Mathematics of Operational Research

Contents

Richard Weber, Michaelmas Term 2015

Last updated on March 15, 2016.

Mathematics of Operational Research

In these notes I attempt a 'Goldilocks path' by being neither too detailed or too brief. These 2015 notes are newly revised, drawing heavily on notes written by Felix Fischer when he lectured the course for the four preceding years.

- Each lecture has a title and focuses upon just one or two ideas.
- The notes for each lecture are usually just 4–5 pages.

Schedules

• Lagrangian sufficiency theorem. Lagrange duality. Supporting hyperplane theorem. Sufficient conditions for convexity of the optimal value function. Lagrangian necessity. Fundamentals of linear programming. Linear program duality. Shadow prices. Complementary slackness. [2]

• Simplex algorithm. Two-phase method. Dual simplex algorithm. Gomory's cutting plane method. [2]

• Complexity of algorithms. NP-completeness. Exponential complexity of the simplex algorithm. Polynomial time algorithms for linear programming. Elipsoid method [3]

• Graphs and flows. Network simplex algorithm. Transportation and assignment problems, Ford-Fulkerson algorithm and max-flow/min-cut theorem. Shortest paths and minimum spanning trees. Bellman-Ford, Dijkstra's and Prim's algorithms. Hungarian algorithm. Interior point methods. Max-cut problem. Semidefinite programming. [5]

• Branch and bound. Dakin's method. Exact, approximate, and heuristic methods for the travelling salesman problem. Simulated annealing algorithm. [2]

• Cooperative and non-cooperative games. Strategic equilibrium. Two-player zero-sum games. Existence and computation of Nash equilibria in non-zero sum games. Lemke-Howson algorithm. Linear complementarity problems. Complementary pivoting. *Sperner's lemma*. [3]

• Cooperative and coalitional games, core, nucleolus, Shapley value. Bargaining. Social choice. Arrow's theorem. Gibbard-Satterthwaite theorem. [4]

• Auctions, revenue equivalence, optimal auctions. Mechanism design. Strategyproofness. Incentive compatibility. Vickrey-Clarke-Groves mechanisms. Mechanisms with payments. [3]

1 Lagrangian Methods

We consider an **optimization problem**, denoted P_b, having the standard form

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h(x) = b \\ &&& x \in X. \end{aligned} \tag{1.1}$$

Example 1.1. Minimize $x_1^2 + x_2^2$ subject to $a_1x_1 + a_2x_2 = b$ and $x_1, x_2 \geq 0$ for some given a_1, a_2 and b.

It consists of a vector $x \in \mathbb{R}^n$ of **decision variables**, an **objective function** $f : \mathbb{R}^n \to \mathbb{R}$, a **functional constraint** $h(x) = b$ where $h : \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$, and a **regional constraint** $x \in X$ where $X \subseteq \mathbb{R}^n$. The set $X(b) = \{x \in X : h(x) = b\}$ is called the **feasible set**, and P_b is called **feasible** if $X(b)$ is non-empty and **bounded** if $f(x)$ is bounded from below on $X(b)$. A vector x^* is called **optimal** if it is in the feasible set and minimizes f among all vectors in the feasible set. The assumption that the functional constraint holds with equality is without loss of generality since an inequality constraint like $g(x) \leq b$ can be re-written as $g(x) + z = b$, where z is a new **slack variable** with the additional regional constraint $z \geq 0$.

1.1 Lagrangian methods

A beautiful and powerful method for solving constrained optimization problems is that of Lagrange multipliers. The idea is to reduce constrained optimization to unconstrained optimization, and to take the (functional) constraints into account by augmenting the objective function with a weighted sum of them. To this end, define the **Lagrangian** associated with (1.1) as

$$L(x, \lambda) = f(x) - \lambda^T(h(x) - b), \tag{1.2}$$

where $\lambda \in \mathbb{R}^m$ is a vector of **Lagrange multipliers**.

The following result provides a condition under which minimizing the Lagrangian, subject only to the regional constraints, yields a solution to the original constrained problem. The result is easy to prove, yet extremely useful in practice.

Theorem 1.2 (Lagrangian Sufficiency Theorem)**.** *Suppose $x \in X$ and $\lambda \in \mathbb{R}^m$ such that $L(x, \lambda) = \inf_{x' \in X} L(x', \lambda)$ and $h(x) = b$. Then x is an optimal solution of* (1.1).

Proof. We have that

$$\begin{aligned} \min_{x' \in X(b)} f(x') &= \min_{x' \in X(b)} [f(x') - \lambda^T(h(x') - b)] \\ &\geq \min_{x' \in X} [f(x') - \lambda^T(h(x') - b)] \\ &= f(x) - \lambda^T(h(x) - b) = f(x). \end{aligned}$$

Equality in the first line holds because $h(x') - b = 0$ when $x' \in X(b)$. The inequality on the second line holds because the minimum is taken over a larger set. In the third line we finally use that x minimizes L and that $h(x) = b$. $\square$

Two remarks are in order. First, a vector λ of Lagrange multipliers satisfying the conditions of the theorem is not guaranteed to exist in general, but it does exist for a large class of problems.

Second, if we wish to find an optimal solution our general strategy is to minimize $L(x, \lambda)$ for all values of λ, in order to obtain a minimizer $x^*(\lambda)$ that depends on λ, and then find λ^* such that $x^*(\lambda^*)$ satisfies the constraints.

Let us apply this strategy to a concrete example.

Example 1.1. Consider minimizing $x_1^2 + x_2^2$ subject to $a_1x_1 + a_2x_2 = b$ and $x_1, x_2 \geq 0$ for some $a_1, a_2, b \geq 0$. The Lagrangian is

$$L(x_1, x_2), \lambda) = x_1^2 + x_2^2 - \lambda(a_1x_1 + a_2x_2 - b).$$

Taking partial derivaties reveals that it has a unique stationary point at $(x_1, x_2) = (\lambda a_1/2, \lambda a_2/2)$. We now choose λ so that the constraint $a_1x_1 + a_2x_2 = b$ is satisfied, which happens for $\lambda = 2b/(a_1^2 + a_2^2)$. Since since $\partial^2 L/\partial^2 x_1^2 > 0$, $\partial^2 L/\partial^2 x_2^2 > 0$, and $\partial^2 L/(\partial x_1 \partial x_2) = 0$ for this value of λ, we have found a minimum, having value $b^2/(a_1^2 + a_2^2)$ at $(x_1, x_2) = (a_1 b, a_2 b)/(a_1^2 + a_2^2)$.

More generally, to

$$\text{minimize } f(x) \text{ subject to } h(x) \leq b,\ x \in X, \tag{1.3}$$

we proceed as follows:

1. Introduce a vector z of slack variables to obtain the equivalent problem

$$\text{minimize } f(x) \text{ subject to } h(x) + z = b,\ x \in X,\ z \geq 0.$$

2. Compute the Lagrangian $L(x, z, \lambda) = f(x) - \lambda^T(h(x) + z - b)$.
3. Define the set

$$Y = \{\lambda \in \mathbb{R}^m : \inf_{x \in X, z \geq 0} L(x, z, \lambda) > -\infty\}.$$

4. For each $\lambda \in Y$, minimize $L(x, z, \lambda)$ subject only to the regional constraints, i.e. find $x^*(\lambda), z^*(\lambda)$ satisfying

$$L(x^*(\lambda), z^*(\lambda), \lambda) = \inf_{x \in X, z \geq 0} L(x, z, \lambda). \tag{1.4}$$

5. Find $\lambda^* \in Y$ so that $(x^*(\lambda^*), z^*(\lambda^*))$ is feasible, i.e. so $x^*(\lambda^*) \in X$, $z^*(\lambda^*) \geq 0$, and $h(x^*(\lambda^*)) + z^*(\lambda^*) = b$. By Theorem 1.2, $x^*(\lambda^*)$ is optimal for (1.3).

1.2 The Lagrange dual

A useful concept arising from the method of Lagrange multipliers is that of a dual problem. Denote by $\phi(b) = \inf_{x\in X(b)} f(x)$ the solution of (1.1), and define the (Lagrange) dual function $g : \mathbb{R}^m \to \mathbb{R}$ as the minimum value of the Lagrangian over X, i.e.

$$g(\lambda) = \inf_{x\in X} L(x, \lambda).$$

Then, for all $\lambda \in \mathbb{R}^m$,

$$\inf_{x\in X(b)} f(x) = \inf_{x\in X(b)} L(x, \lambda) \geq \inf_{x\in X} L(x, \lambda) = g(\lambda), \tag{1.5}$$

i.e. the dual function provides a lower bound on the optimal value of (1.1). Since this holds for every value of λ, it is interesting to choose λ to make the lower bound as large as possible. This motivates the **dual problem**, defined as

$$\begin{array}{ll} \text{maximize} & g(\lambda) \\ \text{subject to} & \lambda \in Y, \end{array}$$

where $Y = \{\lambda \in \mathbb{R}^m : g(\lambda) > -\infty\}$. In this context (1.1) is called the **primal problem**. Equation (1.5) is a proof of the **weak duality theorem**, which states that

$$\inf_{x\in X(b)} f(x) \geq \max_{\lambda\in Y} g(\lambda).$$

The primal problem (1.1) is said to satisfy **strong duality** if this holds with equality, i.e. if there exists λ such that

$$\phi(b) = g(\lambda).$$

If this is the case, then (1.1) can be solved using the method of Lagrangian multipliers. We can of course just try the method and see whether it works, as we did for Example 1.1. For certain important classes of optimization problems, however, it can be guaranteed that strong duality always holds.

1.3 Supporting hyperplanes

We say that ϕ has a (non-vertical) **supporting hyperplane** at b if there exists finite $\lambda \in \mathbb{R}^m$, such that for all $c \in \mathbb{R}^m$

$$\phi(c) \geq \phi(b) + \lambda^T(c - b).$$

Theorem 1.3. *The following are equivalent:*

1. *there exists a (non-vertical) supporting hyperplane to ϕ at b;*
2. *the problem satisfies strong duality.*

Proof. Suppose there exists a supporting hyperplane to ϕ at b. This means that there exists $\lambda \in \mathbb{R}^m$ such that

$$\begin{aligned}
\phi(b) &= \inf_{c \in \mathbb{R}^m} \big(\phi(c) - \lambda^T(c-b)\big) \\
&= \inf_{c \in \mathbb{R}^m} \inf_{x \in X(c)} \big(f(x) - \lambda^T(h(x)-b)\big) \\
&= \inf_{x \in X} L(x,\lambda) \\
&= g(\lambda).
\end{aligned}$$

Now suppose that the problem satisfies strong duality. Then there exists $\lambda \in \mathbb{R}^m$ such that

$$\phi(b) \leq L(x,\lambda) = f(x) - \lambda^T(h(x)-b)$$

Minimizing the right hand side over $x \in X(c)$ yields that for all $c \in \mathbb{R}^m$

$$\phi(b) \leq \phi(c) - \lambda^T(c-b),$$

$$\phi(b) - \lambda^T(b-c) \leq \phi(c).$$

This describes a supporting hyperplane to ϕ at b. □

We can also give an interpretation of the dual problem in terms of supporting hyperplanes. Consider the hyperplane given by $\alpha : \mathbb{R}^m \to \mathbb{R}$ with

$$\alpha(c) = \beta - \lambda^T(b-c).$$

This hyperplane has intercept β at b and slope λ. We now try to find $\phi(b)$ as follows:

1. For each λ, find $\beta_\lambda = \max\{\beta : \alpha(c) \leq \phi(c) \text{ for all } c \in \mathbb{R}^m\}$.
2. Choose λ to maximize β_λ.

This is the dual problem, since

$$\begin{aligned}
g(\lambda) &= \inf_{x \in X} L(x,\lambda) \\
&= \inf_{c \in \mathbb{R}^m} \inf_{x \in X(c)} \big(f(x) - \lambda^T(h(x)-b)\big) \\
&= \inf_{c \in \mathbb{R}^m} \big(\phi(c) - \lambda^T(c-b)\big) \\
&= \sup\big\{\beta : \beta - \lambda^T(b-c) \leq \phi(c) \text{ for all } c \in \mathbb{R}^m\big\} \\
&= \beta_\lambda
\end{aligned}$$

We again see the weak duality result as $\max_\lambda \beta_\lambda \leq \phi(b)$, and that equality holds if there is a supporting hyperplane to ϕ at b.

In the next lecture we will see that a supporting hyperplane exists for all $b \in \mathbb{R}^m$ if $\phi(b)$ is a convex function of b, and we will give sufficient conditions for this to be the case.

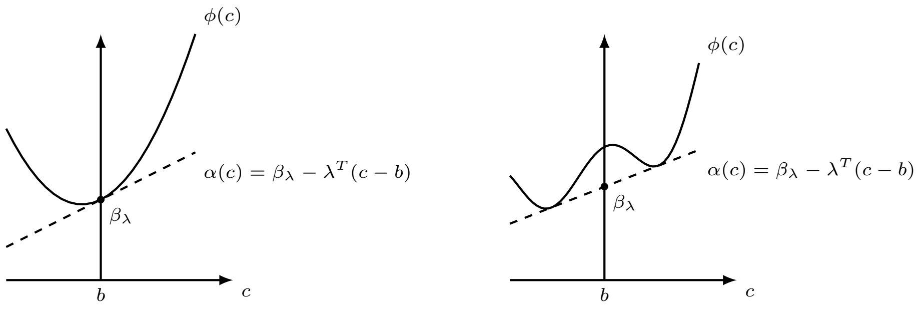

$\phi(c)$
$\alpha(c) = \beta_\lambda - \lambda^T(c-b)$
β_λ
b
c
$\phi(c)$
$\alpha(c) = \beta_\lambda - \lambda^T(c-b)$
β_λ
b
c

Figure 1: Geometric interpretation of the dual with optimal value $g(\lambda) = \beta_\lambda$. In the situation on the left strong duality holds, and $\beta_\lambda = \phi(b)$. In the situation on the right, strong duality does not hold, and $\beta_\lambda < \phi(b)$.

Homework

1. Consider P_b when $f(x) = -x$, and in two cases
   (a) $h(x) = \sqrt{x}$, $b = 2$;
   (b) $h(x) = x^2$, $b = 16$.

   What happens when you try to solve these problems using Lagrangian methods? In each case, find $\phi(b)$ and explain whether strong duality holds.

2. Given $a_1, \dots, a_n > 0$,

$$\begin{array}{ll} \text{minimize} & -\sum_{i=1}^n \log(a_i + x_i) \\ \text{subject to} & x_1, \dots, x_n \geq 0 \text{ and } \sum_i x_i = b. \end{array}$$

   The optimal x corresponds to one that can be found by a so-called 'water filling algorithm'. Imagine placing bars of heights a_i side by side in the fashion of a histogram and then flooding above these bars so as to cover area of b. Draw a picture to illustrate this idea.

2 Convex and Linear Optimization

2.1 Convexity and strong duality

A set $S \subseteq \mathbb{R}^n$ is **convex set** if for all $\delta \in [0,1]$, $x, y \in S$ implies that $\delta x + (1-\delta)y \in S$. A function $f : S \to \mathbb{R}$ is called **convex function** if for all $x, y \in S$ and $\delta \in [0,1]$, $\delta f(x) + (1-\delta)f(y) \geq f(\delta x + (1-\delta)y)$. A point $x \in S$ is called an **extreme point** of S if for all $y, z \in S$ and $\delta \in (0,1)$, $x = \delta y + (1-\delta)z$ implies that $x = y = z$. A point $x \in S$ is called an **interior point** of S if there exists $\epsilon > 0$ such that $\{y : ||y - x||_2 \leq \epsilon\} \subseteq S$. The set of all interior points of S is called the **interior** of S.

We saw in the previous lecture that strong duality is equivalent to the existence of a supporting hyperplane. The following result gives a sufficient condition for this.

Theorem 2.1 (Supporting Hyperplane Theorem)**.** *Suppose that ϕ is convex and $b \in \mathbb{R}$ lies in the interior of the set of points where ϕ is finite. Then there exists a (non-vertical) supporting hyperplane to ϕ at b.*

A **convex optimization problem** is one in which the objective function and feasible region are convex. The following identifies a condition that guarantees convexity of ϕ.

Theorem 2.2. *Let*

$$\phi(b) = \min\{f(x) : h(x) \leq b,\ x \in X\}.$$

Suppose X, f and h are convex. Then ϕ is convex.

Proof. Consider $b_1, b_2 \in \mathbb{R}^m$ such that $\phi(b_1)$ and $\phi(b_2)$ are defined, and let $\delta \in [0,1]$ and $b = \delta b_1 + (1-\delta)b_2$. Further consider $x_1 \in X(b_1)$, $x_2 \in X(b_2)$, and let $x = \delta x_1 + (1-\delta)x_2$. Then convexity of X implies that $x \in X$, and convexity of h that

$$\begin{aligned} h(x) &= h(\delta x_1 + (1-\delta)x_2) \\ &\leq \delta h(x_1) + (1-\delta)h(x_2) \\ &= \delta b_1 + (1-\delta)b_2 \\ &= b. \end{aligned}$$

Thus $x \in X(b)$, and by convexity of f,

$$\phi(b) \leq f(x) = f(\delta x_1 + (1-\delta)x_2) \leq \delta f(x_1) + (1-\delta)f(x_2).$$

This holds for all $x_1 \in X(b_1)$ and $x_2 \in X(b_2)$, so taking infima on the right hand side yields

$$\phi(b) \leq \delta\phi(b_1) + (1-\delta)\phi(b_2).$$ □

Observe that an equality constraint $h(x) = b$ is equivalent to constraints $h(x) \leq b$ and $-h(x) \leq -b$. In this case, the above result requires that X, f, h, and $-h$ are all convex, which in particular requires that h is linear.

2.2 Linear programs

A **linear program** is an optimization problem in which the objective and all constraints are linear. It has the form

$$\begin{aligned}
\text{minimize} \quad & c^T x \\
\text{subject to} \quad & a_i^T x \geq b_i, && i \in M_1 \\
& a_i^T x \leq b_i, && i \in M_2 \\
& a_i^T x = b_i, && i \in M_3 \\
& x_j \geq 0, && j \in N_1 \\
& x_j \leq 0, && j \in N_2
\end{aligned}$$

where $c \in \mathbb{R}^n$ is a cost vector, $x \in \mathbb{R}^n$ is a vector of decision variables, and constraints are given by $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ for $i \in \{1, \dots, m\}$. Index sets $M_1, M_2, M_3 \subseteq \{1, \dots, m\}$ and $N_1, N_2 \subseteq \{1, \dots, n\}$ are used to distinguish between different types of constraints.

An equality constraint $a_i^T x = b_i$ is equivalent to the pair of constraints $a_i^T \leq b_i$ and $a_i^T x \geq b_i$, and a constraint of the form $a_i^T x \leq b_i$ can be rewritten as $(-a_i)^T x \geq -b_i$. Each occurrence of an unconstrained variable x_j can be replaced by $x_j^+ + x_j^-$, where x_j^+ and x_j^- are two new variables with $x_j^+ \geq 0$ and $x_j^- \leq 0$. We can thus write every linear program in the *general form*

$$\min \{c^T x : Ax \geq b, x \geq 0\} \tag{2.1}$$

where $x, c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$. Observe that constraints of the form $x_j \geq 0$ and $x_j \leq 0$ are just special cases of constraints of the form $a_i^T x \geq b_i$, but we often choose to make them explicit.

A linear program of the form

$$\min \{c^T x : Ax = b, x \geq 0\} \tag{2.2}$$

is said to be in *standard form*. The standard form is of course a special case of the general form. On the other hand, we can also bring every general form problem into the standard form by replacing each inequality constraint of the form $a_i^T x \leq b_i$ or $a_i^T x \geq b_i$ by a constraint $a_i^T x + s_i = b_i$ or $a_i^T x - s_i = b_i$, where s_i is a new so-called **slack variable**, and an additional constraint $s_i \geq 0$.

The general form is typically used to discuss the theory of linear programming, while the standard form is more convenient when designing algorithms for linear programming.

Example 2.3. Consider the following linear program, as illustrated in Figure 2:

$$\begin{aligned}
\text{minimize} \quad & -(x_1 + x_2) \\
\text{subject to} \quad & x_1 + 2x_2 \leq 6 \\
& x_1 - x_2 \leq 3 \\
& x_1, x_2 \geq 0
\end{aligned}$$

Solid lines indicate sets of points for which one of the constraints is satisfied with equality. The feasible set is shaded. Dashed lines, orthogonal to the cost vector c, indicate sets of points for which the value of the objective function is constant. The optimal value over the feasible set is attained at point C.

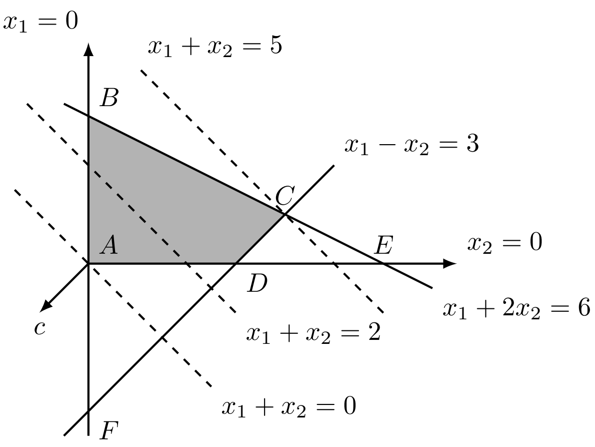


Figure 2: Geometric interpretation of the linear program of Example 2.3

2.3 Linear program duality

Consider problem (2.1) and introduce slack variables z to turn it into

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

We have $X = \{(x, z) : x \geq 0, z \geq 0\} \subseteq \mathbb{R}^{m+n}$. The Lagrangian is given by

$$L((x, z), \lambda) = c^T x - \lambda^T (Ax - z - b) = (c^T - \lambda^T A)x + \lambda^T z + \lambda^T b$$

and has a finite minimum over X if and only if

$$\lambda \in Y = \{\mu \in \mathbb{R}^m : c^T - \mu^T A \geq 0, \mu \geq 0\}.$$

For $\lambda \in Y$, the minimum of $L((x, z), \lambda)$ is attained when both $(c^T - \lambda^T A)x = 0$ and $\lambda^T z = 0$, and thus

$$g(\lambda) = \inf_{(x,z)\in X} L((x, z), \lambda) = \lambda^T b.$$

We obtain the dual

$$\max\{b^T \lambda : A^T \lambda \leq c, \lambda \geq 0\}. \tag{2.3}$$

The dual of (2.2) can be determined analogously as $\max\{b^T \lambda : A^T \lambda \leq c\}$. This differs from (2.3) in that the sign of λ is now unconstrained.

2.4 Complementary slackness

An important relationship between primal and dual solutions is provided by conditions known as **complementary slackness**. Complementary slackness requires that slack does not occur simultaneously in a variable, of the primal or dual, and the corresponding constraint, of the dual or primal. Here, a variable is said to have slack if its value is non-zero, and an inequality constraint is said to have slack if it does not hold with equality. It is not hard to see that complementary slackness is a necessary condition for optimality. Indeed, if complementary slackness was violated by some variable and the corresponding constraint, reducing the value of the variable would reduce the value of the Lagrangian, contradicting optimality of the current solution. Recall that the variables of the dual correspond to the Lagrange multipliers. The following result formalizes this intuition.

Theorem 2.4. *Let x and λ be feasible solutions for the primal (2.1) and the dual (2.3), respectively. Then x and λ are optimal if and only if they satisfy complementary slackness, i.e. if*

$$(c^T - \lambda^T A)x = 0 \quad \text{and} \quad \lambda^T(Ax - b) = 0. \tag{2.4}$$

Proof. Since x and λ are feasible, (2.4) holds if and only if $(c^T - \lambda^T A)x + \lambda^T(Ax - b) = 0$. But this is equivalent to $c^T x = \lambda^T b$, which holds if and only if x and λ are optimal. □

2.5 Shadow prices

A more intuitive understanding of Lagrange multipliers can be obtained by again viewing (1.1) as a family of problems parameterized by $b \in \mathbb{R}^m$. As before, let $\phi(b) = \inf\{f(x) : h(x) \leq b, x \in \mathbb{R}^n\}$. It turns out that at the optimum, the Lagrange multipliers equal the partial derivatives of ϕ.

Theorem 2.5. *Suppose that f and h are continuously differentiable on $\mathbb{R}^n$, and that there exist unique functions $x^* : \mathbb{R}^m \to \mathbb{R}^n$ and $\lambda^* : \mathbb{R}^m \to \mathbb{R}^m$ such that for each $b \in \mathbb{R}^m$, $h(x^*(b)) = b$ and $f(x^*(b)) = \phi(b) = \inf\{f(x) - \lambda^*(b)^T(h(x) - b) : x \in \mathbb{R}^n\}$. If x^* and λ^* are continuously differentiable, then*

$$\frac{\partial \phi}{\partial b_i}(b) = \lambda_i^*(b).$$

Proof. We have that

$$\begin{aligned}\phi(b) &= f(x^*(b)) - \lambda^*(b)^T(h(x^*(b)) - b) \\ &= f(x^*(b)) - \lambda^*(b)^T h(x^*(b)) + \lambda^*(b)^T b.\end{aligned}$$

Taking partial derivatives

$$\frac{\partial \phi(b)}{\partial b_i} = \sum_{j=1}^{n} \left(\frac{\partial f}{\partial x_j}(x^*(b)) - \lambda^*(b)^T \frac{\partial h}{\partial x_j}(x^*(b)) \right) \frac{\partial x_j^*}{\partial b_i}(b) \\ - \frac{\partial \lambda^*(b)^T}{\partial b_i}(h(x^*(b)) - b) + \lambda^*(b)^T \frac{\partial b}{\partial b_i}.$$

The first term on the right-hand side is zero, because $L(x, \lambda^*(b))$ is stationary with respect to x_j at $x^*(b)$. The second term is zero as well, because $x^*(b)$ is feasible and thus $(h(x^*(b)) - b) = 0$. The claim follows. □

This result continues to hold when the functional constraints are inequalities: $h(x) \leq b$. If the ith constraint is not satisfied with equality, then $\lambda_i^* = 0$ by complementary slackness, and therefore also $\partial \lambda_i^* / \partial b_i = 0$. Also, $\lambda^*(b) \leq 0$.

In light of Theorem 2.5, Lagrange multipliers are also known as **shadow prices**, due to an economic interpretation of the problem to

$$\begin{array}{ll} \text{maximize} & f(x) \\ \text{subject to} & h(x) \leq b \\ & x \in X. \end{array}$$

Consider a firm that produces n different goods from m different raw materials. Vector $b \in \mathbb{R}^m$ describes the amount of each raw material available to the firm, vector $x \in \mathbb{R}^n$ the quantity produced of each good. Functions $h : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}$ finally describe the amounts of raw material required to produce, and the profit derived from producing, particular quantities of the goods. The goal in the above problem thus is to maximize the profit of the firm for given amounts of raw materials available to it. The **shadow price** of raw material i then is the price the firm would be willing to pay per additional unit of this raw material, which of course should be equal to the additional profit derived from it, i.e. to $\partial \phi(b)/\partial b_i$. In this context, complementary slackness corresponds to the basic economic principle that a particular raw material has a non-zero price if and only if it is scarce, in the sense that increasing its availability would increase profit.

3 The Simplex Algorithm

3.1 Basic solutions

In the LP of Example 2.3, the optimal solution happened to lie at an extreme point of the feasible set. This was not a coincidence. Consider an LP in general form,

$$\text{maximize } c^T x \text{ subject to } Ax \leq b,\ x \geq 0. \tag{3.1}$$

The feasible set of this LP is a convex polytope in $\mathbb{R}^n$, i.e. an intersection of half-spaces. Each level set of the objective function $c^T x$, i.e. each set $L_\alpha = \{x \in \mathbb{R}^n : c^T x = \alpha\}$ of points for which the value of the objective function is equal to some constant $\alpha \in \mathbb{R}$, is a k-dimensional flat for some $k \leq n$. The goal is to find the largest value of α for which $L_\alpha(f)$ intersects with the feasible set. If such a value exists, the intersection contains either a single point or an infinite number of points, and it is guaranteed to contain an extreme point of the feasible set. This fact is illustrated in Figure 3.

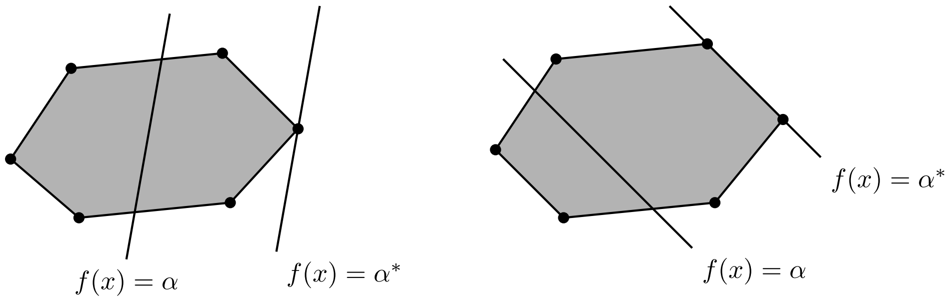


Figure 3: Illustration of linear programs with one optimal solution (left) and an infinite number of optimal solutions (right)

The geometric characterization of extreme points, as points that cannot be written as a convex combination of two different points, is somewhat hard to work with. We therefore use an alternative, algebraic characterization. To this end, consider the following LP in standard form, which can be obtained from (3.1) by introducing slack variables:

$$\text{maximize } c^T x \text{ subject to } Ax = b,\ x \geq 0, \tag{3.2}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

The **support** of a vector x is the set of indices $S(x) = \{i : x_i \neq 0\}$. A solution $x \in \mathbb{R}^n$ of the equation $Ax = b$ is called basic if the size of its support is no more than m, i.e. if there exists a set $B \subseteq \{1, \ldots, n\}$ with $|B| = m$ such that $x_i = 0$ if $i \notin B$. The set B is then called **basis**, and variable x_i is called **basic** if $i \in B$ and **non-basic** if $i \notin B$. A basic solution x that also satisfies $x \geq 0$ is a **basic feasible solution** (BFS) of (3.2).

We finally distinguish basic solutions that have exactly m non-zero entries from those that have strictly fewer, and refer to the latter as degenerate.

From now on, we make the following assumptions, which are without loss of generality.

(i) The rows of A are linearly independent and thus it has rank m.

(ii) Every set of m columns of A are linearly independent.

These assumptions rule out degeneracies of the following types: that we might have $Ax + z = b$ in the primal for some (x, z) having less than m non-zero components, or in the dual we might have $A^T\lambda - w = c$ for some (λ, w) having less than n non-zero components.

3.2 Extreme points and optimal solutions

The extreme points of the feasible set are precisely the basic feasible solutions.

Theorem 3.1. *A vector is a basic feasible solution of $Ax = b$ if and only if it is an extreme point of the set $X(b) = \{x : Ax = b, x \geq 0\}$.*

Proof. Suppose x is not a BFS. So $|S(x)| > m$ and there exists y such that $S(y) \subseteq S(x)$ and $Ay = 0$. But then x is the midpoint of $x + \epsilon y$ and $x - \epsilon y$, both of which lie in $X(b)$ for small enough non-zero ϵ. So x is not an extreme point.

Suppose x is not an extreme point, so that $x = \delta y + (1-\delta)z$ for some distinct $y, z \in X(b)$. Then $Ax = b = Az + \delta A(y - z)$. So $A(y - z) = 0$, with $S(y - z) \subseteq S(x)$. So x is not a BFS. □

When looking for an optimum, we can restrict our attention basic feasible solutions.

Theorem 3.2. *If the linear program (3.2) is feasible and bounded, then it has an optimal solution that is a basic feasible solution.*

Proof. Suppose x is an optimal solution of (3.2) but it is not an extreme point. Suppose $x = \delta y + (1 - \delta)z$ for some distinct $y, z \in X(b)$. Since $c^T x \geq c^T y$ and $c^T x \geq c^T z$ by optimality of x, and since $c^T x = \delta c^T y + (1-\delta)c^T z$, we must have that $c^T x = c^T y = c^T z$. Consider $x' = \delta' y + (1 - \delta')z = z + \delta'(y - z)$. This is also optimal. Now by thinking about increasing δ' from 0 until some component of x' hits 0, we see that for some δ', $S(x') \subset S(x)$. Thus we have a new optimal solution with a smaller support. We can continue this way until we obtain an optimal solution that is an extreme point. □

Since there are only finitely many basic solutions, a naive approach to solving an LP is to go over all basic solutions and pick one that optimizes the objective. However, this would not be efficient, as the number of basic solutions may grow exponentially in the number of variables. We now study a famous method for solving linear programs, the simplex method, which explores the set of basic solutions in a more organized way.

3.3 The simplex tableau

We can understand the simplex method in terms of the so-called **simplex tableau**, which stores all the information required to explore the set of basic solutions.

Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $x \in \mathbb{R}^n$ such that $Ax = b$. Let B be a *basis*, i.e. a set $B \subseteq \{1, \dots, n\}$ with $|B| = m$, corresponding to a choice of m non-zero variables. Then

$$A_B x_B + A_N x_N = b,$$

where $A_B \in \mathbb{R}^{m\times m}$ and $A_N \in \mathbb{R}^{m\times(n-m)}$ respectively consist of the columns of A indexed by B and those not indexed by B, and x_B and x_N respectively consist of the rows of x indexed by B and those not indexed by B. Moreover, if x is a basic solution, then there is a basis B such that $x_N = 0$ and $A_B x_B = b$, and if x is a basic feasible solution, there is a basis B such that $x_N = 0$, $A_B x_B = b$, and $x_B \geq 0$.

For x with $Ax = b$ and basis B, we have that $x_B = A_B^{-1}(b - A_N x_N)$, and thus

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

Suppose that we want to maximize $c^T x$ and find that

$$c_N^T - c_B^T A_B^{-1} A_N \leq 0 \quad \text{and} \quad A_B^{-1} b \geq 0. \tag{3.3}$$

Then, for any feasible $x \in \mathbb{R}^n$, it holds that $x_N \geq 0$ and therefore $f(x) \leq c_B^T A_B^{-1} b$. The basic solution x^* with $x_B^* = A_B^{-1} b$ and $x_N^* = 0$, on the other hand, is feasible and satisfies $f(x^*) = c_B^T A_B^{-1} b$. It must therefore be optimal.

If alternatively $(c_N^T - c_B^T A_B^{-1} A_N)_i > 0$ for some i, then we can increase the value of the objective by increasing $(x_N)_i$. Either this can be done indefinitely, which means that the maximum is unbounded, or the constraints force some of the variables in the basis to become smaller and we have to stop when the first such variable reaches zero. In that case we have found a new BFS and can repeat the process.

Assuming that the LP is feasible and has a bounded optimal solution, there exists a basis B^* for which (3.3) is satisfied. The basic idea behind the simplex method is to start from an initial BFS and then move from basis to basis until B^* is found. The information required for this procedure can conveniently be represented by the so-called **simplex tableau**. For a given basis B, it takes the following form:[1]

[1]The columns of the tableau have been permuted such that those corresponding to the basis appear on the left. This has been done just for convenience: in practice we will always be able to identify the columns corresponding to the basis by the embedded identity matrix.

$$\begin{array}{c|c|c|c|}
 & \overbrace{\qquad B \qquad}^{m} & \overbrace{\qquad N \qquad}^{n-m} & \overbrace{\qquad\qquad}^{1} \\ \hline
m\left\{\right. & A_B^{-1}A_B = I & A_B^{-1}A_N & A_B^{-1}b \\ \hline
1\left\{\right. & c_B^T - c_B^T A_B^{-1} A_B = 0 & c_N^T - c_B^T A_B^{-1} A_N & -c_B^T A_B^{-1} b \\ \hline
\end{array}$$

The first m rows consist of the matrix A and the column vector b, multiplied by the inverse of A_B. Notice that for any basis B, the LP with constraints $A_B^{-1}Ax = A_B^{-1}b$ is equivalent to the one with constraints $Ax = b$. The first n columns of the last row are equal to $c^T - \lambda^T A$ for $\lambda^T = c_B^T A_B^{-1}$. The vector λ can be interpreted as a solution, not necessarily feasible, to the dual problem. In the last column of the last row we finally have the value $-f(x)$, where x is the BFS with $x_B = A_B^{-1}b$ and $x_N = 0$.

We will see later that the simplex method always maintains feasibility of this solution x. As a consequence it also maintains complementary slackness for x and $\lambda^T = c_B^T A_B^{-1}$: since we work with an LP in standard form, $\lambda^T(Ax - b) = 0$ follows automatically from the feasibility condition, $Ax = b$; the condition $(c^T - \lambda^T A)x = 0$ holds because $x_N = 0$ and $c_B^T - \lambda^T A_B = c_B^T - c_B^T A_B^{-1} A_B = 0$. What it then means for (3.3) to become satisfied is that $c^T - \lambda^T A \leq 0$, i.e. that λ is a feasible solution for the dual. Optimality of x is thus actually a consequence of Theorem 2.4.

3.4 The simplex method in tableau form

Consider a tableau of the following form, where the basis can be identified by the identity matrix embedded in (a_{ij}):

$$\begin{array}{|c|c|}
\hline
(a_{ij}) & a_{i0} \\ \hline
a_{0j} & a_{00} \\ \hline
\end{array}$$

The simplex method then proceeds as follows:

1. Find an initial BFS with basis B.
2. Check whether $a_{0j} \leq 0$ for every j. If yes, the current solution is optimal, so stop.
3. Choose j such that $a_{0j} > 0$, and choose $i \in \{i' : a_{i'j} > 0\}$ to minimize a_{i0}/a_{ij}. If $a_{ij} \leq 0$ for all i, then the problem is unbounded, so stop. If multiple rows minimize a_{i0}/a_{ij}, the problem has a degenerate BFS.
4. Update the tableau by multiplying row i by $1/a_{ij}$ and adding a $-(a_{kj}/a_{ij})$ multiple of row i to each row $k \neq i$. The goal is to get a 1 in row i and 0 everywhere else, so that after this step row i corresponds to the jth variable Then return to Step 2.

We will now describe the different steps of the simplex method in more detail and illustrate them using the LP of Example 2.3.

Finding an initial BFS

Finding an initial BFS is very easy when the constraints are of the form $Ax \leq b$ for $b \geq 0$. We can then write the constraints as $Ax + z = b$ for a vector z of slack variables with regional constraint $z \geq 0$, and obtain a BFS by setting $x = 0$ and $z = b$. Alternatively one may think of extending x to (x, z) and setting $(x_B, x_N) = (z, x) = (b, 0)$. We then have $A_B^{-1} = I$ and $c_B = 0$, and the entries in the tableau become A_N and c_N^T for the variables x_1 and x_2 that are not in the basis, and b and 0 in the last column. For the LP of Example 2.3 we obtain the following tableau, where rows and columns are labelled with the names of the corresponding variables:

$$
\begin{array}{c|cccc|c|}
 & x_1 & x_2 & z_1 & z_2 & a_{i0} \\
\cline{2-6}
z_1 & 1 & 2 & 1 & 0 & 6 \\
z_2 & 1 & -1 & 0 & 1 & 3 \\
\cline{2-6}
a_{0j} & 1 & 1 & 0 & 0 & 0 \\
\cline{2-6}
\end{array}
$$

If the constraints do not have this convenient form, finding an initial BFS requires more work. We will discuss this case in the next lecture.

Choosing a pivot column

If $a_{0j} \leq 0$ for all $j \geq 1$, the current solution is optimal. Otherwise we can choose a column j such that $a_{0j} > 0$ as the pivot column and let the corresponding variable enter the basis. If multiple candidate columns exist, choosing any one of them will lead to a new basis, but we could for example break ties toward the column that maximizes a_{0j} or the one with the smallest index. The candidate variables in our example are x_1 and x_2, so let us choose x_1. The pivot operation causes this variable to enter the basis.

Choosing the pivot row

If $a_{ij} \leq 0$ for all i, then the problem is unbounded and the objective can be increased by an arbitrary amount. Otherwise we choose a row $i \in \{i' : a_{i'j} > 0\}$ that minimizes a_{i0}/a_{ij}. This row is called the pivot row, and a_{ij} is called the pivot. If multiple rows minimize a_{i0}/a_{ij}, the problem has a degenerate BFS. In our example there is a unique choice, namely variable z_2. The pivot operation causes this variable to leave the basis.

Pivoting

The purpose of the pivoting step is to get the tableau into the appropriate form for the new BFS. For this, we multiply row i by $1/a_{ij}$ and add a $-(a_{kj}/a_{ij})$ multiple of row i

to each row $k \neq i$, including the last one. Our choice of the pivot row as a row that minimizes a_{i0}/a_{ij} turns out to be crucial, as it guarantees that the solution remains feasible after pivoting. In our example, we need to subtract the second row from both the first and the last row, after which the tableau looks as follows:

	x_1	x_2	z_1	z_2	a_{i0}
z_1	0	3	1	-1	3
x_1	1	-1	0	1	3
a_{0j}	0	2	0	-1	-3

The second row now corresponds to variable x_1, which has replaced z_2 in the basis.

We are now ready to choose a new pivot column. In our example, one further iteration yields the following tableau:

	x_1	x_2	z_1	z_2	a_{i0}
x_2	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
x_1	1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
a_{0j}	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	-5

This corresponds to the BFS where $x_1 = 4$, $x_2 = 1$, and $z_1 = z_2 = 0$, with an objective of -5. All entries in the last row are non-positive, so this solution is optimal.

3.5 Degeneracies and cycling

In the absence of degeneracies, the value of the objective function increases in every iteration of the simplex method, and an optimal solution or a certificate for unboundedness is found after a finite number of steps. When the simplex method encounters a degenerate BFS, however, it may remain at the same BFS despite changing basis. This would obviously cause the value of the objective function to remain the same as well, and the simplex method may in fact cycle indefinitely through a number of bases that all represent the same BFS. Such cycling can be avoided by a more careful choice of pivot rows and columns, and thus of the variables entering and leaving the basis. Bland's rule achieves this by fixing some ordering of the variables and then choosing, among all variables that could enter and leave in a given iteration, those that are minimal according to the ordering.

4 Advanced Simplex Procedures

4.1 The two-phase simplex method

The LP we solved in the previous lecture allowed us to find an initial BFS very easily. If an obvious candidate for an initial BFS does not exist, we can use an additional phase I to find a BFS. In phase II we then proceed as in the previous lecture.

Consider the LP

$$\begin{array}{ll} \text{maximize} & -6x_1 - 3x_2 \\ \text{subject to} & x_1 + x_2 \geq 1 \\ & 2x_1 - x_2 \geq 1 \\ & 3x_2 \leq 2 \\ & x_1, x_2 \geq 0, \end{array}$$

and introduce slack variables to obtain

$$\begin{array}{ll} \text{maximize} & -6x_1 - 3x_2 \\ \text{subject to} & x_1 + x_2 - z_1 = 1 \\ & 2x_1 - x_2 - z_2 = 1 \\ & 3x_2 + z_3 = 2 \\ & x_1, x_2, z_1, z_2, z_3 \geq 0. \end{array}$$

Unfortunately, the basic solution with $x_1 = x_2 = 0$, $z_1 = z_2 = -1$, and $z_3 = 2$ is not feasible. However, we can add an **artificial variable** to the left-hand side of each constraint, where the slack variable and the right-hand side have opposite signs, and then minimize the sum of the artificial variables starting from the obvious BFS where the artificial variables are non-zero instead of the corresponding slack variables. E.g.,

$$\begin{array}{ll} \text{minimize} & y_1 + y_2 \\ \text{subject to} & x_1 + x_2 - z_1 + y_1 = 1 \\ & 2x_1 - x_2 - z_2 + y_2 = 1 \\ & 3x_2 + z_3 = 2 \\ & x_1, x_2, z_1, z_2, z_3, y_1, y_2 \geq 0, \end{array}$$

and the goal of phase I is to solve this LP starting from the BFS where $x_1 = x_2 = z_1 = z_2 = 0$, $y_1 = y_2 = 1$, and $z_3 = 2$. If the original problem is feasible, we will be able to find a BFS where $y_1 = y_2 = 0$. This automatically gives us an initial BFS for the original problem.

In summary, the two-phase simplex method proceeds as follows:

1. Bring the constraints into equality form. For each constraint in which the slack variable and the right-hand side have opposite signs, or in which there is no slack variable, add an artificial variable that has the same sign as the right-hand side.
2. Phase I: minimize the sum of the artificial variables, starting from the BFS where the absolute value of the artificial variable for each constraint, or of the slack variable in case there is no artificial variable, is equal to the right-hand side.
3. If some artificial variable has a positive value in the optimal solution, the original problem is infeasible; stop.
4. Phase II: solve the original problem, starting from the BFS found in phase I.

While the original objective is not needed for phase I, it is useful to carry it along as an extra row in the tableau, because it will then be in the appropriate form at the beginning of phase II. In the example, phase I starts with the following tableau:

	x_1	x_2	z_1	z_2	z_3	y_1	y_2	
y_1	1	1	-1	0	0	1	0	1
y_2	2	-1	0	-1	0	0	1	1
z_3	0	3	0	0	1	0	0	2
II	-6	-3	0	0	0	0	0	0
I	3	0	-1	-1	0	0	0	2

Note that the objective for phase I is written in terms of the variables that are not in the basis. This can be obtained by first writing it in terms of y_1 and y_2, such that we have -1 in the columns for y_1 and y_2 and 0 in all other columns because we are **maximizing** $-y_1 - y_2$, and then adding the first and second row to make the entries for all variables in the basis equal to zero.

Phase I now proceeds by pivoting on a_{21} to get

	x_1	x_2	z_1	z_2	z_3	y_1	y_2	
	0	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	1	$-\frac{1}{2}$	$\frac{1}{2}$
	1	$-\frac{1}{2}$	0	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$
	0	3	0	0	1	0	0	2
II	0	-6	0	-3	0	0	3	3
I	0	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	0	$-\frac{3}{2}$	$\frac{1}{2}$

and on a_{14} to get

	x_1	x_2	z_1	z_2	z_3	y_1	y_2	
	0	3	-2	1	0	2	-1	1
	1	1	-1	0	0	1	0	1
	0	3	0	0	1	0	0	2
II	0	3	-6	0	0	6	0	6
I	0	0	0	0	0	-1	-1	0

Note that we could have chosen a_{12} as the pivot element in the second step, and would have obtained the same result.

This ends phase I as $y_1 = y_2 = 0$, and we have found a BFS for the original problem with $x_1 = z_2 = 1$, $z_3 = 2$, and $x_2 = z_1 = 0$. After dropping the columns for y_1 and y_2 and the row corresponding to the objective for phase I, the tableau is in the right form for phase II:

x_1	x_2	z_1	z_2	z_3	
0	3	-2	1	0	1
1	1	-1	0	0	1
0	3	0	0	1	2
0	3	-6	0	0	6

By pivoting on a_{12} we obtain the following tableau, corresponding to an optimal solution of the original problem with $x_1 = 2/3$, $x_2 = 1/3$, and value -5:

x_1	x_2	z_1	z_2	z_3	
0	1	$-\frac{2}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$
1	0	$-\frac{1}{3}$	$-\frac{1}{3}$	0	$\frac{2}{3}$
0	0	2	-1	1	1
0	0	-4	-1	0	5

It is worth noting that the problem we have just solved is the dual of the LP in Example 2.3, which we solved in the previous lecture, augmented by the constraint $3x_2 \leq 2$. Ignoring the column and row corresponding to z_3, the slack variable for this new constraint, the final tableau is essentially the negative of the transpose of the final tableau we obtained in the previous lecture. This makes sense because the additional constraint is not tight in the optimal solution, as we can see from the fact that $z_3 \neq 0$.

4.2 The dual simplex method

The (primal) simplex method maintains feasibility of the primal solution along with complementary slackness and seeks feasibility of the dual solution. Alternatively one could maintain feasibility of the dual solution and complementary slackness and seek feasibility of the primal solution. This is known as the dual simplex method.

One situation where the dual simplex method can be useful is when an initial feasible solution for the dual is easier to find than one for the primal. Consider the following LP, to which we have already added slack variables z_1 and z_2:

$$\begin{array}{ll} \text{minimize} & 2x_1 + 3x_2 + 4x_3 \\ \text{subject to} & x_1 + 2x_2 + x_3 - z_1 \quad = 3 \\ & 2x_1 - x_2 - 3x_3 - z_2 = 4 \\ & x_1, x_2, x_3, z_1, z_2 \qquad \geq 0. \end{array}$$

The primal simplex algorithm would have to use two phases, since the solution where $z_1 = -3$ and $z_2 = -4$ is not feasible. On the other hand, $c \geq 0$, so the dual solution with $\lambda_1 = \lambda_2 = 0$ satisfies $c^T - \lambda^T A \geq 0$ and is therefore feasible. We obtain the following tableau:

-1	-2	-1	1	0	-3
-2	1	3	0	1	-4
2	3	4	0	0	0

In the dual simplex algorithm the pivot is selected by picking a row i such that $a_{i0} < 0$ and a column $j \in \{j' : a_{ij'} < 0\}$ that minimizes $-a_{0j}/a_{ij}$. Pivoting then works just like in the primal algorithm. In the example we can pivot on a_{21} to obtain

0	$-\frac{5}{2}$	$-\frac{5}{2}$	1	$-\frac{1}{2}$	-1
1	$-\frac{1}{2}$	$-\frac{3}{2}$	0	$-\frac{1}{2}$	2
0	4	7	0	1	-4

and then on a_{12} to obtain

0	1	1	$-\frac{2}{5}$	$\frac{1}{5}$	$\frac{2}{5}$
1	0	-1	$-\frac{1}{5}$	$-\frac{2}{5}$	$\frac{11}{5}$
0	0	3	$\frac{8}{5}$	$\frac{1}{5}$	$-\frac{28}{5}$

We have reached the optimum of 28/5 with $x_1 = 11/5$, $x_2 = 2/5$, and $x_3 = 0$.

It is worth pointing out that for problems in which all constraints are inequality constraints, the optimal dual solution can also be read off from the final tableau. For problems of this type, the last m columns of the extended constraint matrix A correspond to the slack variables and therefore contain values 1 or -1 on the diagonal and 0 everywhere else. For the same reason, the last m columns of the vector c^T are 0. The values of the dual variables, each of them with opposite sign of the slack variable in the corresponding constraint, thus appear in the last m columns of the vector $(c^T - \lambda^T A)$ in the last row of the final tableau. In our example, we have $\lambda_1 = 8/5$ and $\lambda_2 = 1/5$.

4.3 Gomory's cutting plane method

Another situation where the dual simplex method can be useful is when we need to add constraints to an already solved LP. While such constraints can make the primal solution infeasible, they do not affect feasibility of the dual solution. We can therefore simply add the constraint and continue running the dual LP algorithm from the current solution until the primal solution again becomes feasible. The need to add constraints to an LP for example arises naturally in Gomory's cutting plane approach for solving integer programs (IPs). An IP is a linear program with the additional requirement that variables should be integral.

Assume that for a given IP we have already found an optimal (fractional) solution x^* with basis B, and let a_{ij} denote the entries of the final tableau, i.e. $a_{ij} = (A_B^{-1} A_j)_i$

and $a_{i0} = (A_B^{-1}b)_i$. If x^* is not integral, there has to be a row i such that a_{i0} is not integral, and for every feasible solution x,

$$x_i + \sum_{j \in N} \lfloor a_{ij} \rfloor x_j \leq x_i + \sum_{j \in N} a_{ij} x_j = a_{i0}.$$

The inequality holds because x is feasible, i.e. $x \geq 0$, the equality follows from the properties of the final tableau. If x is integral, the left-hand side is integral as well, and the inequality must still hold if the right-hand side is rounded down. Thus,

$$x_i + \sum_{j \in N} \lfloor a_{ij} \rfloor x_j \leq \lfloor a_{i0} \rfloor.$$

This inequality is satisfied by every (integral) feasible solution, but not by the current solution x^*, for which $x_i^* = a_{i0}$. It corresponds to a so-called **cutting plane**, a hyperplane that separates the current solution x^* from the feasible set. The idea behind the cutting plane method is to iteratively add cutting planes and solve the resulting linear programs using the dual simplex algorithm. As it turns out, this always leads to an optimal integral solution after a finite number of steps.

Consider again the final tableau on Page 20, and assume we now want an integral solution. By the first row, and assuming that all variables are integral and non-negative,

$$x_2 + x_3 - 1z_1 + 0z_2 \leq x_2 + x_3 - \frac{2}{5}z_1 + \frac{1}{5}z_2 = \frac{2}{5},$$

and so in fact

$$x_2 + x_3 - z_1 \leq 0.$$

We turn this into an equality constraint using a new slack variable, add it to the tableau, and bring it into the right form by subtracting the first constraint from it, obtaining

0	1	1	$-\frac{2}{5}$	$\frac{1}{5}$	0	$\frac{2}{5}$
1	0	-1	$-\frac{1}{5}$	$-\frac{2}{5}$	0	$\frac{11}{5}$
0	0	0	$-\frac{3}{5}$	$-\frac{1}{5}$	1	$-\frac{2}{5}$
0	0	3	$\frac{8}{5}$	$\frac{1}{5}$	0	$-\frac{28}{5}$

After one more round of the dual simplex algorithm we reach the optimal integral solution with $x_1 = 3$ and $x_2 = x_3 = 0$:

0	1	1	-1	0	1	0
1	0	-1	1	0	-2	3
0	0	0	3	1	-5	2
0	0	3	1	0	1	-6

5 Complexity of Algorithms

We have seen that the simplex algorithm inspects basic feasible solutions and is guaranteed to find an optimal solution after a finite number of steps. However, the number of basic feasible solutions is generally exponential in n, and evaluating the objective function at all of them would take a long time. It is therefore an interesting question whether there exist cases where the simplex algorithm has to look at a significant fraction of the set of all basic feasible solutions. If this is the case, then we can ask whether a similar property holds for every algorithm that solves the linear programming problem. Before we pursue this question in the next lecture, we prepare with some theory of algorithmic complexity.

5.1 Theory of algorithmic complexity

Formally, an **instance** of an optimization problem is given by its input. For linear programming this input consists of two vectors $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ and a matrix $A \in \mathbb{R}^{m \times n}$. If each real value is represented using at most k bits, the whole instance can be described by a string of $(mn + m + n)k$ bits. We will call this the **input size**.

A sensible way to define the complexity of a problem is via the complexity of the fastest algorithm that can solve it. The latter is typically measured in terms of the number of arithmetic or bit-level operations as a function of the input size, ignoring lower-order terms resulting from details of the implementation. The following notation is useful in this context: given two functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$, write

- $f(n) = O(g(n))$ if there exist constants c and n_0 such that for every $n \geq n_0$, $f(n) \leq cg(n)$,
- $f(n) = \Omega(g(n))$ if there exist constants c and n_0 such that for every $n \geq n_0$, $f(n) \geq cg(n)$, and
- $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

In other words, $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ mean that the asymptotic growth of $f(n)$ is respectively bounded from above or below by $g(n)$, up to a constant factor.

Gaussian elimination, for example, shows that solving a linear system $Ax = b$ with $A \in \mathbb{R}^{n \times n}$ has arithmetic complexity $O(n^3)$. The same bound can also be shown to hold for bit complexity.

5.2 P, NP, and polynomial-time reductions

In computational complexity theory, **efficient computation** is typically associated with running times that are at most **polynomial** in the size of the input. In many situations of interest, and also in this course, it suffices to study complexity-theoretic questions for **decision problems**, i.e. problems where the answer is just a single bit. An example of a decision problem in the context of linear programming would be the following: given a linear program and a number $k \in \mathbb{R}$, is the minimum value of the objective function less than k? Formally, a decision problem can be described by a language $L \subseteq \{0,1\}^*$, containing precisely the instances for which the answer is 1 in some encoding as strings of bits.

One might expect the answer to the question whether a particular problem can be solved efficiently to depend a lot on the details of the computational model one is using. Quite surprisingly, this turns out not to be the case: all computational models that are known to be physically realizable can simulate each other, and a particular model, the **Turing machine**, can simulate all others with polynomial overhead. A Turing machine has a finite number of states, finite control, and a readable and writable tape that can store intermediary results as strings of bits. The Turing machine is started with the input written on the tape. It then runs for a certain number of steps, and when it halts the output is inferred from the state or the contents of some designated part of the tape. In the context of decision problems, a Turing machine is said to **accept** input $x \in \{0,1\}^*$ if it halts with output 1.

The most important open problem in complexity theory concerns the relationship between the complexity classes P and NP. P is the class of decision problems that can be solved in **polynomial time**. Formally, a function $f : \{0,1\}^* \to \{0,1\}^*$ is computable in polynomial time if there exists a Turing machine M and $k \in \mathbb{N}$ with the following property: for every $x \in \{0,1\}^*$, if M is started with input x, then after $O(|x|^k)$ steps it halts with output $f(x)$. NP is the class of decision problems for which a given solution can be verified in polynomial time. Formally, $L \subseteq \{0,1\}^*$ is in NP if there exists a Turing machine M and $k \in \mathbb{N}$ with the following property: for every $x \in \{0,1\}^*$, $x \in L$ if and only if there exists a certificate $y \in \{0,1\}^*$ with $|y| = O(|x|^k)$ such that M accepts (x, y) after $O(|x|^k)$ steps. The name NP, for **nondeterministic polynomial time**, derives from an alternative definition as the class of decision problems solvable in polynomial time by a nondeterministic Turing machine. A nondeterministic Turing machine is a Turing machine that can make a non-deterministic choice at each step of its computation and is required to accept $x \in L$ only for some sequence of these choices. Finding a solution is obviously at least as hard as verifying a solution described by a certificate. Most people believe that it must be strictly harder, i.e. that P $\neq$ NP.

The relative complexity of different decision problems can be captured in terms of **reductions**. Intuitively, a reduction from one problem to another transforms every instance of the former into an equivalent instance of the latter, where equivalence means that both of them yield the same decision. For this transformation to preserve the complexity of the original problem, the reduction should of course have less power than is required to actually solve the original problem. In our case it makes sense to use reductions that can be computed in polynomial time. A decision problem $L \subseteq \{0,1\}^*$ is called **polynomial-time reducible** to a decision problem $K \subseteq \{0,1\}^*$, denoted $L \leq_p K$, if there exists a function $f : \{0,1\}^* \to \{0,1\}^*$ computable in polynomial time such that for every $x \in \{0,1\}^*$, $x \in L$ if and only if $f(x) \in K$. A problem K is called **NP-hard** if for every problem L in NP, $L \leq_p K$. A problem is called **NP-complete** if it is both in NP and NP-hard. The relation $\leq_p$ is transitive. NP-complete problems are thus the hardest problems in NP, in the sense that membership of any NP-complete problem in P would imply that P = NP. The existence of NP-complete problems is less obvious, but holds nonetheless. Figure 4 shows the relationship between P and NP.

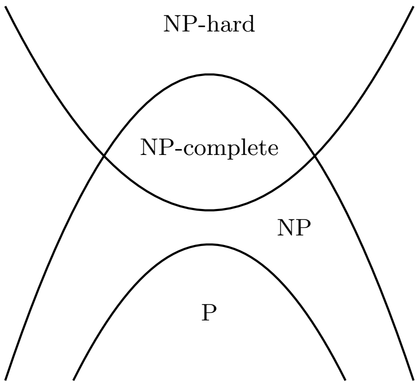


Figure 4: Relationship between P, NP, and the sets of NP-hard and NP-complete problems. It is not known whether the intersection between P and the set of NP-complete problems is empty. If it is not, then P = NP.

What is nice about the asymptotic worst-case notions of complexity considered above is that they do not require any assumptions about low-level details of the implementation or about the type of instances we will encounter in practice. We do, however, have to be a bit careful in interpreting results that use these notions. The fact that a problem is in P does not automatically mean that it can always be solved efficiently in practice, as the constant overhead hidden in the asymptotic notation might be prohibitively large. In fact, it does not even have to be the case that an algorithm with a polynomial worst-case running time is better in practice than an algorithm whose worst-case running time is exponential. Experience has shown, however, that for problems in P one is usually able to find algorithms that are fast in practice. On the other hand, NP-hardness of a problem does not mean that it can never be solved in practice, and we will consider approaches for solving NP-hard optimization problems in a later lecture. NP-hardness is still a very useful concept because it can help to direct efforts away from algorithms that are always efficient and toward algorithms with good practical performance.

5.3 Some NP-complete problems

The first problem ever shown to be NP-complete is the **Boolean satisfiability problem** (SAT), which asks whether a given Boolean formula is satisfiable. A Boolean formula consists of a set of clauses $C_i \subseteq X$ for $i = 1, \ldots, m$, where

$X = \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$ is a set of literals. It is called satisfiable if there exists a set $S \subseteq X$ such that $|S \cap \{x_j, \bar{x}_j\}| \leq 1$ for all $j = 1, \dots, n$ and $|S \cap C_i| \geq 1$ for all $i = 1, \dots, m$. Since the set S can serve as a certificate, it is easy to see that SAT is in NP. NP-hardness can be shown by encoding the operation of an arbitrary nondeterministic Turing machine as a Boolean formula.

Theorem 5.1 (Cook, 1971; Levin, 1973)**.** *Boolean satisfiability is NP-complete.*

An instance of the 0−1 **integer programming problem** consists of a matrix $A \in \mathbb{Z}^{m \times n}$ and a vector $b \in \mathbb{Z}^m$, and asks whether there exists a vector $x \in \{0,1\}^n$ such that $Ax \geq b$. It is associated with a special case of the integer programs we encountered in the previous lecture, in the context of the cutting plane method.

Theorem 5.2 (Karp, 1972)**.** 0−1 *integer programming is NP complete.*

Proof. Membership in NP is again easy to see. NP-hardness can be shown by a reduction from SAT. Consider a Boolean formula with literals $X = \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$ and clauses C_i, $i = 1, \dots, m$, and assume without loss of generality that $|C_i \cap \{x_j, \bar{x}_j\}| \leq 1$ for all $i = 1, \dots, m$ and $j = 1, \dots, n$. Now let $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$ be given by

$$a_{ij} = \begin{cases} 1 & \text{if } x_j \in C_i \\ -1 & \text{if } \bar{x}_j \in C_i \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, \dots, m \text{ and } j = 1, \dots, n,$$

$$b_i = 1 - |\{ j : \bar{x}_j \in C_i\}| \qquad \text{for } i = 1, \dots, m.$$

Intuitively, this integer program represents each Boolean variable by a binary variable, and each clause by a constraint that requires its literals to sum up to at least 1. To this end, the left hand side of the constraint contains x_j if the corresponding Boolean variable occurs as a positive literal in the clause, and $(1 - x_j)$ if it occurs as a negative literal. The above form is then obtained by moving all constants to the right hand side. It is now easy to see that there exists $x \in \{0,1\}^n$ such that $Ax \geq b$ if and only if the Boolean formula is satisfiable. □

The last problem we consider is the **travelling salesman problem** (TSP). For a given matrix $A \in \mathbb{N}^{n \times n}$, distances between n vertices, and a number $k \in \mathbb{N}$, it asks whether there exists a permutation $\sigma \in S_n$ such that $a_{\sigma(n)\sigma(1)} + \sum_{i=1}^{n-1} a_{\sigma(i)\sigma(i+1)} \leq k$. If the entries of the matrix A are interpreted as pairwise distances among a set of locations, we are looking for a tour $\sigma(1) \to \sigma(2) \to \cdots \to \sigma(n) \to \sigma(1)$ with length $\leq k$ that visits every location exactly once and returns to the starting point. The special case where A is a symmetric binary matrix and $k = 0$ is also known as the **Hamiltonian cycle problem**: i.e. which asks if there exist a tour of a graph's vertices which visits each vertex exactly once.

Theorem 5.3 (Karp, 1972)**.** *TSP is NP-complete, even if $A \in \{0,1\}^{n \times n}$ symmetric and $k = 0$.*

6 Computational Complexity of LP

6.1 A lower bound for the simplex method

The complexity of the simplex method depends on two factors, the number of steps in each round and the number of iterations. It is not hard to see that the tableau form requires $O(mn)$ arithmetic operations in each round. We will now describe an instance of the linear programming problem, and a specific pivot rule, such that the simplex method requires an exponential number of iterations to find the optimal solution. For this, we construct a polytope with an exponential number of vertices, and a so-called **spanning path** that traverses all of the vertices, in such a way that consecutive vertices are adjacent and a certain linear objective strictly increases along the path. This shows that the simplex method requires an exponential number of iterations in the worst case, for the specific pivoting rule that follows the spanning path.

Consider the unit cube in $\mathbb{R}^n$, given by the constraints

$$0 \leq x_i \leq 1 \quad \text{for } i = 1, \dots, n.$$

The unit cube has 2^n vertices, because either one of the two constraints $0 \leq x_i$ and $x_i \leq 1$ can be active for each dimension i. Further consider a spanning path of the unit cube constructed inductively as follows. In dimension 1, the path moves from $x_1 = 0$ to $x_1 = 1$. In dimension k, the path starts with $x_k = 0$ and traverses the spanning path for dimensions x_1 to x_{k-1}, which exists by the induction hypothesis. It then moves to the adjacent vertex with $x_k = 1$, and traverses the spanning path for dimensions x_1 to x_{k-1} in the reverse direction. This construction is illustrated of the left of Figure 5.

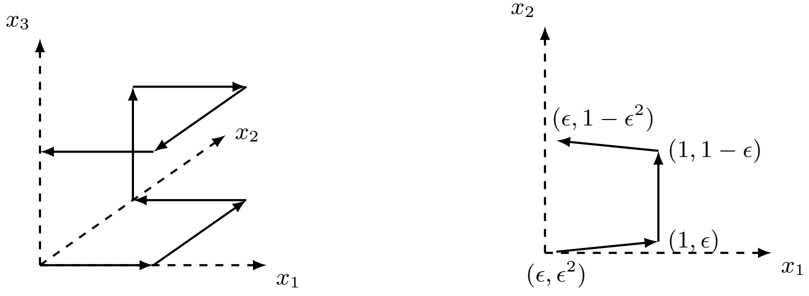


Figure 5: Spanning paths of the three-dimensional unit cube (left) and of the perturbed two-dimensional unit cube with $\epsilon = 1/10$ (right)

Now assume that we are trying to maximize the objective x_n, and observe that so far it increases only once, namely in the middle of the path. This can easily be fixed. Let $\epsilon \in (0, 1/2)$, and consider the perturbed unit cube with constraints

$$\begin{aligned} \epsilon \leq x_1 &\leq 1, \\ \epsilon x_{i-1} \leq x_i &\leq 1 - \epsilon x_{i-1} \quad \text{for } i = 2, \dots, n \end{aligned} \tag{6.1}$$

An example is shown on the right of Figure 5. It is easily verified that x_n now increases strictly along the path described above. We obtain the following result.

Theorem 6.1. *Consider the linear programming problem of maximizing x_n subject to (6.1). Then there exists a pivoting rule and an initial basic feasible solution such that the simplex method requires $2^n - 1$ iterations before it terminates.*

Observe that each of the numbers in the description of the perturbed unit cube can be represented using $O(\log \epsilon^{-n}) = O(n)$ bits, the number of iterations is therefore also exponential in the input size.

Interestingly, the first and last vertices of the spanning paths constructed above are adjacent, which means that a different pivoting rule could reach the optimal solution in a single step. However, similar worst-case instances have been constructed for many other pivot rules, and no pivot rule is known to guarantee a polynomial worst-case running time. The **diameter** of a polytope, i.e. the maximum number of steps necessary to get from any vertex to any other vertex, provides a lower bound of the number of iterations of the simplex method that is independent of the pivoting rule. The Hirsch conjecture, which states that the diameter of a polytope in dimension d with n facets cannot be greater than $n - d$, was disproved in 2010. Whether the diameter is bounded by a polynomial function of n and d remains open.

In practice, the performance of the simplex method is often much better, usually linear in the number of constraints. However, it is not clear how the intuition of a good average-case performance could be formalized, because this would require a natural probability distribution over instances of the linear programing problem. This is a problem that applies more generally to the average-case analysis of algorithms.

6.2 The idea for a new method

Again consider the linear program (2.2) and its corresponding dual:

$$\min\{c^T x : Ax = b, x \geq 0\}$$
$$\max\{b^T \lambda : A^T \lambda \leq c\}.$$

By strong duality, each of these problems has a bounded optimal solution if and only if the following set of linear constraints is feasible:

$$c^T x = b^T \lambda, \quad Ax = b, \quad x \geq 0, \quad A^T \lambda \leq c.$$

We can thus concentrate on the following decision problem: given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, is the set $\{x \in \mathbb{R}^n : Ax \geq b\}$ non-empty? We will now consider a method for solving this problem, known as the **ellipsoid method**.

We need some definitions. A symmetric matrix $D \in \mathbb{R}^{n \times n}$ is called **positive definite** if $x^T Dx > 0$ for all non-zero $x \in \mathbb{R}^n$. A set of vectors $E \subseteq \mathbb{R}^n$ given by

$$E = E(z, D) = \{\, x \in \mathbb{R}^n : (x - z)^T D^{-1}(x - z) \leq 1 \,\}$$

for a positive definite symmetric matrix $D \in \mathbb{R}^{n \times n}$ and a vector $z \in \mathbb{R}^n$ is called an **ellipsoid** with center z. If $D \in \mathbb{R}^{n \times n}$ is non-singular and $b \in \mathbb{R}^n$, then the mapping $S : \mathbb{R}^n \to \mathbb{R}^n$ given by $S(x) = Dx + b$ is called an **affine transformation**. We further write $S(L)$ for the **image** of $L \subseteq \mathbb{R}^n$ under S, i.e. $S(L) = \{y \in \mathbb{R}^n : y = S(x) \text{ for some } x \in \mathbb{R}^n\}$. The **volume** of a set $L \subseteq \mathbb{R}^n$ is finally defined as $\mathrm{Vol}(L) = \int_{x \in L} dx$.

Let $P = \{x \in \mathbb{R}^n : Ax \geq b\}$ for some $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^m$. To decide whether P is non-empty, the ellipsoid method generates a sequence $\{E_t\}$ of ellipsoids E_t with centers x_t. If $x_t \in P$, then P is non-empty and the method stops. If $x_t \notin P$, then one of the constraints is violated, i.e. there exists a row j of A such that $a_j^T x_t < b_j$. Therefore, P is contained in the half-space $\{x \in \mathbb{R}^n : a_j^T x \geq a_j^T x_t\}$, and in particular in the intersection of this half-space with E_t, which we will call a **half-ellipsoid**.

The following is the key result underlying the ellipsoid method. It states that there exists a new ellipsoid E_{t+1} that contains the half-ellipsoid and whose volume is only a fraction of the volume of E_t. This situation is illustrated in Figure 6.

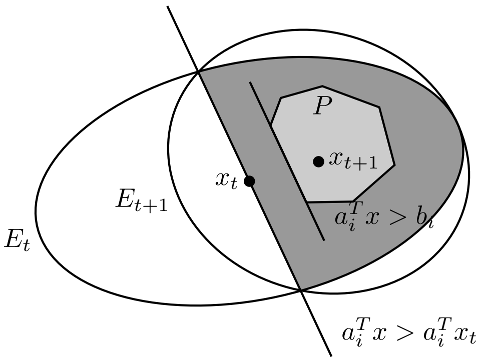


Figure 6: A step of the ellipsoid method where $x_t \notin P$ but $x_{t+1} \in P$. The polytope P and the half-ellipsoid that contains it are shaded.

Theorem 6.2. *Let $E = E(z, D)$ be an ellipsoid in $\mathbb{R}^n$ and $a \in \mathbb{R}^n$ non-zero. Consider the half-space $H = \{x \in \mathbb{R}^n : a^T x \geq a^T z\}$, and let*

$$z' = z + \frac{1}{n+1} \frac{Da}{\sqrt{a^T D a}},$$
$$D' = \frac{n^2}{n^2 - 1} \left(D - \frac{2}{n+1} \frac{D a a^T D}{a^T D a} \right).$$

Then D' is symmetric and positive definite, and therefore $E' = E(z', D')$ is an ellipsoid. Moreover, $E \cap H \subseteq E'$ and $\mathrm{Vol}(E') < e^{-1/(2(n+1))} \mathrm{Vol}(E)$.

If the procedure is repeated, it will either find a point in P or generate smaller and smaller ellipsoids containing P. In the next lecture, this procedure will be turned into an algorithm by observing that the volume of P must either be zero or larger than a certain threshold that depends on the size of the description of P.

We now sketch the proof of Theorem 6.2. We use the following lemma about affine transformations, which is not hard to prove.

Lemma 6.3. *Let $S : \mathbb{R}^n \to \mathbb{R}^n$ be an affine transformation given by $S(x) = Dx + b$ and let $L \subseteq \mathbb{R}^n$. Then, $\mathrm{Vol}(S(L)) = |\det(D)|\mathrm{Vol}(L)$.*

Proof sketch of Theorem 6.2. We prove the theorem for $E = \{x \in \mathbb{R}^n : x^T x \leq 1\}$ and $H = \{x \in \mathbb{R}^n : x_1 \geq 0\}$. Since every pair of an ellipsoid and a hyperplane as in the statement of the theorem can be obtained from E and H via some affine transformation, the general case then follows by observing that affine transformations preserve inclusion and, by Lemma 6.3, relative volume of sets.

Let $e_1 = (1, 0, \ldots, 0)^T$. Then,

$$\begin{aligned} E' &= E\left(\frac{e_1}{n+1}, \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1}e_1 e_1^T\right)\right) \\ &= \left\{x \in \mathbb{R}^n : \frac{n^2-1}{n^2}\sum_{i=1}^n x_i^2 + \frac{1}{n^2} + \frac{2(n+1)}{n^2}x_1(x_1 - 1) \leq 1\right\}. \end{aligned}$$

Consider an arbitrary $x \in E \cap H$, and observe that $0 \leq x_1 \leq 1$ and $\sum_{i=1}^n x_i^2 \leq 1$. It is easily verified that $x \in E'$ and thus $E \cap H \subseteq E'$.

Now consider the affine transformation $F : \mathbb{R}^n \to \mathbb{R}^n$ given by

$$F(x) = \frac{e_1}{n+1} + \left(\frac{n^2}{n^2-1}\left(I - \frac{2}{n+1}e_1 e_1^T\right)\right)^{\frac{1}{2}} x.$$

It is not hard to show that $E' = F(E)$. Therefore, by Lemma 6.3,

$$\begin{aligned} \frac{\mathrm{Vol}(E')}{\mathrm{Vol}(E)} &= \sqrt{\det\left(\frac{n^2}{n^2-1}\left(I - \frac{2}{n+1}e_1 e_1^T\right)\right)} \\ &= \left(\frac{n^2}{n^2-1}\right)^{\frac{n}{2}}\left(1 - \frac{2}{n+1}\right)^{\frac{1}{2}} = \frac{n}{n+1}\left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} \\ &= \left(1 - \frac{1}{n+1}\right)\left(1 + \frac{1}{n^2-1}\right)^{\frac{n-1}{2}} < e^{-\frac{1}{n+1}}\left(e^{\frac{1}{n^2-1}}\right)^{\frac{n-1}{2}} = e^{-\frac{1}{2(n+1)}}, \end{aligned}$$

where the strict inequality follows by using twice that $1 + a < e^a$ for all $a \neq 0$. □

A more detailed description of the ellipsoid method and an overview of the proof of correctness will be given in the next lecture.

7 Ellipsoid Method

7.1 Ellipsoid method

Consider a polytope $P = \{x \in \mathbb{R}^n : Ax \geq b\}$, given by a matrix $A \in \mathbb{Z}^{m\times n}$ and a vector $b \in \mathbb{Z}^m$. Assume for now that P is bounded and either empty or full-dimensional. Here, P is called **full-dimensional** if $\text{Vol}(P) > 0$. The ellipsoid method takes the following steps to decide whether P is non-empty:

1. Let U be the largest absolute value among the entries of A and b, and define
   $x_0 = 0, \quad D_0 = n(nU)^{2n} I, \quad E_0 = E(x_0, D_0),$
   $V = (2\sqrt{n})^n (nU)^{n^2}, \quad v = n^{-n}(nU)^{-n^2(n+1)},$
   $t^* = \lceil 2(n+1)\log(V/v)\rceil.$
2. For $t = 0, \dots, t^*$, do the following:
   1. If $t = t^*$ then stop; P is empty.
   2. If $x_t \in P$ then stop; P is non-empty.
   3. Find a violated constraint, i.e. a row j such that $a_j^T x_t < b_j$.
   4. Let $E_{t+1} = E(x_{t+1}, D_{t+1})$ with

$$x_{t+1} = x_t + \frac{1}{n+1}\frac{D_t a_j}{\sqrt{a_j^T D_t a_j}},$$

$$D_{t+1} = \frac{n^2}{n^2-1}\left(D_t - \frac{2}{n+1}\frac{D_t a_j a_j^T D_t}{a_j^T D_t a_j}\right).$$

The ellipsoid method is a so-called **interior point method**, because it traverses the interior of the feasible set rather than following its boundary.

7.2 Proof of correctness

Observe that E_0 is a ball centered at the origin. Given Theorem 6.2, and assuming that (i) $P \subseteq E_0$ and $\text{Vol}(E_0) < V$ and that (ii) P is either empty or $\text{Vol}(P) > v$, correctness of the ellipsoid method is easy to see: it either finds a point in P, thereby proving that P is non-empty, or an ellipsoid $E_{t^*} \supseteq P$ with $\text{Vol}(E_{t^*}) < e^{-t^*/2(n+1)}\text{Vol}(E_0) < (v/V)\text{Vol}(E_0) < v$, in which case P must be empty.

We now show that the above assumptions hold, starting with the inclusion of P in E_0 and the volume of E_0. We use the following lemma.

Lemma 7.1. *Suppose $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{R}^m$ and $m \geq n$. Let U be the largest absolute value among the entries of A and b. Then every extreme point x of the polytope $P = \{x' \in \mathbb{R}^n : Ax' \geq b\}$ satisfies $-(nU)^n \leq x_i \leq (nU)^n$ for all $i = 1, \dots, n$.*

Proof. Any extreme point x can be written as $x = \hat{A}^{-1}\hat{b}$ for some invertible submatrix $\hat{A} \in \mathbb{Z}^{n\times n}$ of A and subvector $\hat{b} \in \mathbb{R}^n$ of b, corresponding to n linearly independent constraints that are active at x. By Cramer's rule,

$$x_i = \frac{\det \hat{A}^i}{\det \hat{A}},$$

where $\hat{A}^i$ is the matrix obtained by replacing the ith column of $\hat{A}$ by $\hat{b}$. Then, for $i = 1, \dots, n$,

$$\left|\det \hat{A}^i\right| = \left|\sum_{\sigma \in S_n} (-1)^{|\sigma|} \prod_{j=1}^{n} \hat{a}^i_{j,\sigma(j)}\right| \leq \sum_{\sigma\in S_n} \prod_{i=1}^{n} |\hat{a}^i_{j,\sigma(j)}| \leq n! U^n \leq (nU)^n,$$

where $|\sigma|$ is the number of inversions of permutation $\sigma \in S_n$, i.e. the number of pairs i, j such that $i < j$ and $\sigma(i) > \sigma(j)$. Moreover, $\det(\hat{A}) \neq 0$ since $\hat{A}$ is invertible, and $|\det(\hat{A})| \geq 1$ since all entries of A are integers. Therefore, $|x_i| \leq (nU)^n$ for all $i = 1, \dots, n$. □

If P is bounded and $x \in P$, then $\sum_i x_i^2 \leq n(nU)^{2n}$ and so $x \in E_0$. The ball E_0 is contained in a cube of volume $V = (2\sqrt{n})^n (nU)^{n^2}$, and thus $P \subseteq E_0$ and $\mathrm{Vol}(E_0) \leq V$.

We now turn to the lower bound on the volume of P in the case when it is non-empty.

Lemma 7.2. *Consider a full-dimensional and bounded polytope $P = \{x \in \mathbb{R}^n : Ax \geq b\}$, where $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$ and all entries have absolute value at most U. Then $\mathrm{Vol}(P) > n^{-n}(nU)^{-n^2(n+1)}$.*

Proof sketch. If P is full-dimensional and bounded and has at least one extreme point, it has $n+1$ extreme points $v^0, \dots, v^n$ that do not lie on a common hyperplane. Let Q be the **convex hull** of these extreme points, defined as

$$Q = \left\{ x \in \mathbb{R}^n : x = \sum_{k=0}^{n} \lambda_k v^k, \sum_{k=0}^{n} \lambda_k = 1, \lambda_k \geq 0 \right\}.$$

Clearly, $Q \subseteq P$ and thus $\mathrm{Vol}(Q) \leq \mathrm{Vol}(P)$. It can be shown that

$$\mathrm{Vol}(Q) = \frac{1}{n!}\left|\det\begin{pmatrix} 1 & \cdots & 1 \\ v^0 & \cdots & v^n \end{pmatrix}\right|.$$

The ith coordinate of v^k is a rational number p_i^k / q_i^k, and by the same argument as in the proof of Lemma 7.1, $|q_i^k| \leq (nU)^n$. Therefore,

$$\begin{aligned}\mathrm{Vol}(P) \geq \mathrm{Vol}(Q) &\geq \frac{1}{n!}\left|\frac{1}{\prod_{i=1}^{n}\prod_{k=0}^{n} q_i^k}\right| \\ &> \frac{1}{n^n}\frac{1}{\prod_{i=1}^{n}\prod_{k=0}^{n}(nU)^n} = n^{-n}(nU)^{-n^2(n+1)}.\end{aligned}$$

□

So far we have assumed that the polytope P is bounded and full-dimensional. We finally lift these assumptions.

Polytopes that are unbouned By Lemma 7.1, all extreme points of P lie in the set $P_B = \{x \in P : |x_i| \leq (nU)^n \text{ for all } i = 1, \dots, n\}$. Moreover, P is non-empty if and only if it has an extreme point. We can therefore test for non-emptiness of P_B instead of P, and P_B is a bounded polytope.

Polytopes that are not full dimensional For a polytope P that is not full-dimensional, it is not the case that $\mathrm{Vol}(P) < v$ implies $P = \emptyset$, and the ellipsoid method can fail. The following result shows, however, that we can slightly perturb P to obtain a polytope that is either empty or full-dimensional.

Lemma 7.3. *Let $P = \{x \in \mathbb{R}^n : Ax \geq b\}$, where $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$ and all entries have absolute value at most U. Let*

$$P_\epsilon = \{\, x \in \mathbb{R}^n : Ax \geq b - \epsilon e\}$$

where

$$\epsilon = \frac{1}{2(n+1)} \left((n+1)U\right)^{-(n+1)}$$

and $e^T = (1, \dots, 1)$. Then, $P_\epsilon = \emptyset$ if and only if $P = \emptyset$, and either $P_\epsilon = \emptyset$ or $\mathrm{Vol}(P) > 0$.

Proof. We first show that emptiness of P implies emptiness of P_ϵ. If P is empty, then the linear program $\min\{0^T x : Ax \geq b\}$ is infeasible. Therefore its dual, which is $\max\{\lambda^T b : \lambda^T A = 0^T, \lambda \geq 0\}$, must be infeasible or unbounded; since it is feasible it must be unbounded. There thus has to exist a basic feasible solution λ to the $n+1$ constraints $\lambda^T A = 0^T$, $\lambda^T b = 1$, and $\lambda \geq 0$, and, by Lemma 7.1, $\lambda_i \leq ((n+1)U)^{n+1}$ for all i. Since λ is a BFS, at most $n+1$ of its components are non-zero, and therefore $\sum_i^m \lambda_i \leq (n+1)\left((n+1)U\right)^{n+1}$. Then $\lambda^T(b - \epsilon e) = \lambda^T b - \epsilon \sum_{i=1}^m \lambda_i \geq \lambda^T b - \frac{1}{2} > 0$. This means that the dual remains unbounded, and the primal infeasible, if we replace b by $b - \epsilon e$, and thus $P_\epsilon = \emptyset$.

It remains to be shown that P_ϵ is full-dimensional if P is non-empty. For this, consider $x \in P$ and let

$$Y = \left\{ y \in \mathbb{R}^n : x_i - \frac{\epsilon}{nU} \leq y_i \leq x_i + \frac{\epsilon}{nU} \text{ for all } i = 1, \dots, n \right\}.$$

It is easy to see that $\mathrm{Vol}(Y) = (2\epsilon/(nU))^n > 0$ and that $Y \subseteq P_\epsilon$. Thus P_ϵ must be full-dimensional. □

The general case of polytopes P that potentially are unbounded and not full-dimensional can thus be handled by computing the bounded polytope P_B, perturbing it, and then running the ellipsoid method on the resulting polytope.

7.3 The running time of the ellipsoid method

For a bounded and full-dimensional polytope P given by a matrix A and vector b with integer entries bounded by U, the ellipsoid method decides whether P is empty or not in $O(n \log(V/v)) = O(n^4 \log(nU))$ iterations. It can further be shown that $O(n^6 \log(nU))$ iterations suffice even when P might be unbounded or not full-dimensional.

For the ellipsoid method to have a polynomial running time, however, the number of operations in each iteration also has to be bounded by a polynomial function of n and $\log U$. A potential problem is that the computation of the new ellipsoid involves taking a square root. This means that in general calculations cannot be done exactly, and intermediate results have to be stored with sufficiently many bits to ensure that errors don't accumulate. It turns out that the algorithm can be made to work, with the same asymptotic number of iterations as above, when only $O(n^3 \log U)$ bits are used for each intermediate value. The proof of this result is very technical.

The ellipsoid method has high theoretical significance, because it provided the first polynomial-time algorithm for linear programming and can also be applied to larger classes of convex optimization problems. In practice, however, both the simplex method and a different interior point method, **Karmarkar's algorithm**, tend to be much faster. The latter also has a better worst-case performance than the ellipsoid method.

8 Optimization in Networks

8.1 Graph terminology

A directed **graph**, or **network**, $G = (V, E)$ consists of a set V of **vertices** and a set $E \subseteq V \times V$ of **edges**. When the relation E is symmetric, G is called an undirected graph, and we can write edges as unordered pairs $\{i, j\} \in E$ for $i, j \in V$. The **degree** of vertex $i \in V$ in graph G is the number $|\{j \in V : (i, j) \in E \text{ or } (j, i) \in E\}|$ of other vertices connected to it by an edge. A **walk** from $u \in V$ to $w \in V$ is a sequence of vertices $v_1, \ldots, v_k \in V$ such that $v_1 = u$, $v_k = w$, and $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, k-1$. In a directed graph, we can also consider an undirected walk where $(v_i, v_{i+1}) \in E$ or $(v_{i+1}, v_i) \in E$ for $i = 1, \ldots, k-1$. A walk is a **path** if $v_1, \ldots, v_k$ are pairwise distinct, and a **cycle** if furthermore $v_1 = v_k$. A graph that does not contain any cycles is called **acyclic**. A graph is called **connected** if for every pair of vertices $u, v \in V$ there is an undirected path from u to v. A **tree** is a graph that is connected and acyclic. A graph $G' = (V', E')$ is a subgraph of graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. In the special case where G' is a tree and $V' = V$, it is called a **spanning tree** of G.

8.2 Minimum cost flow problem

Consider a network $G = (V, E)$ with $|V| = n$, and let $b \in \mathbb{R}^n$. Here, b_i denotes the amount of flow that enters or leaves the network at vertex $i \in V$. If $b_i > 0$, we say that i is a **source** supplying b_i units of flow. If $b_i < 0$, we say that i is a **sink** with a demand of $|b_i|$ units of flow. Further let $C, \underline{M}, \overline{M} \in \mathbb{R}^{n \times n}$, where c_{ij} denotes the **cost** per unit of flow on edge $(i, j) \in E$, and $\underline{m}_{ij}$ and $\overline{m}_{ij}$ respectively denote lower and upper bounds on the flow across this edge. The **minimum cost flow problem** then asks for flows x_{ij} that conserve the flow at each vertex, respect the upper and lower bounds, and minimize the overall cost. Formally, $x \in \mathbb{R}^{n \times n}$ is a **minimum cost flow** of G if it is an optimal solution of the following optimization problem:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{(i,j)\in E} c_{ij} x_{ij} & \\ \text{subject to} & \displaystyle b_i + \sum_{j:(j,i)\in E} x_{ji} = \sum_{j:(i,j)\in E} x_{ij} & \text{for all } i \in V, \\ & \underline{m}_{ij} \leq x_{ij} \leq \overline{m}_{ij} & \text{for all } (i, j) \in E. \end{array}$$

Note that $\sum_{i \in V} b_i = 0$ is required for any feasible flows to exist, and we make this assumption in the following. We further assume without loss of generality that the network G is connected. Otherwise the problem can be decomposed into several smaller problems that can be solved independently. An important special case is that of **uncapacitated flows**, where $\underline{m}_{ij} = 0$ and $\overline{m}_{ij} = \infty$ for all $(i, j) \in E$.

The minimum cost flow problem is a linear programming problem, with constraints of the form $Ax = b$ where

$$a_{ik} = \begin{cases} 1 & k\text{th edge starts at vertex } i, \\ -1 & k\text{th edge ends at vertex } i, \\ 0 & \text{otherwise.} \end{cases}$$

Given this rather simple structure, we may hope that minimum cost flow problems are easier to solve than general linear programs. Indeed, we will see that basic feasible solutions of a minimum cost flow problem take a special form, and will obtain an algorithm that exploits this form.

8.3 Spanning tree solutions

Consider a minimum cost flow problem for a connected network $G = (V, E)$. A solution x is called a **spanning tree solution** if the edges of G can be partitioned into three disjoint sets T, L, U, such that (V, T) is a spanning tree, and $x_{ij} = \underline{m}_{ij}$ if $(i, j) \in L$ and $x_{ij} = \overline{m}_{ij}$ if $(i, j) \in U$. For every choice of T, L and U, the flow conservation constraints uniquely determine the values x_{ij} for $(i, j) \in T$. An example is in Figure 7.

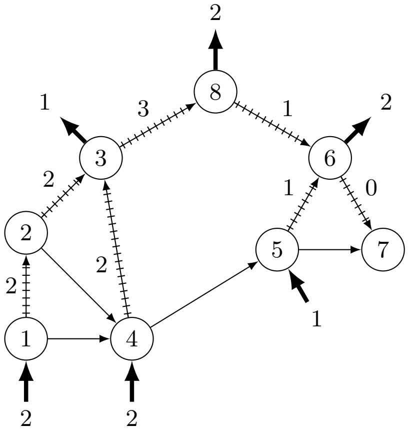


Figure 7: A flow network with a basic feasible solution, which is the spanning tree T indicated by hatched edges. This network is uncapacitated so $L = E \setminus T$ and $U = \emptyset$, and flows are zero for edges not in T. Flows for the edges in T can be determined inductively starting from the leafs. In this example, the spanning tree corresponds to a degenerate BFS, because edge $(6, 7)$ carries flow zero.

It is not hard to show that the basic solutions of a minimum cost flow problem are precisely its spanning tree solutions.

Theorem 8.1. *A flow vector is a basic solution of a minimum cost flow problem if and only if it is a spanning tree solution.*

8.4 The network simplex method

We now derive a variant of the simplex method, the **network simplex method**, that works directly with spanning tree solutions. This method maintains a feasible solution for the primal and a corresponding dual solution, but unlike the simplex method does not guarantee that these two solutions satisfy complementary slackness. Rather, it uses a separate condition to either establish both dual feasibility and complementary slackness, and thus optimality, or identify a new spanning tree solution.

The Lagrangian of the minimum cost flow problem is

$$\begin{aligned} L(x,\lambda) &= \sum_{(i,j)\in E} c_{ij}x_{ij} - \sum_{i\in V}\lambda_i\Bigg(\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} - b_i\Bigg) \\ &= \sum_{(i,j)\in E}(c_{ij}-\lambda_i+\lambda_j)x_{ij} + \sum_{i\in V}\lambda_i b_i \end{aligned} \tag{8.1}$$

Let $\bar{c}_{ij} = c_{ij} - \lambda_i + \lambda_j$ be the **reduced cost** of edge $(i,j)\in E$. Dual feasibility requires that $\bar{c}_{ij} \geq 0$ whenever $\overline{m}_{ij} = \infty$. Minimizing $L(x,\lambda)$ subject to the regional constraints $\underline{m}_{ij} \leq x_{ij} \leq \overline{m}_{ij}$ for $(i,j)\in E$ further yields the following complementary slackness conditions:

$$\begin{aligned} \bar{c}_{ij} > 0 &\quad \text{implies} \quad x_{ij} = \underline{m}_{ij}, \\ \bar{c}_{ij} < 0 &\quad \text{implies} \quad x_{ij} = \overline{m}_{ij}, \text{ and} \\ \underline{m}_{ij} < x_{ij} < \overline{m}_{ij} &\quad \text{implies} \quad \bar{c}_{ij} = 0. \end{aligned}$$

Assume that x is a basic feasible solution associated with sets T, U, and L. Then the system of equations

$$\lambda_{|V|} = 0, \qquad \lambda_i - \lambda_j = c_{ij} \quad \text{for all } (i,j)\in T$$

has a unique solution, which in turn allows us to compute $\bar{c}_{ij}$ for all edges $(i,j)\in E$. Note that $\bar{c}_{ij} = 0$ for all $(i,j)\in T$ by construction, so the third complementary slackness condition is always satisfied.

Pivoting

If $\bar{c}_{ij} \geq 0$ for all $(i,j)\in L$ and $\bar{c}_{ij} \leq 0$ for all $(i,j)\in U$, dual feasibility and the first two complementary slackness are satisfied as well, meaning that the solution is optimal (by appeal to the Lagrangian sufficiency theorem). Otherwise, consider an edge (i, j) that violates these conditions, and observe that this edge and the edges in T forms a unique cycle C. Since (i, j) is the only edge in C with non-zero reduced cost, we can decrease the objective by pushing flow along C to increase x_{ij} if $\bar{c}_{ij}$ is negative and decrease x_{ij} if $\bar{c}_{ij}$ is positive. Doing so will change the flow on all edges in C by the same amount, with the direction of the change depending on whether a specific edge is oriented in the same or the opposite direction as (i, j).

Let $\underline{B}, \overline{B} \subseteq C$ respectively denote the sets of edges whose flow is to decrease or increase, and let

$$\delta = \min\left\{ \min_{(k,\ell)\in\underline{B}} \{x_{k\ell} - \underline{m}_{k\ell}\}, \min_{(k,\ell)\in\overline{B}} \{\overline{m}_{k\ell} - x_{k\ell}\} \right\}.$$

be the maximum amount of flow that can be pushed along C. If $\delta = \infty$, the problem is unbounded. If $\delta = 0$ or if the minimum is attained for more than one edge, the problem is degenerate. Otherwise, pushing δ units of flow along C yields a unique edge $(k, \ell) \in C$ whose flow is either $\underline{m}_{k\ell}$ or $\overline{m}_{k\ell}$. If $(k, \ell) \in T$, we obtain a new BFS with spanning tree $(T \setminus \{(k, \ell)\}) \cup \{(i, j)\}$. If instead $(k, \ell) = (i, j)$, we obtain a new BFS where (i, j) has moved from U to L, or vice versa. An example of the pivoting step is given in Figure 8.

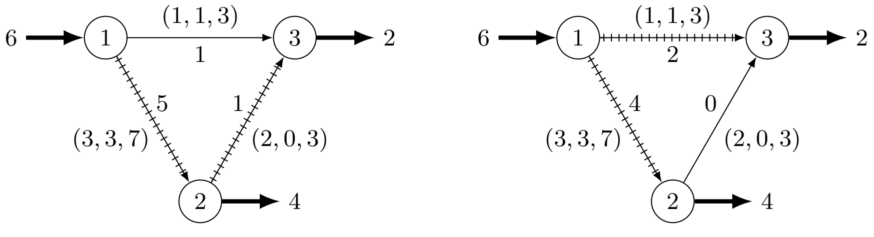


Figure 8: Flow network before and after a pivoting step. Edge (i, j) is labelled with the vector $(c_{ij}, \underline{m}_{ij}, \overline{m}_{ij})$ and the current flow x_{ij}, and spanning trees are indicated by hatched edges. In the situation shown on the left, we have $\lambda_3 = 0$, $\lambda_2 = c_{23} + \lambda_3 = 2$, and $\lambda_1 = c_{12} + \lambda_2 = 5$, and thus $\bar{c}_{13} = c_{13} - \lambda_1 + \lambda_3 = -4$. If we push one unit of flow around the cycle $1, 3, 2, 1$, the flow on $(2, 3)$ reaches the lower bound of $\underline{m}_{23} = 0$ and we obtain a new spanning tree with edges $(1, 2)$ and $(1, 3)$. The new situation is shown on the right. Now, $\lambda_3 = 0$, $\lambda_1 = c_{13} + \lambda_3 = 1$, and $\lambda_2 = \lambda_1 - c_{12} = -2$, and thus $\bar{c}_{23} = c_{23} - \lambda_2 + \lambda_3 = 4$. Since this is positive and $x_{23} = \underline{m}_{23}$, we have found an optimal solution.

In the absence of degeneracies the value of the objective function decreases in every iteration of the network simplex method, and an optimal solution or a certificate of unboundedness is found after a finite number of iterations. If a degenerate solution is encountered it will still be possible to identify a new spanning tree or even a new BFS, but extra care may be required to ensure convergence.

Finding an initial feasible spanning tree solution

Consider a minimum cost flow problem for a network (V, E) and assume without loss of generality that $\underline{m}_{ij} = 0$ for all $(i, j) \in E$. If this is not the case, we can instead consider the problem obtained by setting $\underline{m}_{ij}$ to zero, $\overline{m}_{ij}$ to $\overline{m}_{ij} - \underline{m}_{ij}$, and modifying b_i to $b_i - \underline{m}_{ij}$ and b_j to $b_j + \underline{m}_{ij}$. A solution with flows x_{ij} for the new problem then corresponds to a solution with flows $x_{ij} + \underline{m}_{ij}$ for the original problem.

We now modify the problem such that the set of optimal solutions remains the same, assuming that the problem was feasible, but an initial feasible spanning tree solution is easy to find. For this, we introduce a dummy vertex $d \notin V$ and uncapacitated dummy edges $E' = \{(i, d) : i \in V, b_i \geq 0\} \cup \{(d, i) : i \in V, b_i < 0\}$, each with cost of $\sum_{(i,j)\in E} c_{ij}$. It is easy to see that a dummy edge has positive flow in some optimal solution of the new problem if and only if the original problem is infeasible. Furthermore, a feasible spanning tree solution is now easily obtained by letting $T = E'$, $x_{id} = b_i$ for all $i \in V$ with $b_i > 0$, $x_{di} = -b_i$ for all $i \in V$ with $b_i < 0$, and $x_{ij} = 0$ otherwise.

8.5 Integrality of optimal solutions

Since the network simplex method does not require any divisions, any finite optimal solution it obtains for a problem with integer constants is also integral.

Theorem 8.2. *Consider a minimum cost flow problem that is feasible and bounded. If b_i is integral for all $i \in V$ and $\underline{m}_{ij}$ and $\overline{m}_{ij}$ are integral for all $(i, j) \in E$, then there exists an integral optimal solution. If c_{ij} is integral for all $(i, j) \in E$, then there exists an integral optimal solution to the dual.*

This theorem is important for the many practical problems in which an integer solution is required for a meaningful interpretation (for example, the assignment problems). Later, we investigate linear programming problems subject to the additional constraint that the solution be in integers. Such problems are usually much harder to solve than the problem without the integer constraint. However, for network flow problems we get integer solutions for free.

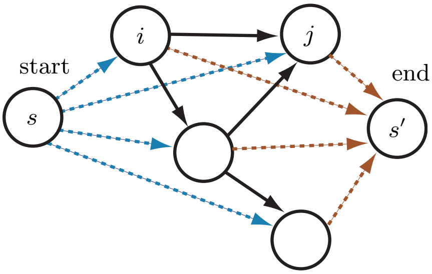
8.6 Longest path problem

The **critical path method** is an operational research technique dating from the late 1950s. We can study it as a minimum cost flow problem. It concerns a project consisting of activities, or jobs. The order in which jobs can be carried out is subject to precedence constraints, in that some jobs cannot start until other jobs are complete. Job i has a duration τ_i. What is the least time in which the project can be completed?

Consider a graph in which there is an edge (i, j) whenever job i must be completed before job j. Introduce two additional jobs, s and s', each of zero duration, to indicate the start and finish of the project, and introduce edges (s, i) and (i, s') for every job i. Denote this graph as $G = (V, E)$.

Suppose we start job i at time t_i. The problem of completing the project in minimum time is the linear program

$$\text{minimize } (t_{s'} - t_s)\,, \text{ subject to } t_j - t_i \geq \tau_i\,, \text{ for all } (i, j) \in E\,.$$



Let f_{ij} be a Lagrange multiplier for constraint $t_j - t_i \geq \tau_i$. The dual linear program is

$$\text{maximize } \sum_{(i,j)\in A} \tau_i f_{ij}$$

subject to

$$\sum_{j\colon (j,i)\in E} f_{ji} - \sum_{j\colon (i,j)\in E} f_{ij} = -b_i\,, \quad \text{for all } i\,, \text{ and } f_{ij} \geq 0\,, \text{ for all } (i, j) \in E\,,$$

where $b_s = 1$, $b_{s'} = -1$ and $b_i = 0$ for $i \neq s, s'$.

This is a minimum cost flow problem with $c_{ij} = -\tau_i$. The path of edges where $f_{ij} = 1$ is is called the **critical path**. It is the longest path from s to s' when edges (s, i) and (i, s') have lengths 0 and each other edge (i, j) has length τ_i.

9 Transportation and Assignment Problems

9.1 Transportation problem

In the **transportation problem** we are given a set of **suppliers** $S = \{1, \dots, n\}$, with supplier i producing s_i units of a good, and a set of **consumers** $C = \{1, \dots, m\}$, with consumer j demanding d_j of the good, such that $\sum_{i=1}^n s_i = \sum_{j=1}^m d_j$. The cost of transporting one unit of the good from supplier i to consumer j is c_{ij}, and the goal is to supply the demand while minimizing overall transportation cost. This can be formulated as an uncapacitated minimum cost flow problem on a **bipartite network**, i.e. a network $G = (S \cup C, E)$ with $E \subseteq S \times C$. The case of $E = \{(i,j) : i \in S,\ j \in C\}$ is known as the **Hitchcock transportation problem**:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{i=1}^{n} x_{ij} = d_j \quad \text{for all } j = 1, \dots, m \\
& \sum_{j=1}^{m} x_{ij} = s_i \quad \text{for all } i = 1, \dots, n \\
& x_{ij} \geq 0 \quad \text{for all } i, j.
\end{aligned}$$

It turns out that transportation problems already capture the full expressiveness of minimum cost flow problems. For that reason, new algorithms for computing are often first implemented and tested for transportation problems.

Theorem 9.1. *Every minimum cost flow problem with finite capacities or non-negative costs has an equivalent transportation problem.*

Proof. Consider a minimum cost flow problem on a network $G = (V, E)$ with supplies or demands b_i, capacities $\underline{m}_{ij}$ and $\overline{m}_{ij}$, and costs c_{ij}. When constructing an initial feasible tree solution in the previous lecture, we saw that we can assume without loss of generality that $\underline{m}_{ij} = 0$ for all i, j. We can further assume that all capacities are finite: if some edge has infinite capacity but costs are non-negative then setting the capacity of this edge to a large enough number, for example $\sum_{i \in V} |b_i|$, does not affect the optimal solution of the problem.

We now construct a transportation problem as follows. For every vertex $i \in V$, we add a sink vertex with demand $\sum_k \overline{m}_{ik} - b_i$. For every edge $(i,j) \in E$, we add a source vertex with supply $\overline{m}_{ij}$, an edge to vertex i with cost $c_{ij,i} = 0$, and an edge to vertex j with cost $c_{ij,j} = c_{ij}$. The situation is shown in Figure 9.

We now claim that there exists a direct correspondence between feasible flows of the two problems, and that these flows have the same costs. To see this, suppose $\{x_{ij}\}$ is

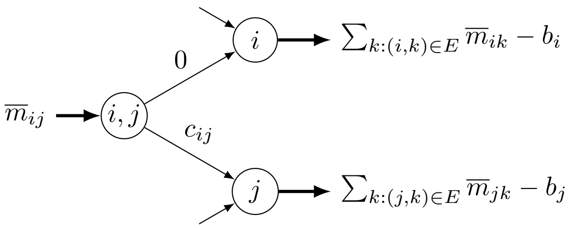

i
$\sum_{k:(i,k)\in E} \overline{m}_{ik} - b_i$
0
$\overline{m}_{ij}$
i,j
c_{ij}
j
$\sum_{k:(j,k)\in E} \overline{m}_{jk} - b_j$

Figure 9: Representation of flow conservation constraints by a transportation problem

a feasible flow in the minimum cost flow problem. Let the flows on edges (ij,i) and (ij,j) be $\overline{m}_{ij} - x_{ij}$ and x_{ij}, respectively. The total flow into vertex i then is $\sum_{k:(i,k)\in E}(\overline{m}_{ik} - x_{ik}) + \sum_{k:(k,i)\in E} x_{ki}$, which must be equal to $\sum_{k:(i,k)\in E} \overline{m}_{ik} - b_i$. This is the case if and only if $b_i + \sum_{k:(k,i)\in E} x_{ki} - \sum_{k:(i,k)\in E} x_{ik} = 0$, which is the flow conservation constraint for vertex i in the original problem. $\square$

9.2 Network simplex method in tableau form

When solving a transportation problem using the network simplex method, it is convenient to write it down in a tableau of the following form, where λ_i for $i = 1, \ldots, n$ and μ_j for $j = 1, \ldots, m$ are the dual variables corresponding to the flow conservation constraints for suppliers and consumers, respectively:

	μ_1	$\cdots$	μ_m	
λ_1	x_{11} $\boxed{c_{11}}$	$\cdots$ $\boxed{\cdots}$	x_{1m} $\boxed{c_{1m}}$	s_1
$\vdots$	$\vdots$ $\boxed{\vdots}$	$\ddots$ $\boxed{\ddots}$	$\vdots$ $\boxed{\vdots}$	$\vdots$
λ_n	x_{n1} $\boxed{c_{n1}}$	$\cdots$ $\boxed{\cdots}$	x_{nm} $\boxed{c_{nm}}$	s_n
	d_1	$\cdots$	d_m	

Consider the Hitchcock transportation problem given by the following tableau:

$\boxed{5}$	$\boxed{3}$	$\boxed{4}$	$\boxed{6}$	8
$\boxed{2}$	$\boxed{7}$	$\boxed{4}$	$\boxed{1}$	10
$\boxed{5}$	$\boxed{6}$	$\boxed{2}$	$\boxed{4}$	9
6	5	8	8	

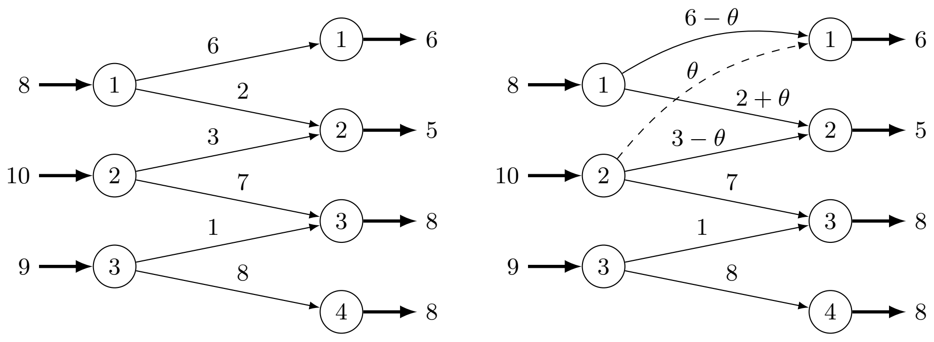


Figure 10: Initial basic feasible solution of a transportation problem (left) and a cycle along which the overall cost can be decreased (right)

An initial BFS can be found by a greedy method of iteratively considering pairs (i, j) of supplier i and customer j, increasing x_{ij} until either the supply s_i or the demand d_j is satisfied, and moving to the next supplier in the former case or to the next customer in the latter. Since $\sum_i s_i = \sum_j d_j$, this process is guaranteed to find a feasible solution, and the corresponding spanning tree consists of the pairs (i, j) that have been visited. If at some point both the supply and the demand are satisfied at the same time, the resulting solution is degenerate. In the example, we can start by setting $x_{11} = \min\{s_1, d_1\} = 6$, moving to customer 2 and setting $x_{12} = 2$, moving to supplier 2 and setting $x_{22} = 3$, and so forth. The resulting spanning tree and flows are shown on the left of Figure 10.

To determine the values of the dual variables λ_i for $i = 1, \dots, 3$ and μ_j for $j = 1, \dots, 4$, observe that $\lambda_i - \mu_j = c_{ij}$ must be satisfied for all $(i, j) \in T$. By setting $\lambda_1 = 0$, we obtain a system of 6 linear equalities with 6 variables, which has a unique solution. It will finally be convenient to write down $\lambda_i - \mu_j$ for $(i, j) \notin T$, which we do in the upper right corner of the respective cells. The tableau now looks as follows:

	-5	-3	0	-2	
0	$6\ \boxed{5}$	$2\ \boxed{3}$	$\overset{0}{\boxed{4}}$	$\overset{2}{\boxed{6}}$	8
4	$\overset{9}{\boxed{2}}$	$3\ \boxed{7}$	$7\ \boxed{4}$	$\overset{6}{\boxed{1}}$	10
2	$\overset{7}{\boxed{5}}$	$\overset{5}{\boxed{6}}$	$1\ \boxed{2}$	$8\ \boxed{4}$	9
	6	5	8	8	

If $c_{ij} \geq \lambda_i - \mu_j$ for all $(i, j) \notin T$, the current flow would be optimal. In our example this condition is violated, for example, for $i = 2$ and $j = 1$. Edge $(2, 1)$ forms a unique cycle with the spanning tree T, and we would like to increase x_{21} by pushing flow along this cycle. Due to the special structure of the network, doing so will alternately increase and decrease the flow for edges along the cycle. In particular, increasing x_{21} by θ increases x_{12} and decreases x_{11} and x_{22} by the same amount. The is shown on the right of Figure 10. Increasing x_{21} by the maximum amount of $\theta = 3$ and re-computing the values of the dual variables λ_1 and μ_j, we obtain left hand tableau below.

Now, $c_{24} < \lambda_2 - \mu_4$, and we can increase x_{24} by 7 to obtain the right hand tableau below, which satisfies $c_{ij} \geq \lambda_i - \mu_j$ for all $(i, j) \notin T$ and therefore is optimal.

	-5	-3	-7	-9
0	3 $\boxed{5}$	5 $\boxed{3}$	7 $\boxed{4}$	9 $\boxed{6}$
-3	3 $\boxed{2}$	0 $\boxed{7}$	7 $\boxed{4}$	6 $\boxed{1}$
-5	0 $\boxed{5}$	-2 $\boxed{6}$	1 $\boxed{2}$	8 $\boxed{4}$

	-5	-3	-2	-4
0	3 $\boxed{5}$	5 $\boxed{3}$	2 $\boxed{4}$	4 $\boxed{6}$
-3	3 $\boxed{2}$	0 $\boxed{7}$	-1 $\boxed{4}$	7 $\boxed{1}$
0	5 $\boxed{5}$	3 $\boxed{6}$	8 $\boxed{2}$	1 $\boxed{4}$

9.3 Assignment problem

An instance of the assignment problem is given by n agents and n jobs, and costs c_{ij} for assigning job j to agent i. The goal is to assign exactly one job to each agent to

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{subject to} \quad & x_{ij} \in \{0,1\} \quad \text{for all } i, j = 1, \dots, n \\
& \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i = 1, \dots, n \\
& \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } j = 1, \dots, n
\end{aligned}
\tag{9.1}
$$

Except for the integrality constraints, this is a special case of the Hitchcock transportation problem. All basic solutions of the **LP relaxation** of this problem, which is obtained by replacing the integrality constraint $x_{ij} \in \{0, 1\}$ by $0 \leq x_{ij} \leq 1$, are spanning tree solutions and therefore integral. Thus, both the network simplex method and the general simplex method yield an optimal solution of the original problem when applied to the LP relaxation. This is not necessarily the case, for example, for the ellipsoid method.

This problem is also known as the **weighted bipartite matching problem**. In the next lecture we will look at a polynomial time algorithm for solving this problem. As a preliminary, we state the following lemma.

Lemma 9.2. *A feasible solution $\{x_{ij}\}$ to (9.1) is optimal if there exist $\{\lambda_i\}$, $\{\mu_j\}$ such that $\lambda_i - \mu_j \leq c_{ij}$ for all i, j, and $\lambda_i - \mu_j = c_{ij}$ if $x_{ij} = 1$.*

10 Maximum Flows and Perfect Matchings

10.1 Maximum flow problem

Consider a flow network (V, E) with a single source 1, a single sink n, and finite capacities $\overline{m}_{ij} = C_{ij}$ for all $(i, j) \in E$. We will also assume for convenience that $\underline{m}_{ij} = 0$ for all $(i, j) \in E$. The **maximum flow problem** then asks for the maximum amount of flow that can be sent from vertex 1 to vertex n, i.e. the goal is to

$$\begin{aligned} \text{maximize} \quad & \delta \\ \text{subject to} \quad & \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} \delta & \text{if } i = 1 \\ -\delta & \text{if } i = n \\ 0 & \text{otherwise} \end{cases} \\ & 0 \leq x_{ij} \leq C_{ij} \quad \text{for all } (i,j) \in E. \end{aligned} \tag{10.1}$$

To see that this is again a special case of the minimum cost flow problem, set $c_{ij} = 0$ for all $(i, j) \in E$, and add an additional edge $(n, 1)$ with infinite capacity and cost $c_{n1} = -1$. Since the new edge $(n, 1)$ has infinite capacity, any feasible flow of the original network is also feasible for the new network. Cost is clearly minimized by maximizing the flow across the edge $(n, 1)$, which by the flow conservation constraints for vertices 1 and n maximizes flow through the original network. This is called a **circulation problem**, because there are no sources or sinks but flow merely circulates in the network.

10.2 Max-flow min-cut theorem

Consider a flow network $G = (V, E)$ with capacities C_{ij} for all $(i, j) \in E$. A **cut** of G is a partition of V into two sets, and the capacity of a cut is defined as the sum of capacities of all edges across the partition. Formally, for $S \subseteq V$, the capacity of the cut $(S, V \setminus S)$ is

$$C(S) = \sum_{(i,j)\in E\cap(S\times(V\setminus S))} C_{ij}. \tag{10.2}$$

Assume that x is a feasible flow vector that sends δ units of flow from vertex 1 to vertex n. It is easy to see that δ is bounded from above by the capacity of any cut S with $1 \in S$ and $n \in V \setminus S$. Indeed, for $X, Y \subseteq V$, let

$$f(X, Y) = \sum_{(i,j)\in E\cap(X\times Y)} x_{ij}.$$

Then, for any $S \subseteq V$ with $1 \in S$ and $n \in V \setminus S$,

$$\delta = \sum_{i\in S} \left(\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} \right) \tag{10.3}$$

$$\begin{aligned}\delta &= f(S,V) - f(V,S) \\ &= f(S,S) + f(S,V\setminus S) - f(V\setminus S,S) - f(S,S) \\ &= f(S,V\setminus S) - f(V\setminus S,S) \leq f(S,V\setminus S) \leq C(S).\end{aligned} \tag{10.4}$$

The following result states that this upper bound is tight, i.e. there exists a flow of size equal to the minimum capacity of a cut that separates vertex 1 from vertex n.

Theorem 10.1 (Max-flow min-cut theorem). *Let δ be the optimal solution of* (10.1) *for a network (V,E) with capacities C_{ij} for all $(i,j) \in E$. Then,*

$$\delta = \min\{\, C(S) : S \subseteq V, 1 \in S, n \in V \setminus S \,\}.$$

Proof. It remains to be shown that there exists a cut that separates vertex 1 from vertex n and has capacity equal to δ. Consider a feasible flow vector x. A path $P = v_0, v_1, \ldots, v_k$ is called an **augmenting path** for x if $x_{v_{i-1}v_i} < C_{v_{i-1}v_i}$ or $x_{v_iv_{i-1}} > 0$ for every $i = 1, \ldots, k$. If there exists an augmenting path from vertex 1 to vertex n, then we can push flow along the path, by increasing the flow on every forward edge and decreasing the flow on every backward edge along the path by the same amount, such that all constraints remain satisfied and the amount of flow from 1 to n increases.

Now assume that x is optimal, and let

$$S = \{1\} \cup \{\, i \in V : \text{there exists an augmenting path for } x \text{ from 1 to } i \,\}.$$

By optimality of x, $n \in V \setminus S$. Moreover,

$$\delta = f(S, V\setminus S) - f(V\setminus S, S) = f(S, V\setminus S) = C(S).$$

The first equality holds by (10.4). The second equality holds because $x_{ji} = 0$ for every $(j,i) \in E \cap ((V\setminus S) \times S)$. The third equality holds because $x_{ij} = C_{ij}$ for every $(i,j) \in E \cap (S \times (V \setminus S))$. □

10.3 The Ford-Fulkerson algorithm

The **Ford-Fulkerson algorithm** attempts to find a maximum flow by repeatedly pushing flow along an augmenting path, until such a path can no longer be found:

1. Start with a feasible flow vector x.
2. If there is no augmenting path from 1 to n, then stop.
3. Otherwise pick some augmenting path from 1 to n, and push a maximum amount of flow along this path without violating any constraints. Then go to Step 2..

Assume that all capacities are integral and that we start with an integral flow vector, e.g., the flow vector x such that $x_{ij} = 0$ for all $(i,j) \in E$. It is then not hard to see that the flow vector always remains integral and overall flow increases by at least one unit in each iteration. The algorithm is therefore guaranteed to find a maximum flow after finitely many iterations. It can in fact be shown that the running time is $O(|E| \cdot \delta)$ where δ is the value of the maximum flow. If all capacities are bounded by the integer U then the running time is $O(|E| \cdot |V| \cdot U)$. This is because the flow leaving s is at most $|V| \cdot U$ and each iteration takes $O(|E|)$ time.

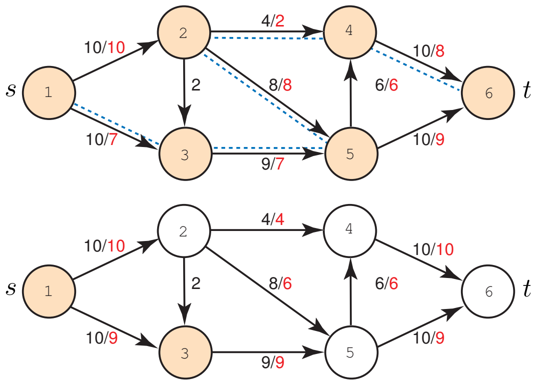


Figure 11: In the first picture the flow is suboptimal, at 17. There is an augmenting path s–3–5–2–4–t. After increasing flow along this path by 2, we reach the optimal flow assignment. Now there is no augmenting path from s to t. The maximum flow is 19, and $S = \{1, 3\}$. On edges are marked C_{ij}/x_{ij}.

10.4 Applications of the max-flow min-cut theorem

The max-flow min-cut theorem has many applications. Here are two.

In what follows we will work with a bipartite graph whose vertices are in two sets L (for left) and R (for right), and $E \subseteq L \times R$. Suppose $|L| = |R| = n$. A **matching** is a subset of edges sharing no common vertices. The bipartite graph is said to have a **perfect matching** if the size of the maximum matching is n, i.e. every $i \in L$ can be matched with some $j \in R$. For $i \in L$. define $N(i) = \{j : (i, j) \in E\}$ and for $X \subseteq L$, $N(X) = \cup_{i \in X} N(i)$.

Theorem 10.2 (Hall's theorem)**.** *Consider a bipartite graph $G = (L \cup R, E)$ with $|L| = |R|$. It has a perfect matching if and only if $|N(X)| \geq |X|$ for every $X \subseteq L$.*

Proof. If a perfect matching exists then clearly $|N(X)| \geq |X|$ for any $X \subseteq L$.

To prove the converse we add two vertices: a 'start' s, and a 'terminus' t, and all edges of form (s, i), $i \in L$, and (j, t), $j \in R$. We give the original edges in E capacity ∞ and the new edges capacity 1. Let $(S, V \setminus S)$ be a cut of minimum capacity. By considering how S is constructed we see that $N(L \cap S) \subseteq R \cap S$. The capacity of the cut comes from edges in $s \times (L \setminus S)$ and $(R \cap S) \times t$. The total number of these edges is $C(S)$. If a perfect matching does not exist, so $C(S) < |L|$, then

$$|N(L \cap S)| \leq |R \cap S| = C(S) - |L \setminus S| < |L| - |L \setminus S| = |L \cap S| \qquad \square$$

A subset of the vertices is said to be a **vertex cover** of G if ever edge has at least one endpoint in the subset.

Theorem 10.3 (König's theorem)**.** *In a bipartite graph $G = (L \cup R, E)$, the maximum size of a matching is equal to the minimum size of a vertex cover.*

Proof. We use the same graph as in the proof of Hall's theorem. Find the maximum flow using the Ford-Fulkerson algorithm. When it terminates there will be paths from s to t, each carrying unit flow and sharing no vertices in E. This proves

$$\text{'min size vertex cover'} \geq \text{'max size matching'} = \text{'max flow'} = \text{'min cut capacity'}.$$

So it remains to show 'min cut capacity' $\geq$ 'min size vertex cover'.

Consider the cut set S when the Ford-Fulkerson algorithm terminates. This consists of the unmatched vertices in L and all other vertices that can be reached from these along augmenting paths. Let $W = (L \setminus S) \cup (R \cap S)$. Note that $C(S) = |W|$. We claim that every edge of E has an endpoint in W.

An edge whose left endpoint is not matched must have its right endpoint in $R \cap S$. An edge whose left end point is matched either is part of an augmenting path and so has its right endpoint in $R \cap S$, or is not part of an augmenting path, and so its left endpoint is in $L \setminus S$. $\square$

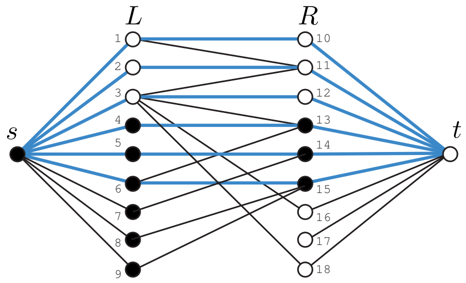


Figure 12: The graph used in proof of König's theorem. Edges (s,i) and (j,t) have capacity 1. Edges in $L \times R$ have capacity ∞. The thick blue edges carry flow 1 and provide a matching of size 6. The vertices in S (shown black) are those reachable from s by augmenting paths. For example, $4 \in S$ because of the augmenting path s–8–15–6–13–4. The six vertices: $\{1,2,3\} = L \cap (V \setminus S)$ and $\{13,14,15\} = R \cap S$ provide a vertex cover of the edges.

10.5 A polynomial-time algorithm for the assignment problem

Recall that in the assignment problem we are to find minimum cost matching in a weighted complete bipartite graph.

Consider the complete bipartite graph with $|L| = |R| = n$ and $E = L \times R$. The weight on edge (i, j) is c_{ij}. A labelling λ of the vertices is deemed (dual) feasible if $\lambda_i - \lambda_j \leq c_{ij}$ for all $(i, j) \in E$. Let G_λ be the subgraph of G consisting of edges where $\lambda_i - \lambda_j = c_{ij}$ and their endpoints, named the **equality graph**. See Figure 10.5

Step 1. If any vertex $i \in L$ is not in the equality graph we can increase λ_i until it becomes so. Similarly if any $j \in R$ is not in the equality graph we can decrease λ_j until it becomes so. Make these changes, so that all vertices are in the equality graph.

Look for an augmenting path in the equality graph by applying the Ford-Fulkerson algorithm to the same graph used in the proof of Hall's and König's theorems. That is, G_λ, with the addition of start vertex s, terminus vertex t, and connecting edges $\{s\} \times L$ and $R \times \{t\}$. Having found a path, augment flow by 1, thereby finding a matching of size one greater. Repeat this until a maximum matching in the equality graph is found.

If this matching is of size n we are done (by Lemma 9.2). If not, proceed to step 2.

Step 2. The Ford-Fulkerson algorithm has stopped with a set S, of vertices that can be reached from s along augmenting paths. Vertices in $V \setminus S$ cannot be so reached. Let us now increase by θ the label of each vertex in S, while keeping labels of vertices in $V \setminus S$ constant. Since step 1 did not create a matching of size n the set $R \cap (V \setminus S)$ is non-empty. We increase θ gradually from 0 until for some $i \in L \cap S$, and $j \in R \cap (V \setminus S)$ the value of $\lambda_i - \lambda_j$ moves from $< c_{ij}$ to $= c_{ij}$. This has the effect of creating a new equality graph, with new edge (i, j). Some edges may have been lost from the equality graph, but not so as remove any vertices from S (since all vertices in S can be reached along augmenting paths from s and the labels of vertices along augmenting paths have all increased by θ). At this point, either there is now an augmenting path to t and the cardinality of the matching increases, in which case we return to step 1, or the cardinality of $R \cap S$ increases, in which case we repeat step 2. The later can happen at most n times, so eventually we must find an augmenting path to t.

This is the so-called **Hungarian algorithm**. The total number times we find an augmenting path is no more than n (since the cardinality of the matching can increase at most n times). Each of these occurs after no more than n changes of the labels in step 2. Each computation of new labels takes time $O(n^2)$ (since we have to check which of the $\lambda_i - \lambda_j$ first increases to c_{ij}). Thus the entire algorithm has running time $O(n^4)$. There is an improved version of the Hungarian algorithm that runs in time $O(n^3)$.

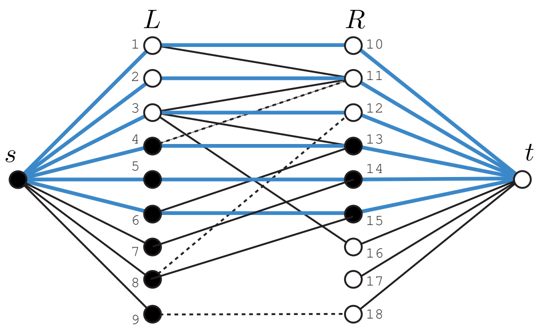


Figure 13: The maximum flow problem underlying the Hungarian algorithm. Edges (s, i) and (j, t) have capacity 1. Edges in $L \times R$ have capacity ∞. At a step 2 the equality graph contains the solid edges between L and R. Dotted edges are not in the equality tree. All other edges between L and R exist but are not in the equality tree and are not shown. There is a maximum matching of size 6, shown by the wide blue lines (1–10, 2–11, …, 6–15), each of which carries flow 1. Other edges are carrying flow 0. The vertices in S are black and the vertices in $V \setminus S$ are white. The set S is vertices reachable from s by edges having flow 0, or by backward flow from R to L along a full edge. As we increase the node numbers on S new edges will join the equality set, such as those shown dotted. This will either create an augmenting path and the cardinality of the matching in the equality tree increases, as in cases of 9–18 and 8–12. Otherwise, as in case of 4–11 the number of vertices in $R \cap S$ increases by 1 (with the addition of 11). Some edges, such as 3–13, may leave the equality tree, but this does not deplete S or invalidate the existing matching.

11 Shortest Paths and Minimum Spanning Trees

11.1 Bellman's equations

In the **single-destination shortest path problem** one is given a destination $t \in V$ and simultaneously looks for shortest paths from any vertex $i \in V \setminus \{t\}$ to t. It is equivalent to the minimum cost flow problem on the same network where one unit of flow is to be routed from each vertex $i \in V \setminus \{t\}$ to t, i.e. the one with supply $b_i = 1$ at every vertex $i \in V \setminus \{t\}$ and demand $b_t = -(|V| - 1)$ at vertex t.

Let λ_i for $i \in V$ be the dual solution corresponding to an optimal spanning tree solution of this flow problem, and recall that for every edge $(i,j) \in E$ with $x_{ij} > 0$,

$$\lambda_i = c_{ij} + \lambda_j.$$

By setting $\lambda_t = 0$ and adding these equalities along a path from i to t, we see that λ_i is equal to the length of a shortest path from i to t. Moreover, since $b_i = 1$ for all $i \in V \setminus \{t\}$, and given $\lambda_t = 0$, the dual problem is to

$$\text{maximize} \sum_{i \in V \setminus \{t\}} \lambda_i \quad \text{subject to } \lambda_i \leq c_{ij} + \lambda_j \text{ for all } (i,j) \in E.$$

In an optimal solution, λ_i will thus be as large as possible subject to the constraints, i.e. it will satisfy the so-called **Bellman equations**

$$\lambda_i = \min_{j:(i,j) \in E} \{c_{ij} + \lambda_j\} \quad \text{for all } i \in V \setminus \{t\},$$

with $\lambda_t = 0$. The intuition behind these equalities is that in order to find a shortest path from i to t, one should choose the first edge (i,j) on the path in order to minimize the sum of the length of this edge and that of a shortest path from j to t. This situation is illustrated in Figure 14.

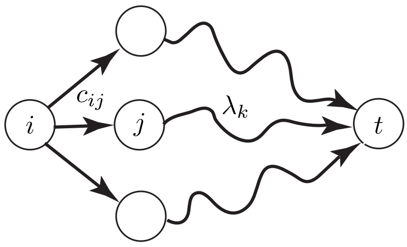


Figure 14: Illustration of the Bellman equations for the shortest path problem

11.2 Bellman-Ford algorithm

Let $\lambda_i(k)$ be the length of a shortest path from i to t that uses at most k edges. Then, $\lambda_t(k) = 0$ for all $k \geq 0$, and

$$\begin{aligned} &\lambda_i(0) = \infty \quad \text{and} \\ &\lambda_i(k) = \min_{j:(i,j)\in E} \{c_{ij} + \lambda_j(k-1)\} \end{aligned}$$

for all $i \in V \setminus \{t\}$ and $k \geq 1$.

The algorithm that successively computes $\lambda_i(k)$ for all i and larger and larger values of k is known as the **Bellman-Ford algorithm**. It is an example of a method called **dynamic programming**, which can be applied to problems that are decomposable into overlapping subproblems and have what is called optimal substructure, such that an overall solution can be constructed efficiently from solutions to the subproblems.

Note that $\lambda_i(|V|) < \lambda_i(|V|-1)$ for some $i \in V$ if and only if there exists a cycle of negative length. In that case $\lambda_i = -\infty$. Otherwise $\lambda_i = \lambda_i(|V|-1)$. So $O(|V|)$ iterations of the Bellman-Ford algorithm suffice to determine λ_i. Each iteration requires $O(|E|)$ steps, for an overall running time of $O(|E| \cdot |V|)$. Given the values λ_i for all $i \in V$, a shortest path from i to t then leads along an edge $(i,j) \in E$ such that $\lambda_i = c_{ij} + \lambda_j$.

11.3 Dijkstra's algorithm

The Bellman-Ford algorithm works even if some edges have negative length. If all edges have non-negative lengths then the running time can sometimes be decreased. The idea is to collect vertices in the order of increasing shortest path length to t. Assume $E = V \times V$, and set $c_{ij} = \infty$ if necessary. The following lemma will be useful.

Lemma 11.1. *Consider a graph with vertices V and edge lengths $c_{ij} \geq 0$ for all $i,j \in V$. Fix $t \in V$ and let λ_i denote the length of a shortest path from $i \in V$ to t. Let $j \in V \setminus \{t\}$ such that $c_{jt} = \min_{i \in V\setminus\{t\}} c_{it}$. Then, $\lambda_j = c_{jt}$ and $\lambda_j = \min_{i\in V\setminus\{t\}} \lambda_i$.*

Proof. For $i \neq t$, consider a shortest path from i to t, and let (ℓ, t) be the last edge on this path. Then, $\lambda_i \geq \lambda_\ell \geq c_{\ell t} \geq c_{jt}$. This holds in particular for $i = j$, and on the other hand $\lambda_j \leq c_{jt}$. Thus $\lambda_j = c_{jt} \leq \lambda_i$. $\square$

Dijkstra's algorithm uses this lemma to determine λ_j for a particular vertex j, removes j from the graph, and repeats the process for the new graph:

1. Find a vertex j with $c_{jt} = \min_{i\in V\setminus\{t\}} c_{it}$. Set $\lambda_j = c_{jt}$.
2. For every vertex $i \in V \setminus \{j\}$, set $c_{it} = \min\{c_{it}, c_{ij} + c_{jt}\}$.
3. Remove vertex j from V. If $|V| > 1$, return to Step 1.

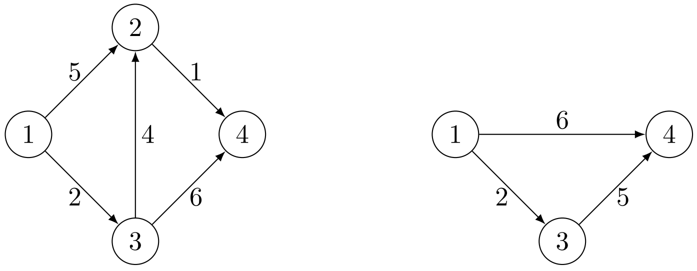


Figure 15: An iteration of Dijkstra's algorithm with $t = 4$. In the graph on the left, $c_{2t} = \min_{i \in V \setminus \{t\}} c_{it}$ and therefore, by Lemma 11.1, $\lambda_2 = c_{2t} = 1$. The graph on the right is then obtained by removing vertex 2 and updating c_{14} to $\min\{c_{14}, c_{12} + c_{24}\} = \min\{\infty, 5 + 1\} = 6$ and c_{34} to $\min\{c_{34}, c_{32} + c_{24}\} = \min\{6, 4 + 1\} = 5$.

The algorithm performs $|V| - 1$ iterations, each of which determines the new length of one edge for each of the remaining $O(|V|)$ vertices. The overall running time is thus $O(|V|^2)$. This improves on the Bellman-Ford algorithm in graphs with many edges, and is optimal in the sense that any algorithm for the single-destination shortest path problem has to inspect all of the edges, of which there are $\Omega(|V|^2)$ in the worst case.

One might wonder if there exists a way to transform the edge lengths to make them non-negative without affecting the structure of the shortest paths, so that Dijkstra's algorithm could be used in the presence of negative lengths as well. Let λ_i be the length of a shortest path from vertex i to vertex t, and recall that $\lambda_i \leq c_{ij} + \lambda_j$ for all $(i, j) \in E$. Let $\bar{c}_{ij} = c_{ij} + \lambda_j - \lambda_i$. Then, $\bar{c}_{ij} \geq 0$ for every edge $(i, j) \in E$, and for an arbitrary path $v_1, v_2, \ldots, v_k$,

$$\sum_{i=1}^{k-1} \bar{c}_{v_i v_{i+1}} = \sum_{i=1}^{k-1} (c_{v_i v_{i+1}} + \lambda_{v_{i+1}} - \lambda_{v_i}) = \lambda_{v_k} - \lambda_{v_1} + \sum_{i=1}^{k-1} c_{v_i v_{i+1}}.$$

So indeed, changing edge lengths from c_{ij} to $\bar{c}_{ij}$ allows Dijkstra's algorithm to work correctly, and it does not affect the structure of the shortest paths.

This observation is not very useful in the context of single-pair or single-destination shortest path problems: we do not know the values λ_i, and computing them is at least as hard as the problem we are trying to solve. For the **all-pairs shortest path problem**, however, which requires us to find a shortest path between every pair of vertices $i, j \in V$, the situation is different. The straightforward solution to this problem is to run the Bellman-Ford algorithm $|V|$ times, once for every possible destination vertex. In a graph with $\Omega(|V|^2)$ edges, this leads to an overall running time of $|V| \cdot O(|V|^3) = O(|V|^4)$. Using the above observation, we can instead invoke the Bellman-Ford algorithm for one destination vertex t to obtain the shortest path lengths λ_i for all $i \in V$, and compute shortest paths for the remaining destination vertices by running Dijkstra's algorithm on the graph with edge lengths $\bar{c}_{ij}$. This improves the asymptotic running time to $O(|V|^3) + |V - 1| \cdot O(|V|^2) = O(|V|^3)$.

11.4 Minimal spanning tree problem

The **minimum spanning tree problem** for a network (V, E) with associated costs c_{ij} for each edge $(i, j) \in E$ asks for a spanning tree of minimum cost, where the cost of a tree is the sum of costs of all its edges. This problem arises, for example, if one wishes to design a communication network that connects a given set of locations. The following property of minimum spanning trees will be useful.

Theorem 11.2. *Let (V, E) be a graph with edge costs c_{ij} for all $(i, j) \in E$. Let $U \subseteq V$ and $(u, v) \in U \times (V \setminus U)$ such that $c_{uv} = \min_{(i,j)\in U\times(V\setminus U)} c_{ij}$. Then there exists a spanning tree of minimum cost that contains (u, v).*

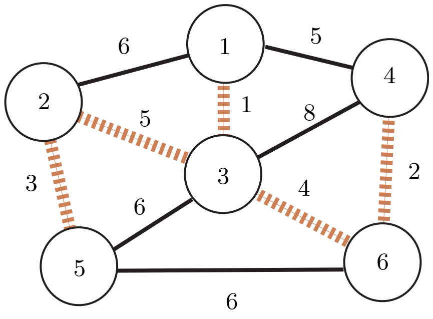
Proof. Let $T \subseteq E$ be a spanning tree of minimum cost. If $(u, v) \in T$ we are done. Otherwise, $T \cup \{(u, v)\}$ contains a cycle, and there must be another edge $(u', v') \in T$ such that $(u', v') \in U \times (V \setminus U)$. Then, $(T \cup \{(u, v)\}) \setminus \{(u', v')\}$ is a spanning tree, and its cost is no greater than that of T. □

Prim's algorithm uses this property to inductively construct a minimum spanning tree. It proceeds as follows:

1. Set $U = \{1\}$ and $T = \emptyset$.
2. If $U = V$, return T. Otherwise find an edge $(u, v) \in U \times (V \setminus U)$ such that $c_{uv} = \min_{(i,j)\in U\times(V\setminus U)} c_{ij}$.
3. Add v to U and (u, v) to T, and return to Step 2.

It is called a **greedy algorithm**, because it always chooses an edge of minimum cost.

Example. In this example, Prim's algorithm adds edges in the sequence $\{1, 3\}$, $\{3, 6\}$, $\{6, 4\}$, $\{3, 2\}$, $\{2, 5\}$.



After each iteration, we may compute and store for every $j \in V \setminus U$ a minimum cost edge to U. This only needs comparison between the previously stored edge and the edge to the vertex newly added to U. We then add to U the vertex that is closest to U. So each iteration needs time $O(|V|)$. The algorithm performs $|V| - 1$ iterations, so has overall running time of $O(|V|^2)$.

12 Semidefinite Programming

12.1 Primal-dual interior-point methods

The ellipsoid algorithm is not the most effective method of solving a linear program. We illustrate the **primal-dual path following method** for the LP and dual LP

$$\text{minimize } c^\top x \text{ s.t. } Ax = b, \quad x \geq 0,$$

$$\text{maximize } \lambda^\top b \text{ s.t. } \lambda^T A + s^T = c^T, \quad s \geq 0.$$

Constraints of the form $x_i \geq 0$ and $s_i \geq 0$ are problematic for Newton's method. We drop those, and consider new primal and dual objective functions, for $\mu > 0$,

$$c^\top x - \mu \textstyle\sum_i \log x_i \quad \text{and} \quad \lambda^\top b + \mu \textstyle\sum_j \log s_j.$$

The term of $-\mu \log x_i$ provides a so-called **barrier function** which prevents x_i from getting close to 0. Its influence decreases as μ tends to 0. By considering the Lagrangian it can be shown that x, λ, s are optimal for the modified primal and dual if

$$\begin{aligned} Ax &= b, \quad x \geq 0, \\ A^T\lambda + s &= c, \quad s \geq 0, \\ x_i s_i &= \mu \quad \text{for all } i = 1, \dots, n. \end{aligned} \tag{12.1}$$

Suppose that we have feasible solutions that satisfy (12.1). Then $c^T x - \lambda^T b = n\mu$. The key idea is to follow a path on which simultaneously we let $\mu \to 0$ and solve (12.1). At each iteration of the algorithm we use Newton's method to compute a solution $(x, \lambda, s)_k + (\delta x, \delta\lambda, \delta s)$ that satisfies (12.1). At iteration k we take $\mu = \mu_k$, where perhaps $\mu_k = (s^\top x)_{k-1}/(2n)$. It is possible to prove that such an algorithm decreases the duality gap μ_k from ϵ_0 to ϵ in a time that is $O(\sqrt{n}\log(\epsilon_0/\epsilon))$.

12.2 Semidefinite programming problem

The set $\{(x, s) : x \geq 0, s \geq 0\}$ is a convex cone. Interior point methods can also be effective when the regional constraint is some other convex cone.

Let $\mathbb{S}^n = \{A \in \mathbb{R}^{n\times n} : A^T = A\}$ be the set of all symmetric $n \times n$ matrices. The subset $\mathbb{S}^n_+$ is the set of positive semidefinite symmetric matrices, i.e. matices A such that $z^T A z \geq 0$ for all $z \in \mathbb{R}^n$. We write $A \succeq 0$. It is easily seen that $\mathbb{S}^n_+$ is a convex cone, i.e. $\alpha A + \beta B \in \mathbb{S}^n_+$ for all $\alpha, \beta \geq 0$ and all $A, B \in \mathbb{S}^n_+$.

A linear function of $X \in \mathbb{S}^n$ can be expressed in terms of the inner product

$$\operatorname{tr}(CX) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

for some $C \in \mathbb{S}^n$. A **semidefinite program** (SDP) has the form

$$\begin{aligned} &\text{minimize} && \operatorname{tr}(CX) \\ &\text{subject to} && \operatorname{tr}(A_i X) = b_i \quad \text{for all } i = 1, \dots, m \\ &&& X \succeq 0, \end{aligned} \tag{12.2}$$

where $C, A_1, \dots, A_m \in \mathbb{S}^n$ and $b \in \mathbb{R}^m$.

The linear program (2.2) is a special case of semidefinite programming:

$$\begin{aligned} &\text{minimize} && \langle \operatorname{diag}(c), \operatorname{diag}(x) \rangle \\ &\text{subject to} && \langle \operatorname{diag}(a_i), \operatorname{diag}(x) \rangle = b_i \quad \text{for all } i = 1, \dots, m \\ &&& \operatorname{diag}(x) \succeq 0, \end{aligned}$$

where $a_i = (a_{i1}, \dots, a_{in})^T$, for $i = 1, \dots, m$, is a vector consisting of the elements of the ith row of A. This problem can be brought into the form of (12.2) by replacing the diagonal matrix $\operatorname{diag}(x)$ by a general symmetric matrix X, and adding linear constraints to ensure that the off-diagonal entries of X are zero.

Semidefinite programming has been called *linear programming for the 21st century.*

SDPs can be viewed as having an infinite number of linear constraints on X, namely, $z^T X z \geq 0$ for all $z \in \mathbb{R}^n$. As a consequence, there are optimization problems that can be written as an SDP, but not as an LP.

Semidefinite programming includes important classes of convex optimization problems as special cases, for example linear programming and quadratically constrained quadratic programming.

While no algorithm is known for solving SDPs in a finite number of steps, they can be solved approximately in polynomial time, by a variant of the ellipsoid method, or a barrier method of the type described above. The constraint $X \succeq 0$ in an SDP can be handled by the barrier

$$-\mu \sum_{i=1}^{n} \log(\kappa_i(X)) = -\mu \log\left(\prod_{i=1}^{n} \kappa_i(X) \right) = -\mu \log(\det(X)),$$

where κ_i is the ith eigenvalue of X. This works because $X \succeq 0$ if and only if $\kappa_i \geq 0$ for all $i = 1, \dots, n$.

The Lagrangian dual of SDP is

$$\begin{aligned} \text{DSDP}: \ \underset{y}{\text{maximize}} \min_{X \succeq 0} \operatorname{tr}(CX) + \sum_i y_i (b_i - \operatorname{tr}(A_i X)) \\ = \underset{y}{\text{maximize}} \ y^\top b \quad \text{s.t. } C - \sum_i y_i A_i \succeq 0. \end{aligned}$$

This is because $\operatorname{tr}(C - \sum_i y_i A_i)X) > -\infty$ for all $X \succeq 0$ iff $Z = C - \sum_i y_i A_i \succeq 0$. The complementary slackness condition is $\operatorname{tr}(ZX) = 0$. This can also be used in a barrier method as was $x_i s_i = \mu$ in (12.1).

We now look at some application of semidefinite programming.

12.3 Max-cut problem

In the **max-cut problem** one is given a undirected graph $G = (V, E)$ and wishes to partition the n vertices into two sets, S and $V \setminus S$, so as to maximize the number of edges in $S \times (V \setminus S)$. The max-cut problem is NP-complete and thus cannot be solved exactly in polynomial time unless P = NP.

A maximization problem is said to lie in class APX (approximable) if there exists a constant $\alpha < 1$ and algorithm with polynomial running time (called an α-approximation algorithm) that will provide a solution guaranteed to differ from the optimum by no worse than a factor of α. A 0.87856-approximation algorithm for the max-cut problem can be obtained using semidefinite programming.

Theorem 12.1 (Goemans and Williamson, 1995)**.** *There exists a* 0.87856*-approximation algorithm for the max-cut problem.*

Proof sketch. The max-cut problem can be written as

$$\begin{aligned} &\text{maximize} && \sum_{\{i,j\}\in E} \frac{1 - x_i x_j}{2} \\ &\text{subject to} && x_i \in \{-1, 1\} \quad \text{for all } i \in V. \end{aligned} \tag{12.3}$$

By taking $x_i = 1$ for $i \in S$ and $x_j = -1$ for $j \in V \setminus S$ the objective function evaluates the number of edges in $S \times (V \setminus S)$.

Since the max-cut problem is NP-complete, an optimal solution of (12.3) cannot be found in polynomial time unless P = NP. Note, however, that

$$\sum_{\{i,j\}\in E} \frac{1 - x_i x_j}{2} = \frac{|E|}{2} - \frac{1}{4} x^T C x = \frac{|E|}{2} - \frac{1}{4} \operatorname{tr}(C x x^T),$$

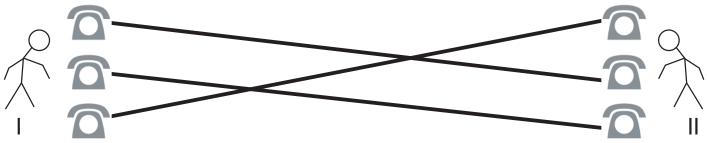
where C is the graph's adjacency matrix with $C_{ij} = 1$ if $\{i, j\} \in E$ and $C_{ij} = 0$ otherwise. Moreover, xx^T is a positive semidefinite matrix. So we can relax the constraints and obtain an upper bound on the optimal solution of (12.3), by replacing xx^T by a general positive semidefinite matrix X with $X_{ii} = 1$ for all $i \in V$. We arrive at the following optimization problem, which is an SDP:

$$\begin{aligned} &\text{maximize} && \frac{|E|}{2} - \frac{1}{4} \operatorname{tr}(CX) \\ &\text{subject to} && X_{ii} = 1 \quad \text{for all } i \in V \\ &&& X \succeq 0. \end{aligned}$$

Goemans and Williamson showed how to use a polynomial time algorithm to find a near optimal solution to this SDP, and then derive from it (by rounding) a feasible solution to (12.3). The reduction in objective function value that takes place in this process is, surprisingly, bounded by a factor of 0.87856. So this procedure finds, in polynomial time, a solution with a cut value that is at least 0.87856 of the true max-cut value. □

12.4 Symmetric rendezvous search game

Suppose that two players are placed in two rooms. Each room has 3 telephones, and these are pairwise connected at random, in a manner unknown to the players. On the stroke of each hour, $1, 2, \dots$, each player picks up one of his telephones and tries to communicate with the other player. They wish to minimize the expected number of attempts required until this occurs. Suppose that at time 1 each player has picked up a telephone at random, and they have failed to connect.



Imagine that for each player the telephones are arranged around a circle, and players have a common notion of clockwise. Player I labels the telephone which he picked at time 1 as 'a'. He labels the two phones that are one and two positions clockwise from a as 'b' and 'c', respectively. His possible pure strategies for time 2 are to pick up a, b or c. Suppose he adopts these with probabilities $x = (x_1, x_2, x_3)$. We assume the players are symmetric (perhaps reading what they should do in these circumstances from the same instruction manual), and so player II must use the same mixed strategy. The probability that the players fail to pick up connected telephones at the next attempt (i.e. fail to rendezvous) is

$$x^\top C_1 x = x^\top \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix} x.$$

The matrix C_1 is positive definite, and so the minimizer is easily found to be $x = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, with $x^\top C_1 x = 2/3$.

Similarly, thinking about both times 2 and 3, there are 9 pure strategies: aa, ab, ac, ba, bb, bc, ca, cb, cc. Suppose the players adopt these with probabilities $x = (x_1, \dots, x_9)$. One can show that the probability that the players have not yet managed to speak after the 3rd attempt is

$$x^\top C_2 x = x^\top \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix} x.$$

C_2 is not positive definite. (It's eigenvalues are $4, 1, 1, 1, 1, 1, 1, -\frac{1}{2}, -\frac{1}{2}$.) This means that the quadratic form $x^\top C_2 x$ has local minima. One such is given by $x^\top = (1/9)(1, 1, 1, 1, 1, 1, 1, 1, 1)$, which gives $x^\top C_2 x = 4/9$. But better is $x^\top = (1/3)(1, 0, 0, 0, 0, 1, 0, 1, 0)$, which gives $x^\top C_2 x = 1/3$. How might we prove this is best?

Let J_2 be the 9×9 matrix of 1s. Note that for x to be a vector of probabilities, we must have $x^\top J x = 9$. As with the max-cut problem we think of relaxing $xx^\top$ to a matrix $X \succeq 0$ and consider the SDP

$$\text{minimize } \operatorname{tr}(C_2 X) \text{ s.t. } X \in S_n,\ X \geq 0,\ X \succeq 0 \text{ and } \operatorname{tr}(J_2 X) = 9.$$

One can numerically compute that the optimal value of this SDP. It is $1/3$. This provides a lower bound on the probability that the players do not rendezvous by the end of the 3rd attempt. This is achieved by $x^\top = (1/3)(1, 0, 0, 0, 0, 1, 0, 1, 0)$ — so this strategy does indeed minimize the probability that they have not yet met by the end of the 3rd attempt.

These ideas can be extended (Weber, 2008) to show that the expected time to rendezvous is minimized when players adopt a strategy in which they choose their first telephone at random, and if this does not connect them then on successive pairs of subsequent attempts they choose aa, bc or cb, each with probability $1/3$. Given that they fail to meet at the first attempt, the expected number of further attempts required is $5/2$. This is less than 3, i.e. the expected number of steps required if players simply try telphones at random at each attempt. There are many simply-stated but unsolved problems in the field of search games.

13 Branch and Bound

There are three conceptually different approaches for optimization problems that are computationally hard: (i) an exact method, which finds optimal solutions but has an exponential worst-case running time, (ii) heuristic methods, which need not offer guarantees regarding running time or solution quality, but often provide a good tradeoff between the two in practice, and (iii) approximation algorithms, which run in polynomial time and return solutions with a guaranteed bound on the suboptimality. We met an example of (iii) for the max-cut problem. We start with another example of (iii) and then continue with (i) and (ii).

13.1 Knapsack problem

In the 0-1 **knapsack problem** we are to

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} x_i v_i \\ \text{subject to} & \sum_{i=1}^{n} x_i w_i \leq B, \quad x_i \in \{0,1\}. \end{array}$$

Assume $v_1/w_1 \geq v_2/w_2 \geq \cdots v_n/w_n$ and $w_i < B$ for all i. A greedy algorithm is to place items in the knapsack in order $1, 2, \ldots, k$, until item $k+1$ does not fit. Suppose the optimal value is OPT. One can easily prove that

$$\text{OPT} \leq v_1 + \cdots + v_k + v_{k+1}. \tag{13.1}$$

So either $v_1 + \cdots + v_k$ or v_{k+1} exceeds OPT/2, and by selecting the greater of these we have a 1/2-approximation algorithm. In fact, we can find a $(1-\epsilon)$-approximation algorithm for any $\epsilon > 0$, as follows.

For a set S let $v(S) = \sum_{i \in S} v_i$ and $w(S) = \sum_{i \in S} w_i$. Suppose O is a set of optimal items for the knapsack and $|O| > m$. Let H be the subset of O consisting of its m most valuable items. Consider filling the knapsack with $H \cup G$, where G is obtained by filling the space $B - w(H)$ by using the greedy algorithm on items with values no more than values in H, i.e. taking such items in decreasing order of v_i/w_i until one does not fit; suppose that is j. By (13.1) we have $v(O) \leq v(H \cup G) + v_j$. But $v_j \leq v(H)/m \leq v(O)/m$. Hence $v(H \cup G) \geq (1 - 1/m)v(O)$.

Suppose we take each subset K for which $|K| \leq m$, and from it create a packing by using the greedy algorithm on the remaining items with values no more than values in K to fill the space $B - w(K)$. We will either find the optimal packing, say O, when $|O| \leq m$, or at some point take $K = H$ and construct $H \cup G$ such that $v(H \cup G) \geq (1 - 1/m)v(O)$.

The number of subsets K which we must examine is at most $\sum_{i=1}^{m} \binom{n}{i} = O(mn^m)$, so running time is polynomial in n for fixed m. The knapsack problem lies in the PTAS, which is defined as the class of problems which having a **polynomial-time approximation scheme**. Clearly PTAS⊆APX. It can be shown that there is strict inequality unless P=NP.

13.2 Branch and bound technique

Suppose we wish to solve the knapsack problem exactly. It is NP-hard, so there is no polynomial time algorithm unless P=NP. However, there is something we can do. A useful fact is that if we have partially filled the knapsack with some items in the set A, and items in set $B = \{i_1, i_2, \dots\}$ are still available for inclusion, with $v_{i_1}/w_{i_1} \geq v_{i_2}/w_{i_2} \geq \cdots$, then the value we can obtain by completing the packing optimally is bounded above by

$$\sum_{i \in A} v_i + v_{i_1} + \cdots + v_{i_k} + \alpha(v_{i_{k+1}}/w_{i_{k+1}}),$$

where $\sum_{i \in A} w_i + w_{i_1} + w_{i_2} + \cdots + w_{i_k} + \alpha = B$, and $\alpha < w_{i_{k+1}}$. This bound is very easy to compute.

Branch and bound is a general method for solving optimization problems, especially in the context of non-convex and combinatorial optimization. Suppose we want to

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in X \end{array}$$

for some feasible region X. Branch and bound uses **divide and conquer**, which splits a problem into smaller and smaller subproblems until they become easy to solve. For the above minimization problem, it works by splitting X into $k \geq 2$ sets $X_1, \dots, X_k$ such that $\bigcup_{i=1,\dots,k} X_i = X$. This step is called **branching**, since its recursive application defines a tree structure, the so-called **search tree**, whose vertices are the subsets of X. Once optimal solutions have been found for the subsets $X_1, \dots, X_k$, it is easy to obtain a solution for X, because $\min_{x \in X} f(x) = \min_{i=1,\dots,k} \min_{x \in X_i} f(x)$.

Of course, branching as such doesn't make the problem any easier to solve, and for an NP-hard problem we may have to explore an exponential number of vertices of the search tree. In practice we might hope, however, that we will be able to **prune** large parts of the tree that cannot contain an optimal solution. The procedure that allows us to do this is known as **bounding**. It tries to find lower and upper bounds on the optimal solution, i.e. functions ℓ and u such that for all $X' \subseteq X$, $\ell(X') \leq \min_{x \in X'} f(x) \leq u(X')$. Then, if $\ell(Y) \geq u(Z)$ for two sets $Y, Z \subseteq X$, then Y can be discarded. A particular situations where this happens is when Y does not contain any feasible solutions, and we assume that $\ell(Y) = \infty$ by convention in this case.

For the upper bound, it suffices to store the value $U = f(x)$ of the best feasible solution $x \in X$ found so far. A good way to obtain a lower bound for a set $Y \subseteq X$ is by letting $\ell(Y) = \min_{x \in Y'} f(x)$ for some set $Y' \supseteq Y$ for which minimization of f is computationally tractable. It is easy to see that this indeed provides a lower bound. Moreover, if minimization over Y' yields a solution $x \in Y$, then this solution is optimal for Y. The branch and bound method stores U and a list L of active sets $Y \subseteq X$ for which no optimal solution has been found so far, corresponding to vertices in the search tree that still need to be explored, along with their associated lower bounds. It then proceeds as follows:

1. Initialize: set $U = \infty$. L is a list of subsets of X. Start with $L = \{X\}$.
2. Branch: pick a set $Y \in L$, remove it from L, and split it into $k \geq 2$ sets $Y_1, \dots, Y_k$.
3. Bound: for $i = 1, \dots, k$, compute $\ell(Y_i)$. If this yields $x \in X$ such that $\ell(Y_i) = f(x) < U$, then set U to $f(x)$. If $\ell(Y_i) < U$, but no $x \in X$ as above is found, then add Y_i to L.
4. If $L = \emptyset$, then stop. The optimum objective value is U. Otherwise go to Step 2.

To apply the method in a concrete setting, we need to specify how a set Y to branch on is chosen and how it is split into smaller sets, and how lower bounds are computed. These decisions are of course critical for the practical performance of the procedure.

13.3 Dakin's method

Dakin (1965) proposed an obvious way of using branch and bound to solved integer linear programs. Lower bounds are obtained by solving the LP relaxation, i.e. the linear program obtained by dropping the integrality constraints.

Assume that we are branching on a set $Y \in L$, and that the LP relaxation corresponding to Y has optimal solution y. If $y \in Y$, then y is optimal for Y. Otherwise, there is some i such that y_i is not integral, and we can split Y into two sets $Y^1 = \{x \in Y : x_i \leq \lfloor y_i \rfloor\}$ and $Y^2 = \{x \in Y : x_i \geq \lceil y_i \rceil\}$. Note that $Y^1 \cup Y^2 = Y$, as desired. Moreover, this branching rule forces the solution away from its current value $y \notin Y$. While this does not guarantee that y_i becomes integral in the next step, and may even force another variable away from its integral value, it works remarkably well in practice. It is worth noting that we do not have to start from scratch when solving the LP relaxation for Y_i: it was obtained by adding a constraint to an LP that is already solved, and the dual simplex method often finds a solution satisfying the additional constraint very quickly. In order to minimize the number of solved LPs that have to be stored to implement this approach, it makes sense to branch on a set obtained in the previous step whenever possible, i.e. to traverse the search tree in a depth-first manner.

Example 13.1. Assume that we want to

$$\begin{array}{ll} \text{minimize} & x_1 - 2x_2 \\ \text{subject to} & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1 \geq 0, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{array}$$

An illustration is shown in Figure 16. Let $f(x) = x_1 - 2x_2$, and $X = \mathbb{Z}^2 \cap \tilde{X}$ where

$$\tilde{X} = \{\, x \in \mathbb{R}^2 : -4x_1 + 6x_2 \leq 9, x_1 + x_2 \leq 4, x_1 \geq 0, x_2 \geq 0 \,\}.$$

The search tree for an application of Dakin's method is shown in Figure 17.

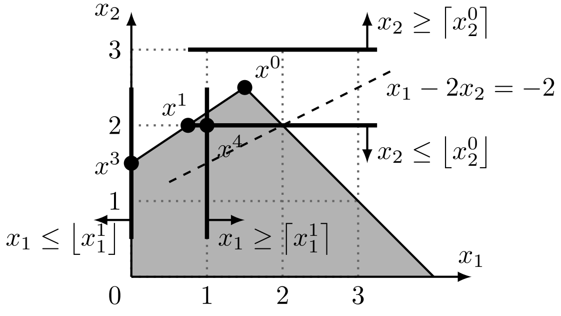

x_2
3
2
1
0
1
2
3
x_1
x^0
x^1
x^3
x^4
$x_2 \geq \lceil x_2^0 \rceil$
$x_1 - 2x_2 = -2$
$x_2 \leq \lfloor x_2^0 \rfloor$
$x_1 \leq \lfloor x_1^1 \rfloor$
$x_1 \geq \lceil x_1^1 \rceil$

Figure 16: Illustration of Dakin's method, applied to the IP of Example 13.1

- Start with $U = \infty$ and $L = \{X\}$. By solving the LP relaxation for X, we find that $\ell(X) = \min_{x \in \tilde{X}} f(x) = f(x^0) = -7/2$ for $x^0 = (3/2, 5/2)$.
- Set X is the only candidate for branching and can be split into $X^1 = \mathbb{Z}^2 \cap \tilde{X}^1$ and $X^2 = \mathbb{Z}^2 \cap \tilde{X}^2$, where

$$\tilde{X}^1 = \{x \in \tilde{X} : x_2 \leq 2\} \quad \text{and} \quad \tilde{X}^2 = \{x \in \tilde{X} : x_2 \geq 3\}.$$

- Bound X^1 and X^2 by solving the corresponding LP relaxations, and obtain $\ell(X^1) = \min_{x \in \tilde{X}^1} f(x) = f(x^1) = -13/4$ for $x^1 = (3/4, 2)$, and $\tilde{X}^2 = \emptyset$. We thus set $L = \{X^1\}$.
- Branch by splitting X^1 into $X^3 = \mathbb{Z}^2 \cap \tilde{X}^3$ and $X^4 = \mathbb{Z}^2 \cap \tilde{X}^4$, where

$$\tilde{X}^3 = \{x \in \tilde{X}^1 : x_1 \leq 0\} \quad \text{and} \quad \tilde{X}^4 = \{x \in \tilde{X}^1 : x_1 \geq 1\},$$

- Bound X^3 and X^4 to obtain $\ell(X^3) = \min_{x \in \tilde{X}^3} f(x) = f(x^3) = -3$ for $x^3 = (0, 3/2)$ and $\ell(X^4) = \min_{x \in \tilde{X}^4} f(x) = f(x^4) = -3$ for $x^4 = (1, 2)$. Since $x^4 \in X$, we set $U = f(x^4) = -3$. Then, $\ell(X^3) \geq U$, so we can discard X^3 and are done.

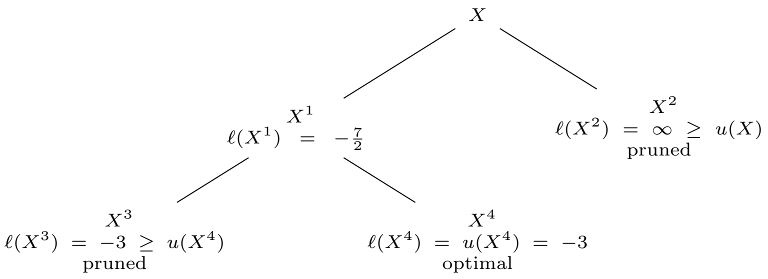

X
X^1
$\ell(X^1) = -\frac{7}{2}$
X^2
$\ell(X^2) = \infty \geq u(X)$
pruned
X^3
$\ell(X^3) = -3 \geq u(X^4)$
pruned
X^4
$\ell(X^4) = u(X^4) = -3$
optimal

Figure 17: Search tree explored by Dakin's method for the IP of Example 13.1

14 Heuristic Algorithms

14.1 The travelling salesman problem

Recall that in the travelling salesman problem (TSP) we are given a matrix $A \in \mathbb{N}^{n \times n}$ and are looking for a permutation $\sigma \in S_n$ that minimizes $a_{\sigma(n)\sigma(1)} + \sum_{i=1}^{n-1} a_{\sigma(i)\sigma(i+1)}$. Matrix entry a_{ij} can be interpreted as a cost associated with edge $(i,j) \in E$ of a graph $G = (V, E)$, and we are then trying to find a **tour**, i.e. a cycle in G that visits every vertex exactly once, of minimum overall cost. We have seen that the TSP is NP-hard, but we could try to encode it as an integer program and solve it using branch and bound. Consider variables

$$x_{ij} \in \{0,1\} \quad \text{for } i,j = 1,\dots,n, \tag{14.1}$$

encoding whether the tour traverses edge (i,j). There are various ways to ensure that these variables indeed encode a tour, i.e. that $x_{ij} = 1$ if and only if $\sigma(n) = i$ and $\sigma(1) = j$, or $\sigma(k) = i$ and $\sigma(k+1) = j$ for some $k \in \{1,\dots,n-1\}$. Of course, there has to be exactly one edge entering and one edge leaving every vertex, i.e.

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1,\dots,n, \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1,\dots,n. \tag{14.2}$$

The so-called cut-set formulation additionally requires that there are at least two edges across every cut $S \subseteq V$, whereas the subtour elimination formulation makes sure that no set $S \subset V$ contains more than $|S| - 1$ edges. The problem with both of these formulations is of course that they require an exponential number of constraints, one for each set $S \subseteq V$.

A polynomial formulation can be obtained by introducing, for $i = 1,\dots,n$, an auxiliary variable $t_i \in \{0,\dots,n-1\}$ indicating the position of vertex i in the tour. If $x_{ij} = 1$, it holds that $t_j = t_i + 1$. If $x_{ij} = 0$, on the other hand, then $t_j \geq t_i - (n-1)$. This can be written more succinctly as

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \quad \text{for all } i \geq 1,\ j \geq 2,\ i \neq j. \tag{14.3}$$

Since values satisfying (14.3) exist for every valid tour, adding this constraint does not affect solutions corresponding to valid tours. On the other hand, it suffices to rule out subtours, i.e. cycles of length less than $|V|$. To see this, consider a solution that satisfies (14.3), and assume for contradiction that it consists of two or more subtours. Summing the constraints over the edges in a subtour that does not contain vertex 1 leads to the nonsense that $0 \geq k$, where k is the number of edges in the subtour.

A minimum cost tour can thus be found by minimizing $\sum_{i,j} x_{ij} a_{ij}$ subject to (14.1), (14.2), and (14.3). This integer program has a polynomial number of variables and constraints and can be solved using Dakin's method, which bounds the optimum by relaxing the integrality constraints (14.1). There are, however, other relaxations that are specific to the TSP and can provide better bounds.

Observe, for example, that the integer program obtained by relaxing the subtour elimination constraints (14.3) is an instance of the assignment problem (9.1). It can be solved efficiently in practice using the network simplex method or Hungarian algorithm, which yields a solution consisting of one or more subtours. If there is more than one subtour, then taking the set $\{e_1, \dots, e_k\} \subseteq E$ of edges of one or more of the subtours and disallowing each of them in turn splits the feasible set Y into $Y_1, \dots, Y_k$, where $Y_i = \{x \in Y : x_{uv} = 0, e_i = (u, v)\}$. Clearly, the optimal TSP tour cannot contain all edges of a subtour, so it must be contained in one of the sets Y_i. Moreover, adding a constraint of the form $x_{ij} = 0$ is equivalent to setting a_{ij} to a large enough value, so the new problem will still be an instance of the assignment problem. Note that none of the sets Y_i contains the optimal solution of the current relaxation, so $\ell(Y_i) \geq \ell(Y)$ for all i, and $\ell(Y_i) > \ell(Y)$ if the optimal solution of the current relaxation was unique.

14.2 Heuristic algorithms

For all we know, a complete exploration of the search tree, either explicitly or implicitly, might be required to guarantee that an optimal solution is found. When this is impractical, heuristic methods can be used to find a satisfactory, but possibly suboptimal, solution. Heuristics sacrifice solution quality in order to gain computational performance or conceptual simplicity.

14.3 Heuristics for the TSP

A straightforward way of constructing a TSP tour is by starting from an arbitrary vertex, traversing a minimum cost edge to an unvisited vertex until all vertices have been visited, and returning to the initial vertex to complete the tour. This greedy algorithm is known as the **nearest neighbor heuristic** and has an asymptotic complexity of $O(n^2)$, where n is the number of vertices. Intuitively it will work well most of the time, but will sometimes have to add an edge with very high cost because all vertices connected to the current one by an edge with low cost have already been visited.

An alternative procedure is to order the edges by increasing cost and adding them to the tour in that order, skipping edges that would lead to a vertex with degree more than two or a cycle of length less than n. This so-called **savings heuristic** has complexity $O(n^2 \log n)$, which is the complexity of sorting a set with n^2 elements.

Another general approach are **insertion heuristics**, which start with a subtour, i.e. a tour on a subset of the set of vertices, and extend it with additional vertices. The initial subtour can be a cycle of two or three vertices . The **cheapest insertion heuristic** then chooses a vertex, and a place to insert it into the subtour, in order to minimize the overall length of the resulting subtour. The **farthest insertion heuristic**, on the other hand, inserts a vertex whose minimum distance to any vertex in the current subtour is maximal. The idea behind the latter strategy is to fix the overall layout of the tour as soon as possible.

14.4 Local search

Of course, optimization does not need to end once we have constructed a tour. Rather, we could try to make a small modification to the tour in order to reduce its cost, and repeat this procedure as long as we can find such an improving modification. An algorithm with this general procedure is known as a **local search algorithm**, because it makes local modifications to a solution to obtain a new solution with a better objective value. A tour created using the nearest neighbor heuristic, for example, will usually contain a few edges with very high cost, so we would be interested in local modifications that eliminate these edges from the tour.

More generally, suppose we want to

$$\begin{array}{ll} \text{minimize} & c(x) \\ \text{subject to} & x \in X, \end{array}$$

and that for any feasible solution $x \in X$, the cost $c(x)$ and a **neighborhood** $N(x) \subseteq X$ can be computed efficiently. Local search then proceeds as follows:

1. Find an initial feasible solution $x \in X$.
2. Find a solution $y \in N(x)$ such that $c(y) < c(x)$.
3. If there is no such solution, then stop and return x; otherwise set the current solution x to y and return to Step 2..

The solution returned by this procedure is a **local optimum**, in that its cost is no larger than that of any solution in its neighborhood. It need not be globally optimal, as there might be a solution outside the neighborhood with strictly smaller cost.

Any of the basic tour construction heuristics can be used to find an initial feasible solution in Step 1., and the whole procedure can also be run several times with different initial solutions. Step 2. requires a choice if more than one neighboring solution provides a decrease in cost. Natural options include the first such solution to be found, or the solution providing the largest decrease.

Most importantly, however, any implementation of a local search method must specify the neighborhood function N. A natural neighborhood for the TSP is the k-OPT neighborhood. Here, the neighbors of a given tour are obtained by removing any set of k edges, for some $k \geq 2$, and reconnecting the k paths thus obtained to a tour by adding k edges. Viewing tours as permutations, k-OPT cuts a permutation into k segments and reverses and swaps these segments in a arbitrary way. An illustration for $k = 2$ and $k = 3$ is shown in Figure 18.

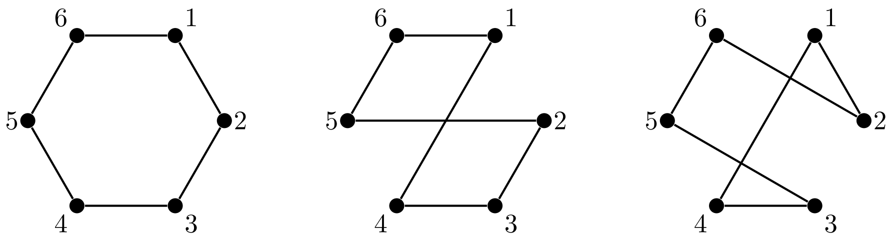


Figure 18: A TSP tour (left) and neighboring tours under the 2-OPT (middle) and 3-OPT neighborhoods (right). The tours respectively correspond to the permutations 123456, 143256, and 126534.

The choice of k provides a tradeoff between solution quality and speed: the k-OPT neighborhood of a solution contains its ℓ-OPT neighborhood if $k \geq \ell$, so the quality of the solution increases with k; the same is also true for the complexity of the method, because the k-OPT neighborhood of a tour of length n has size $O(n^k)$ and computing the change in cost between two neighboring tours requires $O(k)$ operations. Empirical evidence suggests that 3-OPT often performs better than 2-OPT, while there is little gain in taking $k > 3$.

Note that the simplex method for linear programming can also be viewed as a local search algorithm, where two basic feasible solutions are neighbors if their bases differ by exactly one element. We have seen that in this case every local optimum is also a global optimum, so that the simplex method yields a globally optimal solution.

In general, however, local search might get stuck in a local optimum and fail to find a global one. Consider for example the TSP instance given by the cost matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 4 & 4 \\ 4 & 0 & 1 & 0 & 4 \\ 4 & 4 & 0 & 1 & 0 \\ 0 & 4 & 4 & 0 & 1 \\ 1 & 0 & 4 & 4 & 0 \end{pmatrix}.$$

There are $4! = 24$ TSP tours, and

$$\begin{array}{llll} c(12345) = 5, & c(13245) = 6, & c(14235) = 10, & c(15234) = 6, \\ c(12354) = 6, & c(13254) = 12, & c(14253) = 20, & c(15243) = 12, \\ c(12435) = 6, & c(13425) = 10, & c(14325) = 17, & c(15324) = 12, \\ c(12453) = 10, & c(13452) = 6, & c(14352) = 9, & c(15342) = 17, \\ c(12534) = 10, & c(13524) = 0, & c(14523) = 10, & c(15423) = 17, \\ c(12543) = 17, & c(13542) = 12, & c(14532) = 17, & c(15432) = 20. \end{array}$$

It is easily verified that the tour 12345 is a local optimum under the 2-OPT neighborhood, while the global optimum is the tour 13524.

14.5 Simulated annealing

To prevent local search methods from getting stuck in a local optimum, one could allow transitions to a neighbor even if it has higher cost, with the hope that solutions with lower cost will be reachable from there. **Simulated annealing** implements this idea using an analogy to the process of annealing in metallurgy, in which a metal is heated and then cooled gradually in order to bring it to a low-energy state that comes with better physical properties.

In each iteration, simulated annealing considers a neighbor y of the current solution x and moves to the new solution with probability

$$p_{xy} = \min\left(1, \exp\left(-\frac{c(y)-c(x)}{T}\right)\right),$$

where $T \geq 0$ is a parameter, the **temperature**, that can vary over time. With the remaining probability the solution stays the same. When T is large, the method allows transitions even when $c(y)$ exceeds $c(x)$ by a certain amount. As T approaches zero, so does the probability of moving to a solution with larger cost.

It can further be shown that with a suitable **cooling schedule** that decreases T sufficiently slowly from iteration to iteration, the probability of reaching an optimal solution after t iterations tends to 1 as t tends to infinity. To motivate this claim, consider the special case where every solution has k neighbors and a neighbor of the current solution is chosen uniformly at random. The behavior of the algorithm can then be modeled as a **Markov chain** with transition probabilities

$$P_{xy} = \begin{cases} p_{xy}/k & \text{if } y \in N(x), \\ 1 - \sum_{z \in N(x)} p_{xz}/k & \text{if } y = x, \\ 0 & \text{otherwise.} \end{cases}$$

This Markov chain has a unique **stationary distribution** π, i.e. a distribution over X such that for all $x \in X$, $\pi_x = \sum_{y \in X} \pi_y P_{yx}$. In addition it can be shown that π must satisfy the **detailed balance** condition that $\pi_x P_{xy} = \pi_y P_{yx}$ for every pair of solutions $x, y \in X$. In fact, detailed balance is not only necessary but also sufficient for stationary, because it implies that $\sum_{x \in X} \pi_x P_{xy} = \sum_{x \in X} \pi_y P_{yx} = \pi_y \sum_{x \in X} P_{yx} = \pi_y$. It is not hard to show that π with

$$\pi_x = \frac{e^{-c(x)/T}}{\sum_{z \in X} e^{-c(z)/T}}$$

for every $x \in X$ is a distribution and satisfies detailed balance, and must therefore be the stationary distribution. Letting $Y \subseteq X$ be the set of solutions with minimum cost and $\pi_Y = \sum_{x \in Y} \pi_x$, we conclude that $\pi_Y/(1-\pi_Y) \to \infty$ as $T \to 0$.

The idea now is to decrease T slowly enough for the Markov chain to be able to reach its stationary distribution. A common cooling schedule is to set $T = c/\log t$ in iteration t, for some constant c.

15 Non-cooperative Games

The rest of the course is about situations in which multiple self-interested entities, or *agents*, operate in the same environment. **Game theory** provides mathematical models, so-called games, for studying such situations. We focus for now on **non-cooperative games**, in which agents independently optimize different objectives and outcomes must be self-enforcing. Later, we consider **cooperative games** and focus upon conditions under which cooperation among subsets of agents can be sustained.

15.1 Games and solutions

The central object of study in non-cooperative game theory is the **normal-form game**. This is a tuple $\Gamma = (N, (A_i)_{i \in N}, (p_i)_{i \in N})$ where N is a finite set of **players**, and for each player $i \in N$, A_i is a non-empty and finite set of **actions** available to i and $p_i : (\times_{i \in N} A_i) \to \mathbb{R}$ is a function mapping each action profile, i.e. each combination of actions, to a real-valued **payoff** for i. Unless noted otherwise, the results we consider are invariant under positive affine transformations of payoffs, and payoffs will not be comparable across players.

More complicated games in which players move sequentially and base their decisions on their and others' earlier moves can also be represented as normal-form games, by encoding every possible course of action in the former by an action of the latter. However, this generally leads to a large increase in the number of actions.

A two-player game with m actions for player 1 and n actions for player 2 can be represented by matrices $P, Q \in \mathbb{R}^{m \times n}$, where p_{ij} and q_{ij} are the payoffs of players 1 and 2 when player 1 plays action i and player 2 plays action j. Two-player games are therefore sometimes referred to as **bimatrix games**, and players 1 and 2 as the **row and column player**, respectively. The concepts and results we address for two-person games extend in a straightforward way to games with more than two players.

Assume players can choose their actions randomly and denote the set of possible **strategies** of the two players by X and Y, respectively, i.e. $X = \{x \in \mathbb{R}^m_{\geq 0} : \sum_{i=1}^m x_i = 1\}$ and $Y = \{y \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n y_i = 1\}$. A **pure strategy** is a strategy that chooses some action with probability 1. A profile $(x, y) \in X \times Y$ of strategies induces a lottery over outcomes, and we write $p(x, y) = x^T P y$ and $q(x, y) = x^T Q y$ for the expected payoff of the two players in this lottery.

Consider for example the well-known **prisoner's dilemma**, involving two suspects accused of a crime who are being interrogated separately. If both remain silent, they walk free after spending a few weeks in detention. If one of them testifies against the other and the other remains silent, the former is released immediately while the latter receives a ten-year sentence. If both testify, each of them receives a five-year sentence. The representation as a normal-form game is shown in Figure 19.

$$\begin{array}{c|cc|}
 & S & T \\
\hline
S & (2,2) & (0,3) \\
T & (3,0) & (1,1) \\
\hline
\end{array}$$

Figure 19: Representation of the prisoner's dilemma as a normal-form game. The matrices P and Q are displayed as a single matrix with entries (p_{ij}, q_{ij}), and players 1 and 2 respectively choose a row and a column of this matrix. Action S corresponds to remaining silent, action T to testifying.

It is easy to see that both players should choose T because this action yields a strictly larger payoff than action S for *every* action of the respective other player. More generally, for two strategies $x, x' \in X$ of the row player, x is said to **strictly dominate** x' if for every strategy $y \in Y$ of the column player, $p(x, y) > p(x', y)$. Dominance for the column player is defined analogously. Strategy profile (T, T) in the prisoner's dilemma is what is called a **dominant strategy equilibrium**, a profile of strategies that dominate every other strategy of the respective player. The source of the dilemma is that outcome resulting from (T, T) is strictly worse for both players than the outcome resulting from (S, S). More generally, an outcome that is weakly preferred to another outcome by all players, and strictly preferred by at least one player is said to **Pareto dominate** that outcome. An outcome that is Pareto dominated is clearly undesirable.

In the absence of dominant strategies, it is less obvious how players should proceed. Consider for example the game of chicken illustrated in Figure 20. It models a situation in which two cars drive towards each other on a collision course. Unless one of the drivers yields, both may die in a crash. If one of them yields while the other goes straight, however, the former will be called a "chicken", or coward. It is easily verified that this game does not have any dominant strategies.

$$\begin{array}{c|cc|}
 & C & D \\
\hline
C & (2,2) & (1,3) \\
D & (3,1) & (0,0) \\
\hline
\end{array}$$

Figure 20: The game of chicken, where players can chicken out or dare

The most cautious choice in a situation like this would be to ignore that the other player is self-interested and choose a strategy that maximizes the payoff in the worst case, taken over all of the other player's strategies. A strategy of this type is known as a **maximin strategy**, and the payoff thus obtained as the player's **security level**. It is easy to see that it suffices to maximize the minimum payoff over all **pure** strategies of the other player, i.e. to choose x such that $\min_{j \in \{1,\ldots,n\}} \sum_{i=1}^{m} x_i p_{ij}$ is maximized. Thus the maximin strategy and the security level for the row player can be found from the following linear program with variables $v \in \mathbb{R}$ and $x \in \mathbb{R}^m$:

$$\begin{aligned} &\text{maximize} && v \\ &\text{subject to} && \sum_{i=1}^m x_i p_{ij} \geq v \quad \text{for } j = 1, \dots, n, \quad \sum_{i=1}^m x_i = 1, \quad x \geq 0. \end{aligned} \tag{15.1}$$

The unique maximin strategy in the game of chicken is to yield, for a security level of 1. This also illustrates that a maximin strategy need not be optimal: assuming that the other player yields, the best response is in fact to go straight. Formally, strategy $x \in X$ of the row player is a **best response** to strategy $y \in Y$ of the column player if for all $x' \in X$, $p(x, y) \geq p(x', y)$. The concept of a best response for the column player is defined analogously.

A pair of strategies $(x, y) \in X \times Y$ such that x is a best response to y and y is a best response to x is called a (Nash) **equilibrium**. It is easily verified that both (C, D) and (D, C) are equilibria of the game of chicken. There is one more equilibrium, in which both players randomize between their two actions, giving each probability $1/2$.

15.2 Minimax theorem for zero-sum games

In the special case that interests of the two players are diametrically opposed, maximin strategies are optimal in a very strong sense. A two-player game is a **zero-sum game** if $q_{ij} = -p_{ij}$ for all $i = 1, \dots, m$ and $j = 1, \dots, n$. It is also then called a **matrix game**, because it can be represented just by the matrix P containing the payoffs of the row player. Assuming invariance of utilities under positive affine transformations, results for zero-sum games in fact apply to the larger class of **constant-sum** games, in which the payoffs of the two players always sum up to the same constant. For games with more than two players, these properties are far less interesting, as one can always add an extra player who "absorbs" the payoffs of the others.

Theorem 15.1 (von Neumann, 1928)**.** *Let $P \in \mathbb{R}^{m \times n}$, $X = \{x \in \mathbb{R}^m_{\geq 0} : \sum_{i=1}^m x_i = 1\}$, $Y = \{y \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n y_i = 1\}$. Then,*

$$\max_{x \in X} \min_{y \in Y} p(x, y) = \min_{y \in Y} \max_{x \in X} p(x, y).$$

Proof. Again consider the linear program (15.1), and recall that the optimal solution of this linear program is equal to $\max_{x \in X} \min_{y \in Y} p(x, y)$. By adding a slack variable $z \in \mathbb{R}^n$ with $z \geq 0$ we obtain the Lagrangian

$$\begin{aligned} L(v, x, z, w, y) &= v + \sum_{j=1}^n y_j \Big(\sum_{i=1}^m x_i p_{ij} - z_j - v \Big) - w \Big(\sum_{i=1}^m x_i - 1 \Big) \\ &= \Big(1 - \sum_{j=1}^n y_j \Big) v + \sum_{i=1}^m \Big(\sum_{j=1}^n p_{ij} y_j - w \Big) x_i - \sum_{j=1}^n y_j z_j + w, \end{aligned}$$

where $w \in \mathbb{R}$ and $y \in \mathbb{R}^n$. The Lagrangian has a finite maximum for $v \in \mathbb{R}$ and $x \in \mathbb{R}^m$ with $x \geq 0$ if and only if $\sum_{j=1}^n y_j = 1$, $\sum_{j=1}^n p_{ij} y_j \leq w$ for $i = 1, \dots, m$, and $y \geq 0$. The dual of (15.1) is therefore

$$\begin{aligned} &\text{minimize} \quad w \\ &\text{subject to} \quad \sum_{j=1}^n p_{ij} y_j \leq w \quad \text{for } i = 1, \dots, m, \quad \sum_{j=1}^n y_j = 1, \quad y \geq 0. \end{aligned}$$

It is easy to see that the optimal solution of the dual is $\min_{y \in Y} \max_{x \in X} p(x, y)$, and the theorem follows from strong duality. □

The number $\max_{x \in X} \min_{y \in Y} p(x, y) = \min_{y \in Y} \max_{x \in X} p(x, y)$ is called the **value** of the matrix game with payoff matrix P.

The solution of a matrix game can be found by solving the linear program (15.1). This problem can be simplified by first adding a constant to every element of P so that $P > 0$. This does not change the equilibrium of the game, but ensures that at the solution we must have $v > 0$. By setting $x' = x/v$, and noting that $1/v = \sum_i x'_i$, we can rewrite (15.1) as

$$\text{minimize} \quad \sum_{i=1}^m x'_i \quad \text{subject to} \quad \sum_{i=1}^m x'_i p_{ij} \geq 1 \text{ for } j = 1, \dots, n, \quad x' \geq 0.$$

Alternatively, we might apply a similar transformation to the dual and solve

$$\text{maximize} \quad \sum_{i=1}^n y'_i \quad \text{subject to} \quad \sum_{i=1}^n p_{ij} y'_i \leq 1 \text{ for } j = 1, \dots, n, \quad \bar{y}' \geq 0.$$

15.3 Equilibria of matrix games

The minimax theorem implies that every matrix game has an equilibrium, and in fact characterizes the set of equilibria of these games.

Theorem 15.2. *A pair of strategies $(x, y) \in X \times Y$ is an equilibrium of the matrix game with payoff matrix P if and only if it is a minimax point, i.e.*

$$\begin{aligned} \min_{y' \in Y} p(x, y') &= \max_{x' \in X} \min_{y' \in Y} p(x', y') \quad \textit{and} \\ \max_{x' \in X} p(x', y) &= \min_{y' \in Y} \max_{x' \in X} p(x', y'). \end{aligned} \tag{15.2}$$

Proof. For all $(x, y) \in X \times Y$,

$$\min_{y' \in Y} \max_{x' \in X} p(x', y') \leq \max_{x' \in X} p(x', y) \geq p(x, y) \geq \min_{y' \in Y} p(x, y') \leq \max_{x' \in X} \min_{y' \in Y} p(x', y'),$$

and the first and last term are equal by Theorem 15.1.

If (x, y) is an equilibrium, the second and third inequality hold with equality. This means that the first and last inequality have to hold with equality as well, and (15.2) follows.

On the other hand, if (15.2) is satisfied, then the first and last inequality hold with equality. This means that the second and third inequality have to hold with equality as well, so (x, y) is an equilibrium. □

Some other properties specific to matrix games are stated in the following theorem. These are that all equilibria yield the same payoffs and that any pair of strategies of the two players, such that each of them is played in some equilibrium, is itself an equilibrium.

Theorem 15.3. *Let $(x, y), (x', y') \in X \times Y$ be equilibria of the matrix game with payoff matrix P. Then $p(x, y) = p(x', y')$, and (x, y') and (x', y) are equilibria as well.*

Proof. Since equilibrium strategies are best responses to each other, we have that

$$p(x, y) \leq p(x, y') \leq p(x', y') \leq p(x', y) \leq p(x, y).$$

Since the first and last term are the same, the inequalities have to hold with equality and the first claim follows. Then,

$$\begin{aligned}
p(x, y') &= p(x', y') \geq p(z, y') && \text{for all } z \in X, \\
p(x, y') &= p(x, y) \leq p(x, z) && \text{for all } z \in Y, \\
p(x', y) &= p(x, y) \geq p(z, y) && \text{for all } z \in X, \text{ and} \\
p(x', y) &= p(x', y') \geq p(x', z) && \text{for all } z \in X,
\end{aligned}$$

where the inequalities hold because (x, y) and (x', y') are equilibria. Thus (x, y') and (x', y) are pairs of strategies that are best responses to each other, and the second claim follows as well. □

Theorems 15.1, 15.2, and 15.3 together also imply that the set of equilibria of a matrix game is convex.

16 Solution of Two-person Games

16.1 Nash's theorem

Many of the results concerning equilibria of matrix games do *not* carry over to bimatrix games, with the exception of existence.

Theorem 16.1 (Nash, 1951)**.** *Every bimatrix game has an equilibrium.*

We use the following result.

Theorem 16.2 (Brouwer fixed point theorem)**.** *Let $f : S \to S$ be a continuous function, where $S \subseteq \mathbb{R}^n$ is closed, bounded, and convex. Then f has a fixed point.*

Proof of Theorem 16.1. Define X and Y as before, and observe that $X \times Y$ is closed, bounded, and convex. For $x \in X$ and $y \in Y$ define $s_i(x, y)$ and $t_j(x, y)$ as the additional payoff the two players could obtain by playing their ith or jth pure strategy instead of x or y, i.e.

$$\begin{aligned} s_i(x,y) &= \max\{0, p(e_i^m, y) - p(x,y)\} && \text{for } i = 1,\dots,m \text{ and} \\ t_j(x,y) &= \max\{0, q(x, e_j^n) - q(x,y)\} && \text{for } j = 1,\dots,n, \end{aligned}$$

where e_ℓ^k denotes the ℓth unit vector in $\mathbb{R}^k$. Further define $f : (X \times Y) \to (X \times Y)$ by letting $f(x, y) = (x', y')$ with

$$x_i' = \frac{x_i + s_i(x,y)}{1 + \sum_{k=1}^m s_k(x,y)} \qquad \text{and} \qquad y_j' = \frac{y_j + t_j(x,y)}{1 + \sum_{k=1}^n t_k(x,y)}$$

for $i = 1, \dots, m$ and $j = 1, \dots, n$. Function f is continuous, so by Theorem 16.2 is must have a fixed point, i.e. a pair of strategies $(x, y) \in X \times Y$ such that $f(x, y) = (x, y)$.

Further observe that there has to exist $i \in \{1, \dots, m\}$ such that $x_i > 0$ and $s_i(x, y) = 0$, since otherwise

$$p(x,y) = \sum_{k=1}^m x_k p(e_k^m, y) > \sum_{k=1}^m x_k p(x,y) = p(x,y).$$

Therefore, and since (x, y) is a fixed point,

$$x_i = \frac{x_i + s_i(x,y)}{1 + \sum_{k=1}^m s_k(x,y)}$$

and thus

$$\sum_{k=1}^m s_k(x,y) = 0.$$

This means that for $k = 1, \dots, m$, $s_k(x, y) = 0$, and therefore

$$p(x, y) \geq p(e_k^m, y).$$

It follows that

$$p(x, y) \geq p(x', y) \quad \text{for all } x' \in X.$$

An analogous argument shows that $q(x, y) \geq q(x, y')$ for all $y' \in Y$, so (x, y) must be an equilibrium. □

Our requirement that a bimatrix game has a finite number of actions is crucial for this result. This can be seen very easily by considering a game where the set of actions of each player is the set of natural numbers, and players get a payoff of 1 if they choose a number that is greater than the one chosen by the other player, and zero otherwise.

16.2 The complexity of finding an equilibrium

The proof of Theorem 16.1 relies on fixed points of a continuous function and does not give rise to a finite method for finding an equilibrium. Quite surprisingly, equilibrium computation turns out to be more or less a combinatorial problem.

Define the **support** of strategy $x \in X$ as $S(x) = \{i \in \{1, \dots, m\} : x_i > 0\}$, and that of strategy $y \in Y$ as $S(y) = \{j \in \{1, \dots, m\} : y_j > 0\}$. It is easy to see that a mixed strategy is a best response if and only if all pure strategies in its support are best responses: if one of them was not a best response, then the payoff could be increased by reducing the probability of that strategy, and increasing the probabilities of the other strategies in the support appropriately. In other words, randomization over the support of an equilibrium does not happen for the player's own sake, but to allow the other player to respond in a way that sustains the equilibrium.

It also follows from these considerations that finding an equilibrium boils down to finding its supports. Once the supports are known, the precise strategies can be computed by solving a set of equations, which in the two-player case are linear. For supports of sizes k and ℓ, there is one equation for each player stating that the probabilities sum up to one, and $k - 1$ or $\ell - 1$ equations, respectively, stating that the expected payoff is the same for every pure strategy in the support. Solving these $k + \ell$ equations in $k + \ell$ variables yields k values for player 1 and ℓ values for player 2. If the solution corresponds to a strategy profile and expected payoffs are maximized by the pure strategies in the support, then an equilibrium has been found. An equilibrium with supports of size two in the game of chicken would have to satisfy $x_1 + x_2 = 1$, $y_1 + y_2 = 1$, $2x_1 + 1x_2 = 3x_1 + 0x_2$, and $2y_1 + 1y_2 = 3y_1 + 0y_2$. The unique solution, $x_1 = x_2 = y_1 = y_2 = 1/2$, also satisfies the additional requirements and therefore is an equilibrium. No equilibrium with full supports exists in the prisoner's dilemma, because the corresponding system of equalities does not have a solution.

A procedure for finding an equilibrium, and in fact all equilibria, is to iterate over all possible supports and check for each of them whether there is an equilibrium with that support. The running time of this method is finite, but clearly exponential in general. It is natural to ask whether there is a hardness result that stands in the way of a polynomial-time algorithm. While the equilibrium condition can easily be verified for a given pair of strategies, which implies membership in NP, the notion of NP-hardness seems inappropriate: equilibria always exist and the decision problem is therefore trivial. On the other hand, NP-hardness follows immediately if the problem is modified slightly to obtain a non-trivial decision problem.

Theorem 16.3. *Given a bimatrix game, it is NP-complete to decide whether it has at least two equilibria; an equilibrium in which the expected payoff of the row player is at least a given amount; an equilibrium in which the expected sum of payoff of the two players is a least a given amount; an equilibrium with supports of a given minimum size; an equilibrium whose support includes a given pure strategy; or an equilibrium whose support does not include a given pure strategy.*

Theorem 16.1 is an existence statement, but its proof does not identify an equilibrium. We now consider an algorithm that searches the possible supports in an organized way. It provides an alternative, combinatorial, proof of the existence of an equilibrium.

16.3 Symmetric games

A **symmetric game** is one in which $Q = P^T$. An equilibrium can be identified using the following lemma.

Lemma 16.4. *Consider the symmetric game with payoff matrix P. Suppose x, z are such that*

$$x \geq 0,\ \ x \neq 0,\ \ z \geq 0,\ \ Px + z = 1 \text{ and } x^T z = 0. \tag{16.1}$$

Then a symmetric equilibrium is one in which both players use the same mixed strategy, $\bar{x} = x / \sum_{i=1}^m x_i$.

Proof. Suppose the column player uses mixed strategy $\bar{x}$. For any strategy $\tilde{x} \in X$

$$p(\tilde{x}, \bar{x}) = \tilde{x}^T P \bar{x} = \frac{1}{\sum_{i=1}^m x_i} - \frac{\tilde{x}^T z}{\sum_{i=1}^m x_i}.$$

So $p(\tilde{x}, \bar{x}) \leq 1 / \sum_{i=1}^m x_i$, with equality if $\tilde{x} = \bar{x}$. $\square$

16.4 Lemke-Howson algorithm for a symmetric game

Returning to (16.1), we now describe an algorithm for finding an equilibrium of a symmetric game. The basic feasible solution $v_0 = (x, z) = (0, 1)$ satisfies all conditions of Lemma 16.1 except for $1^\top x > 0$. For convenience in describing an algorithm, let us say that strategy i is **represented** if either it is a best response ($z_i = 0$) or is not used ($x_i = 0$). It is **twice-represented** if $x_i = z_i = 0$. It is **missing** if $x_i z_i > 0$.

Let $S = \{(x, z) : Px + z = 1,\ x \geq 0,\ z \geq 0\}$. Let us assume no degeneracy, so that every vertex of S has exactly m neighbours. Fix a strategy, say $i = 1$, and consider the set V, consisting of all vertices of S in which every strategy is represented or twice-represented, except possibly strategy 1 (which may be missing). Let v_0 be the vertex $(x, z) = (0, 1)$. Note that $v_0 \in V$ since in v_0 every strategy is represented.

There is exactly one neighbouring vertex to v_0 that is also in V. This is the vertex reached by increasing x_1 from 0. As we increase x_1 from 0 some z_j becomes 0. (This can be done in a tableau, similarly to the simplex algorithm). If it is z_1 that becomes 0 then we have a solution to (16.1). Otherwise, we now have $x_1 z_1 > 0$ and $x_j = z_j = 0$; so increase x_j until some other variable becomes 0. Continue in this manner. After each pivot, in which some variable has been increased from 0, we are at a node where $x_j = z_j = 0$ for some k, and still $x_1 z_1 > 0$, and for all other k we have $x_k z_k = 0$. Either x_j or z_j has just become 0. At the next step we increase the other member of this pair, until some other variable becomes 0. If this is x_1 or z_1 we now have a solution to (16.1). Otherwise we continue.

This algorithm (**Lemke-Howson**, 1964) must terminate in an equilibrum. The reasoning is cute. Firstly, note that since S is bounded it must be the case that as we increase a nonbasic variable from 0, some basic variable must decrease and eventually reach 0, completing a pivot step from one vertex to another. Second, there is only one way to leave v_0 and remain in V; similarly, having reached v_i there is only one way to leave v_i and remain in V without returning to the vertex from which the path just arrived. Thus, the path never revisits a vertex. Thirdly, there are a finite number of vertices in V, so eventually the algorithm can go no further. The only way this can happen is to have reached a vertex at which no strategy is twice-represented. Since this is not v_0 it must be one for which $1^\top x > 0$.

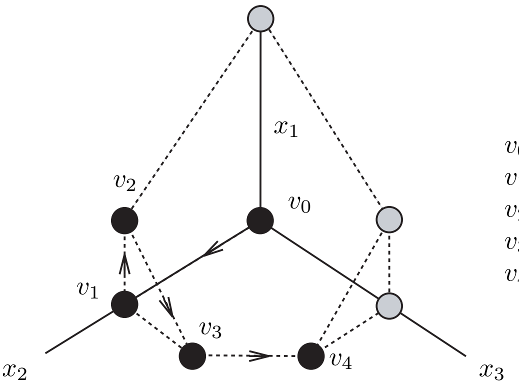
Example 16.5. Suppose

$$P = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{pmatrix}.$$

We start at $v_0 = (x, z) = (0, 0, 0, 1, 1, 1)$. If we choose $i = 2$ then the unique neighbouring member of V which has strategy 2 missing is obtained by increasing x_2, to reach $v_1 = (0, \frac{1}{3}, 0, 0, 1, \frac{1}{3})$. In moving from v_0 to v_1, the variable z_1 has decreased to 0 and strategy 1 is now twice-represented. So let us now increase x_1 and move to $v_2 = (\frac{1}{6}, \frac{1}{3}, 0, 0, 1, 0)$. Now z_3 has been decreased to 0 and strategy 3 is twice-represented, so we increase x_3 to reach $v_3 = (0, \frac{1}{3}, \frac{1}{6}, 0, \frac{1}{2}, 0)$. Now x_1 has decreased to 0, strategy 1 is again twice-represented, so we increase z_1, to reach $v_4 = (0, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}, 0, 0)$. Now z_2 is decreased to 0, all strategies are represented, and $x^T z = 0$. We have the equilibrium $\bar{p} = (0, \frac{1}{3}, \frac{2}{3})$.

As we move amongst vertices in V we are at each step increasing some variable x_i

(or z_i) associated with an twice-represented strategy i, which is complementary to the variable z_i or (or x_i) that was decreased to 0 at the previous step.



	x_1	x_2	x_3	z_1	z_2	z_3
v_0 :	0	0	0	1	1	1
v_1 :	**0**	$\frac{1}{3}$	0	**0**	1	$\frac{1}{3}$
v_2 :	$\frac{1}{6}$	$\frac{1}{3}$	**0**	0	1	**0**
v_3 :	**0**	$\frac{1}{3}$	$\frac{1}{6}$	**0**	$\frac{1}{2}$	0
v_4 :	0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	0	0

In this example there is only one equilibrium. If we had started with $i = 1$ we would have reached v_4 along a different path.

The algorithm will find one equilibrium, but if there is more than one it cannot guarantee to find them all. Starting with different i to be dropped we might reach the same equilibrium or a different equilibrium. If we start at an one equilibrium we will follow a path to a different equilibrium or to v_0.

There is an interesting corollary of this analysis.

Corollary 16.6. *A nondegenerate bimatrix game has an odd number of Nash equilibria.*

Proof. Let V be the set of vertices in which only Player I's first strategy might be missing (i.e. such that $x_1 z_1 > 0$). Every equilibrium of $P \times Q$ is a member of V (since equilibriums are vertices for which all strategies are represented). In the graph formed by vertices in V, each vertex has degree 1 or 2. So this graph consists of disjoint paths and cycles. The endpoints of the paths are the Nash equilibriums and the special vertex $(x, y) = (0, 0)$. There are an even number of endpoints, so the number of Nash equilibria must be odd. □

17 Linear complimentarity problem

17.1 Linear complementarity problem

The **linear complementarity problem** (LCP) unifies linear programing, quadratic programing and two-person non-zero sum games (**bimatrix games**). Let M be a $n \times n$ matrix, and $q \in \mathbb{R}^n$. The LCP is to find $z, w \in \mathbb{R}^n$, such that

$$w - Mz = q, \quad z, w \geq 0 \quad \text{and } z^\top w = 0. \tag{17.1}$$

Note that nothing is to be maximized or minimized in this problem.

17.2 Quadratic programming as a LCP

Let D be a positive definite symmetric matrix and consider

$$\begin{aligned} \text{QP}: \quad & \text{minimize } c^\top x + \tfrac{1}{2} x^\top D x \\ & \text{subject to } Ax \geq b, \quad x \geq 0. \end{aligned}$$

Consider minimizing over non-negative x and v, the Lagrangian

$$L = c^\top x + \tfrac{1}{2} x^\top D x + \bar{y}^\top (b - Ax + v).$$

From the Lagrangian sufficiency theorem we see that $(\bar{x}, \bar{v})$ minimizes L if

$$\begin{gathered} A\bar{x} - b = \bar{v}, \quad \frac{\partial}{\partial x} L = c + D\bar{x} - A^\top \bar{y} = \bar{u}, \\ \bar{y}^\top \bar{v} = 0, \quad \bar{x}^\top \bar{u} = 0, \quad \bar{x},\, \bar{v},\, \bar{y},\, \bar{u} \geq 0. \end{gathered}$$

So $\bar{x}$ is an optimal solution to QP if there exist vectors $\bar{x}$, $\bar{y}$, $\bar{u}$ and $\bar{v}$ such that

$$\begin{gathered} \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} - \begin{pmatrix} D & -A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} c \\ -b \end{pmatrix} \\ \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} \geq 0 \quad \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \geq 0 \quad \text{and} \quad \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix}^\top \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = 0. \end{gathered}$$

This defines a LCP, whose solution is an optimal solution to QP. A linear programming problem is the special case $D = 0$.

17.3 Knapsack as a LCP

Suppose one is given item of sizes $\{a_1, \dots, a_n\}$ and wishes to find a subset which exactly fill a knapsack of size B.

$$\text{Find } x : a^T x = B, \quad x_i \in \{0,1\},\ i = 1, \dots, n. \tag{17.2}$$

The constraint $x_i \in \{0,1\}$ can be written as $w_i = 1 - x_i$ and $x_i w_i = 0$. This makes (17.2) equivalent to the LCP of finding $x, w \geq 0$ such that $x^T w = 0$ and

$$\begin{pmatrix} w_1 \\ \vdots \\ w_n \\ w_{n+1} \\ w_{n+2} \end{pmatrix} - \begin{pmatrix} -1 & \cdots & 0 & 0 & 0 \\ \vdots & \cdots & \vdots & 0 & 0 \\ 0 & \cdots & -1 & 0 & 0 \\ & a^T & & -\alpha & 0 \\ & -a^T & & 0 & -\beta \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ x_{n+1} \\ x_{n+2} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ -B \\ B \end{pmatrix}$$

where α and β are any positive numbers. Notice that the constaints in the last two rows force $a^T x = B$.

17.4 Lemke's algorithm

Lemke's algorithm finds a solution to a linear complementarilty problem. If $q > 0$ then a solution to the LCP is $w = q$ and $z = 0$. So suppose $q \not> 0$. Let $d = (1, 1, \dots, 1)^\top$, introduce a new variable, λ, and consider

$$w - Mz - \lambda d = q.$$

For $\lambda > 0$ large enough, we have the solution $w = q + \lambda d \geq 0$, and $z = 0$. Imagine decreasing λ from a large value until some component of w becomes 0. At this point (where $\lambda = -\min_i q_i = -q_k$) we have $w_k = z_k = 0$, and complementary slackness, i.e. $w_i z_i = 0$ for all i. We have found an intial basic feasible solution.

Our next step is to move to a neighbouring basic feasible solution by increasing z_k (i.e. we perform a pivot step that puts z_k into the basis). Suppose that as we increase z_k one of the existing basic variables decreases, eventually reaches 0 and so must leave the basis. (If this does not happen then the algorithm fails. But there are special cases for which this cannot happen, such as the bimatrix games.)

- If the departing basic variable is λ then we now have a solution to the LCP.
- If the departing basic variable is not λ then we are left with a new BFS in which $w_\ell = z_\ell = 0$ for some ℓ.

We continue in this fashion, moving amongst solutions that are always complementary slack, i.e. $w_i z_i = 0$ for all i, and such that $\lambda > 0$ and $w_\ell = z_\ell = 0$ for some ℓ. After each pivot, we note which which of the variables it was, w_ℓ or z_ℓ, that has just left the basis, and then increase the other one — continuing in this fashion until eventually it is λ that leaves the basis. At that point we have a solution to the LCP. Note that there are only a finite number of complementary bases and they do not repeat. Each BFS that we can reach has exactly two neighbouring BFS, one of which precedes it and one of which follows it. Thus no BFS can be reached more than once and so λ must eventually leave the basis.

17.5 Complementary pivoting

A general bimatrix game can be recast as symmetric game with $m+n$ pure strategies and payoff matrices

$$\bar{P} = \begin{pmatrix} 0 & P \\ Q^\top & 0 \end{pmatrix}, \quad \bar{Q} = \bar{P}^\top = \begin{pmatrix} 0 & Q \\ P^\top & 0 \end{pmatrix}. \tag{17.3}$$

Lemma 16.4 may be reformulated as follows.

Lemma 17.1. *Consider the symmetric game with payoff matrix $\bar{P}$ given in (17.3). Suppose $s = (x, y)$ such that $s \geq 0$, $\bar{P}s \leq 1$, $s^\top(1 - \bar{P}s) = 0$, $1^\top x > 0$, and $1^\top y > 0$. Then $s/1^\top s$ is a symmetric equilibrium for this symmetric game, and $\bar{x} = x/1^\top x$ and $\bar{y} = y/1^\top y$ are an equilibrium pair for the original nonsymmetric game.*

We can implement the Lemke-Howson algorithm by a process of **complementary pivoting** in two tableaus. Adding slack variables $z \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$ turns the best response conditions into $Q^T x + w = 1$, $Py + z = 1$, $x^T z = y^T w = 0$, where $x, y, z, w \geq 0$.

Notice that we can write the conditions for an equilibrium as in (17.1):

$$\begin{pmatrix} z \\ w \end{pmatrix} - \begin{pmatrix} 0 & -P \\ -Q^\top & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1_n \\ 1_m \end{pmatrix},$$

$$\begin{pmatrix} x \\ y \end{pmatrix} \geq 0, \quad \begin{pmatrix} z \\ w \end{pmatrix} \geq 0, \quad \begin{pmatrix} x \\ y \end{pmatrix}^\top \begin{pmatrix} z \\ w \end{pmatrix} = 0, \quad x \neq 0, \ y \neq 0.$$

The pair (x, y) then is said to be completely labelled if and only if $x^T z = 0$ and $y^T w = 0$. We might start with $x = y = 0$. We pick some i and increase x_i from 0 so now $x_i > 0$ and $z_i > 0$; we say label i is being dropped. When x_i is increased as much as possible, a different constraint starts to hold with equality, for example, $w_j = 0$; we say j is picked up. Since now $y_j = w_j = 0$ this label is twice represented. So the next step is to increase y_j from 0. This might cause some y_k or z_k to reach 0, so k is picked up. From this point we increase w_k or x_k, respectively. We continue in this way until label i is again picked up, which means that (x, y) is again fully labelled, but now $x \neq 0$ and $y \neq 0$. This procedure can be carried out through alternatively pivoting in two tableaus, similarly to pivoting in the simplex method.

Consider for example the bimatrix game given by

$$P = \begin{pmatrix} 3 & 3 \\ 2 & 5 \\ 0 & 6 \end{pmatrix} \qquad \text{and} \qquad Q = \begin{pmatrix} 3 & 2 \\ 2 & 6 \\ 3 & 1 \end{pmatrix},$$

Indexing z and w by $M = \{1, \ldots, m\}$ and $N = \{m+1, \ldots, m+n\}$, respectively, the constraint $Q^T x + w = 1$ can be written in tableau form as follows:

x_1	x_2	x_3	w_4	w_5	
3	2	3	1	0	1
2	6	1	0	1	1

Assume that label 2 is dropped by increasing x_2 from 0. By pivoting we obtain the following tableau:

$\frac{7}{3}$	0	$\frac{8}{3}$	1	$-\frac{1}{3}$	$\frac{2}{3}$
$\frac{1}{3}$	1	$\frac{1}{6}$	0	$\frac{1}{6}$	$\frac{1}{6}$

The second row now corresponds to variable x_2 that has entered the basis. On the other hand, variable w_5 has left the basis. We thus want to turn to the constraint $Py + z = 1$ and drop the duplicate label 5. The initial tableau for this constraint looks as follows:

y_4	y_5	z_1	z_2	z_3	
3	3	1	0	0	1
2	5	0	1	0	1
0	6	0	0	1	1

By pivoting on the second column, corresponding to y_5, and on the third row, we pick up label 3 and obtain the following tableau:

3	0	1	0	$-\frac{1}{2}$	$\frac{1}{2}$
2	0	0	1	$-\frac{5}{6}$	$\frac{1}{6}$
0	1	0	0	$\frac{1}{6}$	$\frac{1}{6}$

Pivoting one more time in each of the two polytopes, we drop label 3 to pick up label 4:

$\frac{7}{8}$	0	1	$\frac{3}{8}$	$-\frac{1}{8}$	$\frac{1}{4}$
$\frac{3}{16}$	1	0	$-\frac{3}{48}$	$\frac{3}{16}$	$\frac{1}{8}$

and then drop label 4 to pick up label 2:

0	0	1	$-\frac{3}{2}$	$\frac{3}{4}$	$\frac{1}{4}$
1	0	0	$\frac{1}{2}$	$-\frac{5}{12}$	$\frac{1}{12}$
0	1	0	0	$\frac{1}{6}$	$\frac{1}{6}$

At this point we have a fully labelled pair. The final tableaus are the final two above. Reading off the values of x and y from the last column of each tableau and scaling them appropriately yields the equilibrium $x = (0, 1/3, 2/3)$, $y = (1/3, 2/3)$.

17.6 Sperner's lemma

We have seen two approaches to proving that a Nash equilibrium of a bimatrix game exists: Brouwer's fixed point theorem and the path following parity argument used in the Lemke-Howson algorithm. They can be connected via **Sperner's lemma**.

Sperner's lemma is a combinatorial version of Brouwer's fixed point theorem. Consider the triangulation shown in Figure 17.6. The colour of each vertex x depends on whether $f(x)$ lies to the northeast, northwest, or south of x. Along the base, there are always an odd number of edges with differently coloured endpoints, shown here black and white. Each small triangle has either 0, 1 or 2 such edges. Coming in from the outside and then entering and leaving each triangle through a side of black and white endpoints, we must eventually leave through the base, or reach a vertex with just one such edge. That is a triangle whose vertices have all three colours (a rainbow triangle). The path following is reminescent of the Lemke-Howson algorithm. Sperner's lemmas says that there must always be a rainbow triangle. To prove Brouwer's fixed point theorem one imagines the triangulation becoming increasingly fine. In the limit, rainbow triangles become fixed points.

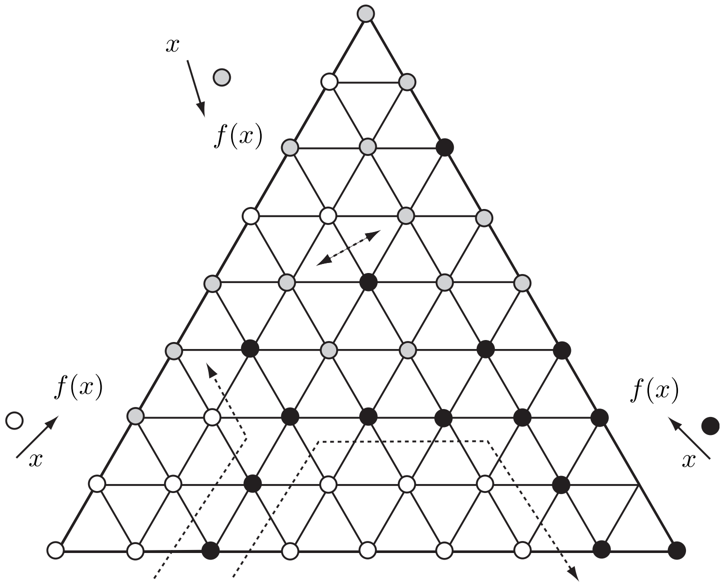


Figure 21: Sperner's lemma. Along each side of the large triangle there is an odd number of edges whose two endpoints are differently coloured.Entering along the base across such an edge and following a path crossing edges which are similarly coloured, we must, with at least one choice of initial entry edge, reach a rainbow triangle. Notice that this also proves that the number of rainbow triangles is odd. In this example there are 3 rainbow triangles. Only one is found by a following a path entering at the base.

18 Cooperative Games

18.1 Coalitional games

A **coalitional game** is given by a set $N = \{1, \ldots, n\}$ of **players**, and a valuation or **characteristic function** $v : 2^N \to \mathbb{R}$ that maps each coalition of players to its **value**, the joint payoff the coalition can obtain by working together. The payoff can be distributed arbitrarily among its members due to their having **transferable utility** (or payoff). We are interested in which coalitions of players are likely to form and how payoff is distributed so that their members are satisfied.

For a given game (N, v), a vector of payoffs $x \in \mathbb{R}^n$ is said to satisfy **(economic) efficiency** if $\sum_{i \in N} x_i = v(N)$ and **individual rationality** if $x_i \geq v(\{i\})$ for $i = 1, \ldots, n$. The first condition intuitively ensures that no payoff is wasted, while the second condition ensures that each player obtains at least the same payoff it would be able to obtain on its own. A payoff vector that is both efficient and individually rational is also called an **imputation**. The set of imputations might be empty. However, it is non-empty for a **superadditive game**, i.e. one in which $v(S \cup T) \geq v(S) + v(T)$ for all disjoint subsets S and T.

18.2 The core

Efficiency and individual rationality may not be enough to guarantee a stable outcome. For any two imputations x and y, $\sum_{i \in N} x_i = \sum_{i \in N} y_i = v(N)$, so $y_i > x_i$ for some $i \in N$ implies that $y_j < x_j$ for some other $j \in N$. However, there could be some coalition $S \subseteq N$ such that $y_i > x_i$ for all $i \in S$. If in addition $\sum_{i \in S} y_i \leq v(S)$, the members of S could increase their respective payoffs by exiting from the grand coalition, forming the coalition S, and distributing the payoff thus obtained according to y. The core is the set of imputations that are stable against this kind of action. Formally, imputation x is in the core of game (N, v) if $\sum_{i \in S} x_i \geq v(S)$ for all $S \subseteq N$.

Consider a situation where $n \geq 2$ members of an expedition have discovered a treasure, and any pair of them can carry one piece of the treasure back home. This situation can be modeled by a coalitional game (N, v) where $N = \{1, \ldots, n\}$ and $v(S) = |S|/2$ if $|S|$ is even and $v(S) = (|S| - 1)/2$ if $|S|$ is odd. The core then contains all imputations if $n = 2$, the single imputation $(1/2, \ldots, 1/2)$ if $n \geq 4$ is even, and is empty if n is odd. The latter can be shown using the following characterization of games with a non-empty core.

Call a function $\lambda : 2^N \to [0, 1]$ **balanced** if for every player the weights of all coalitions containing that player sum to 1, i.e. if for all $i \in N$, $\sum_{S \subseteq N \setminus \{i\}} \lambda(S \cup \{i\}) = 1$. A game (N, v) is called **balanced** if for every balanced function λ, $\sum_{S \subseteq N} \lambda(S) v(S) \leq v(N)$. The intuition behind this definition is that each player allocates one unit of time among the coalitions it is a member of, and each coalition earns a fraction of its value proportional to the minimum amount of time devoted to it by any of its members. Balancedness of a collection of weights imposes a feasibility condition on players' allocations of time, and a game is balanced if there is no feasible allocation that yields more than $v(N)$.

Theorem 18.1 (Bondareva 1963, Shapley 1967)**.** *A game has a non-empty core if and only if it is balanced.*

Proof. The core of a game (N, v) is non-empty if and only if the linear program to

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in N} x_i \\ \text{subject to} \quad & \sum_{i \in S} x_i \geq v(S) \quad \text{for all } S \subseteq N \end{aligned}$$

has an optimal solution with value $v(N)$. This linear program has the following dual:

$$\begin{aligned} \text{maximize} \quad & \sum_{S \subseteq N} \lambda(S) v(S) \\ \text{subject to} \quad & \sum_{S \subseteq N, i \in S} \lambda(S) = 1 \quad \text{for all } i \in N \\ & \lambda(S) \geq 0 \quad \text{for all } S \subseteq N, \end{aligned}$$

where $\lambda : 2^N \to [0, 1]$. Note that λ is feasible for the dual if and only if it is a balanced function. Both primal and dual are feasible, so by strong duality their optimal objective values are the same. This means that the core is non-empty if and only if $\sum_{S \subseteq N} \lambda(S) v(S) \leq v(N)$ for every balanced function λ. □

To see that the core of our example game is empty if n is odd, define $\lambda : 2^N \to [0, 1]$ such that $\lambda(S) = 1/(n-1)$ if $|S| = 2$ and $\lambda(S) = 0$ otherwise. Then, for all $i \in N$, $\sum_{S \subseteq N \setminus \{i\}} \lambda(S \cup \{i\}) = 1$, because each player is contained in exactly $(n-1)$ sets of size 2. Moreover, $\sum_{S \subseteq N} \lambda(S) v(S) = n(n-1)/2 \cdot 1/(n-1) = n/2$, which is greater than $v(N)$ if n is odd.

18.3 The nucleolus

If the core is empty then one might consider weakening the requirement that no coalition should be able to gain, and instead look for an efficient payoff vector that minimizes the possible gain over all coalitions. This can intuitively be interpreted as minimizing players' incentive to deviate from the solution by forming another coalition, or as a natural notion of fairness when distributing the joint payoff $v(N)$ among the players.

To this end, define the **excess** $e(S, x)$ of coalition $S \subseteq N$ for payoff vector x as its gain from leaving the grand coalition, i.e. $e(S, x) = v(S) - \sum_{i \in S} x_i$. Excess may be thought of as dissatisfaction, because it is the difference between what the coalition could earn on its own and what it receives with payoff vector x. For a given vector x, let

$$E(x) = (e(S_1^x, x), e(S_2^x, x), \ldots, e(S_{2^n-1}^x, x))$$

where $S_1^x, \ldots, S_{2^n-1}^x$ is an ordering of the coalitions in decreasing order of excess, so components of $E(x)$ are nonincreasing. If $E(x) \le 0$ then x is in the core. (No coalition in positively dissatisified.)

The **nucleolus** is defined as the set of efficient payoff vectors x for which $E(x)$ is lexicographically minimal. Note that x does not need to be an imputation since we do not require $x_i \ge v(\{i\})$. The following algorithm proves that a lexicographic minimum exists and shows how to compute it.

Step 1. Solve a linear program to minimize the maximum excess.

$$\begin{array}{lll} \text{minimize} & \epsilon & \\ \text{subject to} & \epsilon \ge v(S) - \sum_{i \in S} x_i & \text{for all } S \subset N \\ & 0 = v(N) - \sum_{i \in N} x_i. & \end{array} \tag{P_1}$$

Suppose the optimal value is ϵ_1. Let Δ_1 be the set of coalitions for which equality holds in (P_1) in every optimal solution. Clearly if x is in the nucleolus then $E_i(x) = \epsilon_1$ for all $i = 1, \ldots, |\Delta_1|$. Notice that $|\Delta_1| \ge 1$. This is because if each coalition S involved in the inequality constraint satisfied $\epsilon_1 > v(S) - \sum_{i \in S} x_i$ for some optimal solution, x, then we could take the average of all optimal solutions, say $\bar{x}$, and deduce that for this feasible $\bar{x}$ all the inequality constraints would be strict, contradicting ϵ_1 being the optimal value. Now if $|\Delta_1| < 2^n - 2$ go to step 2. Otherwise stop.

Step k.

$$\begin{array}{llll} \text{minimize} & \epsilon & & \\ \text{subject to} & \epsilon_1 & = v(S) - \sum_{i \in S} x_i & \text{for all } S \in \Delta_1 \\ & \vdots & & \\ & \epsilon_{k-1} & = v(S) - \sum_{i \in S} x_i & \text{for all } S \in \Delta_{k-1} \\ & \epsilon & \ge v(S) - \sum_{i \in S} x_i & \text{for all } S \subset N,\ S \notin \Delta_1 \cup \cdots \cup \Delta_{k-1} \\ & 0 & = v(N) - \sum_{i \in N} x_i. & \end{array} \tag{P_k}$$

Suppose the optimal value is ϵ_k. Let Δ_k be the set of all coalitions in which equality newly holds in (P_k) in every optimal solution. By a similar argument as we used for $|\Delta_1| \ge 1$, we have $|\Delta_k| \ge 1$. If $|\Delta_1 \cup \cdots \cup \Delta_k| < 2^n - 2$ then go to step $k+1$. Otherwise stop.

The algorithm terminates by the end of step $2^n - 2$. Also notice that consideration of this algorithm prove the following theorem.

Theorem 18.2. *The nucleolus of any coalitional game is a singleton.*

Proof. If x and y are both in the nucleolus then both are optimal solutions at step 1. This implies that for all $S \in \Delta_1$, we must have $v(S) - \sum_{i \in S} x_i = \epsilon_1 = v(S) - \sum_{i \in S} y_i$. The same reasoning holds at each step k. At some point we will consider $S = \{i\} \in \Delta_j$, and then it must be that $v(\{i\}) - x_i = \epsilon_j = v(\{i\}) - y_i$ and so $x_i = y_i$. $\square$

Example 18.3. Consider a game with $N = \{1, 2, 3\}$ and characteristic function values

$$\begin{gathered} v(\{1\}) = 1 \quad v(\{2\}) = 2 \quad v(\{3\}) = 1 \\ v(\{1,2\}) = 2 \quad v(\{1,3\}) = 3 \quad v(\{2,3\}) = 5 \quad v(\{1,2,3\}) = 4. \end{gathered} \tag{18.1}$$

To find the nucleolus of this game, we first

$$\begin{aligned} &\text{minimize} && \epsilon \\ &\text{subject to} && x_1 \geq 1 - \epsilon, \quad x_2 \geq 2 - \epsilon, \quad x_3 \geq 1 - \epsilon \\ &&& x_1 + x_2 \geq 2 - \epsilon, \quad x_1 + x_3 \geq 3 - \epsilon \\ &&& x_2 + x_3 \geq 5 - \epsilon, \quad x_1 + x_2 + x_3 = 4. \end{aligned}$$

It is easily verified that $\epsilon = 1$ for the feasible solution $x = (0, 1, 3)$. By adding the constraints for $\{1\}$ and $\{2, 3\}$ and subtracting the constraint for $\{1, 2, 3\}$ we see that $\epsilon \geq 1$, so the solution must be optimal. For $\epsilon = 1$, the constraints for $\{1\}$ and $\{2, 3\}$ have to hold with equality in every optimal solution, and the constraints for $\{3\}$, $\{1, 2\}$, and $\{1, 2, 3\}$ become redundant. Thus $x_1 = 0$, $x_2 \geq 1$, $x_3 \geq 2$, and $x_2 + x_3 = 4$. We now

$$\begin{aligned} &\text{minimize} && \epsilon \\ &\text{subject to} && x_1 = 0, \quad x_2 \geq 2 - \epsilon \\ &&& x_3 \geq 3 - \epsilon, \quad x_2 + x_3 = 4 \end{aligned}$$

and obtain a unique optimal solution, with $\epsilon = 1/2$,

$$x = \left(0, \tfrac{3}{2}, \tfrac{5}{2}\right), \text{ and } E(x) = \left(1, 1, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, 0, -\tfrac{3}{2}\right).$$

18.4 Nucleolus in terms of objections

Suppose a subset of players, S, has an objection to payoff x because S would have lesser excess (be less dissatisfied) if the payoff were changed to y, i.e. $e(S, x) > e(S, y)$. However, a different coalition T might counter-object on the grounds that

$$e(T, y) > e(T, x) \quad \text{and} \quad e(T, y) \geq e(S, x).$$

That is, "Our dissatisfation would increase, and would be as great as that which you are experiencing." It can shown that the nucleolus is the unique payoff such that every objection has a counter-objection.

19 Bargaining problems

19.1 The Shapley value

A different notion of fairness in distributing the joint payoff of a coalition among its members was proposed by Shapley, starting from a set of axioms. Call player $i \in N$ a **dummy** if its contribution to every coalition is exactly its value, i.e. if $v(S \cup \{i\}) = v(S) + v(\{i\})$ for all $S \subseteq N \setminus \{i\}$. Call two players $i, j \in N$ **interchangeable** if they contribute the same to every coalition, i.e. if $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq N \setminus \{i, j\}$. Let a *solution* be a function $\phi : \mathbb{R}^{2^n} \to \mathbb{R}^n$ that maps every characteristic function v to an efficient payoff vector $\phi(v)$. Solution ϕ is said to satisfy

- **dummies** if $\phi_i(v) = v(\{i\})$ whenever i is a dummy;
- **symmetry** if $\phi_i(v) = \phi_j(v)$ whenever i and j are interchangeable; and
- **additivity** if $\phi(v + w) = \phi(v) + \phi(w)$.

It turns out that there is a unique solution satisfying these axioms.

Theorem 19.1. *The **Shapley value**, given by*

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \big(v(S \cup \{i\}) - v(S)\big),$$

is the unique solution that satisfies dummies, symmetry, and additivity.

The Shapley value of player i can be interpreted as its average contribution over all possible sequences in which the players can join the grand coalition. For the three-player game with characteristic function given in (18.1), we for example have

$$\begin{aligned}\phi_1(v) &= \frac{0!2!}{3!}(v(\{1\}) - v(\emptyset)) + \frac{1!1!}{3!}(v(\{1,2\}) - v(\{2\})) + \\ &\quad \frac{1!1!}{3!}(v(\{1,3\}) - v(\{3\})) + \frac{2!0!}{3!}(v(\{1,2,3\}) - v(\{2,3\})) \\ &= \frac{1}{3}\,1 + \frac{1}{6}\,0 + \frac{1}{6}\,2 + \frac{1}{3}\,(-1) = \frac{1}{3}.\end{aligned}$$

19.2 Shapley value in terms of objections

The Shapley value may also be defined in terms of objections and counter-objections.

If $\phi_j(N) > \phi_j(N - \{i\})$, then player i might threaten player j, "*Give me more or I will leave the coalition and you will lose.*" Player j has a valid counter-objection if he can point out that if he leaves the coalition then i loses just as much:

$$\phi_i(N) - \phi_i(N - \{j\}) \geq \phi_j(N) - \phi_j(N - \{i\})$$

If $\phi_j(N) < \phi_j(N - \{i\})$, player j might threaten player i, "*Give me more or I will convince the others to exclude you and I will be better off.*" Player i has a valid counter-objection if he can point out that if he gets the others to exclude j then i will be better off by at least as much.

If every such objection has a counter-objection, then

$$\phi_i(N) - \phi_i(N - \{j\}) = \phi_j(N) - \phi_j(N - \{i\}).$$

The only solution to this is the Shapley value.

19.3 Bargaining theory

Bargaining theory investigates how agents should cooperate when non-cooperation may result in outcomes that are Pareto dominated. Formally, a (two-player) **bargaining problem** is a pair (F, d) where $F \subseteq \mathbb{R}^2$ is a convex set of **feasible outcomes** and $d \in F$ is a **disagreement point** that results if players fail to agree on an outcome. Here, convexity corresponds to the assumption that any lottery over feasible outcomes is again feasible. A **bargaining solution** then is a function that assigns to every bargaining problem (F, d) a unique element of F.

An example of a bargaining problem is the so-called ultimatum game given by $F = \{(v_1, v_2) \in \mathbb{R}^2 : v_1 + v_2 \leq 1\}$ and $d = (0, 0)$, in which two players receive a fixed amount of payoff if they can agree on a way to divide this amount among themselves. This game has many equilibria when viewed as a normal-form game, since disagreement results in a payoff of zero to both players. Players' preferences regarding these equilibria differ, and bargaining theory tries to answer the question which equilibrium should be chosen. More generally, a two-player normal-form game with payoff matrices $P, Q \in \mathbb{R}^{m \times n}$ can be interpreted as a bargaining problem where $F = \text{conv}(\{(p_{ij}, q_{ij}) : i = 1, \ldots, m,\ j = 1, \ldots, n\})$, $d_1 = \max_{x \in X} \min_{y \in Y} p(x, y)$, and $d_2 = \max_{y \in Y} \min_{x \in X} q(x, y)$, given that $(d_1, d_2) \in F$. Here, $\text{conv}(S)$ denotes the convex hull of set S.

Two kinds of approaches to bargaining exist in the literature: a strategic one that considers iterative procedures resulting in an outcome in F, and an axiomatic one that tries to identify bargaining solutions that possess certain desirable properties. We will focus on the axiomatic approach in this lecture.

19.4 Nash's bargaining solution

For a given bargaining problem (F, d), Nash proposed to

$$\begin{aligned} \text{maximize} \quad & (v_1 - d_1)(v_2 - d_2) \\ \text{subject to} \quad & v \in F \\ & v \geq d. \end{aligned} \tag{19.1}$$

Optimization problem (19.1) thus defines the so-called **Nash bargaining solution**.

Consider for example the two-player game with payoff matrices

$$P = \begin{pmatrix} 0 & 5 \\ 3 & 1 \end{pmatrix} \qquad \text{and} \qquad Q = \begin{pmatrix} 2 & 2 \\ 4 & 0 \end{pmatrix}.$$

The row player can guarantee a payoff of 15/7 by playing the two rows with probabilities 2/7 and 5/7, respectively. The column player can guarantee a payoff of 2 by playing the left column.

The bargaining problem corresponding to this game is shown in Figure 22. The set F is the convex hull of the four payoff vectors $(0, 2)$, $(5, 2)$, $(3, 4)$, and $(1, 0)$, and it contains the feasible set $B = \{v \in F : v \geq d\}$ of (19.1). The disagreement point is $d = (15/7, 2)$.

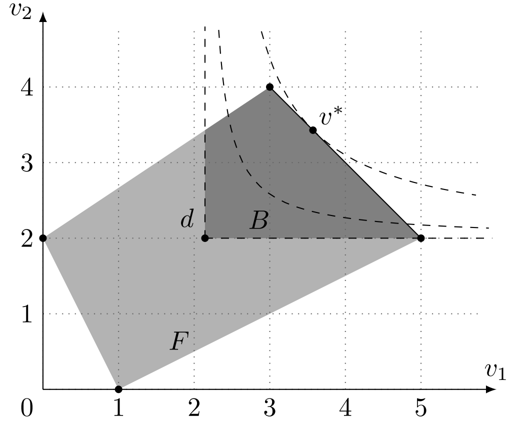


Figure 22: Illustration of the Nash bargaining solution

Level sets of the objective function corresponding to values 0 and 1 and to the optimal value are drawn as dashed curves. The Nash bargaining solution v^* is the unique point in the intersection of F with the optimal level set.

To compute v^*, we first observe that $v^* \in \{(v_1, v_2) : v_2 = 7 - v_1, 3 \leq v_1 \leq 5\}$. The objective function becomes

$$(v_1 - d_1)(v_2 - d_2) = (v_1 - \tfrac{15}{7})(5 - v_1) = \tfrac{50}{7}v_1 - v_1^2 - \tfrac{75}{7},$$

and has a stationary point if $50/7 - 2v_1 = 0$. We obtain $v^* = (25/7, 24/7)$, which is indeed a maximum.

While it is not obvious that maximizing the product of the excess of the two players is a good idea, it turns out that the Nash bargaining solution can be characterized using a set of simple axioms. Bargaining solution f is

1. **Pareto efficient** if $f(F,d)$ is not Pareto dominated in F for any bargaining problem (F,d);
2. **symmetric** if $(f(F,d))_1 = (f(F,d))_2$ for every bargaining problem (F,d) such that $(y,x) \in F$ whenever $(x,y) \in F$ and $d_1 = d_2$;
3. **invariant under positive affine transformations** if $f(F',d') = \alpha \circ f(F,d) + \beta$ for any $\alpha, \beta \in \mathbb{R}^2$ with $\alpha > 0$ and any two bargaining problems (F,d) and (F',d') such that $F' = \{\alpha \circ x + \beta : x \in F\}$ and $d' = \alpha \circ d + \beta$; and
4. **independent of irrelevant alternatives** if $f(F,d) = f(F',d)$ for any two bargaining problems (F,d) and (F',d) such that $F' \subseteq F$ with $d \in F'$ and $f(F,d) \in F'$.

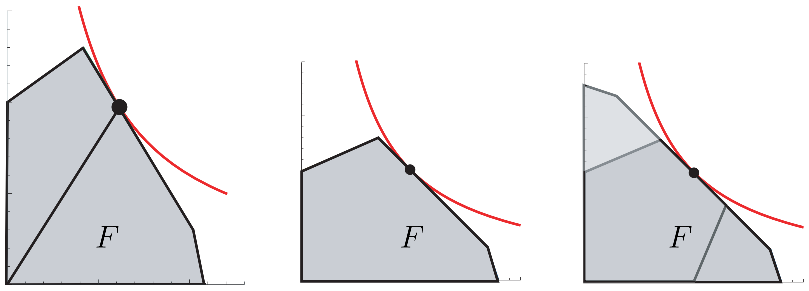
Here, $\circ$ denotes component-wise multiplication of vectors, i.e. $(s \circ t)^T = (s_1 t_1, s_2 t_2)$ for all $s, t \in \mathbb{R}^2$.

In the context of bargaining, Pareto efficiency means that no payoff is wasted, and symmetry is an obvious fairness property. Invariance under positive affine transformations should hold because payoffs are just a representation of the underlying ordinal preferences. The intuition behind independence of irrelevant alternatives is that an outcome only becomes easier to justify as a solution when other outcomes are removed from the set of feasible outcomes.

Theorem 19.2. *Nash's bargaining solution is the unique bargaining solution that is Pareto efficient, symmetric, invariant under positive affine transformations, and independent of irrelevant alternatives.*

"Proof idea" It is easy to check that Nash's bargaining solution satisfies the axioms.

It remains to prove that the axioms force the solution to be Nash's bargaining solution. Assume $d_1 = d_2 = 0$. Touch set F with $v_1 v_2 =$ constant; use axiom 3 to move the point of touching to $(1/2, 1/2)$; add the set obtained by reflecting F in the line from the origin through $(1/2, 1/2)$; use axiom 2 to argue $(1/2, 1/2)$ must be the solution; then use axiom 4 to remove the extra set that was just added.



Proof. We denote the Nash bargaining solution by f^N and begin by showing that it satisfies the axioms. For Pareto efficiency, this follows directly from the fact that the objective function is increasing in v_1 and v_2. For symmetry, assume that $d_1 = d_2$

and let $v^* = (v_1^*, v_2^*) = f^N(F, d)$. Clearly (v_2^*, v_1^*) maximizes the objective function, and by uniqueness of the optimal solution $(v_2^*, v_1^*) = (v_1^*, v_2^*)$ and thus $f_1^N(F, d) = f_2^N(F, d)$. For invariance under positive affine transformations, define F' and d' as above, and observe that $f^N(F', d')$ is an optimal solution of the problem to maximize $(v_1 - \alpha_1 d_1 - \beta_1)(v_2 - \alpha_2 d_2 - \beta_2)$ subject to $v \in F'$, $v_1 \geq d_1$, and $v_2 \geq d_2$. By setting $v' = \alpha \circ v + \beta$, it follows that $f^N(F', d') = \alpha \circ f^N(F, d) + \beta$. For independence of irrelevant alternatives, let $v^* = f^N(F, d)$ and $F' \subseteq F$. If $v^* \in F'$, it remains optimal and thus $v^* = f^N(F', d)$.

Now consider a bargaining solution f that satisfies the axioms, and fix F and d. Let $z = f^N(F, d)$, and let F' be the image of F under an affine transformation that maps z to $(1/2, 1/2)$ and d to the origin, i.e.

$$F' = \{\alpha \circ v + \beta : v \in F, \alpha \circ z + \beta = (1/2, 1/2)^T, \alpha \circ d + \beta = 0\}.$$

Since both f and f^N are invariant under positive affine transformations, $f(F, d) = f^N(F, d)$ if and only if $f(F', 0) = f^N(F', 0)$. It thus suffices to show that $f(F', 0) = (1/2, 1/2)$.

We begin by showing that for all $v \in F'$, $v_1 + v_2 \leq 1$. Assume for contradiction that there exists $v \in F$ with $v_1 + v_2 > 1$, and let $t^\delta = (1 - \delta)(1/2, 1/2)^T + \delta v$. By convexity of F', $t^\delta \in F'$ for $\delta \in (0, 1)$. Moreover, since the objective function has a unique maximum, we can choose δ sufficiently small such that $t_1^\delta t_2^\delta > 1/4 = f^N(F', 0)$, contradicting optimality of $f^N(F', 0)$.

Now let F'' be the closure of F' under symmetry, and observe that for all $v \in F''$, $v_1 + v_2 \leq 1$. Therefore, by Pareto optimality and symmetry of f, $f(F'', 0) = (1/2, 1/2)^T$. Since f is independent of irrelevant alternatives, $f(F', 0) = (1/2, 1/2)^T$ as required. $\square$

20 Social Choice

20.1 Social welfare functions

Social choice theory asks how the possibly conflicting preferences of a set of agents can be aggregated into a collective decision, and in particular which properties the aggregate choice should satisfy and which properties can be satisfied simultaneously. Examples of settings that can be studied in the framework of social choice theory include voting, resource allocation, coalition formation, and matching.

Let $N = \{1, \ldots, n\}$ be a set of agents, or **voters**, and $A = \{1, \ldots, m\}$ a set of alternatives. Assume that each voter i has a strict linear order $\succ_i \in L(A)$ over A, and the goal is to map the profile $(\succ_i)_{i \in N}$ of individual preference orders to a social preference order. This is achieved by means of a **social welfare function** (SWF) $f : L(A)^n \to L(A)$.

When $m = 2$, selecting the social preference order that is preferred by a majority of the voters is optimal in a rather strong sense. A SWF $f : L(A)^n \to L(A)$ is

- **anonymous** if the labelling of the voters is irrelevant;
- **neutral** if the labelling of the alternatives is irrelevant;
- **monotone** if an alternative cannot become less preferred socially when it becomes more preferred by individuals.

Theorem 20.1. *Consider an SWF $f : L(A)^n \to L(A)$, where $|A| = 2$ and n is odd. Then f is the majority rule if and only if it is anonymous, neutral, and monotone.*

Proof sketch. Let $A = \{a, b\}$. By anonymity, the social preference only depends on the number of voters that prefer a to b. By neutrality, the social preference has to change between a preference profile where $\lfloor n/2 \rfloor$ voters prefer a to b and one where $\lceil n/2 \rceil$ voters prefer a to b. By monotonicity, the socially preferred alternative can never change from a to b when the number of voters who prefer a to b increases, so this is actually the unique change, and it follows that f is the majority rule. $\square$

In light of this result, it might seem promising to base the decision on pairwise comparisons of alternatives even when $m > 2$. But this is somewhat problematic, since the pairwise majority relation may contain cycles. In Figure 23 each column lists the pref-

a	b	c
b	c	a
c	a	b

Figure 23: Marquis de Condorcet's paradox, 1786.

erences of a particular voter. It is easily verified that a majority of the voters prefers a over b, a majority prefers b over c, and a majority prefers c over a.

A SWF is said to be

- **Pareto optimal**: if every voter prefers a to b then a is socially preferred to b;
- **independent of irrelevant alternatives** (IIA): the social preference with respect to a and b only depends on individual preferences with respect to a and b, but not on those with respect to other alternatives;
- **dictatorial**: the social preference order is determined by a single voter.

It turns out that dictatorships are the only SWFs for three or more alternatives that are Pareto optimal and IIA.

Theorem 20.2 (Arrow, 1951)**.** *Consider an SWF $f : L(A)^n \to L(A)$, where $|A| \geq 3$. If f is Pareto optimal and IIA, then f is dictatorial.*

Requiring non-dictatorship and Pareto optimality is rather uncontroversial. So IIA must be abandoned. For example, **Kemeny's rule** does this, choosing a social preference order $\succ'$ to maximize the number of agreements with the individual preferences. This maximization problem is NP-hard, but can be written as an integer program.

20.2 Social choice functions

It is often sufficient to identify a single best alternative rather than giving a complete ranking. This is achieved by a **social choice function** (SCF) $f : L(A)^n \to A$. Two familiar SCFs are **plurality**, which chooses an alternative ranked first by the largest number of voters, and **single transferable vote** (STV), which successively eliminates alternatives ranked first by the fewest voters until only one alternative remains.

Consider a situation with three alternatives a, b, and c, and nine voters with preferences as shown on the left of Figure 24. In this situation, plurality selects alternative a because it is ranked first by 4 voters, compared to 3 for c and 2 for b. STV first eliminates alternative b, which is ranked first by only 2 voters. Restricting attention to the remaining alternatives, a is ranked first by 4 voters and c by 5 voters. Alternative a is thus eliminated next, while alternative c remains and is selected.

The graph of the majority relations shown in Figure 24 illustrates that alternative b is a so-called **Condorcet winner**, i.e. it is preferred to any other alternative by a majority of the voters, while alternative a is a **Condorcet loser**, i.e. a majority of voters prefer any other alternative to a. The example of Figure 23 shows that a Condorcet winner or loser need not exist, but it is certainly reasonable to require that a Condorcet winner is selected when it exists, and that a Condorcet loser is never selected. An SCF satisfying the former property is called Condorcet consistent, and the example of Figure 24 shows that that neither plurality nor STV are Condorcet consistent.

4	3	2
a	c	b
b	b	c
c	a	a

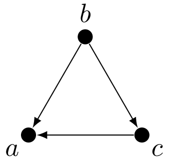


Figure 24: Preferences of three types of voters over three alternatives and a graph of majority relations. At the top of each column of the table is the number of voters having the preference order below. There is a directed edge in the graph from alternative x to alternative y indicates when a majority of the voters prefer x to y.

20.3 Strategic manipulation

In the situation of Figure 24, plurality results in selection of a. This ignores, however, that voters of the third type have an incentive to misrepresent their preferences and claim that they prefer c to b: assuming that ties are broken in favor of c, only a single voter of the third type would have to change its reported preferences in this way to ensure that c is selected instead of a, an outcome this voter prefers. A similar problem exists with STV, since voters of the first type could benefit by pretending their most preferred alternative is b, with the goal of having this alternative selected instead of their least preferred alternative, c. More generally, we say that SCF f is **manipulable** if there exist situations in which by misrepresenting his true preferences agent i can cause f to select an outcome which he prefers to the one which would be selected if he were to give his true preferences. We say 'i can manipulate f'. SCF f is called **strategyproof** if it is not manipulable.

It is easy to see that when there are just two alternatives then majority rule is strategyproof. So when there are more than two alternatives there are two obvious ways to achieve strategyproofness: choose an alternative based on the preferences of a single voter, or ignore all but two alternatives and using majority rule to choose between them. The first case gives a dictatorship; the second case gives a SCF that is not **unanimous** in the sense that there are some alternatives that are not chosen even when all agents rank it top. It turns out that these trivial cases are in fact the only SCFs that are strategyproof.

Theorem 20.3 (Gibbard, 1973; Satterthwaite, 1975)**.** *Consider an SCF $f : L(A)^n \to A$, where $|A| \geq 3$. If f is unanimous and strategyproof, then it is dictatorial.*

Lemma 20.4. *Suppose f is strategyproof and there is a profile in which a is top ranked by i, bottom ranked by all others, and a is selected. Then i is **decisive** for a in the sense that $f(\succ) = a$ whenever $\succ_i$ ranks a top.*

Proof. Suppose $i = 1$. Notice that a must remain selected however $\succ_2$ is changed, for else 2 can manipulate f to ensure that a, which he prefers least, is not selected. Similarly, a must remain selected however $\succ_3$ is subsequently changed, and so on.

Having changed $\succ_2, \dots, \succ_n$, alternative a must still remain selected however $\succ_1$ is subsequently changed, provided 1 ranks a top, since otherwise 1 can manipulate f. □

Lemma 20.5. *GS holds when $n = 2$.*

Proof. If neither 1 and 2 is a dictator then they must both be not-decisive for at least one alternative. If the only alternative for which each is not-decisive is the same alternative, say a, then both are decisive for every other alternative, including b and c. But this results in a contradiction if 1 ranks b top and 2 ranks c top. Hence there must exist distinct a and b for which 1 and 2 are not-decisive respectively.

Consider two preferences of the form

$$a \succ_1 b \succ_1 \cdots$$
$$b \succ_2 a \succ_2 \cdots$$

With these profiles $f(\succ_1, \succ_2) \in \{a, b\}$, else if it were c, either agent could manipulate by swapping his preference order for a and b, which by unanimity would produce a selection of either a or b, which he prefers to c. But the selection cannot be a, for then agent 2 does better by demoting a to the bottom position, where since 1 is not decisive for a, we know a is not selected and so b is selected (some c cannot be selected since then 1 could manipulate by swapping preferences of a and b). We are forced to revise our first assumption and conclude that either 1 or 2 is a dictator. □

Proof of GS. Now suppose f is a counterexample to GS for n agents, for least possible n. Choose an arbitrary $\succ_n^*$ and consider a SCF for the first $n-1$ agents defined as

$$h(\succ_1, \succ_2, \dots, \succ_{n-1}) = f(\succ_1, \succ_2, \dots, \succ_{n-1}, \succ_n^*).$$

This is obviously strategy proof, so as GS holds for $n-1$, either (i) it is not unanimous, or (ii) it has a dictator. In case (i) there is some a that is not selected even when all of $1, \dots, n-1$ rank it top. In case (ii) the dictator amongst agents $1, \dots, n-1$ can ensure the top choice of $\succ_n^*$ is not selected, even if all others also rank it top. In either case there exists some alternative that some individual can ensure is not chosen, even if all others rank it top. Suppose, without loss of generality, the alternative is a^* and the individual is n.

Next consider a SCF for two agents: $g(\succ_1, \succ_2) = f(\succ_1, \succ_1, \dots, \succ_1, \succ_2)$. Then g is unanimous, obviously. It is also strategyproof because if 2 can manipulate g it can manipulate f, and if 1 can manipulate g by changing $\succ_1$ to $\succ_1'$, then by changing $(\succ_1, \succ_1, \dots, \succ_1)$ to $(\succ_1', \succ_1', \dots, \succ_1')$ one agent at a time by increasing index, we see that at some point $f(\succ_1', \succ_1', \dots, \succ_1', \succ_1, \dots, \succ_1, \succ_2)$ is being manipulated by the single agent who is changing $\succ_1$ to $\succ_1'$, in contradiction to f being strategyproof.

Lemma 20.5 implies that g must have a dictator. However, agent 1 cannot be a dictator because agent 2 can ensure that a^* is not selected. Agent 2 cannot be a dictator because otherwise she would be decisive on every alternative and so be a dictator in the original f also, in contradition to our premise that f has no dictator. □

20.4 Alternative proof of Gibbard-Satterthwaithe theorem

In the following proof the condition of unanimity is replaced by a condition that f be surjective (i.e. every alternative can be selected by some profile). They are easily seen to be equivalent conditions. The proof that follows shares elements with the one on the previous page, but personally I find it harder to read. It is twice as long. Another issue is notation. I believe it is sometimes better to try to express things in English sentences rather than overload the reader with the task of parsing notation.

Proof of Gibbard-Satterthwaithe theorem. We need two lemmas. The first lemma states that a strategyproof SCF is monotone in the sense that the selected alternative does not change as long as all alternatives ranked below it are still ranked below it for all voters.

Lemma 20.6. *Let f be a strategyproof SCF, $\succ \in L(A)^n$ with $f(\succ) = a$. Then, $f(\succ') = a$ for every $\succ' \in L(A)^n$ such that for all $i \in N$ and $b \in A \setminus \{a\}$, $a \succ'_i b$ if $a \succ_i b$.*

Proof. We start from $\succ$ and change the preferences of one voter at a time until we get to $\succ'$, showing that the chosen alternative remains the same in every step. Let $b = f(\succ'_1, \succ_{-1})$. By strategyproofness, $a \succeq_1 b$, and thus $a \succeq'_1 b$ by assumption. Also by strategyproofness, $b \succeq'_1 a$, and thus $a = b$. The claim now follows by repeating the same argument for the remaining voters. ☐

The second lemma states that the alternative selected by a surjective and strategyproof SCF must be Pareto optimal.

Lemma 20.7. *Let f be a surjective and strategyproof SCF, $a, b \in A$, and $\succ \in L(A)^n$ such that $a \succ_i b$ for all $i \in N$. Then, $f(\succ) \neq b$.*

Proof. Assume for contradiction that $f(\succ) = b$. By surjectivity, there exists $\succ' \in L(A)^n$ such that $f(\succ') = a$. Let $\succ'' \in L(A)^n$ be a preference profile such that for all $i \in N$

$$a \succ''_i b \succ''_i x$$

for all $x \in A \setminus \{a, b\}$. Then, $x \succ_i b$ whenever $x \succ''_i b$ for some $i \in N$ and $x \in A \setminus \{b\}$, and $x \succ'_i a$ whenever $x \succ''_i a$ for some $i \in N$ and $x \in A \setminus \{a\}$. Thus, by Lemma 20.6, $f(\succ'') = f(\succ) = b$ and $f(\succ'') = f(\succ') = a$, a contradiction. ☐

Proof of Theorem 20.3. We first prove the theorem for $n = 2$ and then perform an induction on n.

Let $a, b \in A$ with $a \neq b$ and consider $\succ \in L(A)^2$ such that

$$a \succ_1 b \succ_1 x \qquad \text{and} \qquad b \succ_2 a \succ_2 x$$

for all $x \in A \setminus \{a, b\}$. Then, by Lemma 20.7, $f(\succ) \in \{a, b\}$.

Suppose that $f(\succ) = a$, and let $\succ' \in L(A)^2$ such that

$$a \succ'_1 b \succ'_1 x \qquad \text{and} \qquad b \succ'_2 x \succ'_2 a$$

for all $x \in A \setminus \{a, b\}$. Then, $f(\succ') = a$, since $f(\succ') \in \{a, b\}$ by Lemma 20.7 and $f(\succ') \neq b$ by strategyproofness. Lemma 20.6 now implies that f selects alternative a for any preference profile in which voter 1 ranks alternative a first.

By repeating the above analysis for every pair of distinct alternatives in A, we obtain two sets $A_1, A_2 \subseteq A$ such that A_i is the set of alternatives that are selected for every preference profile in which voter $i \in \{1, 2\}$ ranks them first. Let $A_3 = A \setminus (A_1 \cup A_2)$, and observe that $|A_3| \leq 1$: otherwise we would have performed the above analysis for two elements in A_3, which would place one of these elements in A_1 or A_2 and thus not in A_3.

Now observe that $|A| \geq 3$ and $|A_3| \leq 1$, so $|A_1 \cup A_2| \geq 2$. Moreover, for $x, y \in A$ with $x \neq y$, it cannot be the case that $x \in A_1$ and $y \in A_2$, because this would lead to a contradiction when voter 1 ranks x first and voter 2 ranks y first. Since $a \in A_1$, it follows that $A_1 \cap A_2 = \emptyset$ and thus that $A_2 = \emptyset$. It finally follows that $A_3 = \emptyset$: otherwise we could repeat the above analysis for $c \in A_3$ and $\succ'' \in L(A)^2$ with

$$c \succ''_1 a \succ''_1 x \qquad \text{and} \qquad a \succ''_2 c \succ''_2 x$$

for all $x \in A \setminus \{a, c\}$, and conclude that $c \in A_1$ or $a \in A_2$, a contradiction. It follows that $A_1 = A$, so voter 1 is a dictator.

Now we assume that the statement of the theorem holds for n voters and prove that it also holds for $n+1$ voters. Consider a surjective and strategyproof SCF $f : L(A)^{n+1} \to A$, and define $g : L(A)^2 \to A$ by letting

$$g(\succ_1, \succ_2) = f(\succ_1, \succ_2, \ldots, \succ_2)$$

for all $\succ_1, \succ_2 \in L(A)$.

Since f is surjective and strategyproof, and by Lemma 20.7, g is surjective as well. Assume for contradiction that g is not strategyproof. By strategyproofness of f, the manipulator must be voter 2, so there must exist $\succ_1, \succ_2, \succ'_2 \in L(A)$ and $a, b \in A$ such that $g(\succ_1, \succ_2) = a$, $g(\succ_1, \succ'_2) = b$, and $b \succ_2 a$. For $k = 0, \ldots, n$, let $\succ^k = (\succ_1, \succ'_2, \ldots, \succ'_2, \succ_2, \ldots, \succ_2) \in L(A)^{n+1}$ be the preference profile where k voters have preference order $\succ'_2$ and $n - k$ voters have preference order $\succ_2$, and let $a^k = f(\succ^k)$. Since $a^n = b \succ_2 a = a^0$, it must be the case that $a^{k+1} \succ_2 a^k$ for some k with $0 \leq k < n$, which means that f is manipulable, a contradiction. It follows that g is strategyproof, and therefore dictatorial.

If the dictator for g is voter 1, then by Lemma 20.6 voter 1 must also be a dictator for f. Assume instead that the dictator for g is voter 2, and let $h : L(A)^n \to A$ be given by

$$h(\succ_2, \ldots, \succ_{n+1}) = f(\succ^*_1, \succ_2, \ldots, \succ_{n+1})$$

for an arbitrary $\succ_1^* \in L(A)$. Then, h is strategyproof by strategyproofness of f, and surjective because voter 2 is a dictator for g. Therefore, by the induction hypothesis, h is dictatorial.

Assume without loss of generality that the dictator for h is voter 2, and let $e : L(A)^2 \to A$ be given by

$$e(\succ_1, \succ_2) = f(\succ_1, \succ_2, \succ_3^*, \dots, \succ_{n+1}^*)$$

for arbitrary $\succ_3^*, \dots, \succ_{n+1}^* \in L(A)$. Then e is strategyproof and surjective, and hence dictatorial. In fact, the dictator for e must be voter 2, because voter 1 is not a dictator for g and thus cannot be a dictator for e. Since $\succ_i^*$ for $i = 1, 3, \dots, n+1$ was chosen arbitrarily, it follows that voter 2 is a dictator for f. $\square$

21 Auctions

21.1 Types of auctions

Flowers, wines, art, U.S. treasury bonds, government contracts, and real estate are sold in auctions (and indeed the Roman empire was auctioned by the Praetorian Guards in A.D. 193). In recent years auctions have been important in selling natural resources, such as oil drilling rights and mobile telephone spectrum.

An auction is a type of multi-player game. The rules specify the way bidding occurs, what information the **bidders** have about the state of bidding, how the winner is determined and how much he must pay. It is a **partial information game** because each bidder's valuation of an item is hidden from the auctioneer and other bidders. The equilibrium is a function of the auction's rules. These rules can affect the revenue obtained by the seller, as well as how much this varies in successive instants of the auction. The problem of designing the best auction is a **mechanism design problem** . An auction design is said to be **economically efficient**, if it allocates the item to the bidder who values it most. In practice, auction design is an art. There is no one auction design that is efficient and can be applied in most situations.

In the **private values** model each bidder knows the value he places on the item, but does not know how it is valued by other bidders. As bidding takes place, his valuation does not change, though he gains information from the other players' bids. In the **common value** model the item's actual value is the same for all bidders, but they have different *a priori* information about that value. Think, for example, of a jar of coins. Each player makes an estimate of the value of the coins in the jar, and as bidding occurs he can adjust his estimate based on what other players say. In this case the winner generally overestimates the value (since he had the highest estimate), and so he pays more than the jar of coins is worth. This is called the **winner's curse**.

Auctions can be **oral** (bidders hear each other's bids and make counter-offers) or **written** (bidders submit sealed-bids in writing). Some popular auction types are:

1. **Dutch auction**: the price decreases continuously until some bidder calls stop.
2. **English auction** (or **ascending price auction**): bids increase in small increments until only one bidder remains.
3. **first price sealed-bid**: the winner pays his bid.
4. **second price sealed-bid** (or **Vickrey auction**): the winner pays the second highest bid.
5. **all-pay sealed-bid auction**: highest bidder wins, but all pay their bid.

Auctions 1 and 3 are equivalent (the item selling for the greatest valuation). Auctions 2 and 4 are equivalent (the item selling for the second greatest valuation).

21.2 The revenue equivalence theorem

The **symmetric independent private values model** (SIPV) concerns the auction of a single item, with risk neutral seller and bidders. Each bidder knows his own valuation of the item, which he keeps secret, and valuations of the bidders can be modelled as i.i.d. random variables. Important questions are

- what type of auction generates the most revenue for the seller?
- if seller or bidders are risk averse, which auction would they prefer?
- which auctions make it harder for the bidders to collude?

Let us begin with an intuitive result.

Lemma 21.1. *In any SIPV auction in which the bidders bid optimally and the item is awarded to the highest bidder, the bids are ordered the same as the valuations.*

Proof. Consider an auction satisfying the terms of the lemma. Let $e(p)$ be the minimal expected payment that a bidder can make if he wants to win the item with probability p. Notice that $e(p)$ must be a convex function of p, i.e. $e(\alpha p + (1-\alpha)p') \leq \alpha e(p) + (1-\alpha)e(p')$. This is because one strategy for winning with probability $\alpha p + (1-\alpha)p'$ is to bid so as to either win with probability p or p', doing these with probabilities α and $1-\alpha$ respectively. Since $e(p)$ is convex it is differentiable at all but a countable number of points. A bidder who has valuation θ and bids so as to win with probability p has expected profit $\pi(\theta) = p\theta - e(p)$. Assuming that p is chosen optimally, the relation between p and θ is determined by

$$\frac{\partial \pi}{\partial p} = \theta - e'(p) = 0. \tag{21.1}$$

Since $e'(p)$ is nondecreasing in p, it follows that $p(\theta)$ must be nondecreasing in θ. As the item goes to the highest bidder, the probability of winning increases with the the bid, and so the optimal bid must be nondecreasing in the valuation θ. □

We say that two auctions have the same **bidder participation** if any bidder who finds it profitable to participate in one auction also finds it profitable to participate in the other. The following result is remarkable, as different auctions can have completely different rules and the bidders' optimal bidding strategies will differ.

Theorem 21.2 (Revenue equivalence theorem)**.** *The expected revenue obtained by the seller is the same for any two SIPV auctions that (a) award the item to the highest bidder, and (b) have the same bidder participation.*

Proof. Suppose there are n bidders. From (21.1) we have

$$\frac{d}{d\theta} e(p(\theta)) = e'(p)p'(\theta) = \theta p'(\theta).$$

Suppose a bidder finds it profitable to participate iff $\theta_i \geq \theta^*$. By assumption (b) this is the same for any auction being considered. Integrating this gives

$$e(p(\theta_i)) - e(p(\theta^*)) = \int_{\theta^*}^{\theta_i} w p'(w)\,dw = \theta_i p(\theta_i) - \theta^* p(\theta^*) - \int_{\theta^*}^{\theta_i} p(w)\,dw\,, \tag{21.2}$$

where $e(p(\theta^*)) = \theta^* p(\theta^*)$ and so

$$e(p(\theta_i)) = \theta_i p(\theta_i) - \int_{\theta^*}^{\theta_i} p(w)\,dw.$$

Thus $e(p(\theta_i))$ depends only on the function $p(w)$ and the value of θ^*. We know from Lemma 21.1 that if bidders bid optimally then bids will be in the same order as the valuations, so if F is the distribution function of the valuations, then $p(w) = F(w)^{n-1}$, independently of the precise auction mechanism. Assume for simplicity that $\lim_{x\to\infty} x(1-F(x)) = 0$. The expected revenue is $\sum_{i=1}^n E_{\theta_i} e(p(\theta_i)) = nE_{\theta_1}(p(\theta_1))$. This is

$$\begin{aligned}
nE_{\theta_1}(p(\theta_1)) &= n\int_{\theta_1=\theta^*}^{\infty}\left[\theta_1 p(\theta_1) - \int_{\theta^*}^{\theta_1} p(w)dw\right] f(\theta_1)d\theta_1 \\
&= n\int_{\theta_1=\theta^*}^{\infty} \theta_1 p(\theta_1) f(\theta_1)d\theta_1 + n(1-F(\theta_1))\int_{\theta^*}^{\theta_1} p(w)dw\,\Bigg|_{\theta_1=\theta^*}^{\infty} \\
&\qquad\qquad - n\int_{\theta^*}^{\infty}(1-F(\theta_1))p(\theta_1)d\theta_1 \qquad (21.3)\\
&= n\int_{\theta_1=\theta^*}^{\infty}\left(\theta_1 - \frac{1-F(\theta_1)}{f(\theta_1)}\right)F(\theta_1)^{n-1} f(\theta_1)d\theta_1 \qquad (21.4)
\end{aligned}$$

where (21.3) is a step of integration by parts. □

Example 21.3. Assume valuations are i.i.d. with distribution function F.

(a) We might simply offer the item at price p and see if any player values it above p. The probability of making a sale is is $x(p) = 1 - F(p)^n$ and $px(p)$ is maximized where

$$p - \frac{1-F(p)^n}{nF(p)^{n-1}f(p)} = 0\,.$$

For the distribution $U[0,1]$, $F(u) = u$, and the optimal price is $p^* = \sqrt[n]{1/(n+1)}$, and the resulting (expected) revenue is $[n/(n+1)]\sqrt[n]{1/(n+1)}$. For $n = 2$, $p^* = \sqrt{1/3}$, and the expected revenue is $(2/3)\sqrt{1/3} = 0.3849$.

(b) If $n = 2$ and the item is auctioned by any of the five mechanisms above and all bidders bid optimally then the probabilty that a bidder with valuation θ wins is $F(\theta) = \theta$, i.e. $p(\theta) = \theta$. From (21.2) we see that $e(p(\theta)) = \theta^2/2$. So in all these auctions the seller's expected revenue is $2E[\theta^2/2] = 1/3 = 0.3333$.

The optimal bids differ in the different auctions. Clearly, in the all-pay sealed-bid auction the optimal bid is $e(p(\theta)) = \theta^2/2$. In the Dutch or first price sealed-bid auctions, a bidder's expected payment is $p(\theta)$ times his bid. Since this must equal $\theta^2/2$ the optimal bid must be $\theta/2$. In the second price sealed-bid (Vickrey) auction, the winner pays the bid of the second highest bidder. If bidder 1 bids u, then his profit is $(\theta_1 - \theta_2)1_{\{u>\theta_2\}}$. For every possible value of θ_2, this is maximized by bidding $u = \theta_1$.

The seller's expected revenue is greater with (a) than (b). However, in (a) he assumes information about the distribution of the valuations. In (b) no assumption is made.

Example 21.4. Consider a 3-person game in which player 0, the government, solicits tenders from two contractors to do a piece of work. The contractors' costs, C_1, C_2, are private and independently distributed $U[0,1]$. If the government could know the low bidder's cost and pay just a bit above that then its expected payment would be a bit over $E\min(C_1, C_2) = 1/3$, which we call **first best**. However, not knowing this information, the government takes written bids and awards the contract to the lowest bidder. By a similar analysis we find that a contractor with cost C_i will bid $(1 + C_i)/2$. This results in expected payment of $2/3$. We say that the **price of anarchy** is $(2/3)/(1/3) = 2$.

We might also in Example 21.3 consider **risk sensitivity**. Suppose bidders are risk-neutral but the seller is risk sensitive, meaning that he cares not just about expected revenue, but also about its variance. Again, consider (a). In a first price sealed-bid auction each bids half his valuation, so the seller's revenue is $(1/2)\max\{\theta_1, \theta_2\}$. In an all-pay sealed-bid auction each pays half the square of his valuation and revenue is $\frac{1}{2}\theta_1^2 + \frac{1}{2}\theta_2^2$. In the Vickrey auction each bids his valuation and revenue is $\min\{\theta_1, \theta_2\}$. All these have expectation $1/3$, but the variances are $1/72$, $2/45$ and $1/18$ respectively. Thus a risk adverse seller prefers the first price auction to the all-pay auction, which is preferred to the Vickrey auction.

21.3 Mechanism design

The design of a social choice function and design of an auction are both examples of **mechanism design** problems. We are given a set of **agents** $N = \{1, \ldots, n\}$. Agent i has private knowledge of his **type**, say $\theta_i \in \Theta_i$, where Θ_i is the set of his possible types. As a function of θ_i, agent i sends a **message** from his message space Σ_i.

The idea is that the agents send messages to the mechanism, providing information about their types, and as a function of these messages the mechanism selects an alternative $f(\theta)$ from a set A.

If $\Sigma_i = \Theta_i$ for each agent i, i.e. the content of a message is a declaration of a type, then the mechanism is said to be **direct**. A direct **mechanism** with payments, (f, p), receives the agents' type declarations $\theta = (\theta_1, \ldots, \theta_n)$ and selects $f(\theta) \in A$, where A is a set of alternatives. It is convenient to write $\theta = (\theta_i, \theta_{-i})$, where θ_{-i} is the vector of types declared by agents other than i. Agent i wishes to make a declaration that maximizes his received **quasilinear utility**

$$u_i(f(\theta_i',\theta_{-i}),\theta_i) = v_i(f(\theta_i',\theta_{-i}),\theta_i) - p_i(\theta_i',\theta_{-i}), \tag{21.5}$$

where $v_i : A \times \Theta_i \to \mathbb{R}$ is a valuation function over alternatives, agent i has true type θ_i, declares θ_i', and pays $p_i(\theta_i',\theta_{-i}) = (p(\theta_i',\theta_{-i}))_i$.

A mechanism is **manipulable** if there exist situations in which an agent can do better by declaring a type that differs from his true type. A direct mechanism is called **incentive compatible**, or **strategyproof** if it is a not manipulable. In this case it is a Nash equilibrium for each agent to truthfully report his true type. That is,

$$u_i(f(\theta_i,\theta_{-i}),\theta_i) \geq u_i(f(\theta_i',\theta_{-i}),\theta_i), \quad \text{for all } \theta_i' \in \Theta_i.$$

Our aim is to design the mechanism to force the Nash equilibrium of the resulting game to satisfy desirable properties or optimize some objective function. One natural objective is to maximize social welfare. The **social welfare** of alternative $a \in A$ is $\sum_{i \in N} v_i(a,\theta_i)$, i.e. the sum of all agents' valuations for this alternative.

For example, in a **sealed-bid auction** an agent's type corresponds to his valuation θ_i and in a direct mechanism he bids a valuation. Since he must submit his bid before hearing other agents' bids it is appropriate for the agent to maximize his ex-ante expected utility. The equilibrium condition is that $E_{\theta_{-i}} u_i(f(\theta_i',\theta_{-i}),\theta_i)$ is maximized by making the truthful declaration $\theta_i' = \theta_i$, where the expectation is taken using a prior distribution over θ_{-i}, which denotes the vector of the other agents' types and assumes they are all declaring truthfully. This is called a **Bayes-Nash equilibrium**, but we shall often just say Nash-equilibrium, or equilibrium. In an auction, maximizing social welfare means maximizing the expected valuation of the bidder who wins the item.

22 Optimal Auctions

22.1 Revelation principle

One might think that arbitrarily complicated mechanisms would be needed to implement certain desired results. However, it is sufficient to restrict attention to mechanisms that incentivize truthful revelation of type. These are called **direct revelation mechanisms**. The following result implies that we can restrict our attention to these.

Theorem 22.1 (Revelation principle)**.** *A direct revelation mechanism can generally be designed to achieve the same result as any other mechanism.*

Proof. Consider some mechanism, f, which implements some result (an equilibrium) via agents making untruthful revelations about their types. Consider now a mechanism f' which receives truthful information from the agents, and then inputs to f the information that agents would have themselves submitted to f and then outputs the alternative that f would choose. Then it is an equilibrium of f' for players to truthfully report to f'. So f' is a direct mechanism that achieves the same result as f. □

This is important because it means that if we wish to design an optimal mechanism (e.g. maximizing the seller's expected revenue), we may restrict attention to mechanism that incentivize truthtelling. This is called an **incentive compatibility condition**.

22.2 Vickrey-Clark-Groves mechanisms

One way to construct a direct revelation mechanism that maximizes social welfare is the **Vickrey-Clark-Groves mechanism** in which (f, p) is such that

$$f(\theta) \in \arg\max_{a \in A} \sum_{i \in N} v_i(a, \theta_i) \qquad \text{and}$$
$$p_i(\theta) = h_i(\theta_{-i}) - \sum_{j \in N \setminus \{i\}} v_j(f(\theta), \theta_j) \qquad \text{for all } i \in N,$$

where $h_i : \Theta_{-i} \to \mathbb{R}$ is some function that depends on the types of all agents but i. The crucial component is the term $\sum_{j \in N \setminus \{i\}} v_j(f(\theta), \theta_j)$, which is equal to the social welfare for all agents but i. The utility of agent i adds its own valuation $v_i(f(\theta), \theta_i)$ and thus becomes equal to the social welfare of alternative $f(\theta)$ minus the term $h_i(\theta_{-i})$. The latter does not depend on θ_i and therefore has no strategic implications.

Theorem 22.2. *VCG mechanisms are strategyproof.*

Proof. Let $i \in N$, $\theta \in \Theta$, and $\theta_i' \in \Theta_i$. Then,

$$u_i(\theta, \theta_i) = v_i(f(\theta), \theta_i) - p_i(\theta)$$

$$
\begin{aligned}
&= \sum_{j \in N} v_j(f(\theta), \theta_j) - h_i(\theta_{-i}) \\
&\geq \sum_{j \in N} v_j(f(\theta_i', \theta_{-i}), \theta_j) - h_i(\theta_{-i}) \\
&= u_i(f(\theta_i', \theta_{-i}), \theta_i),
\end{aligned}
$$

where the inequality holds because $f(\theta)$ maximizes social welfare with respect to θ. $\square$

In the case that a Bayes-Nash equilibrium is the appropriate concept the above proof adapts to show that $E_{\theta_{-i}} u_i((\theta_i, \theta_{-i}), \theta_i) \geq E_{\theta_{-i}} u_i((\theta_i', \theta_{-i}), \theta_i)$.

Strategyproofness holds for any choice of the functions h_i, so it is natural to ask for a good way to define these functions. In many cases it makes sense that agents are charged rather than paid, but not more than their gain from participating in the mechanism. Formally, mechanism (f, p) makes **no positive transfers** if $p_i(\theta) \geq 0$ for all $i \in N$ and $\theta \in \Theta$, and is **ex-post individually rational** if it always yields non-negative utility for all agents, i.e. if $v_i(f(\theta), \theta_i) - p_i(\theta) \geq 0$ for all $i \in N$ and $\theta \in \Theta$. It turns out that these two properties can indeed be achieved simultaneously. The so-called **Clark pivot rule** sets $h_i(\theta_{-i}) = \max_{a \in A} \sum_{j \in N \setminus \{i\}} v_j(a, \theta_j)$, such that the payment of agent i becomes $p_i(\theta) = \max_{a \in A} \sum_{j \in N \setminus \{i\}} v_j(a, \theta_j) - \sum_{j \in N \setminus \{i\}} v_j(f(\theta), \theta_j)$. Intuitively, this latter amount is equal to the externality agent i imposes on the other agents, i.e. the difference between their social welfare with and without i's participation. The payment makes the agent internalize this externality.

It is easy to check that in the auction a single item, the above prescription provides that bidders bid their true valuations. The one with highest valuation wins and pays the second highest valuation. Others pay 0. This is the second-price, or Vickrey, auction. Similarly, if one is auctioning k identical bottles a direct revelation mechanism is to award the bottles to the k highest bidders, who all then pay the value of the $(k+1)$th highest bid.

22.3 Optimal auctions

Recall that in the SIPV model a single item is being auctioned, with risk neutral seller and bidders. Each bidder's valuation is private information, and valuations of the bidders can be modelled as i.i.d. random variables, with known distribution function F, and probability density function f.

We ask the bidders to declare their valuations, say $\theta_1, \dots, \theta_n$. As a function of these declarations, $\theta = (\theta_1, \dots, \theta_n)$, we make bidder i pay $p_i(\theta)$, and award the item to bidder i with probability $v_i(\theta)$. Suppose that we are designing a direct revelation mechanism such that it is optimal for all bidders to declare their valuations truthfully.

Assuming that all other bidders are declaring their valuations truthfully, a bidder i who has valuation θ_i will declare it to be θ_i', where

$$\begin{aligned}\theta_i' &= \arg\max_{\theta_i'}\Big\{\theta_i E_{\theta_{-i}} v_i(\theta_1,\ldots,\theta_i',\ldots,\theta_n) - E_{\theta_{-i}} p(\theta_1,\ldots,\theta_i',\ldots,\theta_n)\Big\} \\ &= \arg\max_{\theta_i'}\Big\{\theta_i V_i(\theta_i') - P_i(\theta_i')\Big\}, \qquad (22.1)\end{aligned}$$

where $E_{\theta_{-i}}$ denotes expectation with respect to the other bidders' valuations, and V_i and P_i are defined implicitly in (22.1). They are the probability of winning the item, and the expected payment of agent i, when he declares θ_i'. We know that it is sufficient to look for an optimal auction amongst those that are direct revelation mechanisms, so we apply the incentive compatibility condition, that bidder i is to be incentivized to be truthful. Supposing his true valuation is θ_i, that derivatives exist and agent i has maximum profit at a stationary point, we require

$$\frac{d}{d\theta_i'}[\theta_i V_i(\theta_i') - P_i(\theta_i')]\,\Big|_{\theta_i'=\theta_i} = \theta_i V_i'(\theta_i) - P_i'(\theta_i) = 0, \qquad (22.2)$$

for all θ_i such that it is optimal to participate. Suppose there is some θ_i^* such that $\theta_i V_i(\theta_i) - P_i(\theta_i) \geq 0$ iff $\theta_i \geq \theta_i^*$, and $\theta_i^* V_i(\theta_i^*) - P_i(\theta_i^*) = 0$. Integrating (22.2) we have

$$\begin{aligned}P_i(\theta_i) &= P_i(\theta_i^*) + \int_{\theta_i^*}^{\theta_i} wV_i'(w)\,dw = P_i(\theta_i^*) + wV_i(w)\,\Big|_{\theta_i^*}^{\theta_i} - \int_{\theta_i^*}^{\theta_i} V_i(w)\,dw. \\ &= \theta_i V_i(\theta_i) - \int_{\theta_i^*}^{\theta_i} V_i(w)\,dw. \qquad (22.3)\\ EP_i(\theta_i) &= \int_{\theta_i^*}^{\infty}\left[\theta_i V_i(\theta_i) - \int_{\theta_i^*}^{\theta_i} V_i(w)\,dw\right] f(\theta_i)\,d\theta_i \\ &= \int_{\theta_i^*}^{\infty}\left(\theta_i - \frac{1-F(\theta_i)}{f(\theta_i)}\right) V_i(\theta_i) f(\theta_i)\,d\theta_i, \qquad (22.4)\end{aligned}$$

where (22.4) is via integration by parts, as in (21.3). Compare this to (21.4), which was derived by supposing a bidder bids optimally; here we find the same thing, but the derivation is via the incentive compatibility condition (22.2) that the bidder is incentivized to made a truthful declaration of her valuation when bidding optimally.

Let

$$g(\theta_i) = \theta_i - \frac{1-F(\theta_i)}{f(\theta_i)}.$$

and suppose it is nondecreasing in θ_i. The auctioneer's expected revenue is

$$\sum_i EP_i(\theta_i) = \sum_i \int_{\theta_i^*}^{\infty} g(\theta_i) V_i(\theta_i) f(\theta_i)\,d\theta_i = E\left[\sum_i \phi_i(\theta_i) g(\theta_i) v_i(\theta_1,\ldots,\theta_n)\right],$$

where $\phi_i(\theta_i)$ is 1 or 0 as $\theta_i \geq \theta_i^*$ or $\theta_i < \theta_i^*$, and he maximizes this by setting $v_i = 1$ for bidder i with greatest value of $g(\theta_i)$, but awarding it to no bidder if this is negative for all declared θ_i.

Consider, for example, the case that $\theta_1, \dots, \theta_n$ are uniformly distributed on $[0, 1]$. Then $g(\theta_i) = 2\theta_i - 1$, $\theta_i^* = 1/2$, $V_i(\theta_i) = \theta_i^{n-1}$, and then from (22.3)

$$P_i(\theta_i) = \theta_i\theta_i^{n-1} - \int_{1/2}^{\theta_i} w^{n-1}dw = \frac{n-1}{n}\theta_i^n + \frac{1}{n2^n}.$$

For instance, if $n = 2$ we might take $p_i(\theta_i) = P_i(\theta_i) = \theta_i^2/2 + 1/8$. This could be the payment in an all-pay auction in which a participant who wishes to bid θ_i must commit to paying $\theta_i^2/2 + 1/8$ (whether or not he wins). Thus a player with valuation θ_i will declare $\theta_i = u$ so as to maximize

$$\theta_i u - (u^2/2 + 1/8).$$

This is maximized by $u = \theta_i$ and has a positive maximum if $\theta_i > 1/2$. The expected sum of the payments is

$$2EP_1(\theta_1) = 2\int_{1/2}^{1} (1/8 + \theta_i^2/2)\, d\theta_i = 5/12,$$

which exceeds the 1/3 we have seen in other auctions. The revenue equivalent theorem does not apply because the bidder participation is not always the same.

There are other ways to create an optimal auction (i.e. one that maximizes seller's revenue). We could conduct an English auction with **reservation price**, p_0. Bidder 1 in this auction pays p_0 if bidder 2 has valuation less than p_0, and otherwise pays Bidder 2's valuation if $\theta_1 > \theta_2 > p_0$. This makes his expected payment $p_0^2 + \int_{p_0}^{\theta_1} u\, du = (1/2)(\theta_1^2 + p_0^2)$ provided $\theta_1 > p_0$. The seller's expected revenue is maximized by $p_0 = 1/2$.

Or we might make each bidder pay a **participation fee** c (which must be paid by a player if he wishes to submit a bid). Now bidder 1 will participate only if $\theta_1 > \theta^*$, and in that case if he bids u then his expected payment is

$$c + \int_{\theta^*}^{u} \theta_2\, d\theta_2 = c + (u - \theta^*)\frac{\theta^* + u}{2}.$$

The fact that $\theta_1 = \theta^*$ has expected profit 0 means that $(\theta^*)^2 - c = 0$. One can check that the seller's expected revenue is maximized by $c = 1/4$, $\theta^* = 1/2$.

Note that if $\theta_1, \dots, \theta_n$ are i.i.d. $U[0, 1]$ then all optimal auctions have the property that a bidder will participate only if his valuation exceeds 1/2, and this is true for any n, as this critical value is determined solely by $g(\theta_i^*) = 2\theta_i^* - 1 = 0$.

22.4 Heterogenous bidders

The working the previous section can be generalized to the case of heterogenous bidders, having data F_i, f_i. For example, (22.3) becomes

$$P_i(\theta_i) = \theta_i \prod_{j\neq i} F_j(\theta_i) - \int_{\theta_i^*}^{\theta_i} \prod_{j\neq i} F_j(w)\,dw.$$

The auctioneer's expected revenue is

$$E\left[\sum_i \phi_i(\theta_i) g_i(\theta_i) v_i(\theta_1,\ldots,\theta_n)\right].$$

So the optimal auction mechanism awards the item to the bidder i for whom $g_i(\theta_i)$ is both maximal and nonnegative.

One way to construct payments in an optimal auction is require the winner to pay the smallest value of θ_i for which he would still be the winner. To prove that it this payment scheme works we should do a calculation to check that the expected payment of bidder i, when she has valuation θ_i, is indeed $P_i(\theta_i)$ as given in (22.3).

Example 22.3. An interesting property of optimal auctions with heterogeneous bidders is that the winner is not necessarily the highest bidder.

(a) Consider first the case of homogeneous bidders with valuations uniformly distributed on $[0,1]$. In this case $g_i(\theta_i) = \theta_i - (1-\theta_i)/1 = 2\theta_i - 1$. The object is sold to the highest bidder, but only if $2\theta_i - 1 > 0$, i.e., if his valuation exceeds $1/2$. The winner pays either $1/2$ or the second greatest bid, whichever is greatest. In the case of two bidders with the identical uniformly distributed valuations the seller's expected revenue is $5/12$. This agrees with what we have found above.

(b) Now consider the case of two heterogeneous bidders, say 1 and 2, whose valuations are uniformly distributed on $[0,1]$ and $[0,2]$ respectively. So $g_1(\theta_1) = 2\theta_1 - 1$, $\theta_1^* = 1/2$, and $g_2(\theta_2) = 2\theta_2 - 2$, $\theta_2^* = 1$. Under the bidding rules described above, bidder 2 wins only if $2\theta_2 - 2 > 2\theta_1 - 1$ and $2\theta_2 - 2 > 0$, i.e., if and only if $\theta_2 - \theta_1 > 1/2$ and $\theta_2 > 1$; so the lower bidder can sometimes win. For example, if $\theta_1 = 0.8$ and $\theta_2 = 1.2$, then 1 should be declared the winner and pay 0.7 (which is the smallest u such that $g_1(u) = 2u - 1 \geq 2\theta_2 - 2 = 0.4$).

23 Mechanism Design for a Shared Facility

23.1 Paying for a club good

Suppose we propose to provide a new web service that allows its users to share digital content in the cloud. We ask potential users to declare their private values for the service, and on that basis we decide whether or not to provide it, and if so, how large to build it. If we do build it, some persons are told the login password, while others can be excluded.

There are n agents who will potentially share the facility. Agent i will obtain utility $\theta_i v(Q)$, where Q is the size of the facility, which is built at cost $c(Q)$. The value of θ_i is private information. However, it is public knowledge *a priori* that θ_i is distributed with distribution function F_i. For simplicity we take $F_i = F$.

Our problem is to design a mechanism in which (i) agents declare values of θ_i, (ii) as a function of declared $\theta_1, \dots, \theta_n$ a facility is built of optimal size Q, and (iii) **budget balance** is achieved, in the sense that the agents' payments cover the cost $c(Q)$.

The analysis follows the same lines in Lecture 22. The revelation principle means that we can restrict attention to direct revelation mechanisms. Fix a mechanism and suppose that under this mechanism if agent i declares θ_i, then the expected value of $v(Q)$ is $V_i(\theta_i) = E_{\theta_{-i}}[v(Q(\theta)) \mid \theta_i]$, and his expected payment is $P(\theta_i)$. Knowing this, agent i will declare θ_i to be θ_i' so as to maximize his expected **net benefit** of

$$\theta_i V_i(\theta_i') - P_i(\theta_i').$$

If this is to be made maximal (and stationary) by the truthful declaration $\theta_i' = \theta_i$ then we need $\theta_i V_i'(\theta_i) - P_i'(\theta_i) = 0$. Integrating this gives,

$$P_i(\theta_i) - P_i(\theta_i^*) = \int_{\theta_i^*}^{\theta_i} w V_i'(w)dw = \theta_i V_i(\theta_i) - \theta_i^* V_i(\theta_i^*) - \int_{\theta_i^*}^{\theta_i} V_i(w)dw,$$

where θ_i^* is the least value of θ_i for which it would be ex-ante individually rational for agent i to participate, and so $\theta_i^* V_i(\theta_i^*) - P_i(\theta_i^*) = 0$. As before, let $g(\theta_i) = \theta_i - [1 - F(\theta_i)]/f(\theta_i)$. The expected payment of agent i is

$$\begin{aligned}
&\int_{\theta_i \geq \theta_i^*} P_i(\theta_i) f(\theta_i) d\theta_i = \int_{\theta_i \geq \theta_i^*} \left[\theta_i V_i(\theta_i) - \int_{\theta_i^*}^{\theta_i} V_i(w)dw\right] f(\theta_i)d\theta_i \\
&= \int_{\theta_1} \cdots \int_{\theta_n} \phi(\theta_i)\left(\theta_i - \frac{1 - F(\theta_i)}{f(\theta_i)}\right) v(Q(\theta_1, \dots, \theta_n)) f(\theta_1) \cdots f(\theta_n)\, d\theta_1 \dots d\theta_n \\
&= \int_{\theta_1} \cdots \int_{\theta_n} \phi(\theta_i) g(\theta_i) v(Q(\theta_1, \dots, \theta_n)) f(\theta_1) \cdots f(\theta_n)\, d\theta_1 \dots d\theta_n \\
&= E[\phi(\theta_i) g(\theta_i) v(Q(\theta_1, \dots, \theta_n))]
\end{aligned}$$

where $\phi(\theta_i)$ is 1 or 0 as $\theta_i \geq \theta_i^*$ or $\theta_i < \theta_i^*$. Notice that an important aspect of the mechanism design is that with $\phi = 0$, it can **exclude** an agent from using the facility. This threat is part of the incentive for agents not to understate their types.

Our aim is to maximize expected social welfare of

$$E\Big[\sum_i \phi(\theta_i)\theta_i v(Q(\theta_1, \dots, \theta_n)) - c(Q)\Big] \tag{23.1}$$

subject to a constraint that the payments cover the cost. This constraint is difficult to handle, so let us instead consider a weaker constraint, that the expected payments cover expected cost, i.e.

$$\sum_i E[\phi(\theta_i) g(\theta_i) v(Q(\theta_1, \dots, \theta_n))] \geq E[c(Q(\theta_1, \dots, \theta_n))]. \tag{23.2}$$

This leads us to the Lagrangian, combining (23.1) and (23.2),

$$\begin{aligned} L = \int_{\theta_1} \cdots \int_{\theta_n} \Big[\sum_i \phi(\theta_i)(\theta_i + \lambda g(\theta_i)) v(Q(\theta_1, \dots, \theta_n)) \\ - (1+\lambda) c(Q(\theta_1, \dots, \theta_n)) \Big] f(\theta_1) \cdots f(\theta_n) \, d\theta_1 \dots d\theta_n. \end{aligned}$$

which is to be maximized pointwise in the integrand with respect to ϕ_is and Q. If one assume v is concave and c is convex, it can be shown that the problem strong-Lagrangian.

To illustrate ideas, let us suppose that $\theta_i \sim U[0,1]$, so $g(\theta_i) = 2\theta_i - 1$. Then the maximum of L is achieved by taking $\phi(\theta_i) = 1$ if $\theta_i + \lambda g(\theta_i) \geq 0$, i.e. if $\theta_i \geq \lambda/(2\lambda+1) = \theta_i^*$. Otherwise $\phi_i(\theta_i) = 0$. Note that θ_i^* increases from 0 to 1/2 as λ increases from 0 to infinity. For a given $\lambda \geq 0$ we know how to find the ϕ_i. We also know that Q should be chosen to maximize the integrand, so

$$Q(\theta) = \arg\max_Q \Big\{ v(Q) \sum_i (\theta_i + \lambda g(\theta_i))^+ - (1+\lambda) c(Q) \Big\}.$$

The solution is completed by finding the value of λ such that the expected payments taken from the agents match expected cost.

The calculation of the optimal mechanism for given data is complicated. However, as n becomes large, one can prove that nearly the same welfare can be achieved by a simple scheme that announces that the facility will be of size Q^* and charges a fixed subscription fee p^* to any agent who then chooses to participate, which will be agents for whom $\theta_i v(Q^*) - p^* \geq 0$. The values of parameters Q^* and p^* are chosen so that $nP(\theta_i v(Q^*) \geq p^*) = c(Q^*)$ and $nE[\theta_i \mid \theta_i v(Q^*) \geq p^*] v(Q^*) - c(Q^*)$ is maximized.

Example 23.1. Suppose a facility may or may not be built. If it is built, the cost is 1. There are two agents, whose utility for the facilty are θ_1, θ_2, which are a priori distributed independently $U[0,1]$.

If it were possible to know θ_1, θ_2 then social welfare would be maximized by building the facility iff $\theta_1 + \theta_2 \geq 1$. The expected social welfare would be $E[(\theta_1 + \theta_2 - 1)^+] = 1/6 = 0.1666$. This solution is called **first best**.

Applying the analysis above, we find that the **second-best** solution is to build the facility iff $\theta_1 + \theta_2 \geq 1.25$. The expected social welfare is $E[(\theta_1 + \theta_2 - 1.25)^+] = 9/64 = 0.104625$. The payments can be

$$P(\theta_i) = \begin{cases} \frac{1}{2}\theta_i^2 - \frac{1}{32}, & \theta_i > \frac{1}{4}, \\ 0, & \text{otherwise.} \end{cases}$$

However, this is unsatisfactory because it is only ex-ante budget balanced, but not ex-post budget balanced.

$$P(\theta_1) + P(\theta_2) \neq 1_{\{\theta_1+\theta_2 \geq 1.25\}}.$$

Also, payments are taken even when the facility is not built.

23.2 Ex-post budget balance

It is possible, to take the above mechanism, in which there is ex-ante budget balance, and strengthen it to a mechanism in which is ex-post budget balanced, i.e. where payments cover cost exactly. Suppose the ex-post budget imbalance is

$$x(\theta) = c(Q(\theta)) - \textstyle\sum_i P(\theta_i).$$

Pick two agents, say 1 and 2. Let new payments be

$$\begin{aligned} p_1(\theta) &= P(\theta_1) + x(\theta) - E[x(\theta) \mid \theta_1] \\ p_2(\theta) &= P(\theta_2) + E[x(\theta) \mid \theta_1] \\ p_i(\theta) &= P(\theta_i), \quad i \neq 1, 2. \end{aligned}$$

Now there is ex-post budget balance because

$$\textstyle\sum_i p_i(\theta) = \sum_i P(\theta_i) + x(\theta) = c(Q(\theta)).$$

Also, each agent's incentive to be truthful remains the same because

$$\begin{aligned} E[p_1(\theta) \mid \theta_1] &= P(\theta_1) \\ E[p_2(\theta) \mid \theta_2] &= P(\theta_2) + Ex(\theta) = P(\theta_2). \end{aligned}$$

Applying this to Example 23.1 we obtain a mechanism design in which agent 1 pays

$$\begin{aligned} p_1(\theta) = \tfrac{1}{2}1_{\{\theta_1+\theta_2\geq 1.25\}} &+ \left(\tfrac{3}{32} - \tfrac{1}{2}\theta_1(1-\theta_1)\right)1_{\{\theta_1\geq .25\}} \\ &- \left(\tfrac{3}{32} - \tfrac{1}{2}\theta_2(1-\theta_2)\right)1_{\{\theta_2\geq .25\}} \end{aligned}$$

and $p_2(\theta)$ is symmetrical. However, this is still unsatisfactory because agents are being made to pay even when the facility is not built. An even better scheme is

$$p_1^*(\theta) = \left(\tfrac{1}{3}(\theta_1 - \theta_2) + \tfrac{1}{2}\right)1_{\{\theta_1+\theta_2\geq 1.25\}}\,.$$

This design has the advantages that (i) it is incentive compatible, (ii) $p_1(\theta) \geq 0$, (iii) $p_1(\theta) + p_2(\theta) = 1_{\{\theta_1+\theta_2\geq 1.25\}}$ (ex-post budget balance), (iv) $p_i(\theta) = 0$ when the facility is not built; $\theta_1 - p_1(\theta) > 0$ when it is built (ex-post individual rationality). However, we cannot have everything ex-post. For instance, this mechanism is not ex-post incentive compatible, because $\theta_1 - p_1(\theta_1')$ is not maximized by $\theta_1' = \theta_1$.

23.3 Refining a mechanism design

Here is an alternative way to change a scheme that is not ex-post budget balanced to create one that is. We will take a discretized model in which θ takes values $t_1, t_2 < \cdots < t_m$ with equal probability. So there are m^n different, equally likely, θ. Start with a mechanism that is ex-ante budget balanced but not ex-post budget balanced. There must be some $\theta = (\theta_1, \ldots, \theta_n)$ for which

$$p_1(\theta) + \cdots + p_n(\theta) > c(Q(\theta)).$$

Ex-ante budget balance says that $Ec(Q(\theta)) = E\sum_i p_i(\theta)$, so there must exist some other profile of types ϕ for which

$$p_1(\phi) + \cdots + p_n(\phi) < c(Q(\phi)).$$

Case I. There may be some i such that $\theta_i = \phi_i$. Suppose $i = 1$. Then adjust payments in two situations

$$\begin{aligned} p_1^*(\theta_1, \theta_2, \ldots, \theta_n) &= p_1(\theta_1, \theta_2, \ldots, \theta_n) - \epsilon \\ p_1^*(\theta_1, \phi_2, \ldots, \phi_n) &= p_1(\theta_1, \phi_2, \ldots, \phi_n) + \epsilon \end{aligned}$$

with $p_j^* = p_j$ elsewhere. Increase ϵ until there is exact budget balance with either θ, ϕ (or both). The two vectors of types, θ and ϕ, are equally likely. So incentives do not change because when agent 1 has type θ_1 his expected payment is $E\big[p_1^*(\theta) \bigm| \theta_1\big] = E\big[p_1(\theta) \bigm| \theta_1\big]$.

Case II. If $\theta_1 \neq \phi_1$ and $\theta_2 \neq \phi_2$. Adjust four payments:

$$\begin{aligned} p_1^*(\theta_1, \theta_2, \theta_3, \ldots, \theta_n) &= p_1(\theta_1, \theta_2, \theta_3, \ldots, \theta_n) - \epsilon \\ p_1^*(\theta_1, \phi_2, \theta_3, \ldots, \theta_n) &= p_1(\theta_1, \phi_2, \theta_3, \ldots, \theta_n) + \epsilon \\ p_2^*(\theta_1, \phi_2, \theta_3, \ldots, \theta_n) &= p_2(\theta_1, \phi_2, \theta_3, \ldots, \theta_n) - \epsilon \\ p_2^*(\phi_1, \phi_2, \phi_3, \ldots, \phi_n) &= p_2(\phi_1, \phi_2, \phi_3, \ldots, \phi_n) + \epsilon \end{aligned}$$

and $p_i^* = p_i$ elsewhere. The effect can be appreciated in a table

	p_1	p_2
$\theta_1, \theta_2, \theta_3, \ldots, \theta_m$	$-$	
$\theta_1, \phi_2, \theta_3, \ldots, \theta_m$	$+$	$-$
$\phi_1, \phi_2, \phi_3, \ldots, \phi_m$		$+$

As before, increase ϵ until there is budget balance with either θ, ϕ (or both). The three rows of the table are equally likely. So again, the incentives for agents 1 and 2 are unchanged, e.g., given that his type is θ_1 the expected payment agent 1 unchanged. One can repeat this process, always preserving the same incentives, until budget balance is obtained for every vector of types. The process is reminiscent of pivot steps in the transportationn algorithm.

By a similar adjustment mechanism it is possible to show that there exists a mechanism design which has ex-post budget balance, and the additional property that

$$\phi_i(\theta) = 0 \implies p_i(\theta) = 0.$$

This is desirable, since it might be difficult in practice to extract a payment towards the cost of a facility from someone who is being excluded from using it! However, there is a limit to ‘good properties’ that we can demand of our mechanism design. As with Arrow’s theorem, and the Gibbart-Satterwaite theorem, we find it is impossible to simultaneously satisfy ex-post versions of budget balance, individual rationality and incentive compatibility.

Index