{ij} c_{ij} \\
\sum_{i \in F} x_{ij} &= 1 \quad \forall j \in C \\
x_{ij} &\leq y_i \quad \forall i \in F \forall j \in C \\
\sum_{i \in F} y_i &\leq k \\
y_i, x_{ij} &\in \{0, 1\} \quad \forall i \in F \forall j \in C
\end{align*}
\]

\[= OPT\]

Lagrangian Relaxation \((z \geq 0)\):

\[
\begin{align*}
\text{min} & \quad \sum_{i \in F} \sum_{j \in C} x_{ij} c_{ij} + z \cdot \left( \sum_{i \in F} y_i - k \right) \\
\sum_{i \in F} x_{ij} &= 1 \quad \forall j \in C \\
x_{ij} &\leq y_i \quad \forall i \in F \forall j \in C \\
y_i, x_{ij} &\in \{0, 1\} \quad \forall i \in F \forall j \in C
\end{align*}
\]

\[=: OPT_{LR}(z)\]

optimum facility location value for instance with \(f_i := z\)

\[= -zk\]
Approximating the Lagrangean Relaxation (1)

Recall the previous result:

**Theorem**

One can compute a Facility Location solution in poly-time, with

\[
\text{connection cost} + 3 \cdot \text{facility cost} \leq 3 \cdot \text{OPT}_{FL}.
\]

- Let \( FL(z) \subseteq F \) be the set of facilities, opened by approximation algorithm if \( f_i := z \) for all facilities \( i \in F \).
- For \( F' \subseteq F \) and \( j \in C \) let \( c(F', j) := \min_{i \in F'} \{c_{ij}\} \) be the distance of city \( j \) to nearest facility in \( F' \).
- Let \( c(F') := \sum_{j \in C} c(F', j) \) be the connection cost of a Facility Location or \( k\text{-Median} \) solution \( F' \).
Approximating the Lagrangean Relaxation (2)

\[ |FL(\varepsilon)| = \# \text{ facilities opened by apx-algo} \]

\[ |F_2| \]

sol \( F_2 \): infeasible / \( c(F_2) \leq 3 \cdot OPT \)

\[ k \]

sol \( F_1 \): feasible / \( c(F_1) \geq 3 \cdot OPT \)

\[ |F_1| \]

\[ z^* \]

\[ |FL(0)| = |F| \geq k, \lim_{z \to \infty} |FL(z)| = 1 \leq k \]

\[ \text{By binary search in the interval } [0, |C| \cdot \max_{i,j} \{ c_{ij} \}], \text{ find } z^* \geq 0, \text{ where } |FL(z^*)| \geq k \geq |FL(z^* + \varepsilon)| \]

\[ \text{Let } F_1 := FL(z^* + \varepsilon), \ F_2 := FL(z^*) \text{ be the obtained approximate solutions (we ignore the } \varepsilon\text{-term from now on, since it can be made exponentially small).} \]
Bounding the cost of $F_1, F_2$

Lemma

Choose $0 \leq \lambda \leq 1$ with $\lambda|F_1| + (1 - \lambda)|F_2| = k$. Then

$$\lambda \cdot c(F_1) + (1 - \lambda) \cdot c(F_2) \leq 3 \cdot OPT.$$  

- Since we use a $(3, 1)$-apx algo for FACILITY LOCATION:

  $$c(F_1) + 3z \cdot |F_1| \leq 3 \cdot OPT_{FL}(z)$$

  $$c(F_2) + 3z \cdot |F_2| \leq 3 \cdot OPT_{FL}(z)$$

- Adding both inequalities with coefficient $\lambda$ and $1 - \lambda$, resp.:

  $$\lambda c(F_1) + (1 - \lambda)c(F_2) + 3z \cdot (\lambda|F_1| + (1 - \lambda)|F_2|)$$

  $$= k$$

  $$\leq 3 \cdot OPT_{FL}(z) = 3 \cdot OPT_{LR}(z) + 3z \cdot k$$

- The $3zk$ term cancels out and

  $$\lambda c(F_1) + (1 - \lambda)c(F_2) \leq 3 \cdot OPT_{LR}(z) \leq 3 \cdot OPT \quad \square$$
Combining $F_1$ and $F_2$ (1)

**Lemma**

We can randomly choose a subset $I \subseteq F_1 \cup F_2$ of size $|I| \leq k$ of cost $E[c(I)] \leq 6 \cdot OPT$.

- We want to choose $I$ s.t.

\[
E[c(I, j)] \leq 2 \cdot (\lambda \cdot c(F_1, j) + (1 - \lambda) \cdot c(F_2, j)).
\]

Then

\[
E[c(I)] = \sum_{j \in C} E[c(I, j)] \leq \sum_{j \in C} 2(\lambda \cdot c(F_1, j) + (1 - \lambda) \cdot c(F_2, j))
\]

\[
\leq 2 \cdot (\lambda \cdot c(F_1) + (1 - \lambda) \cdot c(F_2)) \leq 6 \cdot OPT \leq 3 \cdot OPT
\]

288 / 292
Combining $F_1$ and $F_2$ (2)

Case (1): With prob $1 - \lambda$:

- Choose $F'_2 \subseteq F_2$ with $|F_1|$
- $|F'_2| = |F_1|$ so that for
- any facility $i_1 \in F_1$, also
- the facility $i_2 \in F_2$
- minimizing $c_{i_1,i_2}$ is in $F'_2$

- Choose $F''_2 \subseteq F_2 \setminus F'_2$ with $|F''_2| = k - |F_1|$ uniformly at random. Open $I := F'_2 \cup F''_2$.

- Let $i_1 \in F_1$ and $i_3 \in F_2$ be nearest facilities to $j$. Suppose $i_3 \notin F'_2$ (other case later).

- Note that $\Pr[i_3 \in I] = \frac{k - |F_1|}{|F_2| - |F_1|} = 1 - \lambda$. Hence

$$E[c(I, j)] \leq \Pr[i_3 \in I] \cdot c(i_3, j) + \Pr[i_3 \notin I] \cdot \frac{c(i_2, j)}{\lambda} \leq (1 - \lambda + \lambda) \cdot c(F_2, j) + 2\lambda \cdot c(F_1, j) \leq c(F_2, j) + 2\lambda \cdot c(F_1, j)$$
Combining $F_1$ and $F_2$ (2)

Case (2): With prob $\lambda$:

- Choose $I := F_1 \cup F'_2$
- Then

\[
E[c(I, j)] \leq \Pr[i_3 \in I] \cdot c(i_3, j) + \Pr[i_3 \notin I] \cdot c(i_1, j)
\]

\[
= (1-\lambda) \cdot c(F_2, j) + \lambda \cdot c(F_1, j)
\]

\[
\leq \lambda c(F_1, j) + (1 - \lambda) c(F_2, j)
\]
Combining $F_1$ and $F_2$ (3)

- Overall:

\[
E[c(I, j)] \\
\leq \Pr[\text{case (1)}] \cdot E[c(I, j) \text{ in (1)]} + \Pr[\text{case (2)}] \cdot E[c(I, j) \text{ in (2)]}
\]

\[
= (1 - \lambda) \cdot 2\lambda c(F_1, j) + c(F_2, j) + \lambda \cdot \lambda c(F_1, j) + (1 - \lambda) \cdot (1 + \lambda) \cdot c(F_2, j) \\
\leq 2(1 - \lambda) c(F_1, j) + (1 - \lambda) c(F_2, j) \leq 2
\]

- (For case $i_3 \in F'_2$: $E[c(I, j)] \leq \lambda c(F_1, j) + (1 - \lambda) c(F_2, j)$).
The main result

Theorem

There is an expected 6-approximation for $k$-MEDIAN in polynomial time (which can be easily derandomized).

State of the art:

Theorem (Arya et al.)

One can obtain a $(3 + \varepsilon)$-apx in time $O(n^{2/\varepsilon})$.

- Algorithm uses local search.
- The natural LP relaxation has an integrality gap of 3, but no algorithm is known that achieves this value.