算法设计与分析(2024年春季学期)

Linear Programming

授课老师: 栗师

南京大学计算机科学与技术系

Outline

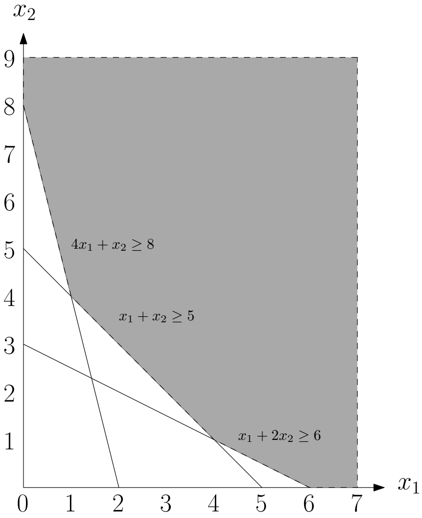
1. Linear Programming
   - Introduction
   - Preliminaries
   - Methods for Solving Linear Programs
2. Linear Programming Duality
3. Integral Polytopes: Exact Algorithms Using LP
   - Bipartite Matching Polytope
   - s-t Flow Polytope
   - Weighted Interval Scheduling Problem and Totally Unimodular Matrices

Outline

1. Linear Programming
   - Introduction
   - Preliminaries
   - Methods for Solving Linear Programs
2. Linear Programming Duality
3. Integral Polytopes: Exact Algorithms Using LP
   - Bipartite Matching Polytope
   - s-t Flow Polytope
   - Weighted Interval Scheduling Problem and Totally Unimodular Matrices

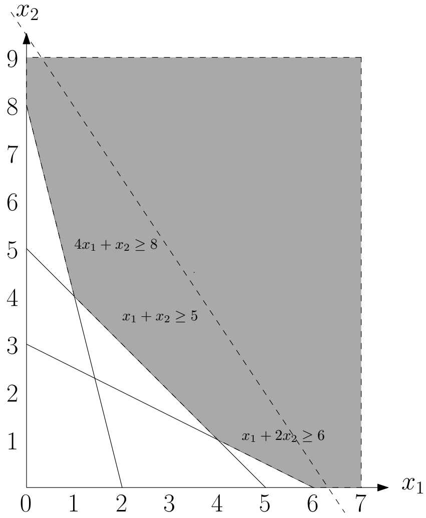
Example of Linear Programming

$$\begin{aligned} \min \quad & 7x_1 + 4x_2 \\ x_1 + x_2 &\geq 5 \\ x_1 + 2x_2 &\geq 6 \\ 4x_1 + x_2 &\geq 8 \\ x_1, x_2 &\geq 0 \end{aligned}$$



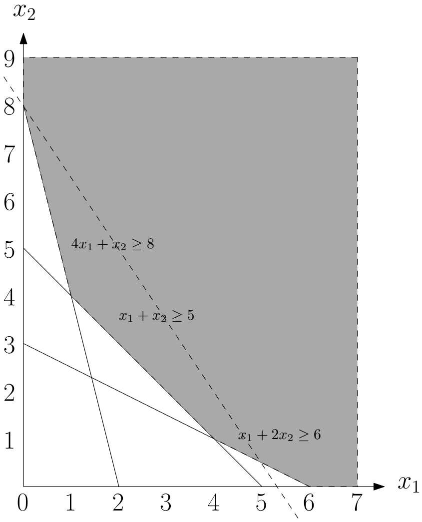
Example of Linear Programming

$$\begin{aligned} \min \qquad & 7x_1 + 4x_2 \\ x_1 + x_2 &\geq 5 \\ x_1 + 2x_2 &\geq 6 \\ 4x_1 + x_2 &\geq 8 \\ x_1, x_2 &\geq 0 \end{aligned}$$



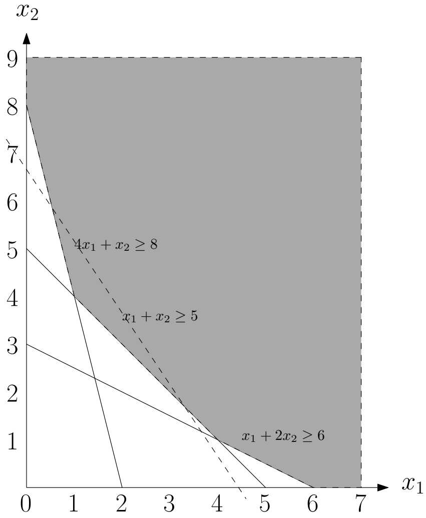
Example of Linear Programming

$$
\begin{aligned}
\min \quad & 7x_1 + 4x_2 \\
& x_1 + x_2 \geq 5 \\
& x_1 + 2x_2 \geq 6 \\
& 4x_1 + x_2 \geq 8 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Example of Linear Programming

$$\begin{aligned} \min \quad & 7x_1 + 4x_2 \\ x_1 + x_2 &\geq 5 \\ x_1 + 2x_2 &\geq 6 \\ 4x_1 + x_2 &\geq 8 \\ x_1, x_2 &\geq 0 \end{aligned}$$



Example of Linear Programming

$$
\begin{aligned}
\min \quad & 7x_1 + 4x_2 \\
x_1 + x_2 &\geq 5 \\
x_1 + 2x_2 &\geq 6 \\
4x_1 + x_2 &\geq 8 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

Example of Linear Programming

$$
\begin{aligned}
\min \qquad & 7x_1 + 4x_2 \\
& x_1 + x_2 \geq 5 \\
& x_1 + 2x_2 \geq 6 \\
& 4x_1 + x_2 \geq 8 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

- optimum point: $x_1 = 1, x_2 = 4$

Example of Linear Programming

$$
\begin{aligned}
\min \qquad & 7x_1 + 4x_2 \\
x_1 + x_2 &\geq 5 \\
x_1 + 2x_2 &\geq 6 \\
4x_1 + x_2 &\geq 8 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

- optimum point: $x_1 = 1, x_2 = 4$
- value $= 7 \times 1 + 4 \times 4 = 23$

Standard Form of Linear Programming

$$
\begin{aligned}
\min \qquad & c_1x_1 + c_2x_2 + \cdots + c_nx_n \qquad \text{s.t.} \\
\sum A_{1,1}x_1 + A_{1,2}x_2 + \cdots + A_{1,n}x_n &\geq b_1 \\
\sum A_{2,1}x_1 + A_{2,2}x_2 + \cdots + A_{2,n}x_n &\geq b_2 \\
\vdots \qquad \vdots \qquad \vdots \qquad & \vdots \\
\sum A_{m,1}x_1 + A_{m,2}x_2 + \cdots + A_{m,n}x_n &\geq b_m \\
x_1, x_2, \cdots, x_n &\geq 0
\end{aligned}
$$

Standard Form of Linear Programming

Let $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$, $\qquad c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$,

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Then, LP becomes

$$\begin{aligned} \min \quad & c^{\mathrm{T}} x \quad \text{s.t.} \\ & Ax \geq b \\ & x \geq 0 \end{aligned}$$

- $\geq$ means coordinate-wise greater than or equal to

Standard Form of Linear Programming

$$\min \quad c^{\mathrm{T}}x \quad \text{s.t.}$$
$$Ax \geq b$$
$$x \geq 0$$

- Linear programmings can be solved in polynomial time

Algorithm	Theory	Practice
Simplex Method	Exponential Time	Works Well
Ellipsoid Method	Polynomial Time	Slow
Internal Point Methods	Polynomial Time	Works Well

History

- [Fourier, 1827]: Fourier-Motzkin elimination method
- [Kantorovich, Koopmans 1939]: formulated the general linear programming problem

History

- [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 pratical

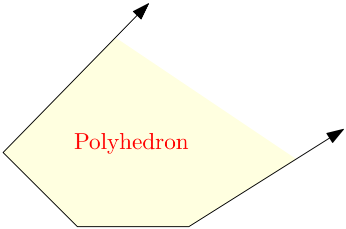
Outline

Preliminaries

- feasible region: the set of x's satisfying
  $Ax \geq b, x \geq 0$

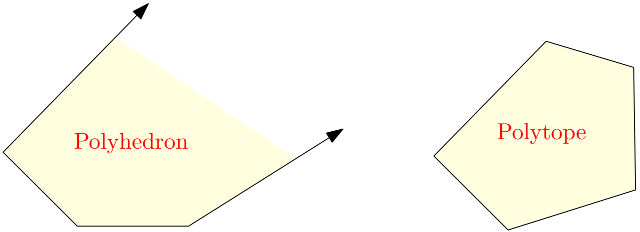
Preliminaries

- feasible region: the set of x's satisfying $Ax \geq b, x \geq 0$
- feasible region is a polyhedron



Preliminaries

- 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



Preliminaries

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

$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

Preliminaries

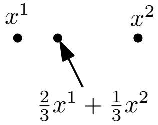
- x is a convex combination of $x^{(1)}, x^{(2)}, \cdots, x^{(t)}$ if the following condition holds: there exist $\lambda_1, \lambda_2, \cdots, \lambda_t \in [0, 1]$ such that

$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

Preliminaries

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

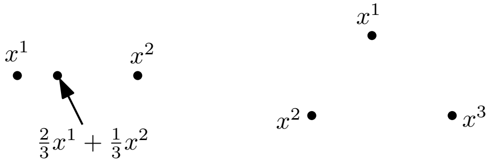
$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$



Preliminaries

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

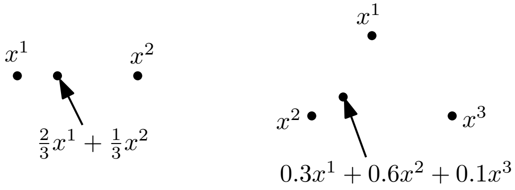
$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$



Preliminaries

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

$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

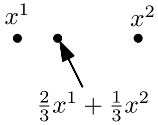


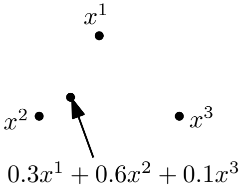
Preliminaries

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

$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

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



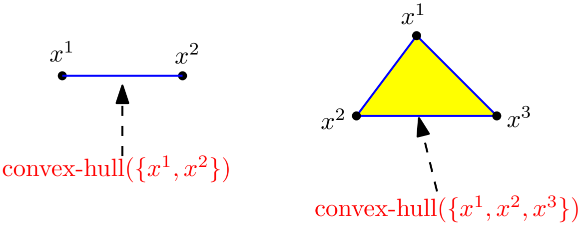


Preliminaries

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

$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

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

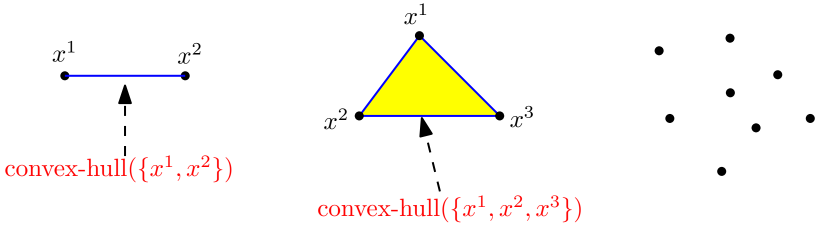


Preliminaries

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

$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

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

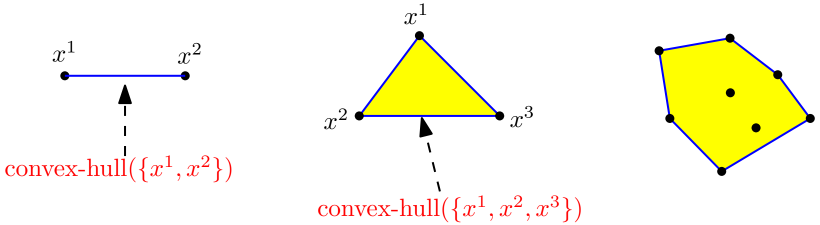


Preliminaries

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

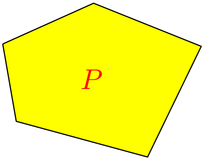
$$\lambda_1 + \lambda_2 + \cdots + \lambda_t = 1, \qquad \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \cdots + \lambda_t x^{(t)} = x$$

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



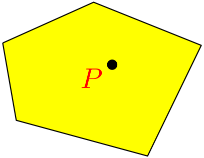
Preliminaries

- 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



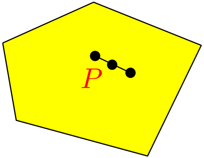
Preliminaries

- 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



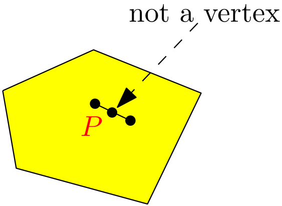
Preliminaries

- 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



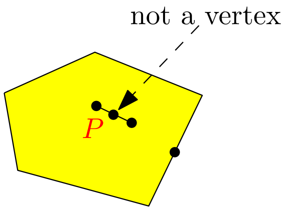
Preliminaries

- 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



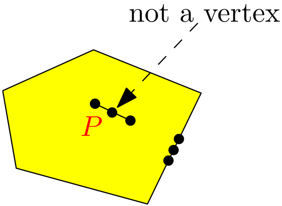
Preliminaries

- 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



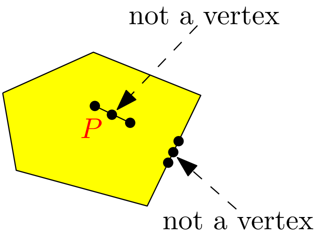
Preliminaries

- 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



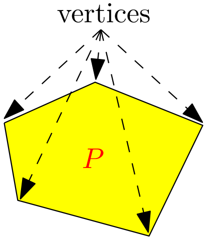
Preliminaries

- 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



Preliminaries

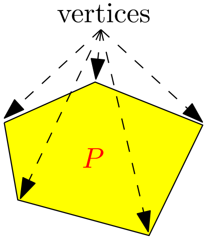
- let P be polytope, $x \in P$. If there are no other points $x', x'' \in P$ such that x is a convex combination of x' and x'', then x is called a vertex/extreme point of P



Preliminaries

- 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

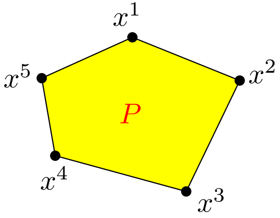
Lemma A polytope has finite number of vertices, and it is the convex hull of the vertices.



Preliminaries

- 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

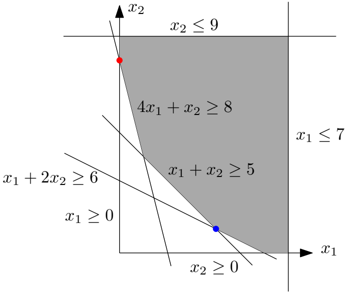
Lemma A polytope has finite number of vertices, and it is the convex hull of the vertices.



$$P = \text{convex-hull}(\{x^1, x^2, x^3, x^4, x^5\})$$

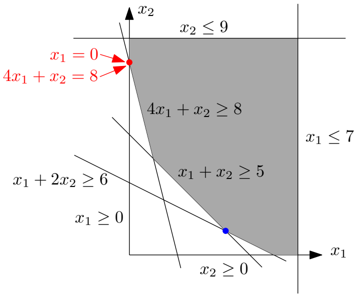
Preliminaries

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.



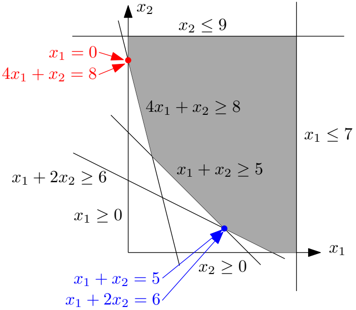
Preliminaries

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.



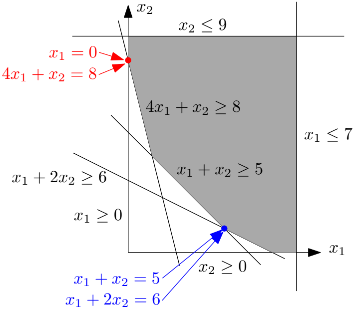
Preliminaries

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.



Preliminaries

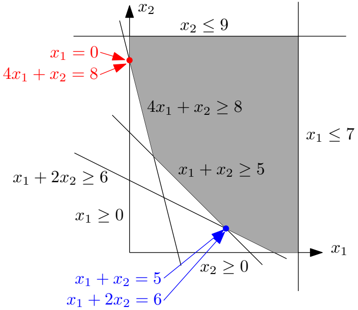
Lemma Let $x \in \mathbb{R}^n$ be an extreme point in a n-dimensional polytope. Then, there are n constraints in the definition of the polytope, such that x is the unique solution to the linear system obtained from the n constraints by replacing inequalities to equalities.



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

Preliminaries

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 opimum 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 ∞
- if the feasible region is unbounded, then its value can be $-\infty$

Outline

1. Linear Programming
   - Introduction
   - Preliminaries
   - Methods for Solving Linear Programs
2. Linear Programming Duality
3. Integral Polytopes: Exact Algorithms Using LP
   - Bipartite Matching Polytope
   - s-t Flow Polytope
   - Weighted Interval Scheduling Problem and Totally Unimodular Matrices

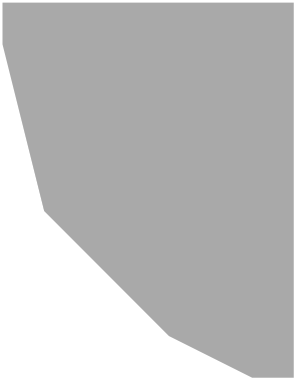
Simplex Method

- [Dantzig, 1946]
- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex

Simplex Method

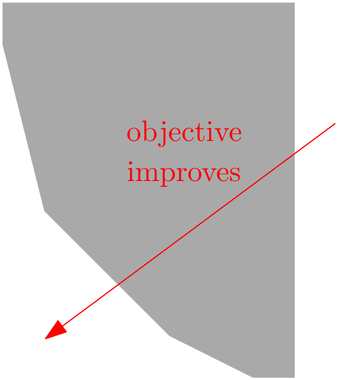
- [Dantzig, 1946]

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



Simplex Method

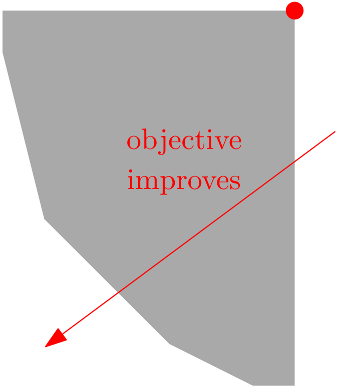
- [Dantzig, 1946]
- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex



Simplex Method

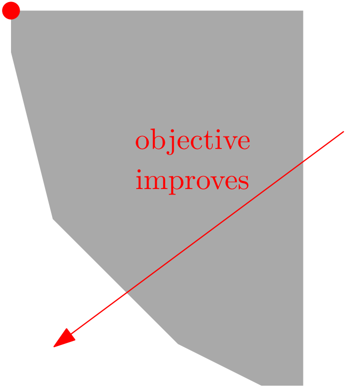
- [Dantzig, 1946]

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



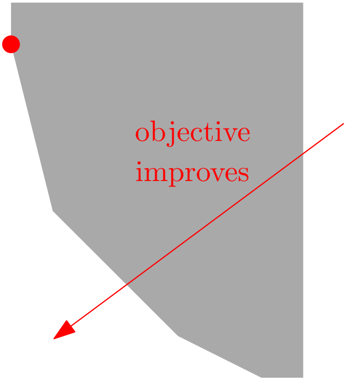
Simplex Method

- [Dantzig, 1946]
- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex



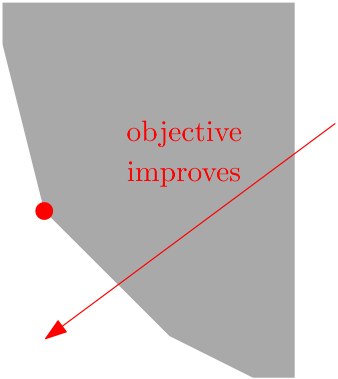
Simplex Method

- [Dantzig, 1946]
- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex



Simplex Method

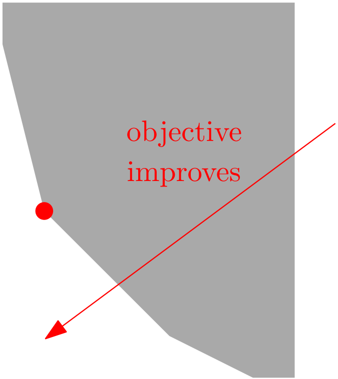
- [Dantzig, 1946]
- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex



Simplex Method

- [Dantzig, 1946]

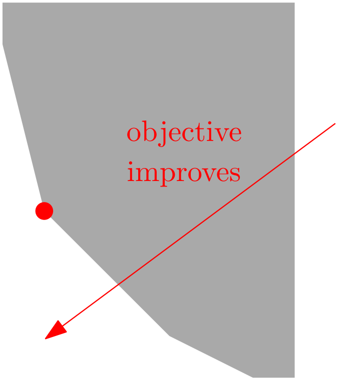
- move from one vertex to another, so as to improve the objective
- repeat until we reach an optimum vertex



- the number of iterations might be expoentially large; but algorithm runs fast in practice

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 expoentially large; but algorithm runs fast in practice
- [Spielman-Teng,2002]: smoothed analysis

Interior Point Method

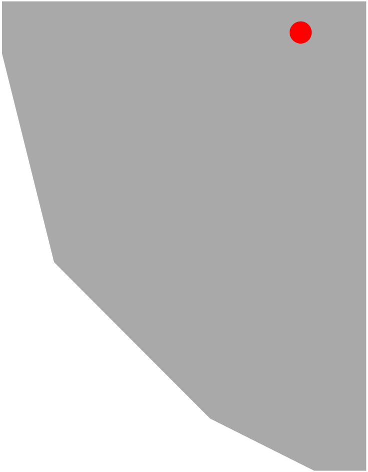
- [Karmarkar, 1984]

- 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

Interior Point Method

- [Karmarkar, 1984]

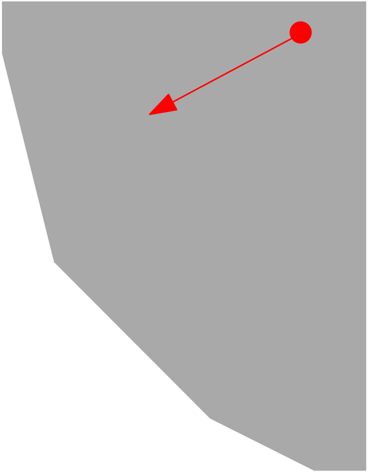
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



Interior Point Method

- [Karmarkar, 1984]

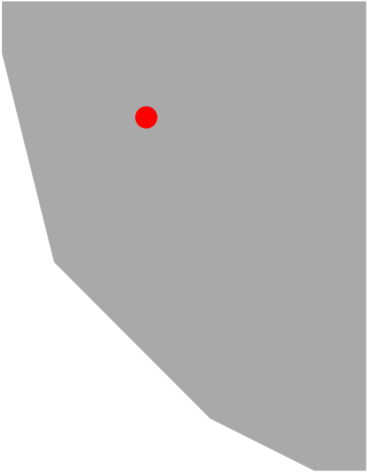
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



Interior Point Method

- [Karmarkar, 1984]

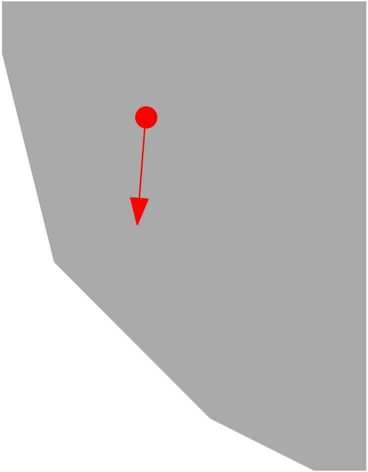
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



Interior Point Method

- [Karmarkar, 1984]

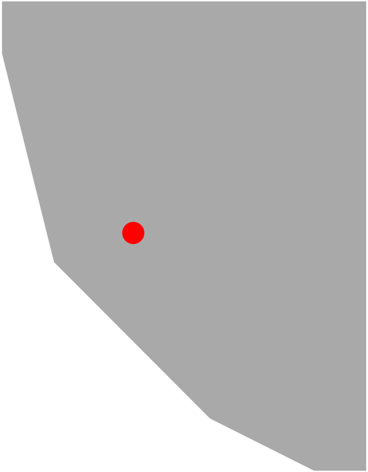
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



Interior Point Method

- [Karmarkar, 1984]

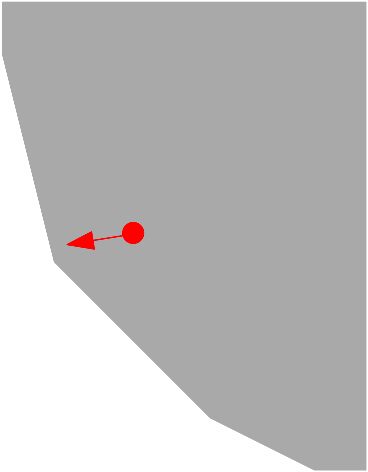
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



Interior Point Method

- [Karmarkar, 1984]

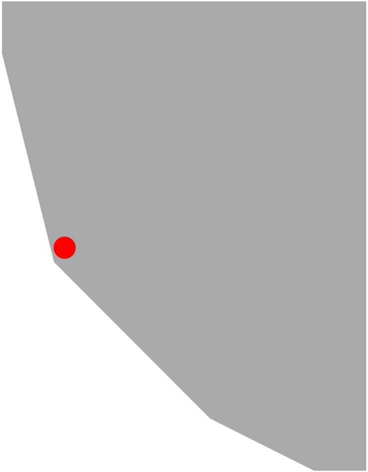
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



Interior Point Method

- [Karmarkar, 1984]

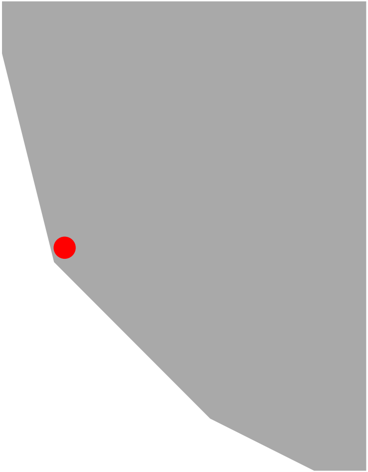
- keep the solution inside the polytope
- design penalty function so that the solution is not too close to the boundary
- the final solution will be arbitrarily close to the optimum solution



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]

Ellipsoid Method

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

Ellipsoid Method

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

- maintain an ellipsoid that contains the feasible region

Ellipsoid Method

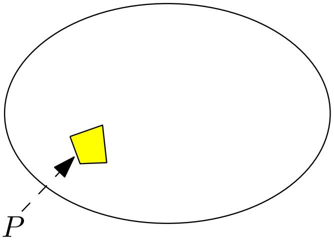
- [Khachiyan, 1979]
- used to decide if the feasible region is empty or not

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

Ellipsoid Method

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

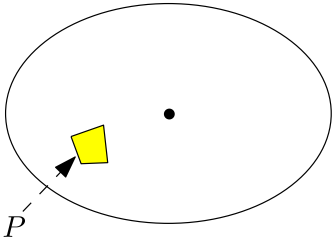
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

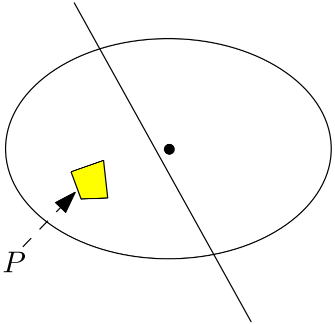
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

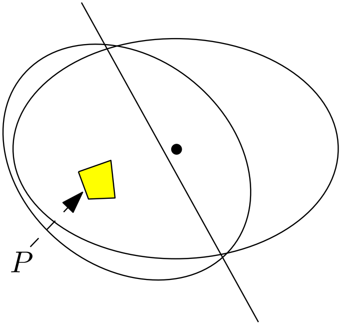
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

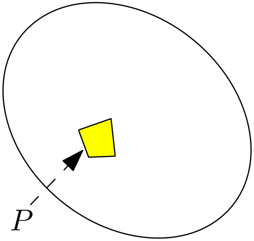
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

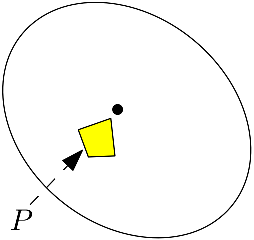
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

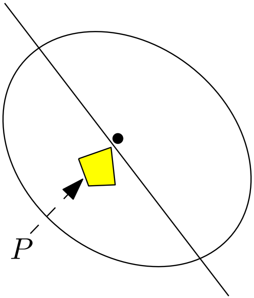
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

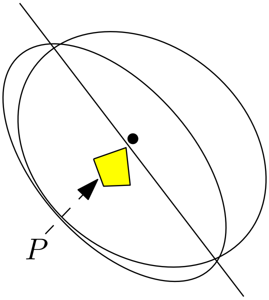
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

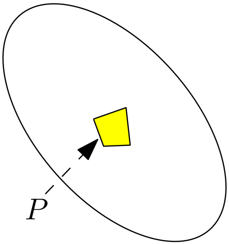
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

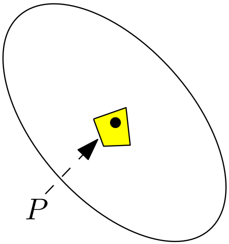
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



Ellipsoid Method

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

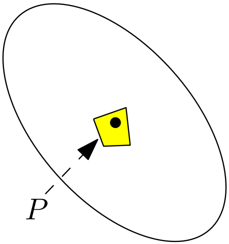
- maintain an ellipsoid that contains the feasible region
- query a separation oracle if the center of ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



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 ellipsid is in the feasible region:
  - yes: then the feasible region is not empty
  - no: cut the elliposid in half, find smaller ellipsoid to enclose the half-ellipsoid, and repeat



- polynomial time, but impractical

Q: The exact running time of these algorithms?

Q: The exact running time of these algorithms?

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

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

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

Outline

$$
\begin{aligned}
\min \qquad 7x_1 + 4x_2 & \\
x_1 + x_2 &\geq 5 \\
x_1 + 2x_2 &\geq 6 \\
4x_1 + x_2 &\geq 8 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

- optimum point: $x_1 = 1, x_2 = 4$
- value $= 7 \times 1 + 4 \times 4 = 23$

Q: How can we prove a lower bound for the value?

$$\begin{aligned} \min \quad & 7x_1 + 4x_2 \\ x_1 + x_2 &\geq 5 \\ x_1 + 2x_2 &\geq 6 \\ 4x_1 + x_2 &\geq 8 \\ x_1, x_2 &\geq 0 \end{aligned}$$

- optimum point: $x_1 = 1, x_2 = 4$
- value $= 7 \times 1 + 4 \times 4 = 23$

Q: How can we prove a lower bound for the value?

- $7x_1 + 4x_2 \geq 2(x_1 + x_2) + (x_1 + 2x_2) \geq 2 \times 5 + 6 = 16$
- $7x_1 + 4x_2 \geq (x_1 + 2x_2) + 1.5(4x_1 + x_2) \geq 6 + 1.5 \times 8 = 18$
- $7x_1 + 4x_2 \geq (x_1 + x_2) + (x_1 + 2x_2) + (4x_1 + x_2) \geq 5 + 6 + 8 = 19$
- $7x_1 + 4x_2 \geq 4(x_1 + x_2) \geq 4 \times 5 = 20$
- $7x_1 + 4x_2 \geq 3(x_1 + x_2) + (4x_1 + x_2) \geq 3 \times 5 + 8 = 23$

Primal LP

$$
\begin{aligned}
\min \quad & 7x_1 + 4x_2 \\
& x_1 + x_2 \geq 5 \\
& x_1 + 2x_2 \geq 6 \\
& 4x_1 + x_2 \geq 8 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Primal LP

$$\begin{aligned} \min \quad & 7x_1 + 4x_2 \\ & x_1 + x_2 \geq 5 \\ & x_1 + 2x_2 \geq 6 \\ & 4x_1 + x_2 \geq 8 \\ & x_1, x_2 \geq 0 \end{aligned}$$

A way to prove lower bound on the value of primal LP

$$\begin{aligned} & 7x_1 + 4x_2 && (\text{if } 7 \geq y_1 + y_2 + 4y_3 \text{ and } 4 \geq y_1 + 2y_2 + y_3) \\ \geq\ & y_1(x_1 + x_2) + y_2(x_1 + 2x_2) + y_3(4x_1 + x_2) && (\text{if } y_1, y_2, y_3 \geq 0) \\ \geq\ & 5y_1 + 6y_2 + 8y_3. \end{aligned}$$

- Goal: need to maximize $5y_1 + 6y_2 + 8y_3$

Primal LP

$$\begin{aligned} \min \quad & 7x_1 + 4x_2 \\ & x_1 + x_2 \geq 5 \\ & x_1 + 2x_2 \geq 6 \\ & 4x_1 + x_2 \geq 8 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Dual LP

$$\begin{aligned} \max \quad & 5y_1 + 6y_2 + 8y_3 \quad \text{s.t.} \\ & y_1 + y_2 + 4y_3 \leq 7 \\ & y_1 + 2y_2 + y_3 \leq 4 \\ & y_1, y_2 \geq 0 \end{aligned}$$

A way to prove lower bound on the value of primal LP

$$\begin{aligned} & 7x_1 + 4x_2 && (\text{if } 7 \geq y_1 + y_2 + 4y_3 \text{ and } 4 \geq y_1 + 2y_2 + y_3) \\ & \geq y_1(x_1 + x_2) + y_2(x_1 + 2x_2) + y_3(4x_1 + x_2) && (\text{if } y_1, y_2, y_3 \geq 0) \\ & \geq 5y_1 + 6y_2 + 8y_3. \end{aligned}$$

- Goal: need to maximize $5y_1 + 6y_2 + 8y_3$

Primal LP

$$\min \quad 7x_1 + 4x_2$$
$$x_1 + x_2 \geq 5$$
$$x_1 + 2x_2 \geq 6$$
$$4x_1 + x_2 \geq 8$$
$$x_1, x_2 \geq 0$$

Dual LP

$$\max \quad 5y_1 + 6y_2 + 8y_3 \quad \text{s.t.}$$
$$y_1 + y_2 + 4y_3 \leq 7$$
$$y_1 + 2y_2 + y_3 \leq 4$$
$$y_1, y_2 \geq 0$$

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 4 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 5 \\ 6 \\ 8 \end{pmatrix} \quad c = \begin{pmatrix} 7 \\ 4 \end{pmatrix}$$

$$\min \quad c^T x \quad \text{s.t.}$$
$$Ax \geq b$$
$$x \geq 0$$

$$\max \quad b^T y \quad \text{s.t.}$$
$$A^T y \leq c$$
$$y \geq 0$$

Primal LP

$$\min \quad c^T x \qquad \text{s.t.}$$
$$Ax \geq b$$
$$x \geq 0$$

Dual LP

$$\max \quad b^T y \qquad \text{s.t.}$$
$$A^T y \leq c$$
$$y \geq 0$$

- $P =$ value of primal LP
- $D =$ value of dual LP

Theorem (weak duality theorem) $D \leq P$.

Theorem (strong duality theorem) $D = P$.

- Can always prove the optimality of the primal solution, by adding up primal constraints.

Proof of Strong Duality Theorem

Lemma (Variant of Farkas Lemma) $Ax \leq b, x \geq 0$ is infeasible, if and only if $y^{\mathrm{T}}A \geq 0, y^{\mathrm{T}}b < 0, y \geq 0$ is feasible.

Proof of Strong Duality Theorem

Lemma (Variant of Farkas Lemma) $Ax \leq b, x \geq 0$ is infeasible, if and only if $y^{\mathrm{T}} A \geq 0, y^{\mathrm{T}} b < 0, y \geq 0$ is feasible.

- $\forall \epsilon > 0$, $\begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} x \leq \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix}, x \geq 0$ is infeasible

Proof of Strong Duality Theorem

Lemma (Variant of Farkas Lemma) $Ax \leq b, x \geq 0$ is infeasible, if and only if $y^{\mathrm{T}} A \geq 0, y^{\mathrm{T}} b < 0, y \geq 0$ is feasible.

- $\forall \epsilon > 0$, $\begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} x \leq \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix}, x \geq 0$ is infeasible
- There exists $y \in \mathbb{R}^m_{\geq 0}, \alpha \geq 0$, such that $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} \geq 0$, $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix} < 0$

Proof of Strong Duality Theorem

Lemma (Variant of Farkas Lemma) $Ax \leq b, x \geq 0$ is infeasible, if and only if $y^{\mathrm{T}}A \geq 0, y^{\mathrm{T}}b < 0, y \geq 0$ is feasible.

- $\forall \epsilon > 0, \begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} x \leq \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix}, x \geq 0$ is infeasible
- There exists $y \in \mathbb{R}^m_{\geq 0}, \alpha \geq 0$, such that $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} \geq 0$,

  $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix} < 0$
- we can prove $\alpha > 0$; assume $\alpha = 1$

Proof of Strong Duality Theorem

Lemma (Variant of Farkas Lemma) $Ax \leq b, x \geq 0$ is infeasible, if and only if $y^{\mathrm{T}}A \geq 0, y^{\mathrm{T}}b < 0, y \geq 0$ is feasible.

- $\forall \epsilon > 0, \begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} x \leq \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix}, x \geq 0$ is infeasible
- There exists $y \in \mathbb{R}^m_{\geq 0}, \alpha \geq 0$, such that $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} \geq 0$, $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix} < 0$
- we can prove $\alpha > 0$; assume $\alpha = 1$
- $-y^{\mathrm{T}}A + c^{\mathrm{T}} \geq 0, -y^{\mathrm{T}}b + P - \epsilon < 0 \iff A^{\mathrm{T}}y \leq c, b^{\mathrm{T}}y > P - \epsilon$

Proof of Strong Duality Theorem

Lemma (Variant of Farkas Lemma) $Ax \leq b, x \geq 0$ is infeasible, if and only if $y^{\mathrm{T}}A \geq 0, y^{\mathrm{T}}b < 0, y \geq 0$ is feasible.

- $\forall \epsilon > 0$, $\begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} x \leq \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix}, x \geq 0$ is infeasible
- There exists $y \in \mathbb{R}^m_{\geq 0}, \alpha \geq 0$, such that $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -A \\ c^{\mathrm{T}} \end{pmatrix} \geq 0$, $(y^{\mathrm{T}}, \alpha) \begin{pmatrix} -b \\ P - \epsilon \end{pmatrix} < 0$
- we can prove $\alpha > 0$; assume $\alpha = 1$
- $-y^{\mathrm{T}}A + c^{\mathrm{T}} \geq 0, -y^{\mathrm{T}}b + P - \epsilon < 0 \iff A^{\mathrm{T}}y \leq c, b^{\mathrm{T}}y > P - \epsilon$
- $\forall \epsilon > 0, D > P - \epsilon \quad \Longrightarrow \quad D = P$ (since $D \leq P$) $\square$

Example

Primal LP

$$\min \quad 5x_1 + 6x_2 + x_3 \quad \text{s.t.}$$

$$\begin{aligned} 2x_1 + 5x_2 - 3x_3 &\geq 2 \\ 3x_1 - 2x_2 + x_3 &\geq 5 \\ x_1 + 2x_2 + 3x_3 &\geq 7 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Dual LP

$$\max \quad 2y_1 + 5y_2 + 7y_3 \quad \text{s.t.}$$

$$\begin{aligned} 2y_1 + 3y_2 + y_3 &\leq 5 \\ 5y_1 - 2y_2 + 2y_3 &\leq 6 \\ -3y_1 + y_2 + 3y_3 &\geq 1 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}$$

Primal Solution

$x_1 = 1.6, x_2 = 0.6$

$x_3 = 1.4, \text{value} = 13$

Dual Solution

$y_1 = 1, y_2 = 5/8$

$y_3 = 9/8, \text{value} = 13$

$$
\begin{aligned}
& 5x_1 + 6x_2 + x_3 \\
\geq\ & (2x_1 + 5x_2 - 3x_3) + \frac{5}{8}(3x_1 - 2x_2 + x_3) + \frac{9}{8}(x_1 + 2x_2 + 3x_3) \\
\geq\ & 2 + \frac{5}{8} \times 5 + \frac{9}{8} \times 7 \\
=\ & 13
\end{aligned}
$$

Outline

Def. A polytope $P \subseteq \mathbb{R}^n$ is said to be integral, if all vertices of P are in $\mathbb{Z}^n$.

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.

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:

$$\max / \min \quad c^{\mathrm{T}}x \quad Ax \leq b.$$

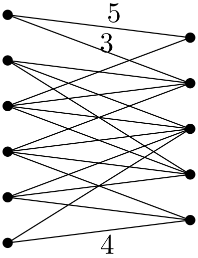
Outline

Maximum Weight Bipartite Matching

Input: bipartite graph $G = (L \uplus R, E)$

edge weights $w \in \mathbb{Z}_{>0}^E$

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

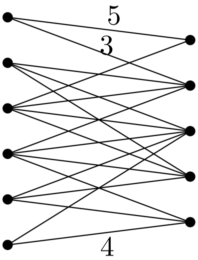


Maximum Weight Bipartite Matching

Input: bipartite graph $G = (L \uplus R, E)$

edge weights $w \in \mathbb{Z}_{>0}^E$

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



LP Relaxation

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

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

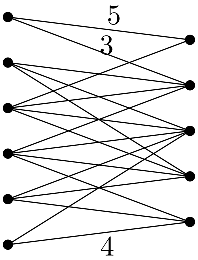
$$x_e \geq 0 \quad \forall e \in E$$

Maximum Weight Bipartite Matching

Input: bipartite graph $G = (L \uplus R, E)$

edge weights $w \in \mathbb{Z}_{>0}^{E}$

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



LP Relaxation

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

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

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

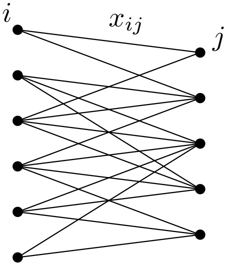
- In IP: $x_e \in \{0, 1\}$: $e \in M$?

Maximum Weight Bipartite Matching

Input: bipartite graph $G = (L \uplus R, E)$

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

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

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

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

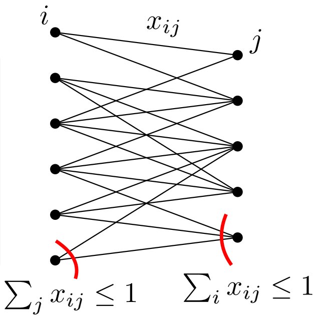
- In IP: $x_e \in \{0, 1\}$: $e \in M$?

Maximum Weight Bipartite Matching

Input: bipartite graph $G = (L \uplus R, E)$

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

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

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

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

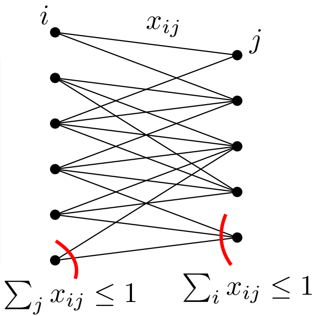
- In IP: $x_e \in \{0, 1\}$: $e \in M$?

Maximum Weight Bipartite Matching

Input: bipartite graph $G = (L \uplus R, E)$

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

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

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

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

- 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 $\{\chi^M : M$ is a matching$\}$.

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

Proof.

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

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

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

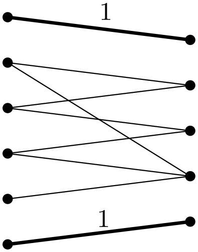
Proof.

- take x in the polytope P
- prove: x non integral $\Longrightarrow x$ non-vertex
- find $x', x'' \in P$: $x' \neq x'', x = \frac{1}{2}(x' + x'')$

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

Proof.

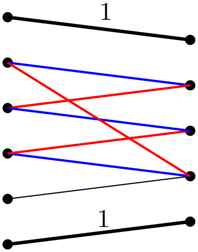
- take x in the polytope P
- prove: x non integral $\implies$ x non-vertex
- find $x', x'' \in P$: $x' \neq x'', x = \frac{1}{2}(x' + x'')$
- case 1: fractional edges contain a cycle



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

Proof.

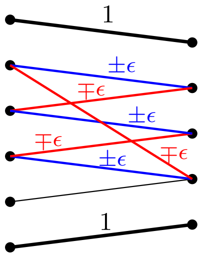
- take x in the polytope P
- prove: x non integral $\Longrightarrow$ x non-vertex
- find $x', x'' \in P$: $x' \neq x'', x = \frac{1}{2}(x' + x'')$
- case 1: fractional edges contain a cycle
  - color edges in cycle blue and red



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

Proof.

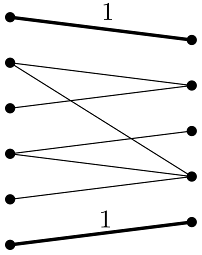
- take x in the polytope P
- prove: x non integral $\Longrightarrow$ x non-vertex
- find $x', x'' \in P$: $x' \neq x'', x = \frac{1}{2}(x' + x'')$
- case 1: fractional edges contain a cycle
  - color edges in cycle blue and red
  - x': $+\epsilon$ for blue edges, $-\epsilon$ for red edges
  - x'': $-\epsilon$ for blue edges, $+\epsilon$ for red edges



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

Proof.

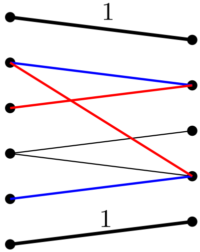
- take x in the polytope P
- prove: x non integral $\Longrightarrow$ x non-vertex
- find $x', x'' \in P$: $x' \neq x'', x = \frac{1}{2}(x' + x'')$
- case 1: fractional edges contain a cycle
  - color edges in cycle blue and red
  - x': $+\epsilon$ for blue edges, $-\epsilon$ for red edges
  - x'': $-\epsilon$ for blue edges, $+\epsilon$ for red edges
- case 2: fractional edges form a forest



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

Proof.

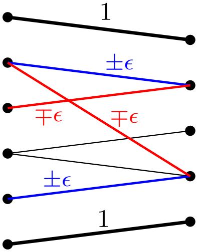
- take x in the polytope P
- prove: x non integral $\implies x$ non-vertex
- find $x', x'' \in P$: $x' \neq x'', x = \frac{1}{2}(x' + x'')$
- case 1: fractional edges contain a cycle
  - color edges in cycle blue and red
  - x': $+\epsilon$ for blue edges, $-\epsilon$ for red edges
  - x'': $-\epsilon$ for blue edges, $+\epsilon$ for red edges
- case 2: fractional edges form a forest
  - color edges in a leaf-leaf path blue and red



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 $\square$

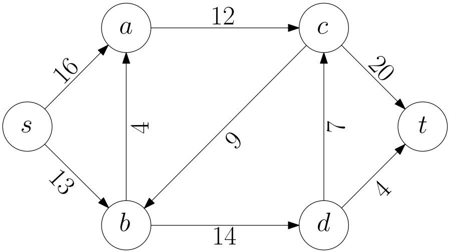


Outline

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 s-t flow is a vector $f \in \mathbb{R}^E_{\geq 0}$ satisfying the following conditions:

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

$$\sum_{e \in \delta^{\mathsf{in}}(v)} f_e = \sum_{e \in \delta^{\mathsf{out}}(v)} f_e \qquad \text{(flow conservation)}$$

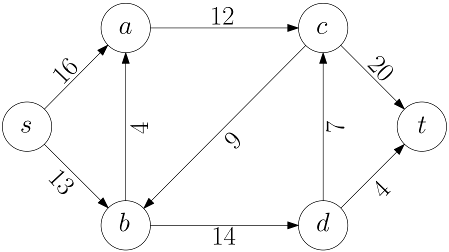
The value of flow f is defined as:

$$\mathsf{val}(f) := \sum_{e \in \delta^{\mathsf{out}}(s)} f_e = \sum_{e \in \delta^{\mathsf{in}}(t)} f_e$$

Maximum Flow Problem

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

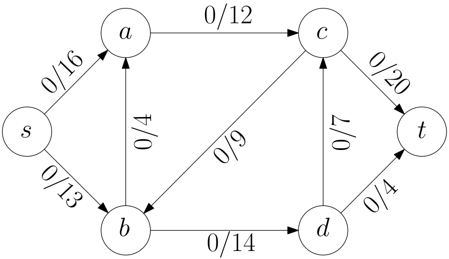
Output: maximum value of a s-t flow f



Maximum Flow Problem

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

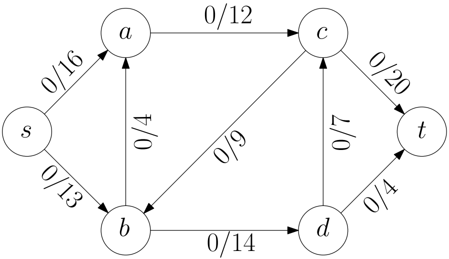
Output: maximum value of a s-t flow f



Maximum Flow Problem

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

Output: maximum value of a s-t flow f

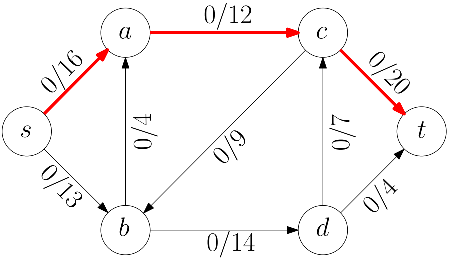


- Ford-Fulkerson method

Maximum Flow Problem

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

Output: maximum value of a s-t flow f

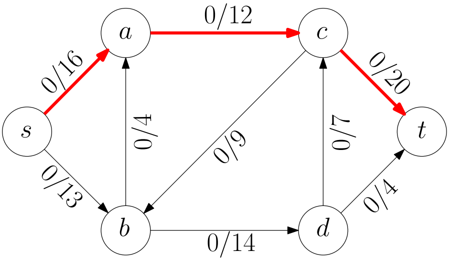


- Ford-Fulkerson method
- Maximum-Flow Min-Cut Theorem: value of the maximum flow is equal to the value of the minimum s-t cut

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

$$
\begin{aligned}
\max \quad \sum_{e \in \delta_{\mathsf{in}}(t)} x_e & \\
x_e \le c_e & \qquad \forall e \in E \\
\sum_{e \in \delta_{\mathsf{out}}(v)} x_e - \sum_{e \in \delta_{\mathsf{in}}(v)} x_e = 0 & \qquad \forall v \in V \setminus \{s,t\} \\
x_e \ge 0 & \qquad \forall e \in E
\end{aligned}
$$

LP for Maximum Flow

$$\max \sum_{e \in \delta_{\text{in}}(t)} x_e$$

$$\begin{aligned} x_e &\le c_e && \forall e \in E \\ \sum_{e \in \delta_{\text{out}}(v)} x_e - \sum_{e \in \delta_{\text{in}}(v)} x_e &= 0 && \forall v \in V \setminus \{s,t\} \\ x_e &\ge 0 && \forall e \in E \end{aligned}$$

Theorem The LP polytope is integral.

LP for Maximum Flow

$$\max \quad \sum_{e \in \delta_{\text{in}}(t)} x_e$$

$$\begin{aligned} x_e &\leq c_e && \forall e \in E \\ \sum_{e \in \delta_{\text{out}}(v)} x_e - \sum_{e \in \delta_{\text{in}}(v)} x_e &= 0 && \forall v \in V \setminus \{s, t\} \\ x_e &\geq 0 && \forall e \in E \end{aligned}$$

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 ≥ 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 cyle/path $\square$

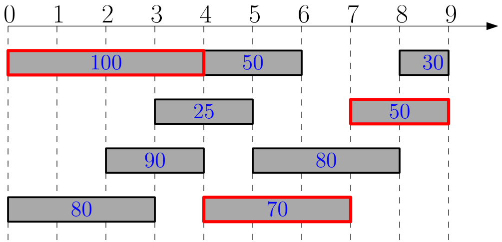
Outline

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



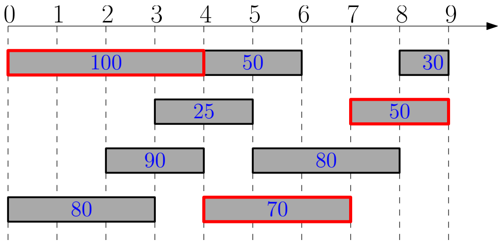
- optimum value$= 220$

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

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

Weighted Interval Scheduling Problem

Linear Program

$$\max \quad \sum_{j\in[n]} x_j w_j$$

$$\sum_{j\in[n]:t\in[s_j,f_j)} x_j \leq 1 \qquad \forall t \in [T]$$

$$x_j \geq 0 \qquad \forall j \in [n]$$

Theorem The LP polytope is integral.

Weighted Interval Scheduling Problem

Linear Program

$$\max \quad \sum_{j \in [n]} x_j w_j$$

$$\sum_{j \in [n]: t \in [s_j, f_j)} x_j \le 1 \qquad \forall t \in [T]$$

$$x_j \ge 0 \qquad \forall j \in [n]$$

Theorem The LP polytope is integral.

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

Weighted Interval Scheduling Problem

Linear Program

$$\max \quad \sum_{j \in [n]} x_j w_j$$

$$\sum_{j \in [n]: t \in [s_j, f_j)} x_j \leq 1 \qquad \forall t \in [T]$$

$$x_j \geq 0 \qquad \forall j \in [n]$$

Theorem The LP polytope is integral.

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

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

Weighted Interval Scheduling Problem

Linear Program

$$\max \quad \sum_{j\in[n]} x_j w_j$$

$$\sum_{j\in[n]:t\in[s_j,f_j)} x_j \leq 1 \qquad \forall t \in [T]$$

$$x_j \geq 0 \qquad \forall j \in [n]$$

Theorem The LP polytope is integral.

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

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

Theorem If a polytope P is defined by $Ax \geq b, x \geq 0$ 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): $\begin{pmatrix} A' & 0 \\ 0 & I \end{pmatrix} x = \begin{pmatrix} b' \\ 0 \end{pmatrix}$, where
  - 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'.

Theorem If a polytope P is defined by $Ax \geq b, x \geq 0$ 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): $\begin{pmatrix} A' & 0 \\ 0 & I \end{pmatrix} x = \begin{pmatrix} b' \\ 0 \end{pmatrix}$, where
  - 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 $x^2 = 0$.

Theorem If a polytope P is defined by $Ax \geq b, x \geq 0$ 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): $\begin{pmatrix} A' & 0 \\ 0 & I \end{pmatrix} x = \begin{pmatrix} b' \\ 0 \end{pmatrix}$, where
  - 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 □

Example for the Proof

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \geq \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

Example for the Proof

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \geq \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

The following equation system may give a vertex:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Equivalently, the vertex satisfies

$$\begin{pmatrix} a_{1,2} & a_{1,3} & 0 & 0 & 0 \\ a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_1 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

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.

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

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

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

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
- #edges $=$ #vertices $\implies$ underlying undirected graph contains a cycle $\implies$ det$(A') = 0$ □

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
- #edges $=$ #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.

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
- #edges $=$ #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.

Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \color{red}{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Example for the Proof

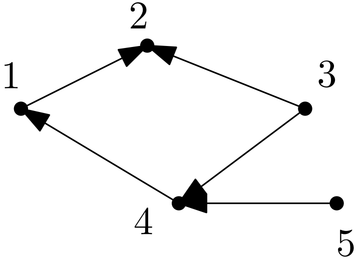
$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \color{red}{1} & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 \end{pmatrix}$$

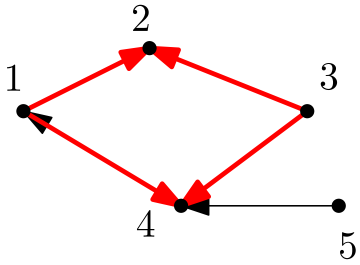
Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 \end{pmatrix}$$



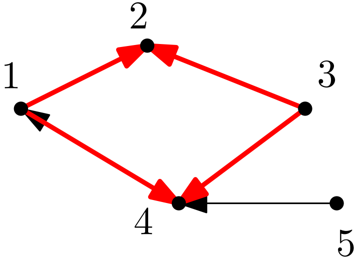
Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 \end{pmatrix}$$



Example for the Proof

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 \end{pmatrix}$$



$$\begin{array}{cc} + & \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \end{pmatrix} \\ - & \begin{pmatrix} 0 & -1 & 1 & 0 & 0 \end{pmatrix} \\ + & \begin{pmatrix} 0 & 0 & 1 & -1 & 0 \end{pmatrix} \\ - & \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \end{pmatrix} \\ \\ = & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

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

Proof.

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,

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.

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

$$M = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}. \ \det(M) = 1.$$

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

$$M = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}. \ \det(M) = 1.$$

- $\det(A'M) \in \{0, \pm 1\} \implies \det(A') \in \{0, \pm 1\}$. □

Example for the Proof

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} \color{red}{0} & \color{red}{1} & \color{red}{1} & \color{red}{1} & \color{red}{0} & 0 \\ \color{red}{1} & \color{red}{1} & \color{red}{1} & \color{red}{1} & \color{red}{0} & 0 \\ \color{red}{0} & \color{red}{0} & \color{red}{1} & \color{red}{1} & \color{red}{1} & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ \color{red}{0} & \color{red}{0} & \color{red}{0} & \color{red}{1} & \color{red}{1} & 0 \\ \color{red}{0} & \color{red}{1} & \color{red}{1} & \color{red}{1} & \color{red}{1} & 0 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Example for the Proof

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- $(\text{col } 1, \text{col } 2 - \text{col } 1, \text{col } 3 - \text{col } 2, \text{col } 4 - \text{col } 3, \text{col } 5 - \text{col } 4)$

Example for the Proof

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

- (col 1, col 2 − col 1, col 3 − col 2, col 4 − col 3, col 5 − col 4)

Example for the Proof

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

- $(\text{col } 1, \text{col } 2 - \text{col } 1, \text{col } 3 - \text{col } 2, \text{col } 4 - \text{col } 3, \text{col } 5 - \text{col } 4)$
- every row has at most one 1, at most one -1

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

Example

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: the bipartite graph

Example

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: the bipartite graph
- A': obtained from A by negating columns correspondent to R

Example

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: 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

Example

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: 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 $\square$

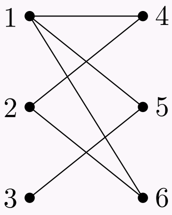
Example

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: 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 $\square$

Example

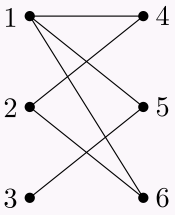


Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: 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 $\square$

Example



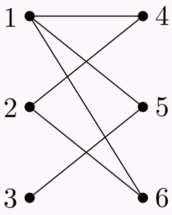
$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Lemma The edge-vertex incidence matrix A of a bipartite graph is totally-unimodular.

Proof.

- $G = (L \uplus R, E)$: 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 $\square$

Example



$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

- 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.

- 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.