UNIVERSIDADE DE CAMPINAS - UNICAMP
INSTITUTO DE COMPUTAÇÃO - IC

Algoritmos de Aproximação

Flávio Keidi Miyazawa

Campinas-SP

Contents I

Contents II

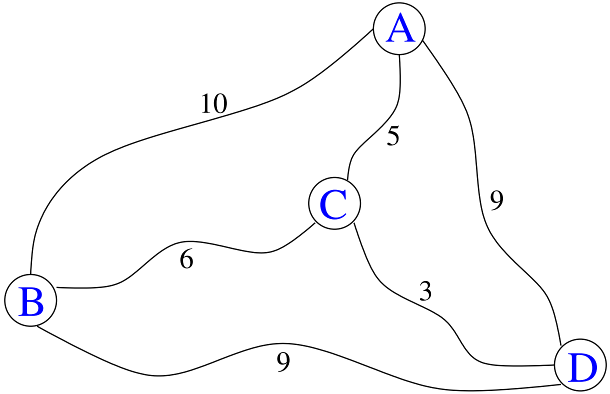
Combinatorial Optimization Problems

- Finite Domain (enumerable)
- Optimization function: costs, lengths, quantities
- Objective: minimization or maximization

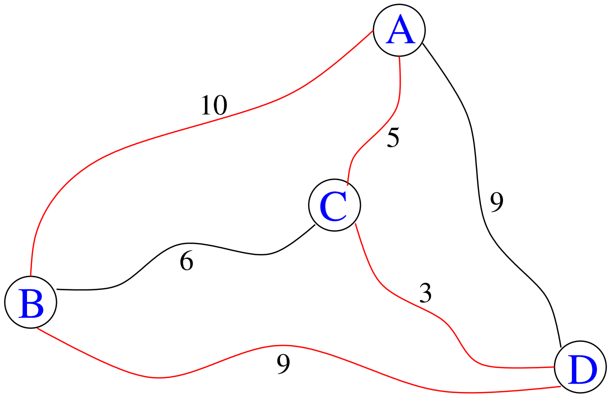
Example: Traveling Salesman Problem (TSP)

- *Input:*
  - Non-oriented graph: $G = (V, E)$,
  - cost in the edges: $c_e \geq 0, \forall e \in E$.
- *Objective:*
  Find a *tour* (hamiltonian cycle) of minimum cost that visits each node exactly once.

Find a hamiltonian cycle with minimum cost



hamiltonian cycle of minimum cost: 27



Naive algorithm for TSP: Try all possible tours.
Complexity: $O(n!)$, where $n = |V|$.

Good algorithm = polynomial time algorithm
(Cobham'64&Edmonds'65)
Probably there will not exists efficient algorithms for TSP
Other hard problems:

- Frequency assignment in cellular phones
- Packing of objects into containers
- Scheduling of workers/people
- Scheduling of tasks in computers
- Object classification
- Map coloring
- Computer network design
- Many others...

Computational Complexity

Decision Problems

- Example: Given a graph G, do there exists a hamiltonian cycle in G ?
- Example: Given a complete graph $G = (V, E)$, with edge weights $c : E \to \mathbb{Z}^+$ and a positive integer K, do there exists a hamiltonian cycle in G, of cost at most K ?
- Example: Given a map M and an integer K, can I color M with K colors without conflict ?
- Example: Given a complete graph $G = (V, E)$, weight in the edges $c : E \to \mathbb{Z}^+$, nodes s and t and a positibe integer K, do there exists a path in G, from s to t, of weight at most K ?

Polynomial time reduction between problems

P_1 is polynomially reducible to P_2 ($P_1 \preceq P_2$) if

- $\exists$ *Ti* transforms instance I_1 of P_1 to instance I_2 of P_2
- $\exists$ *Ts* tranforms solution S_2 of I_2 to solution S_1 of I_1
- *Ti* and *Ts* have polynomial time complexity

$$\begin{tikzcd}
\text{Instância } I1 \arrow[r, bend left, "I2:=Ti(I1)"] & I2 \arrow[d, "S2:=A2(I2)"] \\
\text{Solução } S1 & S2 \arrow[l, bend left, "S1:=Ts(S2)"]
\end{tikzcd}$$

Consequences:

If P_2 is “polynomial” then P_1 is “polinomial”.
If P_1 is “exponential” then P_2 is at least “exponential”.

Polynomial time reduction between decision problems

P_1 is polynomial time reducible to P_2 ($P_1 \preceq P_2$) if

- $\exists$ T that tranforms instance I_1 of P_1 to instance I_2 of P_2
- I_1 has solution if and only if I_2 has solution
- T is "polynomial"

Consequences:

If P_2 is "polynomial" then P_1 is "polynomial".
If P_1 is "exponential" then P_2 is at least "exponential".

Complexity classes:

P, NP, NP-Complete, NP-Hard

$X \in \mathrm{P}$:

X can be decided in polynomial time.

$X \in \mathrm{NP}$:

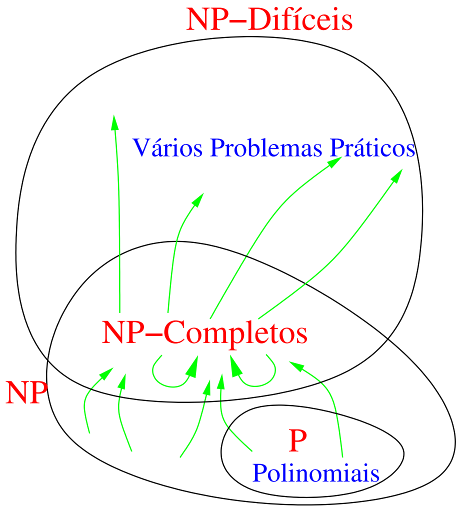
X has "short" certificate verifiable in polynomial time.

$X \in \mathrm{NP\text{-}Complete}$:

$X \in \mathrm{NP}$ and $\forall Y \in NP$, $Y \preceq X$.

$X \in \mathrm{NP\text{-}hard}$:

$\forall Y \in \mathrm{NP}$, $Y \preceq X$, where X does not necessarily belongs to NP.



Conjecture: P=NP ?

US$ 1 million prize.

Problems in **P**

- Given a set of integers S, do there exists $n \in S$ such that $n = \sum_{i \in S \setminus \{n\}} i$?
- Is a graph G connected ?
- Given a complete graph $G = (V, E)$, weight on the edges $c : E \to \mathbb{Z}^+$, nodes s and t and a positive integer K, do there exists a path in G, from s to t, of cost at most K ?
- Can a map be colored with 4 colors without conflicts ?

NP-complete problems

- Given a set of integers S, do there exists $N \subseteq S$ such that $\sum_{i \in N} i = \sum_{i \in S \setminus N} i$?
- Given a graph G, do there is a hamiltonian cycle in G ?
- Given a complete graph $G = (V, E)$, weight in the edges $c : E \rightarrow \mathbb{Z}^*$ and a positive integer K, do there exists a hamiltonian cycle in G, of weight at most K ?
- Given a complete graph $G = (V, E)$, weight in the edges $c : E \rightarrow \mathbb{Z}$, nodes s and t and a positive integer K, do there exists a path in G, from s to t, of weight at most K ?
- Can I color a map with 3 colors without conflicts ?

NP-hard problems

- Several practical problems are NP-hard
  Examples:
  - Frequency assignment in cellular phones
  - Packing of objects into containers
  - Scheduling of workers/people
  - Scheduling of tasks in computers
  - Object classification
  - Map coloring
  - Computer network design
  - Many others...
- **P$\neq$NP $\Rightarrow$ there is no efficient algorithmf for NP-hard problems**

Comparing polynomial and exponential times

$f(n)$	$n = 20$	$n = 40$	$n = 60$	$n = 80$	$n = 100$
n	$2{,}0\times10^{-11}$seg	$4{,}0\times10^{-11}$seg	$6{,}0\times10^{-11}$seg	$8{,}0\times10^{-11}$seg	$1{,}0\times10^{-10}$seg
n^2	$4{,}0\times10^{-10}$seg	$1{,}6\times10^{-9}$seg	$3{,}6\times10^{-9}$seg	$6{,}4\times10^{-9}$seg	$1{,}0\times10^{-8}$seg
n^3	$8{,}0\times10^{-9}$seg	$6{,}4\times10^{-8}$seg	$2{,}2\times10^{-7}$seg	$5{,}1\times10^{-7}$seg	$1{,}0\times10^{-6}$seg
n^5	$2{,}2\times10^{-6}$seg	$1{,}0\times10^{-4}$seg	$7{,}8\times10^{-4}$seg	$3{,}3\times10^{-3}$seg	$1{,}0\times10^{-2}$seg
2^n	$1{,}0\times10^{-6}$seg	$1{,}0$seg	$13{,}3$dias	$1{,}3\times10^{5}$séc	$1{,}4\times10^{11}$séc
3^n	$3{,}4\times10^{-3}$seg	$140{,}7$dias	$1{,}3\times10^{7}$séc	$1{,}7\times10^{19}$séc	$5{,}9\times10^{28}$séc

Considering a 1 Terahertz computer (one thousand faster than a 1 Gigahertz computer).

Comparing polynomial and exponential times

$f(n)$	Current computer	100×faster	1000×faster
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 4.19$	$N_6 + 6.29$

Fixing execution time

Approximation Algorithms

- **Approach to deal with NP-hard problems**
  – NP-hard problems need solutions
- **Fast algorithms (polynomial time algorithms)**
  – In general approximation algorithms are fast
  – adequate for large instances where exact algorithms are prohib
- **Guarantee of proximity to an optimal solution**
  – Formal analysis about the generated solutions
- **New complexity classes in NP-hard**
  – NP-complete problems are polynomially equivalent, but
  – optimization problems present many different complexities
  under the approximation approach
- **Aggregate theoretical value to algorithms/heuristics**
  – Academic area values formal analysis

- **Inspiration to the development of heuristics**
  – Heuristics take advantage of the combinatorial structures used in approximation algorithms
- **Development of recent theories**
  – Large attention academic community
  – New techniques in the development of algorithms
  – new theories about inapproximability of problems
- **Development of exact algorithms**
  – Exact algorithms use feasible solutions to bound search
  – Most exact algorithms are based in enumeration
- **Algorithmic game theory**
  – similar analysis/measures to find *price of anarchy*
- **Analysis of online algorithms**
  – algorithms without information about further events

Computational experiments

- **Williamson'93**
  Matching problem: Graphs up ot 131.000 nodes, within 2% of optimum
- **Goemans & Willamson'94**
  Max-Cut problem: Graphs with up to 200 nodes, within 4% of the optimum
- **Homer & Peinado'94**
  Max-Cut problem: Graphs with up to 13.000 nodes distributed implementation of Goemans & Williamson algorithm
- **Hall'95**
  Steiner tree problem: Graphs with up to 1000 nodes / 60.000 edges within 7% of optimum

- **Hu & Wein**
  Steiner Forest: Graphs with up to 64 nodes, within 5% of optimum
- **Johnson, Minkoff, Phillips'00**
  Steiner tree problem with penalties: graphs with up to 76.000 nodes. Close to optimum solutions
- **Barahona & Chudak'99**
  Facility Location Problem: hybrid technique in graphs with 3000 facilities and 3000 clients, within 1% of optimum
- **Mihail, Mostrel, Dean & Shallcross'95**
  Survivable Network Design Problem: Bellcore software. Solutions close to the optimum.

Great challenges

- VLSI circuit design instances
  Johnson'90: TSP graphs with 1.2 million nodes (and increasing)
  Other large instances occur in crystallography
- **expensive bounds**
  For many instances it is impossible to solve large linear/semidefinite programs in reasonable time (many used to develop approximation algorithms)
- **Real applications have large instances**
  Currently, solved by parts.
  Examples: VLSI circuit design, scheduling of workers, facility locations...

Main techniques

- Combinatorial methods
- Dynamic programming
- Methods based on linear programming
- Probabilistic methods
- Semidefinite programming
- inapproximability techniques

Some problems

- Task scheduling problem
- Set cover problem
- Knapsack problem
- Bin packing
- Traveling salesman problem
- Steiner forest
- Maximum satisfiability
- Maximum cut
- Maximum clique
- Facility Location

Measures of approximation

- Absolute approximation
- Approximation factor
- Asymptotic approximation factor
- Approximation factor for probabilistic algorithms

Absolute Approximation

Given an algorithm A and an instance I,

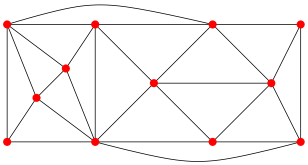
- $\mathrm{OPT}(I)$ is the value of an optimum solution
- $A(I)$ is the value of a solution generated by A over I

A has *absolute approximation k* if

$$|A(I) - \mathrm{OPT}(I)| \leq k \qquad \forall \text{ instance } I.$$

Coloring of planar graphs

Def.: *A graph $G = (V, E)$ is planar if there is an immersion of the graph in the plane so that two edges can only intersect on the nodes they are incident.*



Problema PLANAR-COLORING: Given a simple planar graph $G = (V, E)$, color the nodes of G with the minimum number of colors, in such a way that adjacent nodes have distinct colors.

Theorem: PLANAR-COLORING is NP-hard.

Theorem: *There exists a 1-absolute approximation for the* PLANAR-COLORING *(Appel & Haken'76)*

A 3-absolute approximation

Idea of the algorithm: Induction based on the following theorem

Theorem: *If G is simple and planar, then G has at least one node with degree at most 5.*

Proof. Euler theorem for connected planar graphs says:

$$|E| = |V| + |F| - 2,$$

where F is the number of faces of a immersion of G in the plane. Proof: Exercise.

Using this theorem, we can prove that

$$|E| \leq 3|V| - 6 \quad \text{(Exercise)}$$

Using the relation of number of edges and the total sum of node degrees, the result follows. (Exercise). □

ALGORITHM 6-COLORS $G = (V, E)$

```
1   If $E = \emptyset$ then,
2       $color(v) := 1, \quad \forall v \in V$.
3   else, if $G$ is bipartite, $V = (X, Y)$ then
4       $color(x) := 1, \forall x \in X$ and $color(y) := 2, \forall y \in Y$.
5   else,
6       let $v \in V$ such that $degree(v) \leq 5$
7       Paint $G' := G - v$ recursively.
8       Let $c \in \{1, \ldots, 6\}$ a color not used by adjacent nodes of $v$
9       $color(v) := c$.
```

Theorem: *6-Colors is a 3-absolute approximation.*

Proof. If $E = \emptyset$ or G is bipartite,

$$\text{6-Colors}(G) = \mathrm{OPT}(G)$$

otherwise, $\mathrm{OPT}(G) \geq 3$

$$\text{6-Colors}(G) - \mathrm{OPT}(G) \leq 3.$$

□

A 2-absolute approximation

Theorem: *Describe an algorithm 5-Colors that is a 2-absolute approximation.*

Proof. Exercise.
Idea: Based on the algorithm 6-Colors. Let v the removed node of degree at most 5.
• If the neighbours of v use ≤ 4 cores, paint v with one of remaining colors for it.
• Otherwise, consider an immersion of G in the plane.
W.L.G. (Without Loss of Generality) let $1, \ldots, 5$ the neighbours of v, numbered from the horary sense around v and let i and j be two non-consecutive neighbours (in the acyclic order) of v.

Try to recolor i with the color of j changing the two colors from i. If possible, we obtain a new color available for v. Otherwise, repaint other nodes. $\square$

Edge Coloring

Problem EDGE COLORING: Given a graph G, color the edges of G with the minimum number of colors, where the adjacent edges must have distinct colors.

Applications: Scheduling, coloring of circuit wires, etc.

Theorem: *(Holyer'81) The Edge Coloring problem is NP-hard.*

Theorem: *(Vizing'64) It is possible to color the edges of G with* $\Delta(G) + 1$ *colors, where* $\Delta(G)$ *is the largest degree of a node in G.*

Corollary: *There exists an algorithm with absolute approximation 1 for the Edge Coloring problem.*

Negative Results: Scale sensibility

Knapsack problem

Knapsack Problem: Given set of items $S = \{1, \ldots, n\}$, each item i with integer weight v_i and integer size s_i, $i = 1, \ldots, n$, and integer B, find $S' \subseteq S$ that maximizes $\sum_{i \in S'} v_i$ such that $\sum_{i \in S'} s_i \leq B$.

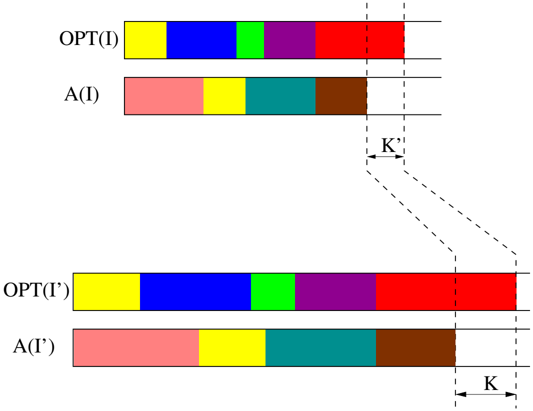
Theorem: *If $P \neq NP$ then there is no polynomial time algorithm with absolute approximation k for the Knapsack problem, for any fixed k.*

Proof. Suppose there exists algorithm A and k such that

$$|\mathrm{OPT}(I) - A(I)| \leq k,$$

where k is bounded by a polynomial in n.

Let $I := \{S = (1, \ldots, n), v = (v_1, \ldots, v_n), (s_1, \ldots, s_n), B\}$ an instance.
Let $I' := \{S = (1, \ldots, n), v = (v'_1, \ldots, v'_n), (s_1, \ldots, s_n), B\}$ an instance where $v'_i := (k + 1) \cdot v_i$.



$$(k+1)\mathrm{OPT}(I) = \mathrm{OPT}(I')$$

Let *Sol* a solution of I corresponding the one generated by A for I'.

$$|\mathrm{OPT}(I') - A(I')| \le k$$

$\Longrightarrow$

$$|(k+1)\cdot\mathrm{OPT}(I) - (k+1)Sol| \le k$$

$\Longrightarrow$

$$(k+1)\cdot|\mathrm{OPT}(I) - Sol| \le k.$$

$\Longrightarrow$

$$|\mathrm{OPT}(I) - Sol| \le \frac{k}{k+1}.$$

$\Longrightarrow$

$$|\mathrm{OPT}(I) - Sol| \le 0.$$

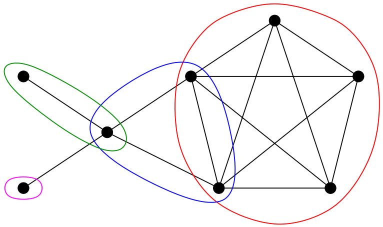
$\Longrightarrow$

$$\mathrm{OPT}(I) = Sol$$

$\square$

Maximum Clique Problem

Def.: *Given a graph $G = (V, E)$, a set $S \subseteq V$ is a clique if $\forall u, v \in S \Rightarrow \{u, v\} \in E$.*



Clique Problem: Given a graph G, find a clique in G of maximum cardinality.

Theorem: *(Karp'72) Clique is an NP-hard problem.*

Absolute inapproximability of Maximum Clique Problem

Def.: *Let G^m a graph with m copies of G adding all edges between nodes of distinct copies.*

Fato: *If the largest clique of G has size k then, the largest clique of G^m has size $k \cdot m$.*

Theorem: *If $P \neq NP$ then there does not exists a polynomial time algorithm with absolute approximation k for the Maximum Clique Problem, for any value of k bounded by a polynomial in n.*

Proof.

Suppose there exists an algorithm A and positive integer k, bounded by a polynomial in n, such that

$$|\mathrm{OPT}(G) - A(G)| \leq k \qquad \forall \text{ instance } I$$

Then

$$\begin{aligned} |\mathrm{OPT}(G^{k+1}) - A(G^{k+1})| &\leq k \\ |(k+1)\cdot \mathrm{OPT}(G) - A(G^{k+1})| &\leq k \end{aligned}$$

Let *Sol* the largest sub-clique in a copy of the clique produced by $A(G^{k+1})$.

Sol has size at least $\frac{A(G^{k+1})}{k+1}$.

$\Longrightarrow$

$$|(k+1)\cdot \mathrm{OPT}(G) - (k+1)Sol| \leq k.$$

$\Longrightarrow$

$$|(k+1)(\mathrm{OPT}(G) - Sol)| \leq k.$$

$\Longrightarrow$

$$|\mathrm{OPT}(G) - Sol| \leq \frac{k}{k+1}.$$

$\Longrightarrow$

$$|\mathrm{OPT}(G) - Sol| \leq 0.$$

$\Longrightarrow$

$$Sol = \mathrm{OPT}(G)$$

Approximation Factor

Given an algorithm A and instance I,

- OPT(I) is the value of an optimum solution
- $A(I)$ is the value of the solution produced by A
- A is an approximation algorithm with factor α if

$$A(I) \leq \alpha \cdot \mathrm{OPT}(I) \qquad \forall I, \quad \text{for minimization}$$

$$A(I) \geq \alpha \cdot \mathrm{OPT}(I) \qquad \forall I, \quad \text{for maximization}$$

- A has a tight factor α if $\forall \epsilon > 0$ we have

$$\exists I \quad \text{such that} \quad A(I) > (\alpha - \epsilon)\mathrm{OPT}(I), \quad \text{for minimization}$$

$$\exists I \quad \text{such that} \quad A(I) < (\alpha + \epsilon)\mathrm{OPT}(I), \quad \text{for maximization}$$

Task scheduling

Scheduling problem: Given a list of tasks $L = (J_1, \ldots, J_n)$, task J_i with time $t(J_i)$ and m identical machines, find a partition of L, $(M_1, \ldots, M_m)$, such that $\max_i t(M_i)$ is minimum.

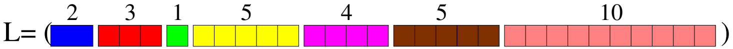
Theorem: SCHEDULING is NP-hard.

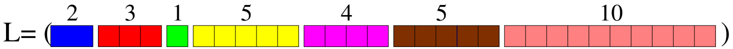
Graham Algorithm: Schedule the next task in a less loaded machine.

GRAHAM (m, n, t)

```
1    for $j$ from 1 to $m$ do $M_j \leftarrow \emptyset$
2    for $i$ from 1 to $n$ do
3        let $k$ a machine such that $\sum_{i \in M_k} t_i$ is minimum
4        $M_k \leftarrow M_k \cup \{i\}$
5    return $\{M_1, \ldots, M_m\}$
```

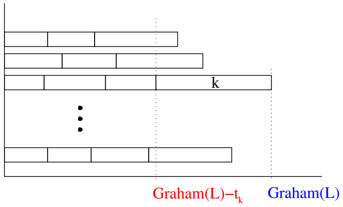
Example:





Theorem: $\mathit{Graham}(L) \leq 2\,\mathrm{OPT}(L)$.

Proof. Let J_k the last executed task in the scheduling.



$$\begin{aligned} \mathrm{OPT}(L) &\geq \max_i t(J_i) &\geq& \; t(J_k) \\ \mathrm{OPT}(L) &\geq \frac{1}{m}\sum_i t(J_i) &\geq& \; \mathrm{Graham}(L) - t(J_k) \end{aligned}$$

I.e., $2 \cdot \mathrm{OPT}(m,n,t) \geq \mathrm{Graham}(L)$. $\square$

Exercício: O fator de aproximação do algoritmo Graham(L) pode ser melhorado para $2 - 1/m$.

Exercício: O fator de aproximação do algoritmo Graham(L) é justo.

Algorithm LPT - Longest Processing Time

```
LPT (m, n, t)
1    Sort tasks such that $t_1 \geq t_2 \geq \ldots \geq t_n$.
2    return Graham (m, n, t)
```

Theorem: $LPT(m,n,t) \;\leq\; \frac{3}{2}\,\mathrm{OPT}(m,n,t)$.

Proof.

Let k the last task executed in the LPT scheduling.

If $n \leq m$ or $t_k =$LPT(m,n,t) then LPT$(m,n,t) = \mathrm{OPT}(m,n,t)$.

Else, we have $\mathrm{OPT}(m,n,t) \geq 2t_k$. I.e., $t_k \leq \frac{\mathrm{OPT}(m,n,t)}{2}$.

We know that: $\mathrm{LPT}(L) \leq \frac{\sum_i t_i}{m} + t_k$, $\quad \frac{\sum_i t_i}{m} \leq \mathrm{OPT}(L)$ and $t_k \leq \frac{\mathrm{OPT}(m,n,t)}{2}$

So, $\mathrm{LPT}(L) \leq \mathrm{OPT}(L) + \frac{\mathrm{OPT}(m,n,t)}{2} = \frac{3}{2}\mathrm{OPT}(m,n,t)$ □

Is this analysis tight ?

Theorem: $LPT(m, n, t) \leq \frac{4}{3} \mathrm{OPT}(m, n, t)$.

Proof. By contradiction. Supose there exists an instance such that $\frac{\mathrm{LPT}(L)}{\mathrm{OPT}(L)} > \frac{4}{3}$. Consider such an instance with smallest number of tasks.

As the instance has minimum number of tasks, t_n is the last executed task.

$$\mathrm{LPT}(L) \leq \frac{\sum_i t_i}{m} + t_n \quad \text{and} \quad \mathrm{OPT}(L) \geq \frac{\sum_i t_i}{m}$$

So, $\mathrm{LPT}(L) \leq \mathrm{OPT}(L) + t_n$.

That is, $\frac{4}{3} < \frac{\mathrm{LPT}(L)}{\mathrm{OPT}(L)} \leq 1 + \frac{t_n}{\mathrm{OPT}(L)}$.

Therefore: $\mathrm{OPT}(L) < 3t_n$.

As t_n is the smallest task, all machines have at most 2 tasks. In this case, we can conclude that the obtained LPT scheduling is optimum (exercise), leading to a contradiction. □

Lema: *The factor $\frac{4}{3}$ for algorithm LPT is tight.*

Proof. Exercise.

Idea: Consider $m = 4$ machines and $n = 9$ tasks, where $t_1 = t_2 = 7, \quad t_3 = t_4 = 6, \quad t_5 = t_6 = 5, \quad t_7 = t_8 = 4,$ and $t_9 = 4$. Extrapolate the instance for large values of m. □

Vertex Cover

Vertex Cover Problem: Given graph $G = (V, E)$, find $C \subseteq V$ such that $\{u, v\} \cap C \neq \emptyset \quad \forall \{u, v\} \in E$ and $|C|$ is minimum.

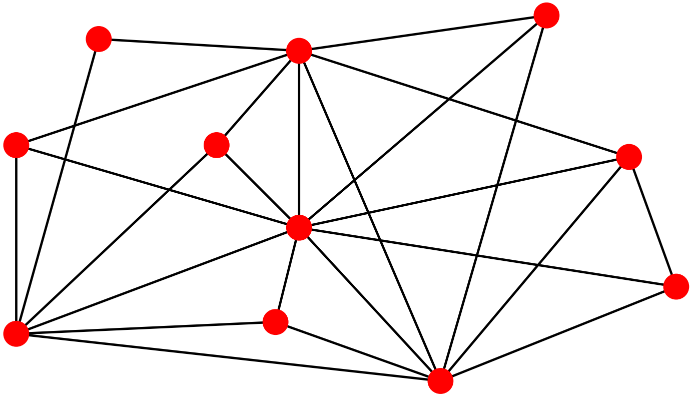
Applications: Given museum, dispose the minimum number of guards that cover all rooms of the museum.

Negative Results

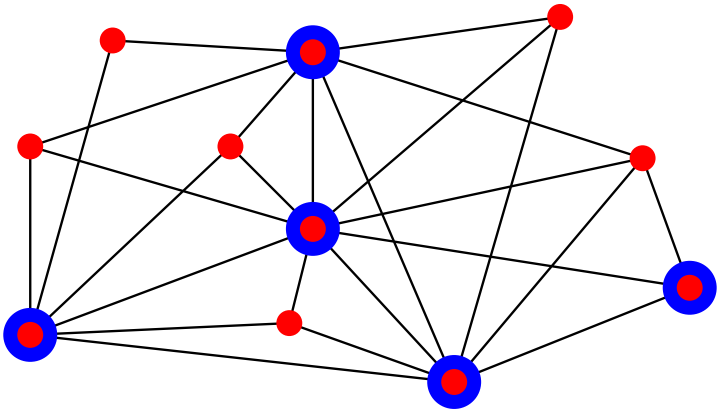
Theorem: VERTEX COVER is NP-hard.

Theorem: If there exists α-approximation for the Vertex Cover problem with $\alpha < 7/6$ then P=NP *(Hastad'97)*.

Find a vertex cover:



Vertex Cover:



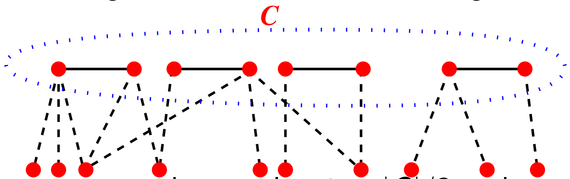
Greedy algorithm by edge

Idea: Find a maximal matching.

```
EDGE-GREEDY-VERTEX-COVER(G)
1   $C \leftarrow \emptyset$
2   while $E \neq \emptyset$
3       choose $(i, j) \in E$
4       $C \leftarrow C \cup \{i, j\}$
5       remove all adjacent edges $i$ and $j$
6   return $C$
```

Theorem: Greedy-Vertex-Cover is 2-approximation *(Gavril).*

Proof. Chosen edges form a maximal matching.



Any vertex cover must have at least $\geq |C|/2$ nodes. $\square$

Set Covering

Def.: Given a collection $\mathcal{S}$ of subsets of E we say that $\mathcal{S}$ *cover* E, or is a *covering* of E, if $\cup_{S \in \mathcal{S}} S = E$.

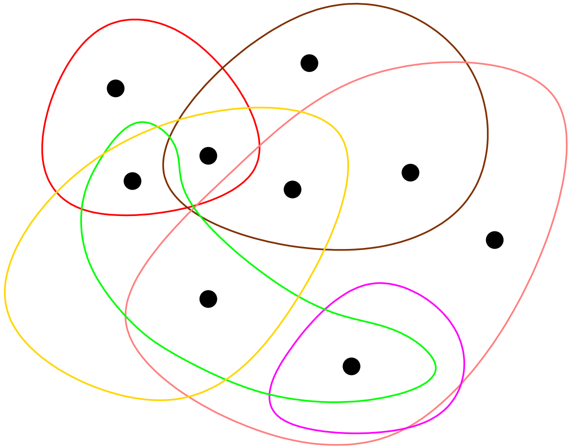
Set Cover Problem: Given set E, subsets $\mathcal{S}$ of E, costs $c(S)$, $S \in \mathcal{S}$, find a covering $\mathcal{S}' \subseteq \mathcal{S}$ that minimizes $c(\mathcal{S}')$.

Negative results

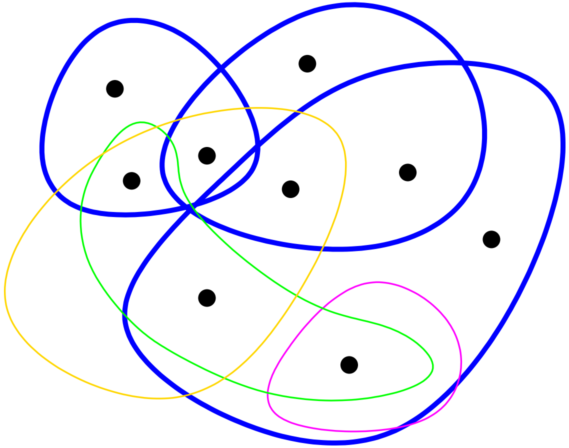
Theorem: SET COVER PROBLEM is NP-Hard.

Theorem: *There is no approximation algorithm with factor* $(1 - \epsilon) \ln |E|$, *for* $\epsilon > 0$, *to the Set Cover Problem, unless* $\mathrm{P} = \mathrm{NP}$ *(Dinur, Steurer'14).*

Find a set covering:



Set covering:



Virus detection

Application *(Williamson/IBM)***:** There exists some sequences to detect each virus, each sequence has 20 or more bytes, in a total of 900 sequences. The objective is to find the minimum number of sequences that detect all 500 virus.

- Sets: Each sequence identify a set of virus.
- Base set: Set of all virus.

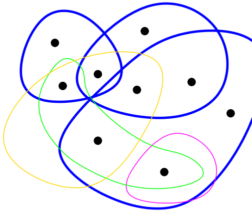
Greedy algorithm: 190 sequences
Lower bound the optimum solution: 185 sequences

Greedy approach:

Select the set with smallest relative weight.

```
MinCC-Chvátal ($E, \mathcal{S}, c$)
1   if $E = \emptyset$
2       then return $\emptyset$
3       else let $Z$ in $\mathcal{S}$ such that $c_Z / |Z \cap E|$ is minimum
4           $E' \leftarrow E \setminus Z$
5           $\mathcal{S}' \leftarrow \{ S \in \mathcal{S} : S \cap E' \neq \emptyset \}$
6           let $c'$ be a restriction of $c$ to $\mathcal{S}'$
7           $\mathcal{T}' \leftarrow$ MinCC-Chvátal ($E', \mathcal{S}', c'$)
8           return $\{Z\} \cup \mathcal{T}'$
```

Lemma: If $Z \in \mathcal{S}$ is such that $c(Z)/|Z|$ is minimum, then
$$\frac{c_Z}{|Z|} \leq \frac{\mathrm{OPT}(E,\mathcal{S},c)}{|E|}$$



Proof.
Let $\mathcal{S}^*$ be an optimum covering.

$$\begin{aligned}
\frac{c_Z}{|Z|}|E| &\leq \frac{c_Z}{|Z|}\sum_{S\in\mathcal{S}^*}|S| \\
&\leq \sum_{S\in\mathcal{S}^*}\frac{c_S}{|S|}|S| \\
&= \sum_{S\in\mathcal{S}^*} c_S \\
&= c(\mathcal{S}^*) \\
&= \mathrm{OPT}(E,\mathcal{S},c)\,.
\end{aligned}$$

$\square$

Theorem: Algorithm MinCC-Chvátal is an H_n-approximation to the MinCC $(E, \mathcal{S}, c)$, where $n := |E|$ and

$$H_n := 1 + \tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{n}\,.$$

Proof. By induction in $|E|$.

If $|E| = 0$, the theorem is clearly valid.

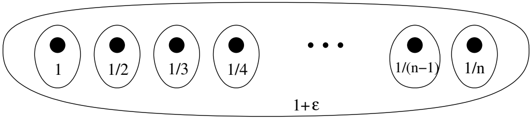
Consider the instance $(E, \mathcal{S}, c)$, $|E| > 0$.

$$\begin{aligned}
c(\{Z\} \cup \mathcal{T}') &= \\
&= c_Z + c'(\mathcal{T}') \\
&\leq \tfrac{|Z|}{n}\,\mathrm{OPT}(E, \mathcal{S}, c) + H_{|E'|}\,\mathrm{OPT}(E', \mathcal{S}', c') \\
&\leq \tfrac{|Z|}{n}\,\mathrm{OPT}(E, \mathcal{S}, c) + H_{n-|Z|}\,\mathrm{OPT}(E, \mathcal{S}, c) \\
&= \left(\frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n} + H_{n-|Z|}\right)\mathrm{OPT}(E, \mathcal{S}, c) \\
&\leq \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{n-|Z|+1} + H_{n-|Z|}\right)\mathrm{OPT}(E, \mathcal{S}, c) \\
&= H_n\,\mathrm{OPT}(E, \mathcal{S}, c)\,.
\end{aligned}$$

$\Box$

Theorem: The approximation factor of the algorithm MINCC-CHVÁTAL is tight.

Proof.



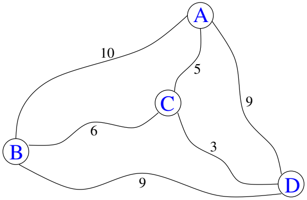
$$
\begin{aligned}
E &:= \{1, \ldots, n\}, \\
\mathcal{S} &:= \{E, \{1\}, \ldots, \{n\}\}, \\
c(E) &:= 1+\epsilon \quad \text{e} \\
c(\{i\}) &:= 1/i \quad \text{para cada } i.
\end{aligned}
$$

Optimum covering has cost $\{E\}$.
$\mathrm{OPT}(E, \mathcal{S}, c) = 1+\epsilon$.

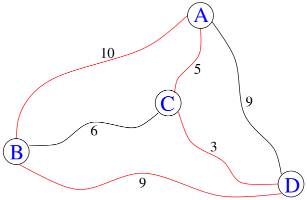
MINCC-CHVÁTAL produces set $\{\{1\}, \ldots, \{n\}\}$.
MINCC-CHVÁTAL$(E, \mathcal{S}, c) = H_n$. □

Traveling Salesman Problem

Problema TSP: Given a graph G and cost function $c_e : E \to \mathbb{Q}_{\geq}$, find a hamiltonian cycle C that miminizes $c(C)$.



Graph G



Hamiltonian Cycle in G of cost 27

Applications:

- Soldering or drilling points in circuit boards.
- Finding minimum routes.

Negative results

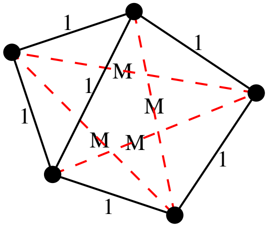
Theorem: Given a graph G, the problem to decide if G has hamiltonian cycle is NP-complete.

Theorem: There is no α-approximation to TSP, for any value α that can be computed in polynomial time, unless P=NP

Proof. Suppose there exists α-approximation for the TSP.

Given graph $G = (V, E)$ let $G' = (V', E', c')$ be such that

- $V' = V$
- $E' = V \times V$
- $c'(e) = \begin{cases} 1 & \text{if } e \in E, \\ \alpha|V| & \text{otherwise} \end{cases}$



If G has hamiltonian cycle, $\mathrm{OPT}(G') = |V|$

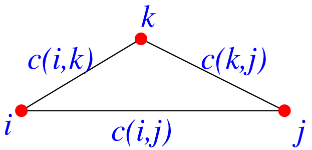
otherwise, $\mathrm{OPT}(G') > \alpha|V|$.

$\Longrightarrow$ we can decide the existence of a hamiltonian cycle in G. $\square$

Metric TSP

Def.: The cost of a graph satisfy triangular inequality if

$$c(i,j) \leq c(i,k) + c(k,j), \quad \forall i,j,k \in V$$



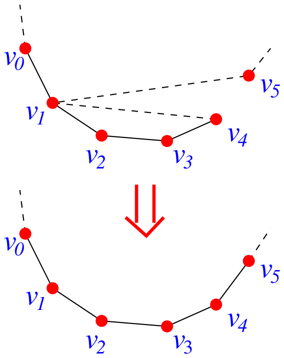
Metric TSP Problem: Given a graph G with cost function c_e in $\mathbb{Q}_{\geq}$ for each edge e, satisfying triangular inequality, find a hamiltonian cycle C with minimum cost.

Strategy:

Find a spanning cycle and perform “shortcuts”.

ALGORITHM SHORTCUT(P),
Find a closed trail $P = (v_0, \ldots, v_m = v_0)$

```
1   $w_0 \leftarrow v_0$
2   $n \leftarrow 0$
3   for $i$ from 1 to $m$ do
4       if $v_i \notin \{w_0, \ldots, w_n\}$
5           then $n \leftarrow n + 1$
6               $w_n \leftarrow v_i$
7   return $\{w_0, \ldots, w_n\}$
```

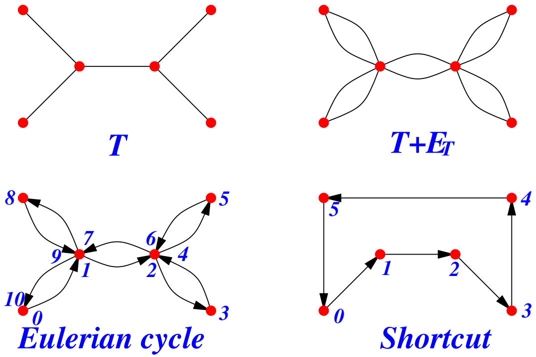


Theorem: *A minimum spanning tree can be found in polynomial time.*

Theorem: *G has eulerian cycle $\Leftrightarrow$ G is connected and degree(v) is even, $\forall v \in V$.*

Theorem: *Finding an eulerian cycle can be done in polynomial time.*

```
TSPM-ROSENKRANTZ-STEARNS-LEWIS ($G, c$)
1    $T \leftarrow$ MINIMUM-SPANNING-TREE ($G, c$)
2    $T' \leftarrow T \dotplus E_T$
3    $P \leftarrow$ EULERIAN-CYCLE ($T'$)
4    $C \leftarrow$ SHORTCUT ($P$)
5    return $C$
```



Theorem: TSPM-*Rosenkrantz-Stearns-Lewis is a 2-approximation.*

Proof.

$$c(T) \leq \mathrm{OPT}(G) \implies c(T + E_T) \leq 2\mathrm{OPT}(G).$$

Shortcuts cannot worsen the value of the solution cost. $\Box$

Christofides algorithm

Strategy:

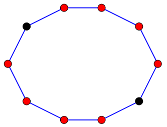
Augment the spanning tree with light edges, in a way to obtain an eulerian cycle.

Theorem: *The problem to find a minimum perfect matching can be done in polynomial time.*

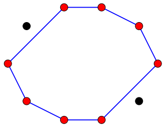
```
TSPM-CHRISTOFIDES ($G, c$)
1    $T \leftarrow$ MINIMUM-SPANNING-TREE ($G, c$)
2    let $I$ be the set of nodes with odd degree in $T$
3    $M \leftarrow$ MINIMUM-PERFECT-MATCHING ($G[I], c$)
4    $T' \leftarrow T \dot{+} M$
5    $P \leftarrow$ EULERIAN-CYCLE ($T'$)
6    $C \leftarrow$ SHORTCUT ($P$)
7    return $C$
```

Lema. $c(M) \leq \frac{1}{2}\mathrm{OPT}(G)$.

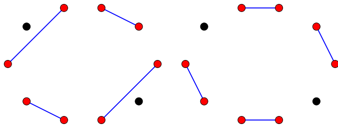
An optimum circuit C.
Odd nodes in red.

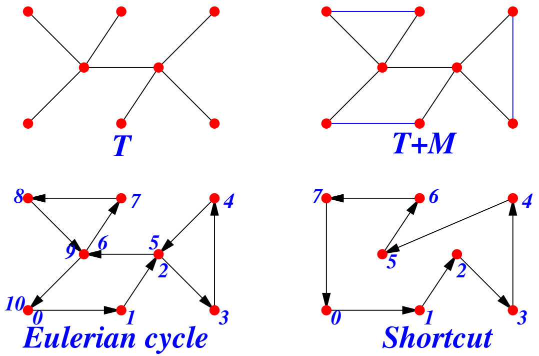


Circuit C' after shortcut on even nodes
We have $c(C') \leq c(C)$



Decomposition of C' into M' and M''
$(c(M')+c(M'')=c(C')\leq c(C)=\mathrm{OPT}(G))$
Therefore,
$c(M) \leq \frac{c(M')+c(M'')}{2} \leq \frac{c(C)}{2} = \frac{\mathrm{OPT}(G)}{2}$





Theorem: TSPM-*Christofides is a $\frac{3}{2}$-approximation.*

Proof. From previous lema: $c(M) \leq \frac{1}{2}\mathrm{OPT}(G)$.

Therefore

$c(C) \leq c(T) + c(M) \leq \mathrm{OPT}(G) + \frac{1}{2}\mathrm{OPT}(G) = \frac{3}{2}\mathrm{OPT}(G)$. □

Approximation Schemes

Def.: *Polynomial Time Approximation Schemes (PTAS):*
Family of algorithms $\{A_\epsilon\}$, $\epsilon > 0$, for a problem such that A_ϵ is a
$(1+\epsilon)$-approximation for minimization problem,
$(1-\epsilon)$-approximation for maximization problem.

Example of complexity times: $O(n^{\frac{1}{\epsilon}})$, $O(n^2 5^{\frac{1}{\epsilon}})$ e $O(\frac{1}{\epsilon}n^3)$.

Def.: *Fully Polynomial Time Approximation Scheme (FPTAS):*
PTAS $\{A_\epsilon\}$ such that A_ϵ is polynomial time in $\frac{1}{\epsilon}$.

Examples: $O(\frac{1}{\epsilon}n^2)$ e $O(\frac{1}{\epsilon^2}n^5)$.

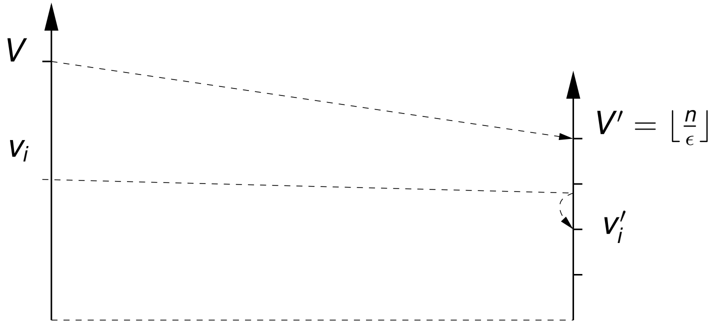
Knapsack Problem

Knapsack Problem: Given items $S = \{1, \ldots, n\}$, each item i with integer value v_i and integer weight s_i, for $i = 1, \ldots, n$, and integer B, find $S' \subseteq S$ that maximizes $\sum_{i \in S'} v_i$ such that $\sum_{i \in S'} s_i \leq B$.

Idea:
Change the scaling of the values, round down considering some discretization, and apply exact algorithm.

Def.: *Let $A(i, v) \equiv$ be the minimum weight of a set of $\{1, \ldots, i\}$ with value exactly v (∞ if such set does not exist).*

We consider that $s_i \leq B$ and $v_i > 0$ for any $i \in S$.



Changing the scale and rounding

Value loss for each item: $\frac{V}{n/\epsilon}$.

In the worst case: $n\frac{V}{n/\epsilon} = \epsilon\, V \leq \epsilon\, \mathrm{OPT}$

Exact algorithm for integer values

```
ALGORITHM EXACT-KNAPSACK($B, n, s, v$)
1  $V \leftarrow \max_i v_i$
2  for $i \leftarrow 0$ to $n$ do $A(i, 0) \leftarrow 0$
3  for $w \leftarrow 1$ to $nV$ do $A(0, w) \leftarrow \infty$
4  for $i \leftarrow 1$ to $n$ do
5      for $w \leftarrow 1$ até $nV$ do
6          if $v_i \leq w$ then
7              $A(i, w) \leftarrow \min(A(i-1, w), s_i + A(i-1, w-v_i))$
8          else
9              $A(i, w) \leftarrow A(i-1, w)$
```

Time complexity: $O(n^2 \cdot V)$.

Solution in cell (n, w): $w = \max\{w' : A(n, w') \leq B\}$.

Items \ Val.	0	1	2	$\dots$	$w-v_i$	$\dots$	w	$\dots$	nV
$\emptyset$	0	∞	∞	$\dots$	∞	$\dots$	∞	$\dots$	∞
$1 \dots 1$	0			$\dots$		$\dots$		$\dots$	
$1 \dots 2$	0			$\dots$		$\dots$		$\dots$	
$\vdots$	$\vdots$			$\dots$		$\dots$		$\dots$	
$1 \dots i-1$	0			$\dots$	$A(i-1, w-v_i)$	$\dots$	$A(i-1, w)$	$\dots$	
$1 \dots i$	0			$\dots$			$A(i, w)$	$\dots$	
$\vdots$	$\vdots$			$\dots$		$\dots$		$\dots$	
$1 \dots n$	0			$\dots$		$\dots$		$\dots$	

where $A(i, w) \equiv$ is the minimum weight of a subset of $\{1, \dots, i\}$ with value exactly w (∞ if such set does not exist).

$$A(i, w) = \begin{cases} \min((A(i-1, w)\ ,\ s_i + A(i-1, w-v_i)) & \text{se } \ v_i \leq w \\ A(i-1, w) & \text{C. c.} \end{cases}$$

ALGORITHM KNAPSACK-IK$_\epsilon(B, n, s, v)$

```
1     $V \leftarrow \max_i v_i$
2     $K \leftarrow \frac{\epsilon V}{n}$
3     $v'_i \leftarrow \left\lfloor \frac{v_i}{K} \right\rfloor \forall i$     % I.e., $v'_i = \left\lfloor \frac{n}{\epsilon} \frac{v_i}{V} \right\rfloor$
4     Run Exact-Knapsack$(B, n, s, v')$
```

Theorem: *Knapsack-IK$_\epsilon$ is an FPTAS.*

Prova. Let

S be a solution found by Knapsack-IK$_\epsilon$,
O^* be an optimum solution of I and
OPT be the value of O^*.

$$Kv'_i \leq v_i < K(v'_i + 1)$$

$$v_i \leq K(v'_i + 1) \implies Kv'_i \geq v_i - K$$

Therefore,

$$\begin{aligned}
\sum_{i\in S} v_i &\geq \sum_{i\in S} K \cdot v_i' \\
&\geq \sum_{i\in O^*} K \cdot v_i' \\
&\geq \sum_{i\in O^*} (v_i - K) = \sum_{i\in O^*} v_i - |O^*|K \\
&\geq \sum_{i\in O^*} v_i - nK = \sum_{i\in O^*} v_i - \epsilon V \\
&\geq \mathrm{OPT} - \epsilon \cdot \mathrm{OPT} = (1-\epsilon) \cdot \mathrm{OPT}.
\end{aligned}$$

Time Complexity: $O(n^2 V') = O(n^2 \left\lfloor \frac{V}{K} \right\rfloor) = O(n^3 \frac{1}{\epsilon})$. $\square$

PTAS for the Task Scheduling Problem

Problem SCHEDULING: Given a list of tasks $L = (J_1, \ldots, J_n)$, each task with time $t(J_i)$ and m identical machines, find a partition of L, $(M_1, \ldots, M_m)$, such that $\max_i t(M_i)$ is minimum.

Theorem: *Problem* SCHEDULING *is strongly* NP*-complete.*

Corollary: *There is no FPTAS to* SCHEDULING*, unless* $\mathrm{P} = \mathrm{NP}$*.*

There is a PTAS for the SCHEDULING

Def.: *A polynomial time algorithm R_ϵ is said to be $(1+\epsilon)$-restricted for the problem* SCHEDULING *if given ϵ, time T and an instance (L, t, m) it returns:*

- $\emptyset$ - *if there is no scheduling within time T or*
- $\mathcal{S}$ - *where $\mathcal{S}$ is a scheduling with $\mathrm{val}(\mathcal{S}) \leq (1+\epsilon)T$.*

Lemma: *Let $L := \max\{\max_j t(J_j)\,,\ \frac{\sum_j t(J_i)}{m}\}$, then*

$$\mathrm{OPT}(L) \in [L, 2L]$$

Proof. Note that algorithm SCHEDULING-GRAHAM returns a scheduling bounded by $\max_j t(J_j) + \frac{\sum_j t(J_i)}{m} \leq 2L$ □

Algorithm of Hochbaum and Shmoys'88

Idea: Build algorithm $(1+\epsilon)$-restricted and use binary search.
This algorithm uses a $(1+\epsilon)$-restricted subroutine R_ϵ.

```
SCHEDULING-HS$_\epsilon$ $(L, m, t)$
1    $\mathcal{S} \leftarrow$ SCHEDULING-GRAHAM$(L, m, t)$
2    $T' \leftarrow \max\{\max_j t(J_j)\ ,\ \frac{\sum_j t(J_i)}{m}\}$
3    $T'' \leftarrow 2T'$
4    $\epsilon' \leftarrow \frac{\epsilon}{3}$
4    while $T'' - T' > \epsilon' T'$ do
5        let $T \leftarrow \frac{T'+T''}{2}$
6        $\mathcal{S}' \leftarrow R_{\epsilon'}(L, m, t, T)$
7        if $\mathcal{S}' \neq \emptyset$ then
8            $T'' \leftarrow T$;
9            $\mathcal{S} \leftarrow \mathcal{S}'$;
10       else
11           $T' \leftarrow T$
12   return $\mathcal{S}$
```

Theorem: *(Hochbaum & Shmoys'88)* SCHEDULING-HS *is a PTAS for the Scheduling problem.*

Proof.

In each iteration, we have a scheduling $\mathcal{S}'$ and interval $[T', T'']$ that:

- $\mathrm{val}(\mathcal{S}') \in [T', T''(1+\epsilon')]$,
- $T' \leq \mathrm{OPT}$.

The binary search stops with a scheduling $\mathcal{S}$ such that $\frac{T''}{T'} \leq 1+\epsilon'$. So,

$$\begin{aligned}
\mathrm{val}(\mathcal{S}) &\leq (1+\epsilon')T'' \\
&\leq (1+\epsilon')(T' + \epsilon' T') \\
&= (1+\epsilon')^2 T' \\
&\leq (1+2\epsilon'+\epsilon'^2)\mathrm{OPT} \\
&\leq (1+\epsilon)\mathrm{OPT}
\end{aligned}$$

Time complexity: polynomial in the instance size and in $\frac{1}{\epsilon}$. $\square$

A $(1+\epsilon)$-restricted algorithm

Subroutine: algorithm $(1+\epsilon)$-restricted, RG_ϵ, for tasks with time larger than ϵT.

```
$R_\epsilon(L, m, t, T)$
1 $G \leftarrow \{j \in L : t(j) > \epsilon T\}$
2 $P \leftarrow L \setminus G$
3 $\mathcal{S}_G \leftarrow RG_\epsilon(G, m, t, T)$
4 if $\mathcal{S}_G = \emptyset$ then return $\emptyset$
5 else
6     schedule the items of $P$ into $\mathcal{S}_G$ using algorithm GRAHAM
7     if $\mathrm{val}(\mathcal{S}) > (1+\epsilon)T$ then return $\emptyset$
8     else return $\mathcal{S}$
```

Lema: *R_ϵ is a $(1+\epsilon)$-restricted algorithm.*

Proof. Exercise. □

A $(1+\epsilon)$-restricted algorithm for large items

Subrotine: algorithm, $Exact_{\epsilon,k}$, obtains a scheduling in time T, when there is up to k different times.
If there is no such scheduling, return $\emptyset$.

Idea: Define fixed times and round down processing times.

```
$RG_\epsilon(G, m, t, T)$
1   for each $j \in G$ do
2       let $i$ be such that $t(j) \in [\ \epsilon T + i\epsilon^2 T\ ,\ \epsilon T + (i+1)\epsilon^2 T\ )$
3       $t'(j) \leftarrow \epsilon T + i\epsilon^2 T$
4   $k \leftarrow \lfloor 1/\epsilon \rfloor$
5   $\mathcal{S} \leftarrow Exact_{\epsilon,k}(G, m, t', T)$
6   return a partition $\mathcal{S}$
```

Lema: *RG_ϵ is a $(1+\epsilon)$-restricted algorithm.*
Proof.
Note that $t'(j) \in \{\epsilon T,\ \epsilon T + \epsilon^2 T, \ldots, \epsilon T + k^2 \epsilon^2 T\}$.

Let $\mathcal{S} \neq \emptyset$ a scheduling produced by $RG_\epsilon(G, m, t', T)$.

Note that

- there are at most $k = \frac{1}{\epsilon}$ tasks in each machine;
- there are at most $k^2 = \frac{1}{\epsilon^2}$ different task sizes;
- the loss due to the rounding is at most $\epsilon^2 T$ for each task.
- the total loss due to the rounding is at most ϵT.

Therefore, the total processing time in each machine in $\mathcal{S}$ is at most $(1+\epsilon)T$. □

Exact algorithm for big tasks and restricted times

Given instance (G, m, t', T) and integer k, where $t'(j) \geq \epsilon T$ and there are at most k^2 different times in t', find a scheduling $\mathcal{S}$ with makespan at most T, if one exists.

We give a decision version of this algorithm:

Decide if a set L of tasks can be scheduled with makespan T using at most m machines

Approach: Dinamic Programming.

- Number of possibilities to assign jobs in one machine is polynomial.
- For m machines, test all possibilities in one machine and verify each possibility if the remaing can be executed in $m-1$ machines.

$\text{DecisionExact}_{\epsilon,l}(G, m, t', T)$

```
1  for $i = 1$ to $k^2$ do
2      $s_i \leftarrow \epsilon T + (i-1)\epsilon^2 T$
3      $n_i \leftarrow |\{j \in G : t'(j) = s_i\}|$
4  let $\mathcal{Q} \leftarrow \{(a_1, \ldots, a_{k^2}) : \sum_{i=1}^{k^2} a_i s_i \leq T$ e $0 \leq a_i \leq n_i\}$
5  $M(0, \ldots, 0) \leftarrow 0$
6  for each $(a_1, \ldots, a_{k^2}) \in \mathcal{Q}$ do $M(a_1, \ldots, a_{k^2}) \leftarrow 1$
7  for $a_1 \leftarrow 0$ to $n_1$ do
8      for $a_2 \leftarrow 0$ to $n_2$ do
9              $\ldots$
10         for $a_{k^2} \leftarrow 0$ to $n_{k^2}$ do
11             if $(a_1, \ldots, a_{k^2}) \notin \mathcal{Q}$ then
12                 let $(b_1,\ldots,b_{k^2}) \in \mathcal{Q}$ s.t. $M(a_1 - b_1, \ldots, a_{k^2} - b_{k^2})$ is minimum
13                 $M(a_1, \ldots, a_{k^2}) \leftarrow 1 + M(a_1 - b_1, \ldots, a_{k^2} - b_{k^2})$
14 if $M(n_1, \ldots, n_{k^2}) \leq m$ return YES
15 else return NO
```

Lema: *Given instance (G, m, t') and time T, where G has at most k^2 distinct processing times in t', algorithm DecisionExact$_{\epsilon,k^2}$ decide if G can be scheduled into m machines within time T in polynomial time.*

Proof. As $n_i \leq n$, $i = 1, \ldots, k^2$, we have

- $|\mathcal{Q}| = O((k+1)^{k^2})$
- Steps 4 and 6: time $O(k\,(k+1)^{k^2})$
- Steps 7–13: time $O(n^{k^2} k\,(k+1)^{k^2})$

For constant value of k, the algorithm has polynomial time complexity. □

Integer Linear Programming

LP Problem: Given matrix $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$, vectors $c = (c_i) \in \mathbb{Q}^n$ and $b = (b_i) \in \mathbb{Q}^m$, find a vector $x = (x_i) \in \mathbb{Q}^n$ (if exists) that

$$\begin{array}{ll} \text{minimize} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to} & \left\{ \begin{array}{lcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq & b_2 \\ \vdots & \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = & b_m \\ x_i \in \mathbb{Q} & & \end{array} \right. \end{array}$$

Theorem: *LP can be solved in polynomial time.*

ILP Problem: Given matrix $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$, vectors $c = (c_i) \in \mathbb{Q}^n$ and $b = (b_i) \in \mathbb{Q}^m$, find vector $x = (x_i) \in \mathbb{Z}^n$ (if exists) that

$$\begin{array}{ll} \text{minimize} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to} & \left\{ \begin{array}{lcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq & b_2 \\ \vdots & \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = & b_m \\ x_i \in \mathbb{Z} \end{array} \right. \end{array}$$

Theorem: *ILP is an NP-hard problem.*

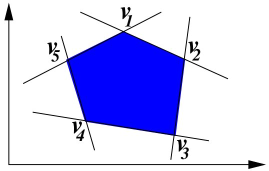
Def.: We say that a set of points P,

$$P := \left\{ x \in \mathbb{Q}^n : \begin{array}{ccc} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq & b_m \end{array} \right\}$$

as a polyhedron.

Def.: If $y_i \in P$ and $\alpha_i \in \mathbb{Q}$, $i = 1, \ldots, n$ we say that $y = \alpha_1 y_1 + \alpha_2 y_2 + \cdots + \alpha_n y_n$ is a convex combination of points $y_i' s$ if $\alpha_i \geq 0$ and $\alpha_1 + \cdots + \alpha_n = 1$

Def.: The vertices of P are the points of P that cannot be written by convex combination of other points of P



Theorem: *A solution that is a vertex of a LP can be found in polynomial time.*

Polynomial time algorithms to solve LP:

- Ellipsoid algorithm and
- Interior Point Method.

Exponential time method to solve LP:

- Simplex Method (polynomial time on the average).

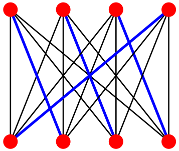
Formulation using binary variables

Consider an event e and variables x_e formulated as:

$$x_e = \begin{cases} 1 & \text{if event } e \text{ occurs,} \\ 0 & \text{if event } e \text{ does not occur.} \end{cases}$$

Matching in bipartite graphs

Def.: Given a graph $G = (V, E)$, we say that $M \subseteq E$ is a matching of G if M does not have edges with vertices in common.



Bipartite matching problem: Given a bipartite graph $G = (V, E)$, and costs in the edges $c : E \to \mathbb{Z}$, find a matching $M \subseteq E$ that maximizes $c(M)$.
Application: Find assignment of candidates to vacancies, maximizing global score.

Formulation to bipartite matching problem

Integer programming formulation

$$
\begin{array}{ll}
\text{maximize} & \sum_{e \in E} c_e x_e \\
\text{subject to} & \left\{ \begin{array}{rcll} \sum_{e \in \delta(v)} x_e & \leq & 1 & \forall v \in V \\ x_e & \in & \{0,1\} & \forall e \in E \end{array} \right.
\end{array}
$$

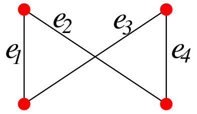
Linear Relaxation

$$
\begin{array}{ll}
\text{maximize} & \sum_{e \in E} c_e x_e \\
\text{subject to} & \left\{ \begin{array}{rcccll} & & \sum_{e \in \delta(v)} x_e & \leq & 1 & \forall e \in E \\ 0 & \leq & x_e & \leq & 1 & \forall e \in E \end{array} \right.
\end{array}
$$

Theorem: *The vertices of the relaxed polyhedron are integers.*

Example:

Consider the following bipartite graph with unit costs:



Linear Programming

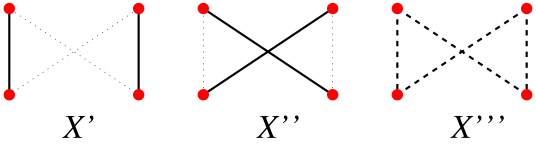
$$\begin{array}{ll}\text{maximize} & x_{e_1} + x_{e_2} + x_{e_3} + x_{e_4} \\ \text{subject to} & \left\{\begin{array}{l} x_{e_1} + x_{e_2} \le 1, \\ x_{e_3} + x_{e_4} \le 1, \\ x_{e_1} + x_{e_3} \le 1, \\ x_{e_2} + x_{e_4} \le 1, \\ 0 \le x_{e_1} \le 1, \\ 0 \le x_{e_2} \le 1, \\ 0 \le x_{e_3} \le 1, \\ 0 \le x_{e_4} \le 1 \end{array}\right.\end{array}$$

We have 4 binary variables in the corresponding LP:

$$X = (x_{e_1}, x_{e_2}, x_{e_3}, x_{e_4})$$

The following vectors are optimum solutions:

$$X' = (1,0,0,1) \quad X'' = (0,1,1,0) \quad X''' = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$$



- X' and X'' are vertices of the matching polyhedron
- X''' is optimum solution, but is convex combination of X' and X'' ($X''' = \frac{1}{2}X' + \frac{1}{2}X''$).

Maximum Flow and Minimum Cut

Def.: Given an oriented graph $G = (V, E)$, capacities $c : E \to \mathbb{Q}^+$ and nodes s and t, a flow from s to t is a vector $f \in \mathbb{Q}^E$, such that

$$\sum_u f(u, v) - \sum_w f(v, w) = 0 \quad \forall v \in V - \{s, t\},$$

$$0 \leq f(e) \leq c(e) \quad \forall e \in E,$$

The value of flow f is equal to $\sum_v f(s, v)$

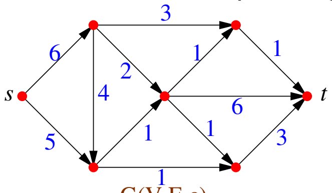
Maximum Flow Problem: Given a oriented graph $G = (V, E)$, capacities $c : E \to \mathbb{Q}^+$ and vertices s and t, find a flow of maximum value from s to t.

Theorem: *If the capacities c_e are integers, then the vertices of the flow polyhedron are integers.*

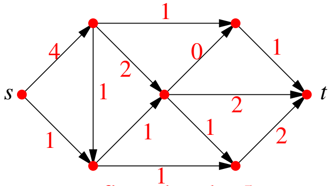
Def.: Given $S \subseteq V$, $s \in S$ and $t \notin S$, the capacity of the cut $(S, \overline{S})$ is equal to $\sum_{(u,v) \in (S, \overline{S})} c(v, w)$.

Theorem: (Menger) If $c_e = 1, \forall e$, then the maximum number of edge disjoint paths is equal to the minimum number of edge removal to disconnect s and t.

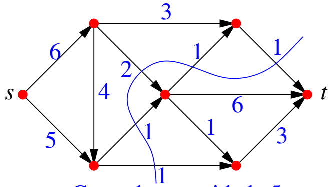
Theorem: *The value of the maximum flow from s to t is equal to the capacity of the minimum cut separating s and t.*



G(V,E,c)



fluxo de valor 5



Corte de capacidade 5

Corolário: *The maximum number of edge disjoint paths from s to t is equal to the minimum number of edge removal to disconnect s and t.*

Knapsack problem

Knapsack problem: Given items $S = \{1, \ldots, n\}$ with value v_i and integer size s_i, $i = 1, \ldots, n$, and integer B, find $S' \subseteq S$ that maximizes $\sum_{i \in S'} v_i$ such that $\sum_{i \in S'} s_i \leq B$.

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\sum_{i \in [n]} v_i x_i \\
\text{subject to} & \left\{ \begin{array}{rcll} \displaystyle\sum_{i \in [n]} s_i x_i & \leq & B & \\ x_i & \in & \{0, 1\} & \forall i \in [n] \end{array} \right.
\end{array}
$$

where $[n] = \{1, \ldots, n\}$.

Covering, Packing and Partitions

Let E be a set and $\mathcal{C}$ a collection of subsets of E.

Let $\mathcal{C}_e := \{C \in \mathcal{C} : e \in C\}$ and $\mathcal{S} \subseteq \mathcal{C}$ such that $x_C = 1 \Leftrightarrow C \in \mathcal{S}$.

- **$\mathcal{S}$ is a covering if**

$$\sum_{C \in \mathcal{C}_e} x_C \geq 1 \quad \forall e \in E,$$

- **$\mathcal{S}$ is a packing if**

$$\sum_{C \in \mathcal{C}_e} x_C \leq 1 \quad \forall e \in E,$$

- **$\mathcal{S}$ is a partition if**

$$\sum_{C \in \mathcal{C}_e} x_C = 1 \quad \forall e \in E.$$

Set covering problem

Set covering problem: Given set E, subsets $\mathcal{S}$ of E, costs $c(S)$, $S \in \mathcal{S}$, find covering $\mathcal{S}' \subseteq \mathcal{S}$ that minimizes $c(\mathcal{S}')$.

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{S \in \mathcal{S}} c_S x_S \\ \text{subject to} & \left\{ \begin{array}{rcll} \displaystyle\sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\ x_S & \in & \{0,1\} & \forall S \in \mathcal{S} \end{array} \right. \end{array}$$

Facility Location Problem

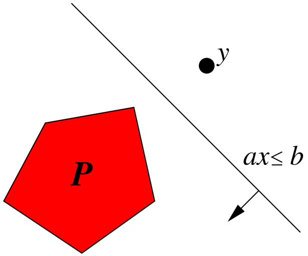
Facility Location Problem: Given potential facilities $F = \{1, \ldots, n\}$, clients $C = \{1, \ldots, m\}$, costs f_i to “open” a facility i and costs $c_{ij} \in \mathbb{Z}$ for each client j attended by facility i. Find facilities $A \subseteq F$ such that, the cost to open facilities in A and the cost to attend all clients

Application: Install distribution outlets.

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i \in F} f_i y_i + \sum_{ij \in E} c_{ij} x_{ij} \\
\text{subject to} & \left\{ \begin{array}{rcll} \displaystyle\sum_{ij \in E} x_{ij} & = & 1 & \forall j \in C, \\ x_{ij} & \leq & y_i & \forall ij \in E, \\ y_i & \in & \{0, 1\} & \forall i \in F \\ x_{ij} & \in & \{0, 1\} & \forall i \in F \text{ e } j \in C. \end{array} \right.
\end{array}
$$

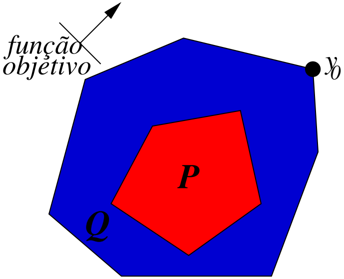
Optimization $\times$ Separation

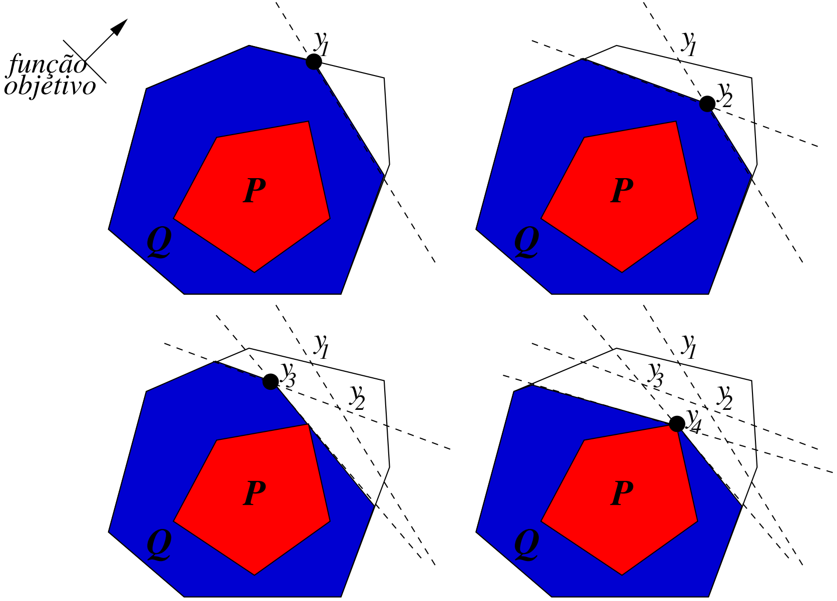
Def.: Separation Problem: Let $P \subseteq \mathbb{R}^n$ a convex set and $y \in \mathbb{R}^n$, determine if $y \in P$, otherwise, find constraint $ax \leq b$ such that $P \subseteq \{x : ax \leq b\}$ and $ay > b$.



Theorem: *The optimization of a linear program can be solved in polynomial time if and only if the separation problem can be solved in polynomial time.*

```
CUTTING PLANE ALGORITHM($P, c$)
        $P$ polyhedron (non necessarily in explicit way),
        $c$ objective function
1   $Q \leftarrow$ {Initial Polyhedron}
2   $y \leftarrow$OPT-LP($Q, c$)
3   while $y$ can be separated from $Q$ do
4       let $ax \leq b$ be a valid inequality that separate $y$
5       $Q \leftarrow Q \cap \{x : ax \leq b\}$
6       $y \leftarrow$OPT-LP($Q, c$)
7   if $Q = \emptyset$ return $\emptyset$
8   else return $y$.
```




função objetivo
P
Q
y_1
y_2
y_3
y_4

Matching problem

Def.: Given a graph $G = (V, E)$, we say that $M \subseteq E$ is a matching of G if M does not have edges with common vertices.

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\sum_{e\in E} c_e x_e \\
\text{subject to} & \left\{
\begin{array}{rclll}
\displaystyle\sum_{e\in\delta(v)} x_e & \leq & 1 & \forall v \in V, & \\
\displaystyle\sum_{e\in E[S]} x_e & \leq & \dfrac{|S|-1}{2} & \forall S \subseteq V, & |S| \text{ odd}, \\
x_e & \in & \{0,1\} & \forall e \in E. &
\end{array}
\right. ,
\end{array}
$$

where $E[S] := \{e \in E : e$ have both vertices in $S\}$.

Theorem: *The vertices of the relaxed matching polytope are integers (Edmonds'65).*

Theorem: *The separation for the relaxed matching polytope constraints can be solved in polynomial time (Padberg & Rao'82).*

Traveling Salesman Problem

Problema TSP: Given a graph $G = (V, E)$ and cost c_e in $\mathbb{Q}_{\geq}$ for each edge e, find a hamiltonian circuit C that minimizes $c(C)$.

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{e \in E} c_e x_e \\
\text{subject to} & \left\{ \begin{array}{rcll}
\displaystyle\sum_{e \in \delta(v)} x_e & = & 2 & \forall v \in V \\
\displaystyle\sum_{e \in \delta(S)} x_e & \geq & 2 & \forall S \subset V, \quad S \neq \emptyset \\
 & & & \text{(subtour elimination constraints)} \\
x_e & \in & \{0, 1\} & \forall e \in E
\end{array} \right.
\end{array}
$$

onde $E[S] := \{e \in E : e$ has both vertices in $S\}$.

Theorem: *The separation of the subtour elimination constraints can be solved in polynomial time.*

Steiner Forest Problem

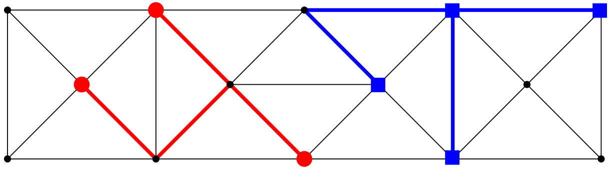
Let G be a graph and $\mathcal{R}$ a collection of subsets of V_G.

Def.: *A $\mathcal{R}$-forest of G is any spanning forest F of G such that for any $R \in \mathcal{R}$, the elements of R belongs to some component of F.*

Steiner Forest Problem: Given a graph G, cost c_e in $\mathbb{Q}_{\geq}$ for each edge e and a collection $\mathcal{R}$ of sets of V_G, find a $\mathcal{R}$-forest F that minimizes $c(F)$.

Applications:

- VLSI circuit routing.
- Network design problems.



Formulation:

$$\begin{array}{lrcll}
\text{minimize} & \displaystyle\sum_{e \in E} c_e x_e & & & \\
\text{subject to} & \left\{ \begin{array}{rcll} \displaystyle\sum_{e \in \delta(S)} x_e & \geq & 1 & \forall S \subset V : \exists R \in \mathcal{R}, \emptyset \neq S \cap R \neq \\ & & & \text{(connectivity constraints)} \\ x_e & \in & \{0,1\} & \forall e \in E \end{array} \right. & & &
\end{array}$$

Theorem: *The separation problem of the connectivity constraints can be solved in polynomial time.*

LP Rounding Method

Vertex Cover Problem: Given graph $G = (V, E)$ and costs $c : V \to \mathbb{Q}_{\geq}$, find $S \subseteq V$ such that $\{u, v\} \cap S \neq \emptyset \quad \forall \{u, v\} \in E$ and $c(S)$ is minimum.

$$
(P) \qquad \begin{array}{rrcll}
\text{minimize} & cx & & & \\
\text{subject to} & x_i + x_j & \geq & 1 & \text{for each } ij \text{ in } E\,, \\
 & x_i & \geq & 0 & \text{for each } i \text{ in } V\,.
\end{array}
$$

```
MINCV-NT ($G = (V, E), c$)
1    let $\hat{x}$ be an optimum solution of $(P)$
2    $S \leftarrow \{v \in V : \hat{x}_v \geq 1/2\}$
3    return $S$
```

Theorem: MinCV-NT is a 2-approximation (Nemhauser,Trotter'75; Hochbaum'83).

Proof. MinCV-NT produces a vertex cover:

$(S := \{v \in V : \hat{x}_v \geq 1/2\} \quad \text{and} \quad x_i + x_j \geq 1 \quad \forall ij \in E) \Longrightarrow S$ is a covering.

$$\begin{aligned} c(S) &= \sum_{i \in S} c_i \\ &\leq \sum_{i \in S} c_i 2\,\hat{x}_i \\ &= 2 \sum_{i \in S} c_i \hat{x}_i \\ &\leq 2 \sum_{i \in V} c_i \hat{x}_i \\ &= 2\, c\, \hat{x} \\ &\leq 2\,\mathrm{OPT} \end{aligned}$$

$\square$

Generalization to the Set Covering Problem

Set Covering Problem: Given set E, family of subsets $\mathcal{S}$ of E, costs $c(S) \in \mathbb{Q}_{\geq}$, $S \in \mathcal{S}$, find covering $\mathcal{S}' \subseteq \mathcal{S}$ that minimizes $\sum_{S \in \mathcal{S}'} c(S)$.

$$
\begin{array}{llrcll}
 & \text{minimize} & cx & & & \\
(P) & \text{subject to} & \displaystyle\sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \text{for each } e \text{ in } E\,, \\
 & & x_S & \geq & 0 & \text{for each } S \text{ in } \mathcal{S}\,.
\end{array}
$$

where $\mathcal{S}_e = \{S \in \mathcal{S} : \ e \in S\}$

MINCC-HOCHBAUM $(E, \mathcal{S}, c)$

```
1   let $\hat{x}$ an optimal solution of $(P)$
2   for each $e$ in $E$ do $f_e \leftarrow |\{S \in \mathcal{S} : e \in S\}|$
3   $\beta \leftarrow \max_{e \in E} f_e$
4   $\mathcal{T} \leftarrow \{S \in \mathcal{S} : \hat{x}_S \geq 1/\beta\}$
5   return $\mathcal{T}$
```

Theorem: *The algorithm* MinCC-Hochbaum *is a* β*-approximation for* MinCC $(E, \mathcal{S}, c)$*, where* β *is the maximum number of times an element of* E *appear in subsets of* $\mathcal{S}$*.*
Proof. The cost of the covering $\mathcal{T}$ produced by the algorithm is

$$\begin{aligned} c(\mathcal{T}) &= \sum_{S \in \mathcal{T}} c_S \\ &\leq \sum_{S \in \mathcal{T}} c_S \beta\, \hat{x}_S \\ &= \beta \sum_{S \in \mathcal{T}} c_S \hat{x}_S \\ &\leq \beta\, c \hat{x} \\ &\leq \beta\, \mathrm{OPT}(E, \mathcal{S}, c)\ , \end{aligned}$$

where the first inequality is valid because $\beta\, \hat{x}_S \geq 1$ for each S in $\mathcal{T}$. $\square$

Unrelated Machine Scheduling

Unrelated Machine Scheduling problem: Given a list of tasks $J = (1, \ldots, n)$, and machines $M = (1, \ldots, m)$, times $p_{ij} \geq 0$ for each $ij \in M \times J$, find a partition of J, $(M_1, \ldots, M_m)$, such that $\max\left\{\sum_{j \in M_i} p_{ij} : i \in M\right\}$ is minimum.

Theorem: UNRELATED MACHINE SCHEDULING is NP-hard.

Here is an Integer Linear Programming (ILP) formulation:

$$\begin{array}{lll} \text{minimize} & T & \\ \text{such that} & \displaystyle\sum_{i \in M} x_{ij} = 1, & \text{for any } j \in J, \\ & \displaystyle\sum_{j \in J} p_{ij} x_{ij} \leq T, & \text{for any } i \in M, \\ & x_{ij} \in \{0, 1\}, & \text{for any } i \in M \text{ and } j \in J. \end{array}$$

Unrelated Machine Scheduling

Linear Relaxation:

$$\begin{array}{lrl} \text{minimize} & T & \\ \text{such that} & \displaystyle\sum_{i\in M} x_{ij} = 1, & \text{for any } j \in J, \\ & \displaystyle\sum_{j\in J} p_{ij}x_{ij} \le T, & \text{for any } i \in M, \\ & x_{ij} \ge 0, & \text{for any } i \in M \text{ and } j \in J. \end{array}$$

Fact: *The integrality gap of the relaxation is unbounded.*

Proof. Consider $J = \{1\}$, $M = \{1, \dots, m\}$, and $p_{i1} = m$. The optimum solution value of the relaxed formulation has value 1, while the optimum integer solution has value m, which leads to an integrality ratio of m. □

Unrelated Machine Scheduling

Consider a time $T \geq 0$ and an ILP to decide if there exists a scheduling of J into M in time at most T.

If this ILP is feasible, $x_{ij} = 0$ whenever $p_{ij} > T$.

Let $\mathcal{E}(T) = \{(i,j) \in M \times J : p_{ij} \leq T\}$.

Consider the relaxed parameterized formulation $\mathrm{P}(T)$

$$
\mathrm{P}(T) \qquad
\begin{aligned}
\sum_{i:\,(i,j)\in\mathcal{E}(T)} x_{ij} &= 1, && \text{para todo } j \in J,\\
\sum_{j:\,(i,j)\in\mathcal{E}(T)} p_{ij}x_{ij} &\leq T, && \text{para todo } i \in M,\\
x_{ij} &\geq 0, && \text{para todo } (i,j) \in \mathcal{E}(T).
\end{aligned}
$$

Let T^* be the smallest value such that $\mathrm{P}(T^*)$ is feasible and

let x^* be a corresponding optimum solution for $\mathrm{P}(T^*)$ that is an extremal point.

Unrelated Machine Scheduling

```
Algorithm $\mathcal{A}_{UMS}$: $(J, M, p)$
1   Let $T^*$ be the smallest value such that $\mathrm{P}(T^*)$ is feasible.
2   Let $x^*$ be an extremal point solution to $\mathrm{P}(T^*)$.
3   Let $N_i \leftarrow \{j \in J : x^*_{ij} = 1\}$ for $i = 1, \ldots, m$.
4   Let $G$ the graph with vertices set $V(G) = M \cup J$ and
    edges $E(G) = \{(i, j) \in M \times J :\ 0 < x^*_{ij} < 1\}$.
5   Let $A \subseteq E(G)$ a maximum matching in $G$.
6   Let $M_i \leftarrow N_i \cup \{j \in J : (i, j) \in A\}$, for $i = 1, \ldots, m$.
7   Return $(M_1, \ldots, M_m)$.
```

Unrelated Machine Scheduling

Lemma: *x^* has at most $n + m$ non-null variables.*

Proof.

- Let v be the number of variables in P(T).
- There exists a total of $n + m + v$ constraints
- and v constraints are in the form $x^*_{ij} \geq 0$.
- x^* must satisfy v linear independent constraints with equality
- Therefore, at least $v - (n + m)$ constraints of type $x^*_{ij} \geq 0$ are satisfied with equality.

So, at most $v - [v - (n + m)] = n + m$ entries in x^* can be non-zero. □

Unrelated Machine Scheduling

Def.: *A connected graph is a pseudotree if it contains at most one cycle.*

A graph is a pseudoforest if each one of its components is a pseudotree.

Given time T and an extremal point x of $\mathrm{P}(T)$, let G_x the graph with vertices $V(G_x) = M \cup J$ and edges $E(G_x) = \{(i,j) \in \mathcal{E}(T) : 0 < x_{ij} < 1\}$.

Unrelated Machine Scheduling

Lemma: *Let $T \geq 0$ and an extremal point x of $P(T)$, then G_x is a pseudoforest.*

Proof. Let C be a conected component G_x and M' and J' the sets of machines and tasks obtained in the edges of C.

Let $\mathrm{P}_C(T)$ and x_C the constraints of the linear program $\mathrm{P}(T)$ and the vector x for the connected component C and the sets of machines and tasks M' and J'.

Let $x_{\overline{C}}$ the vector of the other variables in x that are not in x_C.

For simplicity, reorder the coordinates so that x can be written as the concatenation of x_C and $x_{\overline{C}}$, i.e., $x = x_C \,|\, x_{\overline{C}}$.

Cont.

Fact: *x_C is an extremal point of $\mathrm{P}_C(T)$.*

Proof. Suppose by contradiction that x_C is not extremal point. I.e., existem solutions distintas x'_C and x''_C such that x_C is convex combination of x'_C and x''_C.

Let $x' = x'_C | x_{\overline{C}}$ the vector obtained by concatenating the vectors x'_C and $x_{\overline{C}}$ and $x'' = x''_C | x_{\overline{C}}$ the vector obtained by concatenation of the vectors x''_C and $x_{\overline{C}}$.
Note that x' and x'' are distinct solutions from $\mathrm{P}(T)$.

It follows that

$$\begin{aligned} x &= x_C \,|\, x_{\overline{C}} \\ &= (\alpha x'_C + \beta x''_C) \,|\, (\alpha x_{\overline{C}} + \beta x_{\overline{C}}) \\ &= \alpha(x'_C \,|\, x_{\overline{C}}) + \beta(x''_C \,|\, x_{\overline{C}}) \\ &= \alpha x' + \beta x''. \end{aligned}$$

That is, x is convex combination of x' and x'', contradicting the fact that x is extremal point of $\mathrm{P}(T)$. □

Cont.

Analogously to the formulation $\mathrm{P}(T)$, the linear program $\mathrm{P}_C(T)$ has $n' + m' + v'$ constraints, where $n' = |J'|$, $m' = |M'|$ and v' is the number of variables in $\mathrm{P}_C(T)$.

From these contraints, v' of them are of type $x_{ij} \geq 0$. As x_C is extremal point of $\mathrm{P}_C(T)$, it follows from Lemma 112 that x_C has at most $n' + m'$ non-null variables.

As C is a connected graph with $n' + m'$ vertices, there is $n' + m' - 1$ variables of x_C that produces a tree of C and therefore C has at most one cycle, and therefore, a pseudotree of X.

□

Lemma: *Let $T \geq 0$ and a extremal point x of $\mathrm{P}(T)$, then G_x has a perfect matching.*

Proof.

- G_x is bipartite (machines $\times$ tasks) and pseudoforest.
- All leaves of G_x are machines (cannot be tasks with only one fractional edge)
- Start with an empty matching M.
- Whenever there is an edge (i, j), where i is a leaf include (i, j) into M and remove i and j.
  Repeat this process while there are leaves.
- The residual graph is a set of cycles, each one of even size.
- For each cycle, include half of its edges to M alternating positions.

$\square$

Teorema: *The algorithm $\mathcal{A}_{UMS}$ is a 2-approximation.*

Proof.
Let

- T^* e x^* obtained in steps 1 and 2. Clearly $T^* \leq \mathrm{OPT}$
- N the assignment made in step 3 (with integer variables).
- A the assignment made via maximum matching in step 5.

The final scheduling is $N \cup A$.

- The makespan of N is at most T^*. That is
$$makespan(N) \leq \mathrm{OPT}$$
- The makespan of A is at most the size of the largest task.
$$makespan(A) \leq \mathrm{OPT}$$

Therefore,

$$\mathcal{A}_{UMS}(I) \leq makespan(N) + makespan(A) \leq 2\mathrm{OPT}$$

□

Unrelated Machine Scheduling

Unrelated Machine Scheduling

Asympotic approximation factor

Adequate when the approximation factor is only attended for “small” instances

Given an algorithm *A* for a minimization problem and instance *I*,

- OPT(*I*) is the value of an optimum solution
- *A*(*I*) is the value of a solution produced by algorithm *A*
- *A* has asymptotic approximation factor α if

$$R_A^{\infty} := \inf \left\{ r \geq 1 : \ \exists N > 0, \frac{A(I)}{\mathrm{OPT}(I)} \leq r, \mathrm{OPT}(I) > N \right\} \leq \alpha$$

Analogous definitions can be done for maximization problems.

Notation: Given set $B \subseteq \{1, \ldots, n\}$ and a function $s : \mathbb{N} \to \mathbb{R}$, we denote by $s(B)$ the value $\sum_{i \in B} s_i$.

Bin Packing Problem (n, s) Given positive integer n and, for each i in $\{1, \ldots, n\}$, a rational number s_i in the interval $[0, 1]$, find a partition $\mathcal{B}$ of $\{1, \ldots, n\}$ such that $s(B) \leq 1$ for any B in $\mathcal{B}$ and $|\mathcal{B}|$ is minimum.

Theorem: *There is no α-approximation for the* BIN PACKING *problem with $\alpha < 3/2$, unless* P $=$ NP*.*

Theorem: *(Fernandez de la Vega and Lucker'81) There exists a polynomial time algorithm A_ϵ, $\epsilon > 0$, for the* BIN PACKING *problem such that*

$$A_\epsilon(L) \le (1+\epsilon)\mathrm{OPT}(L) + 1,$$

for any instance L.

We will prove the following result:

Theorem: *(Fernandez de la Vega and Lucker'81) There exists a linear time algorithm A_ϵ, $\epsilon > 0$, for the bin packing problem, such that*

$$A_\epsilon(L) \le (1+\epsilon)\mathrm{OPT}(L) + \beta_\epsilon,$$

for any instance L, where β_ϵ only depends on ϵ.

NF (n, c)

```
1   $k \leftarrow 1$
2   $B_k \leftarrow \emptyset$
3   for $i$ from 1 to $n$ do
4       if $s_i \leq 1 - s(B_k)$
5           then $B_k \leftarrow B_k \cup \{i\}$
6           else $k \leftarrow k + 1$
7                $B_k \leftarrow \{i\}$
8   return $\{B_1, \ldots, B_k\}$
```

Theorem: NF *is a* 2*-approximation for the bin packing problem.*
Proof. Exercise. $\square$

Lemma: *If* $\mathcal{B} = \{B_1, \ldots, B_k\}$ *is a packing of* L *such that* $s(B_i) \geq 1 - \frac{\epsilon}{2}$, *for* $i = 1, \ldots, k-1$ *and* $\epsilon \in (0, 1)$ *then*

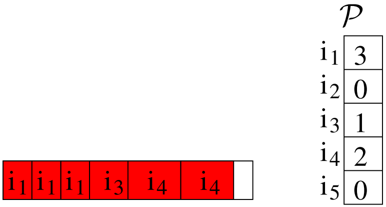
$$k \leq (1 + \epsilon) \cdot \mathrm{OPT}(L) + 1$$

Proof. Exercise. $\square$

Restricting itens in the number of types and size

L has at most *k* different item types, each with size at least ϵ where *k* and ϵ are constants.

Representation of a pattern in a vector



k different types in *L*, each item with size at least ϵ

$$\Downarrow$$

number of different patterns is bounded by constant.

Algorithm RST$_{k,\epsilon}(L)$

1. Produce all possible patterns $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_w$ of packing items of L into one unit bin.
2. Let x^* be an optimum solution of the following LP with at most k non-null variables:

$$\begin{array}{ll} & \text{minimize } \sum x_i \\ \text{s.t.} & \mathcal{P}_1 \cdot x_1 + \mathcal{P}_2 \cdot x_2 + \cdots + \mathcal{P}_w \cdot x_w = M, \\ & x_i \in \mathbb{Q}^* \end{array}$$

   where M is the vector of multiplicities of the items in L.
3. Get a packing $\mathcal{P}$ of L rounding up the variables x^*
4. Return $\mathcal{P}$

Lemma: $\mathrm{RST}_{k,\epsilon}(L) \leq \mathrm{OPT}(L) + k$

Proof. As we have at most k diffent item types in L, the LP

$$\text{minimize} \sum_{i=1}^{w} x_i$$

$$\text{s.a.} \quad \mathcal{P}_1 \cdot x_1 + \mathcal{P}_2 \cdot x_2 + \cdots + \mathcal{P}_w \cdot x_w = M,$$

$$x_i \in \mathbb{Q}^*$$

has k lines and therefore, there exists a solution that is an extremal point with at most k non-null variables. □

Restricting the items in the size

L has items with size at least ϵ.

Idéia: Linear Rounding
Group items by size in k sublists and round up the size of the items of each sublist to the largest size in the sublist.

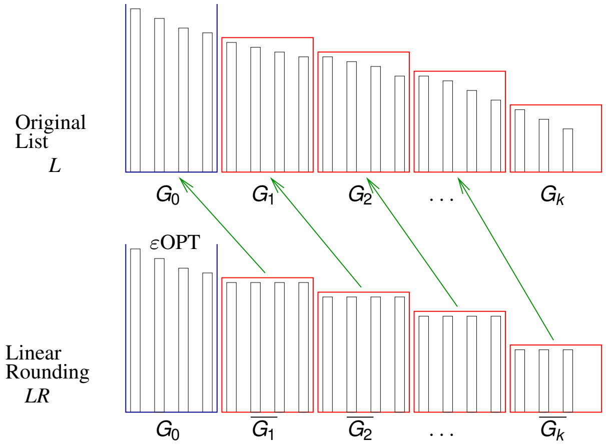
Def.: *We have* $A \preceq B$ *(*$B \succeq A$*) if* $\exists$ *injective function* $f : A \rightarrow B$ *such that* $s(i) \leq s(f(i))$.

Fact: *If* $A \preceq B$ *then* $\mathrm{OPT}(A) \leq \mathrm{OPT}(B)$

Def.: *Let* $\overline{A}$ *a list with items of* A *rounded up to the largest item in* A.

Linear Rounding: Supose all items in L are big ($s_i > \epsilon$)

Partition the list into groups of same size and
round up the items in each group, except for the largest group



Idea: $\mathrm{OPT}(LR) \leq (1+\varepsilon)\mathrm{OPT}(L)$

Let L sorted in non-increasing order and partitioned into groups
$L = (G_0 \| G_1 \| \dots \| G_k)$
where k and G_i are such that $|G_i| = q,\ i = 0, \dots, k-1$ and
$|G_k| \leq q$.

Fact: $\overline{G_i} \preceq G_{i-1}$.

Fact: $\mathrm{OPT}(\overline{G_1} \| \overline{G_2} \| \dots \| \overline{G_k}) \leq \mathrm{OPT}(L)$
Proof. Note that

$$\begin{aligned}
(\overline{G_1} \| \overline{G_2} \| \dots \| \overline{G_k}) &\preceq (G_0 \| G_1 \| \dots \| G_{k-1}) \\
&\preceq (G_0 \| G_1 \| \dots \| G_{k-1} \| G_k) \\
&= L
\end{aligned}$$

□

Algorithm $\mathrm{RS}_\epsilon(L)$

1. Sort L in non-increasing order of size.
2. Given $q = \lceil n \cdot \epsilon^2 \rceil$, partition L into groups $L = (G_0 \| G_1 \| \ldots \| G_k)$ where k and G_i are such that

$$|G_i| = q,\ i = 0, \ldots, k-1 \quad \text{and} \quad |G_k| \leq q$$

3. Let $J \leftarrow (\overline{G_1} \| \overline{G_2} \| \ldots \| \overline{G_k})$.
4. $\mathcal{P}_{1..k} \leftarrow \mathrm{RST}_{k,\epsilon}(J)$
5. $\mathcal{P}_0 \leftarrow \mathrm{NF}(G_0)$
6. Return $\mathcal{P}_0 \| \mathcal{P}_{1..k}$

Lemma: $RS_\epsilon(L) \leq (1+\epsilon)\cdot \mathrm{OPT}(L) + \beta_\epsilon$

Proof. As $J \preceq L$

$$\begin{aligned}\mathrm{RST}_{k,\epsilon}(J) &\leq \mathrm{OPT}(J) + k_\epsilon \\ &\leq \mathrm{OPT}(L) + k_\epsilon\end{aligned}$$

$$\begin{aligned}\mathrm{NF}(G_0) &\leq q \;=\; \lceil n\cdot\epsilon^2\rceil \\ &\leq n\cdot\epsilon^2 + 1 \\ &\leq \epsilon\cdot s(L) + 1 \\ &\leq \epsilon\cdot\mathrm{OPT}(L) + 1\end{aligned}$$

Therefore

$$\begin{aligned}\mathrm{RS}(L) &= \mathrm{RST}_{k,\epsilon}(J)+\mathrm{NF}(G_0) \\ &\leq (\mathrm{OPT}(L)+k_\epsilon)+(\epsilon\cdot\mathrm{OPT}(L)+1) \\ &= (1+\epsilon)\cdot\mathrm{OPT}(L)+\beta_\epsilon\end{aligned}$$

□

Algorithm $\mathcal{A}_\epsilon(L)$

1. Partition the list L into L' and L'':
   $$L' := \{i \in L : \ s(i) > \epsilon/2\} \quad \text{e} \quad L'' := L \setminus L'.$$

2. $\mathcal{P}'_1 \leftarrow \mathrm{RS}_{\epsilon/2}(L')$

3. Pack the items of L'' using NF strategy to pack into each bin of $\mathcal{P}'_1$ (if necessary, use new bins).

4. Return the packing generated in the previous step.

Theorem: $\mathcal{A}_\epsilon(L) \leq (1+\epsilon)\mathrm{OPT}(L) + \beta_\epsilon$

Proof. Let $B_1, \ldots, B_k$ be the set of bins generated by $\mathcal{A}_\epsilon$. We have two cases:

Caso 1: NF generated new bins to pack L''.
Then

$$s(B_i) \geq (1 - \epsilon/2), \qquad \text{for each } i = 1, \ldots, k-1.$$

Therefore $\mathcal{A}_\epsilon(L) \leq (1+\epsilon)\mathrm{OPT} + 1$.

Caso 2: NF did not generate new bins to pack L''.
Therefore

$$\mathcal{A}_\epsilon(L) = RS(L') \leq (1+\epsilon)\mathrm{OPT} + \beta_\epsilon$$

$\square$

PROBABILITY AXIOMS

Def.: *A probability space has 3 components:*

- A sample space Ω
- A family $\mathcal{F}$ of events, each $E \in \mathcal{F}$ is such that $E \subseteq \Omega$.
- A probability function $\Pr : \mathcal{F} \to \mathbb{R}^+$

$E \in \mathcal{F}$ is said to be simple or elementar if $|E| = 1$

We will consider only discrete probability spaces.

Def.: *A probability function is any function* $\Pr : \mathcal{F} \to \mathbb{R}^+$ *such that*

- $\forall E \in \mathcal{F}$ we have $0 \leq \Pr(E) \leq 1$
- $\Pr(\Omega) = 1$
- For any finite or enumerable sequence of mutually disjoint events $E_1, E_2, \ldots,$ we have

$$\Pr(E_1 \cup E_2 \ldots) = \Pr(E_1) + \Pr(E_2) + \ldots$$

Example

Consider we roll a six-sided die.

- $\Omega = \{1, \ldots, 6\}$

Example of events we can consider, based on the number it shows

- E' = the number is even.
- E'' = the number is at most 3.
- E''' = the number is a prime number.

Def.: *Two events E and F are said to be independent iff*

$$\Pr(E \cap F) = \Pr(E) \cdot \Pr(F)$$

and events $E_1, \ldots, E_k$ are mutually independent iff $\forall I \subseteq \{1, \ldots, k\}$ we have

$$\Pr(\bigcap_{i \in I} E_i) = \Pi_{i \in I} \Pr(E_i).$$

Lemma: *(Union bound) Given events $E_1, E_2, \ldots$ we have*

$$\Pr(\bigcup_{i \geq 1} E_i) \leq \sum_{i \geq 1} \Pr(E_i)$$

Def.: *A discrete random variable (r.v.) over Ω is a function*

$$X : \Omega \to \{\lambda_1, \ldots, \lambda_n\}, \text{ where each } \lambda_i \text{ is a real number.}$$

Def.: *An event $(X \in S)$ is a set $X^{-1}(S)$ for $S \subseteq \{\lambda_1, \ldots, \lambda_n\}$.*

Notation: Given random variable X and real value a, the event "$X = a$" represents a set $\{e \in \Omega : X(e) = a\}$.
Analogously for $(X \neq x)$, $(X < x)$, $(X \leq x)$, $(X > x)$ and $(X \geq x)$.

Def.: *The probability of event $(X \in S)$ is the number $\Pr(X^{-1}(S))$, denoted by $\Pr(X \in S)$.*

So,

$$\Pr(X = a) = \sum_{e \in \Omega:\ X(e) = a} \Pr(e).$$

Def.: *The expectation of a random variable X is the number*

$$\mathbf{E}[X] := \sum_{i=1}^{n} \lambda_i \Pr(X=\lambda_i) \, .$$

Example

Consider the roll of two dice and let X_i the value of the i-th die, $i = 1, 2$. Let $X = X_1 + X_2$. Then

$$E[X] = 2\frac{1}{36} + 3 \cdot \frac{2}{36} + \cdots + 12 \cdot \frac{1}{36} = 7$$

Linearity of Expectation

Theorem: *For any finite set of discrete random variables $X_1, \dots, X_n$ with finite expectations*

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]$$

Observation: Note that there is no restrictions on the independence of the random variables $X_1, \dots, X_n$.

Lemma: *Given a random variable X and constant c, we have $E[c \cdot X] = c \cdot E[X]$.*

Example

Consider the example of the two dice.

Let X_1 the value of the first die and let X_2 the value of the second die.
Let X the r.v. of the sum of two dice.

Note that $X = X_1 + X_2$. So,

$$\begin{aligned}
E[X] &= E[X_1 + X_2] \\
&= E[X_1] + E[X_2] \\
&= 2 \cdot \sum_{i=1}^{6} i \cdot \frac{1}{6} \\
&= 7
\end{aligned}$$

Bernoulli and Binomial Random Variables

Consider the experiment that has success with probability p and fail with probability $1 - p$.

Let Y be a random variable such that

$$Y = \begin{cases} 1 & \text{the experiment has success} \\ 0 & \text{otherwise} \end{cases}$$

Then, Y is said to be a Bernoulli random variable.

Lemma: *If Y is a Bernoulli random variable with $\Pr(Y = 1) = p$, then $E[Y] = p$.*

Consider a sequence of n independent experiments, each one with probability of success equal to p.

If X is the number of successes in the n experiments, we say that X is a binomial r.v. and has binomial distribution.

Def.: *A binomial r.v. (b.r.v.) X with parameters n and p, denoted by $B(n, p)$ is defined by the following probability distribution with $j = 0, 1, \ldots, n$:*

$$\Pr(X = j) \ = \ \binom{n}{j} p^j (1 - p)^{n-j}$$

Exercise

Prove that $\sum_{j=0}^{n} \Pr(X = j) \ = \ 1$.

Lemma: *If X is a binomial r.v. $B(n, p)$, then $E[X] = n \cdot p$.*

Proof. Let X_i a bernoulli r.v. of the i-th experiment.

Then, $X = \sum_{i=1}^{n} X_i$ and therefore

$$\begin{aligned} E[X] &= \sum_{i=1}^{n} E[X_i] \\ &= n \cdot p \end{aligned}$$

$\square$

Geometric distribution

Def.: *A r.v. X is said to be geometric (g.r.v.) with parameter p if it has distribution*

$$\Pr(X = n) = (1 - p)^{n-1} \cdot p.$$

I.e., the probability to flip a coin $n - 1$ times with tail (or fail) and in the n-th we obtain head (success).

Lemma: *If X is a geometric r.v. with parameter p, then*

$$E[X] = \frac{1}{p}.$$

Proof. Exercise. □

Lemma: *(Markov inequality) If $(\Omega, \Pr)$ is a discrete probability space and X is a random variable over Ω with non-negative values, then*

$$\Pr(X \geq \lambda) \;\leq\; \frac{1}{\lambda}\,\mathbf{E}[X]$$

for any positive number λ.

Proof. If $\{\lambda_1, \ldots, \lambda_n\}$ is the set of all possible values of X, then

$$\begin{aligned} \mathbf{E}[X] &= \sum_i \lambda_i \Pr(X = \lambda_i) \\ &\geq \sum_{\lambda_i \geq \lambda} \lambda \Pr(X = \lambda_i) \\ &= \lambda \Pr(X \geq \lambda)\,. \end{aligned}$$

$\square$

Sometimes it is better to write the Markov inequality as:

$$\Pr(X \geq \lambda\,\mathbf{E}[X]) \leq \frac{1}{\lambda}.$$

Useful inequalities

Fact: *If $m \geq 1$ and $|t| \leq m$ then*

$$\left(1 + \frac{t}{m}\right)^m \leq e^t.$$

Example: If $n \geq 1$ then

$$\left(1 + \frac{1}{n}\right)^n \leq e \qquad \text{and} \qquad \left(1 - \frac{1}{n}\right)^n \leq \frac{1}{e}.$$

See more in
`http://dl.acm.org/citation.cfm?doid=242581.242585`

Randomized Approximation Algorithms

Def.: *Randomized algorithms: Algorithms that have access to random number generators.*

Consider the existence of a function RAND. Given $\rho \in [0, 1]$
RAND(ρ): return 1 with probability ρ and 0 with probability $1 - \rho$.

Def.: *A randomized algorithm has polynomial time if the number of calls to* RAND *and the consumption of the other operations is bounded by a polynomial in the input size.*

Def.: *Given an instance I and randomized algorithm A, let X_I be the random variable representing the value of the solution produced by A over I.*
We say that A is a randomized α-approximation if

$\mathbf{E}[X_I] \geq \alpha\,\mathrm{OPT}(I)$ *for maximization problems and*
$\mathbf{E}[X_I] \leq \alpha\,\mathrm{OPT}(I)$ *for minimization problems.*

Approximate Solutions with High Probability

Let $X_I \geq 0$ be a random variable representing the value of the solution produced by algorithm A over I for a minimization problem.
By the Markov inequality, we have

$$\begin{aligned}\Pr(X_I \geq (\alpha+\epsilon)\mathrm{OPT}(I)) &\leq \frac{\mathbf{E}[X_I]}{(\alpha+\epsilon)\mathrm{OPT}(I)}\\ &\leq \frac{\alpha\,\mathrm{OPT}(I)}{(\alpha+\epsilon)\mathrm{OPT}(I)}\\ &= \frac{\alpha}{\alpha+\epsilon}\,.\end{aligned}$$

Given $\lambda \in (0,1)$, let $k := \lceil \log_{\frac{\alpha}{\alpha+\epsilon}} \lambda \rceil$ and Y_I be the best result obtained applying A k times over I. Then

$$\Pr(Y_I \geq (\alpha+\epsilon)\mathrm{OPT}(I)) \leq \lambda.$$

λ small $\Rightarrow$ solutions within the approximation with good probability.

Probabilistic Rounding

Set Cover Problem: Given set E, subsets $\mathcal{S}$ of E, costs $c : \mathcal{S} \to \mathbb{Q}_{\geq}$, find covering $\mathcal{S}' \subseteq \mathcal{S}$ that minimizes $\sum_{S \in \mathcal{S}'} c(S)$.

Relaxation:

$$
(P) \qquad \begin{array}{rrcll}
\min & \sum_{S \in \mathcal{S}} c_S x_S & & & \\
 & \sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\
 & x_S & \geq & 0 & \forall S \in \mathcal{S},
\end{array}
$$

where $\mathcal{S}_e := \{S \in \mathcal{S} : \; e \in S\}$

Idea: Solve (P) and consider each x_S as a probability to obtain S.

MinCC-AP1 $(\mathcal{S}, E, c)$

```
1   let $\hat{x}$ (fractional) optimum solution for $(P)$
2   let $\mathcal{T} \leftarrow \emptyset$
3   for each $S \in \mathcal{S}$ do
4       include $S$ in $\mathcal{T}$ with probability $\hat{x}_S$
5   return $\mathcal{T}$
```

Lemma: *If $\mathcal{T}$ is produced by* MinCC-AP1, *then* $E[\sum_{S\in\mathcal{T}} c_S] \leq \mathrm{OPT}$

Proof.

$$\begin{aligned} E[\sum_{S\in\mathcal{T}} c_S] &= \sum_{S\in\mathcal{S}} c_S \cdot \Pr(S \text{ belongs to } \mathcal{T}) \\ &= \sum_{S\in\mathcal{S}} c_S \hat{x}_S \leq \mathrm{OPT} \end{aligned}$$

□

Lemma: *If $\mathcal{T}$ is produced by* MINCC-AP1 *and f is an element of E, then $\Pr(f$ is not covered by $\mathcal{T}) \leq \frac{1}{e} \approx 0,37$.*

Proof. Let $f \in E$ and w.l.o.g. consider $\mathcal{S}_f = \{S_1, \dots, S_t\}$ the sets that contain f.

$$\begin{aligned}
&\Pr(f \text{ is not covered by } \mathcal{T}) \\
&\qquad = \Pr(S_1 \notin \mathcal{T}) \cdot \Pr(S_2 \notin \mathcal{T}) \cdot \ldots \cdot \Pr(S_t \notin \mathcal{T}) \\
&\qquad = (1 - \hat{x}_{S_1}) \cdot (1 - \hat{x}_{S_2}) \cdot \ldots \cdot (1 - \hat{x}_{S_t}) \\
&\qquad \leq (1 - \frac{1}{t}) \cdot (1 - \frac{1}{t}) \cdot \ldots \cdot (1 - \frac{1}{t}) \\
&\qquad = (1 - \frac{1}{t})^t \\
&\qquad \leq \frac{1}{e}
\end{aligned}$$

□

Increasing the probability to obtain a covering
Monte Carlo Version

```
MinCC-AP ($\mathcal{S}, E, c$)
1   let $\mathcal{S}' \leftarrow \emptyset$ and $k = 2\log|E|$
2   for $i \leftarrow 1$ to $k$ do
3       $\mathcal{T}_i \leftarrow$ MinCC-AP1 ($\mathcal{S}, E, c$).
4       $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathcal{T}_i$
5   return $\mathcal{S}'$
```

Lemma: *If $\mathcal{S}'$ is produced by* MinCC-AP *then*
$E[c(\mathcal{S}')] \leq 2\log|E|\mathrm{OPT}$
Proof. Exercise. □

Lemma: *If $\mathcal{S}'$ is produced by* MinCC-AP *then* $\Pr(\mathcal{S}'$ *is a covering of* $E) \geq 1 - \frac{1}{|E|}$.

Proof. Given $f \in E$, we have that $\Pr(f$ is not covered by $\mathcal{T}_i) \leq \frac{1}{e}$.

So, $\Pr(f$ is not covered by $\mathcal{S}') \leq \left(\frac{1}{e}\right)^{2\log|E|} = \frac{1}{|E|^2}$.

Given $e \in E$, let $\mathcal{E}_e$ the event the element e is not covered by $\mathcal{S}'$.

$$\begin{aligned}
&\Pr(\mathcal{S}' \text{ is not a covering}) \\
&\quad = \Pr\Big(\bigcup_{e\in E} \mathcal{E}_e\Big) \\
&\quad \leq \sum_{e\in E} \Pr(\mathcal{E}_e) \\
&\quad \leq \sum_{e\in E} \frac{1}{|E|^2} \\
&\quad = \frac{1}{|E|}
\end{aligned}$$

□

Las Vegas Version

```
MinCC-AP-Las-Vegas ($\mathcal{S}, E, c$)
1   let $\mathcal{S}' \leftarrow \emptyset$ and $i \leftarrow 1$
2   while $\mathcal{S}'$ is not a covering, do
3       $\mathcal{T}_i \leftarrow$ MinCC-AP1 ($\mathcal{S}, E, c$).
4       $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathcal{T}_i$
5       $i \leftarrow i + 1$
6   return $\mathcal{S}'$
```

Theorem: *Show that the expected number of steps of algorithm* MinCC-AP-Las-Vegas *is $O(\log |E|)$.*
Proof. Exercise. Idea: Divide the sequence of iterations into subsequences of $2 \log |E|$ iterations: $S_1, S_2, \ldots$. Consider the possibility to stop in the sequence S_i as a geometric random variable. $\square$

Maximum Satisfiability - MaxSat

Def.: *A literal is a variable or its negation.*

Def.: *A clause C over boolean variables V, is a disjunction of literals, all associated to different variables.*

Given clause C over variables of V, we denote by

$C_0 \subseteq V$: the set of variables “negated”

$C_1 \subseteq V$: the set of variables “non-negated”.

Example: If C ia a clause $(a \vee \overline{b} \vee \overline{c})$ then, $C_1 = \{a\}$ and $C_0 = \{b, c\}$.

Def.: *An assignment of V ia a vector x indexed by V with values in $\{0, 1\}$ (or false/true).*

Def.: *An assignment x satisfies clause C if $x_v = 1$ for some $v \in C_1$ or $x_v = 0$ for some $v \in C_0$.*

Def.: *A boolean formula is in Conjunctive Normal Form (CNF) if the formula is a conjunction of clauses.*

Example of boolean formula in CNF

$$\phi = (a \vee \overline{b} \vee \overline{c}) \wedge (\overline{a} \vee \overline{c}) \wedge (a \vee b \vee d) \wedge (d \vee \overline{c}) \wedge (d \vee \overline{a})$$

Problem MAXSAT $(V, \mathcal{C})$ Given a colection $\mathcal{C}$ of clauses over set of variables V, find an assignment x of V that satisfies the maximum number of clauses of $\mathcal{C}$.

Theorem: *The* MAXSAT *problem is NP-hard.*

Johnson's Algorithm

Assign 0 or 1 to the variables with probability $\frac{1}{2}$

```
MAXSAT-JOHNSON ($V, \mathcal{C}$)
1   for each $v$ in $V$ do
2       $\dot{x}_v \leftarrow$ RAND ($\frac{1}{2}$)
3   return $\dot{x}$
```

where RAND (p), for $p \in [0, 1]$, return 1 with probability p and 0 with probability $1 - p$.

Theorem: *If X is a random variable with value equal to the number of satisfied clauses by the assignment produced by* MAXSAT-JOHNSON *and each clause has at least k variables, then*

$$\mathbf{E}[X] \geq \left(1 - \frac{1}{2^k}\right) \mathrm{OPT}(V, \mathcal{C}) .$$

Proof. For each clause C, define a random variable Z_C as :

$Z_C := 1$ if the assignment produced satisfy C and
$Z_C := 0$ otherwise.

$$\Pr(Z_C{=}0) \leq 1/2^k \quad \Rightarrow \quad \Pr(Z_C{=}1) \geq 1 - 1/2^k.$$

So,

$$\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}[\sum_{C \in \mathcal{C}} Z_C] \\
&= \sum_{C \in \mathcal{C}} \mathbf{E}[Z_C] \\
&\geq (1 - \frac{1}{2^k})|\mathcal{C}| \\
&\geq (1 - \frac{1}{2^k})\,\mathrm{OPT}(V, \mathcal{C})
\end{aligned}$$

□

Theorem: *The algorithm* MAXSAT-JOHNSON *is a randomized* 0.5*-approximation for* MAXSAT*.*

Proof. Straightforward from previous theorem, using $k = 1$. $\square$

Randomized Rounding Algorithm

Given value $\rho \in (0,1)$, round ρ with probabilityρ

$$(P) \qquad \begin{array}{rcccll} \max & \multicolumn{5}{l}{\sum_{C \in \mathcal{C}} z_C} \\ \multicolumn{2}{r}{\sum_{v \in C_0} (1 - x_v) + \sum_{v \in C_1} x_v} & \geq & z_C & & \forall\, C \in \mathcal{C}\,, \\ 0 & \leq & z_C & \leq & 1 & \forall\, C \in \mathcal{C}\,, \\ 0 & \leq & x_v & \leq & 1 & \forall\, v \in V\,. \end{array}$$

MaxSat-GW $(V, \mathcal{C})$

```
1  let $(\hat{x}, \hat{z})$ be an optimum (possible rational) solution of $(P)$
2  for each $v$ in $V$ do
3      $\dot{x}_v \leftarrow$ Rand $(\hat{x}_v)$
4  return $\dot{x}$
```

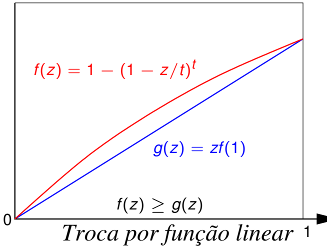
Theorem: *If $(V, \mathcal{C})$ is the instance of* MAXSAT, *each clause with at most k variables and $X_{\mathcal{C}}$ is a random variable which value is the number of satisfied clauses by an assignment produced by* MAXSAT-GW $(V, \mathcal{C})$, *then*

$$\mathbf{E}[X_{\mathcal{C}}] \ \geq \ \left(1 - (1 - \frac{1}{k})^k\right) \mathrm{OPT}(V, \mathcal{C})\,.$$

Proof. Let $C \in \mathcal{C}$ and Z_C be a variable with value 1 if assignment satisfy C and 0 otherwise. If t is the number of literals in C then

$$\begin{aligned}
\Pr(Z_C{=}1) &= 1 - \prod_{v \in C_0} \hat{x}_v \prod_{v \in C_1} (1 - \hat{x}_v) \\
&\geq 1 - \left(\frac{\sum_{v \in C_0} \hat{x}_v + \sum_{v \in C_1} (1 - \hat{x}_v)}{t} \right)^t \\
&= 1 - \left(\frac{t - \sum_{v \in C_0} (1 - \hat{x}_v) - \sum_{v \in C_1} \hat{x}_v}{t} \right)^t
\end{aligned}$$

$$
\begin{aligned}
&\geq 1 - \left(\frac{t - \hat{z}_C}{t}\right)^t \\
&= 1 - \left(1 - \frac{\hat{z}_C}{t}\right)^t \\
&\geq \left(1 - \left(1 - \frac{1}{t}\right)^t\right)\hat{z}_C \\
&\geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right)\hat{z}_C .
\end{aligned}
$$



□

Theorem: *The algorithm* MAXSAT-GW *is a randomized 0.63-approximation for the* MAXSAT.

Proof. Follows from the previous theorem together with the following inequality

$$\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e} \quad \forall k \geq 1$$

□

Combined Algorithm

MAXSAT-JOHNSON: Better performance when clauses are large

MAXSAT-GW: Better performance when clauses are small

Idea: Take the best solution between MAXSAT-JOHNSON and MAXSAT-GW

```
MAXSAT-COMBINED-GW ($V, \mathcal{C}$)
1   $\dot{x} \leftarrow$ MAXSAT-JOHNSON ($V$)
2   $\ddot{x} \leftarrow$ MAXSAT-GW ($V, \mathcal{C}$)
3   let $\dot{s}$ be the number of clauses of $\mathcal{C}$ satisfied by $\dot{x}$
4   let $\ddot{s}$ be the number of clauses of $\mathcal{C}$ satisfied by $\ddot{x}$
5   if $\dot{s} \geq \ddot{s}$
6       then return $\dot{x}$
7       else return $\ddot{x}$
```

Theorem: *The algorithm* MAXSAT-COMBINED-GW *is a randomized* 0.75*-approximation for* MAXSAT*.*

Proof. Let $\mathcal{C}_k := \{C \in \mathcal{C} : |C| = k\}$ (clauses with k variables),
Let $X_{\mathcal{C}}$ r.v. of solution produced by MAXSAT-COMBINED-GW.
Let $\dot{X}_{\mathcal{C}}$ r.v. of solution produced by MAXSAT-JOHNSON.
Let $\ddot{X}_{\mathcal{C}}$ r.v. of solution produced by MAXSAT-GW.

$$\begin{aligned}
\mathbf{E}[X_{\mathcal{C}}] &\geq \mathbf{E}[\frac{1}{2}(\dot{X}_{\mathcal{C}} + \ddot{X}_{\mathcal{C}})] = \frac{1}{2}(\mathbf{E}[\dot{X}_{\mathcal{C}}] + \mathbf{E}[\ddot{X}_{\mathcal{C}}]) \\
&\geq \frac{1}{2}\sum_{k}\sum_{C\in\mathcal{C}_k}\left((1-2^{-k}) + (1-(1-k^{-1})^k)\hat{z}_C\right) \\
&\geq \frac{1}{2}\sum_{k}\sum_{C\in\mathcal{C}_k}\left(1-2^{-k}+1-(1-k^{-1})^k\right)\hat{z}_C \\
&\geq \frac{1}{2}\sum_{k}\sum_{C\in\mathcal{C}_k}\frac{3}{2}\,\hat{z}_C \\
&\geq \frac{3}{4}\,\mathrm{OPT}(V,\mathcal{C})\,.
\end{aligned}$$

$\square$

Derandomization

Method of conditional expectations (Erdős & Selfridge' 74):

Idea:

- Compute conditional expectations efficiently
- Iteratively do the next choice without worsening the expectation.
- After all choices, we have a deterministic algorithm without worsening the expected value.

Example: Derandomization of the Johnson's Algorithm

```
MaxSat-Johnson ($V$)
1   for each $v$ in $V$ do
2       $\dot{x}_v \leftarrow \textsc{Rand}(\frac{1}{2})$
3   return $\dot{x}$
```

$$\begin{tikzcd}
 & & & E[X|x_1=0,x_2=0,x_3=0] \\
 & & E[X|x_1=0,x_2=0] \arrow[ur, dash] \arrow[dr, dash] & \\
 & & & E[X|x_1=0,x_2=0,x_3=1] \\
 & E[X|x_1=0] \arrow[uur, dash] \arrow[ddr, dash] & & \\
 & & & E[X|x_1=0,x_2=1,x_3=0] \\
 & & E[X|x_1=0,x_2=1] \arrow[ur, dash] \arrow[dr, dash] & \\
 & & & E[X|x_1=0,x_2=1,x_3=1] \\
E[X] \arrow[uuuur, dash] \arrow[ddddr, dash] & & & \\
 & & & E[X|x_1=1,x_2=0,x_3=0] \\
 & & E[X|x_1=1,x_2=0] \arrow[ur, dash] \arrow[dr, dash] & \\
 & & & E[X|x_1=1,x_2=0,x_3=1] \\
 & E[X|x_1=1] \arrow[uur, dash] \arrow[ddr, dash] & & \\
 & & & E[X|x_1=1,x_2=1,x_3=0] \\
 & & E[X|x_1=1,x_2=1] \arrow[ur, dash] \arrow[dr, dash] & \\
 & & & E[X|x_1=1,x_2=1,x_3=1]
\end{tikzcd}$$

Chose x_1, x_2 and x_3 such that

$$E[X] \leq E[X|x_1] \leq E[X|x_1, x_2] \leq E[X|x_1, x_2, x_3]$$

MaxSat-Johnson-Derandomized $(V, \mathcal{C})$

```
1   $\mathcal{D} \leftarrow \mathcal{C}$
2   for each $v$ in $V$ do
3       if EspCond $(v, 1, \mathcal{D}) \geq$ EspCond $(v, 0, \mathcal{D})$
4           then $\dot{x}_v \leftarrow 1$
5               for each $C$ in $\mathcal{D}$ do
6                   if $v \in C_1$
7                       then $\mathcal{D} \leftarrow \mathcal{D} \setminus \{C\}$
8                       else $C_0 \leftarrow C_0 \setminus \{v\}$
9           else $\dot{x}_v \leftarrow 0$
10              for each $C$ in $\mathcal{C}$ do
11                  if $v \in C_0$
12                      then $\mathcal{D} \leftarrow \mathcal{D} \setminus \{C\}$
13                      else $C_1 \leftarrow C_1 \setminus \{v\}$
14  return $\dot{x}$
```

Procedure to compute conditional expectation:

v: boolean variable
i: value assigned to variable v
$\mathcal{D}$: set of clauses

Procedure ESPCOND $(v, i, \mathcal{D})$

```
1   $esp \leftarrow 0$
2   for each $C$ in $\mathcal{D}$ do
3       $k \leftarrow |C_1| + |C_0|$
4       if $v \in C_i$
5           then $esp \leftarrow esp + 1$
6           else if $v \in C_{1-i}$
7                   then $esp \leftarrow esp + (1 - 2^{-k+1})$
8                   else $esp \leftarrow esp + (1 - 2^{-k})$
9   return $esp$
```

Theorem: *The algorithm* MAXSAT-JOHNSON-DERANDOMIZED *is a* 0.5*-approximation for* MAXSAT*.*

Proof. Let X be a random variable of the number of clauses in $\mathcal{C}$ satisfied by an assignment produced by MAXSAT-JOHNSON(V). As the probability of $\dot{x}_v = 1$ is $\frac{1}{2}$ and the probability of $\dot{x}_v = 0$ is $\frac{1}{2}$, we have

$$\begin{aligned}
\mathbf{E}[X] &= \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}1] + \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}0] \\
&\leq \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}i] + \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}i] \\
&= \mathbf{E}[X|\dot{x}_v{=}i] \;=\; \mathbf{E}[X|\dot{x}_v]
\end{aligned}$$

where i is the value chosen by the algorithm for variable $\dot{x}_v$. The same reasoning follows for other variables. I.e.,

$$\mathbf{E}[X] \leq \mathbf{E}[X|\dot{x}_{v_1}] \leq \mathbf{E}[X|\dot{x}_{v_1}, \dot{x}_{v_2}] \leq \ldots \leq \mathbf{E}[X|\dot{x}_{v_1}, \ldots, \dot{x}_{v_n}]\,.$$

As $0.5\,\mathrm{OPT}(V, \mathcal{C}) \leq \mathbf{E}[X]$ and the last term is a deterministic value, the theorem follows. $\square$

Duality in Linear Programming

Consider the following LP:

$$
\begin{array}{ll}
\text{minimize} & c_1x_1 + c_2x_2 + c_3x_3 \\
\text{sujeito a} & \left\{
\begin{array}{lcl}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 & \geq & b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & = & b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & \leq & b_3 \\
x_1 \geq 0, \qquad\qquad x_3 \leq 0 & &
\end{array}\right.
\end{array}
$$

Let us delimit the optimum value of the LP:
Multiply the constraints by $y_1 \geq 0$, y_2 and $y_3 \leq 0$:

$$
\begin{array}{lcl}
y_1(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & \geq & y_1b_1 \\
y_2(a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & = & y_2b_2 \\
y_3(a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & \geq & y_3b_3
\end{array}
$$

Summing these constraints, we obtain:

$$\begin{array}{lcl}
y_1(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & \geq & y_1 b_1 \\
y_2(a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & = & y_2 b_2 \\
y_3(a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & \geq & y_3 b_3 \\
\hline
(y_1 a_{11} + y_2 a_{21} + y_3 a_{31})x_1 + & & \\
(y_1 a_{12} + y_2 a_{22} + y_3 a_{32})x_2 + & & \\
(y_1 a_{13} + y_2 a_{23} + y_3 a_{33})x_3 & \geq & (y_1 b_1 + y_2 b_2 + y_3 b_3) = yb
\end{array}$$

Comparing with the objective function $cx = c_1 x_1 + c_2 x_2 + c_3 x_3$, if

$$c_1 x_1 \geq (y_1 a_{11} + y_2 a_{21} + y_3 a_{31})x_1$$

$$c_2 x_2 = (y_1 a_{12} + y_2 a_{22} + y_3 a_{32})x_2$$

$$c_3 x_3 \geq (y_1 a_{13} + y_2 a_{23} + y_3 a_{33})x_3$$

In these conditions, we have $cx \geq yb$ and therefore yb bound cx from below

As $x_1 \geq 0$ and $x_3 \leq 0$, we can simplify the conditions to have $cx \geq yb$ as:

$$\begin{aligned} c_1 &\geq y_1 a_{11} + y_2 a_{21} + y_3 a_{31} \\ c_2 &= y_1 a_{12} + y_2 a_{22} + y_3 a_{32} \\ c_3 &\leq y_1 a_{13} + y_2 a_{23} + y_3 a_{33} \end{aligned}$$

Clearly, between all values of y, we want the largest bound yb, i.e., with yb maximum. Therefore, we have the following dual program:

$$\begin{array}{ll} \text{maximize} & y_1 b_1 + y_2 b_2 + y_3 b_3 \\ \text{subject to} & \left\{ \begin{array}{l} y_1 a_{11} + y_2 a_{21} + y_3 a_{31} \leq c_1 \\ y_1 a_{12} + y_2 a_{22} + y_3 a_{32} = c_2 \\ y_1 a_{13} + y_2 a_{23} + y_3 a_{33} \geq c_3 \\ y_1 \geq 0, \qquad\qquad y_3 \leq 0 \end{array} \right. \end{array}$$

Convention to name these systems:

Primal Program

$$\text{minimize} \quad c_1x_1 + c_2x_2 + c_3x_3$$
$$\text{subject to} \quad \begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 & \geq & b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & = & b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & \leq & b_3 \\ x_1 \geq 0, \qquad x_3 \leq 0 & & \end{cases}$$

Dual Program

$$\text{maximize} \quad y_1b_1 + y_2b_2 + y_3b_3$$
$$\text{subject to} \quad \begin{cases} y_1a_{11} + y_2a_{21} + y_3a_{31} \leq c_1 \\ y_1a_{12} + y_2a_{22} + y_3a_{32} = c_2 \\ y_1a_{13} + y_2a_{23} + y_3a_{33} \geq c_3 \\ y_1 \geq 0, \qquad y_3 \leq 0 \end{cases}$$

Exercise

Perform the same analysis as before to obtain a dual program (as a bound to a linear program), but instead to start with a minimization program, start with a maximization program, and obtain a dual that is a minimization program.

Exercise

Obtain the dual formulation of the relaxed formulation of the following problems:

- *Vertex Cover Problem*
- *Set Cover Problem*
- *Facility Location Problem*
- *Steiner Tree Problem*

Primal and Dual Programs

Primal Program

$$
(P)\quad
\begin{array}{rrcll}
\text{minimize} & c\,x & & & \\
\text{subject to} & (Ax)_i & \geq & b_i & \text{for each } i \text{ in } M_1\,, \\
 & (Ax)_i & = & b_i & \text{for each } i \text{ in } M_2\,, \\
 & (Ax)_i & \leq & b_i & \text{for each } i \text{ in } M_3\,, \\
 & x_j & \geq & 0 & \text{for each } j \text{ in } N_1\,, \\
 & x_j & \leq & 0 & \text{for each } j \text{ in } N_3\,.
\end{array}
$$

Dual Program

$$
(D)\quad
\begin{array}{rrcll}
\text{maximize} & y\,b & & & \\
\text{subject to} & (yA)_j & \leq & c_j & \text{for each } j \text{ in } N_1\,, \\
 & (yA)_j & = & c_j & \text{for each } j \text{ in } N_2\,, \\
 & (yA)_j & \geq & c_j & \text{for each } j \text{ in } N_3\,, \\
 & y_i & \geq & 0 & \text{for each } i \text{ in } M_1\,, \\
 & y_i & \leq & 0 & \text{for each } i \text{ in } M_3\,.
\end{array}
$$

Lemma: *Let (P) be a primal minimization program and (D) its dual maximization program, then $cx \geq yb$ for any $x \in P$ and $y \in D$.*

Complementary Slackness Conditions

Vectors x and y, indexed by N and M resp., have complementary slackness condition if,

$x_j = 0 \quad \text{or} \quad (yA)_j = c_j \quad \forall j \in N_1 \cup N_3$
(primal complementary slackness)
and
$y_i = 0 \quad \text{or} \quad (Ax)_i = b_i \quad \forall i \in M_1 \cup M_3$ (dual complementary slackness).

Lemma: *(complementary slackness) If x and y are feasible solutions of a linear program (P) and its dual (D), resp., then*

$$(cx = yb) \;\Leftrightarrow\; (x \text{ and } y \text{ satisfy complementary slackness conditions})$$

Theorem: *[of duality] If* (P) *is a minimization linear program and* (D) *its dual (of maximization), then exactly one of the possibilities is valid:*

1. (P) *and* (D) *are feasible and* OPT-LP(P)=OPT-LP(D).
2. (P) *is feasible and* (D) *is unfeasible and* OPT-LP$(P) = -\infty$.
3. (P) *is unfeasible and* (D) *is feasible and* OPT-LP$(D) = \infty$.
4. (P) *and* (D) *are unfeasibles.*

Dual method: Rounding

“Rounding” a dual solution

If $\hat{x}$ and $\hat{y}$ are optimum solutions of (P) and (D), then

$$\hat{x}_j > 0 \quad \Rightarrow \quad (\hat{y}A)_j = c_j \qquad \text{(primal complementary slackness condition)}$$

Strategy:

Solve the dual program and select j where $(\hat{y}A)_j = c_j$.

Vertex Cover Problem: Given a graph $G = (V, E)$ and costs $c : V \to \mathbb{Q}_{\geq}$, find $S \subseteq V$ such that $\{u, v\} \cap S \neq \emptyset \quad \forall \{u, v\} \in E$ and $c(S)$ is minimum.

Primal and Dual Formulations

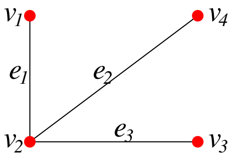
$$(P) \quad \begin{array}{rrcll} \min & \displaystyle\sum_{i \in V} c_i x_i & & & \\ & x_i + x_j & \geq & 1 & \forall ij \in E\,, \\ & x_i & \geq & 0 & \forall i \in V\,. \end{array}$$

$$(D) \quad \begin{array}{rrcll} \max & \displaystyle\sum_{e \in E} y_e & & & \\ & \displaystyle\sum_{e \in \delta(v)} y_e & \leq & c_v & \forall v \in V\,, \\ & y_e & \geq & 0 & \forall e \in E\,. \end{array}$$

MinCV-Hochbaum (G, c)

```
1   let $\hat{y}$ be an optimum solution of $(D)$
2   $C \leftarrow \{v \in V_G : \sum_{e \in \delta(v)} \hat{y}_e = c_v\}$
3   return $C$
```

Example:



$$
(P)\quad
\begin{array}{lrcl}
\min & x_{v_1}+x_{v_2}+x_{v_3}+x_{v_4} & & \\
 & x_{v_1}+x_{v_2} & \geq & 1 \\
 & x_{v_2}+x_{v_4} & \geq & 1 \\
 & x_{v_2}+x_{v_3} & \geq & 1 \\
 & x_v & \geq & 0
\end{array}
\qquad
(D)\quad
\begin{array}{lrcll}
\max & y_{e_1}+y_{e_2}+y_{e_3} & & & \\
 & y_{e_1} & \leq & 1 & (D_{v_1}) \\
 & y_{e_1}+y_{e_2}+y_{e_3} & \leq & 1 & (D_{v_2}) \\
 & y_{e_3} & \leq & 1 & (D_{v_3}) \\
 & y_{e_2} & \leq & 1 & (D_{v_4}) \\
 & y_e & \geq & 0 &
\end{array}
$$

Optimum solution value of $(D) \Rightarrow \hat{y}(E) = 1$

$\hat{y} = (\hat{y}_{e_1} = 1, \quad \hat{y}_{e_2} = 0, \quad \hat{y}_{e_3} = 0)$

Tight constraints of (D): (D_{v_1}) e (D_{v_2})

Approximate solution: $\{v_1, v_2\}$

Theorem: *The algorithm* MinCV-Hochbaum *produces a vertex covering.*
Proof. If $uv \in E$ then ($\sum_{e \in \delta(u)} \hat{y}_e = c_u$ or $\sum_{e \in \delta(v)} \hat{y}_e = c_v$) because $\hat{y}$ is optimum. □

Theorem: *Algorithm* MinCV-Hochbaum *is a polynomial time 2-approximation for* MinCV*.*
Proof. The covering C is such that

$$\begin{aligned} c(C) &= \sum_{v \in C} c_v = \sum_{v \in C} \sum_{e \in \delta(v)} \hat{y}_e \\ &\leq 2 \sum_{e \in E} \hat{y}_e \qquad (*) \\ &\leq 2\,\mathrm{OPT}(G, c)\,. \end{aligned}$$

(*) is valid because, $\hat{y}_e$ appear at most twice in the summation, for each edge e. □

Dual method: “Rounding” a dual maximal solution

Def.: *If $\tilde{y}$ is a feasible solution for the dual program, $\tilde{y}$ is maximal if there is no feasible solution y, with $y_i \geq \tilde{y}_i$ and*

$$\sum_{i=1}^{n} y_i > \sum_{i=1}^{n} \tilde{y}_i.$$

Strategy: “Rounding” a dual maximal solution

1. Find a dual maximal solution
2. Select the objects satisfied with equality in the dual

Set covering problem: Given set E, family $\mathcal{S}$ of E, costs $c : \mathcal{S} \to \mathbb{Q}_{\geq}$, find covering $\mathcal{S}' \subseteq \mathcal{S}$ that minimizes $\sum_{S \in \mathcal{S}'} c(S)$.

$$
\begin{array}{lll}
\min & \sum_{S \in \mathcal{S}} c_S x_S & \\
(P) & \sum_{S \in \mathcal{S}_e} x_S \geq 1 & \forall e \in E \\
 & x_S \geq 0 & \forall S \in \mathcal{S},
\end{array}
\qquad
\begin{array}{lll}
\max & \sum_{e \in E} y_e & \\
(D) & \sum_{e \in S} y_e \leq c_S & \forall S \in \mathcal{S} \\
 & y_e \geq 0 & \forall e \in E
\end{array}
$$

where $\mathcal{S}_e := \{S \in \mathcal{S} : e \in S\}$

MINCC-HOCHBAUM $(\mathcal{S}, E, c)$

```
1   let $\tilde{y}$ be a maximal dual solution of $(D)$
2   $\mathcal{S}' \leftarrow \{S \in \mathcal{S} : \sum_{e \in S} \tilde{y}_e = c_S\}$
3   return $\mathcal{S}'$
```

Let $\tilde{y}$ be a maximal vector of (D) and $\mathcal{S}'$ obtained from $\tilde{y}$ by algorithm MINCC-HOCHBAUM.

Lemma: *$\mathcal{S}'$ is a set covering.*

Proof. If $\mathcal{S}'$ is not a covering, then there exists a non-covered element $e \in E$. Thus, we can “increase $\tilde{y}_e$ until some dual constraint become tight. Contradicting the maximality of $\tilde{y}$. $\square$

Let $\beta := \max\{|\mathcal{S}_e| : e \in E\}$, where $\mathcal{S}_e = \{S \in \mathcal{S} : e \in S\}$.

Theorem: *MinCC-Hochbaum is a β-approximation for the set covering problem.*

Proof. Let $\mathcal{S}'$ be the set returned by the algorithm and $\tilde{y}$ the vector used in step 1 to produce $\mathcal{S}'$.

$$\begin{aligned}
c(\mathcal{S}') &= \sum_{S \in \mathcal{S}'} c(S) = \sum_{S \in \mathcal{S}'} \sum_{e \in S} \tilde{y}_e \\
&\leq \sum_{e \in E} |\mathcal{S}_e| \tilde{y}_e \\
&\leq \sum_{e \in E} \beta \tilde{y}_e = \beta \sum_{e \in E} \tilde{y}_e \\
&\leq \beta \sum_{e \in E} \hat{y}_e = \beta \, \mathrm{OPT\text{-}LP} \\
&\leq \beta \, \mathrm{OPT}
\end{aligned}$$

$\square$

Primal Dual Method

- Based on the complementary slackness conditions and feasibility problems
- Many times, leads to combinatorial algorithms

Classic primal dual method

$$(P)\quad \begin{array}{rcll} \min & cx & & \\ (Ax)_i & \geq & b_i & \forall i \in M, \\ x_i & \geq & 0 & \forall j \in N. \end{array} \qquad (D)\quad \begin{array}{rcll} \max & yb & & \\ (yA)_j & \leq & c_j & \forall j \in N, \\ y_i & \geq & 0 & \forall i \in M. \end{array}$$

Complementary Slackness

If $\hat{x}$ and $\hat{y}$ are optimum solutions of (P) and (D), then
$\hat{x}_j = 0$ or $(\hat{y}A)_j = c_j$ (primal complementary slackness conditions)
$\hat{y}_i = 0$ or $(A\hat{x})_i = b_i$ (dual complementary slackness conditions)

Strategy:

Given vector y feasible for (D),

- find feasible x for (P) satisfying complementary slackness with y
  or
- find y' such that $y'' \leftarrow y + y'$ is feasible with (D) and $y''b > yb$;
  repeat the process with y''

Lemma: *[of Farkas] Exactly one of the restricted programs is feasible:*

$$(RP)\quad \begin{array}{rcll} \exists\; x & \in & \mathbb{Q}^N & \\ (Ax)_i & \geq & b_i & \forall i \in M\,, \\ x_i & \geq & 0 & \forall j \in N\,. \end{array} \qquad (RD)\quad \begin{array}{rcll} \exists\; y & \in & \mathbb{Q}^M & \\ yb & > & 0 & \\ (yA)_j & \leq & 0 & \forall j \in N\,, \\ y_i & \geq & 0 & \forall i \in M\,. \end{array}$$

Proof.

Consider the linear program (P) and its dual (D):

$$(P)\quad \begin{array}{rcll} \min & 0\,x & & \\ (Ax)_i & \geq & b_i & \forall i \in M\,, \\ x_i & \geq & 0 & \forall j \in N\,. \end{array} \qquad (D)\quad \begin{array}{rcll} \max & yb & & \\ (yA)_j & \leq & 0 & \forall j \in N\,, \\ y_i & \geq & 0 & \forall i \in M\,. \end{array}$$

Note that (P) is feasible if and only if (RP) is feasible and program (D) is always feasible, as $y = 0$ is a feasible solution.

As (D) is feasible, only alternatives 1. or 3. of the Duality Lemma can occur.

Case 1: (P) is feasible and OPT-LP(P)=OPT-LP(D).

(P) is feasible $\Rightarrow$ (RP) is feasible.

$\left(\forall y \in (D),\ yb \leq \text{OPT-LP}(D) = \text{OPT-LP}(P) = 0\right) \Rightarrow (RD)$ is unfeasible.

Case 3: OPT-LP$(D) = \infty$ and (P) is unfeasible.

(P) unfeasible $\Rightarrow$ (RP) is unfeasible.

$\left(\text{OPT-LP}(D) = \infty\right) \Rightarrow \left(\exists y \in (D) :\ yb > 0\right) \Rightarrow (RD)$ is feasible. □

Complementary Slackness Conditions

If $\hat{x}$ and $\hat{y}$ are optimum solutions of (P) and (D), then

$\hat{x}_j = 0$ or $(\hat{y}A)_j = c_j$ (primal complementary slackness conditions)

$\hat{y}_i = 0$ or $(Ax)_i = b_i$ (dual complementary slackness conditions)

Given vector y feasible for (D),

$I(y) := \{i \in M : \ y_i = 0\}$ and $J(y) := \{j \in N : \ (yA)_j = c_j\}$.

$$\text{(Feasib.)} \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M \,, \\ x_i & \geq & 0 & \forall j \in N \,. \end{array} \; + \; \text{(C.S.)} \begin{array}{rcll} (Ax)_i & = & b_i & \forall i \in M \setminus I(y) \,, \\ x_j & = & 0 & \forall j \in N \setminus J(y) \,. \end{array}$$

Restricted Primal Problem

$$(RP) \quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in I(y) \,, \\ (Ax)_i & = & b_i & \forall i \in M \setminus I(y) \,, \\ x_j & \geq & 0 & \forall j \in J(y) \,, \\ x_j & = & 0 & \forall j \in N \setminus J(y) \,. \end{array}$$

From the Farkas Lemma, (RP) is feasilbe or (RD) is feasilbe, not both (exercise).

$$(RP)\begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in I(y)\,, \\ (Ax)_i & = & b_i & \forall i \in M \setminus I(y)\,, \\ x_j & \geq & 0 & \forall j \in J(y)\,, \\ x_j & = & 0 & \forall j \in N \setminus J(y)\,. \end{array} \bigoplus (RD)\begin{array}{rcll} y'b & > & 0 & \\ (y'A)_j & \leq & 0 & \forall j \in J(y)\,, \\ y'_i & \geq & 0 & \forall i \in I(y)\,. \end{array}$$

$I(y) := \{i \in M : y_i = 0\}$ and $J(y) := \{j \in N : (yA)_j = c_j\}$.

$$(D) \quad \begin{array}{rcll} \max & yb & & \\ (yA)_j & \leq & c_j & \forall j \in N, \\ y_i & \geq & 0 & \forall i \in M. \end{array}$$

$$(RD) \quad \begin{array}{rcll} y'b & > & 0 & \\ (y'A)_j & \leq & 0 & \forall j \in J(y), \\ y'_i & \geq & 0 & \forall i \in I(y). \end{array}$$

Lema: *If y is feasible for (D) and y' is feasible for (RD), then there exists $\theta > 0$ such that $y'' \leftarrow y + \theta\, y'$ is also feasible for (D).*

Prova. Exercise. □

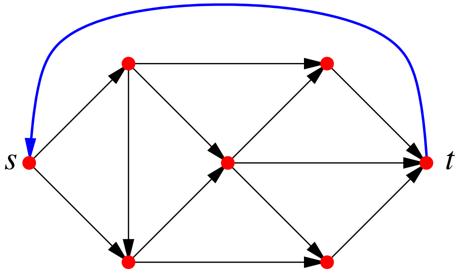
Method PRIMAL-DUAL (A, b, c)

```
1   let $y$ be a feasible vector of $(D)$
2   while $\mathrm{RP}(A, b, y)$ has no solution, do
3       let $y'$ be a solution of $\mathrm{RD}(A, b, y)$
4       if $y + \theta y'$ is solution of $(D)$ for any positive $\theta$
5           then return $y'$
6           else let $\theta$ maximum such that $y + \theta y'$ is feasible in $(D)$
7               $y \leftarrow y + \theta y'$
8   let $x$ be a solution of $\mathrm{RP}(A, b, y)$
9   return $x$ and $y$
```

Maximum Flow Problem

Maximum Flow Problem: Given a directed graph $G = (N, A)$, capacities $c : A \to \mathbb{N}$ and nodes s and t, find a flow of maximum value from s to t.

Simplification: add arc (t, s) with unbounded capacity.



Objective: Find flow that respects flow conservation in all nodes and maximize the edge flow in (t, s).

For convenience, we will denote the formulation by Dual

$$
(D)\quad
\begin{array}{rcll}
\max & y_{ts} & & \\
\sum_{e\in\delta^+(i)} y_e - \sum_{e\in\delta^-(i)} y_e & = & 0 & \forall i \in N, \\
y_{ij} & \leq & c_{ij} & \forall ij \in A, \\
y_{ij} & \geq & 0 & \forall ij \in A.
\end{array}
$$

Primal: find $x = x' \| x''$, x' indexed by N and x'' indexed by A such that

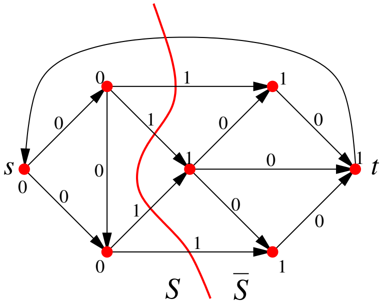
$$
(P)\quad
\begin{array}{rcll}
\min & \sum_{ij\in A} c_{ij} x''_{ij} & & \\
x'_t - x'_s & \geq & 1 & , \\
x'_i - x'_j + x''_{ij} & \geq & 0 & \forall ij \in A, \\
x''_{ij} & \geq & 0 & \forall ij \in A.
\end{array}
$$

About program *(P)*

$$(P)\quad \begin{array}{rcll} \min & \displaystyle\sum_{ij\in A} c_{ij}\, x''_{ij} & & \\ x'_t - x'_s & \geq & 1 & , \\ x'_i - x'_j + x''_{ij} & \geq & 0 & \forall ij \in A, \\ x''_{ij} & \geq & 0 & \forall ij \in A. \end{array}$$

Given a cut $S \subset N$ that separates s from t (i.e. $s \in S$ and $t \notin S$), define

$$x'_i = \begin{cases} 0 & \forall i \in S \\ 1 & \forall i \in N \setminus S \end{cases} \qquad x''_{ij} = \begin{cases} 1 & \forall ij \in \delta^+(S) \\ 0 & \forall ij \in A \setminus \delta^+(S) \end{cases}$$



We have $x = x' \| x''$ feasible to (P) that define a cut with capacity equal to the capacity given by the objective function.

Restricted problems

Given feasible flow y, let

$$I := \{ij \in A : \; y_{ij} = 0\} \qquad \text{e} \qquad J := \{ij \in A : \; y_{ij} = c_{ij}\} \,.$$

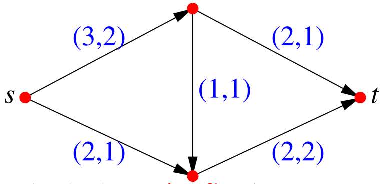
Restricted Primal: find $x = x' \| x''$ feasible in (P) satisfying C.S.

$$(RP) \quad \begin{array}{rcll} x'_t - x'_s & \geq & 1 & , \\ x'_i - x'_j + x''_{ij} & = & 0 & \forall ij \in A \setminus I, \\ x'_i - x'_j + x''_{ij} & \geq & 0 & \forall ij \in I, \\ x''_{ij} & \geq & 0 & \forall ij \in J, \\ x''_{ij} & = & 0 & \forall ij \in A \setminus J. \end{array}$$

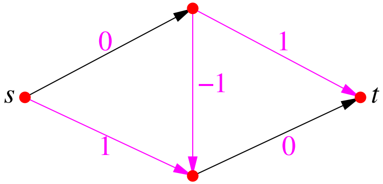
Restricted Dual: If (RP) is unfeasible, by Farkas Lemma, (RD) is feasible

$$(RD) \quad \begin{array}{rcll} y_{ts} & > & 0 & , \\ \sum_{e \in \delta^+(i)} y_e - \sum_{e \in \delta^-(i)} y_e & = & 0 & \forall i \in N, \\ y_{ij} & \leq & 0 & \forall ij \in J, \\ y_{ij} & \geq & 0 & \forall ij \in I. \end{array}$$

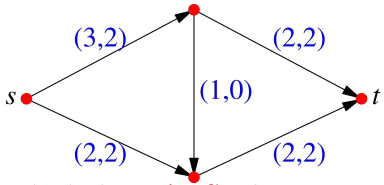
(RD) feasilbe $\Rightarrow \exists$ augmenting path $P = (y')$ from s to t, representing an additional flow.



(c,y)=(capacity,flow)
Value of initial flow = 3



Augmenting path
Flow in the path = 1



(c,y)=(capacity,flow)
Value of the resulting flow = 4

(RD) unfeasible $\Rightarrow$ (RP) is feasilbe

Let $S := \{v \in N : \text{ there exists augmenting path from } s \text{ to } v\}$ and

$$x'_i = \begin{cases} 0 & \forall i \in S \\ 1 & \forall i \in N \setminus S \end{cases} \qquad x''_{ij} = \begin{cases} 1 & \forall ij \in \delta^+(S) \\ 0 & \forall ij \in A \setminus \delta^+(S) \end{cases}$$

We have that $x = x' \| x''$ is feasible in (RP)

Primal-Dual Approximation Method

Generalization of the Primal-Dual Method by Approximate Complementary Slackness Conditions

$$(P)\quad \begin{array}{lrcll} \min & cx & & & \\ & (Ax)_i & \geq & b_i & \forall i \in M\,, \\ & x_j & \geq & 0 & \forall j \in N\,. \end{array} \qquad (D)\quad \begin{array}{lrcll} \max & yb & & & \\ & (yA)_j & \leq & c_j & \forall j \in N\,, \\ & y_i & \geq & 0 & \forall i \in M\,. \end{array}$$

Approximate Complementary Slackness Conditions

Given $0 < \alpha \leq 1 \leq \beta$, x and y feasible solutions of (P) and (D)

x and y have primal α-Approximate Slackness conditions if

$$x_j = 0 \quad \text{ou} \quad \alpha\, c_j \leq (yA)_j \quad [\leq c_j\,]$$

x and y have dual β-Approximate Slackness conditions if

$$y_i = 0 \quad \text{ou} \quad \beta\, b_i \geq (Ax)_i \quad [\geq b_i\,]$$

Lemma: *If x and y are non-negative and satisfy primal α-approximate slackness conditions and dual β-approximate slackness conditions, then $\alpha\, c\, x \leq \beta\, y\, b$.*

Proof.

$$\begin{aligned}
\alpha\, c\, x &= \sum_{j \in N} \alpha\, c_j\, x_j \\
&\leq \sum_{j \in N} (y\, A)_j\, x_j \\
&= \sum_{i \in M} y_i\, (A\, x)_i \\
&\leq \sum_{i \in M} y_i\, \beta\, b_i \\
&= \beta\, y\, b
\end{aligned}$$

□

Lemma: *(of Approximate Slacks) If x and y are feasible solutions of (P) and (D) satisfying α and β approximated slacks then*

x is a $\frac{\beta}{\alpha}$-approximation in (P) and
y is a $\frac{\alpha}{\beta}$-approximation in (D).

Idea:
Given vector y feasible for (D), values α and β, where $0 < \alpha \leq 1 \leq \beta$

- find feasible x for (P) satisfying α and β approximated slackness conditions with y
  or
- find y' st. $y'' \leftarrow y + y'$ is feasible in (D) and $y''b > yb$; repeat the process with y''

Approximate Slackness Conditions

$x_j = 0 \quad \text{or} \quad (yA)_j \geq \alpha\, c_j$ (primal approx. slack. conditions)

$y_i = 0 \quad \text{or} \quad (Ax)_i \leq \beta\, b_i$ (dual approx. slack. conditions)

Given vector y feasible for (D),

$I(y) := \{i \in M : \ y_i = 0\}$ e $J(y, \alpha) := \{j \in N : \ (yA)_j \geq \alpha\, c_j\}$.

$$\text{(Feas.)} \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M \\ x_j & \geq & 0 & \forall j \in N \end{array} + \text{(A.S.C.)} \begin{array}{rcll} (Ax)_i & \leq & \beta\, b_i & \forall i \in M \setminus I(y) \\ x_j & = & 0 & \forall j \in N \setminus J(y,\alpha) \end{array}$$

Approximate Restricted Primal

$$(ARP) \quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M\,, \\ (Ax)_i & \leq & \beta\, b_i & \forall i \in M \setminus I(y)\,, \\ x_j & \geq & 0 & \forall j \in J(y,\alpha)\,, \\ x_j & = & 0 & \forall j \in N \setminus J(y,\alpha)\,. \end{array}$$

(ARP) is unfeasible $\Rightarrow$ (RP) is unfeasible $\Rightarrow$ (RD) is feasible $\Rightarrow$ (ARD) is feasible.

$$(ARP)\quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M\,, \\ (Ax)_i & \leq & \beta\, b_i & \forall i \in M \setminus I(y)\,, \\ x_j & \geq & 0 & \forall j \in J(y,\alpha)\,, \\ x_j & = & 0 & \forall j \in N \setminus J(y,\alpha)\,. \end{array} \quad \vec{\oplus} \quad (ARD)\quad \begin{array}{rcll} y'b & > & 0 & \\ (y'A)_j & \leq & 0 & \forall j \in J(y, \\ y'_i & \geq & 0 & \forall i \in I(y) \end{array}$$

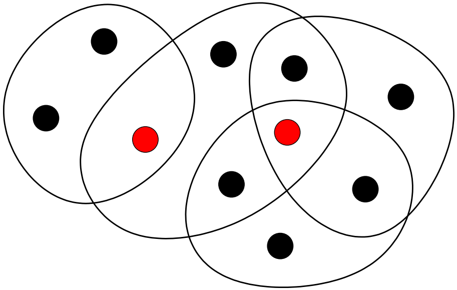
Method PRIMAL-DUAL (A, b, c)

```
1   let $y$ be a feasible solution for $(D)$
2   while $\mathrm{RAP}(A, b, y, \alpha, \beta)$ has no solution, do
3       let $y'$ be a solution for $\mathrm{RAD}(A, b, y, \alpha)$
4       if $y + \theta y'$ is feasible for $(D)$ for any positive $\theta$
5           then return $y'$
6           else let $\theta$ be maximum such that $y + \theta y'$ is feasilbe for $(D)$
7               $y \leftarrow y + \theta y'$
8   let $x$ be a solution for $\mathrm{RAP}(A, b, y, \alpha, \beta)$
9   return $x$ and $y$
```

Hitting Set Problem

Def.: *Given set E and finite collection $\mathcal{S}$ of subsets of E, a set $T \subseteq E$ is a hitting set (or transversal) of $\mathcal{S}$ if $T \cap S \neq \emptyset, \forall S \in \mathcal{S}$.*

Problema MINTC: Given set E, finite collection $\mathcal{S}$ of subsets of E and costs $c : E \to \mathbb{Q}_{\geq}$, find hitting set T of $\mathcal{S}$ such that $\sum_{e \in T} c_e$ is minimum.



Primal and Dual Formulations:

$$(P)\quad \begin{array}{rrcll} \min & \displaystyle\sum_{e \in E} c_e x_e & & & \\ & \displaystyle\sum_{e \in S} x_e & \geq & 1 & \forall S \in \mathcal{S}, \\ & x_e & \geq & 0 & \forall e \in E. \end{array} \qquad (D)\quad \begin{array}{rrcll} \max & \displaystyle\sum_{S \in \mathcal{S}} y_S & & & \\ & \displaystyle\sum_{S \in \mathcal{S}_e} y_S & \leq & c_e & \forall e \in E, \\ & y_S & \geq & 0 & \forall S \in \mathcal{S}, \end{array}$$

where $\mathcal{S}_e = \{S \in \mathcal{S} : \; e \in S\}$.

Idea: Obtain x and y feasible, with binary x, satisfying:

Primal Approximate Slackness Conditions:

$$x_e = 1 \Rightarrow \alpha c_e \leq \sum_{S \in \mathcal{S}_e} y_S \leq c_e \quad \text{for any } e \in E$$

Dual Approximate Slackness Conditions:

$$y_S > 0 \Rightarrow \beta \geq \sum_{e \in S} x_e \geq 1 \quad \text{for any } S \in \mathcal{S}$$

Approximate Restricted Primal

Consider $\alpha := 1$ and $\beta := \max\{|S| : S \in \mathcal{S}\}$. Let y, dual feasible and

$$I(y) := \{S \in \mathcal{S} : y_S = 0\} \quad \text{and} \quad J(y) := \{e \in E : \sum_{S \in \mathcal{S}_e} y_S \geq \alpha\, c_e\}$$

Approximate Restricted Primal: find x feasible in (P) satisfying Ap.Sl.

$$(ARP) \quad \begin{array}{rcll} \sum_{e \in E} x_e & \geq & 1 & \forall S \in \mathcal{S}, \\ \sum_{e \in S} x_e & \leq & \beta & \forall S \in \mathcal{S} \setminus I(y), \\ x_e & \geq & 0 & \forall e \in J(y), \\ x_e & = & 0 & \forall e \in E \setminus J(y). \end{array}$$

$$(ARD) \quad \begin{array}{rcll} \sum_{S \in \mathcal{S}} y'_S & > & 0 & , \\ \sum_{S \in \mathcal{S}} y'_S & \leq & 0 & \forall e \in J(y \\ y'_S & \geq & 0 & \forall S \in I(y \end{array}$$

Approximate Restricted Dual: If (ARP) is unfeasible then (RP) is unfeasible and from Farkas Lemma, (ARD) is feasible

Algorithm of Bar-Yehuda and Even

```
MinTC-BE ($E, \mathcal{S}, c$)
1   $J \leftarrow \{e \in E :\ c_e = 0\}$
2   $y_S \leftarrow 0,\ \forall S \in \mathcal{S}$
3   while there exists $R \in \mathcal{S}$ such that $J \cap R = \emptyset$ do
4       increase $y_R$ at the maximum, maintaining feasibility
5           $\sum_{S \in \mathcal{S}_e} y_S \leq c_e,\ \forall e \in R$
6       let $f$ be an element of $R$ such that $\sum_{S \in \mathcal{S}_f} y_S = c_f$
7       $J \leftarrow J \cup \{f\}$
8   return $J$
```

Lemma: *Let J be the set returned by* MINTC-BE*, x the characteristic vector of J and y the dual vector generated by the algorithm. Then,*

1. *J is a hitting set,*
2. *x satisfy primal app. sl. conditions with $\alpha = 1$*
$$x_e = 0 \quad \text{or} \quad 1 \cdot c_e \leq \sum_{S \in \mathcal{S}_e} y_S \leq c_e$$
3. *y satisfy dual app. sl. conditions with $\beta = \max\{|S| : S \in \mathcal{S}\}$:*
$$y_e = 0 \quad \text{or} \quad \beta \cdot 1 \geq \sum_{e \in S} x_e \geq 1$$

Proof.
To verify (1), note that if $R \in \mathcal{S}$ and $J \cap R = \emptyset$ then, there is a gap to increase y_R.
To verify (2), note that an element $e \in J$ was chosen to satisfy
$$\sum_{S \in \mathcal{S}_e} y_S = c_e$$
To verify (3), note that $\sum_{e \in S} x_e \leq |S| \leq \beta$. □

Theorem: *The algorithm* MINTC-BE *is a β-approximation for* MINTC $(E, \mathcal{S}, c)$*, where* $\beta := \max_{S \in \mathcal{S}} |S|$.

Proof. Follows from the Approximate Slackness Conditions. $\square$

Exercise

Using the same approach of the algorithm for the Hitting Set, present a 2-approximation for the vertex cover problem.

Facility Location Problem

FACILITY LOCATION PROBLEM: Given potential facilities $F = \{1, \ldots, n\}$, clients $C = \{1, \ldots, m\}$, costs f_i to "open" facility i and costs $c_{ij} \in \mathbb{Z}$ to connect client j to facility i. Find set of facilities $A \subseteq F$ minimizing the cost to open facilities in A and attend all clients

Applications: Install distribution warehouses, telecommunication network.

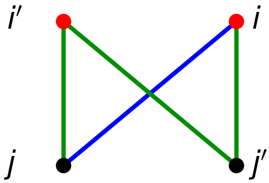
Teorema: *Problem* MINCC *is a particular case of the Facility Location.*

Corolário: *(Raz & Safra'97) The Facility Location Problem cannot be approximated in $\epsilon \log |E|$, for some constant $\epsilon > 0$, unless* $\mathrm{P} = \mathrm{NP}$.

Metric Facility Location Problem

Costs satisfy triangular inequality:

$$c_{ij} \leq c_{ij'} + c_{i'j'} + c_{i'j} \quad i, i' \in F \text{ e } j, j' \in C$$



We will se a 3-approximation from Jain and Vazirani'01 using primal dual technique:

Integer Program:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i\in F} f_i y_i + \sum_{ij\in E} c_{ij} x_{ij} \\ \text{subject to} & \left\{ \begin{array}{rcll} \displaystyle\sum_{ij\in E} x_{ij} & \geq & 1 & \forall j \in C, \\ y_i - x_{ij} & \geq & 0 & \forall ij \in E, \\ x_{ij} & \in & \{0,1\} & \forall i \in F \text{ and } j \in C, \\ y_i & \in & \{0,1\} & \forall i \in F. \end{array} \right. \end{array}$$

Relaxed Primal:

$$(P) \qquad \text{tal que} \quad \begin{array}{rcll} \min & \displaystyle\sum_{i\in F} f_i y_i + \sum_{ij\in E} c_{ij} x_{ij} & & \\ \displaystyle\sum_{ij\in E} x_{ij} & \geq & 1 & \forall j \in C\,, \\ y_i - x_{ij} & \geq & 0 & \forall i \in F,\ j \in C\,, \\ x_{ij} & \geq & 0 & \forall i \in F,\ j \in C\,, \\ y_i & \geq & 0 & \forall i \in F\,. \end{array}$$

Dual Program:

$$(D) \qquad \text{tal que} \quad \begin{array}{rcll} \max & \displaystyle\sum_{j\in C} \alpha_j & & \\ \alpha_j - \beta_{ij} & \leq & c_{ij} & \forall i \in F,\ j \in C\,, \\ \displaystyle\sum_{j\in C} \beta_{ij} & \leq & f_i & \forall i \in F\,, \\ \alpha_j & \geq & 0 & \forall j \in C\,, \\ \beta_{ij} & \geq & 0 & \forall i \in F,\ j \in C\,. \end{array}$$

Primal-Dual method with
primal approximate complementary slackness conditions:

$$\begin{array}{rcl} x_{ij} = 1 & \Rightarrow & \frac{1}{3}c_{ij} \leq \alpha_j - \beta_{ij} \leq c_{ij} \\ y_i = 1 & \Rightarrow & \frac{1}{3}f_i \leq \sum_{j\in C}\beta_{ij} \leq f_i \end{array}$$

dual complementary slackness conditions:

$$\begin{array}{rcl} \alpha_j > 0 & \Rightarrow & \sum_{i\in F} x_{ij} = 1 \\ \beta_{ij} > 0 & \Rightarrow & y_i = x_{ij} \end{array}$$

This give us a 3-approximation for the FACILITY LOCATION. We will prove a stronger result to also obtain an approximation algorithm for the k-MEDIAN problem.

Idea of the algorithm:

- Initially, start the dual variables with zero.
- Uniformly increase variables α's (maintaining feasibility) until a variable α_j is bounded by the condition of some edge *ij* ($\alpha_j - \beta_{ij} \leq c_{ij}$). In this case, the variable β_{ij} must also increase together.
- A facility *i* is tight when the sum of the values β incident to *i* is equal to its cost. At this point, associated dual variables stop increasing.
- The set of open facilities is a subset of the tight facilities.

FACILITY-JV (F,C,c)

Initialization

```
1  $A \leftarrow C$   % Set of active clients
2  $S \leftarrow \emptyset$   % Set of special edges
3  $R \leftarrow \emptyset$   % Set of inactive special edges
4  $F_t \leftarrow \emptyset$   % Temporary open facilities
5  for each $j$ in $C$ do $\alpha_j \leftarrow 0$
6  for each $i \in F$ and $j \in C$ do $\beta_{ij} \leftarrow 0$
```

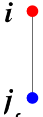
Phase 1

```
7   while $A \neq \emptyset$ do
8       increase uniformly $\alpha_j$ and $\beta_{ij}$ for $j \in A$ and $ij \in S \setminus R$ until
9               (a)  $\alpha_j = c_{ij}$ for some $ij \in (F \setminus F_t \times A) \setminus S$   or
10              (b)  $\alpha_j = c_{ij}$ for some $ij \in (F_t \times A) \setminus S$   or
11              (c)  $\sum_{j \in C} \beta_{ij} = f_i$ for some $i \in F \setminus F_t$.
12      if (a) is satisfied for some $ij$ then
13          $S \leftarrow S \cup \{ij\}$
14      else if (b) is satisfied for some $ij$ then
15          $\tau(j) \leftarrow i$   % witness of $j$
16          $A \leftarrow A \setminus \{j\}$
17      else if (c) was satisfied for some $i$ then
18          for each $j$ such that $ij \in S$ do $\tau(j) \leftarrow i$
19          $A \leftarrow A \setminus \{j : ij \in S\}$
20          $R \leftarrow R \cup \{ij : ij \in S\}$
21          $F_t \leftarrow F_t \cup \{i\}$
```

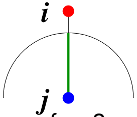
Phase 2

```
22    $H \leftarrow (V_H, E_H)$ where
23        $V_H := F_t$
24        $E_H := \{i'i'' :\ i', i'' \in V_H$ and $\exists j \in C,$ with $i'j \in S$
      and $i''j \in S\}$
25    $I \leftarrow$ (Maximal Independent Set of $H$)
26    for each $j \in C$ do
27        if $\tau(j) \in I$ then
28            $\phi(j) \leftarrow \tau(j)$    ( $(\phi(j), j)$ is direct connection)
29        else
30            let $i \in I$ such that $(i, \tau(j)) \in H$
31            $\phi(j) \leftarrow i$    ( $ij$ is undirected connection)
32    return $(I, \phi)$
```

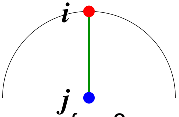
Increasing α_j and β_{ij} with one client



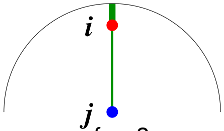
$f_i = 3$
$c_{ij} = 4$
$\alpha_j = 0$
$\beta_j = 0$
(1)



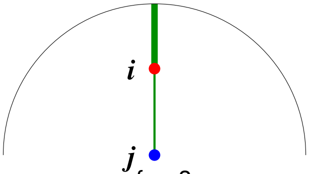
$f_i = 3$
$c_{ij} = 4$
$\alpha_j = 3$
$\beta_j = 0$
(2)



$f_i = 3$
$c_{ij} = 4$
$\alpha_j = 4$
$\beta_j = 0$
(3)



$f_i = 3$
$c_{ij} = 4$
$\alpha_j = 5$
$\beta_j = 1$
(4)



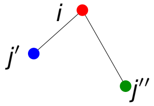
$f_i = 3$
$c_{ij} = 4$
$\alpha_j = 7$
$\beta_j = 3$
(5)

Increasing α_j and β_{ij} with two clients

$f_i = 3$

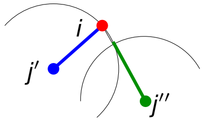
$c_{ij'} = 3$

$c_{ij''} = 4$



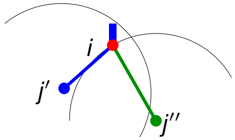
$\alpha_{j'} = 0$
$\alpha_{j''} = 0$
$\beta_{ij'} = 0$
$\beta_{ij''} = 0$

(1)



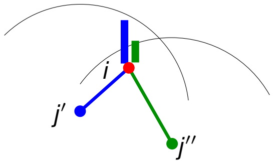
$\alpha_{j'} = 3$
$\alpha_{j''} = 3$
$\beta_{ij'} = 0$
$\beta_{ij''} = 0$

(2)



$\alpha_{j'} = 4$
$\alpha_{j''} = 4$
$\beta_{ij'} = 1$
$\beta_{ij''} = 0$

(3)



$\alpha_{j'} = 5$
$\alpha_{j''} = 5$
$\beta_{ij'} = 2$
$\beta_{ij''} = 1$

(4)

To prove that FACILITY-JV is a 3-approximation, we use the Approximate Complementary Slackness Condition Theorem with the following conditions:

primal approximate complementary slackness condition:

$$\begin{array}{lllcl} x_{ij}=1, & ij \text{ undirected} & \Rightarrow & \frac{1}{3}c_{ij} \le \alpha_j - \beta_{ij} & \le c_{ij} \\ x_{ij}=1, & ij \text{ direct} & \Rightarrow & \alpha_j - \beta_{ij} & = c_{ij} \\ y_i = 1 & & \Rightarrow & \sum_{j\in C}\beta_{ij} & = f_i \end{array}$$

e dual complementary slackness condition:

$$\begin{array}{lcrcl} \alpha_j > 0 & \Rightarrow & \sum_{i\in F} x_{ij} & = & 1 \\ \beta_{ij} > 0 & \Rightarrow & y_i & = & x_{ij} \end{array}$$

Note that the only *gap* is due to the undirected edges.

Lemma: *(Primal approx. compl. slack. cond. 1)*

$$x_{ij} = 1 \text{ and } ij \text{ is direct} \Rightarrow \alpha_j - \beta_{ij} = c_{ij}$$

Proof.

Case 1: $\beta_{ij} = 0$

In this case, α_j increased only to cover the cost c_{ij} as i was already opened.

Case 2: $\beta_{ij} > 0$

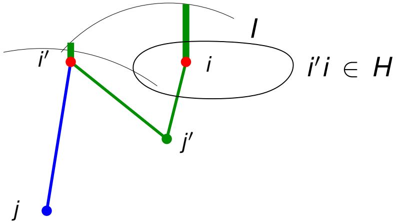
In this case, α_j may be larger than c_{ij}, and when this occur, ij becomes special and β_{ij} increase together with α_j maintaining the equality $\alpha_j - \beta_{ij} = c_{ij}$.

□

Lemma: *(Primal approx. compl. slack. cond. 2)*

$$x_{ij} = 1 \text{ and } ij \text{ is undirected} \Rightarrow \frac{1}{3} c_{ij} \leq \alpha_j - \beta_{ij} \leq c_{ij}$$

Proof. In this case, note that $\beta_{ij} = 0$ and $\alpha_j \leq c_{ij}$.
Now, we show that $c_{ij} \leq 3\alpha_j$.
Let $i' := \tau(j)$. There must exist $i' \in I$ and $j' \in C$ such that



$$\begin{aligned} c_{ij} &\leq c_{i'j} + c_{i'j'} + c_{ij'} \\ &\leq \alpha_j + \alpha_{j'} + \alpha_{j'} \\ &\leq \alpha_j + \alpha_j + \alpha_j \quad \text{(as radius of } j \text{ stoped increasing before of } j'\text{)} \\ &= 3\alpha_j \end{aligned}$$

□

Lemma: *(Primal approx. compl. slack. cond. 3)*

$$y_i = 1 \Rightarrow \sum_{j \in C} \beta_{ij} = f_i$$

Proof. Each temporary open facility (in F_t) must satisfy the condition (c) (line 11) by the command of line 21:

$$\sum_{j \in C} \beta_{ij} = f_i$$

The result follows, as the set I of open facilities ($y_i = 1$) is a subset of F_t. □

Lemma: *(Dual approx. compl. slack. cond. 1)*

$$\alpha_j > 0 \Rightarrow \sum_{i \in F} x_{ij} = 1.$$

Proof. Exercise. □

Lemma: *(Dual approx. compl. slack. cond. 2)*

$$\beta_{ij} > 0 \Rightarrow y_i = x_{ij}$$

Proof. Exercise (it is sufficient to consider the cases when $y_i = 0$ and $y_i = 1$). □

Theorem: *(Jain e Vazirani'01) FACILITY-JV is a 3-approximation for the Facility Location Problem.*

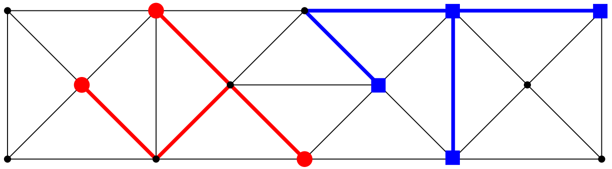
Proof. Follows from the Approximate Complemantary Slackness Condition Lemma and the 5 previous lemmas. □

Steiner Forest Problem

Given a G, let $\mathcal{R}$ a collection of subsets of V_G.

Def.: *A $\mathcal{R}$-forest of G is any spanning forest F of G such that for each $R \in \mathcal{R}$, the elements of R are contained in some component of F.*

Steiner Forest Problem: Given a graph G, costs c_e in $\mathbb{Q}_{\geq}$ for each edge e and a collection $\mathcal{R}$ of subsets of V_G, find a $\mathcal{R}$-forest F that minimize $c(F)$.



Def.: *A set $S \subset V$ is said active if there exits $R \in \mathcal{R}$ such that*

$$R \cap S \neq \emptyset \qquad e \qquad R \setminus S \neq \emptyset$$

$\mathcal{S} := \{S \subset V : \ S \text{ is an active set}\}$ *and*
$\mathcal{S}_e := \{S \in \mathcal{S} : \ e \in \delta(S)\}$

Primal and dual programs:

$$(P) \quad \begin{array}{rlll} \min & \sum_{e \in E} c_e x_e & & \\ & \sum_{e \in \delta(S)} x_e \geq 1 & \forall S \in \mathcal{S}\,, & \\ & x_e \geq 0 & \forall e \in E\,. & \end{array} \qquad (D) \quad \begin{array}{rlll} \max & \sum_{S \in \mathcal{S}} y_S & & \\ & \sum_{S \in \mathcal{S}_e} y_S \leq c_e & \forall e \in & \\ & y_S \geq 0 & \forall S \in & \end{array}$$

If there exists x and y feasible to (P) and (D), x integer, satisfying α and β approximate conditions, with $\alpha = 1$ and $\beta = 2$, then

$$x_e = 1 \quad \Rightarrow \quad \sum_{S \in \mathcal{S}_e} y_S = c_e \qquad \text{(primal approximate conditions)}$$

$$y_S > 0 \quad \Rightarrow \quad \sum_{e \in \delta(S)} x_e \leq 2 \qquad \text{(dual approximate conditions)}$$

From Approx. Compl. Slack. Cond., x is a 2-approximation. But

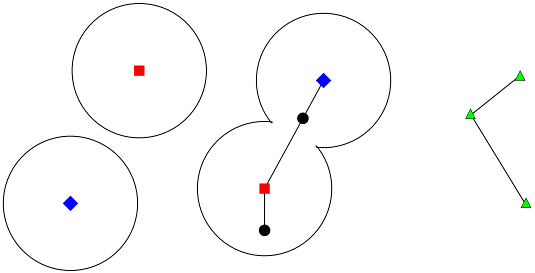
- Primal approximate conditions are easy to obtain.
- Dual approximate conditions are very restrictive.

Idea: Use a more flexible dual approximate condition

Change $$y_S > 0 \quad \Rightarrow \quad \sum_{e \in \delta(S)} x_e \leq 2$$

by $$y_S > 0 \quad \Rightarrow \quad \frac{\displaystyle\sum_{S \in \mathcal{S}_F} \sum_{e \in \delta(S)} x_e}{|\mathcal{S}_F|} \leq 2$$

where $\mathcal{S}_F$ is the set of active components of the forest being built, in each iteration of the algorithm.



Active components in each iteration

Observations about the algorithm:

- Build a dual feasible solution starting from 0.
- Use limited number of active sets.
- Active sets have laminar property.
- Primal approximate conditions are valid in each iteration.
- “Flexible” dual approximate conditions valid in each iteration.
- At the end of the algorithm, unnecessary edges are pruned.

F: Forest being constructed.

$\mathcal{S}_F$: Active components of F

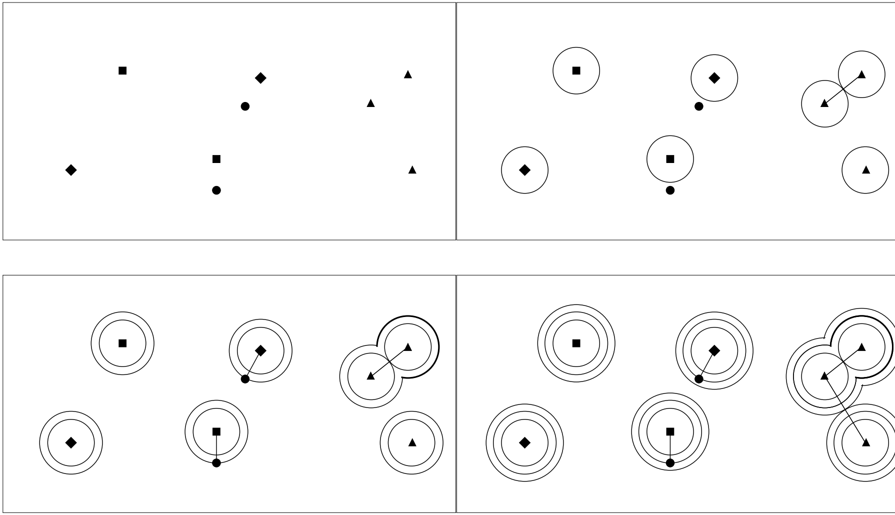
External edge: exactly one of the extremes in a component of $\mathcal{S}_F$

F_0: Forest F before pruning step.

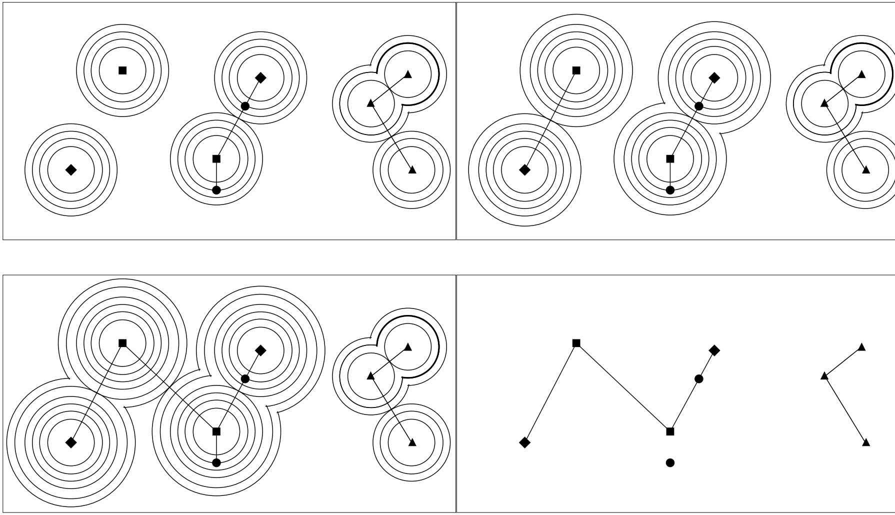
F_1: Final forest after pruning.

MINFS-GW $(G, c, \mathcal{R})$

```
$F \leftarrow (V, \emptyset)$
for each $S$ in $\mathcal{S}$ do $y_S \leftarrow 0$
while $\mathcal{S}_F \neq \emptyset$ do
    increase $y_S$ uniformly to the maximum, $\forall S \in \mathcal{S}_F$, restricted t
        $\sum_{S \in \mathcal{S}_e} y_S \leq c_e$ for each edge $e$
    let $f$ be an external edge such that $\sum_{S \in \mathcal{S}_f} y_S = c_f$
    $F \leftarrow F + f$
$F_0 \leftarrow F$
let $F_1$ be a minimal $\mathcal{R}$-forest of $F_0$
return $F_1$
```



Circles represent the increasing of dual variables.
Points are Steiner nodes (do not have connectivity requirements).



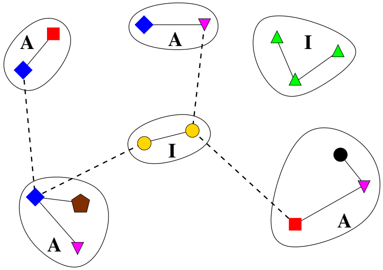
Inactive components do not have dual variables to increase. At last, algorithm perform the pruning step (remove unnecessary edges).

Lemma: *In the beginning of each iteration we have*

$$\frac{\sum_{S\in\mathcal{S}_F}|\delta_{F_1}(S)|}{|\mathcal{S}_F|} \le 2\,.$$

Proof. Let $\mathcal{C}$ be the set of components of F in the beginning of an iteration. Let $H = (\mathcal{C}, E_H)$ such that

$$\{U,W\}\in E_H \Leftrightarrow \exists\{u,w\}\in F_1 : u\in U, w\in W.$$



As F_1 is minimal, all leaves of H are active.

Let

$\mathcal{S}_F$ the active components of F and

$\mathcal{Z}_F$ the inactive components of F with non-null degree.

As H is a forest, we have

$$\begin{aligned}
\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)| + \sum_{S \in \mathcal{Z}_F} |\delta_{F_1}(S)| &= \sum_{S \in V_H} |\delta_H(S)| \\
&= 2|E_H| \\
&\le 2(|\mathcal{S}_F| + |\mathcal{Z}_F| - 1) \\
&< 2|\mathcal{S}_F| + 2|\mathcal{Z}_F|
\end{aligned}$$

Therefore

$$\begin{aligned}
\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)| &\le 2|\mathcal{S}_F| + 2|\mathcal{Z}_F| - \sum_{S \in \mathcal{Z}_F} |\delta_{F_1}(S)| \\
&\le 2|\mathcal{S}_F| \,.
\end{aligned}$$

□

Lemma: *In the beginning of each iteration, we have*

$$\sum_{S \in \mathcal{S}} |\delta_{F_1}(S)|\, y_S \;\leq\; 2 \sum_{S \in \mathcal{S}} y_S \,,$$

Proof. By induction in the number of iterations:
Initially $y = 0$ and the inequality is valid.
Suppose the inequality is valid in the beginning of a iteration.
During the iteration, y_S is increased by θ if and only if $S \in \mathcal{S}_F$.
So, the left hand side of the inequality is increased by

$$\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)|\theta$$

while the right hand side is increased by

$$2|\mathcal{S}_F|\theta.$$

The inequality follows from the previous lemma. □

Theorem: *The algorithm* MINFS-GW *is a 2-approximation for* MINFS.

Proof.

$$\begin{aligned}
c(F_1) &= \sum_{e \in F_1} c_e \\
&= \sum_{e \in F_1} \sum_{S \in \mathcal{S}_e} y_S && \text{(1-primal app. cond.)} \\
&= \sum_{S \in \mathcal{S}} |\delta_{F_1}(S)|\, y_S && \text{(inverting sums)} \\
&\leq 2 \sum_{S \in \mathcal{S}} y_S && \text{(from previous lemma)} \\
&\leq 2\, \mathrm{OPT}(G, c, \mathcal{R})\, .
\end{aligned}$$

□

Exercise

Show that it is possible to improve the analysis of the theorem and obtain an approximation factor of $\left(2 - \frac{1}{n}\right)$*, instead of 2.*

Dual Fitting

Suppose (P) is a problem of type $\left(\min cx \text{ subject to } x \in \mathcal{P}\right)$, and (D) is its dual $\left(\max yb \text{ subject to } y \in \mathcal{D}\right)$.

Idea:

- Obtain an integer solution x to (P) and a vector $\tilde{y}$ (related to the dual) but $\tilde{y}$ is not necessary feasible for (D).
- Although $\tilde{y}$ is not feasible, $\tilde{y}b$ "pay" the value of the primal solution.
- Obtain factor f such that $y \leftarrow \dfrac{\tilde{y}}{f}$ is feasible for (D).
- So, $cx \le \tilde{y}\,b = f\,\dfrac{\tilde{y}\,b}{f} = f\,y\,b \le f\,\mathrm{OPT}$.

Set Cover Problem: Given set E, subsets $\mathcal{S}$ of E, costs $c : \mathcal{S} \rightarrow \mathbb{Q}_{\geq}$, find cover $\mathcal{S}' \subseteq \mathcal{S}$ that minimizes $\sum_{S \in \mathcal{S}'} c(S)$.

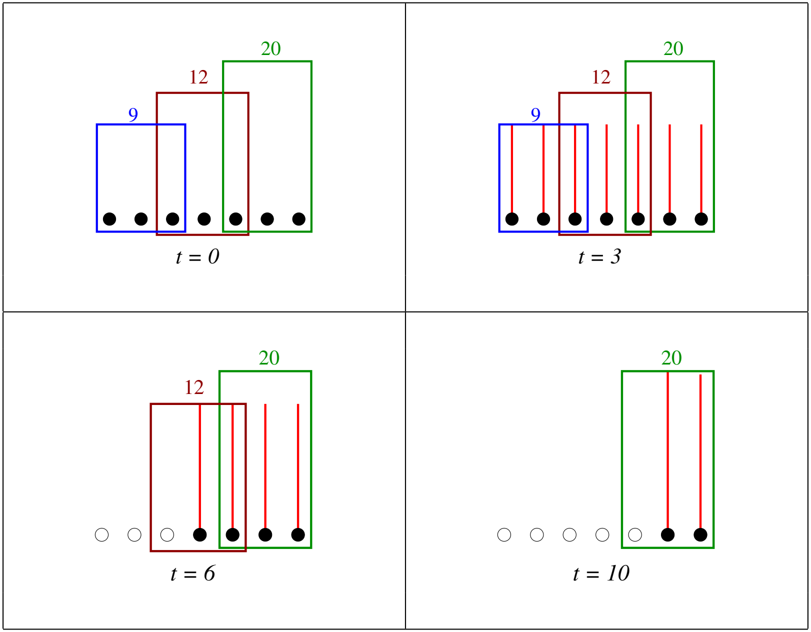
Relaxation and dual:

$$
\begin{array}{lllll}
\min & \sum_{S \in \mathcal{S}} c_S x_S & & & \\
(P) & \sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\
 & x_S & \geq & 0 & \forall S \in \mathcal{S},
\end{array}
\qquad
\begin{array}{lllll}
\max & \sum_{e \in E} y_e & & & \\
(D) & \sum_{e \in S} y_e & \leq & c_S & \forall S \in \mathcal{S} \\
 & y_e & \geq & 0 & \forall e \in E
\end{array}
$$

where $\mathcal{S}_e := \{S \in \mathcal{S} : \; e \in S\}$

Idea: Increase (from 0) the variable $\tilde{y}$ in such a way to pay for an integer solution. Then, obtain a factor f that transform $\tilde{y}/f$ in a dual feasible vector.

Increase $\tilde{y}_e$ uniformly, for each active element $e \in A$



```
MinCC-Dual-Fitting ($\mathcal{S}, E, c$)
1   let $\mathcal{S}' \leftarrow \emptyset$
2   let $A \leftarrow E$   (set of active elements)
3   let $\tilde{y}_e \leftarrow 0$ para todo $e \in E$
4   while $A \neq \emptyset$ do
5       increase $\tilde{y}_e$ uniformly for each $e \in A$ until
            $\sum_{e \in R \cap A} \tilde{y}_e = c_R$   for some   $R \in \mathcal{S} \setminus \mathcal{S}'$
6       $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{R\}$
7       $A \leftarrow A \setminus R$
8   return $(\mathcal{S}', \tilde{y})$
```

Lemma: *Let $(\mathcal{S}', \tilde{y})$ the solution returned by* MinCC-Dual-Fitting*. Then,*

$$\sum_{S \in \mathcal{S}'} c_S = \sum_{e \in E} \tilde{y}_e$$

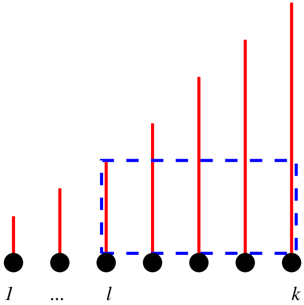
Proof. Exercise. □

Fact: *Note that $\tilde{y}$ can be unfeasible for (D).*
Proof. exercise □

Lemma: *Let $(\mathcal{S}', \tilde{y})$ the pair returned by* MINCC-DUAL-FITTING *and $S = \{e_1, \ldots, e_k\} \in \mathcal{S}'$, w.l.o.g. with $\tilde{y}_{e_1} \leq \tilde{y}_{e_2} \leq \ldots \leq \tilde{y}_{e_k}$. Then*

$$\sum_{i=l}^{k} \tilde{y}_{e_l} = (k - l + 1)\tilde{y}_{e_l} \leq c(S)$$

Proof. Note that if $\tilde{y}_{e_l}$ increased until a value t, all the other values $\tilde{y}_{e_j}$ for $j = l, \ldots, k$ also reached t, without violating the cost of S.



□

Lemma: *If $\tilde{y}$ is returned by* MinCC-Dual-Fitting *then* $\frac{\tilde{y}}{H_n}$ *is dual feasible for* (D)*, where* $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

Proof. Let $R = \{e_1, \ldots, e_k\} \in \mathcal{S}'$ and w.l.o.g. $\tilde{y}_{e_1} \le \tilde{y}_{e_2} \le \ldots \le \tilde{y}_{e_k}$. Applying the previous lemma for $l = 1, \ldots, k$ we have

$$\begin{cases} l = 1, & k\,\tilde{y}_{e_1} \le c(R) & \Rightarrow & \frac{\tilde{y}_{e_1}}{c(R)} \le \frac{1}{k} \\ l = 2, & (k-1)\,\tilde{y}_{e_2} \le c(R) & \Rightarrow & \frac{\tilde{y}_{e_2}}{c(R)} \le \frac{1}{k-1} \\ & & \vdots & \\ l = k, & \tilde{y}_{e_k} \le c(R) & \Rightarrow & \frac{\tilde{y}_{e_k}}{c(R)} \le 1 \end{cases}$$

Summing the above inequalities

$$\sum_{i=1}^{k} \frac{\tilde{y}_{e_i}}{c(R)} \le H_k$$

$$\text{So, } \sum_{e \in R} \frac{\tilde{y}_e}{H_n} \le c(R).$$

□

Theorem: *The algorithm* MINCC-DUAL-FITTING *is a* H_n*-approximation for the Set Cover problem.*

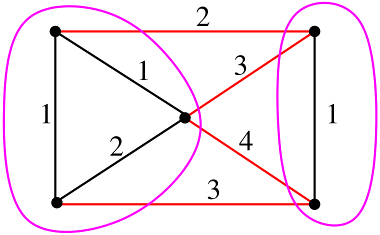
Exercise

Rewrite the algorithm MINCC-DUAL-FITTING *as a greedy algorithm.*

Semidefinite Programming

MaxCut Problem: Given a graph $G = (V, E)$ and weight w_e in $\mathbb{Q}_{\geq}$ for each edge e, find a cut R that maximize $w(R)$.

Exemplo:

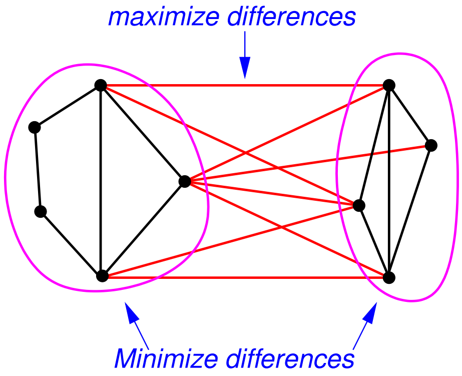


Theorem: MAXCUT *is NP-hard.*

Theorem: *If P≠NP then* MAXCUT *is not approximable within* $16/17 - \epsilon$ *in polynomial time (Håstad'97).*

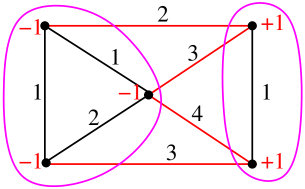
Application: Partition by similarity

w_{ij} = difference between i and j.
i
j



Quadratic Formulation: Find x that

$$\begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - x_i x_j) \\ & x_i x_i = 1 \qquad \forall i \in V \end{array}$$



If $x_i x_j = -1$

$$w_{ij}(1 - x_i x_j) = w_{ij}(1 - (-1)) = 2w_{ij}$$

If $x_i x_j = +1$

$$w_{ij}(1 - x_i x_j) = w_{ij}(1 - (+1)) = 0$$

Vector Program Relaxation

- Replace (relax) x_i by vector $Y_{i\star}$ (n-dimensional)

$$\boxed{x_i} \Longrightarrow \boxed{\qquad Y_{i\star} \qquad}$$

- I.e., replace x by matrix Y.

$$\begin{array}{c} x \\ \hline \multicolumn{1}{|c|}{x_1} \\ \hline \multicolumn{1}{|c|}{x_2} \\ \hline \multicolumn{1}{|c|}{\vdots} \\ \hline \multicolumn{1}{|c|}{x_n} \\ \hline \end{array} \Longrightarrow \begin{array}{c} Y \\ \hline \multicolumn{1}{|c|}{\qquad Y_{1\star} \qquad} \\ \hline \multicolumn{1}{|c|}{Y_{2\star}} \\ \hline \multicolumn{1}{|c|}{\vdots} \\ \hline \multicolumn{1}{|c|}{Y_{n\star}} \\ \hline \end{array}$$

- $Y_{i\star} \cdot Y_{i\star} = 1 \quad \Rightarrow \quad Y_{i\star}$ is a vector in the unit sphere

$x_i \cdot x_j \Rightarrow Y_{i\star} \cdot Y_{j\star} = (YY^\top)_{ij}$

$$(YY^\top)_{ij} = \left(\begin{array}{|c|} \hline \vdots \\ \hline Y_{i\star} \\ \hline \vdots \\ \hline \end{array} \bullet \begin{array}{|c|c|c|} \hline \ldots & Y_{j\star} & \ldots \\ \hline \end{array} \right)_{ij}$$

Quadratic Formulation:

$$(Q) \qquad \begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - x_i x_j) \\ & x_i x_i = 1 \qquad \forall i \in V \end{array}$$

$$\Downarrow$$

Relaxation:

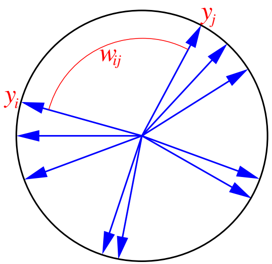
$$(R) \qquad \begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - (Y Y^\top)_{ij}) \\ & (Y Y^\top)_{ii} = 1 \qquad \forall i \in V \end{array}$$

Interpretation

Let $y_i = Y_{i\star}$

Relaxation:

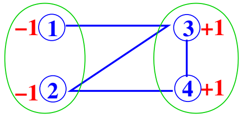
$$(R) \qquad \begin{array}{lll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - y_i \cdot y_j) & \\ & y_i \cdot y_i = 1 & \forall i \in V \\ & y_i \in \mathbb{R}^n & \forall i \in V \end{array}$$



w_{ij} is the repulsion force between y_i and y_j

Vectorial Relaxation

$$x = \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline -1 & -1 & +1 & +1 \end{array}$$



$$Y = \begin{array}{c|cccc} 1 & -1 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & 0 \\ 3 & +1 & 0 & 0 & 0 \\ 4 & +1 & 0 & 0 & 0 \end{array} \qquad Y^{T} = \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline -1 & -1 & +1 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}$$

$$x_i x_j = (YY^T)_{ij} \qquad \text{and} \qquad (YY^T)_{ii} = 1$$

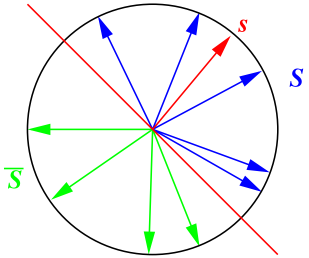
I.e.,

$$\text{Max } \frac{1}{2} \sum_{ij \in E} w_{ij}(1 - (Y\,Y^{\top})_{ij}) \geq \mathrm{OPT}(G, w)$$

Goemans and Williamson Algorithm

Idea: Distribute vectors in the unit sphere, considering repulsion forces

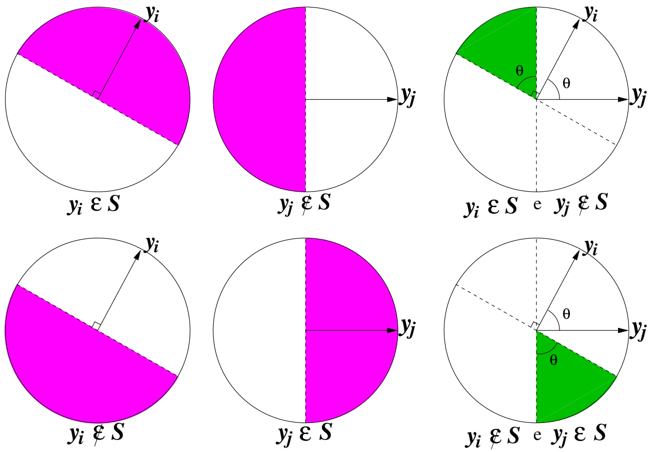
```
MAXCUT-GW $(G, w)$
1    $\hat{Y} \leftarrow$ optimum solution of $(R)$
2    $s \leftarrow$ RANDSPHERE$(V)$
3    $S \leftarrow \{\, i \in V : s\hat{Y}_{i\star} > 0 \,\}$
4    return $\delta(S)$
```



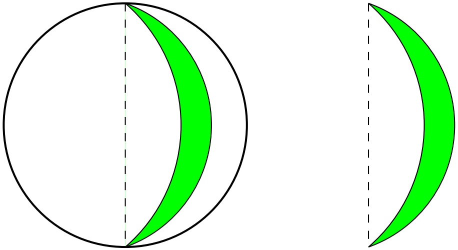
Lemma: $\Pr(\, ij \in \delta(S)\,) \;\geq\; \frac{1}{\pi} \arccos\left((\hat{Y}\,\hat{Y}^\top)_{ij}\right).$

Proof.

$\Pr(\, ij \in \delta(S)\,) = \Pr(\, s y_i > 0 \text{ and } s y_j \leq 0\,) + \Pr(\, s y_i \leq 0 \text{ and } s y_j > 0\,)$,
where $y_i = \hat{Y}_{i\star}$ and $y_j = \hat{Y}_{j\star}$.



Example in $\mathbb{R}^3$ (slice).



$$\begin{aligned}
\Pr(\, ij \in \delta(S)\,) &= \frac{\theta}{2\pi} + \frac{\theta}{2\pi} = \frac{\theta}{\pi} \\
&= \frac{\arccos(y_i y_j)}{\pi} \\
&= \frac{1}{\pi} \arccos\left((\hat{Y}\hat{Y}^\top)_{ij}\right)
\end{aligned}$$

$\square$

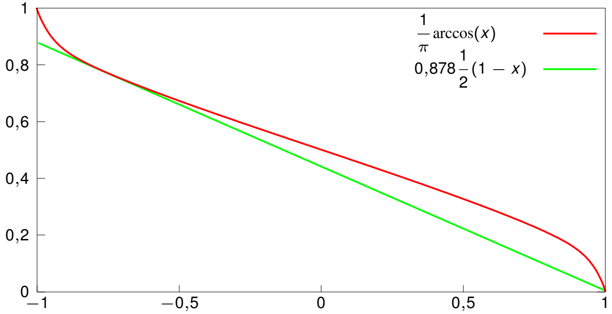
Theorem: $\mathbf{E}[w(\delta(S))] \ \geq \ 0.878\,\mathrm{OPT}(G, w)$.
Proof.

$$\begin{aligned}
\mathbf{E}[w(\delta(S))] &= \sum_{ij\in E} w_{ij}\,\Pr(\, ij \in \delta(S)\,) \\
&\geq \sum_{ij\in E} w_{ij}\,\frac{1}{\pi}\arccos\left((\hat{Y}\hat{Y}^\top)_{ij}\right) \\
&\geq 0.878\,\frac{1}{2}\sum_{ij\in E} w_{ij}(1-(\hat{Y}\hat{Y}^\top)_{ij}) \quad \text{Replace by linear functio} \\
&\geq 0.878\,\mathrm{OPT}(G, w) .
\end{aligned}$$

□

Theorem: MAXCUT-GW *is a randomized* 0.878*-approximation.*
Theorem: *Algorithm* MAXCUT-GW *can be derandomized (Mahajan and Ramesh' 95).*

$$\frac{1}{\pi}\arccos(x) \;\geq\; 0{,}878\,\frac{1}{2}(1-x)\,.$$



Replace by linear function

Vector Programs $\times$ Semidefinite Programs

Def.: *A square matrix X is* **positive semidefinite** $(X \succeq 0)$,
iff $\exists$ square matrix Y such that $X = YY^\top$.

Vector Program:

$$\begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1-(YY^\top)_{ij}) \\ & (YY^\top)_{ii} = 1 \qquad \forall i \in V \end{array}$$

$$\Downarrow$$

Semidefinite Program:

$$\begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1-X_{ij}) \\ & X_{ii} = 1 \qquad \forall i \in V \\ & X \succeq 0 \end{array}$$

Semidefinite Programming

Semidefinite Program: Find square matrix X such that

$$\begin{array}{rcll} \text{Min} & \multicolumn{3}{l}{\sum_{ij \in V \times V} w_{ij}(X_{ij})} \\ & A_{kij} X_{ij} & = & b_k \quad \forall k \in M \\ & X & \succeq & 0 \end{array}$$

- Given positive semidefinite matrix X, it is possible to obtain matrix Y in polynomial time such that $X = YY^T$ (model with real numbers).
- Solutions of semidefinite program may be irrational.
- We can find solutions arbitrary close to the optimum using the Ellipsoid and interior point methods.
- Optimum solutions or close to optimum (some solutions may be irrational) [Goemans & Williamson, Homer & Peinado, Poljak & Rendl]
- RANDSPHERE can be implemented using random number generators in [0,1] (Knuth'98)

Inapproximability

Optimization Problems

- $\mathcal{I}$: Set of Instances.
- Sol(I): Set of solutions for each $I \in \mathcal{I}$.
- $\mathrm{val}(I, S)$: Value of solution $S \in$ Sol(I).

Minimization Problem

Find $S \in$ Sol(I) such that $\mathrm{val}(I, S)$ is minimum.

Maximization Problem

Find $S \in$ Sol(I) such that $\mathrm{val}(I, S)$ is maximum.

Optimum solution

Solution with minimum (maximum) value for minimization (maximization) problem.

Approximability Classes

NPO - Extension of NP to optimization problems

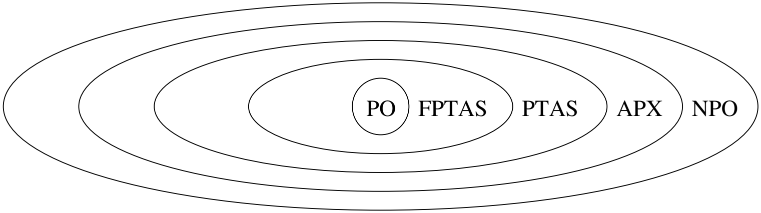
- $\exists$ polynomial function p such that $\langle S\rangle \le p(\langle I\rangle)$, $\forall I \in \mathcal{I}$, $\forall S \in \mathrm{Sol}(I)$.
- Given word X, $\exists$ polynomial time algorithm that decides if $X \in \mathcal{I}$.
- Given object Y and $I \in \mathcal{I}$, $\exists$ polynomial time algorithm that decides $Y \in \mathrm{Sol}(I)$.
- Given $I \in \mathcal{I}$ and $S \in \mathrm{Sol}(I)$, $\exists$ polynomial time algorithm that computes $\mathrm{val}(I, S)$.

PO - Problems of NPO that has polynomial time algorithm

APX - Problems in NPO that has α-approximations, for some constant α.

Fato: *PO* $\subseteq$ *FPTAS* $\subseteq$ *PTAS* $\subseteq$ *APX* $\subseteq$ *NPO*

Possible configuration for these classes:



Theorem: MOCHILA $\in$ *FPTAS.*

Theorem: ESCALONAMENTO $\in$ *PTAS.*

Theorem: *The following optimization problems for planar graphs are NP-hard and admit PTAS (Baker'94): Independent set, minimum cover, minimum dominant set, packing of triangles.*

Theorem: *The problems* BIN PACKING, MAXCUT, MAXSAT, MINCV, MINFS, TSPM *belongs to APX.*

Theorem: MINCC, MINMCUT, MINTC, TSP *belongs to NPO.*

NP-complete in strong sense

Max(I): largest integer in absolute sense that appear in $I \in \mathcal{I}$;
Max(I) $:= 0$ if no integer number occur in I.

$\mathcal{I}_p := \{I \in \mathcal{I} : \mathrm{Max}(I) \leq p(\langle I \rangle)\}$ for a polynomial time function p.

Π is NP*-complete in the strong sense* or *strongly* NP*-complete* if there exists polynomial time function p such that Π_p is NP-complete.

Theorem: *The Knapsack problem is not strongly NP-complete.*
Proof. Exercise. □

Theorem: *The following problems for planar graphs are strongly NP-complete: Independent set, vertex cover, dominating set, packing of triangles.*
Proof. Exercise. □

Theorem: *The problem* SCHEDULING *is strongly NP-complete.*

Theorem: *(Garey & Johnson'78) Let* $\Pi \in$*NPO be strongly* NP*-complete such that* $\mathrm{val}(I, S)$ *is non-negative integer* $\forall I \in \mathcal{I}$ *and* $\forall S \in \mathrm{Sol}(I)$. *If* $p(n, m)$ *is a polynomial function such that*

$$\mathrm{OPT}(I) \leq p(\langle I \rangle, \mathrm{Max}(I)), \quad \forall I \in \mathcal{I}$$

and $\Pi \in$ FPTAS, *then* P $=$ NP.

Proof. Let Π be a minimization problem (analogous to maximization).

Let A be a FPTAS for Π.

Let $\epsilon := \dfrac{1}{p(\langle I \rangle, \mathrm{Max}(I)) + 1}$. In polynomial time, we have

$$\begin{aligned} \mathrm{val}(I, A(\epsilon, I)) - \mathrm{OPT}(I) &\leq \epsilon\, \mathrm{OPT}(I) \\ &= \frac{\mathrm{OPT}(I)}{p(\langle I \rangle, \mathrm{Max}(I)) + 1} \\ &< 1\,, \end{aligned}$$

for each instance I. I.e., $\mathrm{val}(I, A(\epsilon, I)) = \mathrm{OPT}(I)$. □

This theorem is valid only for rational numbers.

Theorem: *If* PO $=$ FPTAS*, then* P $=$ NP*.*
Proof. Exercise. □

Theorem: *If* FPTAS $=$ PTAS*, then* P $=$ NP*.*
Proof. Exercise. □

Theorem: *If PTAS=APX, then P=NP.*
Proof. Exercise. □

Theorem: *If APX=NPO, then P=NP.*
Proof. Exercise. □

Theorem: *If* P $=$ NP*, then* PO $=$ NPO*.*
Proof. See Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela and Protasi'99. □

Completeness for optimization problems

An AP*-reduction* of an optimization problem Π to an optimization problem Π' ($\Pi \leq_{\text{AP}} \Pi'$) is a tuple (f, g, β) where f and g are algorithms and β is a positive rational such that:

(AP1) f receives positive rational δ and instance I of Π, and returns an instance $f(\delta, I)$ of Π';

(AP2) g receives positive rational δ, instance I of Π and element S' in $\mathrm{Sol}(f(\delta, I))$, and returns $g(\delta, I, S')$ in $\mathrm{Sol}(I)$;

(AP3) for each positive rational δ, the algorithms $f(\delta, \cdot)$ and $g(\delta, \cdot, \cdot)$ are polynomial time; and

(AP4) for each instance I of Π, any positive rational δ, and any S' in $\mathrm{Sol}(f(\delta, I))$, it is valid that if
$(1 - \delta)\,\mathrm{OPT}(f(\delta, I)) \leq \mathrm{val}(f(\delta, I), S') \leq$
$(1 + \delta)\,\mathrm{OPT}(f(\delta, I))$,
then
$(1 - \beta\delta)\,\mathrm{OPT}(I) \leq \mathrm{val}(I, g(\delta, I, S')) \leq (1 + \beta\delta)\,\mathrm{OPT}(I)$.

$$\begin{tikzcd}
\Pi & \text{Instância} I \arrow[r, bend left, "I' := f(\delta, I)"] & I' \arrow[d, "S' := A'(I')"] & \Pi' \\
(1 \pm \beta\delta)\text{-}\textit{aproximação} & \text{Solução } S & S' \arrow[l, bend left, "S := g(\delta, I, S')"] & (1 \pm \delta)\text{-}\textit{aproximação}
\end{tikzcd}$$

Theorem: *If $\Pi_1 \leq_{\text{AP}} \Pi_2$ and $\Pi_2 \leq_{\text{AP}} \Pi_3$, then $\Pi_1 \leq_{\text{AP}} \Pi_3$.*
Proof. Exercise. □

Theorem: *If Π is in NPO, Π' is in APX and $\Pi \leq_{\text{AP}} \Pi'$, then Π is in APX.*
Proof. Exercise. □

Theorem: *If* $\Pi \in$ NPO, $\Pi' \in$ PTAS *and* $\Pi \leq_{AP} \Pi'$, *then* $\Pi \in$ PTAS.

Proof. Let A' a PTAS for Π' and (f, g, β) an AP-reduction from Π to Π'. We can do an approximation scheme A for Π.

```
A(ε, I)
1    $\epsilon' \leftarrow \epsilon/\beta$
2    $I' \leftarrow f(\epsilon', I)$
3    $S' \leftarrow A'(\epsilon', I')$
4    $S \leftarrow g(\epsilon', I, S')$
5    return $S$
```

$$\begin{tikzcd}
\Pi & \text{Instance } I \arrow[r, bend left, "I' := f(\epsilon', I)"] & I' \arrow[d, "S' := A'(I')"] & \Pi' \\
(1 \pm \epsilon)\text{-approximation} & \text{Solution } S & S' \arrow[l, bend left, "S := g(\epsilon', I, S')"] & (1 \pm \epsilon')\text{-approximation}
\end{tikzcd}$$

□

Def.: *A problem* Π *in* APX *is* APX*-complete if each problem in* APX *can be* AP*-reduced to* Π*.*

Theorem: *(Papadimitriou & Yannakakis'91 and Khanna, Motwani, Sudan & Vazirani'99) The problem* MAXSAT *is* APX*-complete.*

Def.: *A problem* Π*, non-necessarily in* APX*, is* APX*-hard if the existence of a PTAS for* Π *implies* $\mathrm{P} = \mathrm{NP}$*.*

Theorem: *The problems* BIN PACKING*,* MAXCUT*,* MAXSAT*,* MINCC*,* MINCV*,* MINFS*,* MINMCUT*,* MINTC*,* TSPM *and* TSP *are* APX*-hard.*

Def.: *A problem* Π *in* NPO *is* NPO*-complete if each problem in* NPO *can be* AP*-reduced to* Π*.*

Theorem: *(Orponen and Mannila'87) The problem* TSP *is* NPO*-complete.*

Limits of approximability

Def.: *O limits of approximability (approximation threshold) of a minimization (maximization) problem is the largest (smallest) lower (upper) bound of a possible value α for which there exists a polynomial time α-approximation for the problem.*

Lemma: *If* $\mathsf{P} = \mathsf{NP}$ *then the approximation threshold for a problem in* NPO *is 1.*

Table with some results on approximation threshold for several problem, considering P $\neq$ NP.

problem	approximation threshold
KNAPSACK	$= 1$ (Ibarra & Kim'75) (problem in FPTAS)
SCHEDULING	$= 1$ (Hochbaum & Shmoys'88) (problem in PTAS)
BIN PACKING	$= 3/2$ (Garey & Johnson'79 / Simchi-Levi'94)
MAXCUT	$\leq 16/17$ (Håstad'97)
MAXSAT	$\leq 7/8$ (Håstad'97)
MINCV	$\geq 7/6$ (Håstad'97)
TSPM	$\geq 131/130$ (Engebretsen & Karpinski'00)
MINCC $(E, \mathcal{S}, c)$	$> \epsilon \log \lvert E \rvert$, for some constant $\epsilon > 0$ (Raz & Safra'97)
MINTC $(E, \mathcal{S}, c)$	$> \epsilon \log \lvert E \rvert$, for some constant $\epsilon > 0$ (equivalente ao MINCC (Ausiello, D'Atri & Protasi'80)
TSP (G, c)	$> f(\langle G, c \rangle)$, for any function f computable in polynomial time (Sahni & Gonzalez'76)
CLIQUE(V, E)	$< 1/\lvert V \rvert^{1-\epsilon}$ (Zuckerman'07) for any $\epsilon > 0$

Probabilistic Checkable Proofs

NP Class

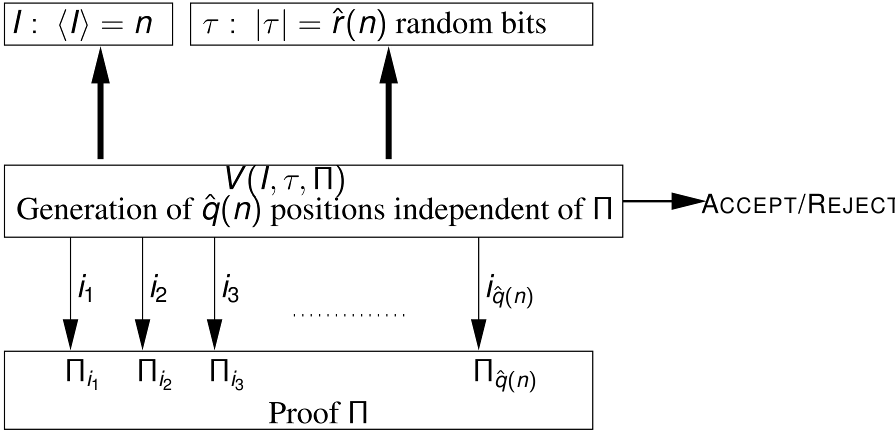
- Alphabet $\Sigma = \{0, 1\}$
- Language $L \subseteq \Sigma^*$
- $L \in \mathrm{NP} \Leftrightarrow \forall I \in L$ there exists a "short" certificate C_I and polynomial time algorithm V that verifies that $I \in L$.

PCP Proof systems (Probabilistically Checkable Proofs)

Def.: *Given instance I, $\langle I \rangle = n$, functions $r(n)$ and $q(n)$, is a sequence τ of random bits, we say that V is a $(r(n), q(n))$-restricted verifier if there exists integer function $\hat{r}(n) = O(r(n))$ and $\hat{q}(n) = O(q(n))$ such that*

- *V, access I and $\hat{r}(n)$ first bits of τ and*
- *determine $\hat{q}(n)$ positions of Π, $i_1, i_2, \ldots, i_{\hat{q}(n)}$*
- *access $\Pi_{i_1}, \Pi_{i_2}, \ldots, \Pi_{i_{\hat{q}(n)}}$ and answer* ACCEPT *or* REJECT *deterministically.*

Verifier $(r(n), q(n))$-restrict



$I : \langle I \rangle = n$
$\tau : |\tau| = \hat{r}(n)$ random bits
$V(I, \tau, \Pi)$
Generation of $\hat{q}(n)$ positions independent of Π
ACCEPT/REJECT
i_1
i_2
i_3
$i_{\hat{q}(n)}$
Π_{i_1}
Π_{i_2}
Π_{i_3}
$\Pi_{\hat{q}(n)}$
Proof Π

Def.: *A language $L \in \Sigma^*$ is in $\mathrm{PCP}(r(n), q(n))$ if and only if there exists a $(r(n), q(n))$-restricted verifier such that*

- *For any $I \in L$, $\exists\, \Pi_I \in \Sigma^*$:*
  $\Pr_\tau(V(I, \tau, \Pi_I) = \textsc{Accept}) = 1$
- *For any $I \notin L$, $\forall\, \Pi \in \Sigma^*$*
  $\Pr_\tau(V(I, \tau, \Pi) = \textsc{Accept}) < \frac{1}{4}$

We can replace $\frac{1}{4}$ by any constant value β, such that $0 < \beta < 1$.

New characterization of NP

Theorem: *(Arora, Lund, Motwani, Sudan & Szegedy'92)*
$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Inapproximability of MAX3SAT

Max3Sat Problem $(V, \mathcal{C})$ Given collection $\mathcal{C}$ of clauses over a set V of variables, each clause with exactly 3 literals (of different variables), find an attribution x of V that satisfy the largest possible of clauses of $\mathcal{C}$.

Theorem: *If* P $\neq$ NP *then* MAX3SAT $\notin$ PTAS.

Proof.

We show that

MAX3SAT $\in$ PTAS $\Rightarrow \exists$ polynomial time algorithm that decides L, for any $L \in$ NP.

Let $L \in \mathrm{NP}$ and $I \in \Sigma^*$. We can consider if $I \in L$.

As $L \in \mathrm{PCP}(\log n, 1)$ there exists verifier V, $(\log n, 1)$-restricted for L.

Given I we can show how to build in polynomial time an instance S_I for MAX3SAT such that

$$\begin{array}{lll} I \in L & \Rightarrow & S_I \text{ is satisfiable} \\ I \notin L & \Rightarrow & \text{at most } \frac{1}{4} \text{ of clauses of } S_I \text{ can be satisfied} \\ & & \text{(the fraction is independent of } I\text{)} \end{array}$$

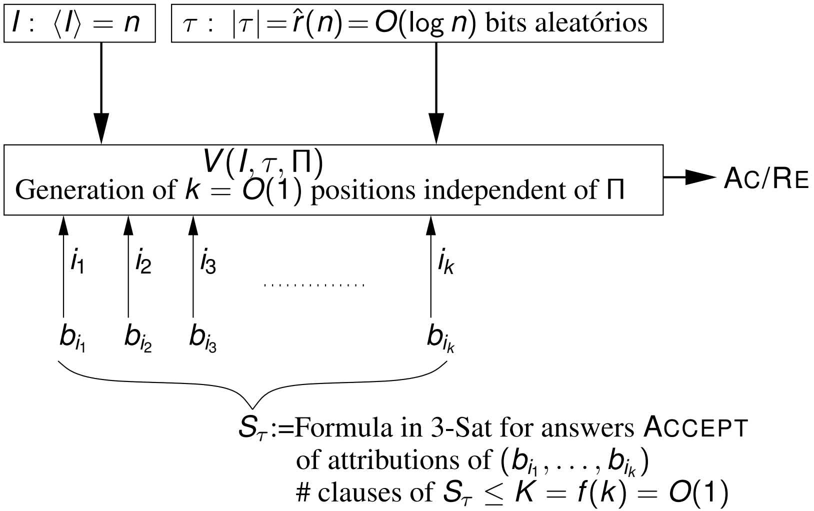
Given sequence τ of random bits, where $|\tau| = \hat{r}(n) = O(\log n)$, verifier V obtain addresses $i_1, i_2, \ldots, i_k$ in Π, where k is constant.

V returns ACCEPT or REJECT considering the k values $\Pi_{i_1}, \Pi_{i_2}, \ldots, \Pi_{i_k}$ of proof Π.

Consider all assignments of k bits for these addresses for which V answer ACCEPT.

Let S_τ a formula in 3-Sat that represents these assignments, S_τ with at most K clauses of 3 variables.

Verifier $(\log n, 1)$**-restrict**



$I : \langle I \rangle = n$
$\tau : |\tau| = \hat{r}(n) = O(\log n)$ bits aleatórios
$V(I, \tau, \Pi)$
Generation of $k = O(1)$ positions independent of Π
AC/RE
i_1
i_2
i_3
i_k
b_{i_1}
b_{i_2}
b_{i_3}
b_{i_k}
S_τ:=Formula in 3-Sat for answers ACCEPT
of attributions of $(b_{i_1}, \ldots, b_{i_k})$
# clauses of $S_\tau \leq K = f(k) = O(1)$

Now, consider all $m := 2^{\hat{r}(n)}$ possible values of bits $\tau_1, \tau_2, \ldots, \tau_m$.

Consider the formula $S := S_{\tau_1} \bigwedge S_{\tau_2} \bigwedge \ldots \bigwedge S_{\tau_m}$

$$
\begin{aligned}
I \in L \quad &\Rightarrow \quad \exists \Pi_I \text{ such that } S_I \text{ is satisfiable} \\
I \notin L \quad &\Rightarrow \quad \forall \Pi \text{ at most } \tfrac{1}{4} \text{ of the formulas } S_{\tau_i} \\
&\qquad \text{can be satisfied simultaneously}
\end{aligned}
$$

For S_{τ_i} to be unsatisfiable, it is sufficient that one of the clauses (of at most K) is not satisfied.

We can compute the maximum fraction of satisfied clauses

$$S := \underbrace{S_{\tau_1} \bigwedge S_{\tau_2} \bigwedge \dots \bigwedge S_{\tau_{m/4}}}_{\text{satisfied}} \bigwedge \underbrace{S_{\tau_{m/4+1}} \bigwedge \dots \bigwedge S_{\tau_m}}_{\text{unsatisfied}}$$

So, if $I \notin L$ at most $\dfrac{K \cdot \frac{m}{4} + (K-1) \cdot \frac{3m}{4}}{K \cdot m} = 1 - \dfrac{3}{4K}$ of the clauses can be satisfied.

So,

- or all clauses of S can be satisfied,
- or at most $\left(1 - \dfrac{3}{4K}\right)$ of the clauses of S can be satisfied.

Therefore, if we have an α-approximation for the MAX3SAT, with $\alpha > \left(1 - \frac{3}{4K}\right)$, we can decide the existence of a proof Π for I.

$\square$

Theorem: *If* P $\neq$ NP*, then* MAXSAT $\notin$ PTAS.

Inapproximability of CLIQUE

Theorem: *(Zuckerman'07) If there exists a polynomial time $n^{1-\epsilon}$-approximation for problem* CLIQUE*, for any $\epsilon > 0$, then* $\mathrm{P} = \mathrm{NP}$*.*

This result uses the PCP system. We will prove a weaker result.

Theorem: *If there exists a polynomial time α-approximation for problem* CLIQUE, *for any constant $\alpha > 0$, then* $\mathrm{P} = \mathrm{NP}$.

Proof.
We show that
CLIQUE $\in$ APX $\Rightarrow \exists$ polynomial time algorithm to decide L, where $L \in$ NP.

Let $L \in$ NP and $I \in \Sigma^*$. We can consider if $I \in L$.

As $L \in \mathrm{PCP}(\log n, 1)$ there exists $(\log n, 1)$-restricted verifier V for L.
Given I we can construct a graph G_I for problem CLIQUE such that

$$\begin{array}{lcl} I \in L & \Rightarrow & \omega(G_I) = f(n) \\ I \notin L & \Rightarrow & \omega(G_I) < \frac{1}{4} f(n) \end{array}$$

Each query of random bits τ, $|\tau| = \hat{r}(n) = O(\log n)$, V we obtain k addresses $i_1, i_2, \dots, i_k$

V verify $\Pi_{i_1}, \Pi_{i_2}, \dots, \Pi_{i_k}$ and return ACCEPT or REJECT.

Let $G_I = (V_I, E_I)$ a graph such that

- V_I are all sequences of $\hat{r}(n) + k$ bits $(\tau, b_{i_1}, b_{i_2}, \dots, b_{i_k})$ and addresses verified by V, say $i_1, i_2, \dots, i_k$, such that values $(b_{i_1}, b_{i_2}, \dots, b_{i_k})$ for these positions makes V return ACCEPT.
- Given nodes v' and v'',

$$v' = (\tau', b_{i'_1}, \dots, b_{i'_k}) \quad \text{e} \quad v'' = (\tau'', b_{i''_1}, \dots, b_{i''_k})$$

  $\{v', v''\} \in E_I$ if there is no conflict in the value of two bits in the same position.

Note that G_I can be constructed in polynomial time.

Given any proof Π,

$$\begin{aligned}\omega(G_I) &\geq |\{\tau : \ V(I,\tau,\Pi) = \textsc{Accept}\}| \\ &= 2^{\hat{r}(n)}\mathrm{Pr}_\tau(V(I,\tau,\Pi) = \textsc{Accept})\end{aligned}$$

Given a clique C in G_I, $|C| = \omega(G_I)$, there exists proof Π_C, consisting of all nodes in C

$$\begin{aligned}\omega(G_I) &\leq |\{\tau : \ V(I,\tau,\Pi_C) = \textsc{Accept}\}| \\ &= 2^{\hat{r}(n)}\mathrm{Pr}_\tau(V(I,\tau,\Pi_C) = \textsc{Accept})\end{aligned}$$

Therefore

$$\omega(G_I) = 2^{\hat{r}(n)} \max_{\Pi} \mathrm{Pr}_\tau(V(I,\tau,\Pi) = \textsc{Accept})$$

By definition of PCP,

$$\max_{\Pi} \Pr_{\tau}(V(I,\tau,\Pi) = \textsc{Accept}) \begin{cases} = 1 & \text{if } I \in L \\ < \frac{1}{4} & \text{if } I \notin L \end{cases}$$

So,

- or exists a clique of size $f(n)$,
- or the maximum clique has size at most $\frac{1}{4}f(n)$,

where $f(n) = 2^{\hat{r}(n)}$.

Therefore, if we have an 4-approximation for the CLIQUE we can decide the existence of a proof Π for I.

We can improve the verifier with probability $1/\alpha$, in the place of $1/4$, proving that there is no α-approximation for CLIQUE. $\square$

Técnica Métrica

Def.: *Dados grafo G e conjunto K de pares de vértices, um caminho de s a t é um K-caminho se $\{s, t\} \in K$.*

Def.: *Um conjunto M de arestas é um K-multicorte se não existe K-caminho no grafo $G - M$.*

Problema MINMCUT (G, K, c) Dados grafo G, conjunto K de pares de vértices e custo $c_e \in \mathbb{Q}_{\geq}$ para cada $e \in E_G$, encontrar um K-multicorte M que minimize $c(M) = \sum_{e \in M} c_e$.

Teorema: *O problema* MINMCUT (G, K, c) *é polinomial quando $|K| = 1$ ou $|K| = 2$ e* NP*-difícil quando $|K| \geq 3$.*

Seja $\mathcal{P}$ o conjunto de todos os K-caminhos. O seguinte programa linear é uma relaxação para MINMCUT.

Encontrar um vetor x indexado por E_G que

$$(P) \quad \begin{array}{lrcll} \text{minimize} & \displaystyle\sum_{e \in E} c_e x_e & & & \\ & \displaystyle\sum_{e \in E_P} x_e & \geq & 1 & \text{para cada } P \text{ em } \mathcal{P}\,, \\ & x_e & \geq & 0 & \text{para cada } e \text{ em } E_G\,. \end{array}$$

Algoritmo de Garg, Vazirani e Yannakakis:

```
MINMCUT-GVY$(G, K, c)$,   $K \neq \emptyset$.
1    seja $\hat{x}$ uma solução ótima racional de (P).
2    $k \leftarrow |K|$
3    $M \leftarrow$ CENTRAL$(G, k, K, c, \hat{x})$
4    devolva $M$
```

Apresentaremos o algoritmo CENTRAL e a prova do seguinte lema posteriormente.

Lema: *O algoritmo* CENTRAL *produz um K-multicorte M em tempo polinomial tal que* $\sum_{e \in M} c_e \leq (4 \ln 2k)\, c x$.

Teorema: *(Garg, Vazirani, Yannakakis'96) O algoritmo* MINMCUT-GVY *é uma* $(4 \ln 2k)$*-aproximação polinomial para o* MINMCUT (G, K, c)*, sendo* $k := |K| > 0$.

Prova.

$$c(M) = \sum_{e \in M} c_e \leq (4 \ln 2k)\, c \hat{x} \leq (4 \ln 2k)\, \mathrm{OPT}(G, K, c) .$$

A linha 1 de MINMCUT-GVY pode ser executada em tempo polinomial, pois temos algoritmo de separação para as desigualdades de (P). □

Algoritmo Central

O algoritmo CENTRAL separa pelo menos um par de $\{s, t\} \in K$ em cada chamada recursiva.

Para isto, o algoritmo encontra um corte (S, T) tal que

1. nenhum par em K está em S,
2. algum par em K tem extamente um vértice em S
3. $c(\delta(S))$ é razoavelmente pequeno.

Seja

$$x(s,u) := \min\{\textstyle\sum_{e \in E_P} x_e : \ P \text{ é um caminho de } s \text{ a } u\}.$$

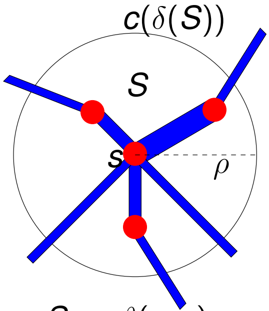
$$V(s,\rho) := \{v \in V_G : x(s,v) \leq \rho\}\ .$$

Idéia: Imagine que as arestas do grafo são tubos, sendo x_e o comprimento e c_e a área da secção transversal do tubo e.

Denote por $\vartheta(s,\rho)$ o volume da parte da tubulação que dista no máximo ρ de s:

$$\vartheta(s,\rho) := c_A x_A + \textstyle\sum_{uv \in \delta(S),\, u \in S} c_{uv}(\rho - x(s,u))\ ,$$

onde $S := V(s,\rho)$ e c_A e x_A são as restrições de c e x, ao conjunto $A := E_{G[S]}$.

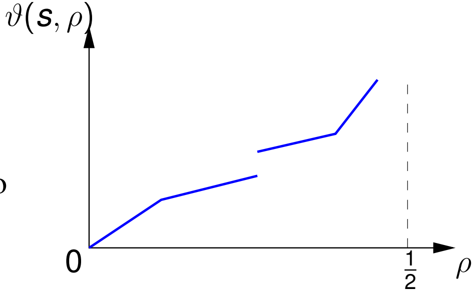


$S := \vartheta(s,\rho)$

c_e = seção transversal do tubo

x_e = comprimento do tubo

$c(\delta(S))$ = custo do corte



Descontinuidades podem ocorrer pela inclusão de toda uma aresta.

```
$\textsc{Central}(G, k, K, c, x)$,   $|K| \le k$
 1 se $K = \emptyset$
 2     então devolva $\emptyset$
 3     senão se $cx = 0$
 4         então devolva $\{e \in E_G : x_e > 0\}$
 5         senão sejam $\{s, t\} \in K$
 6             seja $v_1, \ldots, v_n$ tal que $x(s, v_1) \le \cdots \le x(s, v_n)$
 7             para $i$ de 1 a $n$ faça $p_i \leftarrow x(s, v_i)$
 8             $j \leftarrow 1 + \max\{i : p_i = 0\}$
 9             enquanto $\vartheta(s, p_j) > ((2k)^{2p_j} - 1)\frac{1}{k}cx$  faça $j \leftarrow j+1$
10             $S \leftarrow V(s, p_{j-1})$
11             $T \leftarrow V_G \setminus S$
12             $B \leftarrow E_{G[T]}$
13             $G_B \leftarrow (V_G, B)$
14             $K_B \leftarrow K \setminus \{\{s', t'\} :$  $S$ separa $s'$ de $t'\}$
15             $M_B \leftarrow \textsc{Central}(G_B, k, K_B, c_B, x_B)$
16             devolva $\delta(S) \cup M_B$
```

Lema: *Ao fim da linha 10, temos $p_{j-1} < \frac{1}{2}$, e portanto $x(s,u) < \frac{1}{2}$ para todo $u \in S$.*
Prova.
Seja h o menor natural tal que $p_h \geq \frac{1}{2}$.
Temos que $2 \leq h \leq n$ já que $p_n \geq x(s,t) \geq 1$ e $p_1 = 0$.

Como $\vartheta(s, p_h) \leq cx$ e $k \geq 1$ temos

$$\begin{aligned} \vartheta(s, p_h) &\leq cx \\ &\leq \left((2k)^{2p_h} - 1\right)\frac{1}{k}cx \end{aligned}$$

assim, h não satisfaz condição da linha 9 e portanto $j \leq h$. □

Corolário: *Em uma chamada, para um par $\{s,t\}$, temos $s \in S$ e $t \notin S$. Além disso, se $\{s_i, t_i\} \in K - \{s,t\}$, então $s_i \notin S$ ou $t_i \notin S$.*
Prova. Exercício. □

Lema: *Ao fim da linha 10 do algoritmo* CENTRAL, *temos que*

$$c(\delta(S)) \leq (2 \ln 2k)\Big(c_A x_A + c_{\delta(S)} x_{\delta(S)} + \tfrac{1}{k} c x\Big) \,,$$

onde $A := E_{G[S]}$.

Prova. Note que após a linha 9, temos

$$\begin{aligned} p_{j-1} &< p_j \,, & (1)\\ \vartheta(s, p_j) &\leq ((2k)^{2p_j} - 1)\tfrac{1}{k} c x \,. & (2)\\ \vartheta(s, p_{j-1}) &\geq ((2k)^{2p_{j-1}} - 1)\tfrac{1}{k} c x \quad \text{e} & (3) \end{aligned}$$

De (3) e (2) temos que

$$\frac{\vartheta(s, p_j) + \frac{1}{k} c x}{\vartheta(s, p_{j-1}) + \frac{1}{k} c x} \leq (2k)^{2(p_j - p_{j-1})} \,. \qquad (4)$$

Tomando-se o logaritmo natural do lado esquerdo de

$$\frac{\vartheta(s, p_j) + \frac{1}{k} c x}{\vartheta(s, p_{j-1}) + \frac{1}{k} c x} \leq (2k)^{2(p_j - p_{j-1})} . \qquad (5)$$

obtemos

$$\begin{aligned} \ln\left(\vartheta(s, p_j) + \tfrac{1}{k} c x\right) - \ln\left(\vartheta(s, p_{j-1}) + \tfrac{1}{k} c x\right) &= \\ = \int_{p_{j-1}}^{p_j} \frac{d}{d\rho} \ln\left(\vartheta(s, \rho) + \tfrac{1}{k} c x\right) d\rho & \\ = \int_{p_{j-1}}^{p_j} \frac{c(\delta(S))}{\vartheta(s, \rho) + \frac{1}{k} c x} \, d\rho \, , & \end{aligned}$$

(note que $\vartheta(s, \rho)$ é uma função linear com coeficiente $c(\delta(S))$).

Tomando-se o logaritmo natural do lado direito de

$$\frac{\vartheta(s, p_j) + \frac{1}{k} c x}{\vartheta(s, p_{j-1}) + \frac{1}{k} c x} \ \leq \ (2k)^{2(p_j - p_{j-1})} \, . \tag{6}$$

obtemos

$$2(p_j - p_{j-1}) \ln 2k \ = \ \int_{p_{j-1}}^{p_j} (2 \ln 2k) \, d\rho \, .$$

Como o logaritmo é uma função crescente, concluímos de (6) que

$$\int_{p_{j-1}}^{p_j} \frac{c(\delta(S))}{\vartheta(s, \rho) + \frac{1}{k} c x} \, d\rho \ \leq \ \int_{p_{j-1}}^{p_j} (2 \ln 2k) \, d\rho \, .$$

Então, para algum ρ no intervalo (p_{j-1}, p_j) temos que

$$\frac{c(\delta(S))}{\vartheta(s, \rho) + \frac{1}{k} c x} \leq (2 \ln 2k).$$

Assim,

$$\begin{aligned}
c(\delta(S)) &\leq (2\ln 2k)(\vartheta(s,\rho) + \frac{1}{k}cx) \\
&\leq (2\ln 2k)(c_A x_A + \sum_{uv\in\delta(S),\ u\in S} c_{uv}(\rho - x(s,u)) + \frac{1}{k}cx) \\
&\leq (2\ln 2k)(c_A x_A + \sum_{uv\in\delta(S)} c_{uv}x_{uv} + \frac{1}{k}cx) \\
&= (2\ln 2k)(c_A x_A + c_{\delta(S)}x_{\delta(S)} + \frac{1}{k}cx)
\end{aligned}$$

$\square$

Teorema: *O algoritmo* CENTRAL (G, k, K, c, x) *produz um K-multicorte M em tempo polinomial tal que*

$$c(M) \leq (2\ln 2k)(1 + \tfrac{1}{k}|K|)cx. \tag{7}$$

Prova. Por indução em $|K|$.

Se $K = \emptyset$ ou $cx = 0$, claramente (7) vale.
Suponha que $K \neq \emptyset$ e $cx > 0$.
Neste caso, o algoritmo devolve $M := \delta(S) \cup M_B$. Assim,

$$\begin{aligned}
c(\delta(S) \cup M_B) &= c(\delta(S)) + c_B(M_B) \\
&\leq (2\ln 2k)\Big(c_A x_A + c_{\delta(S)} x_{\delta(S)} + \frac{1}{k}cx + (1 + \frac{1}{k}|K_B|)c_B \\
&= (2\ln 2k)\Big(cx + \frac{1}{k}cx + \frac{1}{k}|K_B|c_B x_B\Big) \\
&\leq (2\ln 2k)\Big(cx + \frac{1}{k}cx + \frac{1}{k}(|K| - 1)cx\Big) \\
&\leq (2\ln 2k)(1 + \frac{1}{k}|K|)cx\,.
\end{aligned}$$

□

Equilíbrio de Nash e Busca Local

- Internet: Rede gigantesca com grande quantidade de usuários e complexa estrutura sócio-econômica
- Usuários podem ser competitivos, cooperativos,...
- Situações envolvendo Teoria dos Jogos e Computação

Ref.: Cap. 12 - Local Search do livro *Algorithm Design* de Kleinberg e Tardos

Um jogo Multicast

- Jogadores podem construir links entre nós
- Há um nó origem
- Cada jogador representa um nó destino
- Cada jogador quer conectar o nó origem até seu nó destino
- Há cooperação na construção da rede. Isto é, o custo de um link é dividido igualmente entre os usuários que o utilizam

Definição

Dados

- Grafo direcionado $G = (V, E)$
- Custo positivo c_e para cada aresta e.
- Vértice fonte s
- k vértices destinos $t_1, \ldots, t_k$

Cada usuário i procura encontrar

- caminho orientado P_i do vértice s até t_i pagando menos

Custo para

- usuário i é $c(P_i) = \sum_{e \in P_i} \frac{c_e}{k_e}$, onde k_e número de caminhos usando
- sistema é $c(P_1, \ldots, P_k) = \sum_i c(P_i)$ (custo social)

Jogo

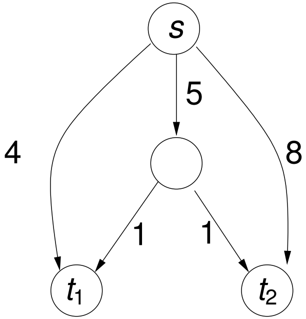
Regras do Jogo:

- Cada usuário fica estável ou muda sua rota (pagando menos) baseado apenas na configuração atual
- Em um estado do jogo com caminhos $(P_1, \ldots, P_k)$, denotamos por $E^+ \subseteq E$ as arestas usadas em pelo menos um caminho.
- O custo social é o custo dos caminhos escolhidos pelos jogadores:

$$c(P_1, \ldots, P_k) = \sum_{i=1}^{k} c(P_i) = \sum_{e \in E^+} c_e$$

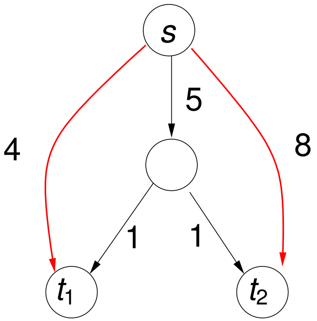
Exemplo 1

- Temos dois jogadores: 1 e 2
- Cada um tem duas alternativas: uma rota externa e uma interna.



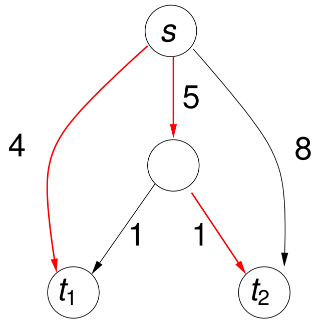
Exemplo 1

- Considere que inicialmente os jogadores usam as rotas externas.
- O jogador 1 paga 4 e o jogador 2 paga 8
- O custo social é igual a 12.



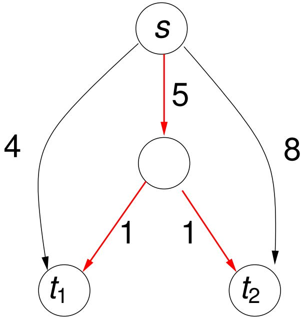
Exemplo 1

- O jogador 2 muda para a rota interna e seu custo cai para 6
- O custo social cai para 10



Exemplo 1

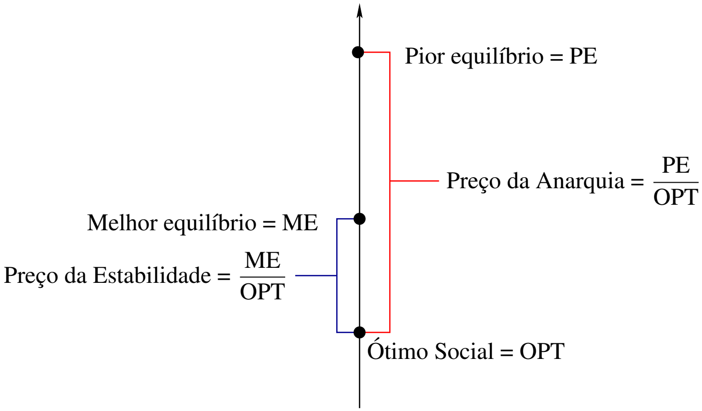
- O jogador 1 tem incentivo a mudar
- Cada jogador paga $2{,}5 + 1$, e estamos em um equilíbrio
- O custo social cai para 7 (solução final também é ótima)



Definições e Notação

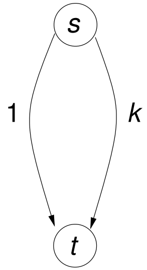
- A Estratégia do jogador i é o conjunto de rotas de s para t_i
- O estado do jogo em um momento é dado pelos k caminhos $(P_1, \ldots, P_k)$ no momento
- O ótimo social é o menor valor possível de uma solução (dos k caminhos), possivelmente não está em equilíbrio.
- Um usuário i está insatisfeito no estado atual, se ele pode mudar sua rota por outra de custo melhor
- Estado em Equilíbrio de Nash quando não há usuários insatisfeitos
- O Preço da Estabilidade razão entre a melhor solução em equilíbrio com o ótimo social
- O Preço da Anarquia razão entre a pior solução em equilíbrio com o ótimo social
- Melhor resposta: movimento para estratégia de maior ganho positivo

Preços da anarquia e da estabilidade para minimização


Pior equilíbrio = PE
Preço da Anarquia = $\frac{PE}{OPT}$
Melhor equilíbrio = ME
Preço da Estabilidade = $\frac{ME}{OPT}$
Ótimo Social = OPT

Exemplo 2

- Na rede abaixo há k jogadores todos com mesmo destino t
- Considere todos usando a aresta da direita
- Estamos em um equilíbrio com custo k (cada jogador paga 1).
- O ótimo social tem custo 1 (cada jogador paga $\frac{1}{k}$).



k Jogadores

Teorema: *O preço da anarquia deste jogo Multicast é k.*

Método da Função Potencial e o Preço da Estabilidade

Def.: *Uma* função potencial exata Φ *é uma função que*

- *mapeia cada vetor de estratégia* $\mathcal{P}$ *para um valor real tal que*
- *se* $\mathcal{P} = (P_1, \ldots, P_i, \ldots, P_k)$ *e*

  $P_i' \neq P_i$ *é uma estratégia alternativa para o jogador* i*, então*

$$\Phi(\mathcal{P}) - \Phi(\mathcal{P}') = c_i(P_i) - c_i(P_i'),$$

  onde $\mathcal{P}' = (P_1, \ldots, P_i', \ldots, P_k)$

Fato: *Seja* Φ *uma função potencial exata para o jogo do Multicast com dinâmica de melhor resposta. Se jogador* i *muda sua estratégia de* P_i *para* P_i'*, e o vetor de estratégia muda de* $\mathcal{P}$ *para* $\mathcal{P}'$*, então*

$$\Phi(\mathcal{P}) > \Phi(\mathcal{P}').$$

Isto é, Φ *é estritamente decrescente após jogadas.*

Função Potencial para Multicast

Dado vetor de estratégias $\mathcal{P} = (P_1, \ldots, P_k)$, denote por

$$\Phi(\mathcal{P}) = \sum_e c_e \cdot H(k_e),$$

onde

$$H(t) = 1 + \frac{1}{2} + \cdots + \frac{1}{t} \quad \text{e} \quad H(0) = 0$$

k_e é o número de caminhos de $\mathcal{P}$ que usam e

Lema: Φ *é uma função potencial exata.*
Prova. Exercício □

Fato: $\Phi(\mathcal{P})$ *é limitado inferiormente.*
Prova. Exercício □

Lema: *O jogo Multicast com a dinâmica de melhor resposta converge para um equilíbrio de Nash.*
Prova. Exercício □

Preço da Estabilidade

Lema: *Se $\mathcal{P} = (P_1, \ldots, P_k)$ é um vetor de estratégia, então*

$$c(\mathcal{P}) \leq \Phi(\mathcal{P}) \leq H(k)c(\mathcal{P}).$$

Teorema: *O preço da estabilidade do jogo Multicast é no máximo $H(k)$.*

Prova. Seja:

OPT um vetor de estratégia ótimo (ótimo social)

$\mathcal{O}$ um vetor de estratégia em equilíbrio obtido a partir de OPT

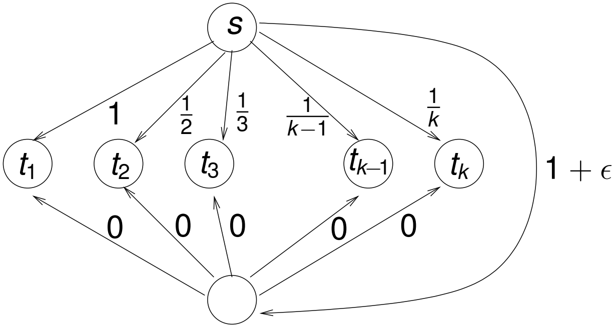
$\mathcal{P}$ um vetor de estratégia em Equilíbrio de Nash de menor custo

$$\begin{aligned} c(\mathcal{P}) &\leq c(\mathcal{O}) \\ &\leq \Phi(\mathcal{O}) \\ &\leq \Phi(\mathrm{OPT}) \\ &\leq H(k) \cdot c(\mathrm{OPT}) \end{aligned}$$

□

Lema: *O preço da estabilidade $H(k)$ do problema de Multicast é justo (melhor possível).*

Prova.



$\square$

Observações

- Tempo de convergência do problema Multicast pode ser exponencial
- Encontrar ótimo social é um problema NP-difícil.

Exercício: Mostre que encontrar o ótimo social do problema Multicast é um problema NP-difícil. Sugestão: por Cobertura por Conjuntos.

Complexidade de se encontrar Equilíbrio de Nash em Jogos Potenciais

O quão difícil é encontrar um algoritmo polinomial para encontrar um equilíbrio de Nash ?

Classe PLS - *Polynomial Local Search Problems*

Em um problema de otimização (minimização) temos:

- conjunto de instâncias I
- para entrada $x \in I$, temos um conjunto de soluções viáveis $F(x)$
- para toda solução $s \in F(x)$ temos um custo $c_x(s)$
- um oráculo que diz se s pertence ou não à $F(x)$ e em caso positivo, computa $c_x(s)$

O problema consiste em dado $x \in I$, encontrar $s \in F(x)$ tal que $c_x(s)$ é mínimo.

A versão de maximização é análoga.

Um *problema de otimização local* é um problema de otimização onde

- há uma vizinhança $N_x(s) \subset F(x)$ para cada $x \in I$ e $s \in F(x)$
- e uma solução s de $F(x)$ é um mínimo local se $c_x(s) \leq c_x(s')$ para todo $s' \in N_x(s)$.

O objetivo é encontrar uma solução que é mínimo local.

Um problema de otimização local pertence à *PLS* se temos um oráculo que, para qualquer instância $x \in I$ e solução $s \in F(x)$, decide se s é ótimo local, e se não for, devolve $s' \in N_x(s)$ com $c_x(s') < c_x(s)$.

Def.: *(Johnson, Papadimitriou, Yannakakis'88) Um problema P é PLS-completo se está em PLS e se para todo problema Q de PLS, há uma redução polinomial de Q para P, tal que qualquer ótimo local de P corresponde a um ótimo local de Q.*

Há vários problemas em PLS-completo (Circuit-SAT com pesos, busca de ótimos locais relativos ao TSP, MAXCUT, SAT, etc)

Problema da Satisfatibilidade com Pesos (LSAT - Local SAT):

- Seja ϕ uma fórmula em Forma Normal Conjuntiva (FNC) $C_1 \wedge \ldots \wedge C_m$
- cada cláusula C_j com peso w_j.
- Uma atribuição lógica qualquer das variáveis é uma solução de ϕ.
- O valor de uma solução s é o peso total das cláusulas satisfeitas por s.
- A vizinhança de s são as atribuições obtidas trocando o valor de apenas uma variável de s.
- O objetivo é encontrar uma solução que é mínimo local.

Teorema: *(Krentel’89) O problema LSAT é um problema PLS-completo.*

Teorema: *(Fabrikant, Papadimitriou, Talwar’04) O problema de se encontrar um equilíbrio puro de Nash em jogos potenciais, onde a melhor resposta é computada em tempo polinomial, é PLS-completo.*