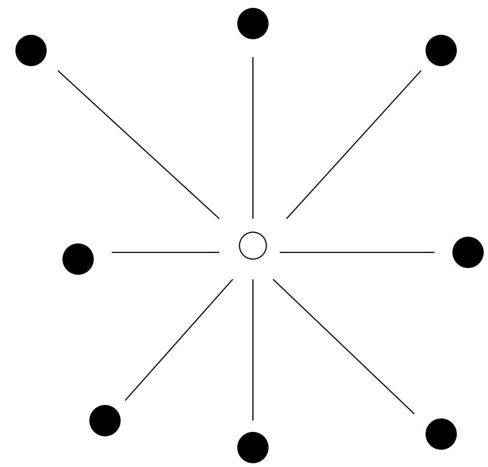
Vertex Cover Problems



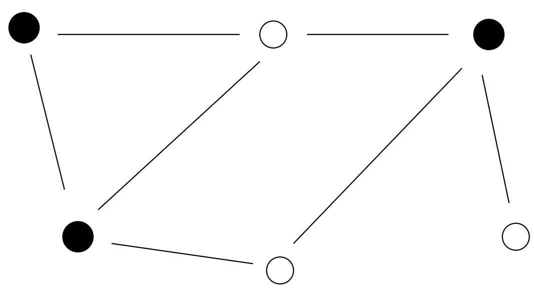
Consider a graph $G = (V,E)$

$S \subseteq V$ is a vertex cover if

$$\forall \{u,v\} \in E : u \in S \vee v \in S$$

minimum vertex cover (MIN-VCP):

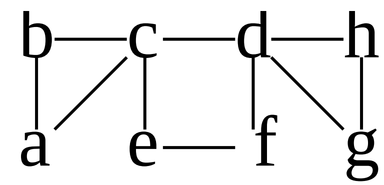
find a vertex cover S that minimizes $|S|$.



Motivation

- This problem has many applications
- Example: placing ATMs in a city
- Each additional ATM costs money
- Want to have an ATM in every street (block, district)
- Where should they be placed so that we need as little ATMs as possible?

Greedy Algorithm



Function greedyVC(V, E)

  $C := \emptyset$

  while $E \neq \emptyset$ **do**

    select any $\{u, v\} \in E$

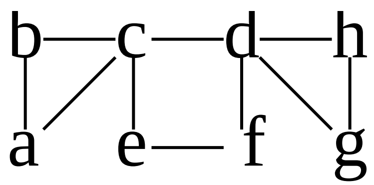
    $C := C \cup \{u, v\}$

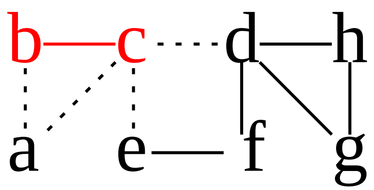
    remove all edges incident to u or v from E

  return C

Exercise: explain how to implement the algorithm to run in time $\mathcal{O}(|V| + |E|)$

Greedy Algorithm





Function greedyVC(V, E)

$C := \emptyset$

while $E \neq \emptyset$ **do**

select any $\{u, v\} \in E$

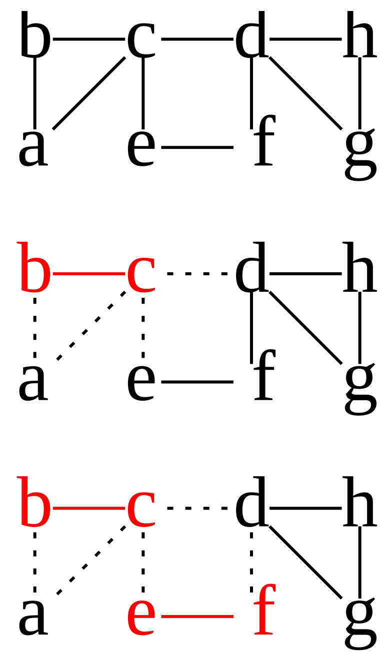
$C := C \cup \{u, v\}$

remove all edges incident to u or v from E

return C

Exercise: explain how to implement the algorithm to run in time $\mathcal{O}(|V| + |E|)$

Greedy Algorithm



Function greedyVC(V,E)

$C := \emptyset$

while $E \neq \emptyset$ **do**

select any $\{u,v\} \in E$

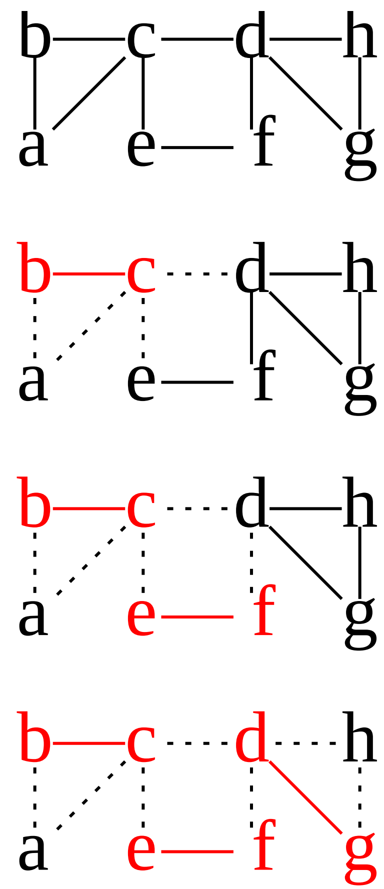
$C := C \cup \{u,v\}$

remove all edges incident to u or v from E

return C

Exercise: explain how to implement the algorithm to run in time $\mathcal{O}(|V| + |E|)$

Greedy Algorithm



Function greedyVC(V,E)

$C := \emptyset$

while $E \neq \emptyset$ **do**

select any $\{u,v\} \in E$

$C := C \cup \{u,v\}$

remove all edges incident to u or v from E

return C

Exercise: explain how to implement the algorithm to run in time $\mathcal{O}(|V| + |E|)$

Theorem 1. *Algorithm greedyVC computes a two-approximation of MIN-VCP.*

Proof. Correctness: trivial since only covered edges are removed.

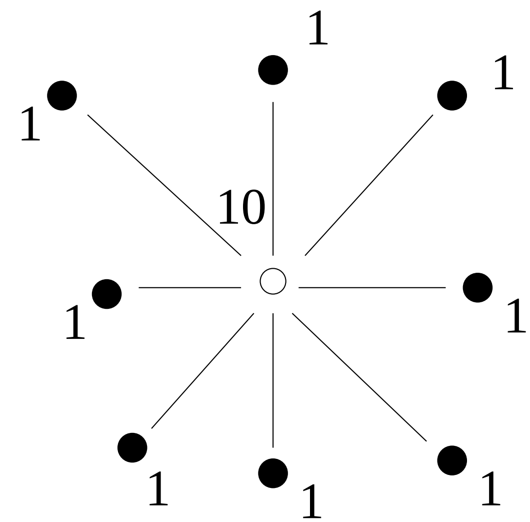
Quality: Let A denote the set of edges selected by greedyVC.
We have $|C| = 2|A|$.
A is a matching, i.e., no node covers two edges in A.
Hence, any vertex cover contains at least one node from each edge in A, i.e., opt$\geq |A|$. □

Weighted Vertex Cover

Consider a graph $G = (V, E)$

$S \subseteq V$ is a vertex cover if

$$\forall \{u, v\} \in E : u \in S \vee v \in S$$

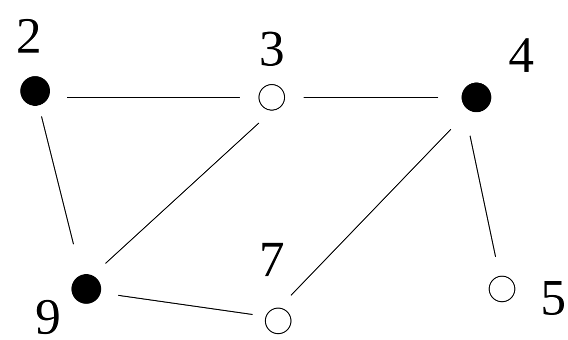


minimum WEIGHT vertex cover

(WEIGHT-VCP):

find a vertex cover S that minimizes

$$\sum_{v \in S} c(s)$$



0-1 ILP Formulation

Assume $V = \{1, \ldots, n\}$

Variables: $x_v = 1$ iff $v \in V$

minimize $\mathbf{c} \cdot \mathbf{x}$

subject to

$\forall \{u, v\} \in E : x_u + x_v \geq 1$

$\forall v \in V : x_v \in \{0, 1\}$

0-1 ILP Formulation

Assume $V = \{1, \ldots, n\}$

Variables: $x_v = 1$ iff $v \in V$

minimize $\mathbf{c} \cdot \mathbf{x}$

subject to

$\forall \{u, v\} \in E : x_u + x_v \geq 1$

$\forall v \in V : x_v \in \{0, 1\}$

Linear Relaxation

Assume $V = \{1, \ldots, n\}$

Variables: $x_v = 1$ iff $v \in V$

minimize $\mathbf{c} \cdot \mathbf{x}$

subject to

$\forall \{u, v\} \in E : x_u + x_v \geq 1$

$\forall v \in V : x_v \geq 0$

0-1 ILP Formulation

Assume $V = \{1, \ldots, n\}$

Variables: $x_v = 1$ iff $v \in V$

minimize $\mathbf{c} \cdot \mathbf{x}$

subject to

$\forall \{u, v\} \in E : x_u + x_v \geq 1$

$\forall v \in V : x_v \in \{0, 1\}$

Linear Relaxation

Assume $V = \{1, \ldots, n\}$

Variables: $x_v = 1$ iff $v \in V$

minimize $\mathbf{c} \cdot \mathbf{x}$

subject to

$\forall \{u, v\} \in E : x_u + x_v \geq 1$

$\forall v \in V : x_v \geq 0$

LP Rounding Algorithm for WEIGHT-VCP

Function lpWeightedVC$(V, E, \mathbf{c})$

  $\mathbf{x}$:= lpSolve(linearRelaxation$(V, E, \mathbf{c})$)

  return $\{v \in V : x_v \geq 1/2\}$

Theorem 2. *Algorithm lpWeightedVC computes a two-approximation of WEIGHT-VCP.*

Correctness:

Consider any edge $\{u,v\} \in E$.

We have $x_u + x_v \geq 1$,

hence, $\max\{x_u, x_v\} \geq 1/2$,

i.e., rounding will put at least one of $\{u,v\}$ into the output.

Theorem 2. *Algorithm lpWeightedVC computes a two-approximation of WEIGHT-VCP.*

Quality: Let

$\mathbf{x}$:= the solution computed by lpWeightedVC

$\mathbf{x}^*$:= the optimal solution, and

$\bar{\mathbf{x}}$:= the optimal solution of the linear relaxation

$$\mathbf{c} \cdot \mathbf{x} = \sum_{\bar{x}_i \geq 1/2} c_i$$

Theorem 2. *Algorithm lpWeightedVC computes a two-approximation of WEIGHT-VCP.*

Quality: Let

$\mathbf{x}$:= the solution computed by lpWeightedVC

$\mathbf{x}^*$:= the optimal solution, and

$\bar{\mathbf{x}}$:= the optimal solution of the linear relaxation

$$\mathbf{c} \cdot \mathbf{x} = \sum_{\bar{x_i} \geq 1/2} c_i \leq \sum_{\bar{x_i} \geq 1/2} 2\bar{x_i} c_i$$

Theorem 2. *Algorithm lpWeightedVC computes a two-approximation of WEIGHT-VCP.*

Quality: Let

$\mathbf{x}$:= the solution computed by lpWeightedVC

$\mathbf{x}^*$:= the optimal solution, and

$\bar{\mathbf{x}}$:= the optimal solution of the linear relaxation

$$\mathbf{c} \cdot \mathbf{x} = \sum_{\bar{x}_i \geq 1/2} c_i \leq \sum_{\bar{x}_i \geq 1/2} 2\bar{x}_i c_i \leq 2 \sum_{i=1}^{n} \bar{x}_i c_i$$

Theorem 2. *Algorithm lpWeightedVC computes a two-approximation of WEIGHT-VCP.*

Quality: Let

$\mathbf{x}$:= the solution computed by lpWeightedVC

$\mathbf{x}^*$:= the optimal solution, and

$\bar{\mathbf{x}}$:= the optimal solution of the linear relaxation

$$\mathbf{c} \cdot \mathbf{x} = \sum_{\bar{x}_i \geq 1/2} c_i \leq \sum_{\bar{x}_i \geq 1/2} 2\bar{x}_i c_i \leq 2 \sum_{i=1}^{n} \bar{x}_i c_i = 2\mathbf{c} \cdot \bar{\mathbf{x}}$$

Theorem 2. *Algorithm lpWeightedVC computes a two-approximation of WEIGHT-VCP.*

Quality: Let

$\mathbf{x}$:= the solution computed by lpWeightedVC

$\mathbf{x}^*$:= the optimal solution, and

$\bar{\mathbf{x}}$:= the optimal solution of the linear relaxation

$$\mathbf{c} \cdot \mathbf{x} = \sum_{\bar{x}_i \geq 1/2} c_i \leq \sum_{\bar{x}_i \geq 1/2} 2\bar{x}_i c_i \leq 2 \sum_{i=1}^{n} \bar{x}_i c_i = 2\mathbf{c} \cdot \bar{\mathbf{x}} \leq 2\mathbf{c} \cdot \mathbf{x}^*$$

■

Iterated Rounding

[Vazirani Section 23.2]

Function iteratedLpWeightedVC$(V, E, \mathbf{c})$

$M := \emptyset$

while $|E| > 0$ **do**

$\mathbf{x} :=$ lpSolve(linearRelaxation$(V, E, \mathbf{c})$)

let v denote the node which maximizes x_v

$M := M \cup \{v\}$

$V := V \setminus \{v\}$

$E := E \setminus \{\{u, v\} \in E\}$

return M

Iterated Rounding: Discussion

- Might give better solutions for many inputs
- No better approximation guarantees for VC
- Larger (still polynomial) execution time
- But: Resolving an LP is often quite fast
- Important technique for other problems

A Randomized Algorithm

[Ausiello et al. Section 5.1]

Function randWeightedVC$(V, E, \mathbf{c})$

$C := \emptyset$

while $E \neq \emptyset$ **do**

select any $\{v, t\} \in E$

flip a coin with sides $\{v, t\}$ and

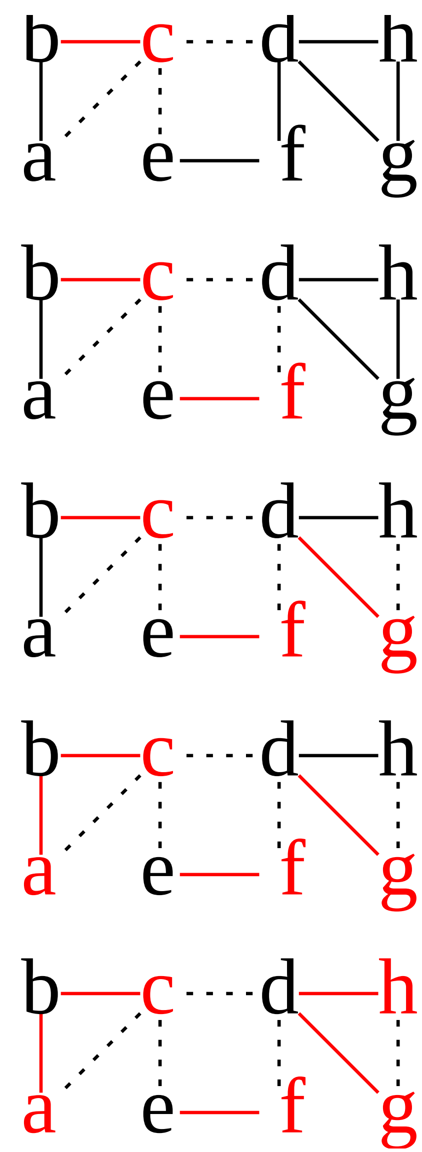
$$\mathbb{P}[v] = \frac{c_t}{c_v + c_t}$$

$x :=$ upper side of coin

$C := C \cup \{x\}$

remove all edges incident to x from E

return C



Theorem 3. *Algorithm randWeightedVC computes a vertex cover* $\mathbf{x}$ *with* $\mathbb{E}[\mathbf{c} \cdot \mathbf{x}] \leq 2\mathbf{c} \cdot \mathbf{x}^*$.

Correctness: as for greedyVC.

Theorem: Algorithm randWeightedVC computes a vertex cover $\mathbf{x}$ with $\mathbb{E}[\mathbf{c}\cdot\mathbf{x}] \leq 2\mathbf{c}\cdot\mathbf{x}^*$.

Quality: Define the random variables

$$X_v := \begin{cases} c_v & \text{if } v \in \mathbf{x} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$X_{\{v,t\},v} := \begin{cases} c_v & \text{if } \{v,t\} \text{ is selected and } v \in \mathbf{x} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that $X_v = \sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}$

Lemma 4. $\mathbb{E}[X_{\{v,t\},v}] = \mathbb{E}[X_{\{v,t\},t}]$

Proof.

$$\mathbb{E}[X_{\{v,t\},v}] = c_v \mathbb{P}\left[\{v,t\} \text{ is selected }\right] \mathbb{P}[v \in \mathbf{x}]$$

□

Lemma 4. $\mathbb{E}[X_{\{v,t\},v}] = \mathbb{E}[X_{\{v,t\},t}]$

Proof.

$$\mathbb{E}[X_{\{v,t\},v}] = c_v \mathbb{P}\left[\{v,t\} \text{ is selected }\right] \mathbb{P}[v \in \mathbf{x}]$$

$$\mathbb{E}[X_{\{v,t\},v}] = c_v \mathbb{P}\left[\{v,t\} \text{ is selected }\right] \frac{c_t}{c_v + c_t}$$

□

Lemma 4. $\mathbb{E}[X_{\{v,t\},v}] = \mathbb{E}[X_{\{v,t\},t}]$

Proof.

$$\mathbb{E}[X_{\{v,t\},v}] = c_v \mathbb{P}\left[\{v,t\} \text{ is selected }\right] \mathbb{P}[v \in \mathbf{x}]$$

$$\begin{aligned}\mathbb{E}[X_{\{v,t\},v}] &= c_v \mathbb{P}\left[\{v,t\} \text{ is selected }\right] \frac{c_t}{c_v + c_t} \\ &= c_t \mathbb{P}\left[\{v,t\} \text{ is selected }\right] \frac{c_v}{c_v + c_t} \\ &= \mathbb{E}[X_{\{v,t\},t}]\end{aligned}$$

□

Lemma 5. $\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] \leq \sum_{t \in \mathbf{x}^*} \mathbb{E}[X_t]$

Proof.

$$\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] = \sum_{v \notin \mathbf{x}^*} \mathbb{E}\left[\sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right] \quad \left(X_v = \sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right)$$

Lemma 5. $\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] \leq \sum_{t \in \mathbf{x}^*} \mathbb{E}[X_t]$

Proof.

$$\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] = \sum_{v \notin \mathbf{x}^*} \mathbb{E}\left[\sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right] \quad \left(X_v = \sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right)$$

$$= \sum_{v \notin \mathbf{x}^*} \sum_{\{t:\{v,t\}\in E\}} \mathbb{E}[X_{\{v,t\},v}] \quad \text{Linearity of } \mathbb{E}[\cdot]$$

Lemma 5. $\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] \leq \sum_{t \in \mathbf{x}^*} \mathbb{E}[X_t]$

Proof.

$$\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] = \sum_{v \notin \mathbf{x}^*} \mathbb{E}\left[\sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right] \quad \left(X_v = \sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right)$$

$$= \sum_{v \notin \mathbf{x}^*} \sum_{\{t:\{v,t\}\in E\}} \mathbb{E}[X_{\{v,t\},v}] \quad \text{Linearity of } \mathbb{E}[\cdot]$$

$$= \sum_{v \notin \mathbf{x}^*} \sum_{\{t:\{v,t\}\in E\}} \mathbb{E}[X_{\{v,t\},t}] (*). \quad \text{Lemma 4}$$

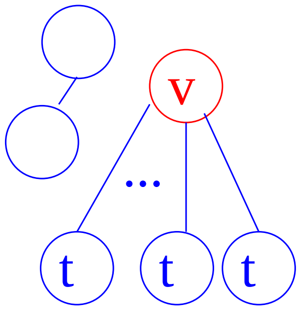
Lemma 5. $\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] \leq \sum_{t \in \mathbf{x}^*} \mathbb{E}[X_t]$

Proof.

$$\sum_{v \notin \mathbf{x}^*} \mathbb{E}[X_v] = \sum_{v \notin \mathbf{x}^*} \mathbb{E}\left[\sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right] \quad \left(X_v = \sum_{\{t:\{v,t\}\in E\}} X_{\{v,t\},v}\right)$$

$$= \sum_{v \notin \mathbf{x}^*} \sum_{\{t:\{v,t\}\in E\}} \mathbb{E}[X_{\{v,t\},v}] \quad \text{Linearity of } \mathbb{E}[\cdot]$$

$$= \sum_{v \notin \mathbf{x}^*} \sum_{\{t:\{v,t\}\in E\}} \mathbb{E}[X_{\{v,t\},t}](*). \quad \text{Lemma 4}$$



But also $\sum_{t \in \mathbf{x}^*} \mathbb{E}[X_t] = \sum_{t \in \mathbf{x}^*} \sum_{\{v:\{v,t\}\in E\}} \mathbb{E}[X_{\{v,t\},t}](**)$.

Every term in (*) shows up in (**). $\square$

Theorem: Algorithm randWeightedVC computes a vertex cover $\mathbf{x}$ with $\mathbb{E}[\mathbf{c}\cdot\mathbf{x}] \leq 2\mathbf{c}\cdot\mathbf{x}^*$.

Quality: (Finishing Up)

$$\sum_{v\in V}\mathbb{E}[X_v] = \sum_{v\notin\mathbf{x}^*}\mathbb{E}[X_v] + \sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t]$$

Theorem: Algorithm randWeightedVC computes a vertex cover $\mathbf{x}$ with $\mathbb{E}[\mathbf{c}\cdot\mathbf{x}] \leq 2\mathbf{c}\cdot\mathbf{x}^*$.

Quality: (Finishing Up)

$$\begin{aligned}\sum_{v\in V}\mathbb{E}[X_v] &= \sum_{v\notin\mathbf{x}^*}\mathbb{E}[X_v]+\sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t]\\ &\overset{\text{Lemma 5}}{\leq} 2\sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t]\end{aligned}$$

Theorem: Algorithm randWeightedVC computes a vertex cover $\mathbf{x}$ with $\mathbb{E}[\mathbf{c}\cdot\mathbf{x}] \leq 2\mathbf{c}\cdot\mathbf{x}^*$.

Quality: (Finishing Up)

$$\begin{aligned}
\sum_{v\in V}\mathbb{E}[X_v] &= \sum_{v\notin\mathbf{x}^*}\mathbb{E}[X_v] + \sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t] \\
&\overset{\text{Lemma 5}}{\leq} 2\sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t] \\
&\leq 2\sum_{t\in\mathbf{x}^*} c_t \qquad X_t = 0 \text{ or } X_t = c_t
\end{aligned}$$

Theorem: Algorithm randWeightedVC computes a vertex cover $\mathbf{x}$ with $\mathbb{E}[\mathbf{c}\cdot\mathbf{x}] \leq 2\mathbf{c}\cdot\mathbf{x}^*$.

Quality: (Finishing Up)

$$\begin{aligned}
\sum_{v\in V}\mathbb{E}[X_v] &= \sum_{v\notin\mathbf{x}^*}\mathbb{E}[X_v] + \sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t] \\
&\stackrel{\textit{Lemma } 5}{\leq} 2\sum_{t\in\mathbf{x}^*}\mathbb{E}[X_t] \\
&\leq 2\sum_{t\in\mathbf{x}^*} c_t \qquad X_t = 0 \text{ or } X_t = c_t \\
&= 2\mathbf{c}\cdot\mathbf{x}^*
\end{aligned}$$

■

More on Vertex Cover

- There are simple deterministic linear time 2-approximations. (Special case of set covering)
- Best known algorithm: ratio $2 - \Theta(1/\sqrt{\log n})$
- fixed parameter algorithms: [Niedermeyer Rossmanith] find optimal solution in time $\mathcal{O}\left(kn + k^2 1.292^k\right)$ if $|\mathbf{x}| \leq k$. Key idea: (clever) exhaustive search + problem reductions.
  Example: include nodes of degree $\geq k$.
  include neighbors of degree 1 nodes

Scheduling on Unrelated Parallel Machines

[Vazirani Chapter 17]

J: set of n jobs

M: set of m machines

p_{ij} : processing time of job j on machine i

$\mathbf{x}(j)$: Machine where job j is executed

L_i: $\sum_{\{j:\mathbf{x}(j)=i\}} p_{ij}$, load of machine i

Objective: Minimize Makespan $L_{\max} = \max_i L_i$

A Misguided ILP model

minimize t subject to

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

$$\forall i \in M: \sum_{j \in J} x_{ij} p_{ij} \leq t$$

$$\forall i \in M, j \in J: x_{ij} \in \{0,1\}$$

The problem with this formulation

minimize t subject to

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

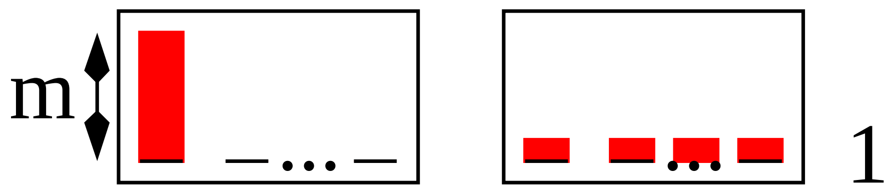
$$\forall i \in M : \sum_{j \in J} x_{ij} p_{ij} \leq t$$

$$\forall i \in M, j \in J : x_{ij} \in \{0,1\}$$

One Job, size m everywhere.

Linear relaxation: makespan 1

Optimal solution: makespan m



The linear relaxation is far away from the optimal solution and hence yields little useful information

LP-speak: integrality gap m

The problem with this formulation

minimize t subject to

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

$$\forall i \in M : \sum_{j \in J} x_{ij} p_{ij} \leq t$$

$$\forall i \in M, j \in J : x_{ij} \in \{0, 1\}$$

In ILP, we always have $x_{ij} = 0$ if $p_{ij} > t$

This is lost in the linear relaxation: some x_{ij} may get small values

We cannot add this constraint since it is not a linear constraint

A Refined LP Relaxation (Parametric Pruning)

guess makespan T e.g., binary search

feasible assignments: $S_T := \{(i,j) : p_{ij} \leq T\}$

A Refined LP Relaxation (Parametric Pruning)

guess makespan T e.g., binary search

feasible assignments: $S_T := \{(i,j) : p_{ij} \leq T\}$

LP(T):

$$\forall j \in J : \sum_{\{i:(i,j)\in S_T\}} x_{ij} = 1$$

$$\forall i \in M : \sum_{\{j:(i,j)\in S_T\}} x_{ij} p_{ij} \leq T$$

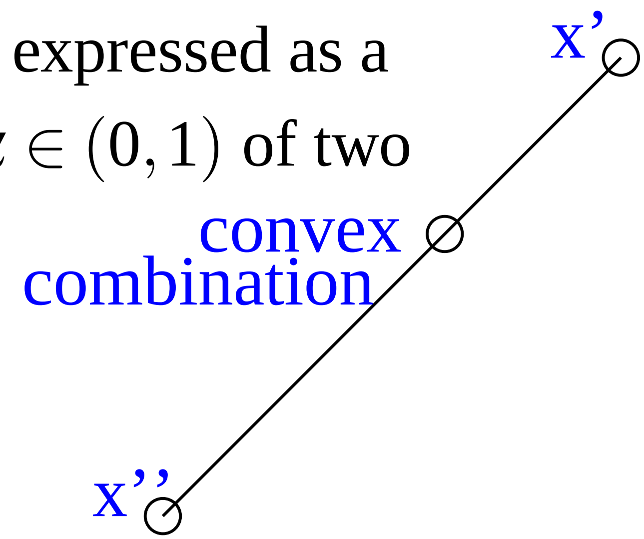
$$\forall (i,j) \in S_T : x_{ij} \geq 0$$

No objective function! We only look for a *feasible* solution

More LP-speak

Consider a solution $\mathbf{x}$ of a given LP.

$\mathbf{x}$ is an extreme point solution if it cannot be expressed as a convex combination $\alpha\mathbf{x}' + (1-\alpha)\mathbf{x}''$ with $\alpha \in (0,1)$ of two other feasible solutions $\mathbf{x}'$ and $\mathbf{x}''$.



Theorem 6. *$\mathbf{x} \in \mathbb{R}^r$ is an extreme point solution iff it corresponds to setting r linearly independent constraints to equality.*

Proof. not here. □

$S_T := \left\{(i,j) : p_{ij} \leq T\right\}$

LP(T):

$\forall j \in J : \quad \sum_{\{i:(i,j)\in S_T\}} x_{ij} = 1$

$\forall i \in M : \quad \sum_{\{j:(i,j)\in S_T\}} x_{ij} p_{ij} \leq T$

$\forall (i,j) \in S_T : x_{ij} \geq 0$

Lemma 7. *An extreme point solution of LP(T) has at most $n+m$ nonzero variables.*

Proof. $r = |S_T|$ variables

$n+m$ constraints (except ≥ 0)

$\overset{Thm6}{\rightsquigarrow} \geq r-(n+m)$ of the ≥ 0 constraints are tight. □

Lemma 7. *An extreme point solution of $LP(T)$ has at most $n+m$ nonzero variables.*

Corollary 8. *An extreme point solution of $LP(T)$ sets $\geq n-m$ jobs integrally.*

Proof.

a integrally set jobs $\rightsquigarrow$ a nonzero entries in $\mathbf{x}$

$n-a$ fractionally set jobs $\rightsquigarrow \geq 2(n-a)$ nonzero entries in $\mathbf{x}$

Lemma 7$\rightsquigarrow$

$2(n-a)+a \leq n+m$

$\Leftrightarrow a \geq n-m$ □

One Reason why LP Relaxation is Useful

Theorem 6. $\mathbf{x} \in \mathbb{R}^r$ *is an extreme point solution iff it corresponds to setting r linearly independent constraints to equality.*

Theorem 6 often implies that only few variables need to be rounded to obtain an solution of the ILP.

…this does not mean rounding the remaining ones is easy.

The Algorithm: Top Level

α:= makespan one gets by assigning each job to the fastest machine for it

α is an upper bound for the optimal makespan

The Algorithm: Top Level

α:= makespan one gets by assigning each job to the fastest machine for it

α is an upper bound for the optimal makespan

Use binary search in the range $\lfloor \alpha/m \rfloor, \alpha$

to find the smallest T such that $LP(T)$ has a feasible solution $\mathbf{x}$

The Algorithm: Top Level

$\alpha :=$ makespan one gets by assigning each job to the fastest machine for it

α is an upper bound for the optimal makespan

Use binary search in the range $\lfloor \alpha/m \rfloor, \alpha$

to find the smallest T such that $LP(T)$ has a feasible solution

For this T, find an extremal point solution $\mathbf{x}$

The Algorithm: Top Level

α:= makespan one gets by assigning each job to the fastest machine for it

α is an upper bound for the optimal makespan

Use binary search in the range $\lfloor \alpha/m \rfloor, \alpha$

to find the smallest T such that $LP(T)$ has a feasible solution

For this T, find an extremal point solution $\mathbf{x}$

assign integrally set jobs in $\mathbf{x}$

The Algorithm: Top Level

$\alpha :=$ makespan one gets by assigning each job to the fastest machine for it

α is an upper bound for the optimal makespan

Use binary search in the range $\lfloor \alpha/m \rfloor, \alpha$

to find the smallest T such that $LP(T)$ has a feasible solution

For this T, find an extremal point solution $\mathbf{x}$

assign integrally set jobs in $\mathbf{x}$

deal with the fractionally set jobs // Rounding

Example

p_{ij}	1	2	3	4	5
1	2	2	4	2	4
2	4	3	3	4	4
3	3	3	3	3	2
4	2	4	4	4	2

Four machines, five jobs

For each job, the best machine for it is marked in blue.

Example

p_{ij}	1	2	3	4	5	Each job on fastest machine:
1	2	2	4	2	4	$6 =: \alpha$
2	4	3	3	4	4	3
3	3	3	3	3	2	5
4	2	4	4	4	2	2

Initial guess for the makespan is 6

Using binary search, we find smallest makespan in the range $[6/4, 6]$ that can be achieved using a fractional assignment

Example

p_{ij}	1	2	3	4	5	Each job on fastest machine:
1	2	2	4	2	4	$6 =: \alpha$
2	4	3	3	4	4	3
3	3	3	3	3	2	5
4	2	4	4	4	2	2

Solution of LP(3):

x_{ij}	1	2	3	4	5
1	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	0
2	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
3	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0
4	$\frac{1}{2}$	0	0	0	1

Dealing with Fractionally Set Jobs

Consider the bipartite graph

$H := (J' \cup M', E')$ where

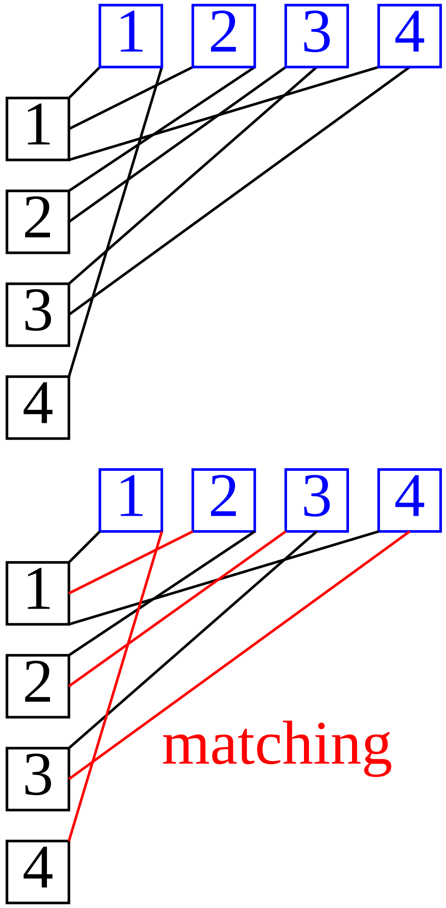
$J' := \{ j \in J : \exists i : 0 < x_{ij} < 1 \}$

$M' := \{ i \in M : \exists j : 0 < x_{ij} < 1 \}$

$E' := \{ \{i, j\} : x_{ij} \neq 0, i \in M', j \in J' \}$

Idea: Find a perfect matching in H

assign jobs according to that matching



Matching

A set of edges M that do not have any nodes in common, i.e., (V,M) has maximum degree one.

Perfect Matching

A matching of size $|V|/2$, i.e., all nodes are matched

Lemma 8. *H is a pseudo forest, i.e., each connected component $H_C = (V_C, E_C)$ has $|E_C| \leq |V_C|$ (a tree plus, possibly, one edge)*

Proof. It suffices to show this for the larger graph

$G := (J \cup M, E)$ where

$E := \{\{i, j\} : x_{ij} \neq 0, i \in M, j \in J\}$

Consider a connected component H_C of G.

restrict $\mathbf{x}$ and $LP(T)$ to H_C: $\mathbf{x}_C$, $LP_C(T)$

$\mathbf{x}_C$ is extreme point solution of $LP_C(T)$

(Otherwise, $\mathbf{x}$ itself could not be extreme point solution)

Lemma 7$\rightsquigarrow$ $LP_C(T)$ has $\leq |V_C|$ nonzero vars.,

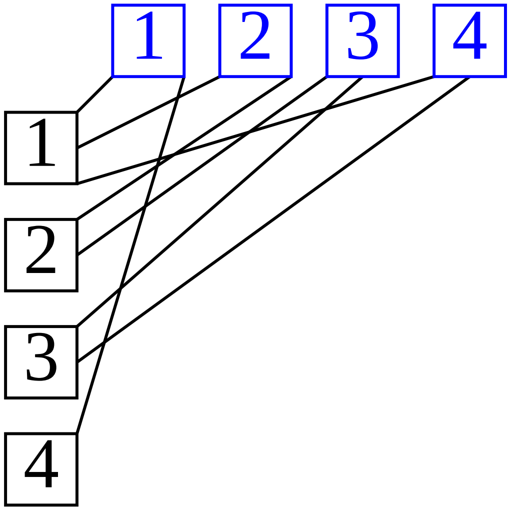
i.e., H_C has $\leq |V_C|$ edges. □

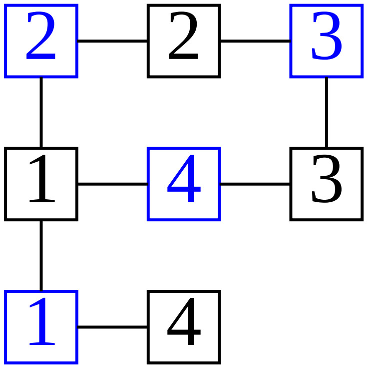
Example

p_{ij}	1	2	3	4	5
1	2	2	4	2	4
2	4	3	3	4	4
3	3	3	3	3	2
4	2	4	4	4	2

x_{ij}	1	2	3	4	5
1	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	0
2	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
3	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0
4	$\frac{1}{2}$	0	0	0	1

$T^* = 3$





Lemma 9. *H has a perfect matching*

Proof. We give an algorithm:

$\mathscr{M} := \emptyset$

invariant H is a bipartite pseudo forest

invariant all degree one nodes are machines

while $\exists i \in M'$ with degree one **do**

$e = \{i, j\} :=$ the sole edge incident to i

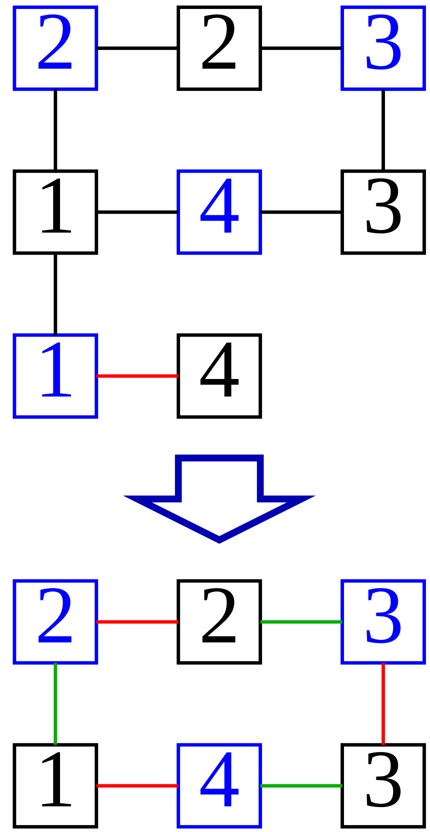
$\mathscr{M} := \mathscr{M} \cup \{e\}$

remove i, j and incident edges

assert H is a collection of disjoint even cycles

foreach cycle $C \in H$ **do**

match alternating edges in C



□

Theorem 10. *The algorithm achieves an approximation guarantee of factor 2 for scheduling unrelated parallel machines.*

Proof. Consider solution $\mathbf{x}$ of LPT(T^*)
makespan due to jobs set integrally in $\mathbf{x}$ is $\leq T^* \leq$ opt.
In addition, each machine i receives ≤ 1 job j from the matching $\mathscr{M} \subseteq H$.
$p_{i,j} \leq T^* \leq$ opt since otherwise $\{i, j\} \notin H$ □