Easter Term 2010 Richard Weber

OPTIMIZATION

Contents

Books

Bazaraa, M., Jarvis, J. and Sherali, H *Linear Programming and Network Flows*, fourth edition, 2010, Wiley.

Luenberger, D. *Introduction to Linear and Non-Linear Programming*, second edition, 1984, Addison-Wesley.

Vanderbei, R. J. *Linear programming: foundations and extensions.* Kluwer 2001(61.50 hardback).

Schedules

Lagrangian methods

General formulation of constrained problems; the Lagrangian sufficiency theorem. Interpretation of Lagrange multipliers as shadow prices. Examples. [2]

Linear programming in the nondegenerate case

Convexity of feasible region; sufficiency of extreme points. Standardization of problems, slack variables, equivalence of extreme points and basic solutions. The primal simplex algorithm, artificial variables, the two-phase method. Practical use of the algorithm; the tableau. Examples. The dual linear problem, duality theorem in a standardized case, complementary slackness, dual variables and their interpretation as shadow prices. Relationship of the primal simplex algorithm to dual problem. Two person zero-sum games. [6]

Network problems

The Ford-Fulkerson algorithm and the max-flow min-cut theorems in the rational case. Network flows with costs, the transportation algorithm, relationship of dual variables with nodes. Examples. Conditions for optimality in more general networks; *the simplex-on-a-graph algorithm*. [3]

Practice and applications

Efficiency of algorithms. The formulation of simple practical and combinatorial problems as linear programming or network problems. [1]

Notation

n, m	numbers of decision variables and functional constraints
x	decision variable in a primal problem, $x \in \mathbb{R}^n$
λ	decision variable in the dual problem, $\lambda \in \mathbb{R}^m$
$f(x)$	objective function in a primal problem
$x \in X$	regional constraints, $X \subseteq \mathbb{R}^n$
$g(x) \le b$, $g(x) = b$	functional constraints, $g : \mathbb{R}^n \to \mathbb{R}^m$
z	slack variable, $z \in \mathbb{R}^m$
$Ax \le b$, $Ax + z = b$	linear constraints
$c^\top x$	linear objective function
$L(x, \lambda)$	Lagrangian, $L(x, \lambda) = f(x) - \lambda^\top(g(x) - b)$
$\lambda \in Y$	$Y = \{\lambda : \min_{x \in X} L(x, \lambda) > -\infty\}$.
B, N	sets of indices of basic and non-basic components of x.
A, p, q	pay-off matrix and decision variables in a matrix game
x_{ij}	flow on arc (i, j)
v, $C(S, \bar{S})$	value of a flow, value of cut $(S, \bar{S})$
c_{ij}^-, c_{ij}^+	minimum/maximum allowed flows on arc (i, j)
d_{ij}	costs per unit flow on arc (i, j)
s_i, d_j	source and demands amounts in transportation problem
λ_i, μ_j	node numbers in transportation algorithm

Index

1 Preliminaries

1.1 Linear programming

Consider the problem P.

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

This is a completely linear problem – the objective function and all constraints are linear. In matrix/vector notation we can write a typical **linear program** (LP) as

$$\text{P: maximize } c^\top x \quad \text{s.t. } Ax \leq b,\ x \geq 0,$$

1.2 Optimization under constraints

The general type of problem we study in this course takes the form

$$\begin{array}{rc} \text{maximize} & f(x) \\ \text{subject to} & g(x) = b \\ & x \in X \end{array}$$

where

$$\begin{array}{rcll} x & \in & \mathbb{R}^n & (n \text{ decision variables}) \\ f : \mathbb{R}^n & \to & \mathbb{R} & (\text{objective function}) \\ X & \subseteq & \mathbb{R}^n & (\text{regional constraints}) \\ g : \mathbb{R}^n & \to & \mathbb{R}^m & (m \text{ functional equations}) \\ b & \in & \mathbb{R}^m & \end{array}$$

Note that minimizing $f(x)$ is the same as maximizing $-f(x)$. We will discuss various examples of constrained optimization problems. We will also talk briefly about ways our methods can be applied to real-world problems.

1.3 Representation of constraints

We may wish to impose a constraint of the form $g(x) \leq b$. This can be turned into an equality constraint by the addition of a **slack variable** z. We write

$$g(x) + z = b, \quad z \geq 0.$$

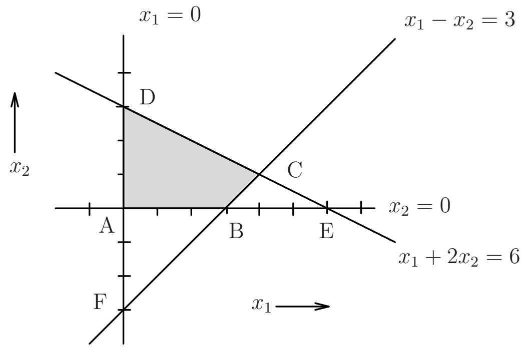
It is frequently mathematically convenient to turn all our constraints into equalities.

We distinguish between **functional constraints** of the form $g(x) = b$ and **regional constraints** of the form $x \in X$.

Together, these define the **feasible set** for x. Typically, 'obvious' constraints like $x \geq 0$ are catered for by defining X in an appropriate way and more complicated constraints, that may change from instance to instance of the problem, are expressed by functional constraints $g(x) = b$ or $g(x) \leq b$.

Sometimes the choice is made for mathematical convenience. Methods of solution typically treat regional and functional constraints differently.

The shaded region shown below is the feasible set defined by the constraints for problem P.

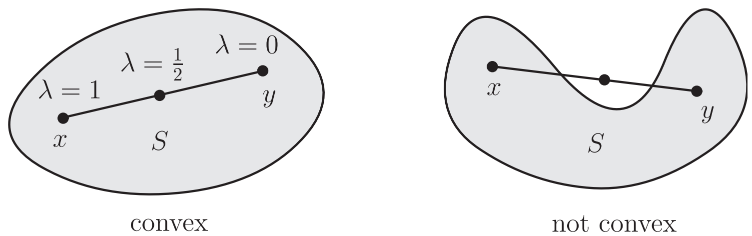


The feasible set for P is a convex set.

1.4 Convexity

Definition 1.1. *A set $S \subseteq \mathbb{R}^n$ is a* **convex set** *if $x, y \in S \implies \lambda x + (1-\lambda)y \in S$ for all $x, y \in S$ and $0 \leq \lambda \leq 1$.*

In other words, the line segment joining x and y lies in S.



The following theorem is easily proved.

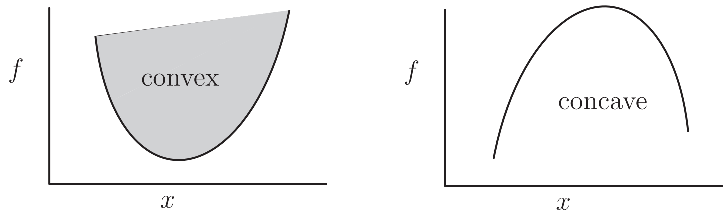
Theorem 1.1. *The feasible set of a LP problem is convex.*

For functions defined on convex sets we make the following further definitions.

Definition 1.2. *A function $f : S \to \mathbb{R}$ is a* **convex function** *if the set above its graph is convex. Equivalently, if*

$$\lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y), \quad \text{for all } 0 \leq \lambda \leq 1.$$

A function f is a **concave function** *if $-f$ is convex.*



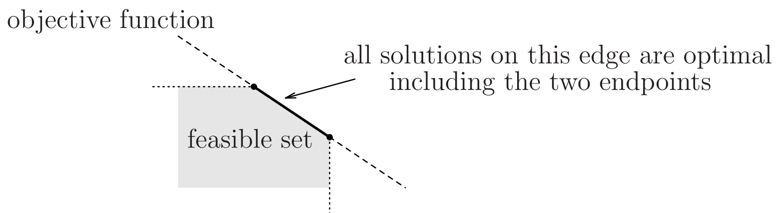
In a general problem of minimizing a general f over a general S there may be local minima of f which are not global minimal. It is usually difficult to find the global minimum when there are lots of local minima.

This is why convexity is important: *if S is a convex set and f is a convex function then any local minimum of f is also a global minimum.*

A linear function (as in LP) is both concave and convex, and so all local optima of a linear objective function are also global optima.

1.5 Extreme points and optimality

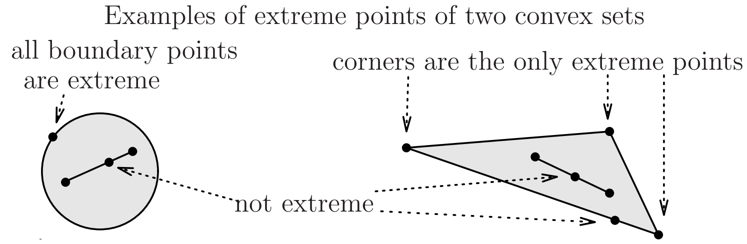
Notice that in problem P the optimum of $c^\top x$ occurs at a 'corner' of the feasible set, regardless of what is the linear objective function. In our case, $c^\top = (1, 1)$ and the maximum is at corner C.



If the objective function is parallel to an edge, then there may be other optima on that edge, but there is always an optimum at a corner. This motivates the following definition.

Definition 1.3. *We say that x is an* **extreme point** *of a convex set S if whenever $x = \theta y + (1-\theta) z$, for $y, z \in S$, $0 < \theta < 1$, then $x = y = z$.*

In other words, x is not in the interior of any line segment within S.



Theorem 1.2. *If an LP has a finite optimum it has an optimum at an extreme point of the feasible set.*

For LP problems the feasible set will always have a finite number of extreme points (vertices). The feasible set is 'polyhedral', though it may be bounded or unbounded. This suggests the following algorithm for solving LPs.

Algorithm:

1. Find all the vertices of the feasible set.
2. Pick the best one.

This will work, but there may be very many vertices. In fact, for $Ax \leq b$, $x \geq 0$, there can be $\binom{n+m}{m}$ vertices. So if $m = n$, say, then the number of vertices is of order $(2n)^n$, which increases exponentially in n. This is **not** a good algorithm!

1.6 *Efficiency of algorithms*

There is an important distinction between those algorithms whose running times (in the worst cases) are exponential functions of 'problem size', e.g., $(2n)^n$, and those algorithms whose running times are polynomial functions of problem size, e.g., n^k. For example, the problem of finding the smallest number in a list of n numbers is solvable in polynomial-time n by simply scanning the numbers. There is a beautiful theory about the **computational complexity** of algorithms and one of its main messages is that problems solvable in polynomial-time are the 'easy' ones.

We shall be learning the **simplex algorithm**, due to Dantzig, 1947. In worst-case instances it does not run in polynomial-time. In 1974, Khachian discovered a polynomial-time algorithm for general LP problems (the ellipsoid method). It is of mainly theoretical interest, being slow in practice. In 1984, Karmarkar discovered a new polynomial-time algorithm (an interior point method) that competes in speed with the simplex algorithm.

In contrast, no polynomial-time algorithm is known for general **integer LP**, in which x is restricted to be integer-valued. ILP includes important problems such as bin packing, job-shop scheduling, traveling salesman and many other essentially equivalent problems of a combinatorial nature.

2 Lagrangian Methods

2.1 The Lagrangian sufficiency theorem

Suppose we are given a general optimization problem,

$$\text{P:}\quad \text{minimize } f(x) \quad \text{s.t. } g(x) = b, \quad x \in X,$$

with $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ (n variables and m constraints). The **Lagrangian** is

$$L(x, \lambda) = f(x) - \lambda^\top (g(x) - b),$$

with $\lambda \in \mathbb{R}^m$ (one component for each constraint). Each component of λ is called a **Lagrange multiplier**.

The following theorem is simple to prove, and extremely useful in practice.

Theorem 2.1 (Lagrangian sufficiency theorem)**.** *If x^* and λ^* exist such that x^* is feasible for P and*

$$L(x^*, \lambda^*) \le L(x, \lambda^*) \quad \forall x \in X,$$

then x^* is optimal for P.

Proof. Define

$$X_b = \{x : x \in X \text{ and } g(x) = b\}.$$

Note that $X_b \subseteq X$ and that for any $x \in X_b$

$$L(x, \lambda) = f(x) - \lambda^\top (g(x) - b) = f(x).$$

Now $L(x^*, \lambda^*) \le L(x, \lambda^*)$ for all $x \in X$, by assumption, and in particular for $x \in X_b$. So

$$f(x^*) = L(x^*, \lambda^*) \le L(x, \lambda^*) = f(x), \quad \text{for all } x \in X_b.$$

Thus x^* is optimal for P. □

Remarks.

1. Note the 'If' which starts the statement of the theorem. There is no guarantee that we can find a λ^* satisfying the conditions of the theorem for general problems P. (However, there is a large class of problems for which λ^* do exist.)

2. At first sight the theorem offers us a method for testing that a solution x^* is optimal for P without helping us to find x^* if we don't already know it. Certainly we will sometimes use the theorem this way. But for some problems, there is a way we can use the theorem to find the optimal x^*.

2.2 Example: use of the Lagrangian sufficiency theorem

Example 2.1.

$$\begin{aligned} &\text{minimize } x_1 - x_2 - 2x_3 \\ &\text{s.t. } x_1 + x_2 + x_3 = 5 \\ &x_1^2 + x_2^2 = 4 \\ &x \in X = \mathbb{R}^3. \end{aligned}$$

Solution. Since we have two constraints we take $\lambda \in \mathbb{R}^2$ and write the Lagrangian

$$\begin{aligned} L(x, \lambda) &= f(x) - \lambda^\top (g(x) - b) \\ &= x_1 - x_2 - 2x_3 - \lambda_1(x_1 + x_2 + x_3 - 5) - \lambda_2(x_1^2 + x_2^2 - 4) \\ &= \Big[x_1(1 - \lambda_1) - \lambda_2 x_1^2\Big] + \Big[x_2(-1 - \lambda_1) - \lambda_2 x_2^2\Big] \\ &\quad + \Big[-x_3(2 + \lambda_1)\Big] + 5\lambda_1 + 4\lambda_2. \end{aligned}$$

We first try to minimize $L(x, \lambda)$ for fixed λ in $x \in \mathbb{R}^3$. Notice that we can minimize each square bracket separately.

First notice that $-x_3(2+\lambda_1)$ has minimum $-\infty$ unless $\lambda_1 = -2$. So we only want to consider $\lambda_1 = -2$.

Observe that the terms in x_1, x_2 have a finite minimum only if $\lambda_2 < 0$, in which case the minimum occurs at a stationary point where,

$$\begin{aligned} \partial L/\partial x_1 &= 1 - \lambda_1 - 2\lambda_2 x_1 = 0 \implies x_1 = 3/2\lambda_2 \\ \partial L/\partial x_2 &= -1 - \lambda_1 - 2\lambda_2 x_2 = 0 \implies x_2 = 1/2\lambda_2. \end{aligned}$$

Let Y be the set of (λ_1, λ_2) such that $L(x, \lambda)$ has a finite minimum. So

$$Y = \{\lambda : \lambda_1 = -2,\ \lambda_2 < 0\},$$

and for $\lambda \in Y$ the minimum of $L(x, \lambda)$ occurs at $x(\lambda) = (3/2\lambda_2, 1/2\lambda_2, x_3)^\top$.

Now to find a feasible $x(\lambda)$ we need

$$x_1^2 + x_2^2 = 4 \implies \frac{9}{4\lambda_2^2} + \frac{1}{4\lambda_2^2} = 4 \implies \lambda_2 = -\sqrt{5/8}.$$

So $x_1 = -3\sqrt{2/5}$, $x_2 = -\sqrt{2/5}$ and $x_3 = 5 - x_1 - x_2 = 5 + 4\sqrt{2/5}$.

The conditions of the Lagrangian sufficiency theorem are satisfied by

$$x^* = \left(-3\sqrt{2/5}, -\sqrt{2/5}, 5 + 4\sqrt{2/5}\right)^\top \text{ and } \lambda^* = \left(-2, -\sqrt{5/8}\right)^\top.$$

So x^* is optimal. ■

2.3 Strategy to solve problems with the Lagrangian sufficiency theorem

Attempt to find x^*, λ^* satisfying the conditions of the theorem as follows.

1. For each λ solve the problem
$$\text{minimize } L(x,\lambda) \text{ subject to } x \in X.$$
Note that the only constraints involved in this problem are $x \in X$ so this should be an easier problem to solve than P.

2. Define the set
$$Y = \{\lambda : \min_{x\in X} L(x,\lambda) > -\infty\}.$$
If we obtain $-\infty$ for the minimum in step 1 then that λ is no good. We consider only those $\lambda \in Y$ for which we obtain a finite minimum.

3. For $\lambda \in Y$, the minimum will be obtained at some $x(\lambda)$ (that depends on λ in general). Typically, $x(\lambda)$ will not be feasible for P.

4. Adjust $\lambda \in Y$ so that $x(\lambda)$ is feasible. If $\lambda^* \in Y$ exists such that $x^* = x(\lambda^*)$ is feasible then x^* is optimal for P by the theorem.

2.4 Example: further use of the Lagrangian sufficiency theorem

Example 2.2.

$$\text{minimize } \frac{1}{1+x_1} + \frac{1}{2+x_2} \quad \text{s.t. } x_1 + x_2 = b, \quad x_1, x_2 \geq 0.$$

Solution. We define $X = \{x : x \geq 0\}$ and the Lagrangian

$$\begin{aligned} L(x,\lambda) &= \frac{1}{1+x_1} + \frac{1}{2+x_2} - \lambda(x_1 + x_2 - b) \\ &= \left(\frac{1}{1+x_1} - \lambda x_1\right) + \left(\frac{1}{2+x_2} - \lambda x_2\right) + \lambda b. \end{aligned}$$

Note that
$$\left(\frac{1}{1+x_1} - \lambda x_1\right) \text{ and } \left(\frac{1}{2+x_2} - \lambda x_2\right)$$
do not have a finite minimum in $x \geq 0$ unless $\lambda \leq 0$. So we take $\lambda \leq 0$. Observe that in the range $x \geq 0$ a function of the form $\left(\frac{1}{a+x} - \lambda x\right)$ will have its minimum either at $x = 0$, if this function is increasing at 0, or at the stationary point of the function, occurring where $x > 0$, if the function is decreasing at 0. So the minimum occurs at

$$x = \begin{cases} 0 \\ -a + \sqrt{-1/\lambda} \end{cases} \quad \text{as} \quad \sqrt{-1/\lambda} \begin{matrix} \leq \\ \geq \end{matrix} a$$

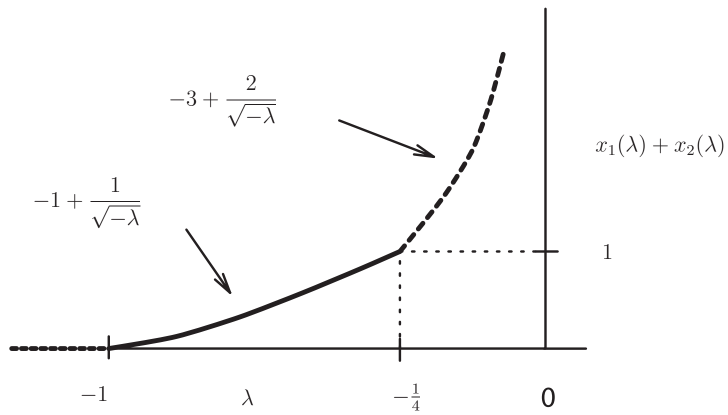
so defining $c^+ = \max(0, c)$,

$$x(\lambda) = \left(-a + \sqrt{-1/\lambda}\right)^+.$$

At first sight it appears that we don't know which values of x_1, x_2 to substitute into the constraint until we know λ, and we don't know λ until we substitute x_1, x_2 into the constraint. But notice that $x_1(\lambda) + x_2(\lambda)$ satisfies

$$\begin{aligned} x_1(\lambda) + x_2(\lambda) &= \left(-1 + \sqrt{-1/\lambda}\right)^+ + \left(-2 + \sqrt{-1/\lambda}\right)^+ \\ &= \begin{cases} 0 & & \leq -1 \\ -1 + 1/\sqrt{-\lambda} & \text{as } \lambda & \in [-1, -1/4] \\ -3 + 2/\sqrt{-\lambda} & & \in [-1/4, 0] \end{cases} \end{aligned}$$

So we can see that $x_1(\lambda)+x_2(\lambda)$ is an increasing and continuous function (although it is not differentiable at $\lambda = -1$ and at $\lambda = -1/4$).



Thus (by the Intermediate Value Theorem) for any $b > 0$ there will be a unique value of λ, say λ^*, for which $x_1(\lambda^*) + x_2(\lambda^*) = b$. This λ^* and corresponding x^* will satisfy the conditions of the theorem and so x^* is optimal. ■

Examples of this kind are fairly common.

3 The Lagrangian Dual

3.1 The Lagrangian dual problem

We have defined the set

$$Y = \{\lambda : \min_{x \in X} L(x,\lambda) > -\infty\}.$$

For $\lambda \in Y$ define

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

The following theorem is almost as easy to prove as the sufficiency theorem.

Theorem 3.1 (weak duality theorem). *For any feasible $x \in X_b$ and any $\lambda \in Y$*

$$L(\lambda) \le f(x).$$

Proof. For $x \in X_b$, $\lambda \in Y$,

$$f(x) = L(x,\lambda) \ge \min_{x \in X_b} L(x,\lambda) \ge \min_{x \in X} L(x,\lambda) = L(\lambda).$$

□

Thus, provided the set Y is non-empty, we can pick any $\lambda \in Y$, and observe that $L(\lambda)$ is a lower bound for the minimum value of the objective function $f(x)$.

We can now try and make this lower bound as great as possible, i.e., let us consider the problem

$$\text{D: maximize } L(\lambda) \text{ subject to } \lambda \in Y,$$

equivalently,

$$\text{D: } \underset{\lambda \in Y}{\text{maximize}} \left\{ \min_{x \in X} L(x,\lambda) \right\}.$$

This is known as the **Lagrangian dual problem**. The original problem is called the **primal problem**. The optimal value of the dual is $\le$ the optimal value of the primal. If they are equal (as happens in LP) we say there is **strong duality**.

Notice that the idea of a dual problem is quite general. For example, we can look again at the two examples we just studied.

Example 3.1. In Example 2.1 we had $Y = \{\lambda : \lambda_1 = -2, \lambda_2 < 0\}$ and that $\min_{x \in X} L(x,\lambda)$ occurred for $x(\lambda) = (3/2\lambda_2, 1/2\lambda_2, x_3)$. Thus

$$L(\lambda) = L(x(\lambda),\lambda) = \frac{10}{4\lambda_2} - 10 + 4\lambda_2\,.$$

The dual problem is thus

$$\underset{\lambda_2 < 0}{\text{maximize}} \left\{ \frac{10}{4\lambda_2} - 10 + 4\lambda_2 \right\}.$$

The max is at $\lambda_2 = -\sqrt{5/8}$, and the primal and dual have the same optimal value, namely $-2(\sqrt{10}+5)$.

Example 3.2. In Example 2.2, $Y = \{\lambda : \lambda \le 0\}$. By substituting the optimal value of x into $L(x,\lambda)$ we obtain

$$L(\lambda) = \begin{cases} 3/2 + \lambda b & \le -1 \\ 1/2 + 2\sqrt{-\lambda} + (b+1)\lambda \quad \text{as} \quad \lambda & \in [-1,-1/4] \\ 4\sqrt{-\lambda} + (b+3)\lambda & \in [-1/4, 0] \end{cases}$$

We can solve the dual problem, which is maximize $L(\lambda)$ s.t. $\lambda \le 0$. The solution lies in $-1 \le \lambda \le -1/4$ if $0 \le b \le 1$ and in $-1/4 \le \lambda \le 0$ if $1 \le b$. You can confirm that for all b the primal and dual here have the same optimal values.

3.2 The dual problem for LP

Construction of the dual problem for LP is straightforward. Consider the primal problem P:

$$\begin{gathered} \text{maximize } c^\top x \\ \text{subject to } Ax \le b,\ x \ge 0 \\ \text{equivalently } Ax + z = b,\ x, z \ge 0. \end{gathered}$$

Write the Lagrangian

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

As in the general case, we can find the set Y such that $\lambda \in Y$ implies $\max_{x,z \ge 0} L(x,z,\lambda)$ is finite, and for $\lambda \in Y$ we compute the minimum of $L(\lambda)$.

Consider the linear term $-\lambda^\top z$. If any coordinate $\lambda_i < 0$ we can make $-\lambda_i z_i$ as large as we like, by taking z_i large. So there is only a finite maximum in $z \ge 0$ if $\lambda_i \ge 0$ for all i.

Similarly, considering the term $(c^\top - \lambda^\top A)x$, this can be made as large as we like unless $(c^\top - \lambda^\top A)_i \le 0$ for all i. Thus

$$Y = \{\lambda : \lambda \ge 0,\ \lambda^\top A - c^\top \ge 0\}.$$

If we pick a $\lambda \in Y$ then $\max_{z \ge 0} -\lambda^\top z = 0$ (by choosing $z_i = 0$ if $\lambda_i > 0$ and any z_i if $\lambda_i = 0$) and also $\max_{x \ge 0} (c^\top - \lambda^\top A)x = 0$ similarly. Thus for $\lambda \in Y$, $L(\lambda) = \lambda^\top b$. So a pair of primal P, and dual D is,

$$\begin{array}{lllll} \text{P:} & \text{maximize } c^\top x & \text{s.t.} & Ax \le b, & x \ge 0 \\ \text{D:} & \text{minimize } \lambda^\top b & \text{s.t.} & \lambda^\top A \ge c^\top, & \lambda \ge 0. \end{array}$$

Notice that D is itself a linear program. For example,

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

$$\begin{array}{llrcl} \text{D:} & \text{minimize} & 6\lambda_1 + 3\lambda_2 & & \\ & \text{subject to} & \lambda_1 + \lambda_2 & \geq & 1 \\ & & 2\lambda_1 - \lambda_2 & \geq & 1 \\ & & \lambda_1, \lambda_2 & \geq & 0 \end{array}$$

Furthermore, we might write D as

$$\text{D:} \quad \text{maximize} \ \ (-b)^\top \lambda \ \ \text{s.t.} \quad (-A)^\top \lambda \leq (-c), \quad \lambda \geq 0.$$

So D is of the same form as P, but with $c \to -b$, $b \to -c$, and $A \to -A^\top$. This means that the dual of D is P, and so we have proved the following lemma.

Lemma 3.2. *In linear programming, the dual of the dual is the primal.*

3.3 The weak duality theorem in the case of LP

We can now apply Theorem 3.1 directly to P and D to obtain the following.

Theorem 3.3 (**weak duality theorem for LP**)**.** *If x is feasible for P (so $Ax \leq b$, $x \geq 0$) and λ is feasible for D (so $\lambda \geq 0$, $A^\top \lambda \geq c$) then $c^\top x \leq \lambda^\top b$.*

Since this is an important result it is worth knowing a proof for this particular case which does not appeal to the general Theorem 3.1. Naturally enough, the proof is very similar.

Proof. Write

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

where $Ax + z = b$, $z \geq 0$. Now for x and λ satisfying the conditions of the theorem,

$$c^\top x \ = \ L(x, z, \lambda) \ = \ (c^\top - \lambda^\top A)x - \lambda^\top z + \lambda^\top b \ \leq \ \lambda^\top b.$$

□

3.4 Sufficient conditions for optimality

Theorem 3.3 provides a quick proof of the sufficient conditions for optimality of x^*, z^* and λ^* in a P and D.

Theorem 3.4 (**sufficient conditions for optimality in LP**)**.** If x^*, z^* is feasible for P and λ^* is feasible for D and $(c^\top - \lambda^{*\top} A)x^* = \lambda^{*\top} z^* = 0$ (complementary slackness) then x^* is optimal for P and λ^* is optimal for D. Furthermore $c^\top x^* = \lambda^{*\top} b$.

Proof. Write $L(x^*, z^*, \lambda^*) = c^\top x^* - \lambda^{*\top}(Ax^* + z^* - b)$. Now

$$\begin{aligned} c^\top x^* &= L(x^*, z^*, \lambda^*) \\ &= (c^\top - \lambda^{*\top} A)x^* - \lambda^{*\top} z^* + \lambda^{*\top} b \\ &= \lambda^{*\top} b \end{aligned}$$

But for all x feasible for P we have $c^\top x \leq \lambda^{*\top} b$ (by the weak duality theorem 3.3) and this implies that for all feasible x, $c^\top x \leq c^\top x^*$. So x^* is optimal for P. Similarly, λ^* is optimal for D (and the problems have the same optimums). □

The conditions $(c^\top - \lambda^{*\top} A)x^* = 0$ and $\lambda^{*\top} z^* = 0$ are called **complementary slackness** conditions.

3.5 The utility of primal-dual theory

Why do we care about D instead of just getting on with the solution of P?

1. It is sometimes easier to solve D than P (and they have the same optimal values).
2. For some problems it is natural to consider both P and D together (e.g., two person zero-sum games, see Lecture 9).
3. Theorem 3.4 says that for optimality we need three things: primal feasibility, dual feasibility and complementary slackness.

   Some algorithms start with solutions that are primal feasible and work towards dual feasibility. Others start with dual feasibility. Yet others (in particular network problems) alternately look at the primal and dual variables and move towards feasibility for both at once.

4 Solutions to Linear Programming Problems

4.1 Basic solutions

Let us return to the LP problem P in Lecture 1 and look for a more algebraic (less geometric) characterisation of the extreme points. Let us rewrite P with equality constraints, using slack variables.

$$\begin{array}{lrcl} \text{P: maximize} & x_1 + x_2 & & \\ \text{subject to} & x_1 + 2x_2 + z_1 & = & 6 \\ & x_1 - x_2 + z_2 & = & 3 \\ & x_1, x_2, z_1, z_2 & \geq & 0 \end{array}$$

Let us calculate the value of the variables at each of the 6 points marked A–F in our picture of the feasible set for P. The values are:

	x_1	x_2	z_1	z_2	f
A	0	0	6	3	0
B	3	0	3	0	3
C	4	1	0	0	5
D	0	3	0	6	3
E	6	0	0	−3	6
F	0	−3	12	0	−3

At each point there are two zero and two non-zero variables. This is not surprising.

Geometrically: The 4 lines defining the feasible set can be written $x_1 = 0$; $x_2 = 0$; $z_1 = 0$; $z_2 = 0$. At the intersection of each pair of lines, two variables are zero.

Algebraically: Constraints $Ax + z = b$ are 2 equations in 4 unknowns. If we choose 2 variables (which can be done in $\binom{4}{2} = 6$ ways) and set them equal to zero we will be left with two equations in the other two variables. So (provided A and b are 'nice') there will be a unique solution for the two non-zero variables.

Instead of calling the slack variables z_1 and z_2, let us call them x_3 and x_4 so that we can write P as

$$\begin{array}{lrcl} \text{P: maximize} & x_1 + x_2 & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0 \end{array}$$

Note we have to extend A to $\begin{pmatrix} 1 & 2 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix}$ and x to $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$.

A is $(m \times n)$ with $n > m$ and there are m equations in $n > m$ unknowns. We can

choose $n - m$ variables in $\binom{n}{m}$ ways. Set them to zero. There is a unique solution to $Ax = b$ for the remaining m variables (provided A and b are 'nice').

Definition 4.1.

- *A* **basic solution** *to $Ax = b$ is a solution with at least $n - m$ zero variables.*
- *A basic solution is* **non-degenerate** *if exactly $n - m$ variables are zero.*
- *The choice of the m non-zero variables is called the* **basis**. *Variables in the basis are called* **basic**; *the others are called* **non-basic**.
- *If a basic solution satisfies $x \geq 0$ then it is called a* **basic feasible solution**.

So A–F are basic solutions (and non-degenerate) and A–D are basic feasible solutions. Henceforth, we make an assumption.

Assumption 4.1. The $m \times n$ matrix A has the property that

- The rank of A is m.
- Every set of m columns of A is linearly independent.
- If x is a b.f.s. of P, then x has exactly m non-zero entries. (*non-degeneracy*)

Theorem 4.1. *Basic feasible solutions $\equiv$ extreme points of the feasible set.*

Proof. Suppose x is a b.f.s. Then x has exactly m non-zero entries. Suppose $x = \theta y + (1 - \theta)z$ for feasible y, z and $0 < \theta < 1$. Then if the ith entry of x is non-basic then $x_i = 0$ and hence $y_i = z_i = 0$, since $y_i, z_i \geq 0$. This means both y and z have at least $n - m$ zero entries. The equation $Ay = b = Az$ implies $A(y - z) = 0$. Since at most m entries of $y - z$ are non-zero, and any set of m columns of A are linearly independent, we have $y = z$, and x is an extreme point as claimed.

Now suppose x is feasible and extreme but not basic. Then x has $r (> m)$ non-zero entries, say $x_{i_1}, \dots, x_{i_r} > 0$. Let a_i denote the ith column of A. Since the columms $a_{i_1}, \dots, a_{i_r}$ are linearly dependent, there exists non-zero numbers $y_{i_1}, \dots, y_{i_r}$ such that $y_{i_1} a_{i_1} + \cdots + y_{i_r} a_{i_r} = 0$. Set $y_i = 0$ if $i \neq i_1, \dots, i_r$. Now $Ay = 0$, so $A(x \pm \epsilon y) = b$. We can choose $\epsilon > 0$ small enough so that both $x + \epsilon y$ and $x - \epsilon y$ are feasible. Hence, we have succeeded in expressing x as a convex combination of two distinct points of X_b since $x = \frac{1}{2}(x + \epsilon y) + \frac{1}{2}(x - \epsilon y)$. That is, x is not extreme. □

Theorem 4.2. *If an LP has a finite solution then there is an optimal basic feasible solution.*

Proof. Suppose x is an optimal solution, but is not basic. Then there exists nonzero y s.t. $x_i = 0 \implies y_i = 0$ and $Ay = 0$. Consider $x(\epsilon) = x + \epsilon y$. Clearly there exist some ϵ chosen positive or negative so that $c^\top x(\epsilon) \geq c^\top x$, and such that $x(\epsilon) \geq 0$, and $Ax(\epsilon) = Ax \leq b$, but $x(\epsilon)$ has fewer nonzero entries than x. □

Taking Theorems 4.1 and 4.2 together, we have proved Theorem 1.2. So we can do algebra instead of drawing a picture (which is good for computer and good for us if there are many variables and constraints.) A simple (and foolish) algorithm can be stated:

Algorithm:

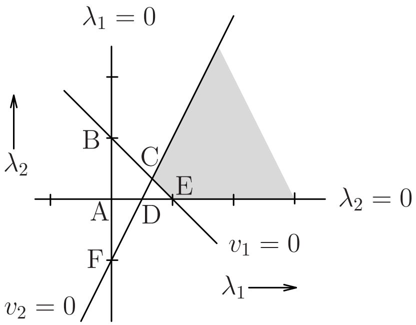
1. Find all the basic solutions.
2. Test to see which are feasible.
3. Choose the best basic feasible solution.

Unfortunately it is not usually easy to know which basic solutions will turn out to be feasible before calculating them. Hence, even though there are often considerably fewer basic feasible solutions we will still need to calculate all $\binom{n}{m}$ basic solutions.

4.2 Primal-dual relationships

Now let us look at problem D which can be written, after introducing slack variables v_1 and v_2 as

$$\begin{array}{llrcl} \text{D:} & \text{minimize} & 6\lambda_1 + 3\lambda_2 & & \\ & \text{subject to} & \lambda_1 + \lambda_2 - v_1 & = & 1 \\ & & 2\lambda_1 - \lambda_2 - v_2 & = & 1 \\ & & \lambda_1, \lambda_2, v_1, v_2 & \geq & 0 \end{array}$$



The value of the variables, etc., at the points A–F in P (as above) and D are:

P:	x_1	x_2	z_1	z_2	f
A	0	0	6	3	0
B	3	0	3	0	3
C	4	1	0	0	5
D	0	3	0	6	3
E	6	0	0	-3	6
F	0	-3	12	0	-3

D:	v_1	v_2	λ_1	λ_2	f
A	-1	-1	0	0	0
B	0	-2	0	1	3
C	0	0	$\frac{2}{3}$	$\frac{1}{3}$	5
D	$-\frac{1}{2}$	0	$\frac{1}{2}$	0	3
E	0	1	1	0	6
F	-2	0	0	-1	-3

Observe, that for D, as for P above, there are two zero and two non-zero variables at each intersection (basic solution). C and E are feasible for D. The optimum is at C with optimum value 5 (assuming we are minimizing and the other basic solutions are not feasible.)

We make the following observations by comparing lists of basic solutions for P and D.

1. For each basic solution for P there is a corresponding basic solution for D. [Labels A–F have been chosen so that corresponding solutions have the same labels.] Each pair

   (a) has the same value of the objective function.

   (b) satisfies **complementary slackness**, i.e., $x_i v_i = 0$, $\lambda_i z_i = 0$,

   so for each corresponding pair,

P		D
variables x		constraints
x_i basic ($x_i \neq 0$)	$\Longleftrightarrow$	constraint: tight ($v_i = 0$)
x_i non-basic ($x_i = 0$)	$\Longleftrightarrow$	constraint: slack ($v_i \neq 0$)
constraints		variables λ
constraint: tight ($z_i = 0$)	$\Longleftrightarrow$	λ_i basic ($\lambda_i \neq 0$)
constraint: slack ($z_i \neq 0$)	$\Longleftrightarrow$	λ_i non-basic ($\lambda_i = 0$)

   (These conditions determine which basic solutions in P and D are paired; the implications go both ways because in this example all basic solutions are non-degenerate.)

2. There is only one pair that is feasible for both P and D, and that solution is C, which is optimal, with value 5, for both.

3. For any x feasible for P and λ feasible for D we have $c^\top x \leq b^\top \lambda$ with equality if and only if x, λ are optima for P and D.

This correspondence between P and D is so symmetrical and pretty that it feels as though it ought to be obvious why it works. Indeed we have already proved the following:

Lemma 3.2 *In linear programming, the dual of the dual is the primal.*

Theorem 3.3 (**weak duality in LP**). *If x is feasible for P and λ is feasible for D then $c^\top x \leq b^\top \lambda$. (In particular, if one problem is feasible then the other is bounded.)*

Theorem 3.4 (**sufficient conditions for optimality in LP**). *If x is feasible for P and λ is feasible for D, and x, λ satisfy complementary slackness, then x is optimal for P and λ is optimal for D. Furthermore $c^\top x = \lambda^\top b$.*

The following will be proved in Lecture 7.

Theorem 4.3 (**strong duality in LP**). *If both P and D are feasible (each has at least one feasible solution), then there exists x, λ satisfying the conditions of Theorem 3.4 above.*

5 The Simplex Method

5.1 Preview of the simplex algorithm

Let us look very closely at problem P and see if we can construct an algorithm that behaves as follows.

Simplex algorithm

1. Start with a basic feasible solution.
2. Test — is it optimal?
3. If YES — stop.
4. If NO, move to 'adjacent' and better b.f.s. Return to 2.

We need to pick a b.f.s. to start. Let us take vertex A.

$$x_1 = x_2 = 0;\ z_1 = 6,\ z_2 = 3.$$

[Even for very large problems it is easy to pick a b.f.s. provided the original constraints are m constraints in n variables, $Ax \le b$ with $b \ge 0$. Once we add slack variables $Ax + z = b$ we have $n + m$ variables and m constraints. If we pick $x = 0$, $z = b$ this is a b.f.s. More about picking the initial b.f. solutions in other cases later.]

Now we can write problem P as:

$$\begin{array}{lrcrcrcrcll} x_i, z_i \ge 0 & & & & & & & & & & \\ & x_1 & + & 2x_2 & + & z_1 & & & = & 6 & (1) \\ & x_1 & - & x_2 & & & + & z_2 & = & 3 & (2) \\ \max & x_1 & + & x_2 & & & & & = & f & (0) \end{array}$$

The peculiar arrangement on the page is deliberate. Now it is obvious that A is not optimal because,

1. At A, $x_1 = x_2 = 0$; $z_1 = 6$, $z_2 = 3$.
2. From the form of the objective function we see that increasing either x_1 or x_2 will improve the solution.
3. From (1) we see that it is possible to increase x_1 to 6 and decrease z_1 to 0 without violating this equation or making any variable infeasible.
4. From (2) we see that it is possible to increase x_1 to 3 and decrease z_2 to 0 before any variable becomes infeasible.
5. Taking (1) and (2) together, then, we can increase x_1 to 3, decrease z_1 to 3 and decrease z_2 to 0, preserving equality in the two constraints, preserving feasibility and increasing the objective function.

That is, we should move to the b.f.s.

$$x_1 = 3,\ x_2 = 0,\ z_1 = 3,\ z_2 = 0,\ (f = 3),$$

which is vertex B. Note that one variable (x_1) has entered the basis and one (z_2) has left; i.e., we have moved to an 'adjacent' vertex. Why was this easy to see?

1. The objective function f was written in terms of the non-basic variables ($x_1 = x_2 = 0$), so it was easy to see that increasing one of them would improve the solution.
2. Each basic variable (z_1, z_2) appeared just once in one constraint, so we could consider the effect of increasing x_1 on each basic variable separately when deciding how much we could increase x_1.

This suggests we try and write the problem so that the conditions above hold at B, our new b.f.s.. We can do this by adding multiples of the second equation to the others (which is obviously allowed as we are only interested in variables satisfying the constraints.)

So P can be written,

$$\begin{array}{llrcrcrcrcll} x_i, z_i \ge 0 & & & & & & & & & & & \\ (1)-(2) & & & & 3x_2 & + & z_1 & - & z_2 & = & 3 & (1)' \\ (2) & & x_1 & - & x_2 & & & + & z_2 & = & 3 & (2)' \\ (0)-(2) & & & & 2x_2 & & & - & z_2 & = & f-3 & (0)' \end{array}$$

This form of P (equivalent to the original) is what we wanted.

1. The objective function is written in terms of the non-basic variables x_2, z_2.
2. Basic variables x_1, z_1 each appear just once in one constraint.

The next step is now easy. Remember we are at B:

$$x_1 = 3,\ x_2 = 0,\ z_1 = 3,\ z_2 = 0,\ (f = 3).$$

1. From the objective function it is obvious that increasing x_2 is good, whereas increasing z_2 would be bad (and since z_2 is zero we can only increase it).
2. Equation $(1)'$ shows that we can increase x_2 to 1 and decrease z_1 to 0.
3. Equation $(2)'$ doesn't impose any restriction on how much we can increase x_2 (we just would need to increase x_1 also.)
4. Thus we can increase x_2 to 1, while decreasing z_1 to 0 and increasing x_1 to 4.

So we move to vertex C:

$$x_1 = 4,\ x_2 = 1,\ z_1 = 0,\ z_2 = 0,\ (f = 5).$$

Now, rewriting the problem again into the desired form for vertex C we obtain

$$\begin{array}{llcccl} x_i, z_i \geq 0 & & & & & \\ \frac{1}{3}(1)' & & x_2 & +\frac{1}{3}z_1 & -\frac{1}{3}z_2 & = 1 \\ (2)' + \frac{1}{3}(1)' & x_1 & & +\frac{1}{3}z_1 & +\frac{2}{3}z_2 & = 4 \\ (0)' - \frac{2}{3}(1)' & & & -\frac{2}{3}z_1 & -\frac{1}{3}z_2 & = f - 5 \end{array}$$

Now it is clear that we have reached the optimum since

1. We know $x_1 = 4, x_2 = 1, z_1 = 0, z_2 = 0$ is feasible.
2. We know that for any feasible x, z we have $f = 5 - \frac{2}{3}z_1 - \frac{1}{3}z_2 \leq 5$. So clearly our solution (with $z_1 = z_2 = 0$) is the best possible.

5.2 The simplex algorithm

The procedure just described for problem P can be formalised. Rather than writing out all the equations each time we write just the coefficients in a table known as the **simplex tableau**. To repeat what we have just done, we would write:–

	x_1	x_2	z_1	z_2	a_{i0}
z_1 basic	1	2	1	0	6
z_2 basic	1	−1	0	1	3
a_{0j}	1	1	0	0	0

If we label the coefficients in the body of the table (a_{ij}), the right hand sides of the equations (a_{i0}), the coefficients in the expression for the objective function as (a_{0j}) and the value of the objective function $-a_{00}$, so the tableau contains

(a_{ij})	a_{i0}
a_{0j}	a_{00}

The algorithm is

1. **Choose a variable to enter the basis.** Pick a j such that $a_{0j} > 0$. (The variable corresponding to column j will enter the basis.) If all $a_{0j} \leq 0$, $j \geq 1$, then the current solution is optimal.

   We picked $j = 1$, so x_1 is entering the basis.

2. **Find the variable to leave the basis.** Choose i to minimize a_{i0}/a_{ij} from the set $\{i : a_{ij} > 0\}$. If $a_{ij} \leq 0$ for all i then the problem is unbounded (see examples sheet) and the objective function can be increased without limit. If there is more than one i minimizing a_{i0}/a_{ij} the problem has a degenerate basic feasible solution (see example sheet.) For small problems you will be OK if you just choose any one of them and carry on regardless.

   We choose $i = 2$ since $3/1 < 6/1$, so the variable corresponding to equation 2 leaves the basis.

3. **Pivot on the element a_{ij}.** (i.e., get the equations into the appropriate form for the new basic solution.)

   (a) multiply row i by $1/a_{ij}$.

   (b) add $-(a_{kj}/a_{ij})\times$(row i) to each row $k \neq i$, including the objective function row.

   We obtain as before

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

which is the appropriate form for the tableau for vertex B.

Check that repeating these instructions on the new tableau, by pivoting on a_{12}, produces the appropriate tableau for vertex C.

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

Note that the columns of the tableau corresponding to the basic variables always make up the columns of an identity matrix.

Since the bottom row is now all ≤ 0 the algorithm stops. Don't hesitate to look back at Subsection 5.1 to see why we take these steps.

6 The Simplex Tableau

6.1 Choice of pivot column

We might have chosen the first pivot as a_{12} which would have resulted in

	x_1	x_2	z_1	z_2	a_{i0}
z_1 basic	1	2	1	0	6
z_2 basic	1	-1	0	1	3
a_{0j}	1	1	0	0	0

$\longrightarrow$

	x_1	x_2	z_1	z_2	a_{i0}
x_2 basic	$\frac{1}{2}$	1	$\frac{1}{2}$	0	3
z_2 basic	$\frac{3}{2}$	0	$\frac{1}{2}$	1	6
a_{0j}	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	-3

This is the tableau for vertex D. A further iteration, with pivot a_{21} takes us to the optimal solution at vertex C. Therefore both choices of initial pivot column resulted in it requiring two steps to reach the optimum.

Remarks.

1. In general, there is no way to tell in advance which choice of pivot column will result in the smallest number of iterations. We may choose any column where $a_{0j} > 0$. A common rule-of-thumb is to choose the column for which a_{0j} is greatest, since the objective function increases by the greatest amount per unit increase in the variable corresponding to that column.

2. At each stage of the simplex algorithm we have two things in mind.

   First — a particular choice of basis and basic solution.

   Second — a rewriting of the problem in a convenient form.

   There is always an identity matrix embedded in the tableau corresponding to the basic variables. Hence, when the non-basic variables are set to zero the equations are trivial to solve for the values of the basic variables. They are just given by the right-hand column.

   Check that provided we start with the equations written in this form in the initial tableau, the simplex algorithm rules ensure that we obtain an identity matrix at each stage.

3. The tableau obviously contains some redundant information. For example, provided we keep track of which equation corresponds to a basic variable, we could omit the columns corresponding to the identity matrix (and zeros in the objective row). This is good for computer programs, but it is probably better to keep the whole thing for hand calculation.

6.2 Initialization: the two-phase method

In our example there was an obvious basic feasible solution with which to start the simplex algorithm. This is not always the case. For example, suppose we have a problem like

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

which we wish to solve using the simplex algorithm. We can add slack variables (sometimes called **surplus variables** when they appear in $\geq$ constraints) to obtain

$$\begin{aligned} \text{maximize} \quad & -6x_1 - 3x_2 \\ \text{subject to} \quad & x_1 + x_2 - z_1 = 1 \\ & 2x_1 - x_2 - z_2 = 1 \\ & 3x_2 + z_3 = 2 \\ & x_i, z_i \geq 0 \end{aligned}$$

but there is no obvious b.f.s. since $z_1 = -1$, $z_2 = -1$, $z_3 = 2$ is not feasible.

The trick is to add extra variables called **artificial variables**, y_1, y_2 so that the constraints are

$$\begin{aligned} x_1 + x_2 - z_1 + y_1 &= 1 \\ 2x_1 - x_2 - z_2 + y_2 &= 1 \\ 3x_2 + z_3 &= 2 \\ x_i, z_i, y_i &\geq 0 \end{aligned}$$

Phase I is to minimize $y_1 + y_2$ and we can start this phase with $y_1 = 1$, $y_2 = 1$ and $z_3 = 2$. (Notice we did not need an artificial variable in the third equation.) Provided the original problem is feasible we should be able to obtain a minimum of 0 with $y_1 = y_2 = 0$ (since y_1 and y_2 are not needed to satisfy the constraints if the original problem is feasible). The point of this is that at the end of Phase I the simplex algorithm will have found a b.f.s. for the original problem. Phase II is then to proceed with the solution of the original problem, starting from this b.f.s.

Note: the original objective function doesn't enter into Phase I, but it is useful to carry it along as an extra row in the tableau since the algorithm will then arrange for it to be in the appropriate form to start Phase II.

Note also: the Phase I objective must be written in terms of the non-basic variables to begin Phase I. This can also be accomplished in the tableau. We start with

	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
Phase II	-6	-3	0	0	0	0	0	0
Phase I	0	0	0	0	0	-1	-1	0

Preliminary step. Add rows 1 and 2 to the Phase I objective so that it is written in terms of non-basic variables.

	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
Phase II	-6	-3	0	0	0	0	0	0
Phase I	3	0	-1	-1	0	0	0	2

Begin Phase I.

Pivot on a_{21} to get

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

Pivot on a_{14} to get

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

End of Phase I. $y_1 = y_2 = 0$ and we no longer need these variables (so we drop the last two columns and Phase I objective row.) But we do have a b.f.s. to start Phase II with ($x_1 = 1$, $z_2 = 1$, $z_3 = 2$) and the rest of the tableau is already in appropriate form. So we rewrite the last tableau without the y_1, y_2 columns.

Begin Phase II.

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

In one more step we reach the optimum, by pivoting on a_{12}.

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

Notice that in fact, the problem we have considered is the same as the problem D, except that x replaces λ and we have added a constraint $2x_2 \leq 3$ (which is not tight at the optimum). It is interesting to compare the final tableau with the final tableau for problem P (shown again below).

In general, artificial variables are needed when there are constraints like

$$\leq -1, \text{ or } \geq 1, \text{ or } = 1,$$

unless constraints happen to be of a special form where it is easy to spot a b.f.s.

If the Phase I objective does not reach zero then the original problem is infeasible.

7 Algebra of Linear Programming

7.1 Sensitivity: shadow prices

Each row of each tableau merely consists of sums of multiples of rows of the original tableau. The objective row = original objective row + scalar multiples of other rows.

Consider the initial and final tableau for problem P.

initial

1	2	1	0	6
1	−1	0	1	3
1	1	0	0	0

final

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

In particular, look at the columns 3 and 4, corresponding to variables z_1 and z_2. We can see that

Final row (1) = $\frac{1}{3}$ initial row (1) $-\frac{1}{3}$ initial row (2)
Final row (2) = $\frac{1}{3}$ initial row (1) $+\frac{2}{3}$ initial row (2)
Final objective row = initial objective row $-\frac{2}{3}$ initial row (1) $-\frac{1}{3}$ initial row (2).

In particular, suppose we want to make a small change in b, so we replace $\begin{pmatrix} 6 \\ 3 \end{pmatrix}$ by $\begin{pmatrix} 6+\epsilon_1 \\ 3+\epsilon_2 \end{pmatrix}$. Providing ϵ_1, ϵ_2 are small enough they will not affect the sequence of simplex operations. Thus if the constraints move just a little the optimum will still occur with the same variables in the basis. The argument above indicates that the final tableau will be

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	$1+\frac{1}{3}\epsilon_1-\frac{1}{3}\epsilon_2$
1	0	$\frac{1}{3}$	$\frac{2}{3}$	$4+\frac{1}{3}\epsilon_1+\frac{2}{3}\epsilon_2$
0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	$-5-\frac{2}{3}\epsilon_1-\frac{1}{3}\epsilon_2$

with corresponding solution $x_1 = 4+\frac{1}{3}\epsilon_1-\frac{1}{3}\epsilon_2$ and $x_2 = 1+\frac{1}{3}\epsilon_1+\frac{2}{3}\epsilon_2$ and objective function value $5+\frac{2}{3}\epsilon_1+\frac{1}{3}\epsilon_2$. If ϵ_1, ϵ_2 are such that we have $x_1 < 0$ or $x_2 < 0$ then vertex C is no longer optimal.

The objective function row of the final tableau shows the **sensitivity** of the optimal solution to changes in b and how the optimum value varies with small changes in b. For this reason the values in the objective row are sometimes known as **shadow prices**. The idea, in the above example, is that we would be willing to a pay price of $\frac{2}{3}\epsilon_1$ for relaxation of the right hand side of the first constraint from 6 to $6+\epsilon_1$.

Notice also that being able to see how the final tableau is related to the initial one without looking at the intermediate steps provides a useful way of checking your arithmetic if you suspect you have got something wrong!

Notice that for problem P the objective rows at the vertices A, B, C and D are:

A	1	1	0	0	0
B	0	2	0	−1	−3
C	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	−5
D	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	−3

Compare these values with the basic solutions of the dual problem (on page 15). You will see that the objective row of the simplex tableau corresponding to each b.f.s. of problem P contains the values of the variables for a complementary slack basic solution to problem D (after a sign change).

The simplex algorithm can (and should) be viewed as searching amongst basic feasible solutions of P, for a complementary-slack basic solution of D which is also feasible.

At the optimum of P the shadow prices (which we can read off in the bottom row) are also the dual variables for the optimal solution of D.

7.2 Algebra of the simplex method

It is convenient to divide the variables into two sets, and to split the matrix A accordingly. For example, given

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

we can partition the variables into two disjoint subsets (basic and non-basic) $B = \{1,2\}$ and $N = \{3\}$ and rewrite the equation

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix} x_3 = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

or

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

where $x_B = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ contains the variables in B and $x_N = \begin{pmatrix} x_3 \end{pmatrix}$ contains the variables in N, and A_B has the columns of A corresponding to variables in B (columns 1 and 2) and A_N has columns corresponding to variables from N (column 3).

You should convince yourself that the two forms of the linear equations are equivalent and that the same trick would work for general $m \times n$ matrices A and partition of variables into two sets. If $A = (a_1, \ldots, a_n)$, where a_i is the ith column, then

$$Ax = \sum_{i \in B} a_i x_i + \sum_{i \in N} a_i x_i = A_B x_B + A_N x_N = b.$$

We usually choose B to have m elements (a basis) and N to have $n-m$ elements. Then setting $x_N = 0$, we solve $A_B x_B = b$ where A_B is an $m \times m$ matrix to find the basic solution $x_B = A_B^{-1} b$, $x_N = 0$.

Let us take problem P in the form

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

Given a choice of basis B we can rewrite the problem as above

$$\max \{c_B^\top x_B + c_N^\top x_N\} \quad \text{s.t. } A_B x_B + A_N x_N = b,\ x_B, x_N \geq 0.$$

At this stage we are just rewriting the equations in terms of the two sets of variables x_B and x_N. The equations hold for any feasible x. Now A_B is invertible by our non-degeneracy assumptions in Assumption 4.1. Thus we can write

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N, \tag{1}$$

and

$$\begin{aligned} f &= c_B^\top x_B + c_N^\top x_N \\ &= c_B^\top (A_B^{-1} b - A_B^{-1} A_N x_N) + c_N^\top x_N \\ &= c_B^\top A_B^{-1} b + (c_N^\top - c_B^\top A_B^{-1} A_N) x_N \end{aligned} \tag{2}$$

Equation (1) gives the constraints in a form where x_B appears with an identity matrix and (2) gives the objective function in terms of the non-basic variables. Thus the tableau corresponding to basis B will contain (after appropriate rearrangement of rows and columns)

basic	non-basic	
I	$A_B^{-1} A_N$	$A_B^{-1} b$
0	$c_N^\top - c_B^\top A_B^{-1} A_N$	$-c_B^\top A_B^{-1} b$

Thus, given a basis (choice of variables) we can work out the appropriate tableau by inverting A_B. Note that for many choices of B we will find that $A_B^{-1} b$ has negative components and so the basic solution $x_B = A_B^{-1} b$ is not feasible; we need the simplex algorithm to tell which Bs to look at.

In programming a computer to implement the simplex algorithm you need only remember what your current basis is, since the whole tableau can then be computed from A_B^{-1}. The **revised simplex algorithm** works this way and employs various tricks to compute the inverse of the new A_B from the old A_B^{-1} (using the fact that only one variable enters and one leaves). This can be very much more efficient.

Now we know that the simplex algorithm terminates at an optimal feasible basis when the coefficients of the objective row are all ≤ 0. In other words, there is a basis B for which

$$c_N^\top - c_B^\top A_B^{-1} A_N \leq 0.$$

Recall the dual problem is

$$\text{D: } \min \lambda^\top b \quad \text{s.t. } A^\top \lambda \geq c.$$

Let us write $\lambda = (A_B^{-1})^\top c_B$. Then we have $A_B^\top \lambda = c_B$ and $A_N^\top \lambda \geq c_N$, and hence λ is feasible. Furthermore, $x_B = A_B^{-1} b$, $x_N = 0$ is a basic solution for P and complementary slackness is satisfied since

$$\begin{aligned} c_B - A_B^\top \lambda = 0 &\Longrightarrow (c_B - A_B^\top \lambda)^\top x_B = 0, \\ x_N = 0 &\Longrightarrow (c_N - A_N^\top \lambda)^\top x_N = 0. \end{aligned}$$

Consequently, $x_B = A_B^{-1} b$, $x_N = 0$ and $\lambda = (A^{-1})^\top c_B$ are respectively optimal for the primal and dual. We also have that with these solutions

$$f = c_B^\top x_B = c_B^\top A_B^{-1} b = \lambda^\top b.$$

So we have a proof of Theorem 4.3, that the primal and dual have the same objective value (if we accept that the simplex algorithm terminates at an optimum with ≤ 0 objective row) for the case of LP problems.

Remark

We have shown that in general the objective row of the final (optimal) tableau will contain $c_N^\top - \lambda^\top A_N$ in the non-basic columns, where λ are dual variables. This is consistent with our observation that the final tableau the vector $-\lambda$ sits in the bottom row, of the columns corresponding to the slack variables. We start with a primal $Ax \leq b$ and add slack variables z. In this case the the objective function is $c^\top x + 0^\top z$, so $c_i = 0$ in columns corresponding to slack variables, and columns of A_N which correspond to the slack variables are the columns of an identity matrix (since $Ax + Iz = b$). So the part of the objective row beneath the original slack variables will contain $0^\top - \lambda^\top I_N = -\lambda^\top$, and λ are the dual variables corresponding to the primal constraints. The rest of the objective row, beneath the original x, contains $c_B^\top - \lambda^\top A_B$, i.e., the values of the slack variables in the dual problem. This is what we observed in our earlier example.

8 Shadow Prices and Lagrangian Necessity

8.1 Shadow prices

Lagrangian multipliers, dual variables and **shadow prices** are the same things. Let us say a bit more about the latter.

Suppose you require an amounts $b_1, \dots, b_m$ of m different vitamins. There are n foodstuffs available. Let

$$a_{ij} = \text{amounts of vitamin } i \text{ in one unit of foodstuff } j,$$

and suppose foodstuff j costs c_j per unit. Your problem is therefore to choose the amount x_j of foodstuff j you buy to solve the LP

$$\begin{aligned} \min \sum_j c_j x_j \\ \text{subject to } \sum_j a_{ij} x_j \geq b_i, \text{ each } i \\ \text{and } x_j \geq 0 \text{ each } j. \end{aligned}$$

Now suppose that a vitamin company decides to market m different vitamin pills (one for each vitamin) and sell them at price p_i per unit for vitamin i. Assuming you are prepared to switch entirely to a diet of vitamin pills, but that you are not prepared to pay more for an artificial carrot (vitamin equivalent) than a real one, the company has to maximize profit by choosing prices p_i to

$$\begin{aligned} \max \sum_i b_i p_i \\ \text{subject to } \sum_i a_{ij} p_i \leq c_j, \text{ each } j \\ \text{and } p_i \geq 0 \text{ each } i. \end{aligned}$$

Note that this is the LP which is dual to your problem. The dual variable p_i is the price you are prepared to pay for a unit of vitamin i and is called a shadow price. By extension, dual variables are sometimes called shadow prices in problems where their interpretation as prices is very hard (or impossible) to see.

The dual variables tell us how the optimum value of our problem changes with changes in the right-hand side (b) of our functional constraints. This makes sense in the example given above. If you require an amount $b_i + \epsilon$ of vitamin i instead of an amount b_i you would expect the total cost of your foodstuff to change by an amount ϵp_i, where p_i is the value to you of a unit of vitamin i, even though in your problem you cannot buy vitamin i separately from the others.

The above makes clear the relationship between dual variables/Langrange multipliers and shadow prices in the case of linear programming.

More generally, in linear problems we can use what we know about the optimal solutions to see how this works. Let us assume the primal problem

$$\begin{aligned} P(b): \qquad & \min c^\top x \\ \text{s.t. } & Ax - z = b, \ x, z \geq 0. \end{aligned}$$

has the optimal solution $\phi(b)$, depending on b. Consider two close together values of b, say b' and b'', and suppose that optimal solutions have the same basic variables (so optimums occur with the same variables being non-zero, though the values of the variables will change slightly). The optimum still occurs at the same vertex of the feasible region though it moves slightly. Now consider the dual problem. This is

$$\begin{aligned} & \max \lambda^\top b \\ \text{s.t. } & A^\top \lambda \leq c, \ \lambda \geq 0. \end{aligned}$$

In the dual problem the feasible set does not depend on b, so the optimum of the dual will occur with the same basic variables and the same values of the dual variables λ. But the value of the optimum dual objective function is $\lambda^\top b'$ in one case and $\lambda^\top b''$ in the other and we have seen that the primal and dual have the same solutions. Hence

$$\phi(b') = \lambda^\top b' \text{ and } \phi(b'') = \lambda^\top b''$$

and the values of the dual variables λ give the rate of change of the objective value with b. The change is linear in this case.

The same idea works in nonlinear problems.

Example 8.1. Recall Example 2.1, where we had constraints

$$\begin{aligned} x_1 + x_2 + x_3 = 5 \\ x_1^2 + x_2^2 = 4 \end{aligned}$$

and obtained values of Lagrange multipliers of $\lambda_1^* = -2, \lambda_2^* = -\sqrt{5/8}$.

If we replace the constraints by

$$\begin{aligned} x_1 + x_2 + x_3 = b_1 \\ x_1^2 + x_2^2 = b_2 \end{aligned}$$

and write $\phi(b)$ = optimal value of the problem with $b = (b_1, b_2)^\top$ then you can check that

$$\left.\frac{\partial \phi}{\partial b_1}\right|_{b=(5,4)} = -2 \qquad \left.\frac{\partial \phi}{\partial b_2}\right|_{b=(5,4)} = -\sqrt{5/8}.$$

Let $P(b)$ be the problem: maximize $f(x) : g(x) \le b,\ x \in \mathbb{R}^n$. Let $\phi(b)$ be its optimal value.

Theorem 8.1. *Suppose f and g are continuously differentiable on $X = \mathbb{R}^n$, and that for each b there exist unique*

- *$x^*(b)$ optimal for $P(b)$, and*
- *$\lambda^*(b) \in \mathbb{R}^m$, $\lambda^*(b) \ge 0$ such that $\phi(b) = \sup_{x\in X}\{f(x) + \lambda^*(b)^\top(b - g(x))\}$.*

If x^* and λ^* are continuously differentiable, then

$$\frac{\partial \phi(b)}{\partial b_i} = \lambda_i^*(b). \tag{3}$$

Proof.

$$\phi(b) = L(x^*, \lambda^*) = f(x^*) + \lambda^*(b)^\top (b - g(x^*))$$

Since $L(x^*, \lambda^*)$ is stationary with respect to x_j^*, we have for each j,

$$\frac{\partial L(x^*, \lambda^*)}{\partial x_j^*} = 0\,.$$

For each k we have either $g_k(x^*) = b_k$, or $g_k(x^*) < b_k$. Since $\lambda^*(b)^\top(b - g(x^*)) = 0$ we have in the later case, $\lambda_k^* = 0$, and so $\partial\lambda_k^*/\partial b_i = 0$. So

$$\frac{\partial \phi(b)}{\partial b_i} = \frac{\partial L(x^*, \lambda^*)}{\partial b_i} + \sum_{j=1}^{n} \frac{\partial L(x^*, \lambda^*)}{\partial x_j^*}\frac{\partial x_j^*}{\partial b_i}.$$

On the r.h.s. above, the second term is 0 and the first term is

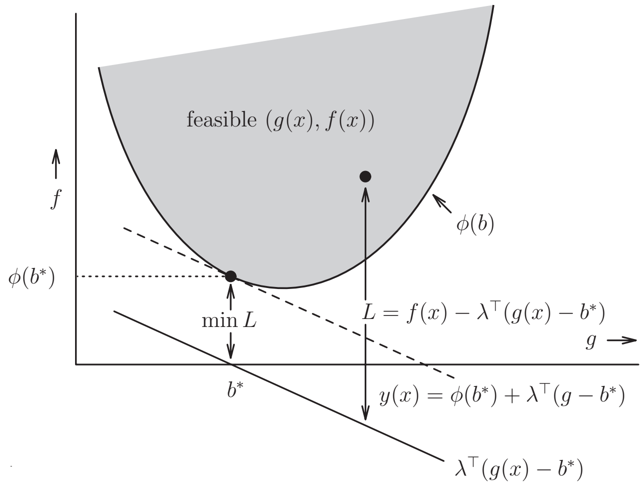
$$\frac{\partial L(x^*, \lambda^*)}{\partial b_i} = \lambda_i^*(b) + \sum_{k=1}^{m} + \frac{\partial \lambda_k^*(b)}{\partial b_i}\Big[b_k - g_k(x^*(b))\Big].$$

Now the second term on the r.h.s. above is 0, and so we have (3). □

8.2 Lagrangian necessity

In the examples we have studied we have been able to find Lagrange multipliers λ that work in the Lagrangian Sufficiency Theorem. We have also observed that the primal and dual problems have the same optimum values. It is outside the scope of the course to establish conditions under which we expect these results to hold, but we can give a brief summary.

Let $P(b)$ be the problem: minimize $f(x)$ s.t. $g(x) = b$ and $x \in X$. Let $\phi(b)$ be its optimal value. Suppose that $\phi(b)$ is convex in b, as shown in the figure. The convexity of ϕ implies that for each b^* there is a tangent hyperplane to $\phi(b)$ at b^* with the graph



of $\phi(b)$ lying entirely above it. This uses the **supporting hyperplane theorem** for convex sets, which is geometrically obvious, but some work to prove. In the (g, f) plane, the equation of this tanget hyperplane, through the point $(b^*, \phi(b^*))$ can be written $y(x) = \phi(b^*) + \lambda^\top(g(x) - b^*)$ for some λ. So $f(x) - y(x)$ is minimized to 0 (over all $x \in X$) by taking x such that $(g(x), f(x)) = (b^*, \phi(b^*))$. Equivalently, $L(x, \lambda) = f(x) - \lambda^\top(g(x) - b^*)$ is minimized to $\phi(b^*)$.

Compare, for example, two problems: (i) minimize x^2 s.t. $x = b$, and (ii) minimize x^2 s.t. $x^4 = b$. In (i) we have $\phi(b) = b^2$ and $L(x, \lambda) = x^2 - \lambda(x - b)$ is minimized at $x = b$ when we take $\lambda = 2b$, whereas in (ii) we have $\phi(b) = b^{1/2}$ and there is no λ such that $L(x, \lambda) = x^2 - \lambda(x^4 - b)$ is minimized at $x = b$, (since for $\lambda > 0$ the minimum is at $x = \infty$, and for $\lambda \le 0$ the minimum is at $x = 0$.)

The following theorem gives simple conditions under which $\phi(b)$ is convex in b.

Theorem 8.2 (**Sufficient conditions for Lagrangian methods to work**). *Let $P(b)$ be the problem: minimize $f(x)$ s.t. $g(x) \le b$ and $x \in X$. If the functions f, g are convex, X is convex and x^* is an optimal solution to P, then there exist Lagrange multipliers $\lambda \in \mathbb{R}^m$ such that $L(x^*, \lambda) \le L(x, \lambda)$ for all $x \in X$.*

In particular, Lagrange multipliers always exist in linear programming programs, provided they have optimal solutions (i.e., are feasible and bounded).

Assuming the supporting hyperplane theorem, the proof of Theorem 8.2 relies on showing that $\phi(b)$ is convex. To see this, suppose that x_i is optimal for $P(b_i)$, $i = 1, 2$. Let $x = \theta x_1 + (1 - \theta)x_2$, and $b = \theta b_1 + (1 - \theta)b_2$. Convexity of X implies $x \in X$, and convexity of g implies $g(x) \le b$, so x is feasible for $P(b)$. So $\phi(b) \le f(x) \le \theta f(x_1) + (1 - \theta)f(x_2) = \theta\phi(b_1) + (1 - \theta)\phi(b_2)$, where the second inequality follows from convexity of f. Thus $\phi(b)$ is convex in b.

9 Two Person Zero-Sum Games

9.1 Games with a saddle-point

We consider games that are **zero-sum**, in the sense that one player wins what the other loses. The players make moves simultaneously. Each has a choice of moves (not necessarily the same). If player I makes move i and player II makes move j then player I wins (and player II loses) a_{ij}. Both players know the $m \times n$ **pay-off matrix** $A = (a_{ij})$.

$$\begin{array}{cc} & \text{II plays } j \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{I plays } i \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{array}{|cccc|} \hline -5 & 3 & 1 & 20 \\ 5 & 5 & 4 & 6 \\ -4 & 6 & 0 & -5 \\ \hline \end{array} \leftarrow (a_{ij}) \end{array}$$

Let us ask what is the best that player I can do if player II plays move j.

II's move: $j =$	1	2	3	4	
I's best response: $i =$	2	3	2	1	
I wins	5	6	**4**	20	$\leftarrow$ column maximums

Similarly, we ask what is the best that player II can do if I plays move i?

I's move: $i =$	1	2	3	
II's best response: $j =$	1	3	4	
I wins	-5	**4**	-5	$\leftarrow$ row minimums

Here the minimal column maximum $= \min_j \max_i a_{ij} = \max_i \min_j a_{ij} =$ maximal row minimum $= 4$, when player I plays 2 and player II plays 3. In this case we say that A has a **saddle-point** $(2, 3)$ and the game is solved.

Remarks. The game is solved by 'I plays 2' and 'II plays 3' in the sense that

1. Each player maximizes his minimum gain.
2. If either player announces any strategy (in advance) other than 'I plays 2' and 'II plays 3', he will do worse.
3. If either player announces that he will play the saddle-point move in advance, the other player cannot improve on the saddle-point.

9.2 Example: Two-finger Morra, a game without a saddle-point

Morra is a hand game dating from Roman and Greek times. Each player displays either one or two fingers and simultaneously guesses how many fingers his opponent will show. If both players guess correctly or both guess incorrectly then the game is a tie. If only one player guesses correctly, then that player wins from the other player an amount equal to the total number of fingers shown. A strategy for a player is $(a, b) =$'show a, guess b'. The pay-off matrix is

$$\begin{array}{c} \\ (1,1) \\ (1,2) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccc} (1,1) & (1,2) & (2,1) & (2,2) \end{array} \\ \begin{array}{|cccc|} \hline 0 & 2 & -3 & 0 \\ -2 & 0 & 0 & 3 \\ 3 & 0 & 0 & -4 \\ 0 & -3 & 4 & 0 \\ \hline \end{array} \end{array} = (a_{ij})$$

Column maximums are all positive and row minimums are all negative. So there is no saddle point (even though the game is symmetric and fair). If either player announces a fixed strategy (in advance), the other player will win.

We must look for a solution to the game in terms of mixed strategies.

9.3 Determination of an optimal strategy

Each player must use a **mixed strategy**. Player I plays move i with probability p_i, $i = 1, \ldots, m$ and player II plays moves j with probability q_j, $j = 1, \ldots, n$. Player I's expected payoff if player II plays move j is

$$\sum_i p_i a_{ij}.$$

So player I attempts to

$$\text{maximize} \left\{ \min_j \sum_i p_i a_{ij} \right\} \text{ s.t. } \sum_i p_i = 1,\ p_i \geq 0.$$

Note that this is equivalent to

$$\text{P: } \max v \quad \text{s.t. } \sum_i a_{ij} p_i \geq v, \text{ each } j, \text{ and } \sum_i p_i = 1,\ p_i \geq 0,$$

since v on being maximized will increase until it equals the minimum of the $\sum_i a_{ij} p_i$. By similar arguments, player II's problem is

$$\text{D: } \min v \quad \text{s.t. } \sum_j a_{ij} q_j \leq v, \text{ each } i, \text{ and } \sum_j q_j = 1,\ q_j \geq 0.$$

It is possible to show that P and D are duals to one another (by the standard technique of finding the dual of P). Consequently, the general theory gives sufficient conditions for strategies p and q to be optimal.

Let e denote a vector of 1s, the number of components determined by context.

Theorem 9.1. *Suppose $p \in \mathbb{R}^m$, $q \in \mathbb{R}^n$, and $v \in \mathbb{R}$, such that*

(a) $p \geq 0$, $e^\top p = 1$, $p^\top A \geq ve^\top$ (primal feasibility);

(b) $q \geq 0$, $e^\top q = 1$, $Aq \leq ve$ (dual feasibility);

(c) $v = p^\top Aq$ (complementary slackness).

Then p is optimal for P and q is optimal for D with common optimum (the **value of the game***) v.*

Proof. The fact that p and q are optimal solutions to linear programs P and D follows from Theorem 3.4. Alternatively, note that Player I can guarantee to get at least

$$\min_q p^\top Aq \geq \min_q (ve^\top)q = v,$$

and Player II can guarantee that Player I gets no more than

$$\max_p p^\top Aq \leq \max_p p^\top(ve) = w = v.$$

In fact, (c) is redundant; it is implied by (a) and (b). □

Remarks.

1. Notice that this gives the right answer for a game with a saddle-point (i^*, j^*), (i.e., $v = a_{i^*j^*}$, with $p_{i^*} = q_{j^*} = 1$ and other $p_i, q_j = 0$).

2. Two-finger Morra has an optimal solution $p = q = (0, \frac{3}{5}, \frac{2}{5}, 0)$, $v = 0$, as can be easily checked. E.g. $p^\top A = (0, 0, 0, 1/5) \geq 0 \times 1^\top$. It is obvious that we expect to have $p = q$ and $v = 0$ since the game is symmetric between the players ($A = -A^\top$). A is called an **anti-symmetric matrix**.

   The optimal strategy is not unique. Another optimal solution is $p = q = (0, \frac{4}{7}, \frac{3}{7}, 0)$. Player I can play any mixed strategy of the form $(0, \theta, 1-\theta, 0)$ provided $\frac{4}{7} \leq \theta \leq \frac{3}{5}$.

3. These conditions allow us to check optimality. For small problems one can often use them to find the optimal strategies, but for larger problems it will be best to use some other technique to find the optimum (e.g., simplex algorithm). Note, however, that the problems P and D are not in a form where we can apply the simplex algorithm directly;v does not have a positivity constraint. Also the constraints are $\sum_i a_{ij}p_i - v = 0$ with r.h.s.= 0. It is possible, however, to transform the problem into a form amenable to the simplex algorithm.

   (a) Add a constant k to each a_{ij} so that $a_{ij} > 0$ each i, j. This doesn't change anything, except the value which is now guaranteed to be positive ($v > 0$).

   (b) Change variables to $x_i = p_i/v$. We now have that P is

$$\max v \quad \text{s.t.} \sum_i a_{ij}x_i \geq 1, \; \sum_i x_i = 1/v, \; x_i \geq 0,$$

   which is equivalent to

$$\min \sum_i x_i \quad \text{s.t.} \sum_i a_{ij}x_i \geq 1, \; x_i \geq 0$$

   and this is the type of LP that we are used to.

9.4 Example: Colonel Blotto

Colonel Blotto has three regiments and his enemy has two regiments. Both commanders are to divide their regiments between two posts. At each post the commander with the greater number of regiments wins one for each conquered regiment, plus one for the post. If the commanders allocate equal numbers of regiments to a post there is a stand-off. This gives the pay-off matrix

		Enemy commander		
		(2,0)	(1,1)	(0,2)
Colonel Blotto	(3,0)	3	1	0
	(2,1)	1	2	−1
	(1,2)	−1	2	1
	(0,3)	0	1	3

Clearly it is optimal for Colonel Blotto to divide his regiments (i, j) and (j, i) with equal probability. So the game reduces to one with the payoff matrix

	(2,0)	(1,1)	(0,2)
(3,0) or (0,3)	$\frac{3}{2}$	1	$\frac{3}{2}$
(2,1) or (1,2)	0	2	0

To derive the optimal solution we can

(a) look at player Colonel Blotto's original problem: maximize $\{\min_j \sum_i p_i a_{ij}\}$, i.e., maximize$_p \min\{\frac{3}{2}p, p + 2(1-p)\}$,

(b) attempt to derive p, q, v from the conditions of Theorem 9.1, or

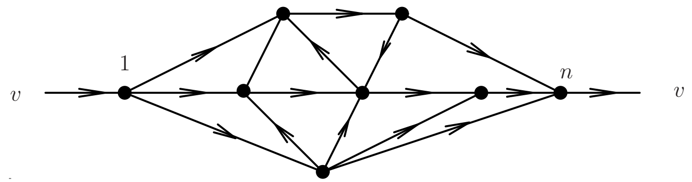
(c) convert the problem as explained above and use the simplex method.

For this game, $p = (\frac{4}{5}, \frac{1}{5})$, $q = (\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$ and $v = \frac{6}{5}$ is optimal. In the the original problem, this means that Colonel Blotto should distribute his regiments as (3,0), (2,1), (1,2), (0,3) with probabilities $\frac{2}{5}, \frac{1}{10}, \frac{1}{10}, \frac{2}{5}$ respectively, and his enemy should distribute hers as (2,0), (1,1), (0,2) with probabilities $\frac{1}{5}, \frac{3}{5}, \frac{1}{5}$ respectively.

10 Maximal Flow in a Network

10.1 Max-flow/min-cut theory

Consider a network consisting of n nodes labelled $1, \ldots, n$ and directed edges between them with capacities c_{ij} on the arc from node i to node j. Let x_{ij} denote the flow in the arc $i \to j$, where $0 \le x_{ij} \le c_{ij}$.



Problem: Find maximal flow from node 1 (the **source**) to node n (the **sink**) subject to the conservation of flow at nodes, i.e.,

$$\text{maximize } v \quad \text{s.t. } 0 \le x_{ij} \le c_{ij}, \text{ for all } i, j$$

and

$$\boxed{\begin{matrix}\text{flow out of}\\ \text{node } i\end{matrix}} - \boxed{\begin{matrix}\text{flow into}\\ \text{node } i\end{matrix}} = \sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & \text{if } i = 1 \\ 0 & \text{if } i = 2, \ldots, n-1 \\ -v & \text{if } i = n \end{cases}$$

where the summations are understood to be over existing arcs only. v is known as the **value of the flow**.

This is obviously an LP problem, but with lots of variables and constraints. We can solve it more quickly (taking advantage of the special network structure) as follows.

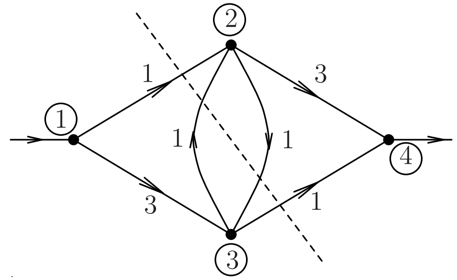
Definition 10.1. *A* **cut** $(S, \bar{S})$ *is a partition of the nodes into two disjoint subsets* S *and* $\bar{S}$ *with* $1 \in S$ *and* $n \in \bar{S}$.

Definition 10.2. *The* **capacity** *of a cut*

$$C(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} c_{ij}.$$

Thus given a cut $(S, \bar{S})$ the capacity of the cut is the maximal flow from nodes in S to nodes in $\bar{S}$. It is intuitively clear that any flow from node 1 to node n must cross the cut $(S, \bar{S})$, since in getting from 1 to n at some stage it must cross from S to $\bar{S}$. This holds for any flow and any cut.

Example 10.1.



Cut $S = \{1, 3\}$, $\bar{S} = \{2, 4\}$, $C(S, \bar{S}) = 3$. Check that the maximal flow is 3.

In fact, we have:

Theorem 10.1 (Max flow/min cut Theorem)**.** *The maximal flow value through the network is equal to the minimal cut capacity.*

Proof. Summing the feasibility constraint

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & \text{if } i = 1 \\ 0 & \text{if } i = 2, \ldots, n-1 \\ -v & \text{if } i = n \end{cases}$$

over $i \in S$, yields

$$\begin{aligned} v &= \sum_{i \in S, j \in N} x_{ij} - \sum_{j \in N, i \in S} x_{ji} \\ &= \sum_{i \in S, j \in \bar{S}} x_{ij} - \sum_{j \in \bar{S}, i \in S} x_{ji} \\ &\le C(S, \bar{S}) \end{aligned}$$

since for all i, j we have $0 \le x_{ij} \le c_{ij}$. Hence the value of any feasible flow is less than or equal to the capacity of any cut.

So any flow $\le$ any cut capacity, (and in particular max flow $\le$ min cut).

Now let f be a maximal flow, and define $S \subseteq N$ recursively as follows:

(1) $1 \in S$.
(2) If $i \in S$ and $x_{ij} < c_{ij}$, then $j \in S$.
(3) If $i \in S$ and $x_{ji} > 0$, then $j \in S$.
Keep applying (2) and (3) until no more can be added to S.

So S is the set of nodes to which we can increase flow. Now if $n \in S$ we can increase flow along a path in S and f is not maximal. So $n \in \bar{S} = N \setminus S$ and $(S, \bar{S})$ is a cut. From the definition of S we know that for $i \in S$ and $j \in \bar{S}$, $x_{ij} = c_{ij}$ and $x_{ji} = 0$, so in the formula above we get

$$v = \sum_{i \in S, j \in \bar{S}} x_{ij} - \sum_{j \in \bar{S}, i \in S} x_{ji} = C(S, \bar{S}).$$

So max flow = min cut capacity. □

Corollary 10.2. *If a flow value v = cut capacity C then v is maximal and C minimal.*

The proof suggests an algorithm for finding the maximal flow.

10.2 Ford-Fulkerson algorithm

1. Start with a feasible flow (e.g., $x_{ij} = 0$).

2. Construct S recursively by the algorithm defined in the box above.

3. If $n \in S$ then there is a path from 1 to n along which we can increase flow by

$$\epsilon = \min_{(ij)} \max[x_{ji}, c_{ij} - x_{ij}] > 0.$$

   where the minimum is taken with respect to all arcs $i \to j$ on the path.

   Replace the flow by this increased flow. Return to 2.

   If $n \notin S$ then the flow is optimal.

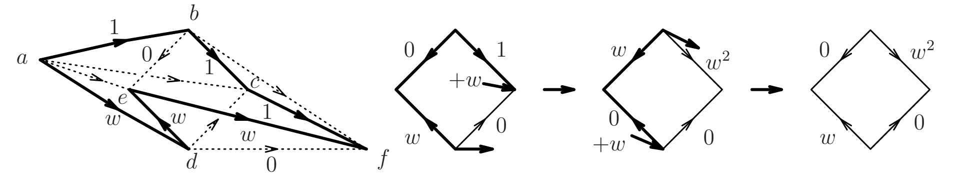
The algorithm is crude and simple; we just push flow through where we can, until we can't do so anymore. There is no guarantee that it will be very efficient. With hand examples it is usually easy to 'see' the maximal flow. You just demonstrate that it is maximal by giving a cut with the same capacity as the flow and appeal to the min cut = max flow theorem.

The algorithm can be made not to converge if the capacities are not rational multiples of one another. However,

Theorem 10.3. *If capacities and initial flows are rational then the algorithm terminates at a maximal flow in a finite number of steps. (Capacities are assumed to be finite.)*

Proof. Multiply by a constant so that all capacities and initial flows are integers. The algorithm increases flow by at least 1 on each step. □

Example: Failure to stop when capacities and initial flows are not rational



The network consists of a square $b\,c\,d\,e$ of directed arcs of capacity 1. The corners of the square are connected to a source at a and a sink at f by arcs of capacity 10. The initial flow of $1+w$ is shown in the first picture, where $w = (\sqrt{5}-1)/2$, so $1-w = w^2$. The first iteration is to increase flow by w along $a \to c \to b \to e \to d \to f$. The second increases it by w along $a \to d \to e \to b \to f$. The flow has increased by $2w$ and the resulting flow in the square is the same as at the start, but multiplied by w and rotated through 180^o. Hence the algorithm can continue in this manner forever without stopping and never reach the optimal flow of 40.

10.3 Minimal cost circulations

Definition 10.3. *A network is a* **closed network** *if there is no flow into or out of the network.*

Definition 10.4. *A flow in a closed network is a* **circulation** *if $\sum_j x_{ij} - \sum_j x_{ji} = 0$ for each node i.*

Most network problems can be formulated as the problem of finding a **minimal cost circulation** in a closed network where there are capacity constraints $c_{ij}^- \leq x_{ij} \leq c_{ij}^+$ on arcs (i,j) and a cost per unit flow of d_{ij} in arcs (i,j). The full problem is

$$\text{minimize} \sum_{ij} d_{ij} x_{ij}$$

$$\text{subject to} \sum_j x_{ij} - \sum_j x_{ji} = 0, \text{ each } i, \text{ and } c_{ij}^- \leq x_{ij} \leq c_{ij}^+.$$

Definition 10.5. *A circulation which satisfies the capacity constraints is called a* **feasible circulation**.

There is a beautiful algorithm called the out-of-kilter algorithm which will solve general problems of this kind. It does not even require a feasible solution with which to start. In the next lecture we shall just derive conditions for a flow to be optimal.

We shall also see, although it should be obvious already, that the max flow problem that is studied in this lecture can be formulated as a minimal cost circulation problem.

11 Minimum Cost Circulation Problems

11.1 Sufficient conditions for a minimal cost circulation

Recall the minimum cost circulation problem:

$$\text{minimize } \sum_{ij} d_{ij} x_{ij}$$

$$\text{subject to } \sum_j x_{ij} - \sum_j x_{ji} = 0, \text{ each } i, \text{ and } c_{ij}^- \le x_{ij} \le c_{ij}^+.$$

Consider the Lagrangian for the problem of finding the minimum cost circulation. We shall treat the capacity constraints as the region constraints, so

$$X = \{x_{ij} : c_{ij}^- \le x_{ij} \le c_{ij}^+\}.$$

We introduce Lagrange multipliers λ_i (one for each node) and write

$$L(x,\lambda) = \sum_{ij} d_{ij} x_{ij} - \sum_i \lambda_i \left(\sum_j x_{ij} - \sum_j x_{ji} \right)$$

Rearranging we obtain

$$L(x,\lambda) = \sum_{ij} (d_{ij} - \lambda_i + \lambda_j) x_{ij}.$$

We attempt to minimize $L(x,\lambda)$ in X.

Provided c_{ij}^-, c_{ij}^+ are finite we see that there is a finite minimum for all λ, achieved such that

$$x_{ij} = \begin{cases} c_{ij}^- & \text{if } \quad d_{ij} - \lambda_i + \lambda_j > 0 \\ c_{ij}^+ & \text{if } \quad d_{ij} - \lambda_i + \lambda_j < 0 \end{cases} \tag{1}$$

$$c_{ij}^- \le x_{ij} \le c_{ij}^+ \quad \text{if} \quad d_{ij} - \lambda_i + \lambda_j = 0. \tag{2}$$

Theorem 11.1. *If (x_{ij}) is a feasible circulation and there exists λ such that $(x_{ij}), \lambda$ satisfy conditions* (1) *and* (2) *above, then (x_{ij}) is a minimal cost circulation.*

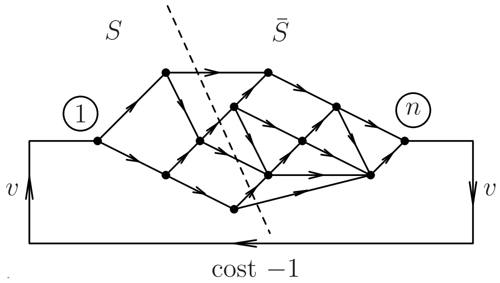
Proof. Apply the Lagrangian sufficiency theorem. □

Definition 11.1. *The Lagrange multipliers λ_i are usually known as* **node numbers** *or* **potentials** *in network problems.*

Definition 11.2. $\lambda_i - \lambda_j$ *is known as the* **tension** *in the arc (i,j).*

11.2 Max-flow as a minimum cost circulation problem

The maximal flow problem we studied earlier can be set up as a minimal cost circulation problem. For each arc in the network we assign a capacity constraint $0 \le x_{ij} \le c_{ij}^+$ and all cost $d_{ij} = 0$. Add an arc from node n to 1 with no capacity constraint and cost -1.



The cost of the circulation is $-v$, so minimizing $-v$ is the same as maximizing v. Let us seek node numbers λ_i which will satisfy optimality for this problem. Since arc $(n,1)$ has no capacity constraints, for a finite optimum we will require

$$d_{n1} - \lambda_n + \lambda_1 = 0 \Longrightarrow \lambda_1 = \lambda_n + 1.$$

Let us set $\lambda_n = 0$, $\lambda_1 = 1$. (Since it is only the differences in the λs that matter, we can pick one arbitrarily.) Let $(S, \bar{S})$ be a minimal cut. Assign $\lambda_i = 1$ for $i \in S$ and $\lambda_i = 0$ for $i \in \bar{S}$. Now check that

(a) For $i, j \in S$ or $i, j \in \bar{S} \Longrightarrow d_{ij} - \lambda_i + \lambda_j = 0$ so x_{ij} can take any feasible value.

(b) For $i \in S$, $j \in \bar{S}$ we have

$$d_{ij} - \lambda_i + \lambda_j = 0 - 1 + 0 = -1 \Longrightarrow x_{ij} = c_{ij}^+.$$

(c) For $i \in \bar{S}$, $j \in S$ we have

$$d_{ij} - \lambda_i + \lambda_j = 0 - 0 + 1 = 1 \Longrightarrow x_{ij} = 0.$$

But conditions (a)–(c) are precisely those satisfied by a maximal flow and minimal cut.

If we like, we can say that the Ford-Fulkerson algorithm in looking for a cut is trying to find node numbers and a flow to satisfy optimality conditions.

Remark

In many problems it is natural to take $c_{ij}^- = 0$, $c_{ij}^+ = \infty$. In this case we will achieve a finite optimum only if $d_{ij} - \lambda_i + \lambda_j \ge 0$ for each arc.

Theorem 11.2. *For a minimal cost circulation problem with capacity constraints $0 \le x_{ij} < \infty$ on each arc (i,j), if we have a feasible circulation (x_{ij}) and node numbers λ_i such that*

$$d_{ij} - \lambda_i + \lambda_j \ge 0, \text{ each } (i,j), \text{ and}$$
$$x_{ij} = 0 \text{ if } d_{ij} - \lambda_i + \lambda_j > 0,$$

then (x_{ij}) is optimal.

Proof. Apply the Lagrangian sufficiency theorem. □

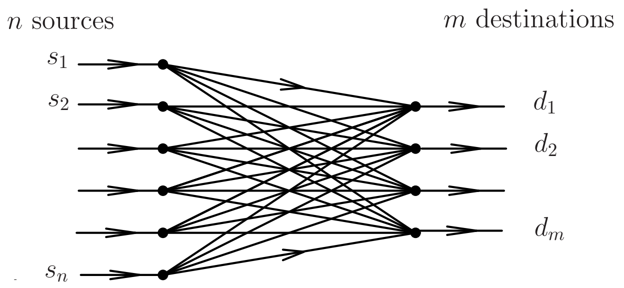
Note: The optimality conditions imply $(d_{ij} - \lambda_i + \lambda_j)x_{ij} = 0$ in this case (complementary slackness).

11.3 The transportation problem

Consider a network representing the problem of a supplier who has n supply depots from which goods must be shipped to m destinations. We assume there are quantities $s_1, \dots, s_n$ of the goods at depots $\{S_1, \dots, S_n\}$ and that the demands at destinations $\{D_1, \dots, D_m\}$ are given by $d_1, \dots, d_m$. We also assume that $\sum_i s_i = \sum_j d_j$ so that total supply = total demand. Any amount of goods may be taken directly from source i to destination j at a cost of d_{ij} $(i = 1, \dots, n; j = 1, \dots, m)$ per unit. One formulation of the problem is

$$\text{minimize} \sum_{ij} d_{ij} x_{ij}$$
$$\text{subject to} \sum_j x_{ij} = s_i \text{ each } i, \quad \sum_i x_{ij} = d_j \text{ each } j$$
$$\text{with } x_{ij} \ge 0 \text{ each } i, j.$$

Here x_{ij} is the flow from S_i to D_j. The network looks like:



with arcs (i,j), $0 \le x_{ij} < \infty$, and cost d_{ij} per unit flow.

The Lagrangian for the problem can be written

$$L(x, \lambda, \mu) = \sum_{ij} d_{ij} x_{ij} - \sum_i \lambda_i \left(\sum_j x_{ij} - s_i \right) + \sum_j \mu_j \left(\sum_i x_{ij} - d_j \right),$$

where we label Lagrange multipliers (node numbers) λ_i for sources and μ_j for destinations. (We choose the sign of μ_j in this apparently unusual way since it is convenient to think of the demands d_j as being negative supplies. In Section 12.2, we describe he simplex-on-a-graph algorithm, for a problem in which we suppose that there is a supply b_i at each node i.) Rearranging,

$$L(x, \lambda, \mu) = \sum_{ij} (d_{ij} - \lambda_i + \mu_j) x_{ij} + \sum_i \lambda_i s_i - \sum_j \mu_j d_j.$$

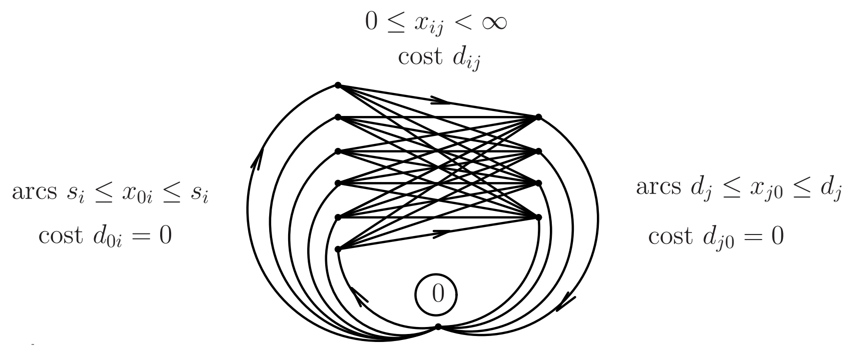
This will have a finite minimum in $x_{ij} \ge 0$, and the minimum occurs with $(d_{ij} - \lambda_i + \mu_j)x_{ij} = 0$ on each arc. Thus the Lagrangian sufficiency theorem give the same optimality conditions as before.

Theorem 11.3. *A flow x_{ij} is optimal for the transportation problem if $\exists\ \lambda_i, \mu_j$ such that $d_{ij} - \lambda_i + \mu_j \ge 0$ each (i,j) and $(d_{ij} - \lambda_i + \mu_j)x_{ij} = 0$.*

Proof. The Lagrangian sufficiency theorem applies. □

Remark

It is no surprise that the same optimality conditions appear as in the minimal cost circulation problem. If we augment the transportation network by connecting all sources and all destinations to a common 'artificial node' by arcs where the flow is constrained to be exactly that which is required (and zero cost) we obtain the same problem as in minimal cost circulation form.



The optimality conditions on the extra arcs are automatically satisfied by a feasible flow since $x_{j0} = d_j$, $x_{0i} = s_i$ regardless of node numbers.

12 Transportation and Transshipment Problems

12.1 The transportation algorithm

1. Set out the supplies and costs in a table as below

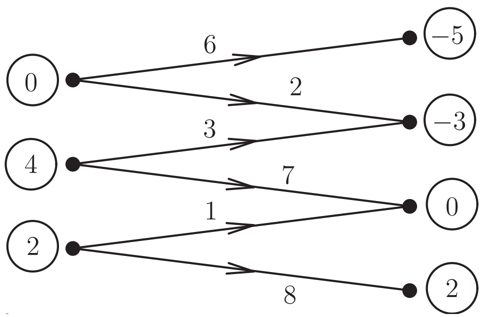
	D_1	D_2	D_3	D_4	
S_1	[5]	[3]	[4]	[6]	8
S_2	[2]	[7]	[4]	[1]	10
S_3	[5]	[6]	[2]	[4]	9
	6	5	8	8	

2. Allocate an initial feasible flow (by North-West corner rule or any other sensible method). NW corner rule says start at top left corner and dispose of supplies and fulfill demands in order i, j increasing. In our case we get

6 [5]	2 [3]	[4]	[6]
[2]	3 [7]	7 [4]	[1]
[5]	[6]	1 [2]	8 [4]
6	5	8	8

In the absence of degeneracy (which we assume) there are not less than $(m+n-1)$ non-zero entries, which appear in a 'stair-case' arrangement.

Remark. In our network picture we have constructed a feasible flow on a **spanning tree** of $m + n - 1$ arcs connecting n sources and m destinations.



A set of undirected arcs is spanning if it connects all nodes. It is a **tree** if it contains no circuits. A spanning tree is the equivalent of a basic solution for this problem.

3. For optimality we require $d_{ij} - \lambda_i + \mu_j = 0$ on any arc with non-zero flow. Set $\lambda_1 = 0$ (arbitrarily) and then compute the remaining λ_i, μ_j by using $d_{ij} - \lambda_i + \mu_j = 0$ on arcs for which $x_{ij} > 0$. On the table we have

$\lambda_i \setminus \mu_j$	-5	-3	0	-2
0	6 [5]	2 [3]	[4]	[6]
4	[2]	3 [7]	7 [4]	[1]
2	[5]	[6]	1 [2]	8 [4]

The node numbers are also shown on the network version above. With non-zero flows forming a spanning tree we will always be able to compute uniquely all node numbers given one of them.

4. We now compute $\lambda_i - \mu_j$ for all the remaining boxes (arcs) and write these elsewhere in the boxes. E.g.,

$\lambda_i \setminus \mu_j$	-5	-3	0	-2
0	6 [5]	2 [3]	*0* [4]	*2* [6]
4	*9* [2]	3 [7]	7 [4]	*6* [1]
2	*7* [5]	*5* [6]	1 [2]	8 [4]

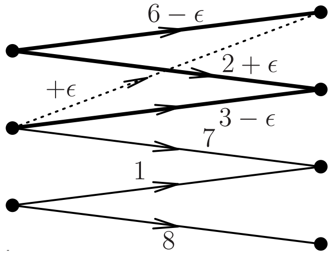
5. If all $d_{ij} \geq \lambda_i - \mu_j$, then the flow is optimal. Stop.

6. If not, (e.g., $i = 2$, $j = 1$, where $\lambda_2 - \mu_1 = 9 > d_{21} = 2$) we attempt to increase the flow in arc (i, j) for some (i, j) such that $\lambda_i - \mu_j > d_{ij}$. We seek an adjustment of $+\epsilon$ to the flow in arc (i, j) which keeps the solution feasible (and therefore preserves total supplies and demands). In our case we do this by

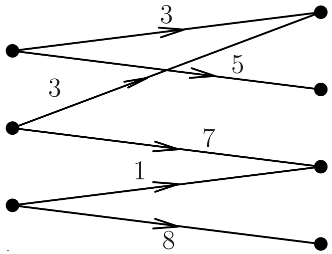
$6-\epsilon$	$2+\epsilon$	0	0
$+\epsilon$	$3-\epsilon$	7	0
0	0	1	8

and pick ϵ as large as possible (without any flow going negative) to obtain a new flow (for $\epsilon = 3$).

There is only one way to do this in a non-degenerate problem. The operation is perhaps clearer in the network picture.



We attempt to increase flow in the dotted arc. Adding an arc to a spanning tree creates a circuit. Increase flow around the circuit until one arc drops out, leaving a new spanning tree. The new solution is



7. Now return to step 3 and recompute node numbers. of

$\lambda_i \setminus \mu_j$	-5	-3	-7	-9
0	3 [5]	5 [3]	7 [4]	9 [6]
-3	3 [2]	0 [7]	4 [0]	6 [1]
-5	0 [5]	-2 [6]	1 [2]	8 [4]

In our example we obtain $\lambda_i = 0, -3, -5$ and $\mu_j = -5, -3, -7, -9$ at the next stage. The expression $d_{ij} - \lambda_i + \mu_j < 0$ for $(i, j) = (1, 3), (2, 4)$ and $(1, 4)$. Increase the flow in $(2, 4)$ by 7 to obtain the new flow below. This is now optimal, as we

can check from the final node numbers:

$\lambda_i \setminus \mu_j$	-5	-3	-2	-4
0	3 [5]	5 [3]	2 [4]	4 [6]
-3	3 [2]	0 [7]	-1 [4]	7 [1]
0	5 [5]	3 [6]	8 [2]	1 [4]

Remark. The route around which you may need to alter flows can be quite complicated though it is always clear how you should do it. For example, had we tried to increase the flow in arc $(3, 1)$ instead of $(2, 1)$ at step 5 we would have obtained

$6 - \epsilon$	$2 + \epsilon$	0	0
0	$3 - \epsilon$	$7 + \epsilon$	0
$+\epsilon$	0	$1 - \epsilon$	8

To summarise:

1. Pick initial feasible solution with $m + n - 1$ non-zero flows (NW corner rule).
2. Set $\lambda_1 = 0$ and compute λ_i, μ_j using $d_{ij} - \lambda_i + \mu_j = 0$ on arcs with non-zero flows.
3. If $d_{ij} - \lambda_i + \mu_j \geq 0$ for all (i, j) then flow is optimal.
4. If not, pick (i, j) for which $d_{ij} - \lambda_i + \mu_j < 0$.
5. Increase flow in arc (i, j) by as much as possible without making the flow in any other arc negative. Return to 2.

12.2 *Simplex-on-a-graph*

The transportation algorithm can easily be generalised to a problem of minimizing costs in a general network in which there is a constraint $0 \leq x_{ij} < \infty$ on each directed arc (i, j), and a flow b_i enters the network at each node i (though it is hard to keep track of all the numbers by hand). Here we don't label sources and destinations separately, but do allow $b_i \geq 0$ and $b_i \leq 0$. Clearly, $\sum_i b_i = 0$ for conservation of flow. The **simplex-on-a-graph algorithm** solves this problem in an identical fashion to the transportation algorithm. Once again a basic solution is a spanning tree of non-zero flow arcs. Suppose there are n nodes.

1. Pick an initial basic feasible solution. Obtain $n - 1$ non-zero flow arcs.

2. Set $\lambda_1 = 0$ and compute λ_i on other nodes using $d_{ij} - \lambda_i + \lambda_j = 0$ on arcs of the spanning tree.

3. Compute $d_{ij} - \lambda_i + \lambda_j$ for other arcs. If all these are ≥ 0 then optimal. If not, pick (i, j) such that $d_{ij} - \lambda_i + \lambda_j < 0$.

4. Add arc (i, j) to the tree. This creates a circuit. Increase flow around the circuit (in direction of arc (i, j)) until one non-zero flow drops to zero and a new basic solution is created. Return to 2.

If it is hard to find an initial basic solution then there is a two-phase version of the algorithm (just as for the ordinary simplex algorithm).

Phase I. Add artificial node 0 with arcs from all nodes with $b_i > 0$ and to all nodes with $b_i < 0$. For Phase I objective put costs of 1 on all arcs to and from node 0 and costs 0 elsewhere in the network. The initial basic solution for Phase I is the spanning tree consisting of the node 0 and all arcs joining it to the original nodes.

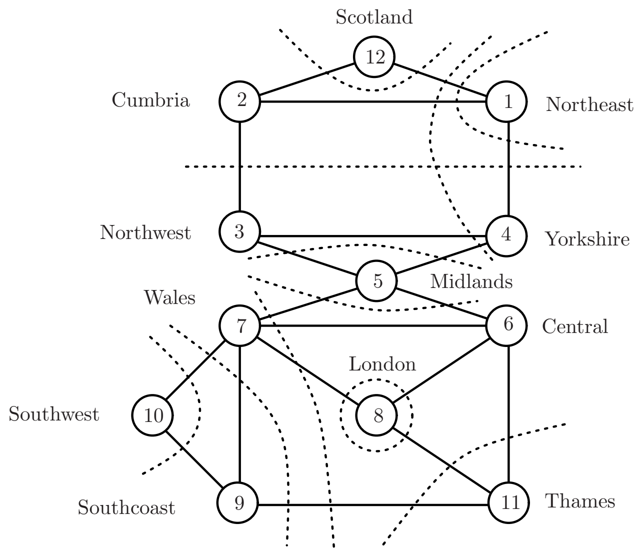
At the end of Phase I (if the original problem was feasible) you will have reduced the Phase I cost to 0 (no flow in arcs to or from node 0), so have a basic solution (Spanning tree solution) for the original problem.

Phase II. Solve the original problem using the initial solution found by Phase I.

There are several applications of the network theory to problems of graph theory and operations research on the further examples sheet.

12.3 Example: optimal power generation and distribution

The following real-life problem can be solved by the simplex-on-a-graph algorithm.



The demand for electricity at node i is d_i. Node i has k_i generators, that can generate electricity at costs of $a_{i1}, \ldots, a_{ik_i}$, up to amounts $b_{i1}, \ldots, b_{ik_i}$. There are $n = 12$ nodes and 351 generators in all. The capacity for transmission from node i to j is c_{ij} $(= c_{ji})$.

Let x_{ij} = amount of electricity carried $i \to j$ and let y_{ij} = amount of electricity generated by generator j at node i. The LP is

$$\text{minimize} \sum_{ij} a_{ij} y_{ij}$$

$$\text{subject to} \sum_j y_{ij} - \sum_j x_{ij} + \sum_j x_{ji} = d_i, \quad i = 1, \ldots, 12,$$

$$0 \leq x_{ij} \leq c_{ij}, \ 0 \leq y_{ij} \leq b_{ij}.$$

In addition, there are constraints on the maximum amount of power that may be shipped across the cuts shown by the dotted lines in the diagram.