COMP323 – Introduction to Computational Game Theory

Mathematical and Algorithmic Foundations of Market Equilibria

Paul G. Spirakis

Department of Computer Science
University of Liverpool

Outline

Linear programs

A linear program (LP) is the problem of optimizing (i.e., minimizing or maximizing) a linear objective function subject to linear equality or inequality constraints.

In standard form, it is expressed as:

$$\begin{array}{rll} \text{minimize} & \sum_{j=1}^{n} c_j x_j & \text{(objective function)} \\ \text{subject to:} & \sum_{j=1}^{n} a_{ij} x_j = b_i\,, \quad i = 1, \ldots, m & \text{(constraints)} \\ & x_j \geq 0\,, \qquad\quad j = 1, \ldots, n & \text{(non-negativity constraints)} \end{array}$$

where a_{ij}, b_i, c_j are given for all i, j.

Expressing LP using matrices

A linear program is expressed more conveniently using matrices:

$$\min \quad c^T x \quad \text{subject to} \quad \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

where

$$\begin{array}{rcll} x & = & (x_j)_{j\in\{1,\ldots,n\}} & \in \mathbb{R}^{n\times 1} \\ b & = & (b_i)_{i\in\{1,\ldots,m\}} & \in \mathbb{R}^{m\times 1} \\ c & = & (c_j)_{j\in\{1,\ldots,n\}} & \in \mathbb{R}^{n\times 1} \\ A & = & (a_{ij})_{i\in\{1,\ldots,m\},j\in\{1,\ldots,n\}} & \in \mathbb{R}^{m\times n} \end{array}$$

Basic terminology

- If x satisfies $Ax = b$, $x \geq 0$, then x is feasible.
- An LP is feasible if there exists a feasible solution, otherwise it is said to be infeasible.
- An optimal solution x^* is a feasible solution such that

$$c^T x^* = \min\{c^T x \ : \ Ax = b, x \geq 0\} \ .$$

- An LP is unbounded (from below) if for all $\lambda \in \mathbb{R}$, there exists a feasible x^* such that

$$c^T x^* \leq \lambda \ .$$

Equivalent forms

An LP can take several forms:

- We might be maximizing instead of minimizing;
- We might have a combination of equality and inequality constraints;
- Some variables may be restricted to be non-positive instead of non-negative, or be unrestricted in sign.

Two forms are said to be equivalent if they have the same set of optimal solutions or are both infeasible or both unbounded.

Equivalent forms

- A maximization problem can be expressed as a minimization problem:

$$\max c^T x \Leftrightarrow \min -c^T x$$

- An equality can be represented as a pair of inequalities:

$$a_i^T = b_i \Leftrightarrow \begin{cases} a_i^T x \leq b_i \\ a_i^T x \geq b_i \end{cases} \Leftrightarrow \begin{cases} a_i^T x \leq b_i \\ -a_i^T x \leq -b_i \end{cases}$$

- By adding a slack variable, an inequality can be represented as a combination of equality and non-negativity constraints:

$$a_i^T x \leq b_i \Leftrightarrow a_i^T x + s_i = b_i \, , \, s_i \geq 0$$

Equivalent forms

- Non-positivity constraints can be expressed as non-negativity constraints: To express $x_j \leq 0$, replace x_j everywhere with $-y_j$ and impose the condition $y_j \geq 0$.
- If x_j is unrestricted in sign, i.e., non-positive or non-negative, replace everywhere x_j by $x_j^+ - x_j^-$, adding the constraints $x_j^+, x_j^- \geq 0$.

In general, an inequality can be represented using a combination of equality and non-negativity constraints, and vice versa.

Canonical and standard forms

1. LP in canonical form:

$$\min\{c^T x \,:\, Ax \geq b\}$$

2. LP in standard form:

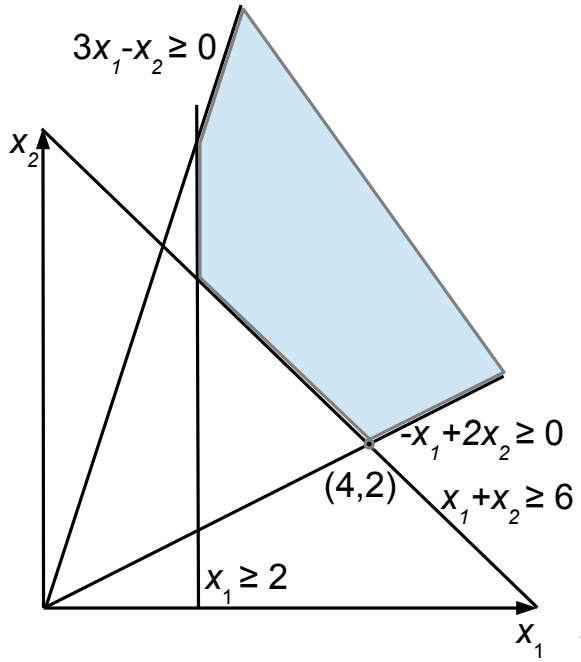
$$\min\{c^T x \,:\, Ax = b,\, x \geq 0\}$$

Using the rules described before, an LP in standard form may be written in canonical form, and vice versa.

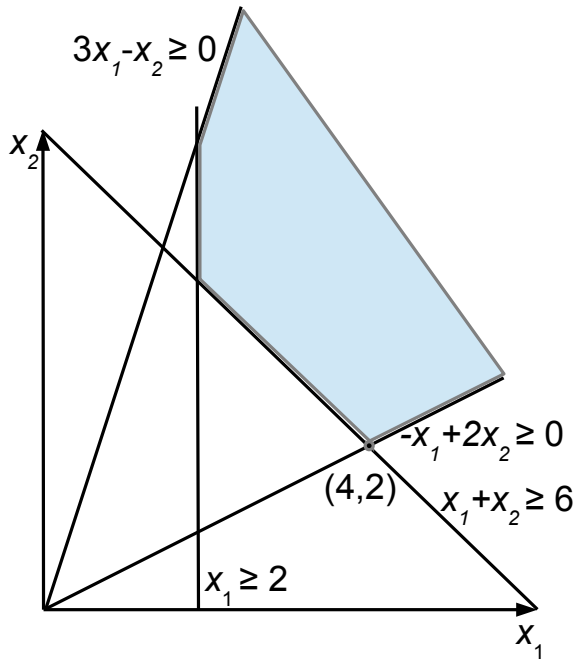
Example

The optimal solution of the following LP is $(4, 2)$ of cost 2:

$$\begin{array}{lrcl} \min & x_2 & & \\ \text{s.t.} & x_1 & \geq & 2 \\ & 3x_1 - x_2 & \geq & 0 \\ & x_1 + x_2 & \geq & 6 \\ & -x_1 + 2x_2 & \geq & 0 \end{array}$$



Example



- If we were maximizing x_2 instead of minimizing under some feasible region, the resulting LP would be unbounded.
- For any objective function for which the LP is bounded, there seems to exist an optimal solution which is a "corner" of the feasible section.

Vertices

Consider an LP in standard form:

$$\min\{c^T x \ : \ Ax = b,\ x \geq 0\}$$

Let $P = \{x \ : \ Ax = b,\ x \geq 0\} \subseteq \mathbb{R}^n$.

Definition

x is a vertex of P if $\nexists y \neq 0$ such that $x + y \in P$ and $x - y \in P$.

Theorem

Assume $\min\{c^T x : x \in P\}$ *is finite. Then, for all* $x \in P$, *there exists a vertex* $x' \in P$ *such that*

$$c^T x' \leq c^T x \ .$$

Vertices

Proof.

- Recall that $P = \{x \ : \ Ax = b, x \geq 0\} \subseteq \mathbb{R}^n$.
- If x is a vertex then take $x' = x$.
- If x is not a vertex then, by definition, there exists $y \neq 0$ such that $x + y \in P$ and $x - y \in P$.
- Then $A(x + y) = b$ and $A(x - y) = b$, implying that $Ay = 0$.
- W.l.o.g. assume $c^T y \leq 0$ (take either y or $-y$).
- If $c^T y = 0$ choose y such that $\exists j : y_j < 0$. Since $y \neq 0$ and $c^T y = c^T(-y) = 0$, this must be true for either y or $-y$.

Vertices

Proof (continued). Consider $x + \lambda y$, $\lambda > 0$. Since $c^T y \leq 0$,

$$c^T(x + \lambda y) = c^T x + \lambda c^T y \leq c^T x \ .$$

Case 1: $\exists j : y_j < 0$.

As λ increases, component j decreases until $x + \lambda y$ is no longer feasible.

- Choose the largest λ such that $x + \lambda y \geq 0$, i.e.,

$$\lambda = \min_{j:y_j<0} \left\{ \frac{x_j}{-y_j} \right\} = \frac{x_k}{-y_k} \ .$$

- Since $Ay = 0$,

$$A(x + \lambda y) = Ax + \lambda Ay = Ax = b \ .$$

- So $x + \lambda y \in P$ and $x + \lambda y$ has one more zero component than x, namely $(x + \lambda y)_k$.
- Replace x by $x + \lambda y$.

Vertices

Proof (continued).

Case 2: $y_j \geq 0$ for all j.

- By assumption, $c^T y < 0$.
- Moreover,

$$\begin{aligned} A(x + \lambda y) &= Ax + \lambda Ay = Ax = b \quad \text{and} \\ x + \lambda y &\geq x \geq 0 \ . \end{aligned}$$

- But $c^T(x + \lambda y) = c^T x + \lambda c^T y \to -\infty$ as $\lambda \to \infty$, implying LP is unbounded, a contradiction.

Case 1 can happen at most n times (the number of components of x). By induction on the number of nonzero components of x, we obtain a vertex x'. □

Vertices and optimal solutions

Corollary

If $\min\{c^T x : Ax = b, x \geq 0\}$ *is finite, there exists an optimal solution* x^* *which is a vertex.*

Proof. Suppose not. Take an optimal solution.
By the previous theorem, there exists a vertex costing no more than the optimal. This vertex must be optimal as well! □

Vertices and optimal solutions

Corollary

If $\min\{c^T x : Ax = b, x \geq 0\}$ *is finite, there exists an optimal solution* x^* *which is a vertex.*

Proof. Suppose not. Take an optimal solution.
By the previous theorem, there exists a vertex costing no more than the optimal. This vertex must be optimal as well! □

Corollary

If $P = \{x : Ax = b, x \geq 0\} \neq \emptyset$ *(i.e., if the LP is feasible), then* P *has a vertex.*

A characterization of vertices

Theorem

Let $P = \{x \ : \ Ax = b, x \geq 0\}$. For $x \in P$, let A_x be a submatrix of A corresponding to j such that $x_j > 0$. Then x is a vertex iff A_x has linearly independent columns (i.e., A_x has full column rank).

Example: Let

$$A = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 7 & 3 & 2 & 1 \\ 0 & 2 & 0 & 5 \end{bmatrix}, \quad x = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Then A_x corresponds to the positive components of x, $j = 1$ and $j = 3$, i.e., to the 1st and 3rd column of A:

$$A_x = \begin{bmatrix} 2 & 3 \\ 7 & 2 \\ 0 & 0 \end{bmatrix}, \text{ and } x \text{ is a vertex.}$$

A characterization of vertices

Proof ($\Leftarrow$).

- Assume x is not a vertex. Then $\exists y \neq 0$ such that $x + y, x - y \in P$.
- Since $A(x + y) = b$ and $A(x - y) = b$, $Ay = 0$.
- Let A_y be a submatrix of A corresponding to non-zero components of y. Since $y \neq 0$ and $Ay = 0$, A_y has dependent columns.
- $x + y \geq 0$ and $x - y \geq 0$ imply that $y_j = 0$ whenever $x_j = 0$.
- Therefore A_y is a submatrix of A_x.
- Therefore A_x has linearly dependent columns.

A characterization of vertices

Proof ($\Rightarrow$).

- Suppose A_x has linearly dependent columns. Then $\exists y \neq 0$ such that $A_x y = 0$.
- Extend y to $\mathbb{R}^n$ by adding zero components.
- Then $\exists y \in \mathbb{R}^n$ such that $Ay = 0$, $y \neq 0$, and $y_j = 0$ whenever $x_j = 0$.
- Choose $\lambda \geq 0$ such that $x + \lambda y, x - \lambda y \geq 0$ and $A(x + \lambda y) = A(x - \lambda y) = Ax = b$.
- So $x + \lambda y \in P$ and $x - \lambda y \in P$.
- Hence x is not a vertex.

□

Basic feasible solutions

Definition (Basic feasible solution)

x is a vertex of $P = \{x : Ax = b, x \geq 0\}$ iff there is a basis $B \subseteq \{1, \ldots, n\}$ such that $|B| = m$ and

1. $x_N = 0$ for $N = \{1, \ldots, n\} - B$
2. A_B is non-singular
3. $x_B = A_B^{-1} b \geq 0$.

and x is a basic feasible solution.

- A vertex can have several basic feasible solutions corresponding to it (by augmenting $\{j : x_j > 0\}$).
- A basis may not lead to any basic feasible solution since $A_b^{-1} b$ is not necessarily nonnegative.

Basic feasible solutions

Example:

$$\begin{aligned} x_1 + x_2 + x_3 &= 5 \\ 2x_1 - x_2 + 2x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

We can select as a basis $B = \{1, 2\}$. Thus, $N = \{3\}$ and

$$A_B = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}$$

$$A_B^{-1} = \begin{bmatrix} 1/3 & 1/3 \\ 2/3 & -1/3 \end{bmatrix}$$

$$A_B^{-1} b = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$x = \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix} .$$

The Simplex method

The Simplex algorithm (Dantzig, 1947) solves LP problems by focusing on basic feasible solutions (bfs).

- The basic idea is to start from some vertex v and look at the adjacent vertices.
- If an improvement in cost is possible by moving to one of the adjacent vertices, then we do so.
- Thus, we will start with a bfs corresponding to a basis B and, at each iteration, try to improve the cost of the solution by removing one variable from the basis and replacing it by another.

The Simplex method

We first rewrite our LP in the form:

$$\begin{aligned} \min \quad & c_B x_B + c_N x_N \\ \text{s.t.} \quad & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{aligned}$$

- Here B is the basis corresponding to the bfs we are staring from.
- For any solution x, $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$ and its cost $c^T x$ is

$$\begin{aligned} c^T x &= c_B x_B + c_N x_N \\ &= c_B (A_B^{-1} b - A_B^{-1} A_N x_N) + c_N x_N \\ &= c_B A_B^{-1} b - (c_N - c_B A_B^{-1} A_N) x_N \ . \end{aligned}$$

The Simplex method

- The cost of a solution x is $c^T x = c_B A_B^{-1} b - (c_N - c_B A_B^{-1} A_N) x_N$.
- We denote the reduced cost of the non-basic variables by $\tilde{c}_N$:

$$\tilde{c}_N = c_N - c_B A_B^{-1} A_N \ ,$$

  i.e., the quantity which is the coefficient of x_N.
- If there is a $j \in N$ such that $\tilde{c}_j < 0$, then by increasing x_j (up from zero) we will decrease the value of the objective function.
- x_B depends on x_N, and we can increase x_j only as long as all the components of x_B remain positive.

The Simplex method

- So, in a step of the Simplex method, we find a $j \in N$ such that $\tilde{c}_j < 0$, and increase it as much as possible while keeping $x_B \geq 0$.
- It is not possible any more to increase x_j when one of the components of x_B is zero.
- What happened is that a non-basic variable is now positive and we include it in the basis, and one variable which was basic is now zero, so we remove it from the basis.
- If there is no $j \in N$ such that $\tilde{c}_j < 0$, then we stop, and the current bfs is optimal. This follows from the expression for $c^T x$ since x_N is nonnegative.

The Simplex method

Remark.

Some of the basic variables may be zero to begin with, and we may not be able to increase x_j at all.

- We can replace j by k in the basis, but without moving from the vertex corresponding to the basis.
- In the next step we might replace k by j, and be stuck in a loop.
- Thus, we need to specify a pivoting rule to determine which index should enter the basis and which should be removed.
- There is a pivoting rule which can not lead to infinite loops: choose the minimal j and k possible.
- There is no known pivoting rule for which the number of pivots in the worst case is better than exponential.

The Simplex method

Complexity of the Simplex algorithm:

What is the length of the shortest path between two vertices of a convex polyhedron, where the path is along edges, and the length of the path is measured in terms of the number of vertices visited?

Randomized pivoting: randomly permute the index columns of A and apply the Simplex method, always choosing the smallest j possible.

$\Rightarrow$ It is possible to show a subexponential bound on the expected number of pivots.

- The question of the existence of a polynomial pivoting scheme is open.
- We will outline later a different algorithm which **is** polynomial: that algorithm will not move from one vertex to another, but will focus on interior points of the feasible domain.

When is a linear program feasible?

When is a linear program of the form $Ax = b, x \geq 0$ feasible?
(We ignore the objective function since it has no effect on the feasibility.)
Example. Consider the system of equations

$$\begin{array}{rcrcrcl} x_1 & + & x_2 & + & x_3 & = & 6 \\ 2x_1 & + & 3x_2 & + & x_3 & = & 8 \\ 2x_1 & + & x_2 & + & 3x_3 & = & 0 \end{array}$$

The linear combination

$$\begin{array}{rcrcrcrcl} -4 & \cdot & (x_1 & + & x_2 & + & x_3) & = & -4 \cdot 6 \\ 1 & \cdot & (2x_1 & + & 3x_2 & + & x_3) & = & 1 \cdot 8 \\ 1 & \cdot & (2x_1 & + & x_2 & + & 3x_3) & = & 1 \cdot 0 \end{array}$$

results in $0x_1 + 0x_2 + 0x_3 = -16$, which means that the system has no feasible solution.

When is a linear program feasible?

An elementary theorem of linear algebra:

Theorem

Exactly one of the following is true for the system $Ax = b$:

1. *There is x such that $Ax = b$.*
2. *There is y such that $A^T y = b$ but $y^T b = 1$.*

I.e., if a system has no solution, there is always a vector y (as in our example $y = [-4, 1, 1]^T$) which proves that the system has no solution.
Note: This is no enough for our purposes:

- A system $Ax = b$ can be feasible, but still have no non-negative solutions $x \geq 0$.
- Farkas' lemma establishes the equivalent results for our system $Ax = b, x \geq 0$.

Farkas' lemma

Farkas' lemma

Exactly one of the following is true for the system $Ax = b, x \geq 0$:

1. There is x such that $Ax = b, x \geq 0$.
2. There is y such that $A^T y \geq 0$ but $y^T b < 0$.

Proof. We will first show that the two conditions can not happen together. Suppose we do have both x and y as in the statement of Farkas' lemma. This implies:

$$\begin{aligned} Ax &= b \\ y^T Ax &= y^T b \\ (y^T Ax)^T &= (y^T b)^T \\ x^T A^T y &= y^T b < 0 \ . \end{aligned}$$

But since $x \geq 0$ and $A^T y \geq 0$, $x^T A^T y \geq 0$, a contradiction.

Farkas' lemma

Proof (continued). To show that at least one of the two conditions must happen, we will use the Projection Theorem:

Projection theorem

Let K be a closed convex non-empty set in $\mathbb{R}^n$, and let b be any point in $\mathbb{R}^n$. The projection of b onto K is a point $p \in K$ that minimizes the Euclidean distance $\|b - p\|$. Then p has the property that, for all $z \in K$,

$$(z - p)^T (b - p) \leq 0 \ .$$

Farkas' lemma

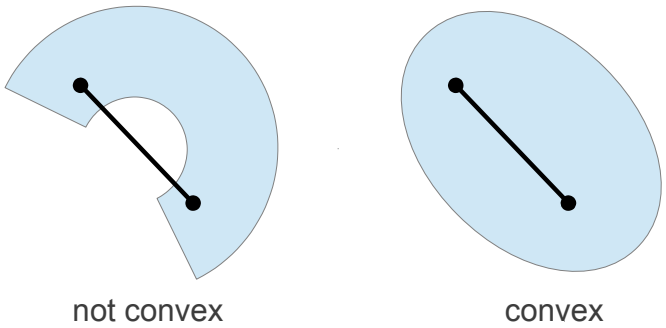
Proof (continued). To show that at least one of the two conditions must happen, we will use the Projection Theorem:

Projection theorem

Let K be a closed convex non-empty set in $\mathbb{R}^n$, and let b be any point in $\mathbb{R}^n$. The projection of b onto K is a point $p \in K$ that minimizes the Euclidean distance $\|b - p\|$. Then p has the property that, for all $z \in K$,

$$(z - p)^T (b - p) \leq 0 \ .$$

Illustration: convex and non-convex sets in $\mathbb{R}^2$



Farkas' lemma

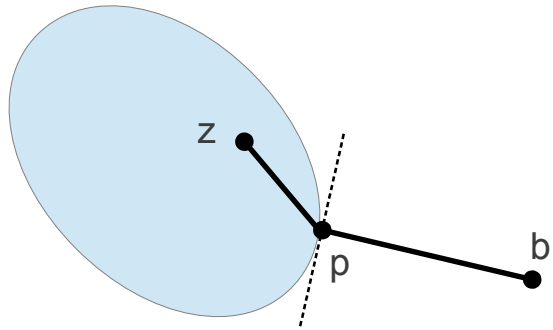
Proof (continued). To show that at least one of the two conditions must happen, we will use the Projection Theorem:

Projection theorem

Let K be a closed convex non-empty set in $\mathbb{R}^n$, and let b be any point in $\mathbb{R}^n$. The projection of b onto K is a point $p \in K$ that minimizes the Euclidean distance $\|b - p\|$. Then p has the property that, for all $z \in K$,

$$(z - p)^T (b - p) \leq 0 \ .$$

Illustration: the Projection Theorem



Farkas' lemma

Proof (continued). Assume that there is no x such that $Ax = b$, $x \geq 0$. We will show that there is y such that $A^T y \geq 0$ but $y^T b < 0$.

- Let $K = \{Ax : x \geq 0\} \subseteq \mathbb{R}^m$ (A is an $m \times n$ matrix). K is a cone in $\mathbb{R}^m$, non-empty, convex, and closed.
- Since $Ax = b, x \geq 0$ has no solution, $b \notin K$. Let p be the projection of b onto K.
- Since $p \in K$, there is a $w \geq 0$ such that $Aw = p$.
- According to the Projection Theorem, for all $z \in K$, $(z - p)^T(b - p) \leq 0$. That is, for all $x \geq 0$, $(Ax - p)^T(b - p) \leq 0$.
- Define $y = p - b$, which implies $(Ax - p)^T y \geq 0$.
- Since $Aw = p$, $(Ax - Aw)^T y \geq 0$ or $(x - w)^T(A^T y) \geq 0$ for all $x \geq 0$ (remember that w was fixed by choosing b).

Farkas' lemma

Proof (continued).

- Set $x = w + [0\ 0\ \cdots\ 1\ \cdots\ 0]^T$ (w plus a unit vector with a 1 in the ith row). Note that x is non-negative, because $w \geq 0$.
- This will extract the ith column of A, so we conclude that the ith component of $A^T y$ is non-negative, and since this is true for all i, $A^T y \geq 0$.
- It remains to show that $y^T b < 0$. We have:

$$y^T b = b^T y = (p - y)^T y = p^T y - y^T y \ .$$

- Since $(Ax - p)^T y \geq 0$ for all $x \geq 0$, taking $x = 0$ gives $p^T y \leq 0$.
- Since $b \notin K$, $y = p - b \neq 0$, so $y^T y > 0$.
- So $y^T b = p^T y - y^T y < 0$. □

Farkas' lemma

Canonical form

Using a very similar proof we can show the same for the canonical form:

Farkas' lemma, canonical form

Exactly one of the following is true for the system $Ax \leq b$:

1. There is x such that $Ax \leq b$.
2. There is $y \geq 0$ such that $A^T y = 0$ but $y^T b < 0$.

The concept of duality

- Duality is the most important concept in LP.
- It allows to provide a proof of optimality.
- It is important algorithmically and also leads to beautiful combinatorial statements.

Example: The statement

In a graph, the smallest number of edges in a path between two specific vertices s and t is equal to the maximum number of $s-t$ cuts (i.e., subsets of edges whose removal disconnects s and t).

is a direct sequence of LP duality.

Motivation of duality

Duality can be motivated by the problem of trying to find lower bounds on the value of the optimal solution to an LP problem:

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

How can we obtain the best possible upper bound on the objective function?

- By multiplying each $A_i x = b_i$ by some y_i and summing up, we obtain $y^T Ax = b^T y$.
- We impose that the coefficient of x_j is at most c_j.
- Then $b^T y$ must be a lower bound on the optimal value since x_j is constrained to be nonnegative.

Motivation of duality

To get the best possible lower bound, we wan to solve the following problem:

$$\begin{aligned} \max \quad & b^T y \\ \text{s.t.} \quad & A^T y \leq c \end{aligned}$$

This is another LP!

- This is the dual LP of the original one.
- The original LP is called the primal.
- Solving the dual will give us a lower bound on the optimum value of the primal.
- Weak duality says precisely this:

$$\max b^T y \leq \min c^T x \ .$$

- We will use Farkas' lemma to prove strong duality: these two quantities are equal.

Duality

Example

$$\begin{aligned} \min \quad & x_1 + 2x_2 + 4x_3 \\ \text{s.t.} \quad & x_1 + x_2 + 2x_3 = 5 \\ & 2x_1 + x_2 + 3x_3 = 8 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Let $z = \min(x_1 + 2x_2 + 4x_3)$ be the optimum value of this LP.

- The first equality gives a lower bound of 5 on z, since

$$x_1 + 2x_2 + 4x_3 \geq x_1 + x_2 + 2x_3 = 5 \ .$$

- We can get an even better lower bound by taking 3 times the first equality minus the second one:

$$x_1 + 2x_2 + 3x_3 = 7 \leq x_1 + 2x_2 + 4x_3 \ .$$

- For $x = [3\ 2\ 0]^T$ the objective function is 7, implying optimality.

Duality

Example

The mechanism of generating lower bounds is formalized by the dual LP:

Primal

$$\begin{array}{lrcl} \min & x_1 + 2x_2 + 4x_3 & & \\ \text{s.t.} & x_1 + x_2 + 2x_3 & = & 5 \\ & 2x_1 + x_2 + 3x_3 & = & 8 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Dual

$$\begin{array}{lrcl} \max & 5y_1 + 8y_2 & & \\ \text{s.t.} & y_1 + 2y_2 & \leq & 1 \\ & y_1 + y_2 & \leq & 2 \\ & 2y_1 + 3y_2 & \leq & 4 \end{array}$$

- y_1 represents the multiplier for the first constraint.
- y_2 represents the multiplier for the second constraint.
- The dual LP's objective function also achieves a maximum value of 7 at $y = [3 \quad -1]^T$.

Duality

Formalization

Formally:

Let P and D be the following pair of linear programs:

$$(P) \quad z = \min\{c^T x \ : \ Ax = b,\ x \geq 0\}$$
$$(D) \quad w = \max\{b^T y \ : \ A^T y \leq c\} \ .$$

P is called the primal and D the dual linear program.

Duality

Formalization

Formally:
Let P and D be the following pair of linear programs:

$$(P) \quad z = \min\{c^T x \ : \ Ax = b,\ x \geq 0\}$$
$$(D) \quad w = \max\{b^T y \ : \ A^T y \leq c\} \ .$$

P is called the primal and D the dual linear program.

Claim

The dual of the dual is the primal.

- I.e., if we formulate D as an LP in standard form (same as P), its dual can be seen to be equivalent to the original primal P.
- Thus, in any statement, we may replace the roles of primal and dual without affecting the statement.

Duality

The dual of the dual is the primal

Claim

The dual of the dual is the primal.

Proof.

- The dual D is equivalent to

$$\min\{-b^T y \ : \ A^T y + Is = c, s \geq 0\} \ .$$

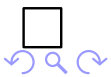
- Changing forms we get

$$\min\{-b^T y^+ + b^T y^- \ : \ A^T y^+ - A^T y^- + Is = c, \ y^+, y^-, s \geq 0\} \ .$$

- Taking the dual of this we obtain:

$$\max\{-c^T x \ : \ A(-x) \leq -b, -A(-x) \leq b, I(-x) \leq 0\} \ .$$

- This is the same as the primal P.



Weak duality

Lemma (Weak duality)

For any primal-dual pair of LPs

$$(P) \quad z = \min\{c^T x \ : \ Ax = b,\, x \geq 0\}$$
$$(D) \quad w = \max\{b^T y \ : \ A^T y \leq c\} \ ,$$

it holds that $z \geq w$.

Proof. Suppose x is primal feasible and y is dual feasible. Then

$$c^T x \geq y^T A x = y^T b \ .$$

Thus

$$\begin{aligned} z &= \min\{c^T x \ : \ Ax = b, x \geq 0\} \\ &\geq \max\{b^T y \ : \ A^T y \leq c\} = w \ . \end{aligned}$$

Strong duality

From weak duality we conclude that the following cases are not possible:

1. P is feasible and unbounded and D is feasible.
2. P is feasible and D is feasible and unbounded.

However, both the primal and the dual might be infeasible.

Strong duality

From weak duality we conclude that the following cases are not possible:

1. P is feasible and unbounded and D is feasible.
2. P is feasible and D is feasible and unbounded.

However, both the primal and the dual might be infeasible.

Stronger version of weak duality:

Theorem (Strong duality)

If P or D is feasible then $z = w$.

Proof of strong duality

Proof. To prove strong duality, we recall the following corollary of Farkas' lemma:

Farkas' lemma, canonical form

Exactly one of the following is true for the system $A'x' \leq b'$:

1. There is x' such that $A'x' \leq b'$.
2. There is $y' \geq 0$ such that $(A')^T y' = 0$ and $(b')^T y' < 0$.

We only need to show that $z \leq w$.

- W.l.o.g. (by duality) assume P is feasible.
- If P is unbounded, then by weak duality we have $z = w = -\infty$.

Proof of strong duality

Proof (continued).

- If P is bounded, let x^* be an optimal solution, i.e., $Ax^* = b$, $x^* \geq 0$, and $c^T x^* = z$.
- We claim that $\exists y$ such that $A^T y \leq c$ and $b^T y \geq z$. If so we are done.
- Suppose no such y exists.
- By Farkas' lemma, with

$$A' = \begin{bmatrix} A^T \\ -b^T \end{bmatrix}, \quad b' = \begin{bmatrix} c \\ -z \end{bmatrix}, \quad x' = y, \quad y' = \begin{bmatrix} x \\ \lambda \end{bmatrix},$$

there exist $x \geq 0, \lambda \geq 0$ such that $Ax = \lambda b$ and $c^T x < \lambda z$.

Proof of strong duality

Proof (continued). We have two cases:

Case 1: $\lambda \neq 0$.

- Since we can normalize by λ we can assume that $\lambda = 1$.
- This means that $\exists x \geq 0$ such that $Ax = b$ and $c^T x < z$.
- This contradicts the optimality of x^*.

Case 2: $\lambda = 0$.

- This means that $\exists x \geq 0$ such that $Ax = 0$ and $c^T x < 0$.
- Then $\forall \mu > 0$, $x^* + \mu x$ is feasible for P and its cost is

$$c^T(x^* + \mu x) = c^T x^* + \mu(c^T x) < z \ ,$$

which is again a contradiction.

□

Duality gap

Let P and D be

$$(P) \quad z = \min\{c^T x \ : \ Ax = b,\, x \geq 0\}$$
$$(D) \quad w = \max\{b^T y \ : \ A^T y \leq c\} \ ,$$

and let x be feasible in P and y feasible in D.

- By weak duality, we know that $c^T x \geq b^T y$.
- The difference $c^T x - b^T y$ is called the duality gap.
- The duality gap is zero iff x is optimal in P and y is optimal in D.
- That is, the duality gap can serve as a good measure of how close a feasible pair x and y are to the optimal solutions for P and D.

Duality gap

It is convenient to write the dual of a LP as

$$(D) \quad w = \max\{b^T y \ : \ A^T y + s = c \text{ for some } s \geq 0\} \ .$$

Then we can write the duality gap as follows:

$$\begin{aligned} c^T x - b^T y &= c^T x - x^T A^T y \\ &= x^T (c - A^T y) \\ &= x^T s \ , \end{aligned}$$

since $A^T y + s = c$. Using the above, we will prove a theorem that will allow us to check optimality of a primal and/or a dual solution.

Complementary slackness

Theorem (Complementary slackness)

Let $x^*, (y^*, s^*)$ be feasible for P, D respectively. The following are equivalent:

1. *x^* is an optimal solution to P and (y^*, s^*) is an optimal solution to D.*
2. $(s^*)^T x^* = 0$.
3. $x_j^* s_j^* = 0$ *for all* $j \in \{1, \ldots, n\}$.
4. *If* $s_j^* > 0$ *then* $x_j^* = 0$.

Complementary slackness

Proof. Suppose (1) holds, i.e., x^* and (y^*, s^*) are optimal solutions to P and D.

- By strong duality, $c^T x^* = b^T y^*$. Since $c = A^T y^* + s^*$ and $Ax^* = b$, we get that
$$(y^*)^T A x^* + (s^*)^T x^* = (x^*)^T A^T y^*$$
and thus $(s^*)^T x^* = 0$, i.e., (2) holds.
- Since $x_j^*, s_j^* \geq 0$, it follows that $x_j^* s_j^* = 0$ for all j, i.e., (3) holds.
- Hence, if $s_j^* > 0$ then $x_j^* = 0$ for all j, i.e., (4) holds.
- The converse also holds.

□

Complementary slackness

Example

Primal

$$\begin{array}{lrcl} \min & x_1 + 2x_2 + 4x_3 & & \\ \text{s.t.} & x_1 + x_2 + 2x_3 & = & 5 \\ & 2x_1 + x_2 + 3x_3 & = & 8 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Dual

$$\begin{array}{lrcl} \max & 5y_1 + 8y_2 & & \\ \text{s.t.} & y_1 + 2y_2 & \leq & 1 \\ & y_1 + y_2 & \leq & 2 \\ & 2y_1 + 3y_2 & \leq & 4 \end{array}$$

The complementary slackness equations corresponding to the primal solution $x = [3\ 2\ 0]^T$ would be:

$$\begin{aligned} y_1 + 2y_2 &= 1 \\ y_1 + y_2 &= 2\ . \end{aligned}$$

- This implies that $y_1 = 3$ and $y_2 = -1$.
- Since this solution satisfies the other constraint of the dual, y is dual feasible, proving that x is an optimum solution to the primal.
- This further implies that y is an optimum solution to the dual.

Complexity of linear programming

Definition (Problem $\mathcal{LP}$)

Input: Integral A, b, c and rational number λ.

Output: Is $\min\{c^T x \ : \ Ax = b, x \geq 0\} \leq \lambda$?

Theorem

$\mathcal{LP} \in$ NP $\cap$ co-NP.

Theorem

$\mathcal{LP} \in$ P.

Solving a linear program in polynomial time

The first polynomial-time algorithm for LP is the ellipsoid algorithm, proposed by (Khachiyan, 1979).

- It was first developed for convex programming (of which LP is a special case).
- It is impractical for linear programming.
- It has extensive applications in combinatorial optimization.

Solving a linear program in polynomial time

The first polynomial-time algorithm for LP is the ellipsoid algorithm, proposed by (Khachiyan, 1979).

- It was first developed for convex programming (of which LP is a special case).
- It is impractical for linear programming.
- It has extensive applications in combinatorial optimization.

(Karmarkar, 1984) presented another polynomial-time algorithm for LP.

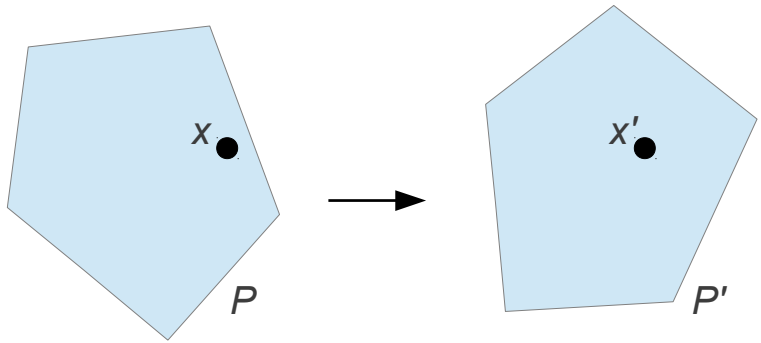
- It avoids the combinatorial complexity (inherent in the simplex algorithm) of the vertices of the polyhedron by staying well inside the polyhedron.
- It lead to many other algorithms for LP based on similar ideas.
- These algorithms are known as interior point methods.

Solving a linear program in polynomial time

High-level description of an interior-point algorithm:

1. If x (current solution) is close to the boundary, then map the polyhedron onto another one so that x is well in the interior of the new polyhedron.
2. Make a step in the transformed space.
3. Repeat (1) and (2) until we are close enough to an optimal solution.

The key is to stay in the interior of the feasible region.



The figure shows a centering mapping: if x is close to the boundary, we map P into P' s.t. the image x' of x is closer to the center of P'.

1. Linear programming

2. Bipartite matching
   - Bipartite graphs and matchings
   - Maximum cardinality matching
   - Minimum weight perfect matching

Bipartite graphs

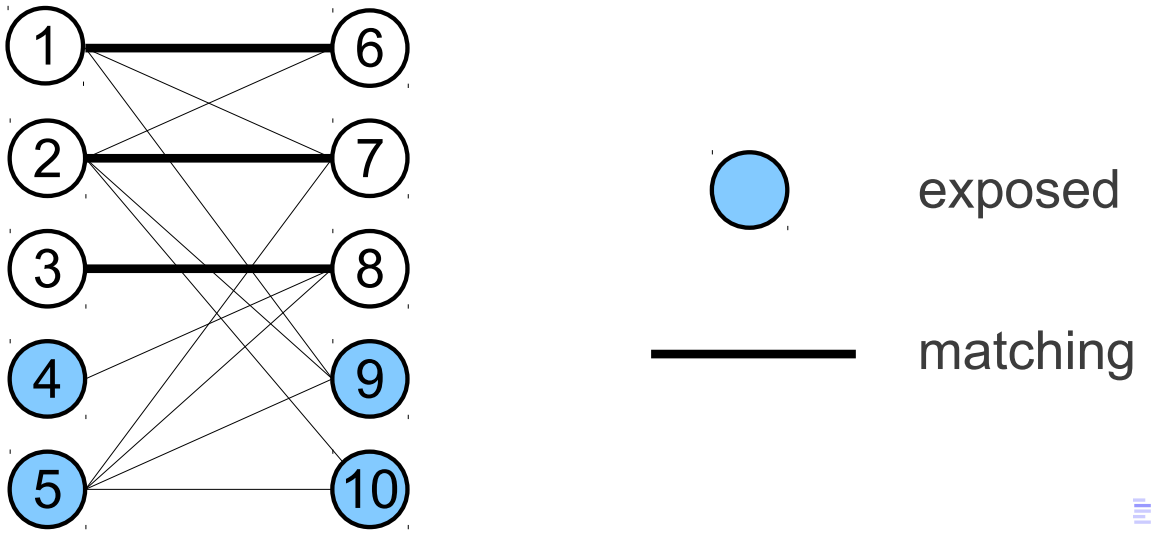
- A graph $G = (V, E)$ consists of a set V of vertices and a set E of pairs of vertices called edges.
- For an edge $e = (u, v)$, we say that the endpoints of e are u and v; we also say that e is incident to u and v.
- A graph $G = (V, E)$ is bipartite if the vertex set V can be partitioned into two sets A and B (the bipartition) such that no edge in E has both endpoints in the same set of the bipartition.
- We denote $V = A \cup B$, $G = ((A \cup B), E)$.

Matchings

Definition (Matching)

A matching $M \subseteq E$ is a collection of edges such that every vertex of V is incident to at most one edge of M.

- If a vertex v has no edge of M incident to it then v is said to be exposed (or unmatched).
- A matching is perfect if no vertex is exposed; in other words, a matching is perfect if its cardinality is equal to $|A| = |B|$.



Matching problems

We are interested in the following two problems on bipartite graphs:

Maximum cardinality matching problem

Find a matching M of maximum size.

Minimum weight perfect matching problem

Given a cost c_{ij} for all $(i,j) \in A \times B$, find a perfect matching of minimum cost where the cost of a matching M is given by

$$c(M) = \sum_{(i,j)\in M} c_{ij} \ .$$

This problem is also called the assignment problem.

Note that similar problems (but more complicated) can be defined on non-bipartite graphs.

Optimality of a matching

- Before describing an algorithm for solving the maximum cardinality matching problem, one would like to be able to prove optimality of a matching (without reference to any algorithm).
- For this purpose, one would like to find upper bounds on the size of any matching and hope that the smallest of these upper bounds be equal to the size of the largest matching.
- This is a duality concept that will be ubiquitous in this subject.

König's theorem

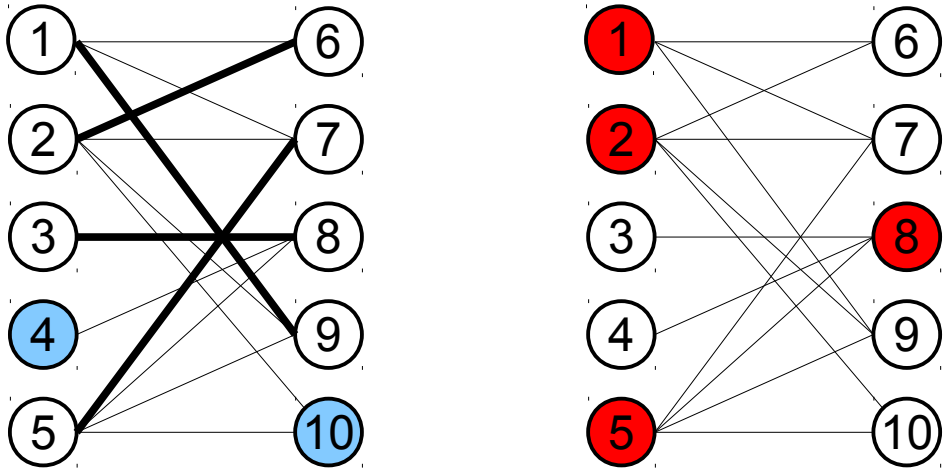
- A vertex cover is a subset C of vertices such that all edges $e \in E$ are incident to at least one vertex of C. In other words, there is no edge completely contained in $V \setminus C$.
- Clearly, the size of any matching is at most the size of any vertex cover: given any matching M, a vertex cover C must contain at least one of the endpoints of each edge in M.
- We have just proved *weak duality*: that the maximum size of a matching is at most the minimum size of a vertex cover. We will prove that equality in fact holds:

Theorem (König's theorem)

For any bipartite graph, the maximum size of a matching is equal to the minimum size of a vertex cover.

König's theorem

- We will prove this minmax relationship of vertex cover and bipartite matching by describing an efficient algorithm which simultaneously gives a maximum matching and a minimum vertex cover.
- König's theorem gives a good characterization of the problem, namely a simple proof of optimality.
- In our example, the matching $(1, 9), (2, 6), (3, 8)$ and $(5, 7)$ is of maximum size since there exists a vertex cover of size 4: $\{1, 2, 5, 8\}$.



Solving the cardinality matching problem

- The natural approach to solving this cardinality matching problem is to try a greedy algorithm: Start with any matching (e.g. an empty matching) and repeatedly add disjoint edges until no more edges can be added.
- This approach, however, is not guaranteed to give a maximum matching.
- We will present an algorithm that does work, and is based on the concepts of alternating paths and augmenting paths.

Alternating and augmenting paths

- A path is simply a collection of edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ where the v_i 's are distinct vertices. A path can simply be represented as $v_0 - v_1 - v_k$.
- An alternating path with respect to M is a path that alternates between edges in M and edges in $E \setminus M$.
- An augmenting path with respect to M is an alternating path in which the first and last vertices are exposed.

Alternating and augmenting paths

Properties

Some properties of augmenting paths:

- An augmenting path with respect to M which contains k edges of M must also contain exactly $k+1$ edges not in M.
- The two endpoints of an augmenting path must be on different sides of the bipartition.

The most interesting property of an augmenting path P with respect to a matching M is that

If we set $M' = (M \setminus P) \cup (P \setminus M)$, then we get a matching M' and, moreover, the size of M' is one unit larger than the size of M.

- This means that we can form a larger matching M' from M by taking the edges of P not in M and adding them to M' while removing from M' the edges in M that are also in the path P.
- We say that we have augmented M along P.

Alternating and augmenting paths

Usefulness

The usefulness of augmenting paths is given in the following theorem.

Theorem

A matching M is maximum if and only if there are no augmenting paths with respect to M.

Proof:

($\Rightarrow$) By contradiction:

- Let P be some augmenting path with respect to M.
- Set $M' = (M \setminus P) \cup (P \setminus M)$.
- Then M' is a matching with cardinality greater than M.
- This contradicts the maximality of M.

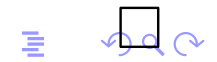
Alternating and augmenting paths

Usefulness

Proof (continued):

($\Leftarrow$) Again by contradiction: If M is not maximum, let M^* be a maximum matching (so that $|M^*| > |M|$). Let $Q = (M \setminus M^*) \cup (M^* \setminus M)$. Then:

- Q has more edges from M^* than from M (since $|M^*| > |M|$ implies $|M^* \setminus M| > |M \setminus M^*|$).
- Each vertex is incident to at most one edge in $M \cap Q$ and one edge $M^* \cap Q$.
- Thus Q is composed of cycles and paths that alternate between edges from M and M^*.
- Therefore there must be some path with more edges from M^* in it than from M (all cycles will be of even length and have the same number of edges from M^* and M). This path is an augmenting path with respect to M.
- Hence there must exist an augmenting path P with respect to M, which is a contradiction.



An algorithm for maximum cardinality matching

The theorem we just proved, i.e.,

A matching M is maximum if and only if there are no augmenting paths with respect to M.

motivates the following algorithm.

1. Start with any matching M, say the empty matching.
2. Repeatedly locate an augmenting path P with respect to M, augment M along P and replace M by the resulting matching.
3. Stop when no more augmenting path exists.

- By the above theorem, we are guaranteed to have found an optimum matching.
- The algorithm terminates in μ augmentations, where μ is the size of the maximum matching.
- Clearly, $\mu \leq \frac{n}{2}$, where $n = |V|$.

An algorithm for maximum cardinality matching

The question now is how to decide the existence of an augmenting path and how to find one, if one exists.
These tasks can be done as follows: Direct edges in G according to M so that

- an edge goes from A to B if it does not belong to the matching M and
- an edge goes from B to A if it does.

Call this directed graph D.

Lemma

There exists an augmenting path in G with respect to M iff there exists a directed path in D between an exposed vertex in A and an exposed vertex in B.

An algorithm for maximum cardinality matching

- This gives an $O(m)$ algorithm (where $m = |E|$) for finding an augmenting path in G.
- Let A^* and B^* be the set of exposed vertices w.r.t. M in A and B respectively.
- We can simply attach a vertex s to all the vertices in A^* and do a depth-first-search from s till we hit a vertex in B^* and then trace back our path.
- Thus the overall complexity of finding a maximum cardinality matching is $O(nm)$.
- This can be improved to $O(\sqrt{n}m)$ by augmenting along several augmenting paths simultaneously.

An algorithm for maximum cardinality matching

- If there is no augmenting path w.r.t. M, then we can also use our search procedure for an augmenting path in order to construct an optimum vertex cover.
- Consider the set L of vertices which can be reached by a directed path from an exposed vertex in A.
- The following claim immediately proves König's theorem.

Claim

When the algorithm terminates, $C^* = (A - L) \cup (B \cap L)$ is a vertex cover. Moreover, $|C^*| = |M^*|$ where M^* is the matching returned by the algorithm.

An algorithm for maximum cardinality matching

Proof. We first show that C^* is a vertex cover.

- Assume not. Then there must exist an edge $e = (a, b) \in E$ with $a \in A \cap L$ and $b \in (B - L)$.
- Edge e cannot belong to the matching. If it did, then b should be in L for otherwise a would not be in L.
- Hence, e must be in $E - M$ and is therefore directed from A to B.
- This implies that b can be reached from an exposed vertex in A (namely go to a and then take the edge (a, b)), contradicting the fact that $b \notin L$.

An algorithm for maximum cardinality matching

Proof (continued). To show that $|C^*| = |M^*|$, we show that $|C^*| \leq |M^*|$, since the reverse inequality is true for any matching and any vertex cover.

Observations:

1. No vertex in $A - L$ is exposed by definition of L.
2. No vertex in $B \cap L$ is exposed since this would imply the existence of an augmenting path and, thus, the algorithm would not have terminated.
3. There is no edge of the matching between a vertex $a \in (A - L)$ and a vertex $b \in (B \cap L)$. Otherwise, a would be in L.

These remarks imply that every vertex in C^* is matched and moreover the corresponding edges of the matching are distinct. Hence, $|C^*| \leq |M^*|$. $\square$

Hall's theorem

Hall's theorem gives a necessary and sufficient condition for a bipartite graph to have a matching which saturates (or matches) all vertices of A (i.e., a matching of size $|A|$).

Theorem (Hall, 1935)

Given a bipartite graph $G = (V, E)$ with bipartition A, B ($V = A \cup B$), G has a matching of size $|A|$ if and only if, for every $S \subseteq A$, we have $|N(S)| \geq |S|$, where $N(S) = \{b \in B \ : \ \exists a \in S$ with $(a, b) \in E\}$.

- Clearly, the condition given in Hall's theorem is necessary.
- Its sufficiency can be derived from König's theorem.

Minimum weighted perfect matching problem

- By assigning infinite costs to the edges not present, one can assume that the bipartite graph $G = ((A \cup B), E)$ is complete, i.e., $E = A \times B$.
- The minimum weight perfect matching problem is often described by the following story:

There are n jobs to be processed on n machines and one would like to process exactly one job per machine such that the total cost of processing the jobs is minimized.

Formally, we are given costs $c_{ij} \in \mathbb{R} \cup \{\infty\}$ for every $i \in A, j \in B$ and the goal is to find a perfect matching M minimizing $\sum_{(i,j)\in M} c_{ij}$.

Minimum weighted perfect matching problem

We start by giving a formulation of the problem as an integer program:

- an optimization problem in which
- the variables are restricted to integer values and
- the constraints and the objective function are linear functions of these variables.

We first need to associate a point to every matching:

Given a matching M, let its incidence vector be x where $x_{ij} = 1$ if $(i, j) \in M$ and 0 otherwise.

Minimum weighted perfect matching problem

Formulation as an integer program

We can now formulate the minimum weight perfect matching problem as follows:

$$\begin{array}{lll} \min & \sum_{i,j} c_{ij} x_{ij} & \\ \text{s. t.} & \sum_j x_{ij} = 1 & i \in A \\ & \sum_i x_{ij} = 1 & j \in B \\ & x_{ij} \in \{0,1\} & i \in A, j \in B \ . \end{array}$$

- This is not a linear program, but a so-called integer program (IP).
- Any solution to this integer program corresponds to a matching and therefore this is a valid formulation of the minimum weight perfect matching problem in bipartite graphs.

Minimum weighted perfect matching problem

Linear program relaxation

Consider now the linear program P obtained by dropping the integrality constraints:

$$\begin{array}{lll} \min & \sum_{i,j} c_{ij} x_{ij} & \\ \text{s. t.} & \sum_j x_{ij} = 1 & i \in A \\ & \sum_i x_{ij} = 1 & j \in B \\ & x_{ij} \geq 0 & i \in A, j \in B \ . \end{array}$$

- This is the linear programming relaxation of the above integer program.
- In a linear program, the variables can take fractional values and therefore there are many feasible solutions to the set of constraints above which do not correspond to matchings.
- But we only care about the optimum solutions.

Minimum weighted perfect matching problem

Linear program relaxation

Recall some facts about LPs:

- The set of feasible solutions to the constraints in P forms a bounded polyhedron or polytope, and when we optimize a linear constraint over a polytope, the optimum will be attained at one of the "corners" or vertices of the polytope.
- A vertex x of a set Q is an element $x \in Q$ which cannot be written as $\lambda y + (1 - \lambda)z$ with $0 < \lambda < 1$, $y, z \in Q$ with $y \neq z$.

Minimum weighted perfect matching problem

Linear program relaxation

- In general, the vertices of a LP are not guaranteed to have all coordinates integral.
- The general integer programming problem is NP-hard, while linear programming is polynomially solvable.
- As a result, in general, there is no guarantee that the value z_{IP} of an IP is equal to the value z_{LP} of its LP relaxation.
- However, since the IP is more constrained than the relaxation, we always have that $z_{IP} \geq z_{LP}$.
- This implies that z_{LP} is a lower bound on z_{IP} for a minimization problem.
- If an optimum solution to a LP relaxation is integral, then it must also be an optimum solution to the IP.

Minimum weighted perfect matching problem

Primal-dual algorithm

However, in the case of the perfect matching problem, the constraint matrix has a very special form and one can show the following very important result:

Theorem

Any vertex of P is a $0-1$ vector and, hence, is the incidence vector of a perfect matching.

Proof. We will describe a primal-dual algorithm for solving the minimum weight perfect matching problem.

Minimum weighted perfect matching problem

Primal-dual algorithm

Let us construct the dual D of P:

- Suppose we have values u_i for $i \in A$ and v_j for $j \in B$ such that $u_i + v_j \leq c_{ij}$ for all $i \in A$ and $j \in B$.
- Then for any perfect matching M, we have that

$$\sum_{(i,j)\in M} c_{ij} \geq \sum_{i\in A} u_i + \sum_{j\in B} v_j \ .$$

- Thus, $\sum_{i\in A} u_i + \sum_{j\in B} v_j$ is a lower bound on the cost of the minimum cost perfect matching.
- To get the best lower bound, we would like to maximize this quantity, and therefore we obtain the dual linear program D of P:

$$\begin{array}{ll} \max & \sum_{i\in A} u_i + \sum_{j\in B} v_j \\ \text{s.t.} & u_i + v_j \leq c_{ij} \qquad i \in A, j \in B \ . \end{array}$$

Minimum weighted perfect matching problem

Primal-dual algorithm

So far, we know that

$$\min_{\text{perfect matchings } M} \sum_{(i,j)\in M} c_{ij} \geq \min_{x\in P} \sum_{i,j} c_{ij}x_{ij} \geq \max_{(u,v)\in D} \left(\sum_{i\in A} u_i + \sum_{j\in B} v_j \right) .$$

If we could find a feasible solution u, v to D and a perfect matching M such that the cost of the perfect matching is equal to the value of the dual solution, then

- we have equality throughout,
- the matching found is optimum, and
- the incidence vector of the matching M is optimum for the LP P.

Minimum weighted perfect matching problem

Primal-dual algorithm

Given a solution u, v to the dual, a perfect matching M would satisfy equality if it contains only edges (i, j) such that $w_{ij} = c_{ij} - u_i - v_j = 0$.
$\Rightarrow$ This is what is referred to as complementary slackness.

Minimum weighted perfect matching problem

Primal-dual algorithm

Given a solution u, v to the dual, a perfect matching M would satisfy equality if it contains only edges (i, j) such that $w_{ij} = c_{ij} - u_i - v_j = 0$.
$\Rightarrow$ This is what is referred to as complementary slackness.

- The algorithm performs a series of iterations to obtain an appropriate u and v.
- It always maintains a dual feasible solution and tries to find an "almost" primal feasible solution x satisfying complementary slackness.
- The fact that complementary slackness is imposed is crucial in any primal-dual algorithm.

Minimum weighted perfect matching problem

Primal-dual algorithm

The algorithm works as follows.

- It first starts with any dual feasible solution, say $u_i = 0$ for all i and $v_j = \min_i c_{ij}$ for all j.
- In a given iteration, the algorithm has a dual feasible solution (u, v) or say (u, v, w).
- Imposing complementary slackness means that we are interested in matchings which are subgraphs of $E = \{(i, j) : w_{ij} = 0\}$.

Minimum weighted perfect matching problem

Primal-dual algorithm

- If E has a perfect matching then the incidence vector of that matching is a feasible solution in P and satisfies complementary slackness with the current dual solution and, hence, must be optimal.
- To check whether E has a perfect matching, one can use the cardinality matching algorithm we saw earlier.
- If the maximum matching output is not perfect then the algorithm will use information from the optimum vertex cover C^* to update the dual solution in such a way that the value of the dual solution increases (we are maximizing in the dual).

Minimum weighted perfect matching problem

Primal-dual algorithm

- As in the cardinality matching algorithm, consider the set L of vertices which can be reached by a directed path from an exposed vertex in A.
- Then there is no edge of E between $A \cap L$ and $B - L$.
- In other words, for every $i \in (A \cap L)$ and every $j \in (B - L)$, we have $w_{ij} = c_{ij} - u_i - v_j > 0$.
- Let $\delta = \min_{i \in (A \cap L), j \in (B - L)} w_{ij} > 0$.
- The dual solution is updated as follows:

$$u_i = \begin{cases} u_i & i \in A - L \\ u_i + \delta & i \in A \cap L \end{cases} \quad \text{and} \quad v_j = \begin{cases} v_j & j \in B - L \\ v_j - \delta & i \in B \cap L \end{cases}$$

Minimum weighted perfect matching problem

Primal-dual algorithm

- The above dual solution is feasible, in the sense that the corresponding vector w satisfies $w_{ij} \geq 0$ for all i and j.
- What is the value of the new dual solution?
- The difference between the values of the new dual solution and the old dual solution is equal to:

$$\delta(|A \cap L| - |B \cap L|) = \delta(|A \cap L| + |A - L| - |A - L| - |B \cap L|) = \delta(n/2 - |C^*|) \ .$$

  where A has size $n/2$ and C^* is the optimum vertex cover for the bipartite graph with edge set E.
- But by assumption $|C^*| < n/2$, implying that the value of the dual solution strictly increases.

Minimum weighted perfect matching problem

Primal-dual algorithm

- One repeats this procedure until the algorithm terminates.
- At that point, we have an incidence vector of a perfect matching and also a dual feasible solution which satisfy complementary slackness.
- They must therefore be optimal and this proves the existence of an integral optimum solution to P.
- By carefully choosing the cost function, one can make any vertex be the unique optimum solution to the linear program, and this shows that any vertex is integral and hence is the incidence vector of a perfect matching.

Minimum weighted perfect matching problem

Primal-dual algorithm

Proof that the algorithm terminates:

At least one more vertex of B must be reachable from an exposed vertex of A (and no vertex of B becomes unreachable), since an edge $e = (i, j)$ with $i \in (A \cap L)$ and $j \in B - L$ now has $w_{ij} = 0$ by our choice of δ.

This also gives an estimate of the number of iterations:

- In at most $n/2$ iterations, all vertices of B are reachable or the matching found has increased by at least one unit.
- Therefore, after $O(n^2)$ iterations, the matching found is perfect.
- The overall running time of the algorithm is thus $O(n^4)$ since it takes $O(n^2)$ to compute the set L in each iteration.
- By looking more closely at how vertices get labeled, one can reduce the running time analysis to $O(n^3)$.

Further reading

- J. H. van Lint and R. M. Wilson: A Course in Combinatorics. Cambridge University Press, 2006.
- Reinhard Diestel: Graph Theory. 4th electronic edition, 2010.
- Vašek Chvátal: Linear Programming. W. H. Freeman, 1983.