Advanced Algorithms (Fall 2023)

Linear Programming Rounding

Lecturers: 尹一通, 刘景铖, 栗师

Nanjing University
Outline

1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - $s$-$t$ Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Algorithm Design Based on Linear Programming (LP)

Opti. Problem $X \iff$ Integer Program (IP) $\overset{\text{relax}}{\implies}$ LP

Integer programming is NP-hard; linear programming is in P

For some problems $LP \equiv IP \implies$ exact algorithms

For some problems, $LP \neq IP$

- solve LP to obtain a fractional solution,
- **round** it to an integral solution

$\implies$ approximation algorithms
min \ 7x_1 + 4x_2
\[ x_1 + x_2 \geq 5 \]
\[ x_1 + 2x_2 \geq 6 \]
\[ 4x_1 + x_2 \geq 8 \]
\[ x_1 , x_2 \geq 0 \]

- optimum solution:  
  \[ x_1 = 1 , x_2 = 4 \]
- optimum value =  
  \[ 7 \times 1 + 4 \times 4 = 23 \]
- general case: many variables and constraints, but objective and constraints are linear
Standard Form of Linear Programs

\[
\begin{align*}
\text{min} & \quad c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\
\text{s.t.} & \quad a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,n} x_n \geq b_1 \\
& \quad a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,n} x_n \geq b_2 \\
& \quad \vdots \quad \vdots \quad \vdots \\
& \quad a_{m,1} x_1 + a_{m,2} x_2 + \cdots + a_{m,n} x_n \geq b_m \\
& \quad x_1, x_2, \cdots, x_n \geq 0
\end{align*}
\]

- \( n \): number of variables
- \( m \): number of constraints
- \( \leq \): constraints? equalities?
- Variables can be negative? maximization problem?
Standard Form of Linear Programs

\[ x := \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n, \quad c := \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n, \]

\[ A := \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \in \mathbb{R}^{n \times m}, \quad b := \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{R}^m. \]
\[
\begin{align*}
\text{min} & \quad c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\
\text{subject to} & \quad a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,n} x_n \geq b_1 \\
& \quad a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,n} x_n \geq b_2 \\
& \quad \vdots \quad \vdots \quad \vdots \\
& \quad a_{m,1} x_1 + a_{m,2} x_2 + \cdots + a_{m,n} x_n \geq b_m \\
& \quad x_1, x_2, \ldots, x_n \geq 0
\end{align*}
\]

• \(\geq\): coordinate-wise less than or equal to

\[
\begin{align*}
\text{Standard Form of Linear Program} \\
\text{min} & \quad c^T x \\
\text{subject to} & \quad Ax \geq b \\
& \quad x \geq 0
\end{align*}
\]
[Fourier, 1827]: Fourier-Motzkin elimination method

[Kantorovich, Koopmans 1939]: formulated the general linear programming problem

[Dantzig 1946]: simplex method

[Khachiyan 1979]: ellipsoid method, polynomial time, proved linear programming is in P

[Karmarkar, 1984]: interior-point method, polynomial time, algorithm is practical
feasible region: the set of \( x \)'s satisfying \( Ax \geq b, x \geq 0 \)

feasible region is a polyhedron

if every coordinate has an upper and lower bound in the polyhedron, then the polyhedron is a polytope

\( x \) is a convex combination of \( x^{(1)}, x^{(2)}, \ldots, x^{(t)} \) if the following condition holds: there exist \( \lambda_1, \lambda_2, \ldots, \lambda_t \in [0, 1] \) such that

\[
\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \quad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x
\]

the set of convex combinations of \( x^{(1)}, x^{(2)}, \ldots, x^{(t)} \) is called the convex hull of these points

let \( P \) be polytope, \( x \in P \). If there are no other points \( x', x'' \in P \) such that \( x \) is a convex combination of \( x' \) and \( x'' \), then \( x \) is called a vertex/extreme point of \( P \)
Lemma  A polytope has finite number of vertices, and it is the convex hull of the vertices.

Lemma  Let \( x \in \mathbb{R}^n \) be an extreme point in a \( n \)-dimensional polytope. Then, there are \( n \) constraints in the definition of the polytope, such that \( x \) is the unique solution to the linear system obtained from the \( n \) constraints by replacing inequalities to equalities.

Lemma  If the feasible region of a linear program is a polytope, then the optimum value can be attained at some vertex of the polytope.

Special cases (for minimization linear programs):
- if feasible region is empty, then its value is \( \infty \)
- if the feasible region is unbounded, then its value can be \( -\infty \)
## Algorithms for Linear Programming

<table>
<thead>
<tr>
<th>algorithm</th>
<th>running time</th>
<th>practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplex Method</td>
<td>exponential time</td>
<td>fast</td>
</tr>
<tr>
<td>Ellipsoid Method</td>
<td>polynomial time</td>
<td>slow</td>
</tr>
<tr>
<td>Interior Point Method</td>
<td>polynomial time</td>
<td>fast</td>
</tr>
</tbody>
</table>
Simplex Method

- [Dantzig, 1946]

- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex

- the number of iterations might be exponentially large; but algorithm runs fast in practice
- [Spielman-Teng, 2002]: smoothed analysis
Interior Point Method

- [Karmarkar, 1984]
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution
- polynomial time
Ellipsoid Method

- [Khachiyan, 1979]
- used to decide if the feasible region is empty or not

- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsoid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the ellipsoid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat

- polynomial time, but impractical
Q: The exact running time of these algorithms?

- it depends on many parameters: \#variables, \#constraints, \#(non-zero coefficients), magnitude of integers
- precision issue

Open Problem
Can linear programming be solved in strongly polynomial time algorithm?
Applications of Linear Programming

- **domain**: computer science, mathematics, operations research, economics
- **types of problems**: transportation, scheduling, clustering, network routing, resource allocation, facility location

**Research Directions**

- polynomial time exact algorithm
- polynomial time approximation algorithm
- sub-routines for the branch-and-bound method for integer programming
- other algorithmic models: online algorithm, distributed algorithms, dynamic algorithms, fast algorithms
**Simple Example: Brewery Problem** *

- Small brewery produces ale and beer.
- Production limited by scarce resources: corn, hops, barley malt.
- Recipes for ale and beer require different proportions of resources.

<table>
<thead>
<tr>
<th>Beverage</th>
<th>Corn (pounds)</th>
<th>Hops (pounds)</th>
<th>Malt (pounds)</th>
<th>Profit ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ale (barrel)</td>
<td>5</td>
<td>4</td>
<td>35</td>
<td>13</td>
</tr>
<tr>
<td>Beer (barrel)</td>
<td>15</td>
<td>4</td>
<td>20</td>
<td>23</td>
</tr>
<tr>
<td>Constraint</td>
<td>480</td>
<td>160</td>
<td>1190</td>
<td></td>
</tr>
</tbody>
</table>

- How can brewer maximize profits?

Brewery Problem

<table>
<thead>
<tr>
<th>Beverage</th>
<th>Corn (pounds)</th>
<th>Hops (pounds)</th>
<th>Malt (pounds)</th>
<th>Profit ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ale (barrel)</td>
<td>5</td>
<td>4</td>
<td>35</td>
<td>13</td>
</tr>
<tr>
<td>Beer (barrel)</td>
<td>15</td>
<td>4</td>
<td>20</td>
<td>23</td>
</tr>
<tr>
<td>Constraint</td>
<td>480</td>
<td>160</td>
<td>1190</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{max} \quad 13x + 23y \\
5x + 15y \leq 480 \\
4x + 4y \leq 160 \\
35x + 20y \leq 1190 \\
x, y \geq 0
\]
Outline

1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - s-t Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Def. A polytope $P \subseteq \mathbb{R}^n$ is said to be integral, if all vertices of $P$ are in $\mathbb{Z}^n$.

- For some combinatorial optimization problems, a polynomial-sized LP $Ax \leq b$ already defines an integral polytope, whose vertices correspond to valid integral solutions.
- Such a problem can be solved directly using the LP:

$$\max / \min \quad c^T x \quad Ax \leq b.$$
1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - \( s-t \) Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Example: Bipartite Matching Polytope

**Maximum Weight Bipartite Matching**

**Input:** bipartite graph $G = (L \cup R, E)$
edge weights $w \in \mathbb{Z}_E > 0$

**Output:** a matching $M \subseteq E$ so as to maximize $\sum_{e \in M} w_e$

**LP Relaxation**

$max \sum_{e \in E} w_e x_e$

$\sum_{e \in \delta(v)} x_e \leq 1 \quad \forall v \in L \cup R$

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

$\sum_j x_{ij} \leq 1 \quad \sum_i x_{ij} \leq 1$

- In IP: $x_e \in \{0, 1\}: e \in M$?
- $\chi^M \in \{0, 1\}^E$: $\chi^M_e = 1$ iff $e \in M$

**Theorem** The LP polytope is integral: It is the convex hull of $\{\chi^M : M \text{ is a matching}\}$. 
Theorem: The LP polytope is integral: It is the convex hull of \( \{ \chi^M : M \text{ is a matching} \} \).

Proof.

- Take \( x \) in the polytope \( P \).
- Prove: \( x \) non integral \( \implies \) \( x \) non-vertex.
- Find \( x', x'' \in P: x' \neq x'', x = \frac{1}{2}(x' + x'') \).
- Case 1: Fractional edges contain a cycle.
  - Color edges in cycle blue and red.
  - \( x' \): +\( \epsilon \) for blue edges, -\( \epsilon \) for red edges.
  - \( x'' \): -\( \epsilon \) for blue edges, +\( \epsilon \) for red edges.
- Case 2: Fractional edges form a forest.
  - Color edges in a leaf-leaf path blue and red.
  - \( x' \): +\( \epsilon \) for blue edges, -\( \epsilon \) for red edges.
  - \( x'' \): -\( \epsilon \) for blue edges, +\( \epsilon \) for red edges.
Outline

1 Linear Programming and Rounding

2 Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - $s$-$t$ Flow Polytope
   - Weighted Interval Scheduling Problem

3 Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Example: $s$-$t$ Flow Polytope

Flow Network

- directed graph $G = (V, E)$, source $s \in V$, sink $t \in V$, edge capacities $c_e \in \mathbb{Z}_{>0}$, $\forall e \in E$
- $s$ has no incoming edges, $t$ has no outgoing edges
Def. A \textit{s-t flow} is a vector $f \in \mathbb{R}^E_{\geq 0}$ satisfying the following conditions:

- $\forall e \in E, 0 \leq f_e \leq c_e$ \hspace{1cm} (capacity constraints)
- $\forall v \in V \setminus \{s, t\}$,

$$\sum_{e \in \delta^{\text{in}}(v)} f_e = \sum_{e \in \delta^{\text{out}}(v)} f_e$$ \hspace{1cm} (flow conservation)

The value of flow $f$ is defined as:

$$\text{val}(f) := \sum_{e \in \delta^{\text{out}}(s)} f_e = \sum_{e \in \delta^{\text{in}}(t)} f_e$$
Maximum Flow Problem

**Input:** flow network \((G = (V, E), c, s, t)\)

**Output:** maximum value of a \(s-t\) flow \(f\)

- Ford-Fulkerson method
- **Maximum-Flow Min-Cut Theorem:** value of the maximum flow is equal to the value of the minimum \(s-t\) cut
- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva, 2022]: nearly linear-time algorithm
**LP for Maximum Flow**

\[
\begin{align*}
\text{max} & \quad \sum_{e \in \delta^{\text{in}}(t)} x_e \\
& \quad \sum_{e \in \delta^{\text{out}}(v)} x_e - \sum_{e \in \delta^{\text{in}}(v)} x_e = 0 \quad \forall v \in V \setminus \{s, t\} \\
& \quad x_e \geq 0 \quad \forall e \in E
\end{align*}
\]

**Theorem**  The LP polytope is integral.

**Sketch of Proof.**

- Take any \( s-t \) flow \( x \); consider fractional edges \( E' \)
- Every \( v \notin \{s, t\} \) must be incident to 0 or \( \geq 2 \) edges in \( E' \)
- Ignoring the directions of \( E' \), it contains a cycle, or a \( s-t \) path
- We can increase/decrease flow values along cycle/path

\[\Box\]
Outline

1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - s-t Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Weighted Interval Scheduling Problem

Input: $n$ activities, activity $i$ starts at time $s_i$, finishes at time $f_i$, and has weight $w_i > 0$

$i$ and $j$ can be scheduled together iff $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: maximum weight subset of jobs that can be scheduled

- optimum value = 220
- Classic Problem for Dynamic Programming
Weighted Interval Scheduling Problem

**Linear Program**

\[
\begin{align*}
\text{max } & \sum_{j \in [n]} x_j w_j \\
\sum_{j \in [n]: t \in [s_j, f_j]} x_j & \leq 1 \quad \forall t \in [T] \\
x_j & \geq 0 \quad \forall j \in [n]
\end{align*}
\]

**Theorem** The LP polytope is integral.

**Def.** A matrix \( A \in \mathbb{R}^{m \times n} \) is said to be totally unimodular (TUM), if every sub-square of \( A \) has determinant in \( \{-1, 0, 1\} \).

**Theorem** If a polytope \( P \) is defined by \( Ax \leq b, x \geq 0 \) with a totally unimodular matrix \( A \) and integral \( b \), then \( P \) is integral.

**Lemma** A matrix \( A \in \{0, 1\}^{m \times n} \) where the 1’s on every column form an interval is TUM.

- So, the matrix for the LP is TUM, and the polytope is integral.
**Theorem**  If a polytope \( P \) is defined by \( Ax \leq b, x \geq 0 \) with a totally unimodular matrix \( A \) and integral \( b \), then \( P \) is integral.

**Proof.**
- Every vertex \( x \in P \) is the unique solution to the linear system (after permuting coordinates): \[
\begin{pmatrix}
A' & 0 \\
0 & I
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} =
\begin{pmatrix} b' \\
0
\end{pmatrix},
\]
- \( A' \) is a square submatrix of \( A \) with \( \det(A') = \pm 1 \), \( b' \) is a sub-vector of \( b \),
- and the rows for \( b' \) are the same as the rows for \( A' \).
- Let \( x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \), so that \( A'x_1 = b' \) and \( x_2 = 0 \).
- Cramer’s rule: \( x_i^1 = \frac{\det(A_i'|b)}{\det(A')} \) for every \( i \implies x_i^1 \) is integer
- \( A_i'|b \): the matrix of \( A' \) with the \( i \)-th column replaced by \( b \)

**Obs.** If \( A \) is TUM, then \( A^T \) is also TUM.
Lemma  Let $A' \in \{0, \pm 1\}^{n \times n}$ such that every row of $A'$ contains at most one 1 and one $-1$. Then $\det(A') \in \{0, \pm 1\}$.

Proof.

- wlog assume every row of $A'$ contains one 1 and one $-1$
- otherwise, we can reduce the matrix
- treat $A'$ as a directed graph: columns $\equiv$ vertices, rows $\equiv$ arcs
- $\#\text{edges} = \#\text{vertices} \implies$ underlying undirected graph contains a cycle $\implies \det(A') = 0$

Lemma  Let $A \in \{0, \pm 1\}^{m \times n}$ such that every row of $A$ contains at most one 1 and one $-1$. Then $A$ is TUM.

Coro. The matrix for $s-t$ flow polytope is TUM; thus, the polytope is integral.
Lemma  Let $A' \in \{0, \pm 1\}^{n \times n}$ such that every row of $A'$ contains at most one $1$ and one $-1$. Then $\det(A') \in \{0, \pm 1\}$.

Lemma  A matrix $A \in \{0, 1\}^{m \times n}$ where the $1$'s on every row form an interval is TUM.

Proof.
- take any square submatrix $A'$ of $A$,
- the $1$'s on every row of $A'$ form an interval.
- $A'M$ is a matrix satisfying condition of first lemma, where
  \[
  M = \begin{pmatrix}
  1 & -1 & 0 & \cdots & 0 \\
  0 & 1 & -1 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & 1 & -1 \\
  0 & 0 & \cdots & 0 & 1
  \end{pmatrix}.
  \]
  $\det(M) = 1$.
- $\det(A'M) \in \{0, \pm 1\} \implies \det(A') \in \{0, \pm 1\}$. 

}\]
**Lemma**  The edge-vertex incidence matrix $A$ of a bipartite graph is totally-unimodular.

**Proof.**
- $G = (L \cup R, E)$: the bipartite graph
- $A'$: obtained from $A$ by negating columns correspondent to $R$
- each row of $A'$ has exactly one $+1$, and exactly one $-1$
- $\implies A'$ is TUM $\iff$ $A$ is TUM

remark: bipartiteness is needed. The edge-vertex incidence matrix
\[
\begin{pmatrix}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{pmatrix}
\]
of a triangle has determinant 2.

**Coro.** Bipartite matching polytope is integral.
Outline

1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - $s-t$ Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

\[
\begin{align*}
\text{min} & \quad c^T x \\
Ax & \geq b \\
x & \in \{0, 1\}^n
\end{align*}
\]

Linear Program Relaxation

\[
\begin{align*}
\text{min} & \quad c^T x \\
Ax & \geq b \\
x & \in [0, 1]^n
\end{align*}
\]

- LP $\leq$ IP
- Integer programming is NP-hard, linear programming is in P
- Solve LP to obtain a fractional $x \in [0, 1]^n$.
- Round it to an integral $\tilde{x} \in \{0, 1\}^n \iff$ solution for $X$
- Prove $c^T \tilde{x} \leq \alpha \cdot c^T x$, then $c^T \cdot \tilde{x} \leq \alpha \cdot \text{LP} \leq \alpha \cdot \text{IP} = \alpha \cdot \text{opt}$
- $\implies \alpha$-approximation
Def. The ratio between \( \text{IP} = \text{opt} \) and LP is called the **integrality gap** of the LP relaxation.

- The approximation ratio based on this analysis can not be better than the worst integrality gap.
Outline

1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - $s$-$t$ Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Weighted Vertex Cover Problem

**Input:** graph $G = (V, E)$, vertex weights $w \in \mathbb{Z}^V_{>0}$

**Output:** vertex cover $S$ of $G$, to minimize $\sum_{v \in S} w_v$
\( x_v \in \{0, 1\}, \forall v \in V \): indicate if we include \( v \) in the vertex cover

**Integer Program**

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} w_v x_v \\
x_u + x_v & \geq 1 \quad \forall (u,v) \in E \\
x_v & \in \{0, 1\} \quad \forall v \in V
\end{align*}
\]

**LP Relaxation**

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} w_v x_v \\
x_u + x_v & \geq 1 \quad \forall (u,v) \in E \\
x_v & \in [0, 1] \quad \forall v \in V
\end{align*}
\]

- \( \text{IP} := \text{value of integer program}, \text{LP} := \text{value of linear program} \)
- \( \text{LP} \leq \text{IP} = \text{opt} \)
Rounding Algorithm

1: Solve LP to obtain solution \( \{x^*_u\}_{u \in V} \)
\[ \therefore \text{So, } \text{LP} = \sum_{u \in V} w_u x^*_u \leq \text{IP} \]
2: return \( S := \{u \in V : x_u \geq 1/2\} \)

Lemma \( S \) is a vertex cover of \( G \).

Proof.
- Consider any \((u, v) \in E\): we have \( x^*_u + x^*_v \geq 1 \)
- So, \( x^*_u \geq 1/2 \) or \( x^*_v \geq 1/2 \) \( \implies \) \( u \in S \) or \( v \in S \). \( \square \)
Rounding Algorithm

1: Solve LP to obtain solution \( \{x_u^*\}_{u \in V} \)
\[ \triangleright \text{So, } \text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP} \]
2: return \( S := \{ u \in V : x_u \geq 1/2 \} \)

Lemma \( S \) is a vertex cover of \( G \).

Lemma \( \text{cost}(S) := \sum_{u \in S} w_u \leq 2 \cdot \text{LP} \).

Proof.
\[
\text{cost}(S) = \sum_{u \in S} w_u \leq \sum_{u \in S} w_u \cdot 2x_u^* = 2 \sum_{u \in S} w_u \cdot x_u^*
\leq 2 \sum_{u \in V} w_u \cdot x_u^* = 2 \cdot \text{LP}.
\]

Theorem The algorithm is a 2-approximation algorithm for weighted vertex cover.
Outline

1. Linear Programming and Rounding

2. Exact Algorithms Using LP: Integral Polytopes
   - Bipartite Matching Polytope
   - $s$-$t$ Flow Polytope
   - Weighted Interval Scheduling Problem

3. Approximation Algorithms Using LP: LP Rounding
   - 2-Approximation Algorithm for Weighted Vertex Cover
   - 2-Approximation Algorithm for Unrelated Machine Scheduling
Unrelated Machine Scheduling

**Input:** \( J, |J| = n \): jobs
\( M, |M| = m \): machines
\( p_{ij} \): processing time of job \( j \) on machine \( i \)

**Output:** assignment \( \sigma : J \mapsto M \):, so as to minimize makespan:

\[
\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_{ij}
\]
Assumption: we are given a target makespan $T$, and $p_{ij} \in [0, T] \cup \{\infty\}$

$x_{ij}$: fraction of $j$ assigned to $i$

$$
\sum_{i} x_{ij} = 1 \quad \forall j \in J \\
\sum_{j} p_{ij} x_{ij} \leq T \quad \forall i \in M \\
\quad \quad x_{ij} \geq 0 \quad \forall ij
$$
2-Approximate Rounding Algorithm of Shmoys-Tardos

\[ \sum_{g} x_{gj} = 1 \]
\[ \sum_{j} x_{gj} \leq 1 \]

**Obs.** \( x \) between \( J \) and sub-machines is a point in the bipartite-matching polytope, where all jobs in \( J \) are matched.
Recall bipartite matching polytope is integral.

- \( x \) is a **convex combination** of matchings.
- Any matching in the combination covers all jobs \( J \).

**Lemma** Any matching in the combination gives an schedule of makespan \( \leq 2T \).
**Lemma** Any matching in the combination gives an schedule of makespan $\leq 2T$.

Proof.
- focus on machine $i$, let $i_1, i_2, \cdots, i_a$ be the sub-machines for $i$
- assume job $k_t$ is assigned to sub-machine $i_t$.

$$(\text{load on } i) = \sum_{t=1}^{a} p_{ik_t} \leq p_{ik_1} + \sum_{t=2}^{a} \sum_{j} x_{i_{t-1}j} \cdot p_{ij}$$

$$\leq p_{ik_1} + \sum_{j} x_{ij} p_{ij} \leq T + T = 2T.$$
linear programming, simplex method, interior point method, ellipsoid method

integral LP polytopes: bipartite matching polytope, $s-t$ flow polytope, weighted interval scheduling polytope

approximation algorithm using LP rounding
  - 2-approximation algorithm for weighted vertex cover
  - 2-approximation for unrelated machine scheduling
<table>
<thead>
<tr>
<th>English</th>
<th>Chinese</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Program</td>
<td>线性规划</td>
</tr>
<tr>
<td>Integer Program</td>
<td>整数规划</td>
</tr>
<tr>
<td>Feasible Region</td>
<td>解域</td>
</tr>
<tr>
<td>Polyhedron</td>
<td>凸多面体</td>
</tr>
<tr>
<td>Polytope</td>
<td>有界凸多面体</td>
</tr>
<tr>
<td>Vertex/Extreme Point</td>
<td>顶点</td>
</tr>
<tr>
<td>Convex Combination</td>
<td>凸组合</td>
</tr>
<tr>
<td>Convex Hull</td>
<td>凸包</td>
</tr>
<tr>
<td>Dual</td>
<td>对偶</td>
</tr>
<tr>
<td>Totally Unimodular</td>
<td>完全单位模的</td>
</tr>
</tbody>
</table>