I214 System Optimization
Lecture Notes

Kunihiko HIRAISHI, Mineo Kaneko and Brian Kurkoski
School of Information Science
Japan Advanced Institute of Science and Technology

2023 February 6

Chapters 1–4 originally by Prof. Kunihiko HIRAISHI. Chapters 5–6 originally by Prof. Mineo Kaneko. In 2021, Brian Kurkoski began teaching I214E at JAIST, with further changes made to this document.

Contents

Chapter 1

Introduction

.

This chapter gives an overview of various optimization problems, and the mathematical programming models used to solve them.

1.1 System Optimization

Optimization is widely used engineering, economics and social science. *Mathematical programming* is a method for solving optimization problems. It formulates a problem in the form of numerical formulas, and then applies a procedure that computes an optimal solution with respect to a given objective function.

We formulate a problem in the form of numerical formulas, and then apply a procedure that computes an optimal solution with respect to a given objective function.

The following topics will be studied in this course:

- Linear programming problems,
- Network optimization problems,
- Nonlinear programming problems,
- Combinatorial optimization problem.

The term "programming" means optimization and not computer programming. In the 1940s, these methods were used to propose training and logistic schedules. Such schedules can be called programs. Wikipedia: Mathematical optimization

References

1. W. Ogata, "TokyoTech Be-TEXT Mathematical Programming (in Japanese)", Ohm-sya.

2. N. Yamashita, M. Fukushima, “Mathematical Programming (in Japanese)”, IEICE Lecture Series C-4, Corona-sya.

3. H. Yabe, “Foundation of Engineering Optimization and its applications (in Japanese)”, Suuri-kougaku-sya.

4. M. Fukushima, “Introduction to Mathematical Programming (in Japanese)”, System-Control-

5. Ulrich Faigle, W. Kern, G. Still, “Algorithmic Principles of Mathematical Programming”, Springer.

6. B. Guenin, J. Könemann, L. Tuncel, “A Gentle Introduction to Optimization,” Cambridge University Press, 2014.

1.2 Linear Programming Problem

Problem 1.1. Problem [Ref. 4] A company, Stone River Manufacturing, produces three kinds of products I, II, and III from four kinds of materials A, B, C, and D.

- The profit per one unit of production:

I	II	III
\$70	\$120	\$30

- The amount of material necessary for one unit of production:

	I	II	III
A	5	0	6
B	0	2	8
C	7	0	15
D	3	11	0

- The maximum amount of material available per day:

A	B	C	D
80	50	100	70

Find a production plan that maximizes the total profit.

Let x_1, x_2, x_3 be variables that represent the amount of production for I, II, and III, respectively. Such variables are called *decision variables.* Then the objective of the problem is to find values of x_1, x_2, x_3 that maximize the value of the following function representing the total profit:

$$f(x_1, x_2, x_3) = 70x_1 + 120x_2 + 30x_3. \tag{1.1}$$

By the constraints on the maximum amount of available material, we have

$$\begin{array}{lllcr} 5x_1 & & +6x_3 & \leq & 80 \\ & 2x_2 & +8x_3 & \leq & 50 \\ 7x_1 & & +15x_3 & \leq & 100 \\ 3x_1 & +11x_2 & & \leq & 70 \end{array}$$

In addition, the amount of production should be nonnegative:

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0.$$

The function to be maximized (or minimized) is called the *objective function*. The restrictions on the variables are called the *constraints*. Any value assignment to the variables that satisfies all the constraints is called a *feasible solution*. A feasible solution is said to be an *optimal solution* if it maximizes (or minimizes) the objective function. The above problem is called a *linear programming problem* because all the constraints are linear equalities or inequalities and the objective function is a linear function of variables.

The general form of a linear programming problem has n variables $x_1, \ldots, x_n$ and m constraints $b_1, \ldots, b_m$. This is expressed in matrix form, given by:

$$\begin{array}{ll} \textbf{Maximize} & \mathbf{c}^t\mathbf{x} \\ \textbf{Subject to} & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{1.2}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \text{ and } \mathbf{0} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.$$

1.3 Network Optimization Problems

An undirected *graph* consists of nodes, and edges. An edge connects exactly two nodes. Each node is connected to at least one node. A node may also be called a vertex or a point. An example is shown in Fig. 1.1

Problem 1.2 (Ref. 4)**.** Fig. 1.2 shows a network of roads that connect seven cities *A*-*G*. The number on each edge indicates the distance between two cities. Find a shortest route from *A* to *G*.

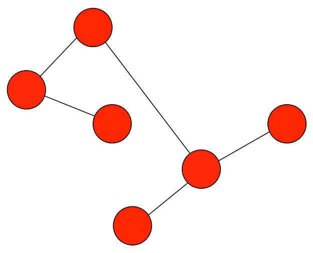


Figure 1.1: An example of a simple graph — 5 edges connect 6 nodes.

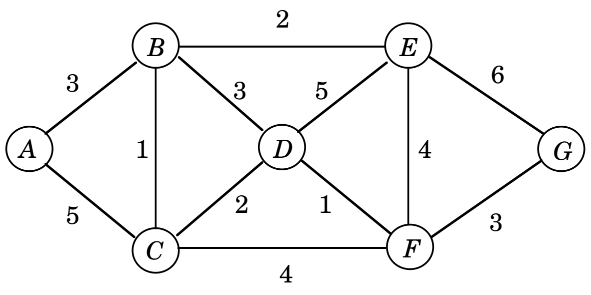


Figure 1.2: Example of the shortest path problem.

This problem is an example of the *shortest path problem* in algorithm theory. Since the distance is nonnegative, only routes without any cycles can be an optimal solution. Such a route is called a *path.* Since the number of paths from A to G is finite, the problem is solvable by enumerating all paths from A to G. However, this procedure is inefficient in general. For many problems the optimal solution contains optimal solutions to subproblems.

Problem 1.3 (Ref. 4)**.** Consider the Stone River Shipping Company. Fig. 1.3 shows a network of transportation routes. The number on each edge indicates the maximum amount of goods this company can transport per day. Find the maximum amount of goods this company can transport from city A to city F per day, where we are allowed to use multiple routes in parallel, and goods can be transshipped at each city.

This is an example of a *maximum flow problem.*

1.4 Nonlinear Programming

In nonlinear programming problems, either the objective function, or at least one of the constraints, or both, are nonlinear functions. In many cases, this nonlinear programming is more challenging than linear programming.

Problem 1.4. The price of goods may decrease if the amount of the goods in the market increases. In Stone River Manufacturing's Problem 1.1, assume that the profit per unit of production is determined by:

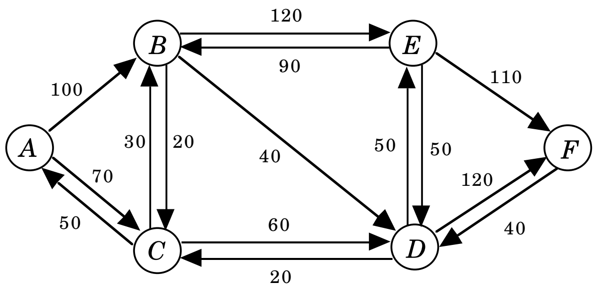


Figure 1.3: Maximum flow problem

I	II	III
$40 - x_1$	$100 - 3x_2$	$80 - 2x_3$

Then the problem becomes

Maximize

$$(40 - x_1)x_1 + (100 - 3x_2)x_2 + (80 - 2x_3)x_3$$

Subject to

$$\begin{array}{rrrcr} 5x_1 & & +6x_3 & \leq & 80 \\ 2x_2 & +8x_3 & & \leq & 50 \\ 7x_1 & & +15x_3 & \leq & 100 \\ 3x_1 & +11x_2 & & \leq & 70 \\ x_1, & x_2, & x_3 & \geq & 0 \end{array}$$

The objective function is not linear. This is a specific type of nonlinear programming problem called *quadratic programming*, because the objective function is quadratic.

The following is a more general nonlinear programming problem.

Problem 1.5 (Ref. 4)**.** Fig. 1.4 is a network of roads, where A is a residential area, D is the center of the city, and B, C are junctions. In every morning, w cars move from A to D. Find a best way of selecting routes that seems to be efficient from the perspective of *utilizing the road network.*

Let $f_i(x_i)$ denote the function indicating the time necessary for passing through road i. The total time is the number of cars times the time per car:

$$\sum_{i=1}^{5} x_i f_i(x_i).$$

Note that the number of cars entering the network is fixed at w.

For simplicity, only cars that move from A to D are on this network. Let x_i, $i = 1, \cdots, 5$ denote the number of cars that move along each road (see Fig. 1.4).

- The total number of cars that depart from A is w.

$$x_1 + x_2 = w.$$

- At junctions B and C, the number of arriving cars and the number of departing cars are the same:

$$x_1 - x_3 - x_4 = 0$$
$$x_2 + x_3 - x_5 = 0.$$

- The total number of cars that arrive at D is w:

$$x_4 + x_5 = w$$

- We cannot have non-negative number of cars on the road:

$$x_i \geq 0, i = 1, 2, 3, 4, 5.$$

Fig. 1.5 is a typical curve indicating the time necessary for *one car* to travel the road. It is a function of the number of cars x on the road. When x is small, the necessary time can be seen as a constant. However, once x reaches some threshold value, it increases very rapidly.

Now we can formulate the problem as follows:

$$\begin{array}{ll} \textbf{Minimize} & \sum_{i=1}^{5} x_i f_i(x_i) \\ \textbf{Subject to} & x_1 + x_2 = w \\ & x_1 - x_3 - x_4 = 0 \\ & x_2 + x_3 - x_5 = 0 \\ & x_4 + x_5 = w \\ & x_i \geq 0 \ \ (i = 1, 2, 3, 4, 5) \end{array}$$

Notice that the objective function is not linear.

1.5 Discrete Optimization

When every variable needs to be an integer, the problem is called an *integer programming problem.* When every feasible solution needs to satisfy some combinatorial constraints (e.g., the sequence of x_is is a permutation), the problem is called a *combinatorial optimization problem.* We use *discrete optimization problems* to denote both problems.

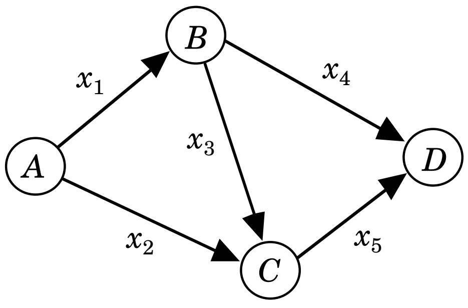


Figure 1.4: Routes between areas A, B, C and D.

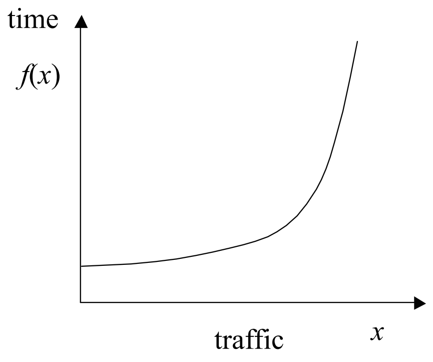


Figure 1.5: Time required to travel $f(x)$, as a function of the number of cars x.

Problem 1.6. There are n climbing items $i = 1, \cdots, n$ that will be packed in a knapsack; multiples of any item *are* allowed. Let a_i (kg) be the weight and let $c_i \in [0, 1]$ be the utility of item i. The total weight must be no more than W (kg). Then, for each item, find the number to be packed so that the total utility is maximized.

Utility (or value) c_i is a measure of the importance of the item; items with $c_i = 1$ have the greatest importance, and $c_1 = 0$ means least importance.

This problem is called the *knapsack problem.* Let x_i be number of item i. Then the problem is formulated as:

$$\begin{array}{ll} \textbf{Maximize} & \sum_{i=1}^{n} c_i x_i \\ \textbf{Subject to} & \sum_{i=1}^{n} a_i x_i \le b \\ & x_i \ge 0,\ x_i \in \mathbb{Z}\ (i = 1, \cdots, n) \end{array} \tag{1.3}$$

Since the items cannot be split into fractions, x_i is a non-negative integer and this is an integer programming problem.

The decision version of this problem (i. e., determine whether the total utility is less than or equal to a given number) is known to be an NP-complete problem.

The *one-machine scheduling problem* considers minimizing *tardiness.* If the deadline is Wednesday, but the job is completed on the following Friday, the tardiness is two days. If the job is completed before Wednesday, the tardiness is always 0.

Problem 1.7. There are n jobs to be processed on a machine. For job i, the processing time p_i and the due date d_i are given. The n jobs may be processed in any order. Find the processing order of jobs that minimizes the total tardiness.

Suppose $\pi(i)$ is the i-th job to be processed. (For example, $\pi(1) = 3$, $\pi(2) = 1$, $\pi(3) = 2$ corresponds to the permutation (3,1,2)). The start time $S_{\pi(i)}$ is given by:

$$\begin{aligned} S_{\pi(1)} &= 0 \\ S_{\pi(2)} &= S_{\pi(1)} + p_{\pi(1)} \\ &\vdots \\ S_{\pi(n)} &= S_{\pi(n-1)} + p_{\pi(n-1)} \end{aligned}$$

Then, the tardiness T_i of job i is:

$$T_i = \begin{cases} S_i + p_i - d_i & \text{if } S_i + p_i \geq d_i \\ 0 & \text{otherwise} \end{cases}$$

The objective function is

$$\textbf{Minimize} \quad \sum_{i=1}^{n} T_i$$

subject to the definition of tardiness using the variables d_i and p_i. This is an example of a combinatorial optimization problem, because the solution is given by a permutation of $1, \cdots, n$, equal to $J_1, J_2, \ldots, J_n$.

1.6 General Form of Mathematical Programming Problem

The general form of mathematical programming problems can be written as:

$$\begin{array}{ll} \textbf{Minimize} & f(\mathbf{x}) \\ \textbf{Subject to} & \mathbf{x} \in \mathcal{S} \end{array} \tag{1.4}$$

where $\mathbf{x}$ is a n-dimensional real vector of variables, and the objective function f is defined on the n-dimensional real vector space $\mathbb{R}^n$. $\mathcal{S} \subset \mathbb{R}^n$ is called the *feasible region.* Many of the problems in this chapter can be written in this general form.

1.7 Exercises

1.1 Formulate the following problem by using mathematical formulas:

A company produces single goods, and the demands for the goods at each quarter is expected as follows:

Spring	Summer	Autumn	Winter
20	30	50	60

- Current production rate is 55 units/quarter. The initial stock and the stock at the end of the fourth quarter are 0.

- When the amount of production changes between two successive quarters, \$500/unit is required for the change.
- At the end of each quarter, \$800/unit is requited for the amount of stock.
- At the end of the fourth quarter, production rate must return to 55.

Find a production plan that minimizes the total cost in the year.

1.2 Formulate the following problem using mathematical expressions. Consider a toll road that requires the number of employees at the toll gates indicated in the table below. Each employee works 4 hours, takes a rest for one hour, then works another 4 hours. An employee may start at any time. Minimize the number of persons to be hired.

time	required number of workers
0:00 – 06:00	2
06:00 – 10:00	8
10:00 – 12:00	4
12:00 – 16:00	3
16:00 – 18:00	6
18:00 – 22:00	5
22:00 – 24:00	3

1.3 Stone River Heating Oil supplies kerosene to customers in the Stone River area. The demand for kerosene is:

Month	1	2	3	4
Demand (liters)	5000	8000	9000	6000

At the beginning of each month, the company purchases oil (kerosene) from a supplier at the following rate:

Month	1	2	3	4
Price (\$/liter)	0.75	0.72	0.92	0.9

The company also has a storage tank with capacity 4000 liters. At the beginning of Month 1, it contains 2000 liters. In any month, purchased oil can be delivered directly to the customer; only oil left at the end of the month should be put into storage.

Write an LP that expresses this problem. The solution would express how much oil the company should purchase at the beginning of each month, in order to satisfy the customers demand.

1.4 The Stone River Hotel wants to rents rooms 1, 2 and 3 on New Year's Eve. There are four possible customers, Alice, Bob, Claire and David. Each customer is willing to pay a fixed amount for each room, as indicated by the following table:

Room	Alice's offer	Bob	Claire	David
1	\$60	\$40	not interested	\$65
2	\$50	\$70	\$55	\$90
3	not interested	\$80	\$75	not interested

The hotel wants to fill Rooms 1, 2, 3 with some potential client A, B, C, D:

- Each room is to be assigned to exactly one client
- Each client should be assigned to *at most* one room (there are more clients than rooms).

(a) Formulate this as an integer programming problem that maximizes the hotel's income.
(b) Alice and Bob have a history of loud and rude behavior when staying in the hotel. The hotel management decided to not rent rooms to *both* A and B. Modify your answer from part (a) to enforce this restriction.

1.5 Consider the following table indicating the nutritional value of different food types:

Foods	Prices ($) per serving	Calories per serving	Fat (g) per serving	Protein (g) per serving	Carbohydrate (g) per serving
Raw carrots	0.14	23	0.1	0.6	6
Baked potatoes	0.12	171	0.2	3.7	30
Wheat bread	0.2	65	0	2.2	13
Chedder cheese	0.75	112	9.3	7	0
Peanut butter	0.15	188	16	7.7	2

You need to decide how many servings of each food to buy each day so that you minimize the total cost of buying your food while satisfying the following daily nutritional requirements:

- calories must be at least 2000
- fat must be at least 50g
- protein must be at least 100g
- carbohydrates must be at least 250g

Write an LP that will help you decide how many servings of each of the aforementioned foods are needed to meet all the nutritional requirement, while minimizing the total cost of the food (you may buy fractional numbers of servings).

1.6 You are about to trek across the desert with a vehicle having 3.6 cubic metres ($3.6m^3$) of cargo space for goods. There are various types of items available for putting in this space, each with a different volume and a different net value for your trip, shown as follows:

Item type i	1	2	3	4	5	6	7
Volume v_i (m^3)	0.55	0.6	0.7	0.75	0.85	0.9	0.95
Net value n_i	250	300	500	700	750	900	950

(a) You need to decide which items to take, not exceeding the volume constraint. You can take at most one of any item. No item can be split into fractions. The total net value must be maximized. Formulate this problem as an LP or IP. (You may use the notation v_i and n_i for volume and net value of item i.)

(b) Your two friends have decided to come as well, each with an identical vehicle. There are exactly two items of each type. The question is, can you fit all 14 items in the vehicles without exceeding the volume constraints? No cutting items into pieces is permitted! Each item taken must be placed entirely in one of the vehicles. Formulate an LP or IP that has a feasible solution if and only if the items can be packed as desired. Describe how to determine from a feasible solution how to pack the items into the vehicles. Note that net value is ignored for part (b).

Chapter 2

Linear Programming 1

This chapter describes how to solve a linear program. After giving the formulation of the problem. Any linear program can be converted to the standard form, where non-trivial inequality constraints are converted to equality constraints. In the canonical form of an LP, some but not all variables depend on the identity matrix, and it is shown how to convert a problem to the canonical form. The geometrical interpretation is useful for understanding how to solve an LP. The simplex algorithm is a central technique to solve a linear program. First, a linear algebra perspective on the simplex algorithm is given. Then, the tabular perspective on the algorithm is given. Finally, the two-phase method is described, where the first phase deals with the problem of finding a basic feasible solution.

2.1 Formulation of Linear Programming Problems

2.1.1 Example

Problem 2.1. The Stone River Mining Company purchases two kinds of ore. The first ore contains iron and copper, and the content is 10% for each. The second ore contains iron and lead, and the iron content is 20% and the lead content is 5%. The price of the first ore is 20,000 yen per unit, and the price of the second ore is 10,000 yen per unit. Suppose that 0.8 units of iron, 0.2 units of copper, and 0.05 units of lead are need to be produced. Find the amount of each type of ore that should be purchased.

Decision variables:

- x_1: the amount of the first ore needed. $x_1 \geq 0$.
- x_2: the amount of the second ore needed. $x_2 \geq 0$.

Constraints:

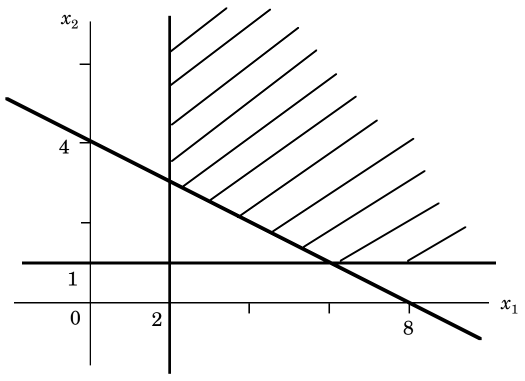


Figure 2.1: Feasible region corresponding to Problem 2.1, after scaling.

- Iron: $0.1x_1 + 0.2x_2 \geq 0.8$.
- Copper: $0.1x_1 \geq 0.2$.
- Lead: $0.05x_2 \geq 0.05$.

Cost:

- Minimize $2x_1 + x_2$ (20,000 yen and 10,000 yen per unit).

Feasible Solutions

We rewrite the constraints by scaling as follows:

$$\begin{array}{rrcl} x_1 & +2x_2 & \geq & 8 \\ x_1 & & \geq & 2 \\ & x_2 & \geq & 1 \\ x_1, & x_2 & \geq & 0 \end{array}$$

A solution that satisfies all the constraints is called a *feasible solution.* The set of all feasible solutions is a called the *feasible region.* Fig. 2.1 shows the region on the x_1-x_2 plane consisting of feasible solutions.

2.1.2 Optimal Solution

Any linear program has one of three possible outcomes:

1. It has one or more optimal solutions
2. It is unbounded
3. It is infeasible — no solution exists

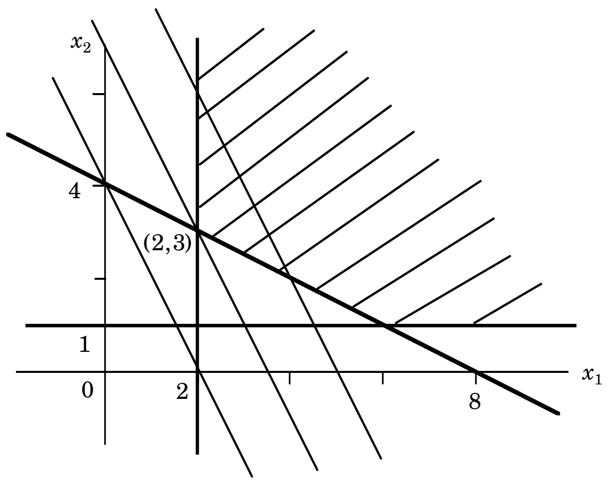


Figure 2.2: Feasible region, objective function and optimal solution. Several lines $k = 2x_1 + x_2$ show where the objective function is constant.

Since there are infinitely many feasible solutions in general, we need to give some criteria to select the best one. As the criterion, we usually give a function to be maximized (minimized). Such a function is called the *objective function.*

In this example, the objective is to minimize the objective function

$$2x_1 + x_2.$$

$$\begin{array}{rl} \textbf{Minimize} & 2x_1 + x_2 \\ \textbf{subject to} & x_1 + 2x_2 \geq 8 \\ & x_1 \geq 2 \\ & x_2 \geq 1 \\ & x_1,\ x_2 \geq 0 \end{array} \tag{2.1}$$

By drawing lines $k = 2x_1 + x_2$ on the feasible region (Fig. 2.2), we can find the optimal solution (2, 3).

- *Multiple optimal solutions* The optimal solution is not always unique, that is, there may be more than one optimal solution. For example, infinitely many optimal solutions exist if the objective function is "Minimize $f(x_1, x_2) = x_1 + 2x_2$".

- *Unbounded problems* Optimal solutions do not always exist, e.g., when "Maximize $2x_1 + x_2$", its value can be increased to the positive infinity. Such a problem is called *unbounded.*

In other words, not all problems have a single solution.

2.2 Standard Form

2.2.1 Definition

A linear programming problem is said to be in *standard form* if

- Every variable is nonnegative.
- Constraints other than nonnegativity of variables are equalities.
- The objective function is to be minimized (maximized).

$$\begin{array}{l} \textbf{Minimize} \\ \qquad c_1x_1 + \cdots + c_nx_n \\ \textbf{Subject to} \\ \qquad \begin{array}{cccccc} a_{11}x_1 & +\cdots & +a_{1n}x_n & = & b_1 \\ \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & +\cdots & +a_{mn}x_n & = & b_m \end{array} \\ \qquad x_j \geq 0 \quad (j = 1, \cdots, n) \end{array} \tag{2.2}$$

As was shown in Chapter 1, the standard form is expressed in matrix form as:

$$\begin{array}{ll} \textbf{Minimize} & \mathbf{c}^t\mathbf{x} \\ \textbf{subject to} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{2.3}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \text{ and } \mathbf{0} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.$$

2.2.2 Conversion to Standard Form

Any linear programming problem can be converted to a standard form problem, using the following rules.

1. **Inequality constraints** can be converted to equality constraints adding *slack variables.* Slack variables do not change the problem, but allow us to change inequality constraints into equality constraints.

For example:

$$x_1 + 2x_2 \leq 3 \tag{2.4}$$

is equivalent to

$$x_1 + 2x_2 + y = 3 \tag{2.5}$$
$$y \geq 0, \tag{2.6}$$

where the new nonnegative variable y is a slack variable. Likewise, for the $\geq$ inequality:

$$x_1 + 2x_2 \geq 3 \tag{2.7}$$

is equivalent to

$$x_1 + 2x_2 - y = 3 \tag{2.8}$$
$$y \geq 0. \tag{2.9}$$

2. **Free variables** can be converted to equality constraints. Suppose that x_j has no sign constraints (for example, $x_j < 0$ is not given). This is converted by replacing x_j with two new variables $x_j^+, x_j^- \geq 0$:

$$x_j = x_j^+ - x_j^-, \tag{2.10}$$
$$x_j^+, x_j^- \geq 0. \tag{2.11}$$

3. **Maximization** of f is equivalent to minimization of $-f$.

   For example:

$$\textbf{Maximize } 4x_1 + 5x_2 \tag{2.12}$$

is equivalent to

$$\textbf{Minimize } -4x_1 - 5x_2 \tag{2.13}$$

4. **Ignore constants in the objective function** Maximization of $f(\mathbf{x}) + c$ is equivalent to maximization of $f(\mathbf{x})$, where c is a constant. Also for minimization. This changes the value of the objective function, but not the optimal solution.

Example 2.1. Convert the following LP to standard form:

$$\begin{aligned}
&\textbf{Maximize}\\
&\qquad 4x_1 + 5x_2\\
&\textbf{subject to}\\
&\qquad 2x_1 + 5x_2 \leq 7\\
&\qquad 5x_1 + 6x_2 \leq 9\\
&\qquad 3x_1 + 2x_2 \geq 5\\
&\qquad x_1 \geq 0
\end{aligned}$$

The standard form of this problem is:

Minimize

$$-4x_1 - 5(x_2^+ - x_2^-)$$

Subject to

$$\begin{array}{cccccccc}
2x_1 & +5(x_2^+ & -x_2^-) & +y_1 & & & = & 7 \\
5x_1 & +6(x_2^+ & -x_2^-) & & +y_2 & & = & 9 \\
3x_1 & +2(x_2^+ & -x_2^-) & & & -y_3 & = & 5 \\
x_1, & x_2^+, & x_2^-, & y_1, & y_2, & y_3 & \geq & 0
\end{array}$$

Example 2.2. Formulate the following problem in standard form:

$$\begin{array}{rl}
\textbf{Maximize} & x_1 + x_2 \\
\textbf{subject to} & x_1 + 2x_2 \leq 3 \\
 & x_1 \leq 2 \\
 & x_2 \leq 1 \\
 & x_1,\ x_2 \geq 0
\end{array} \tag{2.14}$$

Using x_3, x_4 and x_5 as slack variables, the standard form is given by:

$$\begin{array}{rl}
\textbf{Minimize} & -x_1 - x_2 \\
\textbf{subject to} & x_1 + 2x_2 + x_3 = 3 \\
 & x_1 + x_4 = 2 \\
 & x_2 + x_5 = 1 \\
 & x_1,\ x_2,\ x_3,\ x_4,\ x_5 \geq 0
\end{array} \tag{2.15}$$

The matrix form for this LP in standard form is:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

2.3 Bases and Canonical Form

This section introduces the basis of the restrictions, and the canonical form of an LP problem. These are key for understanding the simplex method.

2.3.1 Indefinite Solutions

In this section, the following running example is considered:

Problem 2.2.

$$
\begin{array}{ll}
\textbf{Minimize} & -x_1 - x_2 \\
\textbf{subject to} & x_1 + 2x_2 + x_3 = 3 \\
& x_1 + x_4 = 2 \\
& x_2 + x_5 = 1 \\
& x_1,\ x_2,\ x_3,\ x_4,\ x_5 \geq 0
\end{array}
\tag{2.16}
$$

This problem is the same as (2.15). The matrix form is:

$$
\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.17}
$$

Note that the constraints in (2.16) has 5 variables but only 3 equations. Ignoring the other constraints, $\mathbf{Ax} = \mathbf{b}$ is an underdetermined system of equations. One perspective is to see three variables as dependent on the other two variables. For example, x_3, x_4, x_5 could be seen as dependent on x_1, x_2:

$$
\begin{array}{lllll}
x_3 & = 3 & -x_1 & -2x_2 \\
x_4 & = 2 & -x_1 & \\
x_5 & = 1 & -x_2 &
\end{array}
$$

Instead, x_1, x_2, x_4 could be seen as dependent on x_3, x_5:

$$
\begin{array}{llll}
x_1 & = 1 & -x_3 & +2x_5 \\
x_2 & = 1 & & -x_5 \\
x_4 & = 1 & +x_3 & -2x_5
\end{array}
$$

That is, there are many choices of the dependent and independent variables.

2.3.2 Bases

Recall the matrix $m \times n$ matrix of constraints $\mathbf{A}$, with $m < n$. The n columns of $\mathbf{A}$ can be partitioned into two sets: $\mathcal{B}$ consisting of m columns and $\mathcal{N}$ consisting of the other $n - m$ columns. Then, $\mathbf{A}_\mathcal{B}$ is an $m \times m$ matrix, and $\mathbf{A}_\mathcal{N}$ is an $(n - m) \times m$ matrix. In what follows, $\mathcal{B}$ will be the *basis* set, and $\mathcal{N}$ will be the *nonbasis* set.

- The chosen variables $\mathcal{B}$ are called *basic variables*, and other variables $\mathcal{N}$ are called *nonbasic variables*.
- The set of basic variables is called the *basis*, the set of nonbasic variables is called the *nonbasis*.
- Matrix $\mathbf{A}_\mathcal{B}$ is called the *basic matrix*, and matrix $\mathbf{A}_\mathcal{N}$ is called the *nonbasic matrix*.

Example 2.3. Consider $\mathcal{B} = \{1, 2, 4\}$ and $\mathcal{N} = \{3, 5\}$ for the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Then,

$$\mathbf{A}_{\mathcal{B}} = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{A}_{\mathcal{N}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Note that $\mathbf{A}_{\mathcal{B}}$ is full rank so $\mathbf{A}_{\mathcal{B}}^{-1}$ exists.

Instead consider a different choice, $\mathcal{B} = \{2, 3, 5\}$ and $\mathcal{N} = \{1, 4\}$. Then:

$$\mathbf{A}_{\mathcal{B}} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{A}_{\mathcal{N}} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

Here $\mathbf{A}_{\mathcal{B}}$ has rank 2, and is not invertible.

2.3.3 Basic Solution

A solution obtained by assigning 0 to the nonbasic variables $\mathbf{x}_{\mathcal{N}}$ is called a basic solution.

Definition 2.1. A solution $\mathbf{x}$ is called a *basic solution* if the following two conditions are satisfied:

$$\mathbf{x}_{\mathcal{N}} = \mathbf{0} \text{ and} \tag{2.18}$$
$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{2.19}$$

By choosing $\mathbf{x}_{\mathcal{N}} = \mathbf{0}$, the second condition is equivalent to $\mathbf{A}_{\mathcal{B}}\mathbf{x}_{\mathcal{B}} = \mathbf{b}$. If $\mathbf{A}_{\mathcal{B}}$ is invertible, then $\mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$.

- If a basic solution is feasible, then it is called a *basic feasible solution.*
- But, not all basic solutions are feasible. If at least one of $x_i \leq 0$, then the basic solution is not feasible.
- If a basic feasible solution is optimal, then it is called a *basic optimal solution.*
- We can prove that an optimal solution exists in the set of all basic feasible solutions.

Example 2.4. Continue Example 2.3. Find the basic solution with $\mathcal{B} = \{1, 2, 4\}$ and $\mathcal{N} = 3, 5$.

The definition $\mathbf{x}_{\mathcal{N}} = \mathbf{0}$ means $x_3 = x_5 = 0$. Then:

$$\mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 0 & 1 \\ -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Thus, the basic solution corresponding to $\mathcal{B} = \{1, 2, 4\}$ is $\mathbf{x}' = (1, 1, 0, 1, 0)$, which corresponds to objective function $z = -2$.

2.3.4 Canonical Form

Now, we can define the canonical form of an LP problem. Note that the canonical form is defined with respect to a basis $\mathcal{B}$.

Definition 2.2. An LP problem in the standard form (2.3) is in the *canonical form with respect to* $\mathcal{B}$ if $\mathbf{c}$ and $\mathbf{A}$ have the form:

$$\mathbf{c}_{\mathcal{B}} = 0. \tag{2.20}$$

$$\mathbf{A}_{\mathcal{B}} = \mathbf{I}_m \text{ and} \tag{2.21}$$

where $\mathbf{I}_m$ is the identity matrix.

An advantage of the canonical form is that it is easy to find a basic solution for a problem with constraints $\mathbf{Ax} = \mathbf{b}$. That is, if $\mathbf{A}$ is in canonical form with respect to $\mathcal{B}$, then $\mathbf{A}_{\mathcal{B}} = \mathbf{I}_m$ and:

$$\mathbf{x}_{\mathcal{B}} = \mathbf{b} \tag{2.22}$$

$$\mathbf{x}_{\mathcal{N}} = \mathbf{0} \tag{2.23}$$

is a basic solution. In addition the objective function $\mathbf{c}^t\mathbf{x} = 0$ for the basic solution. This is because $\mathbf{c}_{\mathcal{B}} = \mathbf{0}$ and $\mathbf{x}_{\mathcal{N}} = \mathbf{0}$.

Example 2.5. Continuing Example 2.2, the problem can be written as:

$$\begin{aligned} \textbf{Minimize} \quad & \mathbf{c}^t\mathbf{x} \\ \textbf{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \text{ and } \mathbf{c} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

With $\mathcal{B} = \{3,4,5\}$ and $\mathcal{N} = \{1,2\}$, $\mathbf{A}$ has the form of (2.21) and $\mathbf{c}$ has the form of (2.20), so this problem is in canonical form.

It can be seen that $\mathbf{x} = (0,0,3,2,1)^{\mathrm{t}}$ is a basic solution. In addition, the objective function is $z = 0$.

If an LP is not in canonical form with respect to $\mathcal{B}$, it may be converted to a canonical form if $\mathbf{A}_{\mathcal{B}}$ is nonsingular. We introduce a function $z(\mathbf{x})$ to denote the value of the objective function.

Proposition 2.1. Given an LP

$$\begin{array}{rr} \textbf{Maximize} & z(\mathbf{x}) = \mathbf{c}^{\mathrm{t}}\mathbf{x} \\ \textbf{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array} \tag{2.24}$$

and a basis $\mathcal{B}$ with $\mathbf{A}_{\mathcal{B}}$ nonsingular, then the LP:

$$\begin{array}{rl} \textbf{Maximize} & z(\mathbf{x}) = \mathbf{d}^{\mathrm{t}}\mathbf{b} + (\mathbf{c}^{\mathrm{t}} - \mathbf{d}^{\mathrm{t}}\mathbf{A})\mathbf{x} \\ \textbf{subject to} & \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} \\ & \mathbf{x} \geq 0 \end{array} \tag{2.25}$$

is in canonical form, where $\mathbf{d} = (\mathbf{A}_{\mathcal{B}}^{\mathrm{t}})^{-1}\mathbf{c}_{\mathcal{B}}$.

We can show the constraint part of the theorem as follows:

$$\mathbf{A}_{\mathcal{B}}\mathbf{x}_{\mathcal{B}} + \mathbf{A}_{\mathcal{N}}\mathbf{x}_{\mathcal{N}} = \mathbf{b} \tag{2.26}$$

When $\mathbf{A}_{\mathcal{B}}$ is nonsingular, its inverse $\mathbf{A}_{\mathcal{B}}^{-1}$ exists. Therefore, we obtain

$$\mathbf{x}_{\mathcal{B}} + \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}_{\mathcal{N}}\mathbf{x}_{\mathcal{N}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} \tag{2.27}$$

or

$$\left[\ \mathbf{I} \ \middle|\ \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}_{\mathcal{N}}\ \right] \begin{bmatrix} \mathbf{x}_{\mathcal{B}} \\ \mathbf{x}_{\mathcal{N}} \end{bmatrix} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} \tag{2.28}$$

Example 2.6. Find the canonical form of (2.17) with $\mathcal{B} = \{1,2,4\}$, we have:

$$\mathbf{A}_{\mathcal{B}} = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{d} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Then the LP in canonical form is:

$$\begin{array}{rl} \textbf{Minimize} & z = -2 + (0,0,1,0,-1)^{\mathrm{t}}\mathbf{x} \\ \textbf{subject to} & \begin{bmatrix} 1 & 0 & 1 & 0 & -2 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 2 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{2.29}$$

This is the canonical form for positions $\{1,2,4\}$. The basic solution is $(1,1,0,1,0)$ which gives $z = -2$.

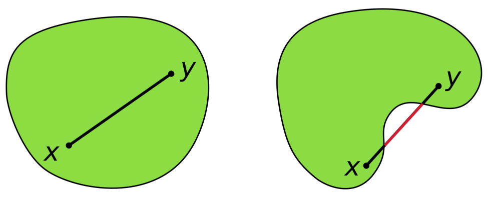


Figure 2.3: Left: a convex set. Right: a non-convex set. Image credit: Wikipedia/-Convex set.

It is important to know whether or not any given feasible solution is optimal or not. If the c_i are non-negative in the canonical form, then it is easy to find an optimal solution.

Proposition 2.2. For an LP in canonical form with a feasible basic solution $\mathbf{x}$, if $c_i \geq 0$ for all i, then $\mathbf{x}$ is an optimal feasible solution.

Proof Let $\mathcal{B}$ be the canonical form basis. $c_i \geq 0$ and $x_i \geq 0$, so the objective function $\mathbf{c}^t\mathbf{x} \geq 0$. But in canonical form, the basic solution $\mathbf{x}$ gives $\mathbf{c}^t\mathbf{x} = 0$, so the objective function is minimized. □

The basis $\mathcal{B}$ which satisfies the condition of Proposition 2.2 is called the *optimal basis.* Given the optimal basis, the optimal solution is easily found. By Proposition 2.1, the "new $\mathbf{b}$" is $\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$. The corresponding optimal solution $\mathbf{x}^*$ is given by $\mathbf{x}_{\mathcal{B}}^* = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$ and $\mathbf{x}_{\mathcal{N}}^* = \mathbf{0}$.

2.4 Geometrical Interpretations

We begin with two geometrical definitions.

Definition 2.3. A set $\mathcal{D} \subseteq \mathbb{R}^n$ is called a *convex set* if for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$:

$$\lambda\mathbf{x} + (1-\lambda)\mathbf{y} \in \mathcal{D}, \tag{2.30}$$

for all $0 \leq \lambda \leq 1$.

In other words, a set is convex if it contains all points on the line segment between $\mathbf{x}$ and $\mathbf{y}$. A set which is not convex is called non-convex. Fig. 5.1 shows examples of a convex set and a non-convex set. Wikipedia: Convex Set

A convex polytope may be defined as an intersection of a finite number of half-spaces. A half space is given by $\mathbf{a}^t\mathbf{x} \leq b$, and the intersection of multiple half-spaces is the set of $\mathbf{x} \in \mathbb{R}^n$ satsifying $\mathbf{Ax} \leq \mathbf{b}$. Wikipedia: Convex polytope

Definition 2.4. Let $\mathcal{S}$ be a convex set. If $\mathbf{z} \in \mathcal{S}$ cannot be represented by a linear combination of two points $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ for $\mathbf{x} \neq \mathbf{y}$:

$$\mathbf{z} = \lambda\mathbf{x} + (1-\lambda)\mathbf{y} \tag{2.31}$$

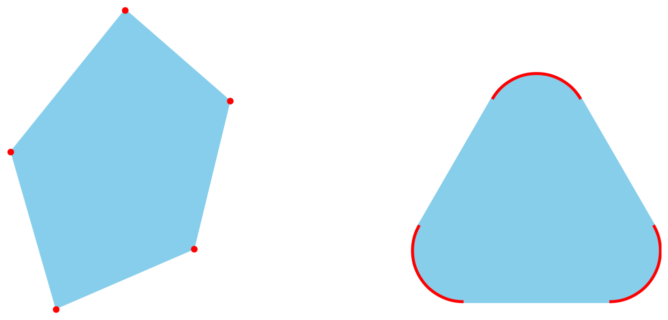


Figure 2.4: Extreme points in red, for two dimensions. Left: for a polyhedron, the extreme points are the corners. Right: if $\mathcal{S}$ has a curved boundary, points in the curved boundary form are extreme points. Image credit: Wikipedia/Extreme point.

for $0 < \lambda < 1$, then $\mathbf{z}$ is called an *extreme point* of $\mathcal{S}$.

For polytope, the extreme points are the corners. For more general shapes, extreme points may be in the curved boundaries. See Fig. 2.4. Wikipedia: Extreme Point

Extreme points are important because at least one solution to the optimization problem will be at an extreme point:

Proposition 2.3. For an LP problem over a feasible region $\mathcal{S}$, if the there is a unique solution, then it will occur at an extreme point of $\mathcal{S}$.

If there is only one solution, then it will occur at an extreme point. But since it is possible that there are multiple solutions, not all are guaranteed to be at an extreme point.

Consider a geometrical interpretation of the running example as given in Example 2.2 in the following form:

$$\begin{array}{rl} \textbf{Minimize} & -x_1 - x_2 \\ \textbf{subject to} & x_1 + 2x_2 \le 3 \\ & x_1 \le 2 \\ & x_2 \le 1 \\ & x_1, \quad x_2 \ge 0 \end{array} \tag{2.32}$$

The feasible region is the set of points surrounded by the following five lines,

$$\begin{array}{llr} x_1 & +2x_2 & = 3 \\ x_1 & & = 2 \\ & x_2 & = 1 \\ x_1 & & = 0 \\ & x_2 & = 0 \end{array}$$

as shown in Fig. 2.5. The feasible region forms a convex polytope.

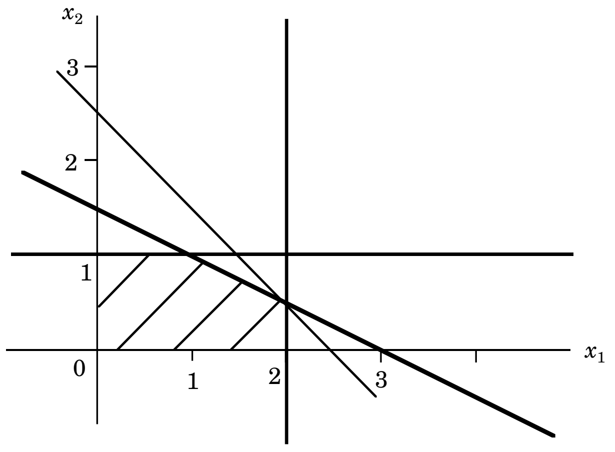


Figure 2.5: Geometrical interpretation.

The feasible region shown in Fig. 2.5 has five extreme points:

$$(0,0), (2,0), (2,0.5), (1,1), (0,1).$$

There exists one-to-one correspondence between basic feasible solutions and extreme points unless more than two lines cross at a single point.

There are $\binom{5}{3} = 10$ ways to choose the basis variables for the running example, generating 10 distinct candidates. In some cases, these correspond to basic feasible solutions, which are extreme points; in some cases no basic solution exists.

1. Basis $\{x_3, x_4, x_5\}$:

$$\begin{array}{llllllll} z & & & +x_1 & +x_2 & = & 0 \\ x_3 & & & +x_1 & +2x_2 & = & 3 \\ & x_4 & & +x_1 & & = & 2 \\ & & x_5 & & +x_2 & = & 1 \end{array}$$

   Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (0,\ 0)$, $z = 0$.

2. Basis $\{x_1,\ x_3,\ x_5\}$:

$$\begin{array}{lllllll} z & & & +x_2 & -x_4 & = & -2 \\ x_1 & & & & +x_4 & = & 2 \\ & x_3 & & +2x_2 & -x_4 & = & 1 \\ & & x_5 & & +x_2 & = & 1 \end{array}$$

   Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (2,\ 0)$, $z = -2$.

3. Basis $\{x_1, x_2, x_5\}$:

$$\begin{array}{llllllr} z & & & -0.5x_3 & +0.5x_4 & = & -2.5 \\ x_1 & & & & +x_4 & = & 2 \\ & x_2 & & +0.5x_3 & -0.5x_4 & = & 0.5 \\ & & x_5 & -0.5x_3 & +0.5x_4 & = & 0.5 \end{array}$$

Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (2,\ 0.5)$, $z = -2.5$.

4. Basis $\{x_1,\ x_2,\ x_4\}$:

$$\begin{array}{lllllcr} z & & & -x_3 & +x_5 & = & -2 \\ x_1 & & & +x_3 & -2x_5 & = & 1 \\ & x_2 & & & +x_5 & = & 1 \\ & & x_4 & -x_3 & +2x_5 & = & 1 \end{array}$$

Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (1,\ 1)$, $z = -2$.

5. Basis $\{x_2,\ x_3,\ x_4\}$:

$$\begin{array}{lllllcr} z & & & +x_1 & -x_5 & = & -1 \\ x_2 & & & & +x_5 & = & 1 \\ & x_3 & & +x_1 & -2x_5 & = & 1 \\ & & x_4 & +x_1 & & = & 2 \end{array}$$

Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (0,\ 1)$, $z = -1$.

6. Basis $\{x_1,\ x_4,\ x_5\}$:

$$\begin{array}{lllllcr} z & & & -x_2 & -x_3 & = & -3 \\ x_1 & & & +2x_2 & +x_3 & = & 3 \\ & x_4 & & -2x_2 & -x_3 & = & -1 \\ & & x_5 & +x_2 & & = & 1 \end{array}$$

Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (3,\ 0)$, $z = -3$. Infeasible.

7. Basis $\{x_1,\ x_2,\ x_3\}$:

$$\begin{array}{lllllcr} z & & & -x_4 & -x_5 & = & -3 \\ x_1 & & & +x_4 & & = & 2 \\ & x_2 & & & +x_5 & = & 1 \\ & & x_3 & -x_4 & -2x_5 & = & -1 \end{array}$$

Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (2,\ 1)$, $z = -3$. Infeasible.

8. Basis $\{x_2,\ x_4,\ x_5\}$:

$$\begin{array}{lllllcr} z & & & +0.5x_1 & -0.5x_3 & = & -1.5 \\ x_2 & & & +0.5x_1 & +0.5x_3 & = & 1.5 \\ & x_4 & & +x_1 & & = & 2 \\ & & x_5 & -0.5x_1 & -0.5x_3 & = & -0.5 \end{array}$$

Assigning 0 to all nonbasic variables, we obtain $(x_1,\ x_2) = (0,\ 1.5)$, $z = -1.5$. Infeasible.

9. $\{x_1,\ x_3,\ x_4\}$ cannot be a basis, because $x_2 = 0,\ x_2 = 1$ are parallel lines.

10. $\{x_2,\ x_3,\ x_5\}$ cannot be a basis, because $x_1 = 0,\ x_1 = 2$ are parallel lines.

2.4.1 Complexity of Brute-Force Solution

There exists at most $\binom{n}{m}$ basic solutions. Since the set of basic solutions is finite, the problem is solvable in finite time. A brute-force approach is to enumerating all basic solutions, and take the solution with greatest objective function. Since $\binom{n}{m}$ is exponential:

$$\begin{aligned}\binom{n}{m} &= \frac{n(n-1)\cdots(n-m+1)}{m(m-1)\cdots 1} \\ &= \frac{n}{m}\frac{n-1}{m-1}\cdots\frac{n-m+1}{1} \\ &\geq (\frac{n}{m})^m.\end{aligned}$$

the brute-force approach is inefficient, as there are an exponential number of candidates.

2.5 Simplex Method, Linear Algebra Perspective

Wikipedia: Simplex Algorithm

2.5.1 Simplex Iteration

The following represents one iteration of the simplex algorithm. The essential idea is to begin with a problem in canonical form, and note the value of the objective function. Then, find a new solution which decreases (increases) the objective function. This new solution is used to find a new canonical form.

1. Start with the problem in canonical form. We easily get a basic solution $\mathbf{x}$ and its value $z(\mathbf{x})$.
2. If all $c_i \geq 0$, then stop and output the basic solution $\mathbf{x}$ as the optimal solution.
3. Choose one of the c_i satisfying $c_i < 0$. The corresponding variable x_i is increased to $t > 0$. Since $c_i < 0$, if we increase x_i, then $c_i x_i$ decreases, and the objective function decreases. Other non-basic variables remain 0. Write $\mathbf{x}_{\mathcal{N}} = (t, 0, \ldots, 0)$.
4. Find the largest value of t that follows the constraints, using the ratio test. $\mathbf{Ax} = \mathbf{b}$ can be written as $t\mathbf{A}_i + \mathbf{x}_{\mathcal{B}} = \mathbf{b}$. Apply the ratio test: $\mathbf{x}_{\mathcal{B}} = \mathbf{b} - t\mathbf{A}_i \geq \mathbf{0} \Rightarrow \mathbf{b} \geq t\mathbf{A}_i$.
5. Using this value of t, get a new solution $\mathbf{x}'$ and corresponding $z(\mathbf{x}')$.
6. The new $\mathbf{x}'$ is a basic solution because it has m non-zero values. These new m values form new basis $\mathcal{B}'$. Put the problem into canonical form with respect to $\mathcal{B}'$.
7. Go to Step 1.

2.5.2 Simplex Iteration Example

Suppose we have a problem in the canonical form, $\min \mathbf{c}^t\mathbf{x}$ with:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \text{ and } \mathbf{c} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.33}$$

With basis $\mathcal{B} = \{3, 4, 5\}$, the basic solution is $\mathbf{x} = (0, 0, 3, 2, 1)$. The value of the objective function is $z(\mathbf{x}) = \mathbf{c}^t\mathbf{x} = 0$.

Can we decrease z by increasing the value the nonbasic variables x_1 or x_2? Yes — let us decrease z by increasing x_1 and keeping $x_2 = 0$. In other words, look for a new feasible solution $\mathbf{x}$ with where $x_1 = t$. Since the new values must satisfy the constraints:

$$\mathbf{b} = \mathbf{A}\mathbf{x} \tag{2.34}$$

$$\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \tag{2.35}$$

$$= \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} t + \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} 0 + \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \tag{2.36}$$

$$\begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} t \tag{2.37}$$

Apply the constraint $x_3, x_4, x_5 \geq 0$:

$$\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} t \tag{2.38}$$

The first line gives $t \leq 3$, and the second line gives $t \leq 2$. Note that the third line does not add any restriction to the choice of t. Also, any inequality $t \geq a/b$ can also be ignored. We want t as large as possible while satisfying the constraints, so choose the smallest t which satisfies the non-negative ratios:

$$t = \min \left\{ \frac{3}{1}, \frac{2}{1}, - \right\} \tag{2.39}$$

That is, $t = 2$. Note that by choosing $t = 2$, we get that $x_4 = 0$.

Apply this value of $t = 2$ to $\mathbf{x}_{\mathcal{B}} = \mathbf{b} - \mathbf{A}_i t$ given in (2.37), to obtain the new solution $\mathbf{x}' = (2, 0, 1, 0, 1)$. This solution gives objective function $z(\mathbf{x}') = \mathbf{c}^t\mathbf{x}' = -2$, which has improved the solution.

Since $\mathbf{x}' = (2, 0, 1, 0, 1)$, choose a new basis $\mathcal{B}' = \{1, 3, 5\}$. Using Proposition 2.1, the problem can be written in a new canonical form: $\min \mathbf{c}^t\mathbf{x}$, with

$$\mathbf{A}' = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{b}' = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \text{ and } \mathbf{c}' = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.40}$$

Note that the linear term $\mathbf{d}^t\mathbf{b}$ given in Proposition 2.1 has been dropped because it does not affect the choice of $\mathbf{x}$ (it does affect the value of $z(\mathbf{x})$, but this is not significant). Note also that we started with $\mathcal{B} = \{3, 4, 5\}$ and we finished with $\mathcal{B}' = \{1, 3, 5\}$ so we say that x_4 left the basis, and x_1 entered the basis.

2.5.3 Simplex Example, Second Iteration

Now we iterate, explicitly using the steps earlier.

1. Clearly (2.40) is in canonical form with $\mathcal{B} = \{1, 3, 5\}$ and $\mathcal{B} = \{2, 4\}$.
2. The only negative c value is c_2, so we choose x_2 to enter the basis.
3. Apply the ratio test to x_2. Using column $\mathbf{A}_2$ and $\mathbf{b}$ from (2.40):

$$\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} t \tag{2.41}$$

Then the result of the ratio test is:

$$t = \min\left\{-, \frac{1}{2}, \frac{1}{1}\right\} = \frac{1}{2} \tag{2.42}$$

The new solution is:

$$\begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} t = \begin{bmatrix} 2 \\ 0 \\ \frac{1}{2} \end{bmatrix}. \tag{2.43}$$

4. With $t = \frac{1}{2}$, the new feasible solution is $\mathbf{x}' = (2, \frac{1}{2}, 0, 0, 1)$.

5. x_3 left the basis, and x_2 entered the basis, and the new basis is $\mathcal{B}' = \{1, 2, 5\}$. Using this we put the problem into canonical form:

$$\mathbf{A}' = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0.5 & -0.5 & 0 \\ 0 & 0 & -0.5 & 0.5 & 1 \end{bmatrix}, \mathbf{b}' = \begin{bmatrix} 2 \\ 0.5 \\ 0.5 \end{bmatrix} \text{ and } \mathbf{c}' = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0.5 \\ 0 \end{bmatrix} \tag{2.44}$$

2.5.4 Simplex Example, Third Iteration

For the third iteration, we again have the problem in canonical form. However, we note that that all $c_i \geq 0$, and thus there is no variable to select to decrease $z(\mathbf{x})$. Since all $c_i \geq 0$,, this solution is optimal, by Proposition 2.2. The optimal solution is $(2, \frac{1}{2}, 0, 0, \frac{1}{2})$. If we apply to $z(\mathbf{x}) = \mathbf{c}^{\mathrm{t}}\mathbf{x}$ in (2.33), the minimum value is $z(\mathbf{x}) = -\frac{5}{2}$.

2.6 Simplex Method Using Tabular

This section describes the tabular simplex method, which is a compact way to perform the operations described in the previous section. Instead of matrix inversion, row and column operations are performed on the tabular.

In what follows, we assume that $\operatorname{rank}(\mathbf{A}) = m$, that is, no redundant equations exist. If they exist, they could be removed.

2.6.1 Generalized Simplex Method

Here, the simplex method is generalized.

Simplex Method:

1. Find a feasible basic solution.

2. If every coefficient in z-row is nonpositive, then stop. The basic solution is optimal.

3. Select the *pivot column*. Find a column with a positive coefficient in z-row. When there are more than one such rows, we can select any of them, but one with the largest value is often selected. The nonbasic variable corresponding to the column *enters* the basis.

4. Apply the ratio test to find the *pivot row*. Ratio test: divide the value of the basic solution by the coefficient in the pivot column. The pivot row is the row the row with the smallest ratio; negative and zero divisor rows are ignored. If every coefficient is nonpositive, then halt (the problem is unbounded). The basic variable corresponding to the row *leaves* the basis.

5. Apply the pivot operation to the pivot row and pivot column. That is, perform linear operations so that the pivot column has zeros in all rows, except a 1 in the pivot row. Go to Step 2.

2.6.2 Simplex Tabular Example, First Iteration

Continue our running example, the standard-form LP has

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \text{ and } \mathbf{c} = \begin{bmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.45}$$

The basis is $\mathcal{B} = \{3, 4, 5\}$. The objective function is $\min z$ where $z = -x_1 - x_2$, but this is instead written as an equation $z + x_1 + x_2 = 0$.

We use the following table, called a *simplex tabular*, to represent this problem.

		x_1	x_2	x_3	x_4	x_5
z	0	1	1	0	0	0
x_3	3	1	2	1	0	0
x_4	2	1	0	0	1	0
x_5	1	0	1	0	0	1

The left column shows z, and the basis variables x_3, x_4, x_5. The top row shows the vector $-\mathbf{c}$ (rather than $\mathbf{c}$) because the objective function has been written as an equation.

Choose a non-negative value from the z row whose variable will enter the basis. As before, choose x_1. Apply the ratio test:

$$min\left\{\frac{3}{1}, \frac{2}{1}, -\right\} = 2, \tag{2.46}$$

noting that the arg min is x_4, that is the minimum is due to x_4 and it should leave the basis.

Since the new basis is $\{1, 3, 5\}$, we want to use linear operations to but the tabular in canonical form with respect to this basis.

Pivot operations, i.e. linear operations, give the new tabular:

		x_1	x_2	x_3	x_4	x_5
z	-2	0	1	0	-1	0
x_3	1	0	2	1	-1	0
x_1	2	1	0	0	1	0
x_5	1	0	1	0	0	1

which is obtained as follows:

1. Multiply x_4-row by some constant so that the coefficient of x_1-column becomes 1. (This step is not necessary in the example because the coefficient is already 1).
2. To other rows (z-row, x_3-row, x_5-row), add x_4-row multiplied by some constant so that the coefficient of x_1-column becomes 0. (In the example, addition of x_4-row multiplied by -1 to both z-row and x_3-row will make the coefficients 0. The coefficient of x_5-row is already 0.)

2.6.3 Simplex Tabular Example, Second Iteration

Continue the example.

		x_1	x_2	x_3	x_4	x_5
z	-2	0	1	0	-1	0
x_3	1	0	[2]	1	-1	0
x_1	2	1	0	0	1	0
x_5	1	0	1	0	0	1

Only x_2 has a positive coefficient in z-row, so it is selected to enter the basis. The ratio test gives:

$$min \left\{ \frac{1}{2}, -, \frac{1}{1} \right\} = \frac{1}{2}$$

and the x_3-row is selected to leave the basis. Applying the pivot operation, the new tabular is obtained:

		x_1	x_2	x_3	x_4	x_5
z	-2.5	0	0	-0.5	-0.5	0
x_2	0.5	0	1	0.5	-0.5	0
x_1	2	1	0	0	1	0
x_5	0.5	0	0	-0.5	0.5	1

Since no coefficient in z-row is positive, the basic solution

$$x_1 = 2, \; x_2 = 0.5$$

is optimal. The optimal value of z is -2.5.

2.6.4 Unbounded Problems

Recall one possible outcome of an LP is that it is unbounded. A problem is unbounded if for any nonbasic variable k in canonical form, if the coefficient k is the z row is positive *and* all the coefficients in the column are $\mathbf{A}_k \leq \mathbf{0}$, then the problem is unbounded.

The following simplex tabular is an example of an unbounded problem.

		x_1	x_2	x_3	x_4
z	0	1	1	0	0
x_3	0.5	-1	2	1	0
x_4	1	0	1	0	1

Fixing the value of x_2 to 0, we consider to increase x_1. From constraint $x_3 = 0.5 + x_1 \geq 0$, we have $x_1 \geq -0.5$. Constraint $x_4 = 1$ is irrelevant. Therefore, there is no upper bound of x_1, and as a result z can be decreased to the minus infinity.

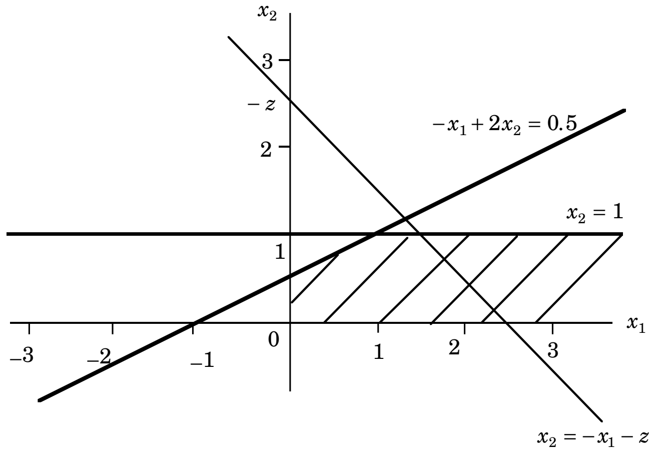


Figure 2.6: Unbounded.

The feasible region of the problem

$$\begin{array}{ll}
\textbf{Minimize} & z, \\
 & z + x_1 + x_2 = 0 \\
\textbf{Subject to} & \\
 & \begin{array}{rrcr} -x_1 & +2x_2 & \leq & 0.5 \\ & x_2 & \leq & 1 \\ x_1, & x_2 & \geq & 0 \end{array}
\end{array}$$

is depicted in Fig. 2.6.

2.6.5 Cycling

Cycling is a situation that the same basis appears more than once in the iteration. This may happen when the improvement of z is 0.

This phenomenon can be avoided by selecting the pivot row and the column using *Bland's Rule*, as follows:

Pivot column Find a column with a positive coefficient in z-row. When there are more than one such rows, select one with the smallest index.

Pivot row Each value of the basic solution is divided by the coefficient in the pivot column, provided that the coefficient is positive (computing the ratio). Select one row with the smallest value. If there are more than one such rows, then select one for which the corresponding basic variable has the smallest index.

In general, this selection method gives slower convergence.

2.6.6 Examples

Example 2.7.

Minimize

$$2x_1 + x_2$$

Subject to

$$\begin{array}{llcl} x_1 & +2x_2 & \geq & 8 \\ x_1 & & \geq & 2 \\ & x_2 & \geq & 1 \\ x_1, & x_2 & \geq & 0 \end{array}$$

Standard form:

Minimize

$$2x_1 + x_2$$

Subject to

$$\begin{array}{lllllcll} x_1 & +2x_2 & -y_1 & & & = & 8 & (a) \\ x_1 & & & -y_2 & & = & 2 & (b) \\ & x_2 & & & -y_3 & = & 1 & (c) \\ x_1, & x_2, & y_1, & y_2, & y_3 & \geq & 0 & \end{array}$$

The initial feasible basic form is derived as follows.

Transformation of the constraints:

$$\begin{array}{llllll} & & -y_1 & +y_2 & +2y_3 & = 4 \quad (a') = (a) - (b) - (c) * 2 \\ x_1 & & & -y_2 & & = 2 \quad (b) \\ & x_2 & & & -y_3 & = 1 \quad (c) \\ x_1, & x_2, & y_1, & y_2, & y_3 & \geq 0 \end{array}$$

$$\begin{array}{llllll} & & -y_1 & +y_2 & +2y_3 & = 4 \quad (a') \\ x_1 & & -y_1 & & +2y_3 & = 6 \quad (b') = (b) + (a') \\ & x_2 & & & -y_3 & = 1 \quad (c) \\ x_1, & x_2, & y_1, & y_2, & y_3 & \geq 0 \end{array}$$

The objective function:

$$z - 2(6 + y_1 - 2y_3) - (1 + y_3) = 0$$

$$z - 2y_1 + 3y_3 = 13$$

Simplex tabular:

		x_1	x_2	y_1	y_2	y_3
z	13	0	0	−2	0	3
y_2	4	0	0	−1	1	2
x_1	6	1	0	−1	0	2
x_2	1	0	1	0	0	−1

y_3-column is selected.

$$2 = min \left\{ \frac{4}{2} = 2, \ \frac{6}{2} = 3 \right\}$$

y_2-row is selected.

		x_1	x_2	y_1	y_2	y_3
z	7	0	0	-0.5	-1.5	0
y_3	2	0	0	-0.5	0.5	1
x_1	2	1	0	0	-1	0
x_2	3	0	1	-0.5	0.5	0

Optimal solution: $x_1 = 2,\ x_2 = 3.$

Example 2.8. Use the simplex method to solve the following problem.

Maximize

$$3x_1 + 2x_2 + 4x_3$$

Subject to

$$\begin{array}{llll} x_1 & +x_2 & +2x_3 & \leq 4 \\ 2x_1 & & +2x_3 & \leq 5 \\ 2x_1 & +x_2 & +3x_3 & \leq 7 \\ x_1, & x_2, & x_3 & \geq 0 \end{array}$$

Standard form:

Minimize

$$-3x_1 - 2x_2 - 4x_3$$

Subject to

$$\begin{array}{lllllll} x_1 & +x_2 & +2x_3 & +y_1 & & & = 4 \\ 2x_1 & & +2x_3 & & +y_2 & & = 5 \\ 2x_1 & +x_2 & +3x_3 & & & +y_3 & = 7 \\ x_1, & x_2, & x_3 & y_1, & y_2, & y_3 & \geq 0 \end{array}$$

Simplex tabular:

		x_1	x_2	x_3	y_1	y_2	y_3
z	0	3	2	4	0	0	0
y_1	4	1	1	[2]	1	0	0
y_2	5	2	0	2	0	1	0
y_3	7	2	1	3	0	0	1

x_3-column with the largest positive coefficient is selected.

$$2 = min\left\{\frac{4}{2}, \frac{5}{2}, \frac{7}{3}\right\}$$

By comparing ratios, y_1-row is selected. Pivot operation is applied.

		x_1	x_2	x_3	y_1	y_2	y_3
z	-8	1	0	0	-2	0	0
x_3	2	0.5	0.5	1	0.5	0	0
y_2	1	[1]	-1	0	-1	1	0
y_3	1	0.5	-0.5	0	-1.5	0	1

x_1-column is selected.

$$1 = min\left\{\frac{2}{0.5}, \frac{1}{1}, \frac{1}{0.5}\right\}$$

y_2-row is selected.

		x_1	x_2	x_3	y_1	y_2	y_3
z	-9	0	1	0	-1	-1	0
x_3	1.5	0	[1]	1	1	-0.5	0
x_1	1	1	-1	0	-1	1	0
y_3	0.5	0	0	0	-1	-0.5	1

We apply pivot operation between x_2-column and x_3-row.

		x_1	x_2	x_3	y_1	y_2	y_3
z	-10.5	0	0	-1	-2	-0.5	0
x_2	1.5	0	1	1	1	-0.5	0
x_1	2.5	1	0	1	0	0.5	0
y_3	0.5	0	0	0	-1	-0.5	1

Optimal solution: $x_1 = 2.5,\ x_2 = 1.5,\ x_3 = 0.$

2.7 Two-Phase Simplex Algorithm

2.7.1 Phase 1: Compute the Initial Basic Solution

In order to start the simplex method, an initial basic solution is needed. If all constants in the right side are nonnegative and all inequalities are $\leq$, then the slack variables can be initial basic variable. However, in general we do not have this form and usually it is not easy to find the initial basic solution.

For example, consider the following problem with x_3, x_4 as the basic variables.

$$\begin{array}{llllll}
\textbf{Minimize} & z, & & & & \\
 & z - 3x_1 - 2x_2 = 0 & & & & \\
\textbf{Subject to} & & & & & \\
 & -2x_1 & -x_2 & +x_3 & & = & -2 \\
 & -4x_1 & -3x_2 & & +x_4 & = & -6 \\
 & x_1, & x_2, & x_3, & x_4 & \geq & 0
\end{array} \tag{2.47}$$

The basic solution is $x_1 = x_2 = 0,\ x_3 = -2,\ x_4 = -6$, which is infeasible.

In general, consider the standard form LP:

$$\begin{array}{ll}
\textbf{Minimize} & \mathbf{c}^t\mathbf{x} \\
\textbf{subject to} & \mathbf{Ax} = \mathbf{b} \\
 & \mathbf{x} \geq 0
\end{array} \tag{2.48}$$

It is possible to construct an *auxiliary problem* by introducing a vector of *artificial variables* $\mathbf{y} = (y_1, y_2, \dots, y_m)$. The auxiliary problem is:

$$\begin{array}{ll} \textbf{Minimize} & y_1 + y_2 + \cdots + y_m \\ \textbf{subject to} & \mathbf{Ax} + \mathbf{y} = \mathbf{b} \\ & \mathbf{x} \geq 0 \\ & \mathbf{y} \geq 0 \end{array} \tag{2.49}$$

The feasible solution to the auxiliary problem $\mathbf{x} = \mathbf{0}$ is $\mathbf{y} = \mathbf{b}$ is easily obtained.

2.7.2 Two-Phase Method

Proposition 2.4. The optimal solution to the auxiliary problem (**??**) is a feasible solution to the main problem (2.48).

This proposition gives a method to solve any LP using the Two-Phase Method:

Phase 1 Construct the auxiliary problem. **During iterations, move the all of the artificial variables into the nonbasis** (it is not necessary to completely solve the auxiliary problem).

Phase 2 Now you have a canonical form of the problem with a basis in the original variables. Remove the artificial variables y and proceed with standard simplex method.

We first solve the following *Phase I* problem:

$$\begin{array}{lccccccccc} \textbf{Minimize} & w, \\ & w - y_1 - y_2 = 0 \\ \textbf{Constraint} \\ z & -3x_1 & -2x_2 & & & & & = & 0 \\ & 2x_1 & +x_2 & -x_3 & & +y_1 & & = & 2 \\ & 4x_1 & +3x_2 & & -x_4 & & +y_2 & = & 6 \\ & x_1, & x_2, & x_3, & x_4, & y_1, & y_2 & \geq & 0 \end{array} \tag{2.50}$$

where $y_1,\ y_2$ are called *artificial variables*, that will be selected as basic variables in the first iteration, and w is the sum of them. If $w = 0$ is obtained after the application of the simplex method, then $y_1 = y_2 = 0$ holds. If the optimal value of w is positive, then the original problem is infeasible.

Notice that to have a feasible basic form, the right side of the constraints must be nonnegative before introducing artificial variables.

Two-Phase Method:

1. Minimize w, the sum of all artificial variables. If the optimal value of w is positive, then halt (infeasible). Otherwise, go to Step 2.

2. (a) If the basis contains no artificial variables, then remove all columns for the artificial variables. Solve the problem with z-row as the objective function (*Phase II* problem).

(b) Suppose that the basis contains some artificial variable(s). Consider the equation corresponding to an artificial variable x_k:

$$x_k + \sum_{j \in J_1} \overline{a}_{kj} x_j + \sum_{j \in J_2} \overline{a}_{kj} x_j = 0 \tag{2.51}$$

where $x_j,\ j \in J_1$ are nonbasic variables that are not artificial, and $x_j,\ j \in J_2$ are nonbasic variables that are artificial. Notice that the right side of the above equation is always 0.

i. If some of $\overline{a}_{kj},\ j \in J_1$ is nonzero: Suppose $\overline{a}_{kt} \neq 0,\ t \in J_1$. Then the equation can be transformed into

$$x_t + \sum_{j \in J_1, j \neq t} (\overline{a}_{kj}/\overline{a}_{kt}) x_j + \sum_{j \in J_2} (\overline{a}_{kj}/\overline{a}_{kt}) x_j + (1/\overline{a}_{kt}) x_k = 0. \tag{2.52}$$

To make x_k nonbasic, remove variable x_t from all other constraints. By repeating this procedure, all artificial variables are moved to the nonbasis, or it becomes the following case ii. (Applying the pivot operation to the t-th row and the k-th column, artificial variable x_k is removed from the basis.)

ii. If $\overline{a}_{kj} = 0,\ \forall j \in J_1$: Equation (2.51) consists of artificial variables only, it can be eliminated.

Example 2.9. In (2.50), the initial basis are $\{y_1, y_2\}$ and they are represented by nonbasic variable as follows:

$$\begin{aligned} w &= y_1 + y_2 \\ &= (2 - 2x_1 - x_2 + x_3) + (6 - 4x_1 - 3x_2 + x_4) \\ &= 8 - 6x_1 - 4x_2 + x_3 + x_4 \end{aligned}$$

Then we obtain the following simplex tabular:

		x_1	x_2	x_3	x_4	y_1	y_2
w	8	6	4	-1	-1	0	0
z	0	-3	-2	0	0	0	0
y_1	2	[2]	1	-1	0	1	0
y_2	6	4	3	0	-1	0	1

		x_1	x_2	x_3	x_4	y_1	y_2
w	2	0	1	2	-1	-3	0
z	3	0	$-\frac{1}{2}$	$-\frac{3}{2}$	0	$\frac{3}{2}$	0
x_1	1	1	$\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	0
y_2	2	0	1	[2]	-1	-2	1

		x_1	x_2	x_3	x_4	y_1	y_2
w	0	0	0	0	0	-1	-1
z	$\frac{9}{2}$	0	$\frac{1}{4}$	0	$-\frac{3}{4}$	0	$\frac{3}{4}$
x_1	$\frac{3}{2}$	1	$\frac{3}{4}$	0	$-\frac{1}{4}$	0	$\frac{1}{4}$
x_3	1	0	$\frac{1}{2}$	1	$-\frac{1}{2}$	-1	$\frac{1}{2}$

Since all artificial variables become nonbasic variables, we remove them from the tabular (Case 2(a)) and restore the objective function.

		x_1	x_2	x_3	x_4
z	$\frac{9}{2}$	0	$\frac{1}{4}$	0	$-\frac{3}{4}$
x_1	$\frac{3}{2}$	1	$\frac{3}{4}$	0	$-\frac{1}{4}$
x_3	1	0	$\frac{1}{2}$	1	$-\frac{1}{2}$

Example 2.10.

Minimize

$$z, \quad z + x_1 + 3x_2 + 2x_3 = 0$$

Subject to

$$\begin{array}{llllcl} x_1 & +x_2 & +x_3 & & = & 10 \\ 2x_1 & +x_2 & +3x_3 & -x_4 & = & 20 \\ 3x_1 & +3x_2 & +2x_3 & +x_4 & = & 30 \\ x_1, & x_2, & x_3 & x_4 & \geq & 0 \end{array}$$

		x_1	x_2	x_3	x_4	y_1	y_2	y_3
w	60	6	5	6	0	0	0	0
z	0	1	3	2	0	0	0	0
y_1	10	[1]	1	1	0	1	0	0
y_2	20	2	1	3	−1	0	1	0
y_3	30	3	3	2	1	0	0	1

		x_1	x_2	x_3	x_4	y_1	y_2	y_3
w	0	0	−1	0	0	−6	0	0
z	−10	0	2	1	0	−1	0	0
x_1	10	1	1	1	0	1	0	0
y_2	0	0	−1	1	−1	−2	1	0
y_3	0	0	0	−1	1	−3	0	1

Artificial variables y_2, y_3 are still in the basis. Look at y_2-row. Since the coefficient of nonbasic variable x_3 is nonzero, we exchange the basic variable between y_2 and x_3 (Case 2(b)i).

		x_1	x_2	x_3	x_4	y_1	y_2	y_3
w	0	0	−1	0	0	−6	0	0
z	−10	0	3	0	1	1	−1	0
x_1	10	1	2	0	1	3	−1	0
x_3	0	0	−1	1	−1	−2	1	0
y_3	0	0	−1	0	0	−5	1	1

Look at y_3-row. Since the coefficient of nonbasic variable x_2 is nonzero, we exchange the basic variable between y_3 and x_2 (Case 2(b)i).

		x_1	x_2	x_3	x_4	y_1	y_2	y_3
w	0	0	0	0	0	−1	−1	−1
z	−10	0	0	0	1	−14	3	3
x_1	10	1	0	0	1	−7	1	2
x_3	0	0	0	1	−1	3	0	−1
x_2	0	0	1	0	0	5	−1	−1

All artificial variables are not in the basis and Phase I has finished. We remove artificial variables.

		x_1	x_2	x_3	x_4
z	-10	0	0	0	1
x_1	10	1	0	0	1
x_3	0	0	0	1	-1
x_2	0	0	1	0	0

Example 2.11. Solve the following problem using the two-phase method

$$\begin{array}{lrrcl}
\textbf{Maximize} & x_1 + x_2 & & & \\
\textbf{Subject to} & & & & \\
& -x_1 & +x_2 & \leq & 2 \\
& 2x_1 & +x_2 & \leq & 8 \\
& x_1 & +x_2 & \geq & 2 \\
& x_1, & x_2 & \geq & 0
\end{array}$$

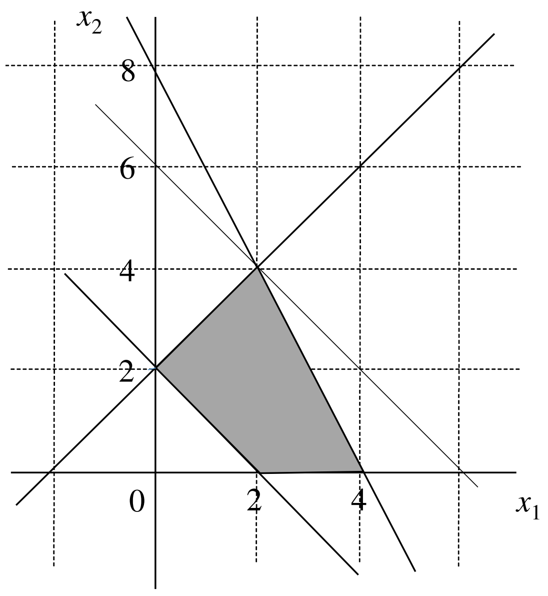


Figure 2.7: Example.

Standard form:

$$\begin{array}{lrrrrrcl}
\textbf{Minimize} & z, & & & & & & \\
& z + x_1 + x_2 = 0 & & & & & & \\
\textbf{Subject to} & & & & & & & \\
& -x_1 & +x_2 & +y_1 & & & = & 2 \\
& 2x_1 & +x_2 & & +y_2 & & = & 8 \\
& x_1 & +x_2 & & & -y_3 & = & 2 \\
& x_1, & x_2, & y_1, & y_2, & y_3 & \geq & 0
\end{array}$$

Two-Phase method:

$$\begin{array}{ll}
\textbf{Minimize} & w, \\
& w + x_1 + x_2 - y_3 = 2 \\
\textbf{Subject to} &
\end{array}$$

$$\begin{array}{rrrrrrrcl}
z & +x_1 & +x_2 & & & & & = & 0 \\
 & -x_1 & +x_2 & +y_1 & & & & = & 2 \\
 & 2x_1 & +x_2 & & +y_2 & & & = & 8 \\
 & x_1 & +x_2 & & & -y_3 & +y_4 & = & 2 \\
 & x_1, & x_2, & y_1, & y_2, & y_3, & y_4 & \geq & 0
\end{array}$$

Since slack variables y_1 and y_2 can be basic variables, we need to introduce only y_4 as an artificial variable.

		x_1	x_2	y_1	y_2	y_3	y_4
w	2	1	1	0	0	-1	0
z	0	1	1	0	0	0	0
y_1	2	-1	1	1	0	0	0
y_2	8	2	1	0	1	0	0
y_4	2	[1]	1	0	0	-1	1

		x_1	x_2	y_1	y_2	y_3	y_4
w	0	0	0	0	0	0	-1
z	-2	0	0	0	0	1	-1
y_1	4	0	2	1	0	-1	1
y_2	4	0	-1	0	1	2	-2
x_1	2	1	1	0	0	-1	1

		x_1	x_2	y_1	y_2	y_3
z	-2	0	0	0	0	1
y_1	4	0	2	1	0	-1
y_2	4	0	-1	0	1	[2]
x_1	2	1	1	0	0	-1

		x_1	x_2	y_1	y_2	y_3
z	-4	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0
y_1	6	0	$[\frac{3}{2}]$	1	$\frac{1}{2}$	0
y_3	2	0	$-\frac{1}{2}$	0	$\frac{1}{2}$	1
x_1	4	1	$\frac{1}{2}$	0	$\frac{1}{2}$	0

		x_1	x_2	y_1	y_2	y_3
z	-6	0	0	$-\frac{1}{3}$	$-\frac{2}{3}$	0
x_2	4	0	1	$\frac{2}{3}$	$\frac{1}{3}$	0
y_3	4	0	0	$\frac{1}{3}$	$\frac{2}{3}$	1
x_1	2	1	0	$-\frac{1}{3}$	$\frac{1}{3}$	0

Optimal solution: $x_1 = 2,\ x_2 = 4$.

2.7.3 Penalty Method

Under the constraints (**??**), we consider the following new objective function:

$$z' = r(y_1 + y_2) + 3x_1 + 2x_2$$

where r is a sufficiently large number. We can obtain an optimal solution by the simplex method. Therefore, two-phase method is not necessary.

r is not necessarily a concrete number if we evaluate inequalities according to the following rule:

$$g_1 < h_1 \Rightarrow g_1 r + g_2 < h_1 r + h_2.$$

$$\begin{array}{lrcl}
\textbf{Minimize} & -x_1 - x_2 & & \\
\textbf{subject to} & -x_1 + 2x_2 + x_3 & = & 4 \\
 & x_1 + 2x_2 - x_4 & = & 2 \\
 & x_1 + x_5 & = & 4 \\
 & x_2 + x_6 & = & 3 \\
 & x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6 & \geq & 0
\end{array} \tag{2.53}$$

Init:

		x_1	x_2	x_3	x_4	x_5	x_6
z	0	1	1	0	0	0	0
x_3	4	-1	2	1	0	0	0
x_4	-2	-1	-2	0	1	0	0
x_5	4	1	0	0	0	1	0
x_6	3	0	1	0	0	0	1

Step 1:

		x_1	x_2	x_3	x_4	x_5	x_6
z	-2	0	-1	0	1	0	0
x_1	2	1	2	0	-1	0	0
x_3	6	0	4	1	-1	0	0
x_5	2	0	-2	0	1	1	0
x_6	3	0	1	0	0	0	1

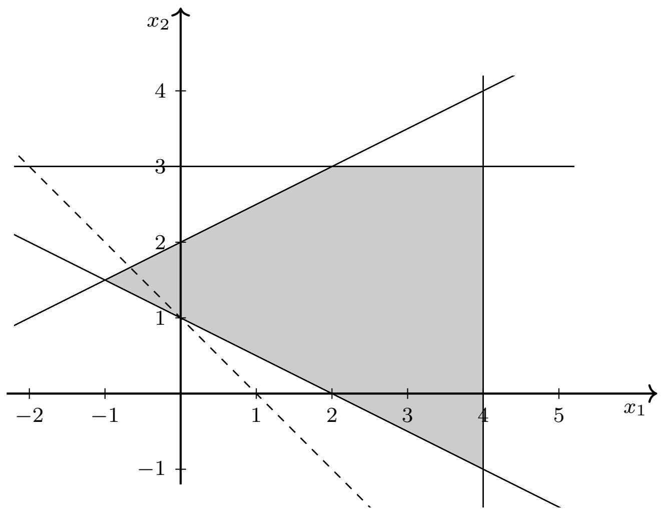
Step 2:

		x_1	x_2	x_3	x_4	x_5	x_6
z	-4	0	1	0	0	-1	0
x_1	4	1	0	0	0	1	0
x_3	8	0	2	1	0	1	0
x_4	2	0	-2	0	1	1	0
x_6	3	0	1	0	0	0	1

Step 3:

		x_1	x_2	x_3	x_4	x_5	x_6
z	-7	0	0	0	0	-1	-1
x_1	4	1	0	0	0	1	0
x_2	3	0	1	0	0	0	1
x_3	2	0	0	1	0	1	-2
x_4	8	0	0	0	1	1	2

Feasible region is shaded in the figure:



2.8 Change of Problem

Suppose that we have the optimal canonical form for some problem P. In some cases, it would be nice to know "nearby" problems P' for which the optimal canonical form is the same. Even if P and P' have different solutions, it should be easy to find the solution to P' using the optimal canonical form for P.

2.8.1 Change of b in Constraint

Consider changing the constraint term from $\mathbf{b}$ to $\mathbf{b}+\Delta\mathbf{b}$. In particular, consider a linear programming problem in standard form:

$$\begin{array}{lrcl} \textbf{Minimize} & & & \\ & \mathbf{c}^t\mathbf{x} & & \\ \textbf{Constraint} & & & \\ & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} \tag{2.54}$$

Let $\mathcal{B}$ be an optimal canonical basis for this problem. By Proposition 2.1, the corresponding optimal feasible solution $\mathbf{x}^*$ is $\mathbf{x}^*_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$ and $\mathbf{x}^*_{\mathcal{N}} = \mathbf{0}$.

Now consider the problem where $\mathbf{b}$ is replaced with $\mathbf{b} + \Delta\mathbf{b}$:

$$\begin{array}{lrcl} \textbf{Minimize} & & & \\ & \mathbf{c}^{\mathrm{t}}\mathbf{x} & & \\ \textbf{Subject to} & & & \\ & \mathbf{A}\mathbf{x} & = & \mathbf{b} + \Delta\mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} \tag{2.55}$$

Proposition 2.5. Problem (2.54) and problem (2.55) have the same optimal canonical form if:

$$\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} + \mathbf{A}_{\mathcal{B}}^{-1}\Delta\mathbf{b} \geq \mathbf{0}. \tag{2.56}$$

Proof The optimal solution to Problem (2.54) is $\mathbf{x}^* = (\mathbf{x}_{\mathcal{B}}, \mathbf{x}_{\mathcal{N}}) = (\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}, \mathbf{0})$. After changing $\mathbf{b}$ to $\mathbf{b} + \Delta\mathbf{b}$, the candidate for the optimal solution to Problem (2.54) is $(\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} + \mathbf{A}_{\mathcal{B}}^{-1}\Delta\mathbf{b}, \mathbf{0})$. The optimal solution must satisfy the non-negativity constraints, $\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} + \mathbf{A}_{\mathcal{B}}^{-1}\Delta\mathbf{b} \geq \mathbf{0}$ is required. Note that $\mathbf{c}'$ does not depend on $\mathbf{b}$, so after the change, the $c_i' \geq 0$. □

Example 2.12. The optimal basic form of Example 2.11 is

$$\begin{array}{ll} \textbf{Minimize} & z, \\ & z - \frac{1}{3}y_1 - \frac{2}{3}y_2 = -6 \\ \textbf{Subject to} & \end{array}$$

$$\begin{array}{cccccccc} & x_2 & +\frac{2}{3}y_1 & +\frac{1}{3}y_2 & & = & 4 \\ & & \frac{1}{3}y_1 & +\frac{2}{3}y_2 & +y_3 & = & 4 \\ x_1 & & -\frac{1}{3}y_1 & +\frac{1}{3}y_2 & & = & 2 \\ x_1, & x_2, & y_1, & y_2, & y_3 & \geq & 0 \end{array}$$

We try to find the range of α so that the above basic form is still optimal in the following modified problem:

$$\begin{array}{lrrcl} \textbf{Subject to} & & & & \\ & -x_1 & +x_2 & \leq & 2 + \alpha \\ & 2x_1 & +x_2 & \leq & 8 \\ & x_1 & +x_2 & \geq & 2 \\ & x_1 & x_2 & \geq & 0 \end{array}$$

Basic matrix:

$$B = \begin{array}{c} \begin{array}{ccc} x_2 & y_3 & x_1 \end{array} \\ \left[\begin{array}{ccc} 1 & 0 & -1 \\ 1 & 0 & 2 \\ 1 & -1 & 1 \end{array}\right] \end{array}, \quad B^{-1} = \left[\begin{array}{ccc} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{2}{3} & -1 \\ -\frac{1}{3} & \frac{1}{3} & 0 \end{array}\right].$$

$$\Delta\mathbf{b} = \left[\begin{array}{c} \alpha \\ 0 \\ 0 \end{array}\right], \quad \Delta\overline{\mathbf{b}} = B^{-1}\Delta\mathbf{b} = \left[\begin{array}{c} \frac{2}{3}\alpha \\ \frac{1}{3}\alpha \\ -\frac{1}{3}\alpha \end{array}\right].$$

$$\mathbf{b} = \left[\begin{array}{c} 2 \\ 8 \\ 2 \end{array}\right], \quad \overline{\mathbf{b}} = \left[\begin{array}{c} 4 \\ 4 \\ 2 \end{array}\right].$$

$$\overline{\mathbf{b}} + \Delta\overline{\mathbf{b}} = \begin{bmatrix} 4 + \frac{2}{3}\alpha \\ 4 + \frac{1}{3}\alpha \\ 2 - \frac{1}{3}\alpha \end{bmatrix} \geq \mathbf{0}.$$

Therefore,

$$-6 \leq \alpha \leq 6.$$

2.8.2 Change of c in Objective Function

Consider changing the objective function from min $\mathbf{c}^{\mathrm{t}}\mathbf{x}$ to $\min(\mathbf{c}^{\mathrm{t}} + \Delta\mathbf{c}^{\mathrm{t}})\mathbf{x}$. That is, the original problem (2.54) changes to:

$$\begin{array}{ll} \textbf{Minimize} & \\ & (\mathbf{c}^T + \Delta\mathbf{c}^T)\mathbf{x} \\ \textbf{Subject to} & \\ & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{2.57}$$

Proposition 2.6. Problem (2.54) and problem (2.57) have the same optimal canonical form if:

$$p_j + \Delta p_j \leq 0 \tag{2.58}$$

for $j = 1, 2, \ldots, n$, where

$$p_j = \mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{a}_j - c_j \tag{2.59}$$

$$\Delta p_j = \Delta\mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{a}_j - \Delta c_j, \tag{2.60}$$

and $\mathbf{a}_j$ is column j of $\mathbf{A}$.

Example 2.13. For some β, consider the following standard form LP:

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & -1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 2 \\ 8 \\ 2 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} -1-\beta \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The case of $\beta = 0$ was solved in Example 2.11; the optimal basis was found to be $\mathcal{B} = \{1, 2, 5\}$.

Apply the proposition taking $\mathbf{c} = [-1, -1, 0, 0, 0]^{\mathrm{t}}$ and $\Delta\mathbf{c} = [-\beta, 0, 0, 0, 0, 0]^{\mathrm{t}}$. Using

$$\mathbf{A}_{\mathcal{B}} = \begin{bmatrix} -1 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \text{ and } \mathbf{c}_{\mathcal{B}} = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} \text{ and } \Delta\mathbf{c}_{\mathcal{B}} = \begin{bmatrix} -\beta \\ 0 \\ 0 \end{bmatrix},$$

compute:

$$\begin{aligned}
p_1 &= 0 & \Delta p_1 &= 0 \\
p_2 &= 0 & \Delta p_2 &= 0 \\
p_3 &= -\frac{1}{3} & \Delta p_3 &= \frac{\beta}{3} \\
p_4 &= -\frac{2}{3} & \Delta p_4 &= -\frac{\beta}{3} \\
p_5 &= 0 & \Delta p_5 &= 0
\end{aligned}$$

Applying the p_3 and p_4 equalities to $p_j + \Delta p_j \leq 0$, the constraint on β is $-2 \leq \beta \leq 1$.

2.9 Exercises

2.1 Consider the following linear programming problem:

Maximize

$$2x_1 + x_2$$

Subject to

$$\begin{array}{rrl}
x_1 & +2x_2 & \geq 2 \\
x_1 & & \leq 3 \\
 & x_2 & \leq 2 \\
x_1 & +x_2 & \leq 4 \\
x_1, & x_2 & \geq 0
\end{array}$$

(a) Draw the feasible region on the x_1-x_2 plane.
(b) Convert the problem to standard form.
(c) Solve the problem by the simplex method.
(d) Suppose the objective function is parameterized by β:

$$\text{Maximize } (2 + \beta)x_1 + x_2$$

Find the range of β so that the optimal solution of the original problem is still optimal in the new problem.

2.2 Consider the following linear programming problem:

Maximize

$$x_1 + x_2$$

Subject to

$$\begin{array}{rrl}
-x_1 & +2x_2 & \leq 4 \\
x_1 & +2x_2 & \geq 2 \\
x_1 & & \leq 4 \\
 & x_2 & \leq 3 \\
x_1, & x_2 & \geq 0
\end{array}$$

(a) Draw the feasible region on the x_1-x_2 plane.
(b) Convert the problem to standard form.

(c) Solve the problem by the simplex method.

(d) Change the constraints as follows:

$$\begin{array}{rrl} -x_1 & +2x_2 & \leq 4 \\ x_1 & +2x_2 & \geq 2 \\ x_1 & & \leq 4+\alpha \\ & x_2 & \leq 3 \\ x_1, & x_2 & \geq 0 \end{array}$$

Find the range of α so that the basis for the optimal basic form in the original problem also gives the basis for the optimal basic problem in the new problem.

2.3 Consider the following program:

$$\begin{array}{rl} \textbf{Maximize} & x_1 + 3x_2 \\ \textbf{subject to} & -x + 2x_2 \leq 2 \\ & x_1 + 2x_2 \leq 6 \\ & x_1 + x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{array}$$

(a) Draw the feasible region in the x_1-x_2 plane

(b) Convert the problem to standard form

(c) Write the canonical form for the basis $\mathcal{B} = \{2, 3, 4\}$.

2.4

Maximize

$$2x_1 + x_2$$

Subject to

$$\begin{array}{rrl} -x_1 & +x_2 & \leq 2 \\ -x_1 & +x_2 & \geq -2 \\ x_1 & +x_2 & \leq 6 \\ & x_2 & \leq 3 \\ x_1, & x_2 & \geq 0 \end{array}$$

(a) Draw the feasible region on the $x_1 - x_2$ plane.

(b) Solve the problem by the simplex method.

(c) We change the objective function to "Maximize $\alpha x_1 + \beta x_2$", where $\alpha, \beta \in \{-1, 0, 1\}$. For each extreme point of the feasible region, find values of α and β for which the extreme point gives an optimal solution.

Chapter 3

Duality in Linear Programing

3.1 Dual Problem

This section includes the dual program, the weak duality theorem and the strong duality theorem.

3.1.1 Motivation

We begin by considering the motivation of the dual problem. In particular, we construct a lower bound on the objective function to be minimized. In doing so, we arrive at a new LP.

Consider an LP, given by:

$$\min \mathbf{c}^{\mathrm{t}}\mathbf{x}, \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq 0. \tag{3.1}$$

Assume that we have some $\mathbf{y}$ satisfying $\mathbf{y} \geq \mathbf{0}$ and $\mathbf{y}^{\mathrm{t}}\mathbf{A} \leq \mathbf{c}^{\mathrm{t}}$. Multiply the LP constraint by $\mathbf{y}^{\mathrm{t}}$:

$$\begin{aligned} \mathbf{A}\mathbf{x} &\geq \mathbf{b} && (3.2)\\ \mathbf{y}^{\mathrm{t}}\mathbf{A}\mathbf{x} &\geq \mathbf{y}^{\mathrm{t}}\mathbf{b} && (3.3)\\ 0 &\geq \mathbf{y}^{\mathrm{t}}\mathbf{b} - \mathbf{y}^{\mathrm{t}}\mathbf{A}\mathbf{x} && (3.4) \end{aligned}$$

This can be added to the objective function as:

$$\begin{aligned} z &= \mathbf{c}^{\mathrm{t}}\mathbf{x} && (3.5)\\ z &\geq \mathbf{c}^{\mathrm{t}}\mathbf{x} + \underbrace{\mathbf{y}^{\mathrm{t}}\mathbf{b} - \mathbf{y}^{\mathrm{t}}\mathbf{A}\mathbf{x}}_{\leq 0} && (3.6) \end{aligned}$$

Since $\mathbf{y}^{\mathrm{t}}\mathbf{A} \leq \mathbf{c}^{\mathrm{t}}$ we have that $\mathbf{c}^{\mathrm{t}} - \mathbf{y}^{\mathrm{t}}\mathbf{A} \geq 0$. If $\mathbf{x}$ is a feasible solution then $\mathbf{x} \geq \mathbf{0}$

and $(\mathbf{c}^{\mathrm{t}} - \mathbf{y}^{\mathrm{t}}\mathbf{A})\mathbf{x} \geq 0$. Continuing:

$$z \geq \underbrace{\mathbf{y}^{\mathrm{t}}\mathbf{b}}_{\text{LB}} + \underbrace{(\mathbf{c}^{\mathrm{t}} - \mathbf{y}^{\mathrm{t}}\mathbf{A})\mathbf{x}}_{\geq \mathbf{0}} \tag{3.7}$$

In (3.7) it can be seen that $\mathbf{y}^{\mathrm{t}}\mathbf{b}$ is a lower bound (LB) on the objective function. In order to make the lower bound as tight as possible, it should be maximized while observing the constraints of the problem. The best lower bound that can be obtained this way is the optimal value $\mathbf{y}$ of:

$$\max \mathbf{b}^{\mathrm{t}}\mathbf{y}, \mathbf{A}^{\mathrm{t}}\mathbf{y} \leq \mathbf{c}, \mathbf{y} \geq \mathbf{0}. \tag{3.8}$$

This is a new linear program in $\mathbf{y}$, rather than in $\mathbf{x}$.

3.1.2 Dual of the Standard Form Problem

Let P be a linear program called the *primal linear program*. The *dual linear program* D is another linear program derived from P in the following way:

- The objective function's c_i in the primal LP becomes the constraints' b_i in the dual LP;
- The constraints b_i in the primal LP becomes part of the objective function in the dual LP;
- The direction of the objective is reversed — maximum in the primal becomes minimum in the dual, or vice versa.

Wikipedia: Dual linear program

In particular, if the primal problem P is given by:

$$\begin{array}{ll} \textbf{Minimize} & \\ & z = \mathbf{c}^{\mathrm{t}}\mathbf{x} \\ \textbf{Subject to} & \\ & \begin{array}{rcl} \mathbf{Ax} & = & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{0} \end{array} \end{array} \tag{P}$$

Then the dual problem D is given by:

$$\begin{array}{ll} \textbf{Maximize} & \\ & w = \mathbf{b}^{\mathrm{t}}\mathbf{y} \\ \textbf{Subject to} & \\ & \mathbf{A}^{\mathrm{t}}\mathbf{y} \ \leq \ \mathbf{c} \end{array} \tag{D}$$

In the dual problem, there is no sign constraint on $\mathbf{y}$.

Example 3.1. Consider the following primal problem.

Minimize

$$z = 4x_1 + 3x_2$$

Subject to

$$\begin{array}{llcr} x_1 & +2x_2 & = & 2 \\ 12x_1 & +18x_2 & = & 19 \\ 6x_1 & +4x_2 & = & 7 \\ x_1, & x_2 & \geq & 0 \end{array}$$

Then, the dual problem is

Maximize

$$w = 2y_1 + 19y_2 + 7y_3$$

Subject to

$$\begin{array}{lllcr} y_1 & +12y_2 & +6y_3 & \leq & 4 \\ 2y_1 & +18y_2 & +4y_3 & \leq & 3 \end{array}$$

where variables y_i have no sign constraint.

Proposition 3.1. The dual of the dual problem is equivalent to the primal problem.

Proof. We first transform the dual problem to a standard form.

1. The vector $\mathbf{y}$ of variables with no sign constraint is represented by two nonnegative vectors of variables.

$$\mathbf{y} = \mathbf{y}_1 - \mathbf{y}_2.$$

2. We introduce vector $\mathbf{y}_3$ of slack variables to the constraints.

Then

Minimize

$$-w = -\mathbf{b}^T\mathbf{y}_1 + \mathbf{b}^T\mathbf{y}_2$$

Subject to

$$\begin{array}{lllcr} A^T\mathbf{y}_1 & -A^T\mathbf{y}_2 & +\mathbf{y}_3 & = & \mathbf{c} \\ \mathbf{y}_1, & \mathbf{y}_2, & \mathbf{y}_3 & \geq & 0 \end{array}$$

Let

$$\mathbf{x}' = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix}$$

$$\mathbf{c}'^T = [-\mathbf{b}^T, \mathbf{b}^T, \mathbf{0}^T]$$

$$\mathbf{b}' = \mathbf{c}$$

$$A' = [A^T, -A^T, I]$$

$$\lambda = \mathbf{y}$$

The problem is written as

$$\begin{array}{ll} \textbf{Maximize} & \\ & \mathbf{b}'^T\lambda \\ \textbf{Subject to} & \\ & A'^T\lambda \quad \leq \quad \mathbf{c}' \end{array}$$

Then the dual problem is obtained as follows:

$$\begin{array}{llll} \textbf{Maximize} & & & \\ & \mathbf{c}^T\lambda & & \\ \textbf{Subject to} & & & \\ & A\lambda & \leq & -\mathbf{b} \\ & -A\lambda & \leq & \mathbf{b} \\ & \lambda & \leq & \mathbf{0} \end{array}$$

That is,

$$\begin{array}{llll} \textbf{Minimize} & & & \\ & \mathbf{c}^T(-\lambda) & & \\ \textbf{Subject to} & & & \\ & A(-\lambda) & = & \mathbf{b} \\ & -\lambda & \geq & \mathbf{0} \end{array}$$

Let $\mathbf{x} = -\lambda$. Then the problem is equivalent to the primal problem. □

To obtain the dual problem in a non-standard form, we first transform it to a standard form and use the relationship between (P) and (D).

3.1.3 Weak Duality Theorem

The weak duality theorem states that the objective value of the dual LP at any feasible solution is always a bound on the objective of the primal LP at any feasible solution (upper or lower bound, depending on whether it is a maximization or minimization problem). In fact, this bounding property holds for the optimal values of the dual and primal LPs.

Proposition 3.2. Let $\mathbf{x}$ be any feasible solution of the primal problem (P), and let $\mathbf{y}$ be any feasible solution of the dual problem (D). Then:

$$\mathbf{b}^{\mathrm{t}}\mathbf{y} \leq \mathbf{c}^{\mathrm{t}}\mathbf{x}. \tag{3.9}$$

Proof. A feasible solution of (P) is $\mathbf{x}$. A feasible solution of (D) is $\mathbf{y}$, and $\mathbf{y}$ satisfies the constraints of (D), that is $\mathbf{y}$ satisfies $\mathbf{A}^{\mathrm{t}}\mathbf{y} \leq \mathbf{c}$. Then:

$$\begin{aligned} \mathbf{b} &= \mathbf{A}\mathbf{x} && \mathbf{x} \text{ satisfies constraint from (P)} \\ \mathbf{y}^{\mathrm{t}}\mathbf{b} &= \mathbf{y}^{\mathrm{t}}\mathbf{A}\mathbf{x} && \\ &= \left(\mathbf{A}^{\mathrm{t}}\mathbf{y}\right)^{\mathrm{t}}\mathbf{x} && \\ &\leq \mathbf{c}^{\mathrm{t}}\mathbf{x} && \mathbf{y} \text{ satisfies constraint from (D)} \end{aligned}$$

□

Proposition 3.3. If the primal problem (P) is unbounded, then the dual problem (D) is infeasible. If the dual problem (D) is unbounded, then the primal problem (P) is infeasible.

This table summarizes the relationship between types of solutions of the primal problem and the dual problem.

		Dual		
		optimal exists	unbounded	infeasible
Primal	optimal exists	Yes	No	No
	unbounded	No	No	Yes
	infeasible	No	Yes	Yes

In particular, note that it is possible that (P) and (D) are both infeasible.

3.1.4 Strong Duality Theorem

The strong duality theorem states that if the primal problem has an optimal solution then the dual problem has an optimal solution too, and the two optimal solutions are equal.

Proposition 3.4. If the primal problem (P) has an optimal solution, then the dual problem (D) also has an optimal solution. Moreover, the minimum value of z in (P) is equal to the maximum value of w in (D).

Proof. We use matrix representation. Let $\mathbf{x}$ be an optimal basic solution of (P).

Optimality condition:

$$\mathbf{c}_{\mathcal{N}}^{\mathrm{t}} - \mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}_{\mathcal{N}} \geq \mathbf{0} \tag{3.10}$$

Let $\lambda^{\mathrm{t}} = \mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{A}_{\mathcal{B}}^{-1}$. Then:

$$\mathbf{c}_{\mathcal{N}}^{\mathrm{t}} - \lambda^{\mathrm{t}}\mathbf{A}_{\mathcal{N}} \geq \mathbf{0} \tag{3.11}$$
$$\mathbf{A}_{\mathcal{N}}^{\mathrm{t}}\lambda \leq \mathbf{c}_{\mathcal{N}} \tag{3.12}$$

Moreover, since $\lambda^{\mathrm{t}} = \mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{A}_{\mathcal{B}}^{-1}$:

$$\mathbf{A}_{\mathcal{B}}^{\mathrm{t}}\lambda = \mathbf{c}_{\mathcal{B}}. \tag{3.13}$$

Since $\mathbf{A} = [\mathbf{A}_{\mathcal{B}}|\mathbf{A}_{\mathcal{N}}]$, $\mathbf{c}^{\mathrm{t}} = [\mathbf{c}_{\mathcal{B}}^{\mathrm{t}}|\mathbf{c}_{\mathcal{N}}^{\mathrm{t}}]$, it follows $\mathbf{A}^{\mathrm{t}}\lambda \leq \mathbf{c}$. Therefore λ is a feasible solution of (D). Since $\mathbf{x}$ is an optimal canonical solution of (P) we have

$$\mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} \tag{3.14}$$
$$\mathbf{x}_{\mathcal{N}} = \mathbf{0} \tag{3.15}$$

Therefore, it follows

$$z = \mathbf{c}^{\mathrm{t}}\mathbf{x} = \mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{x}_{\mathcal{B}} = \mathbf{c}_{\mathcal{B}}^{\mathrm{t}}\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} \tag{3.16}$$

Since $\lambda^t = \mathbf{c}_{\mathcal{B}}^t \mathbf{A}_{\mathcal{B}}^{-1}$:

$$\mathbf{c}_{\mathcal{B}}^t \mathbf{x}_{\mathcal{B}} = \mathbf{b}^t \lambda \tag{3.17}$$

Let $\mathbf{y}$ be any feasible solution of (D). Since $\mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$, it follows:

$$w = \mathbf{b}^t \mathbf{y} = \mathbf{x}_{\mathcal{B}}^t \mathbf{A}_{\mathcal{B}}^t \mathbf{y}. \tag{3.18}$$

Since $\mathbf{y}$ is a feasible solution of (D), $\mathbf{A}^t\mathbf{y} \le \mathbf{c}$ holds, and therefore $\mathbf{A}_{\mathcal{B}}^t \mathbf{y} \le \mathbf{c}_{\mathcal{B}}$. Then we have

$$\mathbf{b}^t \mathbf{y} \le \mathbf{x}_{\mathcal{B}}^t \mathbf{c}_{\mathcal{B}}^t \tag{3.19}$$

By (3.17), (3.19), it follows

$$\mathbf{b}^T \mathbf{y} \le \mathbf{b}^T \lambda.$$

This shows that λ is an optimal solution of the dual problem (D). $\square$

Proposition 3.5. Let $\mathbf{x}^*$ be a feasible solution of the primal problem (P), and let $\mathbf{y}^*$ be a feasible solution of the dual problem (D). If

$$\mathbf{b}^T \mathbf{y}^* = \mathbf{c}^T \mathbf{x}^*$$

then they are optimal solutions of the primal problem (P) and the dual problem (D), respectively.

3.2 Generalized Duality

In Section 3.1.1 a problem and its dual was given in the following form:

$$\begin{array}{rcr} \min \mathbf{c}^t\mathbf{x} & \qquad & \max \mathbf{b}^t\mathbf{y} \\ \mathbf{A}\mathbf{x} \ge \mathbf{b} & & \mathbf{A}^t\mathbf{y} \le \mathbf{c} \\ \mathbf{x} \ge \mathbf{0} & & \mathbf{y} \ge \mathbf{0} \end{array}$$

In Section 3.1.2 a problem and its dual was given in a slightly different form:

$$\begin{array}{rcr} \min \mathbf{c}^t\mathbf{x} & \qquad & \max \mathbf{b}^t\mathbf{y} \\ \mathbf{A}\mathbf{x} = \mathbf{b} & & \mathbf{A}^t\mathbf{y} \le \mathbf{c} \\ \mathbf{x} \ge \mathbf{0} & & \mathbf{y} \text{ is free} \end{array}$$

Both forms are valid, and these definitions can be generalized.

Consider that two problems $P_{\max}$ and $P_{\min}$ are duals, and they have general form:

$$\begin{array}{rcr} P_{\max} : \max \mathbf{c}^t\mathbf{x} & \qquad & P_{\min} : \min \mathbf{b}^t\mathbf{y} \\ \mathbf{A}\mathbf{x} \ ?\ \mathbf{b} & & \mathbf{A}^t\mathbf{y} \ ?\ \mathbf{c} \\ \mathbf{x} \ ?\ \mathbf{0} & & \mathbf{y} \ ?\ \mathbf{0}, \end{array}$$

where ? may be $\le, =$ or $\ge$ for the constraints and ? may be $\le, \ge$ or "free" for the variable. Table 3.1 shows 6 possible primal-dual pairs.

Table 3.1: Primal-dual pairs of problems. (Guenin, Konemann, Tuncel, page 143) .

	$P_{\max}$		$P_{\min}$	
	$\max \mathbf{c}^{\mathrm{t}}\mathbf{x}$		$\min \mathbf{b}^{\mathrm{t}}\mathbf{y}$	
	$\le$ constraint	$\leftrightarrow$	≥ 0 variable	
$\mathbf{Ax}\ ?\ \mathbf{b}$	$=$ constraint	$\leftrightarrow$	free variable	$\mathbf{y}\ ?\ 0$
	$\ge$ constraint	$\leftrightarrow$	≤ 0 variable	
	≥ 0 variable	$\leftrightarrow$	$\ge$ constraint	
$\mathbf{x}\ ?\ \mathbf{0}$	free variable	$\leftrightarrow$	$=$ constraint	$\mathbf{A}^{\mathrm{t}}\mathbf{y}\ ?\ \mathbf{c}$
	≤ 0 variable	$\leftrightarrow$	$\le$ constraint	

Example 3.2. Find the dual of the problem:

$$\begin{array}{rcl} & \min \mathbf{b}^{\mathrm{t}}\mathbf{y} & \\ \mathbf{A}^{\mathrm{t}}\mathbf{y} & \ge & \mathbf{c} \\ \mathbf{y} & \ge & \mathbf{0}. \end{array}$$

This is the minimization problem $P_{\min}$ with "$\ge$ constraint" and "≥ 0 variable." Referring to Table 3.1, these correspond to "$\le$ 0 variable" (line 6) and "$\le$ constraint" (line 1). The dual problem $P_{\max}$ is:

$$\begin{array}{rcl} & \max \mathbf{c}^{\mathrm{t}}\mathbf{x} & \\ \mathbf{Ax} & \le & \mathbf{b} \\ \mathbf{x} & \ge & \mathbf{0}. \end{array}$$

3.2.1 Strong Duality Example — With Solution

The following is an example of strong duality — the solution to a problem and its dual have the same optimal value.

Example 3.3. Recall an earlier problem:

$$\begin{array}{ll} \textbf{Minimize} & z = -x_1 - x_2 \\ \textbf{subject to} & x_1 + 2x_2 + x_3 = 3 \\ & x_1 \quad\quad + x_4 = 2 \\ & \quad x_2 \quad\quad + x_5 = 1 \\ & x_1\ ,\ x_2\ ,\ x_3\ ,\ x_4\ ,\ x_5 \ge 0 \end{array} \tag{3.20}$$

Referring to Table 3.1, the dual problem is given by:

$$\begin{array}{rl}\textbf{Maximize} & w = 3y_1 + 2y_2 + y_3 \\ \textbf{subject to} & y_1 + y_2 \leq -1 \\ & 2y_1 + y_3 \leq -1 \\ & y_1 \leq 0 \\ & y_2 \leq 0 \\ & y_3 \leq 0 \end{array} \tag{3.21}$$

Recall the earlier running example, here in matrix form:

$$\textbf{Minimize} \quad (-1,-1,0,0,0)(x_1,x_2,x_3,x_4,x_5)^{\mathrm{t}}$$

$$\textbf{subject to}$$

$$\begin{bmatrix}1&2&1&0&0\\1&0&0&1&0\\0&1&0&0&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{bmatrix} = \begin{bmatrix}3\\2\\1\end{bmatrix} \qquad \mathbf{x} \geq \mathbf{0} \tag{3.22}$$

It was found that optimal value of $z = -2.5$ is achieved with $\mathbf{x}^* = (2, 0.5, 0, 0, 0.5)$.

Referring to Table 3.1, the dual problem is given by:

$$\textbf{Maximize} \quad (3,2,1)(y_1,y_2,y_3)^{\mathrm{t}}$$

$$\textbf{subject to}$$

$$\begin{bmatrix}1&1&0\\2&0&1\\1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix} \leq \begin{bmatrix}-1\\-1\\0\\0\\0\end{bmatrix} \tag{3.23}$$

Following Table 3.1, $\mathbf{y}$ should be free. However, the dual constraints $\mathbf{A}^{\mathrm{t}}\mathbf{y} \leq \mathbf{c}$ happen to include $y_1, y_2, y_3 \leq 0$. The solution to the dual problem is $w = -2.5$, achieved with $\mathbf{y}^* = (-0.5, -0.5, 0)$. As the duality theorems indicate, the value of the objective functions are the same for the primal problem and the dual problem. However, the optimal solutions $\mathbf{x}$ and $\mathbf{y}$ are not the same.

3.3 Complementary Slackness Condition

The strong duality theorem told us we can get the optimal value of the dual problem w^* from the optimal value of the primal problem z^*. However, it does not tell us about the the optimal solution $\mathbf{y}^*$.

We can obtain an optimal solution of the dual problem using an optimal solution of the primal problem using the complementary slackness theorem.

Proposition 3.6. Let $\mathbf{x}$ be a feasible solution of the primal problem (P) and let $\mathbf{y}$ be a feasible solution of the dual problem (D). They are optimal solutions of (P) and (D), respectively, if and only if

$$x_j = 0 \quad \text{or} \quad \mathbf{a}_j^{\mathrm{t}}\mathbf{y} = c_j \quad \text{for } j = 1, 2, \ldots, n \tag{3.24}$$

$$y_i = 0 \quad \text{or} \quad \mathbf{A}_i\mathbf{x} = b_i \quad \text{for } i = 1, 2, \ldots, m \tag{3.25}$$

Here $\mathbf{A}_i$ is row i of $\mathbf{A}$ and $\mathbf{a}_j$ is column j of $\mathbf{A}$.

This proposition gives the *complementary slackness condition.*

Example 3.4. Consider the following primal-dual pair:

$$\begin{aligned}&\textbf{Maximize} \quad (5,3,5)(x_1,x_2,x_3)^{\mathrm{t}}\\ &\textbf{subject to}\\ &\begin{bmatrix}1 & 2 & -1\\ 3 & 1 & 2\\ -1 & 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} \leq \begin{bmatrix}2\\4\\-1\end{bmatrix}\end{aligned} \tag{3.26}$$

and

$$\begin{aligned}&\textbf{Minimize} \quad (2,4,-1)(y_1,y_2,y_3)^{\mathrm{t}}\\ &\textbf{subject to}\\ &\begin{bmatrix}1 & 3 & -1\\ 2 & 1 & 1\\ -1 & 2 & 1\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix} = \begin{bmatrix}5\\3\\5\end{bmatrix}\\ &\qquad\qquad\qquad \mathbf{y} \geq \mathbf{0}\end{aligned} \tag{3.27}$$

The complementary slackness conditions (3.25) are:

$$\begin{aligned} y_1 &= 0 \quad \text{or} \quad (1,2,-1)(x_1,x_2,x_3) = 2\\ y_2 &= 0 \quad \text{or} \quad (3,1,2)(x_1,x_2,x_3) = 4\\ y_3 &= 0 \quad \text{or} \quad (-1,1,1)(x_1,x_2,x_3) = -1 \end{aligned}$$

Example 3.5. Using the complementary slackness condition, find an optimal solution of the dual problem of Example 3.3 using its primal solution.

The complementary slackness conditions (3.24) $x_j = 0$ or $\mathbf{A}_j\mathbf{y} = c_j$ for $j = 1, 2, \ldots, n$ are:

$$\begin{aligned} x_1 = 2 &\quad \text{or} \quad \boxed{(1,1,0)(y_1,y_2,y_3)^{\mathrm{t}} = -1}\\ x_2 = 0.5 &\quad \text{or} \quad \boxed{(2,0,1)(y_1,y_2,y_3)^{\mathrm{t}} = -1}\\ \boxed{x_3 = 0} &\quad \text{or} \quad (1,0,0)(y_1,y_2,y_3)^{\mathrm{t}} = 0\\ \boxed{x_4 = 0} &\quad \text{or} \quad (0,1,0)(y_1,y_2,y_3)^{\mathrm{t}} = 0\\ x_5 = 0.5 &\quad \text{or} \quad \boxed{(0,0,1)(y_1,y_2,y_3)^{\mathrm{t}} = 0} \end{aligned}$$

At least one of the two conditions must be satisfied — a box indicates the condition which is satisfied. Using the three equations in y_1, y_2, y_3 which are satisfied, we can solve this system of equations:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \\ 0 \end{bmatrix}. \tag{3.28}$$

Thus, the solution of the dual problem is $(y_1, y_2, y_3) = (-0.5, -0.5, 0)$ and the value is $w = -2.5$.

Proof. This proof considers (3.24) only. Including the "or" condition, (3.24) can be written as $(\mathbf{A}^t\mathbf{y} - \mathbf{c})^t\mathbf{x} = \mathbf{0}$.

Sufficiency:

$$\mathbf{c}^T\mathbf{x} = (A^T\mathbf{y})^T\mathbf{x},$$

$$\mathbf{c}^T\mathbf{x} = \mathbf{y}^T A\mathbf{x}.$$

Since $A\mathbf{x} = \mathbf{b}$, we have

$$\mathbf{c}^T\mathbf{x} = \mathbf{y}^T\mathbf{b} = \mathbf{b}^T\mathbf{y}.$$

From Corollary3.5, $\mathbf{x}$ and $\mathbf{y}$ are optimal solutions of (P) and (D), respectively.

Necessity:

Suppose $\mathbf{x}$ and $\mathbf{y}$ are optimal solution of (P) and (D), respectively. From the duality theorem,

$$\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}.$$

Since $A\mathbf{x} = \mathbf{b}$, we have

$$\mathbf{c}^T\mathbf{x} = \mathbf{y}^T\mathbf{b} = \mathbf{y}^T A\mathbf{x}.$$

Therefore,

$$(A^T\mathbf{y} - \mathbf{c})^T\mathbf{x} = \mathbf{0}.$$

□

Let $A = [\mathbf{a}_1, \cdots, \mathbf{a}_n]$. The complementary slackness condition means that for each $j = 1, \cdots, n$:

$$(\mathbf{y}^T\mathbf{a}_j - c_j)x_j = 0.$$

Since

$$\mathbf{y}^T\mathbf{a}_j \leq c_j, \; x_j \geq 0,$$

the complementary slackness condition requires at least one of the above two inequalities holds as an equality.

3.4 Farkas' Lemma

Suppose that you want to prove if a given system of equalities is infeasible.

Proposition 3.7. Let $\mathbf{A}$ be an $m \times n$ matrix and let $\mathbf{b}$ be an m vector. Then exactly one of these statements are true:

1. $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ has a solution
2. there exists a vector $\mathbf{y}$ such that $\mathbf{A}^{\mathrm{t}}\mathbf{y} \geq \mathbf{0}$ and $\mathbf{b}^{\mathrm{t}}\mathbf{y} < \mathbf{0}$.

Suppose that there is a vector $\mathbf{y}$ which satisfies Statement 2. By Farkas' lemma, Statement 1 cannot be true, and thus any corresponding linear program does not have a solution. This $\mathbf{y}$ is a "certificate" or proof that this problem does not have a feasible solution.

To prove, consider the following special case:

Primal Problem (P)

$$\begin{array}{ll} \textbf{Minimize} & \\ & z = \mathbf{0}^T\mathbf{x} \\ \textbf{Subject to} & \\ & \begin{array}{rcl} A\mathbf{x} & = & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{0} \end{array} \end{array} \tag{3.29}$$

Dual Problem (D)

$$\begin{array}{ll} \textbf{Maximize} & \\ & w = \mathbf{b}^T\mathbf{y} \\ \textbf{Subject to} & \\ & A^T\mathbf{y} \leq \mathbf{0} \end{array} \tag{3.30}$$

If (P) has a feasible solution, then it is an optimal solution and the optimal value of the objective function is $z^* = 0$. By the weak duality theorem, the following holds for any feasible solution $\mathbf{y}$ of (D):

$$\mathbf{b}^T\mathbf{y} \leq \mathbf{0}.$$

Conversely, (D) has a feasible solution $\mathbf{y} = \mathbf{0}$. Therefore, if $\mathbf{b}^T\mathbf{y} \leq \mathbf{0}$ holds for any feasible solution $\mathbf{y}$ of (D), then (D) has an optimal solution $\mathbf{y}^* = \mathbf{0}$. By the duality theorem, (P) has an optimal solution, too.

As a result, we can conclude that $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ has a solution if and only if $\mathbf{b}^T\mathbf{y} \leq \mathbf{0}$ holds for any $\mathbf{y}$ that satisfies $A^T\mathbf{y} \leq \mathbf{0}$:

By replacing $\mathbf{y}$ with $-\mathbf{y}$, we obtain the following, which is rephrase Farkas' lemma in a different way (Theorem of Alternatives):

Proposition 3.8. $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ has a solution if and only if $\mathbf{b}^T\mathbf{y} \geq \mathbf{0}$ holds for any $\mathbf{y}$ that satisfies $A^T\mathbf{y} \geq \mathbf{0}$. □

3.5 Exercises

3.1

Maximize

$$z = \frac{1}{2}x_1 + x_2$$

Subject to

$$\begin{array}{rrl} 2x_1 & +x_2 & \geq 1 \\ x_1 & +x_2 & \leq 4 \\ -x_1 & +x_2 & \leq 2 \end{array}$$

(Remark that each variable has no constraint on its sign.)

(a) Make the dual problem.

(b) Find all basic solutions of the dual problem. By comparing them, find the optimal value of the objective function for the primal problem.

3.2 Make the dual problem of the following linear programming problem in a matrix form:

(a)

Minimize

$$\mathbf{c}^T\mathbf{x}$$

Subject to

$$\begin{array}{rl} \mathbf{Ax} & \leq \mathbf{b} \\ \mathbf{x} & \geq \mathbf{0} \end{array}$$

(b)

Maximize

$$\mathbf{c}^T\mathbf{x}$$

Subject to

$$\begin{array}{rl} \mathbf{Ax} & \geq \mathbf{b} \\ \mathbf{x} & \geq \mathbf{0} \end{array}$$

(c)

Minimize

$$\mathbf{c}^T\mathbf{x}$$

Subject to

$$\begin{array}{rl} \mathbf{A}_1\mathbf{x} & \geq \mathbf{b}_1 \\ \mathbf{A}_2\mathbf{x} & = \mathbf{b}_2 \\ \mathbf{x} & \geq \mathbf{0} \end{array}$$

3.3

Maximize

$$x_1 + 2x_2 + 3x_3 + 4x_4$$

Subject to

$$\begin{array}{l} 4x_1 + 3x_2 + 2x_3 + x_4 \leq 10 \\ x_1,\ x_2,\ x_3,\ x_4 \geq 0 \end{array}$$

(a) Make the standard form problem and find all basic feasible solutions.

(b) Make the dual problem of 1 and solve it.

(c) Using the complementary slackness condition together with the result in 2, find an optimal solution of the primal problem.

3.4

Minimize

$$4x_1 - 3x_2 + 2x_3 - x_4$$

Subject to

$$\begin{aligned} &x_1 + 2x_2 + 3x_3 + 4x_4 \le 10 \\ &-x_1 + x_2 - x_3 + x_4 \le 10 \\ &x_1,\ x_2,\ x_3,\ x_4 \ge 0 \end{aligned}$$

(a) Solve the problem by the simplex method.

(b) Find the dual linear program.

(c) Using the complementary slackness condition, find an optimal solution of the dual problem.

3.5 Explain the importance of *Farkas's Lemma.*

Chapter 4

Network Optimization

4.1 Graph Theory Notation

- A *graph* is a pair $G = (\mathcal{V}, \mathcal{E})$, where V is a set of *vertices* or *nodes* and E is a set of *edges*. Wikipedia: Graph
- If all edges have a direction, then G is called a *directed graph*. If no edge has a direction, then G is called an *undirected graph*. An example of a directed graph and an undirected graph are in Fig. 4.1.
- An edge of a directed graph is an ordered pair (i, j) of vertices, where vertex i is called the *initial vertex* and j is called the *terminal vertex*, for $i \neq j$. An edge of an undirected graph is an unordered pair $\{i, j\}$ for $i \neq j$ where vertices i, j are called *end vertices*.
- A route from some vertex to another vertex following the direction of each edge is called *a walk*. If parallel edges (more than one edges with the same initial and terminal vertices) do not exist, then a walk is specified by a sequence of vertices. In a walk $(v_1, v_2, \cdots, v_n)$, we call v_1 the initial vertex and v_n the terminal vertex.
- If a walk does not go through each edge more than once, it is called *a path*. Wikipedia: Path
- A *weighted graph* or *network* is a graph where a number (or weight) is assigned to each edge.

4.2 Shortest Path Problem

4.2.1 Definition

On a weighted graph or network, the weight of a walk is the sum of the weights on each edge of the walk. Given two vertices on a graph, the *shortest path problem* finds the walk with the minimum weight. Wikipedia: Shortest path problem

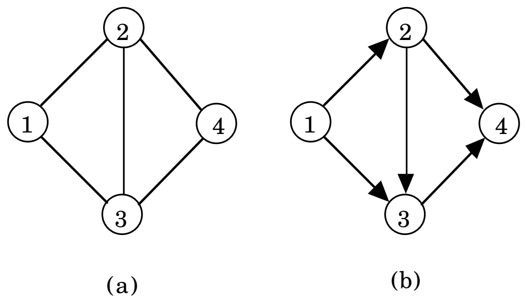


Figure 4.1: (a) An undirected graph and (b) a directed graph.

- Any walk with the smallest length contains no cycles with positive length.
- If a cycle with negative length exists, then we may have a walk with an infinitely small length, i.e., the solution is unbounded. For this reason, we can focus on paths, and not walks.

Example 4.1. Consider the network shown in Fig. 4.2. We study a problem of finding a walk with the smallest length from vertex 1 to vertex 4.

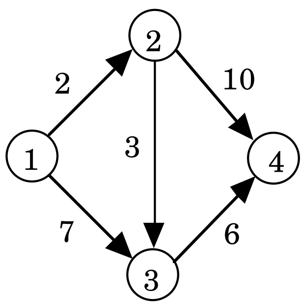


Figure 4.2: A network for solving the shortest path problem.

4.2.2 Dijkstra's Algorithm

Dijkstra's algorithm is a method to find the shortest path in a network. The method is applicable only to networks with non-negative weights. Dijkstra's algorithm finds the shortest path from a specified source to not only one specified destination node, but to all nodes in the network.

Wikipedia: Dijkstra Algorithm

Each node i has a label u_i; the label is a number. The label u_i represents a distance from the start node to node i. Initially, the start node has label

$u_1 = 0$, meaning the distance from the start node to itself is 0. For all other nodes, initially, the labels are temporary and initialized with the value ∞. As the algorithm progresses, a temporary label value may change. When the label changes from temporary to permanent, then its value will no longer change. A permanent label is denoted u_i^*.

For the network, the weight on the edge (i, j) is d_{ji}.

Dijkstra's Algorithm

1. *Initialize* The start vertex 1 begins with (temporary) label $u_1 = 0$. All other label values are $u_j = \infty$, $j \neq 1$.

2. Let i be a vertex with the smallest temporary label, call i the *current vertex.* For all edges (i, j): if vertex j does not have a permanent label and

   $$u_j > u_i^* + d_{ij},$$

   then update the temporary label with

   $$u_j = u_i^* + d_{ij}. \tag{4.1}$$

3. For the current vertex i, change the temporary label u_i to a permanent label u_i^*.

4. If all nodes have permanent labels, then output all $u_1^*, u_2^*, \ldots$ as the minimal distances. Otherwise, go to Step 2.

Each permanent label u_i^* indicates the length of a shortest path from vertex 1 to vertex i. Note that in one iteration, a current vertex whose temporary label with the shortest distance is selected, and its label becomes permanent. In addition, the neighbors of the current vertex are updated, if the distance through the current vertex is lower than the previously found distance.

The shortest path is obtained by adding the following procedure to the algorithm. When a temporary label is updated and becomes $u_j = u_i^* + d_{ij}$, the algorithm remembers the link from i to j. Denote this link by $\pi(j) = i$. That is, $\pi(j)$ is the neighbor i of j which has the lowest weight. Using these π, we can construct a directed tree from vertex 1 to all other vertices, which indicates a shortest path from vertex 1 to other vertices.

Example 4.2. Apply the algorithm to the network in Fig. 4.2.

	u				π			
	1	2	3	4	1	2	3	4
0	0	∞	∞	∞	-			
1		$\underline{2}$	7	∞	-	1	1	
2			$\underline{5}$	12	-	1	2	2
3				$\underline{11}$	-	1	2	3

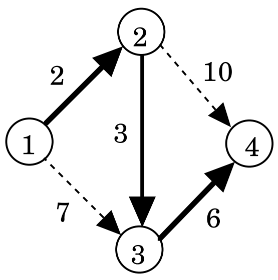


Figure 4.3: The directed tree indicating the shortest path.

Underlined numbers indicate selected vertices at each step. The obtained directed tree is shown in Fig. 4.3.

Shortest path has length 11. Shortest path to node 4 is $4 \leftarrow \pi(4) = 3 \leftarrow \pi(3) = 2 \leftarrow \pi(2) = 1$.

4.2.3 Correctness of Dijkstra's Algorithm

A proof of the correctness of Dijkstra's algorithm is given. Given that all existing permanent labels are correctly decided, show that for a node i not yet decided, its permanent label u_i^* correctly indicates the length of a shortest path from vertex 1 to vertex i.

Proof. Suppose Step 2 has finished. Let $\mathcal{P}$ be the set of vertices with permanent labels, and let $\mathcal{T}$ be the set of vertices with temporary labels. Let ℓ_j denote the length of a shortest path from vertex 1 to vertex j. The inductive assumption is that for all $j \in \mathcal{P}$:

$$u_j^* = \ell_j,$$

that is, the permanent labels were correctly decided.

Now, suppose vertex i is selected in Step 3, i.e.,

$$u_i = \min_{j \in \mathcal{T}} u_j.$$

For a shortest path p from vertex 1 to vertex j, consider two possible cases:

1. A shortest path p goes through only vertices of $\mathcal{P}$ (except i). Since

$$u_i = \min_{j \in \mathcal{P}} u_j^* + d_{ji}$$

the shortest path is given in the above formula, and so $u_i = \ell_i$. Therefore, at Step 3 $u_i^* = \ell_i$ holds and the permanent label was correctly decided.

2. Shortest path p goes through one or more vertices of $\mathcal{T}$ other than i. Let k be the first vertex of $\mathcal{T}$ that appears in p. By the argument for Case 1, it follows that $u_k = \ell_k$. Now let $d(k,i) > 0$ denote the length of a shortest path from vertex k to vertex i. Then
$$\ell_i = \ell_k + d(k,i).$$
Since $u_i \geq \ell_i$, $u_k = \ell_k$, $d(k,i) \geq 0$,
$$\begin{aligned} u_i &\geq \ell_i \\ &= \ell_k + d(k,i) \\ &> u_k. \end{aligned}$$
Since $u_i > u_k$, this contradicts the assumption that node i was selected in Step 3, because the selected node must have the smallest temporary label.

In other words, Case 2 is not possible, and the node selected to change from temporary to permanent will always have the minimum label, that is, $u_i^* = \ell_i$. □

This is a proof by mathematical induction — given a partial set of permanent labels are correctly decided, then the next step is also correctly decided.

4.2.4 Bellman-Ford Algorithm

The Bellman-Ford algorithm also finds the shortest path in a network, but can be applied to networks where some edges have a negative distance. Wikipedia: Bellman-Ford Algorithm

Similar to Dijkstra's algorithm, the Bellman-Ford algorithm maintains a list of labels that indicates the current best distances. A distinction with Dijkstra's algorithm is that the Bellman-Ford algorithm performs all computations in parallel.

The Bellman-Ford algorithm maintains a list $\mathcal{S}^{(k)}$ of nodes that were updated on iteration k. Since these are the only nodes with new labels, these are used on iteration $k+1$ to reduce the amount of computation needed.

Bellman-Ford:

1. *Initialize* Label the start vertex 1 $u_1^{(0)} = 0$, and label other vertices $u_j^{(0)} = \infty$, for $j \neq 1$ Let $\mathcal{S}^{(0)} = \{1\}$ and $k = 1$.

2. *Update neighbors of* $\mathcal{S}^{(k-1)}$ For each $i \in \mathcal{S}^{(k-1)}$ and each edge (i,j), find the minimum value of
$$u_i^{(k-1)} + d_{ij} \text{ for } i \in S^{(k-1)}, (i,j) \in \mathcal{E}.$$
Suppose $u_{i^*}^{(k-1)} + d_{i^*j}$ is the minimum value. If it is less than $u_j^{(k-1)}$, then let $u_j^{(k)} = u_{i^*}^{k-1} + d_{i^*j}$, otherwise let $u_j^{(k)} = u_j^{(k-1)}$. For other j, let $u_j^{(k)} = u_j^{(k-1)}$. Record the link as $\pi(j) = i^*$.

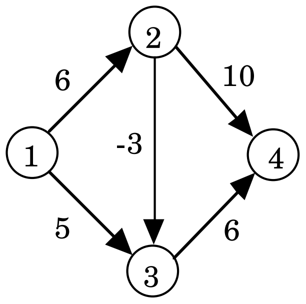


Figure 4.4: A network with a negative cycle.

3. If no labels change, then halt.
4. Let $\mathcal{S}^{(k)}$ be the set of vertices whose labels were updated. Let $k \leftarrow k+1$ and go to Step 2.

If the network has no negative cycles, then the algorithm terminates within at most n iterations, where n is the number of vertices.

Example 4.3. Apply the Bellman-Ford algorithm to the network in Fig. 4.4.

	u				π				
k	1	2	3	4	1	2	3	4	$\mathcal{S}^{(k)}$
0	0	∞	∞	∞	-				1
1	0	6	5	∞	-	1	1		2, 3
2	0	6	3	11	-	1	2	3	3, 4
3	0	6	3	9	-	1	2	3	4
4	0	6	3	9	-	1	2	3	-

The shortest path is $(1, 2, 3, 4)$.

4.3 Maximum Flow Problem

4.3.1 Definition

This section considers a *flow network*, a graphical network where each edge has a capacity and along each edge some amount of flow can be sent, but not exceeding the capacity. The network has two special nodes, called the *source* and the *sink*. This network has various applications: sending packets over a computer network, routing goods in a transportation network, or oil in a network of pipelines. Wikipedia: Flow Network

This section considers finding a maximum amount of flow that can be sent from the source to the sink. This problem is called the maximum flow problem. While the problem can be solved by linear programing, the Ford-Fulkerson method can also be used to find it. The maximum-flow, minimum-cut theorem gives the maximum value for the flow. Wikipedia: Maximum Flow Problem

For a flow network, let $(\mathcal{V}, \mathcal{E})$ be a graph and let $c_{i\to j} \geq 0$ denote the *capacity* from node i to node j. Then $x_{i\to j}$ denotes the *flow* from node i to node j. The total flow from the source node s to sink node t is v, called the *flow value*. Then, the flow $x_{i\to j} \geq 0$ should observe the following rules:

- Flow cannot exceed capacity: $x_{i\to j} \leq c_{i\to j}$
- Flow balance: the flow into a node q matches the flow out of node q:

$$\sum_{i\in\mathcal{S}} c_{i\to q} = \sum_{j\in\mathcal{T}} c_{q\to j}$$

where unconnected nodes have flow 0. Does not apply to sink or source nodes.

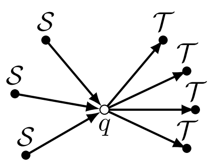


Figure 4.5: Example of flow balance, the flow into node is equal to the flow out of the node.

- Flow from source s matches flow to sink t, and is equal to v:

$$\sum_{i\in\mathcal{V}} x_{s\to i} = \sum_{j\in\mathcal{V}} x_{j\to t} = v$$

Example 4.4. The network shown in Fig. 4.6 has four nodes. The source s is node 1, and the sink t is node 4. The capacity for each edge is shown, for example $c_{1\to 2} = 9$.

The path $1 \to 2 \to 3 \to 4$ can have flow of 7, because 7 is the minimum capacity on that path. An additional flow of 2 can be added on the path $1 \to 2 \to 4$. Together the flow from s to t is 9. However, this is not the maximum possible flow.

For such a network, two questions arise: what is the maximum possible flow, and how to find it? The maximum possible flow is given by the max-flow, min-cut theorem in the next subsection. The maximum flow may be found using linear programing as the following example shows. However, the Ford-Fulkerson method given in Subsection 4.3.3 is generally more efficient.

Example 4.5. Consider the flow network in Fig. 4.6 with source 1 and sink 4.

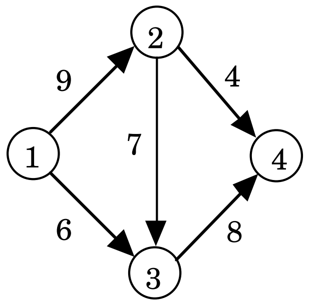


Figure 4.6: Example of a flow network where each edge has a capacity. The source node s is 1, the sink node t is 4.

Then the problem may be formulated as a linear program:

$$
\begin{array}{rl}
\textbf{Maximize} & v \\[1ex]
\textbf{subject to} & x_{12} + x_{13} = v \\
& x_{24} + x_{34} = v \\
& x_{12} - x_{23} - x_{24} = 0 \\
& x_{13} + x_{23} - x_{34} = 0 \\
& x_{12} \le 9 \\
& x_{13} \le 6 \\
& x_{23} \le 7 \\
& x_{24} \le 4 \\
& x_{34} \le 8 \\
& x_{12}\ ,\ x_{13}\ ,\ x_{23}\ ,\ x_{24}\ ,\ x_{34} \ge 0
\end{array}
$$

A possible solution of this LP is:

$$(x_{12}, x_{13}, x_{23}, x_{24}, x_{34}, v) = (6, 6, 2, 4, 8, 12)$$

The maximum flow is $v = 12$.

4.3.2 Max-Flow Min-Cut Theorem

The capacity of a network is the maximum flow on that network. The max-flow min-cut theorem states how to find the capacity of a flow network. A cut partitions a network into two sets of nodes, and the capacity of a cut is the sum of the capacities between the two sets. The capacity of the network is the minimum capacity of all possible cuts. Wikipedia: Max-flow min-cut theorem

Definition 4.1. For a network $G = (\mathcal{V}, \mathcal{E})$ let $\mathcal{S}$ and $\mathcal{T}$ be a partition of $\mathcal{V}$. A *cut* denoted $(\mathcal{S}, \mathcal{T})$ is

$$(\mathcal{S}, \mathcal{T}) = \{(i,j) \in \mathcal{E} \mid i \in \mathcal{S}, j \in \mathcal{T}\} \tag{4.2}$$

That is, a cut $(\mathcal{S}, \mathcal{T})$ is the set of edges (i,j) such that $i \in \mathcal{S}$ and $j \in \mathcal{T}$.

For a flow network with source s and sink t, assume that $s \in \mathcal{S}$ and $t \in \mathcal{T}$.

Definition 4.2. For a flow network $G = (\mathcal{V}, \mathcal{E})$ with capacities c_{ij}, let $\mathcal{S}$ and $\mathcal{T}$ be a partition of $\mathcal{V}$. The *capacity of cut* $(\mathcal{S}, \mathcal{T})$ is:

$$c(\mathcal{S}, \mathcal{T}) = \sum_{(i,j) \in (\mathcal{S}, \mathcal{T})} c_{ij}. \tag{4.3}$$

The capacity of a cut is the sum of the capacity of the edges in a cut.

Example 4.6. Find the capacity of all possible cuts of the flow network shown in Fig. 4.6, with source $s = 1$ and sink $t = 4$. There are four ways to partition the nodes into $(\mathcal{S}, \mathcal{T})$. The partitions and their corresponding capacities are:

$$\begin{aligned}
(\mathcal{S}, \mathcal{T}) &= (\{1\}, \{2,3,4\}) & c(\mathcal{S}, \mathcal{T}) &= 9 + 6 = 15 \\
&= (\{1,2\}, \{3,4\}) & &= 6 + 7 + 4 = 17 \\
&= (\{1,3\}, \{2,4\}) & &= 9 + 8 = 17 \\
&= (\{1,2,3\}, \{4\}) & &= 4 + 8 = 12
\end{aligned}$$

Note that when $\mathcal{S} = \{1, 3\}$ the edge $2 \to 3$ is flowing into $\mathcal{S}$, and is not included.

Proposition 4.1. In a flow network, the value of any flow v is upper bounded by the capacity of any cut.

Proof Consider any feasible flow, and let the value of the flow be v. We want to prove $v \leq c(\mathcal{S}, \mathcal{T})$ for any feasible flow v. For any cut $(\mathcal{S}, \mathcal{T})$:

$$v = \sum_{(i,j) \in (\mathcal{S}, \mathcal{T})} x_{ij} - \sum_{(j,i) \in (\mathcal{T}, \mathcal{S})} x_{ji}. \tag{4.4}$$

Note that all cuts have the same flow.

We can show $v \leq c(\mathcal{S}, \mathcal{T})$ as follows:

$$\begin{aligned}
v &\leq \sum_{(i,j) \in (\mathcal{S}, \mathcal{T})} x_{ij} & & x_{ij} \geq 0 \text{ in } (4.4) \\
&\leq \sum_{(i,j) \in (\mathcal{S}, \mathcal{T})} c_{ij} & & x_{ij} \leq c_{ij} \\
&= c(\mathcal{S}, \mathcal{T}) & & \text{definition of capacity of a cut}
\end{aligned}$$

Now let $(\mathcal{S}^*, \mathcal{T}^*)$ be a cut with minimum capacity, called the *minimum cut*, so clearly:

$$v \leq c(\mathcal{S}^*, \mathcal{T}^*) \tag{4.5}$$

holds. □

Clearly, the maximum flow is upper bounded by the minimum of the capacities of all cuts. That is, the minimum cut gives an upper bound of flow values. In fact, this upper bound is achievable.

Proposition 4.2. *Max-flow min-cut theorem* The maximum value of an s-t flow is equal to the minimum capacity over all s-t cuts.

Example 4.7. Continuing Example 4.6, use the max-flow min-cut theorem to find the maximum value of the $1 \to 4$ flow.

Since minimum of the four capacities in Example 4.6 is 12, we have $(\mathcal{S}^*, \mathcal{T}^*) = 12$. This agrees with the capacity found in Example 4.5.

Proof. Let $\{x_{ij}\}$ be a maximum flow. Let $\mathcal{S}$ be the set of vertices to which an augmenting path from the source s exists, and let $\mathcal{T}$ be the set of other vertices. Since the residual network for a maximum flow does not contain any augmenting paths to the sink t, t is in $\mathcal{T}$. From the definition of $\mathcal{S}$, we have

$$x_{ij} = c_{ij}, \quad (i, j) \in (\mathcal{S}, \mathcal{T}).$$

Moreover,

$$x_{ji} = 0, \quad (j, i) \in (\mathcal{T}, \mathcal{S}).$$

Then, it follows that

$$v^* = \sum_{(i,j)\in(\mathcal{S},\mathcal{T})} x_{ij} - \sum_{(j,i)\in(\mathcal{T},\mathcal{S})} x_{ji} = \sum_{(i,j)\in(\mathcal{S},\mathcal{T})} c_{ij} = c(\mathcal{S}, \mathcal{T}).$$

Since for any cut $(\mathcal{S}', \mathcal{T}')$,

$$v^* \leq c(\mathcal{S}', \mathcal{T}'),$$

Therefore $(\mathcal{S}, \mathcal{T})$ is a minimum cut. □

4.3.3 Ford-Fulkerson Method

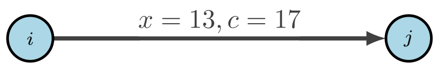
The Ford-Fulkerson method finds maximum flow for a given network. In this method, as long as there is a path from the source node to the to the sink node with available capacity, then additional flow is sent along this path. A path with available capacity is called an augmenting path. This available capacity is removed to form a residual network. These steps repeat until there are no augmenting paths available. Wikipedia: Ford-Fulkerson Method

Residual Network

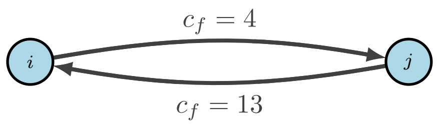
Given a network G and a flow x, a *residual network* G_f is a network that shows how much the flow can be changed with respect to x. A label c_f on the residual network indicates how much additional flow is available. G_f has the same set of vertices as G, but the edges may be added and the capacities are different. The value c_f indicates the margin to increase the flow, and is called the residual capacity.

In the Ford-Fulkerson method, it may be necessary to *decrease* flow on certain edges. In order to represent the possible decrease of flow on an edge $i \to j$, the residual network G_f adds a new edge $j \to i$ which is not in the original graph G.

Consider for example, an edge which has capacity 17, but already it has flow 13, shown in the following figure:



This edge has residual capacity of 4, that is, it could accept 4 more units of flow. In the residual network, this is represented by the following:



Now there is a reverse link with capacity $c_f = 11$. If the algorithm requires flow on the edge $j \to i$, this could be obtained by cancelling flow on the edge $i \to j$. This is equivalent to decreasing the flow $i \to j$.

More formally, if the flow to be added to $i \to j$ is $\Delta > 0$, then the *residual capacity* is updated as:

$$c_f(e) = \begin{cases} c(e) - \Delta & \text{if } e = (i \to j) \\ c(e) + \Delta & \text{if } e = (j \to i) \end{cases} \tag{4.6}$$

The important idea is that if there is a flow of value Δ, then we can cancel it by adding the same amount of flow in the reverse direction.

Augmenting Path

An *augmenting path* is any path $\mathcal{P}$ on the residual network G_f from source s to the sink t with positive flow. The flow on $\mathcal{P}$ is the minimum of the residual capacities:

$$\Delta = \min\{c(e), e \in \mathcal{P}\}$$

If $\mathcal{P}$ is the selected augmenting path, then we can add at most Δ along the following amount of flow along the path. We can update the residual capacities using (4.6).

Ford-Fulkerson Method

With the above definition of an augmenting path and how it can be used to update the residual capacities, the Ford-Fulkerson method is stated as:

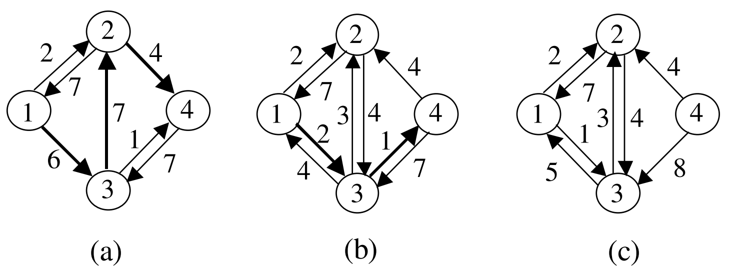


Figure 4.7: Residual network.

Ford-Fulkerson Method As long as an augmenting path exists, select one and add as much flow along it as possible.

The method terminates when there is no feasible flow from s to t. After termination, for any edge $i \to j$ in the original graph, the flow $x_{i\to j}$ is given by:

$$x_{i\to j} = \begin{cases} c_{i\to j} - c_f(i \to j) & \text{if } c_{i\to j} \geq c_f(i \to j) \\ 0 & \text{otherwise} \end{cases}$$

If the original graph G has no anti-parallel edges, then the flow for $i \to j$ is the residual capacity $c_f(j \to i)$ for the reverse link (an antiparallel edge are parallel edges, but with opposite directions). The above does not state *how* to select the augmenting path, so we call it a "method" rather than an algorithm.

Example 4.8. Apply the Ford-Fulkerson method to the network in Fig. 4.6.

1. The first residual network is the original flow network, as in Fig. 4.6. Select augmenting path $(1, 2, 3, 4)$ and add $\min\{9, 7, 8\} = 7$ units of flow along it. The resulting residual network is in Fig. 4.7-(a).

2. Select augmenting path $(1, 3, 2, 4)$ and add $\min\{6, 7, 4\} = 4$ units of flow along it. The resulting residual network is in Fig. 4.7(b).

3. Select augmenting path $(1, 3, 4)$ and add $\min\{2, 1\} = 1$ unit of flow along it. The resulting residual network is in Fig. 4.7-(c).

4. The resulting residual network in Fig. 4.7-(c) has no augmenting path, i.e. there is no flow from s to t. A maximum flow is shown in Fig. 4.8, where the reverse residual capacity is the flow on each each edge:

$$(x_{12}, x_{13}, x_{23}, x_{24}, x_{34}, v) = (7, 5, 3, 4, 8).$$

   This solution differs from the linear programing result in Example 4.5, but both algorithms give the same maximum flow of $v = 12$.

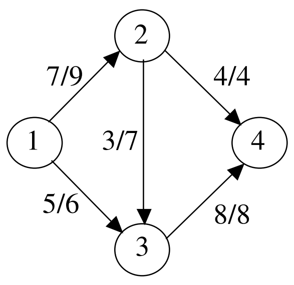


Figure 4.8: Maximum flow.

4.4 Minimum Cost Flow Problem

4.4.1 Definition

Consider the flow network in which each edge (i, j) has a capacity c_{ij} and also a cost d_{ij} for adding one unit of flow along this edge. Given a feasible flow value v, we study a problem of finding a flow with flow value v such that the total cost is minimum in all such flows. This problem is called *the minimum cost flow problem*.

4.4.2 Busaker-Gowen Method

We define the length of an augmenting path as follows:

- for each edge (i, j) with $c_{ij} - x_{ij} > 0$, the length is d_{ij};
- for each edge (i, j) with $x_{ij} > 0$, the length is $-d_{ij}$.

Busaker-Gowen:
Repeat the following 1, 2 until the flow value reaches v.

1. Find a shortest augmenting path p.
2. Compute $\Delta(p)$ and add $\Delta(p)$ units of flow along the path. (If the flow value exceeds v by this addition, then add a flow so that the total flow value is equal to v).

Example 12: Consider the network in Fig. 4.9. Given a flow value $v = 7$, we find a minimum cost flow from vertex 1 to vertex 4.

1. The initial residual network is the same as the network in Fig. 4.9. There are three augmenting paths.
    - $(1, 2, 4)$ with length 11.
    - $(1, 2, 3, 4)$ with length 8.

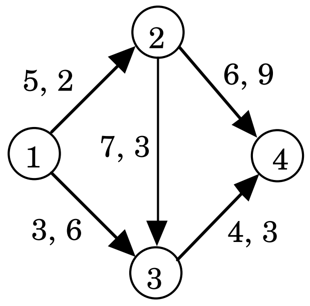


Figure 4.9: Minimum cost flow problem.

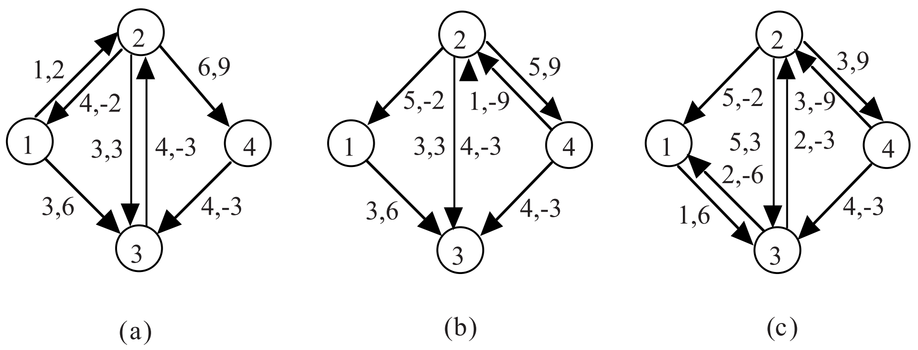


Figure 4.10: Residual networks.

- $(1, 3, 4)$ with length 9.

Select $(1, 2, 3, 4)$ and add $min\{5, 7, 4\} = 4$ units of flow.

2. The residual network is Fig. 4.10(a). There are two augmenting paths.
   - $(1, 2, 4)$ with length 11.
   - $(1, 3, 2, 4)$ with length 12.

   Select $(1, 2, 4)$ and add $min\{1, 6\} = 1$ unit of flow.

3. The residual network is Fig. 4.10(b). There is one augmenting path $(1, 3, 2, 4)$ and $min\{3, 4, 5\} = 3$. If we add 3 units of flow, then the total flow value exceeds 7. Therefore, we add 2 units.

4. The minimum cost flow is shown in 4.11, where the label on each edge (i, j) indicates x_{ij}/c_{ij}.

4.5 Exercises

4.1 Compare the computational cost between the Dijkstra method and the Ford method.

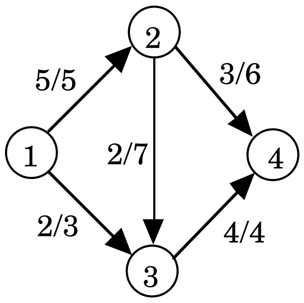
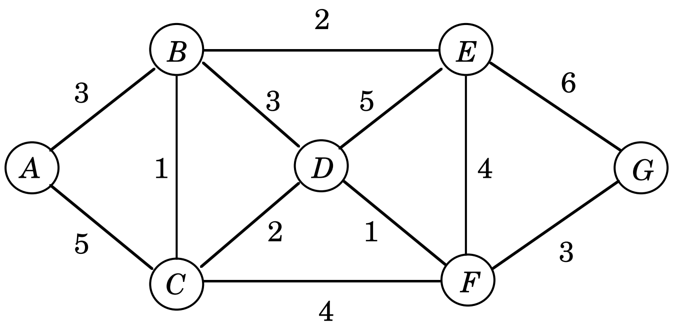
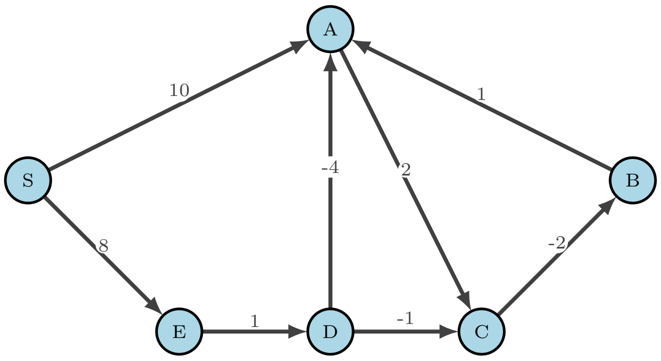


Figure 4.11: Minimum cost flow.

4.2 Using Dijkstra's algorithm, find the shortest path from Node A to Node G, for the network below.



4.3 Using the Bellman-Ford algorithm, find the shortest path from the start node S to other nodes A to E, for the network below.

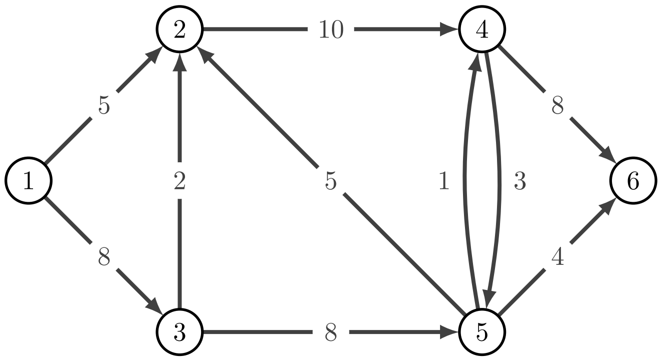


4.4 Solve Problem 3 in Chapter 1 by Ford-Fulkerson method.

4.5 Consider the maximum flow problem for the flow network shown below, where the source is vertex 1 and the sink is vertex 6.

(a) Formulate this problem as a linear programming problem.

(b) Solve the maximum flow problem by the Ford-Fulkerson method.



4.6 Consider the minimum cost flow problem for the network given in Fig. **??** and Fig. 4.12, where the source is vertex 1, the sink is vertex 6, and the flow value 10. Find a minimum cost flow by the Busaker-Gowen method.

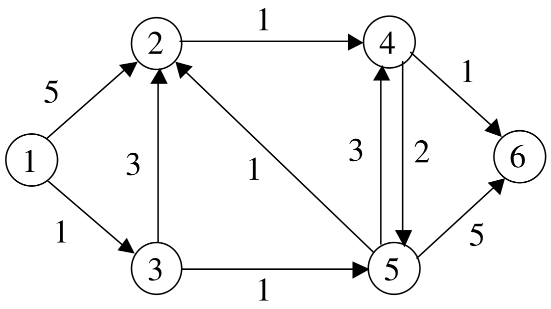


Figure 4.12: Costs.

4.7 Show a network having negative weight(s) for which the Dijkstra's algorithm cannot give the shortest path.

4.8 Consider the maximum flow problem for the network in Fig. 4.13, where the source is vertex 1 and the sink is vertex 4. Find the minimum and maximum number of iterations in the execution of Ford-Fulkerson method. Note that the number of iterations is the number of times augmenting paths are added.

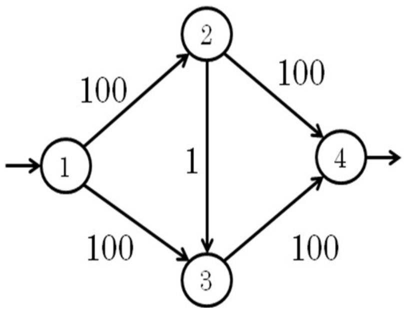


Figure 4.13: A network.

Chapter 5

Nonlinear Programming

5.1 Mathematical Background

5.1.1 Convex Sets and Convex Functions

Definition 5.1. Let $\mathcal{D}$ be a subset of $\mathbb{R}^n$. Then $\mathcal{D}$ is a *convex set* if for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$:

$$\lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in \mathcal{D}, \tag{5.1}$$

for all $0 \leq \lambda \leq 1$.

A set which is not convex is called non-convex. Fig. 5.1 shows examples of a convex set and a non-convex set. Wikipedia: Convex Set

On the other hand, a set $\mathcal{S} \subset \mathbb{R}^n$ satisfies

$$\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x}) = (1-\alpha)\mathbf{x} + \alpha\mathbf{y} \in \boldsymbol{S} \tag{5.2}$$

for all $\mathbf{x}, \mathbf{y} \in \boldsymbol{S}$ and $\alpha \in [0, 1]$, then the set $\boldsymbol{S}$ is called "a convex set".

It is important to distinguish between convex *sets* and convex *functions*. A real-valued function $f(\cdot)$ defined on an interval is called convex (or convex

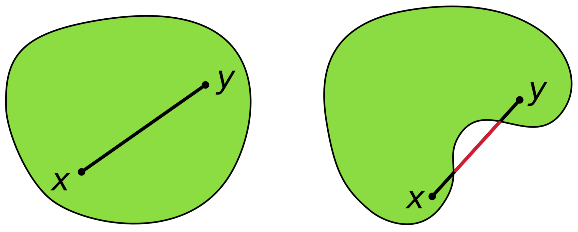


Figure 5.1: Left: a convex set. Right: a non-convex set. Image credit: Wikipedia/-Convex set.

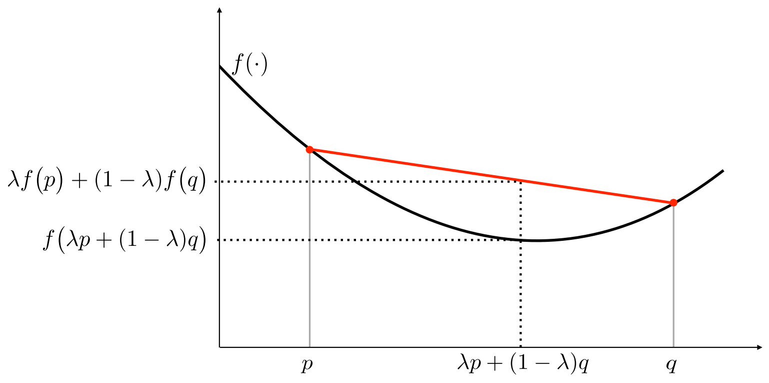

$f(\cdot)$
$\lambda f(p) + (1-\lambda)f(q)$
$f(\lambda p + (1-\lambda)q)$
p
$\lambda p + (1-\lambda)q$
q

Figure 5.2: A convex function.

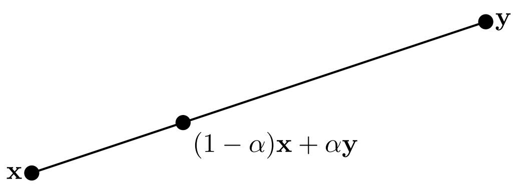


Figure 5.3: For $0 \le \alpha \le 1$, a point on the line segment connecting two points $\mathbf{x}$ and $\mathbf{y}$ is $(1-\alpha)\mathbf{x} + \alpha\mathbf{y}$.

downward or lower convex) if the line segment between any two points $f(\mathbf{p})$ and $f(\mathbf{q})$ on the graph of the function lies above the graph. Examples of convex functions are the quadratic function $f(x) = x^2$ and the exponential function $f(x) = e^x$, for $x \in \mathbb{R}$. A convex function is illustrated in Fig. 5.2. Wikipedia: Convex Function

Definition 5.2. Let f be a real-valued function with domain $\mathcal{D} \subseteq \mathbb{R}^n$. The function f is a *convex function* if for any, $\mathbf{p}, \mathbf{q} \in \mathcal{D}$:

$$f\big(\lambda\mathbf{p} + (1-\lambda)\mathbf{q}\big) \le \lambda f(\mathbf{p}) + (1-\lambda)f(\mathbf{q}), \tag{5.3}$$

for $0 \le \lambda \le 1$.

The function is called *strictly convex* if

$$f\big(\lambda\mathbf{p} + (1-\lambda)\mathbf{q}\big) < \lambda f(\mathbf{p}) + (1-\lambda)f(\mathbf{q}), \tag{5.4}$$

for any $0 \le \lambda \le 1$ and $\mathbf{p} \neq \mathbf{q}$. A function f is said to be (strictly) concave if $-f$ is (strictly) convex.

5.1.2 Differentiability

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be differentiable at $\mathbf{x} \in \mathbb{R}^n$ if

$$f(\mathbf{x} + \boldsymbol{a}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^{\mathrm{t}}\boldsymbol{a} + o(||\boldsymbol{a}||) \tag{5.5}$$

holds for all $\boldsymbol{a} \in \mathbb{R}^n$, where $||\boldsymbol{a}||$ denotes the Euclidean Norm of $\boldsymbol{a}$;

$$||\boldsymbol{a}|| = \sqrt{\boldsymbol{a}^{\mathrm{t}}\boldsymbol{a}},$$

and $o(t)$ denotes a function which satisfies

$$\lim_{t \to 0} \frac{o(t)}{t} = 0.$$

A function f is said to be twice differentiable at $\mathbf{x} \in \mathbb{R}^n$ if

$$f(\mathbf{x} + \boldsymbol{a}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^{\mathrm{t}}\boldsymbol{a} + \frac{1}{2}\boldsymbol{a}^{\mathrm{t}}\nabla^2 f(\mathbf{x})\boldsymbol{a} + o(||\boldsymbol{a}||^2) \tag{5.6}$$

holds for all $\boldsymbol{a} \in \mathbb{R}^n$. A function f is said to be twice differentiable on a set $\mathcal{X} \subset \mathbb{R}^n$, if f is twice differentiable at every point on $\mathcal{X}$.

5.1.3 Gradient Vector and Hessian Matrix

Definition 5.3. The *gradient* $\nabla f(\mathbf{x})$ of a scalar-valued differentiable function f at the point $\mathbf{x}$ is:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix} \tag{5.7}$$

The gradient is the vector field (or vector-valued function) ∇f. At a point $\mathbf{x}$, the gradient $\nabla f(\mathbf{x})$ is the direction of fastest increase. At the point $\mathbf{x}$, the value of $f(\mathbf{x})$ increases most rapidly in the direction parallel with $\nabla f(\mathbf{x})$, that is, when $\mathbf{v} = \nabla f(\mathbf{x})/||\nabla f(\mathbf{x})||$, see Fig. 5.4. The value of f does not change in the direction orthogonal to $\nabla f(\mathbf{x})$, that is, when $\nabla f(\mathbf{x})^{\mathrm{t}}\mathbf{v} = 0$. Wikipedia: Gradient

The *Hessian matrix* or Hessian is a square matrix of second-order partial derivatives of a function f. It describes the local curvature of a function of many variables. Wikipedia: Hessian matrix

The Hessian $\nabla^2 f(\mathbf{x})$ at a point $\mathbf{x}$ is defined as:

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix} \tag{5.8}$$

The Hessian matrix is a symmetrical matrix.

If the gradient $\nabla f(\mathbf{x})$ is continuous with respect to $\mathbf{x}$, f is called *continuously differentiable.* Similarly, if the Hessian matrix $\nabla^2 f(\mathbf{x})$ is continuous with respect to $\mathbf{x}$, then f is called *continuously twice differentiable.*

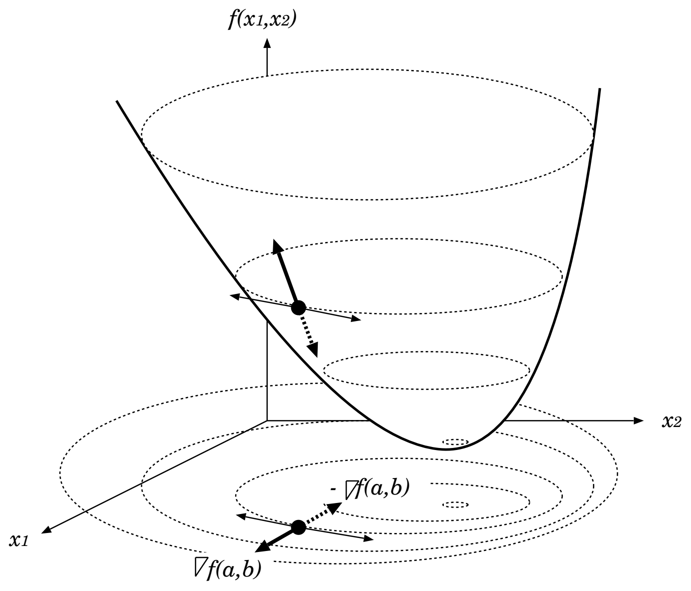


Figure 5.4: Gradient vector.

Example 5.1. For the function $f(x_1, x_2) = x_1^2 + x_2^3$, the gradient and Hessian matrix are:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 3x_2^2 \end{bmatrix} \text{ and } \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 6x_2 \end{bmatrix}.$$

5.1.4 Eigenvalues and Eigenvectors

Definition 5.4. Let $\mathbf{A}$ be an $n \times n$ matrix. Then $\mathbf{v}$ is an *eigenvector* of $\mathbf{A}$ if:

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v} \tag{5.9}$$

where λ is a scalar called the *eigenvalue*.

From the definition, Hessian matrix $\nabla^2 f(\mathbf{x})$ is a symmetric matrix.

Proposition 5.1. The eigenvalues of a real symmetric matrix are all real.

The eigenvalues can be found as follows. Equation (5.9) written as $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{v} = \mathbf{0}$ has a nonzero solution if and only if $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$. Here $\det(\mathbf{A} - \lambda \mathbf{I}_n)$ is a degree-n polynomial:

$$\det(\mathbf{A} - \lambda \mathbf{I}_n) = (\lambda_1 - \lambda)(\lambda_2 - \lambda) \cdots (\lambda_n - \lambda),$$

where the n roots $\lambda_1, \ldots, \lambda_n$ are the n eigenvalues. This polynomial is called the characteristic equation of $\mathbf{A}$. The eigenvalues can be found by finding the roots of the characteristic polynomial. Wikipedia: Eigenvalues and eigenvectors.

Example 5.2. Find the eigenvalues of:

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

Write the characteristic polynomial:

$$\det(\mathbf{A} - \lambda \mathbf{I}_n) = \det \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} = \lambda^2 - 4\lambda + 3.$$

The roots, and thus the eigenvalues, are $\lambda_1 = 3$ and $\lambda_2 = 1$.

5.1.5 Definiteness of Matrices

Definition 5.5. For a real symmetric matrix $\mathbf{A}$, the *quadratic form* of $\mathbf{A}$ is:

$$\mathbf{x}^t \mathbf{A} \mathbf{x}. \tag{5.10}$$

Definition 5.6. A symmetric matrix $\mathbf{A}$ is *positive semi-definite* if

$$\mathbf{x}^t \mathbf{A} \mathbf{x} \geq 0$$

for all $\mathbf{x} \in \mathbb{R}^n$. Similarly, we have:

$$\begin{aligned} \text{positive definite: } & \mathbf{x}^t \mathbf{A} \mathbf{x} > 0 \\ \text{negative semi-definite: } & \mathbf{x}^t \mathbf{A} \mathbf{x} \leq 0 \\ \text{negative definite: } & \mathbf{x}^t \mathbf{A} \mathbf{x} < 0 \end{aligned}$$

Proposition 5.2. Let $\mathbf{A}$ be an $n \times n$ real symmetric matrix (more generally, Hermitian matrices). The real eigenvalues of $\mathbf{A}$ are real, and their sign characterize its definiteness:

- $\mathbf{A}$ is positive definite if and only if all of its eigenvalues are positive.
- $\mathbf{A}$ is positive semi-definite if and only if all of its eigenvalues are non-negative.
- $\mathbf{A}$ is negative definite if and only if all of its eigenvalues are negative
- $\mathbf{A}$ is negative semi-definite if and only if all of its eigenvalues are non-positive.
- $\mathbf{A}$ is indefinite if and only if it has both positive and negative eigenvalues.

Example 5.3. Continuing Example **??**, the matrix $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ has eigenvalues $\lambda_1 = 3$ and $\lambda_2 = 1$. Since both eigenvalues are positive, $\mathbf{A}$ is both positive definite and positive semi-definite.

To determine definiteness of a matrix, besides computing the eigenvalues, the leading principal minors can be used to test for positive definiteness and the principal minors can be used to test for positive semidefiniteness. Wikipedia: Sylvester's criterion

A *principle minor* of an $n \times n$ matrix $\mathbf{A}$ is the determinant of $r \times r$ matrix which is composed of symmetrically selected r rows and r columns from $\mathbf{A}$ (that is, if the ith row is selected, then the ith column is selected also), and their relative orders in row and in column follow the original orders in $\mathbf{A}$. A leading principal minor of a $n \times n$ matrix $\mathbf{A}$ is the determinant of $r \times r$ matrix which is the upper left r-by-r corner of $\mathbf{A}$, where $1 \le r \le n$.

Proposition 5.3 (Sylvester's criterion)**.** A symmetric matrix $\mathbf{A}$ is positive definite if and only if all its leading principal minors are positive.

Proposition 5.4. A symmetric matrix $\mathbf{A}$ is positive semi-definite if and only if all its principal minors are nonnegative.

5.1.6 Exercises

1. Are each of the following matrices positive definite, positive semi-definite, or neither?

   (a)
$$\begin{bmatrix} 3 & 4 \\ 4 & 5 \end{bmatrix}$$

   (b)
$$\begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 0 \\ 3 & 0 & 5 \end{bmatrix}$$

2. Find the range of k such that $\mathbf{A}$ is positive definite.
$$\mathbf{A} = \begin{bmatrix} 6 & k \\ k & 2 \end{bmatrix}$$

3. Let $\mathbf{Q}$ be an $n \times n$ symmetric positive semi-definite matrix, and let $\mathbf{x}$ be a n dimensional vector. Show that the function
$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{t}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{t}}\mathbf{x}$$
   is convex.

5.2 Nonlinear Programming

5.2.1 Definitions

Linear programming problem is characterized by the linearity of both objective function and constraint function. If one or both of the objective function and the constraint function are nonlinear with respect to variables, it is called nonlinear programming problem.

A general statement of the optimization problem is:

$$\begin{aligned} &\textbf{Minimize} && f(\mathbf{x}) \\ &\textbf{subject to} && h_i(\mathbf{x}) = 0, \;\; i = 1, 2, \ldots, \ell \\ & && g_j(\mathbf{x}) \leq 0, \;\; j = 1, 2, \ldots, m \end{aligned} \tag{5.15}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x} = (x_1, \cdots, x_n)^{\mathrm{t}}$. Unless otherwise noted, it is assumed that $\mathbf{x} \in \mathbb{R}^n$. The $h_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R} = 0$ are called the *equality constraints* and the $g_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R} \leq 0$ are called the *inequality constraints.*

The following $\mathcal{S}$ is called the *feasible region.*

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n | h_i(\mathbf{x}) = 0 \text{ for all } i \in \{1, \cdots, \ell\}, g_j(\mathbf{x}) \leq 0 \text{ for all } j \in \{1, \cdots, m\}\}$$

$\mathbf{x}$ is called *feasible solution* when $\mathbf{x} \in \mathcal{S}$. A problem with $\mathcal{S} = \mathbb{R}^n$ is called an *unconstrained optimization problem.*

If $\mathbf{x}^* \in \mathcal{S}$ and $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{S}$, then $\mathbf{x}^*$ is called an an *optimum solution* or a *globally optimum solution*, and the value of $f(\mathbf{x}^*)$ is called the *optimum value.*

5.2.2 Locally Optimal Solution

Let $\mathcal{B}_n(\mathbf{x}, r)$ be a sphere with its center at $\mathbf{x}$ and its radius r in n-dimensional Euclidean space, that is,

$$\mathcal{B}_n(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^n | \; ||\mathbf{x} - \mathbf{y}|| \leq r\}.$$

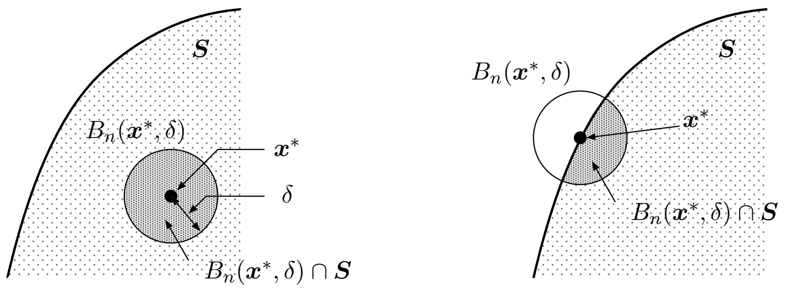
Definition 5.7. A feasible point $\mathbf{x}^* \in \mathcal{S}$ is called a *local optimum* or *local minimum* if there exists $\delta > 0$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \text{ for all } \mathbf{x} \in \big(\mathcal{S} \cap \mathcal{B}_n(\mathbf{x}^*, \delta)\big).$$

Fig. 5.5 illustrates local optimality for the cases $\mathcal{B}$ is entirely inside of $\mathcal{S}$ and partially inside $\mathcal{S}$. Any globally optimal solution is a locally optimal solution. In general, a nonlinear programming problem may have multiple local optimum points. Fig. 5.6 illustrates such an example with $n = 1$.

Comparing a nonlinear programming problem with a linear programming problem, we can say the following (see also Fig. 5.7).

1. The boundary of each constraint is a curved surface in general, while it is a flat surface for a linear programming problem.



$\boldsymbol{S}$
$B_n(\boldsymbol{x}^*, \delta)$
$\boldsymbol{x}^*$
δ
$B_n(\boldsymbol{x}^*, \delta) \cap \boldsymbol{S}$
$\boldsymbol{S}$
$B_n(\boldsymbol{x}^*, \delta)$
$\boldsymbol{x}^*$
$B_n(\boldsymbol{x}^*, \delta) \cap \boldsymbol{S}$

Figure 5.5: Illustration of $B_n(\mathbf{x}^*, \delta) \cap \boldsymbol{S}$. The left hand side is for $\mathbf{x}^*$ inside of $\boldsymbol{S}$, the right hand side for $\mathbf{x}^*$ on the border of $\boldsymbol{S}$.

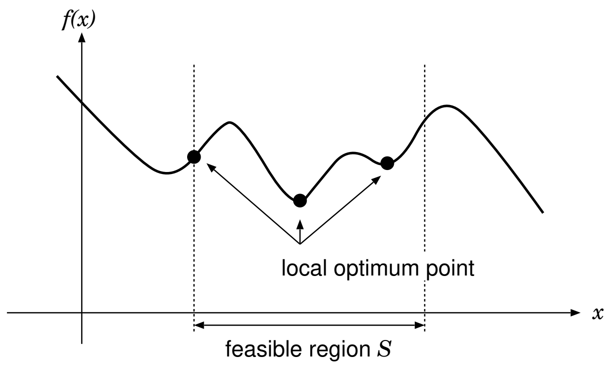


Figure 5.6: Local optimum points.

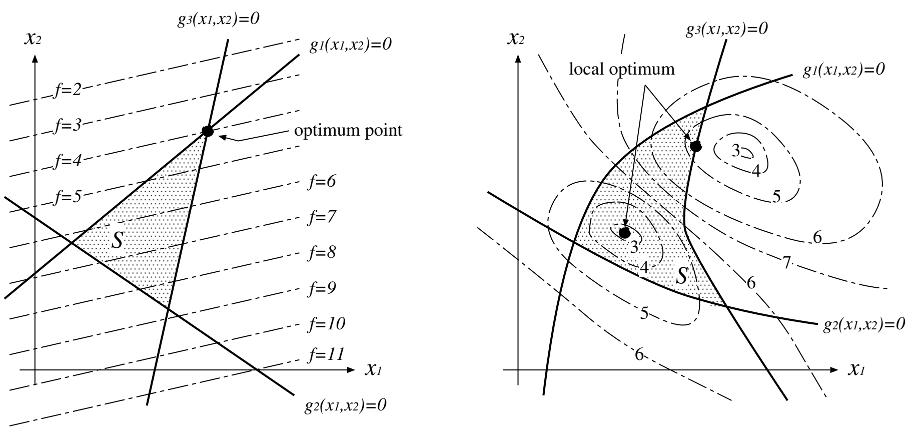


Figure 5.7: Illustration of linear programming problem and nonlinear programming problem with contour lines.

2. The feasible region is not a convex set in general, while it is a convex set for a linear programming problem.
3. A contour (a set of points where the objective function has the same value) forms a curved surface in general, while it is a flat surface for a linear programming problem.
4. Local optimum point may happen on a boundary of the feasible region or the inside of the feasible region, while it happens at a corner of the feasible region.

5.2.3 Convex Programming Problems

An optimization problem with a convex objective function and its convex feasible region is called a *convex programming problem*.

Recall that the Hessian matrix is a symmetrical matrix.

Proposition 5.5. For a convex function, Hessian matrix at any point is positive semi-definite. Conversely, a function whose Hessian matrix is positive semi-definite at any point is a convex function.

Proposition 5.6. In a convex programming problem, a locally optimal solution is a globally optimal solution.

Proof Proof is by contradiction. Assume that $\mathbf{x}^*$ is a locally optimal solution, but not a globally optimal solution, and let $\mathbf{y}^*$ be a globally optimal solution, where $\mathbf{x}^* \neq \mathbf{y}^*$. It is clear that:

$$f(\mathbf{y}^*) < f(\mathbf{x}^*). \tag{5.16}$$

Introduce a new vector $\mathbf{x}(\alpha)$ as

$$\mathbf{x}(\alpha) \triangleq (1-\alpha)\mathbf{x}^* + \alpha\mathbf{y}^*.$$

Since $\mathcal{S}$ is a convex set, $\mathbf{x}(\alpha) \in \mathcal{S}$ for all $\alpha \in [0,1]$. On the other hand, from the convexity of f:

$$\begin{aligned} f(\mathbf{x}(\alpha)) &\leq (1-\alpha)f(\mathbf{x}^*) + \alpha f(\mathbf{y}^*) && \text{Convexity of } f \\ &< (1-\alpha)f(\mathbf{x}^*) + \alpha f(\mathbf{x}^*) && \text{by (5.16)} \\ &= f(\mathbf{x}^*) \end{aligned}$$

unless $\alpha = 0$. Recall the definition of local optimality of $\mathbf{x}^*$, that all $\mathbf{x}'$ in the sphere $\mathcal{B}_n(\mathbf{x}^*, \delta)$ satisfy $f(\mathbf{x}^*) < f(\mathbf{x}')$. But we showed $f(\mathbf{x}(\alpha)) < f(\mathbf{x}^*)$, which contradicts the assumption of local optimality. □

5.3 Unconstrained Optimization

This section considers the case $\mathcal{S} = \mathbb{R}^n$, that is there are no restrictions on the variables, and our goal is to minimize an objective function $f(\mathbf{x})$.

$$\begin{array}{rl} \textbf{Minimize} & f(\mathbf{x}) \\ \textbf{subject to} & \mathbf{x} \in \mathbb{R}^n \end{array} \tag{5.17}$$

5.3.1 Optimality Conditions

This section gives conditions for optimality. Necessary conditions for optimality are considered: if $\mathbf{x}$ is a local optimum, then a necessary condition for $\mathbf{x}$ is satisfied. Sufficient conditions for optimality are also considered: if a sufficient condition for $\mathbf{x}$ is satisfied, then $\mathbf{x}$ is locally optimal. In terms of finding candidate solutions for optimization problems, sufficient conditions are more useful.

The first-order necessary condition states that at a local optimum x^*, the gradient is 0. There are two second-order conditions. The second order *necessary* condition states that if $\mathbf{x}^*$ is a locally optimum solution, then the Hessian at $\mathbf{x}^*$ is **positive semi-definite**. The second order *sufficient* condition states that if the gradient at $\mathbf{x}^*$ is 0 and the Hessian at is **positive definite**, then $\mathbf{x}^*$ is locally optimal.

Definition 5.8. For a differentiable function f, a *stationary point* $\mathbf{x}$ is a point where

$$\nabla f(\mathbf{x}) = \mathbf{0} \tag{5.18}$$

For a differentiable function, a stationary point may be locally optimal, either a local minimum or local maximum. A stationary point may also be a saddle point which has $\nabla f = \mathbf{0}$, but is not locally optimal.

Proposition 5.7 (First-order necessary condition)**.** Let f be differentiable. If $\mathbf{x}^*$ is a local optimum solution of the unconstrained optimization problem (5.17), then $\mathbf{x}^*$ is a stationary point, that is $\nabla f(\mathbf{x}) = \mathbf{0}$.

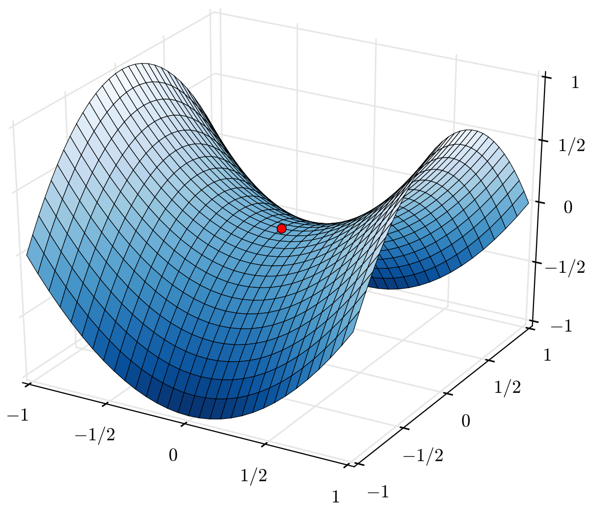


Figure 5.8: Saddle point. Wikipedia/Nicoguaro

A point $\mathbf{x}^*$ which satisfies (5.18) is called a *stationary point*, and the value $f(\mathbf{x}^*)$ is the stationary value.

Proposition 5.8. For an unconstrained optimization problem (5.17), if f is a convex function, then a stationary point $\mathbf{x}^*$ a global optimum.

Proof The proof is by contradiction. Let $\mathbf{x}^*$ be a stationary point, but assume there exists some distinct $\mathbf{y}^* \neq \mathbf{x}^*$ which has a smaller value $f(\mathbf{y}^*) < f(\mathbf{x}^*)$. It is shown that this is not possible. Since f is a convex function, the following holds for all α, $0 \leq \alpha \leq 1$:

$$\begin{aligned}(1-\alpha)f(\mathbf{x}^*) + \alpha f(\mathbf{y}^*) &\geq f((1-\alpha)\mathbf{x}^* + \alpha\mathbf{y}^*)\\ &= f(\mathbf{x}^* + \alpha(\mathbf{y}^* - \mathbf{x}^*)).\end{aligned}$$

Using the differentiability of f and $\nabla f(\mathbf{x}^*) = \mathbf{0}$:

$$\begin{aligned}f(\mathbf{x}^* + \alpha(\mathbf{y}^* - \mathbf{x}^*)) &= f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*) \cdot \alpha(\mathbf{y}^* - \mathbf{x}^*)^{\mathrm{t}} + o(\alpha||\mathbf{y}^* - \mathbf{x}^*||)\\ &= f(\mathbf{x}^*) + o(\alpha||\mathbf{y}^* - \mathbf{x}^*||),\end{aligned}$$

which can be reduced to

$$-\alpha f(\mathbf{x}^*) + \alpha f(\mathbf{y}^*) \geq o(\alpha||\mathbf{y}^* - \mathbf{x}^*||).$$

When we exclude the case $\alpha = 0$, and consider all α, $0 < \alpha \leq 1$, then we have,

$$\frac{-\alpha f(\mathbf{x}^*) + \alpha f(\mathbf{y}^*)}{\alpha||\mathbf{y}^* - \mathbf{x}^*||} = \frac{f(\mathbf{y}^*) - f(\mathbf{x}^*)}{||\mathbf{y}^* - \mathbf{x}^*||} \geq \frac{o(\alpha||\mathbf{y}^* - \mathbf{x}^*||)}{\alpha||\mathbf{y}^* - \mathbf{x}^*||}$$

Since $\mathbf{y}^* \neq \mathbf{x}^*$ and $f(\mathbf{y}^*)$ is strictly smaller than $f(\mathbf{x}^*)$, there must exist $\varepsilon' > 0$ such that

$$\frac{f(\mathbf{y}^*) - f(\mathbf{x}^*)}{||\mathbf{y}^* - \mathbf{x}^*||} < -\varepsilon' < 0.$$

On the other hand, $o(\cdot)$ is a function having the following property. For all $\varepsilon > 0$ there exists $\delta > 0$ such that:

$$\forall\,(\alpha||\mathbf{y}^* - \mathbf{x}^*||) < \delta, \quad \left|\frac{o(\alpha||\mathbf{y}^* - \mathbf{x}^*||)}{\alpha||\mathbf{y}^* - \mathbf{x}^*||}\right| < \varepsilon$$

The existence of δ is guaranteed even if we set $\varepsilon = \varepsilon'$.

$$\exists\delta > 0 \ \text{ s.t. } \quad \forall\alpha < \frac{\delta}{||\mathbf{y}^* - \mathbf{x}^*||}, \ -\varepsilon' < \frac{o(\alpha||\mathbf{y}^* - \mathbf{x}^*||)}{\alpha||\mathbf{y}^* - \mathbf{x}^*||} < \varepsilon'$$

However this contradicts

$$\frac{f(\mathbf{y}^*) - f(\mathbf{x}^*)}{||\mathbf{y}^* - \mathbf{x}^*||} \geq \frac{o(\alpha||\mathbf{y}^* - \mathbf{x}^*||)}{\alpha||\mathbf{y}^* - \mathbf{x}^*||}.$$

for $0 < \alpha \leq 1$. This is a contradiction since $f(\mathbf{y}^*) < f(\mathbf{x}^*)$. □

Proposition 5.9 (Second-order necessary condition)**.** Let f be twice differentiable. If $\mathbf{x}^*$ is a locally optimum solution of (5.17), then $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite.

To prove this theorem, we assume that $\nabla^2 f(\mathbf{x}^*)$ is not positive semi-definite. From this assumption, there exists $\mathbf{d} \in \mathbb{R}^n$ which satisfies

$$\mathbf{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^*)\mathbf{d} < 0$$

From the twice differentiability of f, we have

$$\begin{aligned} f(\mathbf{x}^* + \alpha\boldsymbol{d}) &= f(\mathbf{x}^*) + \alpha\nabla f(\mathbf{x}^*)^{\mathrm{t}}\mathbf{d} + \frac{1}{2}\alpha^2\mathbf{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^*)\mathbf{d} + o(\alpha^2||\mathbf{d}||^2) \\ &= f(\mathbf{x}^*) + \frac{1}{2}\alpha^2\mathbf{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^*)\mathbf{d} + o(\alpha^2||\mathbf{d}||^2) \\ &= f(\mathbf{x}^*) + \alpha^2||\mathbf{d}||^2\left(\frac{1}{2}\frac{\mathbf{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^*)\mathbf{d}}{||\mathbf{d}||^2} + \frac{o(\alpha^2||\mathbf{d}||^2)}{\alpha^2||\mathbf{d}||^2}\right) \end{aligned}$$

The second term of the right hand side is negative when choosing a sufficiently small value for α, and as a result,

$$f(\mathbf{x}^* + \alpha\mathbf{d}) < f(\mathbf{x}^*)$$

holds, which contradicts the fact that $\mathbf{x}^*$ is a local optimum solution.

Proposition 5.10 (Second order sufficient condition)**.** Let f be twice differentiable. If $\mathbf{x}^*$ satisfies the first order necessary condition and $\nabla^2 f(\mathbf{x}^*)$ is positive definite, then $\mathbf{x}^*$ is a local optimum solution.

For an arbitrary nonzero vector $\mathbf{d}$, we have $\mathbf{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^*)\mathbf{d} > 0$ and

$$f(\mathbf{x}^* + \alpha\mathbf{d}) = f(\mathbf{x}^*) + \alpha^2||\mathbf{d}||^2\left(\frac{1}{2}\frac{\mathbf{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^*)\mathbf{d}}{||\mathbf{d}||^2} + \frac{o(\alpha^2||\mathbf{d}||^2)}{\alpha^2||\mathbf{d}||^2}\right)$$

Then we can find an α such that the second term in the right hand side is positive, and we have

$$f(\mathbf{x} + \tilde{\alpha}\mathbf{d}) > f(\mathbf{x}^*),$$

for some $\tilde{\alpha}$ satisfying $0 < \tilde{\alpha} < \alpha$.

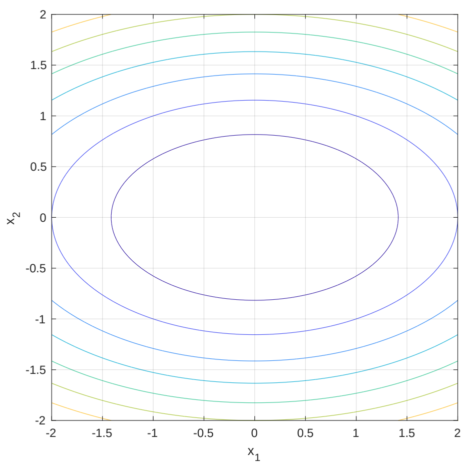
5.3.2 Examples

Example 5.4. In this example, both the second order necessary and sufficient conditions are satisfied.

For the objective function $f(x_1, x_2) = x_1^2 + 3x_2^2$, the gradient and Hessian matrix are:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 6x_2 \end{bmatrix} \text{ and } \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

f is minimum at $(x_1, x_2) = (0, 0)$. At this point, $\nabla f(\mathbf{0}) = \mathbf{0}$ and $\nabla^2 f(\mathbf{0})$ is positive semi-definite, as predicted by Proposition 5.9. In addition, it can be verified that $(0, 0)$ satisfies the second order sufficient condition.



Example 5.5. The following example illustrates that even though $\mathbf{x}^*$ is optimal, its Hessian is not positive definite.

For the function $f(x_1, x_2) = x_1^2 + x_2^4$ the gradient and Hessian matrix are:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 4x_2^3 \end{bmatrix} \text{ and } \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 12x_2^2 \end{bmatrix}$$

f is minimum at $(x_1, x_2) = (0, 0)$, and $\nabla f(\mathbf{0}) = \mathbf{0}$. The Hessian matrix

$$\nabla^2 f(\mathbf{0}) = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$$

is *not* positive definite (even though it is positive semi-definite), thus the second order sufficient condition is not satisfied. However, it is easy to verify that $\mathbf{x}^*$ satisfies the second-order necessary condition.

Example 5.6. The following example illustrates a point which is stationary, but is not a local optimum.

For the objective function $f(x_1, x_2) = x_1^2 + x_2^3$, the gradient and Hessian matrix are:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 3x_2^2 \end{bmatrix} \text{ and } \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 6x_2 \end{bmatrix}.$$

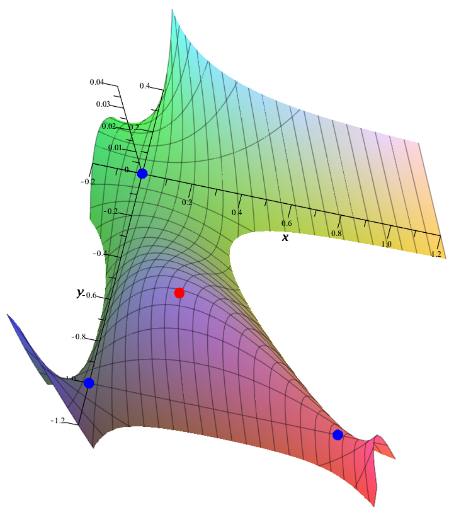
At the point $\mathbf{x} = \mathbf{0}$, $\nabla f(\mathbf{0}) = \mathbf{0}$ and the Hessian matrix

$$\nabla^2 f(\mathbf{0}) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

is not positive definite. Thus, even though $\mathbf{x}^*$ is a stationary point, this Hessian matrix does not satisfy the second-order sufficient condition. This point $\mathbf{x} = \mathbf{0}$ is not a local optimum.

Example 5.7. Find and classify the stationary points of the function:

$$f(x_1, x_2) = (x_1 + x_2)(x_1 x_2 + x_1 x_2^2)$$



First set the gradient equal to zero:

$$\nabla f = \begin{bmatrix} x_2(2x_1 + x_2)(x_2 + 1) \\ x_1\left(3x_2^2 + 2x_2(x_1 + 1) + x_1\right) \end{bmatrix} = \mathbf{0} \tag{5.19}$$

Solving these equations gives four stationary points:

(a) $(0, -1)$

(b) $(1, -1)$

(c) $(\frac{3}{8}, -\frac{3}{4})$

(d) $(0, 0)$

In order to classify the stationary points, find the Hessian:

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 2x_2(x_2+1) & 2x_1 + 2x_2 + 4x_1x_2 + 3x_2^2 \\ 2x_1 + 2x_2 + 4x_1x_2 + 3x_2^2 & 2x_1(x_1 + 3x_2 + 1) \end{bmatrix}$$

Then, evaluate $\nabla^2 f(x_1, x_2)$ at each stationary point:

(a) $(0, -1)$: $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow \lambda_1 = -1, \lambda_2 = 1 \Rightarrow$ mixed signs indicate indefinite matrix $\Rightarrow$ saddle point.

(b) $(1, -1)$: $\mathbf{A} = \begin{bmatrix} 0 & -1 \\ -1 & -2 \end{bmatrix} \Rightarrow \lambda_1 = -1 - \sqrt{2} \approx -2.4142, \lambda_2 = -1 + \sqrt{2} \approx 0.4142 \Rightarrow$ saddle point

(c) $(\frac{3}{8}, -\frac{3}{4})$: $\mathbf{A} = \begin{bmatrix} -\frac{3}{8} & -\frac{3}{16} \\ -\frac{3}{16} & -\frac{21}{32} \end{bmatrix} \Rightarrow \lambda_1 = -\frac{3}{4}, \lambda_2 = -\frac{9}{32} \Rightarrow$ negative eigenvalues indicate negative definite $\Rightarrow$ local maximum

(d) $(0, 0)$: $\mathbf{A} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow \lambda_1 = 0, \lambda_2 = 0 \Rightarrow$ test in insufficient.

In the figure above, the red point is the local maximum. The three blue points are saddle points. Wikipedia: Second partial derivative test

---

5.3.3 Exercises

1. Find a stationary point of the following objective functions. Decide whether the stationary point is a (local) minimum point or not.

   (a)
   $$f(\mathbf{x}) = \mathbf{x}^t \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}$$

   (b)
   $$f(\mathbf{x}) = \mathbf{x}^t \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}$$

5.4 Numerical Approaches to Unconstrained Nonlinear Programming Problem*

In some problem instances, we can find optimum solutions analytically by using necessary conditions and sufficient conditions. In other cases, we usually use some iterative method which is designed to generate a sequence of points $\{\mathbf{x}^{(k)}\}$ which converges to a (local) optimum point.

5.4.1 Steepest Descent Method

Beginning with an initial point $\mathbf{x}^{(0)}$, a sequence of points $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, $\cdots$, is generated by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\boldsymbol{d}^{(k)}$$

such that

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}), \quad k = 0, 1, 2, \cdots .$$

Such method is call "descent method" in general. In this descent method, $\boldsymbol{d}^{(k)}$ and $\alpha^{(k)}$ are called "direction vector" and "step size", respectively.

In the steepest descent method, the direction vector at $\mathbf{x}^{(k)}$ is chosen as $\boldsymbol{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$.

<u>Steepest descent method</u>

```
0. Choose initial point $\mathbf{x}^{(0)}$, $k \leftarrow 0$.
1. Compute $\nabla f(\mathbf{x}^{(k)})$.
   If $\nabla f(\mathbf{x}^{(k)}) = 0$ (*), then quit.
   $\boldsymbol{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$.
2. Find $\alpha^{(k)}$, and $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\boldsymbol{d}^{(k)}$
   $k \leftarrow k + 1$ and go to step 1.
```

(* In a practical implementation, a sufficiently small value $\varepsilon_{spec} > 0$ is prepared, and $||\boldsymbol{d}^{(k)}|| < \varepsilon_{spec}$ is used as a termination criterion.)

- If a sequence points generated by the steepest descent method is bounded, the sequence converges to a stationary point.
- Depending on the choice of the initial point, the sequence of points generated by descent method may converge to different stationary point.

<u>Speed of convergence:</u>

Assume that $\mathbf{x}^*$, the limit of a sequence of points generated by the steepest descent method, satisfies the second order sufficient condition for optimality. That is, f is twice differentiable, $\nabla f(\mathbf{x}^*) = \mathbf{0}$, and $\nabla^2 f(\mathbf{x}^*) \triangleq \boldsymbol{G}$ is positive definite.

Here we use a norm defined as follows.

$$||\mathbf{x}||_G \triangleq \sqrt{\mathbf{x}^t \boldsymbol{G} \mathbf{x}}$$

With respect to the sequence of points, $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, $\cdots$, for any $\varepsilon > 0$ there exists a positive integer K such that

$$^{\forall}k > K, \quad ||\mathbf{x}^{(k+1)} - \mathbf{x}^*||_G \leq \left(\frac{\tau - 1}{\tau + 1} + \varepsilon\right) ||\mathbf{x}^{(k)} - \mathbf{x}^*||_G \tag{5.20}$$

where $\tau = \lambda_{max}/\lambda_{mim}$, and λ_{max} and λ_{min} are the largest eigen value and the smallest eigen value of $\boldsymbol{G}$, respectively. τ is called "condition number" of $\boldsymbol{G}$. Note that, since $\tau \geq 1$, $0 \leq (\tau-1)/(\tau+1) < 1$.

- Eq.(5.20) indicates that the distance between $\mathbf{x}^{(k)}$ and $\mathbf{x}^*$ is decreasing (as k is increasing) with the rate about $(\tau-1)/(\tau+1)$. (Linear convergence)
- Convergence ratio is small (converge rapidly) when τ is close 1, and is large (converge slowly) when τ is close to 0.

5.4.2 Newton's Method

From Taylor expansion of an objective function f around $\mathbf{x}^{(k)}$ and neglecting higher order terms, we have the following.

$$\tilde{f}(\mathbf{x}) \;=\; f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})^{\mathrm{t}}(\mathbf{x}-\mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(k)})^{\mathrm{t}}\nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)})$$

For simplicity, we let $\mathbf{x}-\mathbf{x}^{(k)} = \boldsymbol{d}$.

$$\tilde{f}(\mathbf{x}^{(k)}+\boldsymbol{d}) \;=\; f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})^{\mathrm{t}}\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^{\mathrm{t}}\nabla^2 f(\mathbf{x}^{(k)})\boldsymbol{d} \;\triangleq\; q^{(k)}(\boldsymbol{d})$$

If $\nabla^2 f(\mathbf{x}^{(k)})$ is positive definite, $q^{(k)}(\boldsymbol{d})$ becomes a convex function, and it is minimized with $\boldsymbol{d}$ which satisfies

$$\nabla q^{(k)}(\boldsymbol{d}) \;=\; \nabla f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)})\boldsymbol{d} \;=\; \mathbf{0}.$$

That is,

$$\boldsymbol{d} \;=\; -\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1}\nabla f(\mathbf{x}^{(k)})$$

In Newton's method, the direction vector is chosen as $\boldsymbol{d}^{(k)} = -\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1}\nabla f(\mathbf{x}^{(k)})$.

<u>Newton's method</u>

```
0. Choose an initial point $\mathbf{x}^{(0)}$. $k \leftarrow 0$.
1. Compute $\nabla f(\mathbf{x}^{(k)})$ and $\nabla^2 f(\mathbf{x}^{(k)})$.
   $\boldsymbol{d}^{(k)} \leftarrow$ solution of $\nabla^2 f(\mathbf{x}^{(k)})\boldsymbol{d} = -\nabla f(\mathbf{x}^{(k)})$
   If $\boldsymbol{d}^{(k)} = \mathbf{0}$, then quit.(*).
2. Find step size $\alpha^{(k)}$.
   $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\boldsymbol{d}^{(k)}$
   $k \leftarrow k+1$, and go to step 1.
```

- In this method, the positive definiteness of Hessian matrix is assumed, but such assumption is not satisfied in many problems, but satisfied only locally.

- If a local optimum point $\mathbf{x}^*$ satisfies the second order sufficient condition, Hessian matrix becomes positive definite locally around $\mathbf{x}^*$. So, we need to choose an initial point carefully from a region close to a local optimum point.
- Depending on the choice of an initial point, a generated sequence of points may converges to different local optimum point.
- Most important advantage of Newton's method is its speed of convergence.

Speed of convergence:

Now we assume that f is three times differentiable and $\mathbf{x}^{(k)}$ is sufficiently close to a minimum point $\mathbf{x}^*$. We also assume that Hessian matrix $\nabla^2 f(\mathbf{x}^*)$ of f at the point $\mathbf{x}^*$ is positive definite.

Taylor expansion of ∇f around $\mathbf{x}^{(k)}$ yields

$$\nabla f(\mathbf{x}) \quad = \quad \nabla f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) + O(||\mathbf{x} - \mathbf{x}^{(k)}||^2)$$

where $O(t)$ is an auxiliary function satisfying

$$\exists t_0,\ \exists\beta \text{ such that } ||O(t)|| \leq \beta t \text{ for all } t,\ t_0 > t > 0,$$

Since $\nabla f(\mathbf{x}^*) = \mathbf{0}$ at the point $\mathbf{x}^*$, we have

$$\mathbf{0} \quad = \quad \nabla f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}^* - \mathbf{x}^{(k)}) + O(||\mathbf{x}^* - \mathbf{x}^{(k)}||^2)$$

Multiply $\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1}$,

$$\mathbf{0} \quad = \quad \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)}) + (\mathbf{x}^* - \mathbf{x}^{(k)}) + O(||\mathbf{x}^* - \mathbf{x}^{(k)}||^2)$$

Using $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)})$ (assuming $\alpha^{(k)} = 1$), we have

$$\mathbf{0} \quad = \quad \mathbf{x}^* - \mathbf{x}^{(k+1)} + O(||\mathbf{x}^* - \mathbf{x}^{(k)}||^2)$$

That is,

$$\mathbf{x}^{(k+1)} - \mathbf{x}^* = O(||\mathbf{x}^{(k)} - \mathbf{x}^*||^2)$$

which means that there exits a constant $\beta > 0$ satisfying

$$||\mathbf{x}^{(k+1)} - \mathbf{x}^*|| \quad \leq \quad \beta||\mathbf{x}^{(k)} - \mathbf{x}^*||^2.$$

- Newton's method with $\alpha^{(k)} = 1$ has second order convergence.

5.4.3 Quasi-Newton Method

In Newton's method, we assume the positive definiteness of Hessian matrix $\nabla^2 f(\mathbf{x})$, but it is not satisfied in many cases.

In the generating equation of Newton's method, we will use an estimated quasi-Hessian matrix $\boldsymbol{B}^{(k)}$ instead of $\nabla^2 f(\mathbf{x}^{(k)})$.

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha^{(k)} {B^{(k)}}^{-1} \nabla f(\mathbf{x}^{(k)})$$

Now the problem is how to compute $B^{(k+1)}$ to complete the recursion.

Consider the approximation of $\nabla f(\mathbf{x})$ by Taylor expansion around $\mathbf{x}^{(k+1)}$.

$$\nabla f(\mathbf{x}) \quad \simeq \quad \nabla f(\mathbf{x}^{(k+1)}) + \nabla^2 f(\mathbf{x}^{(k+1)})(\mathbf{x} - \mathbf{x}^{(k+1)})$$

If we evaluate $f(\mathbf{x}^{(k)})$ with this approximation, we have

$$\nabla f(\mathbf{x}^{(k)}) \quad \simeq \quad \nabla f(\mathbf{x}^{(k+1)}) + \nabla^2 f(\mathbf{x}^{(k+1)})(\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)})$$

which can be rewritten as

$$\nabla^2 f(\mathbf{x}^{(k+1)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \quad \simeq \quad \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$$

For simplicity, we let

$$\boldsymbol{s}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}, \qquad \mathbf{y}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$$

and we will an approximation of $\nabla^2 f(\mathbf{x}^{(k+1)})$ as $\boldsymbol{B}^{(k+1)}$. Then we have

$$\boldsymbol{B}^{(k+1)} \boldsymbol{s}^{(k)} \quad = \quad \mathbf{y}^{(k)}$$

Assuming that $\mathbf{x}^{(k+1)}$ is computed by using $\boldsymbol{B}^{(k)}$ as shown above, and $\nabla f(\mathbf{x}^{(k+1)})$ is also computed, $\boldsymbol{s}^{(k)}$ and $\mathbf{y}^{(k)}$ are available. So the problem is to find $\boldsymbol{B}^{(k+1)}$ so that it is symmetric and positive definite, and satisfies the above equation.

One solution proposed by Broyden, Fletcher, Goldfarb, Shanno:

$$\boldsymbol{B}^{(k+1)} \quad = \quad \boldsymbol{B}^{(k)} + \frac{\mathbf{y}^{(k)} \mathbf{y}^{(k)T}}{\mathbf{y}^{(k)T} \boldsymbol{s}^{(k)}} - \frac{\boldsymbol{B}^{(k)} \boldsymbol{s}^{(k)} \boldsymbol{s}^{(k)T} \boldsymbol{B}^{(k)}}{\boldsymbol{s}^{(k)T} \boldsymbol{B}^{(k)} \boldsymbol{s}^{(k)}}$$

- $\boldsymbol{B}^{(k+1)}$ satisfies $\boldsymbol{B}^{(k+1)} \boldsymbol{s}^{(k)} = \mathbf{y}^{(k)}$.
- If $\boldsymbol{B}^{(k)}$ is symmetric, $\boldsymbol{B}^{(k+1)}$ is also symmetric.
- If $\boldsymbol{B}^{(k)}$ is positive definite and $\mathbf{y}^{(k)T} \boldsymbol{s}^{(k)} > 0$, then $\boldsymbol{B}^{(k+1)}$ is also positive definite.

- If $\boldsymbol{B}^{(k)}$ is positive definite and $\nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}$, then $-(\boldsymbol{B}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})$ is a descent direction of $f(\mathbf{x})$. That is;

$$\begin{aligned}&\nabla f(\mathbf{x}^{(k)})^{\mathrm{t}}\left(-(\boldsymbol{B}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})\right)\\ =\ & -\nabla f(\mathbf{x}^{(k)})^{\mathrm{t}}(\boldsymbol{B}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})\end{aligned}$$

  Since the inverse matrix of a positive definite matrix is also positive definite, we have

$$-\nabla f(\mathbf{x}^{(k)})^{\mathrm{t}}(\boldsymbol{B}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)}) < 0$$

<u>Quasi-Newton method</u>:

```
0. Choose $\mathbf{x}^{(0)}$ and $\boldsymbol{B}^{(0)}$. $k \leftarrow 0$.
1. Compute $\nabla f(\mathbf{x}^{(k)})$.
   If $\nabla f(\mathbf{x}^{(k)}) = \mathbf{0}$, then quit. Otherwise,
   compute $\boldsymbol{d}^{(k)} = -(\boldsymbol{B}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})$.
2. Find the step size $\alpha^{(k)}$, and $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\boldsymbol{d}^{(k)}$.
3. Compute $\boldsymbol{B}^{(k+1)}$.
   If $\mathbf{y}^{(k)T}\boldsymbol{s}^{(k)} \leq 0$, then $\boldsymbol{B}^{(k+1)} = \boldsymbol{B}^{(k)}$.
   $k \leftarrow k + 1$. Go to step 1.
```

- If Hessian matrix $\nabla^2 f(\mathbf{x}^*)$ is positive definite at a local optimum point $\mathbf{x}^*$, then

$$\lim_{k\to\infty} \frac{||\mathbf{x}^{(k+1)} - \mathbf{x}^*||}{||\mathbf{x}^{(k)} - \mathbf{x}^*||} = 0$$

  holds, which indicate that quasi-Newton method has superlinear convergence.

5.4.4 Exercise

1. Consider the unconstrained optimization problem with its objective function $f(x_1, x_2)$.

$$f(x_1, x_2) = x_1^2 + 4x_2^2 - 4x_1 - 8x_2 - 2$$

   (1) Applying Steepest Descent Method, find the point $(x_1^{(1)}, x_2^{(1)})$ next to the current point $(x_1^{(0)}, x_2^{(0)}) = (0, 2)$. In addition, compute $f(x_1^{(1)}, x_2^{(1)})$.

   (2) Applying Newton Method, find the point $(x_1^{(1)}, x_2^{(1)})$ next to the current point $(x_1^{(0)}, x_2^{(0)}) = (0, 2)$. In addition, compute $f(x_1^{(1)}, x_2^{(1)})$.

5.5 Optimization with Equality Constraints

This section considers the constrained problem which was given in (5.15) on page 95, but with only equality constraints:

$$\begin{aligned} \textbf{Minimize} \quad & f(\mathbf{x}) \\ \textbf{subject to} \quad & h_i(\mathbf{x}) = 0 \text{ for } i \in \{1, 2, \ldots, \ell\} \end{aligned} \tag{5.21}$$

where f and h_i are twice differentiable. In many cases, this problem may be solved by the method of Lagrange multipliers, which is described here. Although the problem is written as min, in fact finds both min andmax. Wikipedia: Lagrange Multiplier

Proposition 5.11 (Lagrange multiplier)**.** Let $\mathbf{x}^*$ be a local optimum of (5.21). Then there exists a unique Lagrange multiplier $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_\ell)$ such that:

$$\nabla f(\mathbf{x}^*) - \boldsymbol{\lambda} \nabla g(\mathbf{x}^*) = 0. \tag{5.22}$$

Solution of this set of simultaneous equations in (5.22) is a candidate solution of the optimization problem. To understand why, the solution is a stationary point of the *Lagrangian function*:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i \cdot h_i(\mathbf{x})$$

with variables $\mathbf{x}$, $\boldsymbol{\lambda}$.

In other words, we convert the constrained optimization problem (5.21) in the variables $\mathbf{x}$ to an unconstrained optimization problem (5.22) in the variables $\mathbf{x}$ and $\boldsymbol{\lambda}$. The first step of solving this unconstrained problem by taking the gradient and setting equal to zero:

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) \\ \nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda}) \end{bmatrix} = \begin{bmatrix} \nabla f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i \cdot \nabla h_i(\mathbf{x}) \\ h_1(\mathbf{x}) \\ \vdots \\ h_\ell(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The effectiveness is visualized in Fig. 5.9, where the function $f(x, y)$ is minimized (arrows pointing negative gradient) subject to the constraint $g(x, y) = c$. If we walk along the curve $g(x, y) = c$, the objective function f gradually decreases. But the gradient ∇g does not match the gradient of ∇f. Only when the point along g reaches the minimum (at the level set d_1), then the two gradients match. They do not necessarily have the same magnitude, which is accounted for by λ: $\nabla f = \lambda \nabla g$.

Example 5.8. Consider the following constrained optimization problem:

$$\begin{aligned} \textbf{Minimize} \quad & (x_1 + x_2)^2 \\ \textbf{subject to} \quad & x_1^2 + x_2^2 = 1. \end{aligned} \tag{5.23}$$

The problem is illustrated below.

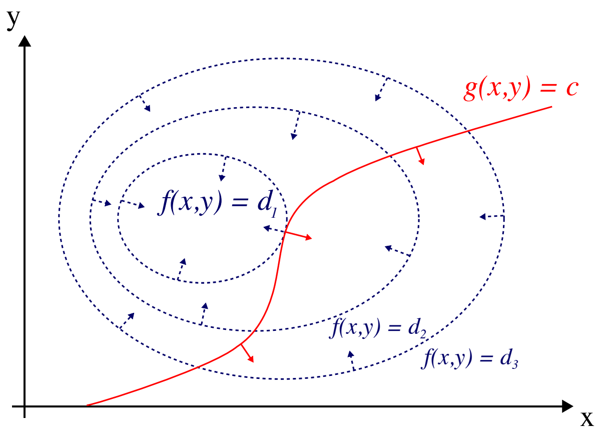
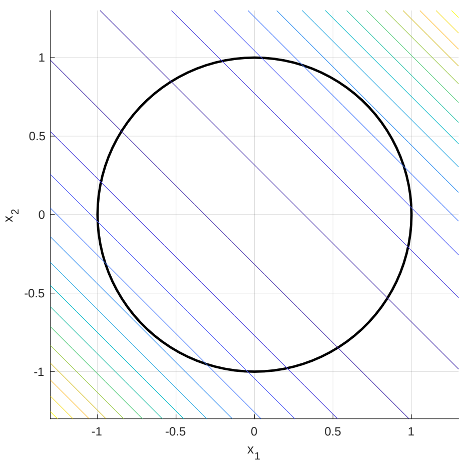


Figure 5.9: Credit: Wikipedia/Nexcis



We solve this using the method of Lagrange multipliers. The Lagrangian is:

$$L(\mathbf{x}, \lambda) = (x_1 + x_2)^2 + \lambda(x_1^2 + x_2^2 - 1)$$

Solve $\nabla L(\mathbf{x}, \lambda) = \mathbf{0}$:

$$\frac{\partial L}{\partial x_1} = 2x_1 + 2x_2 + 2\lambda x_1 = 0 \tag{5.24}$$

$$\frac{\partial L}{\partial x_2} = 2x_1 + 2x_2 + 2\lambda x_2 = 0 \tag{5.25}$$

$$\frac{\partial L}{\partial \lambda} = x_1^2 + x_2^2 - 1 = 0 \tag{5.26}$$

Our goal is to solve this system of equations. Because the problem is nonlinear, there is no systematic method, and in practice it may be difficult to find a solution. In general approach that often works well is to first find λ first, then find the x_i.

Here, note that (5.24) and (5.25) can be written as:

$$\begin{bmatrix} 1+\lambda & 1 \\ 1 & 1+\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{0} \tag{5.27}$$

Non-zero solutions are found when the determinant is 0, that is det $= (1+\lambda)^2-1$, which means $\lambda = 0$ or -2. Consider $\lambda = 0$ first, giving $x_1+x_2 = 0$. Combining this with (5.26), the possible solutions are $\mathbf{x} = (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$. These are local minima.

Consider $\lambda = -2$, giving $x_1-x_2 = 0$. The possible solutions are $\mathbf{x} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$. These are local maximum.

Example 5.9. The following constrained optimization problem arises in information theory.

$$\begin{aligned} \textbf{Maximize} \quad & \textstyle\sum_{i=1}^n a_i \log p_i \\ \textbf{subject to} \quad & \textstyle\sum_{i=1}^n p_i = 1. \end{aligned} \tag{5.28}$$

Here, $a_1, \dots, a_n$ are constants with $a_i > 0$, and the maximization is performed over $p_1, \dots, p_n$.

Form the Lagrangian $L(\mathbf{p}, \lambda)$ with Lagrange multiplier λ:

$$L(\mathbf{p}, \lambda) = \sum_{i=1}^n a_i \log p_i - \lambda \sum_{i=1}^n p_i \tag{5.29}$$

Solve $\nabla L = 0$:

$$\frac{\partial L}{\partial p_i} = \frac{a_i}{p_i} - \lambda = 0 \tag{5.30}$$

$$p_i^* = \frac{a_i}{\lambda} \tag{5.31}$$

Note that $\frac{\partial L}{\partial \lambda}$ gives the constraint. Apply (5.31) to the constraint:

$$\sum_{i=1}^n p_i = 1 \;\Rightarrow\; \sum_{i=1}^n \frac{a_i}{\lambda} = 1 \;\Rightarrow\; \lambda = \sum_{i=1}^n a_i \tag{5.32}$$

Applying (5.32) to (5.31), the solution:

$$p_i^* = \frac{a_i}{a_1 + a_2 + \cdots + a_n}. \tag{5.33}$$

is obtained.

Proposition 5.12. If the objective function $f(\mathbf{x})$ is convex, and equality constraints are all linear with respect to $\mathbf{x}$ as

$$\begin{pmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_\ell(\mathbf{x}) \end{pmatrix} = \mathbf{A}\mathbf{x} - \boldsymbol{b} = \mathbf{0},$$

then $\mathbf{x}^*$ at the stationary point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ of the Lagrangian function

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}} (\mathbf{A}\mathbf{x} - \boldsymbol{b})$$

is an optimum solution of the original nonlinear programming problem, where

$$\nabla L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \begin{bmatrix} \nabla f(\mathbf{x}^*) + \mathbf{A}^{\mathrm{t}}\boldsymbol{\lambda}^* \\ \mathbf{A}\mathbf{x}^* - \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

5.5.1 Exercises

1. Find the ~~stationary points~~ local optimal points of the following equality-constrained optimization problem. ~~Classify the stationary points.~~ Classify each point as a local maximum, local minimum or saddle point.

$$\begin{array}{rc} \textbf{Optimize} & -2x_1x_2^2 \\ \textbf{subject to} & \frac{1}{2}x_1^2 + x_2^2 - \frac{3}{2} = 0 \end{array}$$

5.6 Constrained Optimization Problems

This section describes the characteristics of the optimum solution and solution techniques for the general nonlinear optimization problem which was given in (5.15) on page 95 and repeated here:

$$\begin{array}{rl} \textbf{Minimize} & f(\mathbf{x}) \\ \textbf{subject to} & h_i(\mathbf{x}) = 0, \;\; i = 1, 2, \ldots, \ell \\ & g_j(\mathbf{x}) \leq 0, \;\; j = 1, 2, \ldots, m \end{array} \tag{5.34}$$

5.6.1 Lagrangian Function

For the constrained optimization problem in (5.34), the *Lagrangian function* or simply *Lagrangian* is given as:

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{m} \mu_j g_j(\mathbf{x}), \tag{5.35}$$

where $\mu_j \geq 0,\; j = 1, \cdots, m$ and there is no sign constraint on λ_i. The λ_i and μ_j are called *Lagrangian multipliers*. The Lagrangian function L is a function of variables $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$, $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_\ell) \in \mathbb{R}^\ell$ and $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_m)$.

If we define the following vectors:

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_\ell(\mathbf{x}) \end{bmatrix} \qquad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_\ell(\mathbf{x}) \end{bmatrix}$$

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_\ell)^{\mathrm{t}} \qquad \boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_m)^{\mathrm{t}}.$$

then the Lagrangian may be written in vector form:

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}}\mathbf{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{t}}\mathbf{g}(\mathbf{x}).$$

Consider *maximizing* L with respect to variables $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. Define a new function $\theta(\mathbf{x})$ as:

$$\begin{aligned}\theta(\mathbf{x}) &= \max_{\boldsymbol{\lambda}\in\mathbb{R}^{\ell}, \boldsymbol{\mu}\in\mathbb{R}_{\geq 0}{}^{m}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= \begin{cases} +\infty & \text{if } \exists i,\ h_i(\mathbf{x}) \neq 0 \text{ or } \exists j,\ g_j(\mathbf{x}) > 0 \\ f(\mathbf{x}) & \text{if } \forall i,\ h_i(\mathbf{x}) = 0 \text{ and } \forall j,\ g_j(\mathbf{x}) \leq 0 \end{cases}\end{aligned}$$

In other words, if $\mathbf{x}$ satisfies the constraints then $\theta(\mathbf{x})$ is $f(\mathbf{x})$. If $\mathbf{x}$ does not satisfy the constraints, then $\theta(\mathbf{x})$ is infinity. Why does $\theta(\mathbf{x}) = f(\mathbf{x})$ when $\mathbf{x}$ satisfies the constraints? Clearly, if $h_i(\mathbf{x}) = 0$ then the h_i do not contribute to L. If $g_j(\mathbf{x}) < 0$, then choose $\mu_j \geq 0$ to make L as large as possible, that is choose $\mu_j = 0$.

Now *minimize* $\theta(\mathbf{x})$ with respect to $\mathbf{x}$:

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^n} \theta(\mathbf{x}) &= \begin{cases} +\infty & \text{if } {}^{\forall}\mathbf{x} \in \mathbb{R}^n,\ {}^{\exists}i,\ h_i(\mathbf{x}) \neq 0 \text{ or } {}^{\exists}j,\ g_j(\mathbf{x}) > 0 \\ \displaystyle\min_{\substack{\mathbf{x} \text{ such that } {}^{\forall}i,\ h_i(\mathbf{x}) = 0 \\ \text{and } {}^{\forall}j,\ g_j(\mathbf{x}) \leq 0}} f(\mathbf{x}), & \text{if } {}^{\exists}\mathbf{x} \in \mathbb{R}^n,\ {}^{\forall}i,\ h_i(\mathbf{x}) = 0 \text{ and } {}^{\forall}j,\ g_j(\mathbf{x}) \leq 0 \end{cases} \\ &= \begin{cases} +\infty & \text{if } \mathcal{S} = \emptyset \\ \displaystyle\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}), & \text{if } \mathcal{S} \neq \emptyset \end{cases}\end{aligned}$$

Now the relation between Lagrangian function and constrained programming problem is clear. That is, when the feasible region $\mathcal{S}$ is not empty, the minimization with respect to $\mathbf{x}$ after the maximization with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ of Lagrangian function;

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left(\max_{\boldsymbol{\lambda}\in\mathbb{R}^{\ell},\ \boldsymbol{\mu}\in\mathbb{R}_{\geq 0}{}^{m}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right)$$

is equivalent to the constrained programming problem.

Note that the gradient vector and Hessian matrix, both with respect to variables $\mathbf{x}$, are given as:

$$\begin{aligned}\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \nabla f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i \nabla h_i(\mathbf{x}) + \sum_{j=1}^{m} \mu_j \nabla g_j(\mathbf{x}) \\ \nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \nabla^2 f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i \nabla^2 h_i(\mathbf{x}) + \sum_{j=1}^{m} \mu_j \nabla^2 g_j(\mathbf{x}).\end{aligned}$$

5.6.2 Conditions for Optimality

This subsection introduces the Karush-Kuhn-Tucker condition, called KKT conditions for short. They are necessary conditions for optimality. In addition second-order necessary conditions and second-order sufficient conditions for optimality are given.

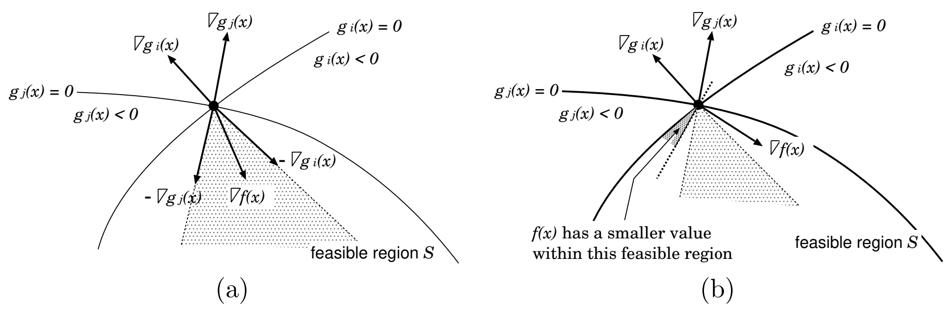


Figure 5.10: Illustration of the Karush-Kuhn-Tucker condition. (a) Lagrangian multiplier $\boldsymbol{\mu}$ exists. (b) Lagrangian multiplier does not exist.

Definition 5.9. When a constraint holds with equality, then the constraint is called *active constraint*.

For example, suppose there exists a constraint $g_j(\mathbf{x}) \leq 0$ and $g_j(\mathbf{a}) = 0$ holds at a point $\mathbf{x} = \mathbf{a} \in \mathcal{S}$. Then the constraint $g_j(\mathbf{x}) \leq 0$ is called active at $\mathbf{x} = \mathbf{a}$.

Proposition 5.13 (First order necessary condition (Karush-Kuhn-Tucker condition))**.** If $\mathbf{x}^*$ is a local optimum point, and if gradients of all active constraints at $\mathbf{x}^*$ are linearly independent (see Appendix A), then there exists $\boldsymbol{\lambda}^* = (\lambda_1^*, \cdots, \lambda_\ell^*)^{\mathrm{t}}$ and $\boldsymbol{\mu}^* = (\mu_1^*, \cdots, \mu_m^*)^{\mathrm{t}}$ which satisfy the following.

$$\begin{cases} \nabla f(\mathbf{x}^*) + \displaystyle\sum_{i=1}^{\ell} \lambda_i^* \nabla h_i(\mathbf{x}^*) + \sum_{j=1}^{m} \mu_j^* \nabla g_j(\mathbf{x}^*) = \mathbf{0} \\ h_i(\mathbf{x}^*) = 0, \qquad \text{for } i \in \{1, \cdots, \ell\} \\ g_j(\mathbf{x}^*) \leq 0, \qquad \text{for } j \in \{1, \cdots, m\} \\ \mu_j^* \geq 0, \qquad \text{for } j \in \{1, \cdots, m\} \\ \mu_j^* \cdot g_j(\mathbf{x}^*) = 0, \quad \text{for } j \in \{1, \cdots, m\} \end{cases}$$

When gradient vectors of all active constraints are not linearly independent, Lagrangian multiplier vector does not always exist. Condition for guaranteeing the existence of Lagrangian multiplier vector is called "constraint qualification". Regularity condition (gradient vectors of all active constraints at a local optimum point are linearly independent) is one of sufficient condition for the constraint qualification.

The following Fig.5.10 demonstrates two different situations, one satisfies the Karush-Kuhn-Tucker condition, the other does not. Here we are considering a point where two inequality-constraints are "active".

The first equation in the Karush-Kuhn-Tucker condition is rewritten as,

$$\nabla f(\mathbf{x}^*) \quad = \quad \sum_{i=1}^{\ell} \lambda_i^* \left(-\nabla h_i(\mathbf{x}^*)\right) + \sum_{j=1}^{m} \mu_j^* \left(-\nabla g_j(\mathbf{x}^*)\right),$$

which means that the gradient vector $\nabla f(\mathbf{x}^*)$ is in a cone spanned by negatively weighted gradient vectors $-\nabla g_j(\mathbf{x}^*)$ of active constraints (Fig.5.10(a)). Compared with $\mathbf{x}^*$, points which have smaller $f(\mathbf{x})$ than $f(\mathbf{x}^*)$ are all in the other side of the plane which is orthogonal to $\nabla f(\mathbf{x}^*)$, and they are all outside of the feasible region. As a result (under the first-order approximation model), we can not find feasible and better solutions around $\mathbf{x}^*$.

On the other hand, if the gradient vector $\nabla f(\mathbf{x}^*)$ is out of the cone spanned by negatively weighted gradient vectors $-\nabla g_j(\mathbf{x}^*)$ of active constraints (Fig.5.10(b)), the other side of the plane which is orthogonal to $\nabla f(\mathbf{x}^*)$ intersects with the feasible region. So, we can always find a feasible and better solution within any neighbor-set of $\mathbf{x}^*$.

Next, the second-order conditions are introduced.

Not all the inequality constraints are active. Let $\mathcal{J}$ denote the set of indexes of all active inequality constraints:

$$\mathcal{J}(\mathbf{x}^*) \quad \triangleq \quad \{j | g_j(\mathbf{x}^*) = 0\},$$

and we assume that gradient vectors of those active constraints are linearly independent.

Also, the set of all vectors orthogonal to all gradient vectors of active constraints is denoted by

$$\mathcal{M} \quad \triangleq \quad \left\{ \mathbf{y} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \nabla h_i(\mathbf{x}^*)^{\mathrm{t}}\mathbf{y} = 0 \text{ for every equality constraint} \\ \text{and } \nabla g_j(\mathbf{x}^*)^{\mathrm{t}}\mathbf{y} = 0 \text{ for } j \in \mathcal{J}(\mathbf{x}^*) \end{array} \right\}$$

Theorem 5.1 (Second order necessary condition)**.** Let $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be Lagrangian multiplier vector which satisfies Karush-Kuhn-Tucker condition with respect to a local **minimum** point $\mathbf{x}^*$. Then:

$$\mathbf{y}^{\mathrm{t}}\nabla_x^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\mathbf{y} \geq 0$$

holds for all $\mathbf{y} \in \mathcal{M}$.

For a local **maximum**, the inequality becomes $\leq$.

Theorem 5.2 (Second order sufficient condition)**.** Assuming that $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ satisfies Karush-Kuhn-Tucker condition, if

$$\mathbf{y}^{\mathrm{t}}\nabla_x^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\mathbf{y} > 0$$

holds for all $\mathbf{y} \in \mathcal{M}$ except $\mathbf{y} \neq \mathbf{0}$, then $\mathbf{x}^*$ is a local **minimum** point.

For a local **maximum**, the inequality becomes $<$.

In the following example, a possible solution has already been found. Use the first-order and second-order conditions to determine if it is locally optimal.

Example 5.10. *Condition checking.* Consider the following problem.

$$\begin{array}{rrl} \textbf{Minimize} & f(\mathbf{x}) = & -2x_1x_2^2 \\ \textbf{subject to} & g_1(\mathbf{x}) = & \frac{1}{2}x_1^2 + x_2^2 - \frac{3}{2} \leq 0 \\ & g_2(\mathbf{x}) = & -x_1 \leq 0 \\ & g_3(\mathbf{x}) = & -x_2 \leq 0 \end{array} \tag{5.36}$$

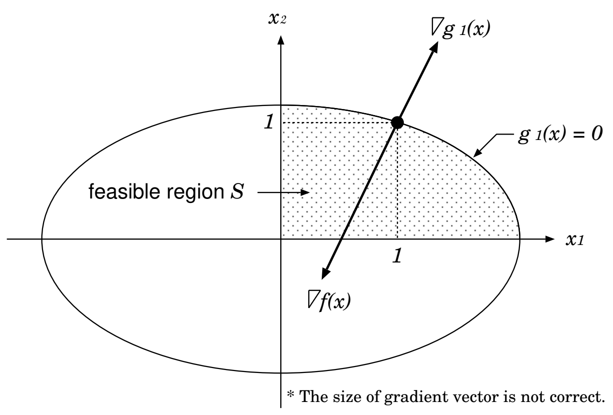


Figure 5.11: Example 1.

Assume that a candidate solution $\mathbf{x}^* = (1,1)^{\mathrm{t}}$ has already been found, and we need to check if it is optimal or not.

Calculate the gradients of the objective function and the active constraint:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -2x_2^2 \\ -4x_1x_2 \end{bmatrix} \qquad \nabla g_1(\mathbf{x}) = \begin{bmatrix} x_1 \\ 2x_2 \end{bmatrix}$$
$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 0 & -4x_2 \\ -4x_2 & -4x_1 \end{bmatrix} \qquad \nabla g_1^2(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

We need to find the active constraints. At this $\mathbf{x}^*$, $g_1(1,1) = 0$, $g_2(1,1) = -1$, $g_3(1,1) = -1$. Since only g_1 is satisfied with equality, only the first constraint is an active constraint, and:

$$\mathcal{J}(\mathbf{x}^*) = \{1\}.$$

Calculate the gradients of the objective function and the active constraint:

$$\nabla f(\mathbf{x}) = (-2x_2^2, -4x_1x_2)^{\mathrm{t}} \qquad \nabla g_1(\mathbf{x}) = (x_1, 2x_2)^{\mathrm{t}}$$
$$\nabla f(\mathbf{x}^*) = (-2,-4)^{\mathrm{t}} \qquad \nabla g_1(\mathbf{x}^*) = (1,2)^{\mathrm{t}}$$

To apply the second-order conditions, we need to know the Lagrange multiplier $\boldsymbol{\mu}^*$ corresponding to $\mathbf{x}^* = (1,1)^{\mathrm{t}}$:

$$\nabla f(\mathbf{x}^*) + \sum_{j=1}^{3} \mu_j^* \nabla g_j(\mathbf{x}^*) = 0$$
$$\begin{pmatrix} -2 \\ -4 \end{pmatrix} + \mu_1^* \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \mu_2^* \cdot \begin{pmatrix} -1 \\ 0 \end{pmatrix} + \mu_3^* \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} = 0$$

Note that $\mu_i^* = 0$ for an inactive constraint. Then, $\mu_1^* = 2$ and $\boldsymbol{\mu}^* = (2,0,0)^{\mathrm{t}}$.

The Lagrangian function is:

$$L(\mathbf{x},\boldsymbol{\mu}) \quad = \quad f(\mathbf{x}) + \mu_1 \cdot g_1(\mathbf{x}) + \mu_2 \cdot g_2(\mathbf{x}) + \mu_3 \cdot g_3\ (\mathbf{x}),$$

and its Hessian matrix evaluated at $(\mathbf{x}^*, \boldsymbol{\mu}^*) = (1, 1, 2, 0, 0)$ is

$$\begin{aligned}\nabla_x^2 L(\mathbf{x}^*, \boldsymbol{\mu}^*) &= \nabla^2 f(\mathbf{x}^*) + \mu_1^* \nabla^2 g_1(\mathbf{x}^*) \\ &= \begin{bmatrix} 0 & -4 \\ -4 & -4 \end{bmatrix} + 2 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -4 \\ -4 & 0 \end{bmatrix}.\end{aligned}$$

The set $\mathcal{M}$ is given as $\mathcal{M} = \{(y_1, y_2)^{\mathrm{t}} \mid y_1 + 2y_2 = 0\}$. It will be convenient to equivalently[1] express this as $\mathcal{M} = \{(-2t, t)^{\mathrm{t}} | t \in \mathbb{R}\}$.

With the above, we can now evaluate the second-order condition:

$$\begin{aligned}\mathbf{y}^{\mathrm{t}} \nabla_x^2 L(\mathbf{x}^*, \boldsymbol{\mu}^*) \mathbf{y} &= \begin{bmatrix} -2t & t \end{bmatrix} \begin{bmatrix} 2 & -4 \\ -4 & 0 \end{bmatrix} \begin{bmatrix} -2t \\ t \end{bmatrix} \\ &= 24t^2\end{aligned}$$

Since $24t^2 \geq 0$ it can be seen that the second-order necessary condition holds.

For the second-order sufficient condition, the case $\mathbf{y} = 0$ is excluded, so $24t^2 > 0$ holds for all $t \neq 0$. Thus $\mathbf{x}^* = (1, 1)$ is an optimal solution.

5.6.3 Exercises

1. Consider the following instance of the nonlinear optimization problem.

$$\begin{array}{rl} \text{Objective} & f(x_1, x_2) = \dfrac{1}{20} x_1^2 - x_1 x_2 \ \rightarrow \ \text{min.} \\ \text{Subject to} & g(x_1, x_2) \leq 0 \end{array}$$

Now, we have the following knowledge about the constraint function $g(x_1, x_2)$.

- $g(x_1, x_2)$ is a convex function.
- It has a minimum point O $(1.6, 0.9)$.
- It becomes zero at points A $(1, 1)$, B $(2, 1.2)$, and C $(2, 0.8)$. That is,

$$g(1, 1) = g(2, 1.2) = g(2, 0.8) = 0$$

- Its gradient vector has the following values at the points A, B, and C, respectively.

$$\begin{aligned}\nabla g(1, 1) &= (-0.25, -0.25)^T \\ \nabla g(2, 1.2) &= (0.25, 0.5)^T \\ \nabla g(2, 0.8) &= (0.5, -0.25)^T\end{aligned}$$

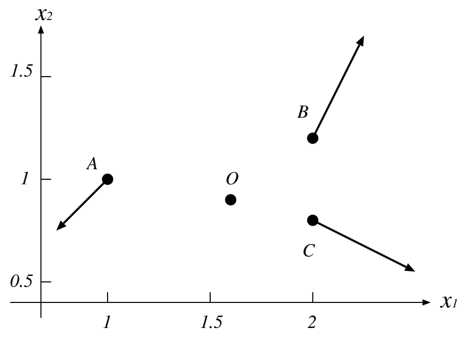


Illustration of the gradient vectors at points A, B, and C on x_1-x_2 plane.

[1]This is parameterization of the line. Let $t = y_2$, then $y_1 + 2y_2 = 0$ can be written as $y_1 = -2t$. The line is parameterized as $(y_1, y_2) = (-2t, t)$. Parameterization is not unique, for example $(y_1, y_2) = (t, -\frac{1}{2}t)$ is also possible. Wikipedia: Parametric equation

(1) Draw points (curved lines) which satisfy $g(x_1, x_2) = 0$. (Since we do not know the exact form of $g(x_1, x_2)$, we cannot draw them uniquely. But be sure to draw them so that the key features which these points must have are clearly recognizable.)

(2) What features on your curved lines drawn in (1) are due to the convexity of $g(x_1, x_2)$?

(3) What features on your curved lines drawn in (1) are due to the relation to the gradient vectors?

(4) Among points A, B, and C, are there points which satisfy the first-order necessary condition (Karush-Kuhn-Tucker condition)?

2. Now we consider the following objective function.

$$\text{Objective:} \qquad f(x_1, x_2) = {x_1}^2 + 4{x_2}^2 - 3x_1x_2 - x_1 - 2x_2 \ \rightarrow \ \text{min.}$$

(1) Considering it as an unconstrained optimization problem, find all stationary points.

(2) Using second-order conditions, verify whether each of stationary points obtained in (1) is an local optimum point or not.

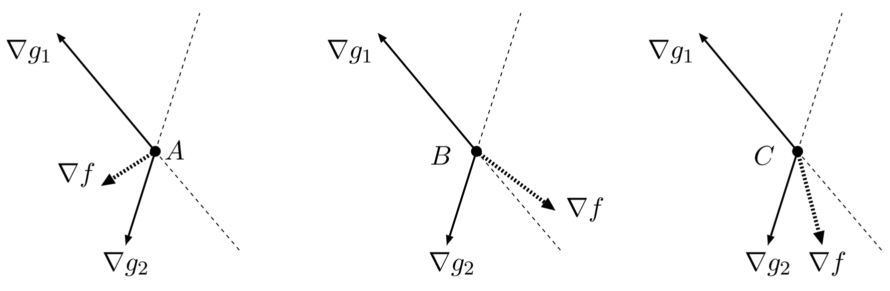
(3) Add one or more constraints so that every stationary point of $f(x_1, x_2)$ is excluded from the resultant feasible region. Note that the added constraint is limited to have the form of $g_j(\boldsymbol{x}) \leq 0$.

(4) With respect to the resultant constrained nonlinear optimization problem (the objective function $f(x_1, x_2)$ with constraints specified in (3)), find all candidates of the local optimum point by using the first-order condition.

3. Consider a constrained nonlinear optimization problem of the form;

$$\begin{aligned} \text{Objective;} \quad & f(x_1, x_2) \rightarrow \text{min.} \\ \text{Constraints;} \quad & g_1(x_1, x_2) \leq 0 \\ & g_2(x_1, x_2) \leq 0 \end{aligned}$$

The following figure shows gradient vectors of g_1, g_2, and f at points A, B, and C. Assume that two constraints are both **active** at A, B, and C.



For each point of A, B and C, discuss whether the first order necessary condition (Karush-Kuhn-Tucker condition) for a local optimum solution is satisfied or not.

4. Consider the following objective function $f(x_1, x_2)$ defined on the two-dimensional real space $\mathbb{R}^2$.

$$\text{Objective:} \quad f(x_1, x_2) = 3{x_1}^2 + {x_2}^2 - 6x_1x_2 + 12x_1 \quad \rightarrow \min$$

   (1) Find stationary point(s) of $f(x_1, x_2)$.

   (2) Using the second-order condition for a local optimum solution, decide whether the point found in (1) is a local optimum point or not.

   (3) Next we will consider the following optimization problem with a constraint. Using the first-order necessary condition (Karush-Kuhn-Tucker Condition), find the candidate(s) of a local optimum point **located on the boundary of the constraint**.

$$\begin{array}{ll} \text{Objective:} & f(x_1, x_2) = 3{x_1}^2 + {x_2}^2 - 6x_1x_2 + 12x_1 \quad \rightarrow \min \\ \text{Constraint:} & x_1 + x_2 \leq 9 \end{array}$$

   (4) Using the second-order condition, decide whether the candidate found in (3) is a local optimum point or not.

5. Consider the optimization problem $\mathcal{P}1$.

$$\boxed{\begin{array}{lll} \mathcal{P}1\text{:} & \text{Objective:} & f_1(\boldsymbol{x}) = \dfrac{N(\boldsymbol{x})}{D(\boldsymbol{x})} \quad \rightarrow \min \\ & & \text{where } D(\boldsymbol{x}) > 0 \end{array}}$$

Note that, $D(\boldsymbol{x}) > 0$ is for defining the domain of $f_1(\boldsymbol{x})$.

Problem $\mathcal{P}1$ can be transformed into Problem $\mathcal{P}1$'.

$$\begin{array}{lll} \mathcal{P}1\text{':} & \text{Objective:} & f_2(\boldsymbol{x}, y) = y \quad \rightarrow \min \\ & \text{Constraint:} & \dfrac{N(\boldsymbol{x})}{D(\boldsymbol{x})} \leq y \\ & & \text{where } D(\boldsymbol{x}) > 0 \end{array}$$

Further, Problem $\mathcal{P}1$' can be transformed into Problem $\mathcal{P}2$.

$$\boxed{\begin{array}{lll} \mathcal{P}2\text{:} & \text{Objective:} & f_2(\boldsymbol{x}, y) = y \quad \rightarrow \min \\ & \text{Constraint:} & g(\boldsymbol{x}, y) = N(\boldsymbol{x}) - y \cdot D(\boldsymbol{x}) \leq 0 \\ & & \text{where } D(\boldsymbol{x}) > 0 \end{array}}$$

Show the first-order necessary condition (KKT condition) for $\mathcal{P}1$ and the one for $\mathcal{P}2$, and explain the equivalency between them.

6. When we consider a nonlinear programming problem, a constraint;

$$|x_1 - x_2| \le 1 \quad \left(\text{that is,} \begin{cases} x_1 - x_2 \le 1 \\ x_2 - x_1 \le 1 \end{cases}\right)$$

and another form of constraint;

$$(x_1 - x_2)^2 \le 1$$

have the same meaning. Now we think about two programming problems;

$$\begin{array}{ll} \text{Objective:} & f(x_1, x_2) \to \text{min.} \\ \text{Constraint:} & x_1 - x_2 \le 1 \\ & x_2 - x_1 \le 1 \end{array} \quad \text{and} \quad \begin{array}{ll} \text{Objective:} & f(x_1, x_2) \to \text{min.} \\ \text{Constraint:} & (x_1 - x_2)^2 \le 1 \end{array}$$

Find the first order necessary conditions (KKT conditions) for these two problem instances, and discuss the equivalency between them.

5.7 Methods to Solve Nonlinear Problems

Recall the general nonlinear programming problem.

$$\begin{array}{rl} \textbf{Minimize} & f(\mathbf{x}) \\ \textbf{subject to} & h_i(\mathbf{x}) = 0, \ \ i = 1, 2, \ldots, \ell \\ & g_j(\mathbf{x}) \le 0, \ \ j = 1, 2, \ldots, m \end{array} \tag{5.37}$$

The following shows a typical procedure to solve nonlinear programming problem using the optimality conditions.

Generate Candidates Using the Karush-Kuhn-Tucker condition, enumerate candidates for optimal solutions.

Testing Candidates: Using the second-order conditions, test whether each candidate is:

- an optimal solution — the sufficient condition is satisfied
- not an optimal solution — the necessary condition is not satisfied, or
- un-decidable — other ad hoc approach is utilized to check candidates.

It must be determined if each inequality constraint is active or inactive. If there are a large number of inequality constraints, it may be necessary to take a brute-force approach, by trying various combinations of active and inactive, as shown int he following procedure:

1. For each inequality constraint, we assume it to be active or inactive. Let $\mathcal{I} \subseteq \{1, \cdots, m\}$ be a set of indices of active inequality constraints.

2. Solve the following simultaneous equations with respect to variables $\mathbf{x}$, $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_\ell)$ and $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_m)$.

$$\begin{cases} \nabla f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i \nabla h_i(\mathbf{x}) + \sum_{j=1}^{m} \mu_j \nabla g_j(\mathbf{x}) = \mathbf{0} \\ h_i(\mathbf{x}) = 0, \quad i \in \{1, \cdots, \ell\} \\ g_j(\mathbf{x}) = 0, \quad j \in \mathcal{I} \\ \mu_k = 0, \qquad k \in \{1, \cdots, m\} \backslash \mathcal{I} \end{cases}$$

3. Check whether the solution satisfies constraints and assumptions. That is;

$$\begin{array}{lll} \text{(a)} & \mu_j \geq 0 \ ? & j \in \mathcal{I} \\ \text{(b)} & g_k(\mathbf{x}) < 0 \ ? & k \in \{1, \cdots, m\} \backslash \mathcal{I} \end{array}$$

   • If all constraints and assumptions are satisfied, we output the solution as a point which satisfies the first-order necessary condition.
   • Otherwise, we reject the solution.
   Go to step 1, and try other active/inactive assumptions.

In some cases, Lagrangian multiplier and Karush-Kuhn-Tucker condition is used for finding an optimum point analytically, as shown in the following example.

Example 5.11. Consider the following problem.

$$\begin{array}{ll} \textbf{Minimize} & f(\mathbf{x}) = \boldsymbol{c}^{\mathrm{t}}\mathbf{x} \\ & \text{where } \boldsymbol{c} \neq \mathbf{0} \\ \textbf{subject to} & \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 \leq 0 \end{array}$$

Note that $\mathbf{x}^{\mathrm{t}}\mathbf{x} = \sum_{i=1}^{n} x_i^2$.

(1) Assume that the constraint is not an active constraint at the point where we are now considering, that is,

$$g(\mathbf{x}) = \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 < 0$$

Then, Karush-Kuhn-Tucker condition is written as

$$\begin{cases} \nabla f(\mathbf{x}) = \mathbf{0} \\ \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 < 0 \end{cases}$$

Because $\nabla f(\mathbf{x}) = \mathbf{c}$, it is clear that $\nabla f(\mathbf{x}) \neq 0$ since $\mathbf{c} \neq 0$ was given. Thus, the problem is unbounded. The constraint is needed for a bounded solution and so the constraint is active.

(2) Next we assume that the constraint is active, that is,

$$g(\mathbf{x}) = \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 = 0$$

Then Karush-Kuhn-Tucker condition with the Lagrangian multiplier μ is written as

$$\begin{cases} \nabla f(\mathbf{x}) + \mu \nabla g(\mathbf{x}) = \boldsymbol{c} + 2\mu\mathbf{x} = \mathbf{0} \\ \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 = 0 \\ \mu \geq 0 \end{cases}$$

From the first equation, we have

$$\mathbf{x} = -\frac{1}{2\mu}\boldsymbol{c}.$$

Substituting it into the constraint, we have

$$\frac{1}{4\mu^2}\boldsymbol{c}^{\mathrm{t}}\boldsymbol{c} = 1$$

So, we can solve it with respect to μ. Since $\mu \geq 0$, we have

$$\mu = \frac{1}{2}\sqrt{\boldsymbol{c}^{\mathrm{t}}\boldsymbol{c}}$$

As a result, we have the point which satisfies Karush-Kuhn-Tucker condition as follows.

$$\mathbf{x} = -\frac{\boldsymbol{c}}{\sqrt{\boldsymbol{c}^{\mathrm{t}}\boldsymbol{c}}}$$

Note that Karush-Kuhn-Tucker condition is a necessary condition for a point to be a local optimum point. In this example, $\mathbf{x} = -\boldsymbol{c}/\sqrt{\boldsymbol{c}^{\mathrm{t}}\boldsymbol{c}}$ is actually a local optimum point of this problem.

Example 5.12.

$$\begin{array}{ll} \textbf{Minimize} & f(\mathbf{x}) = \mathbf{x}^{\mathrm{t}}\mathbf{Q}\mathbf{x} \\ \textbf{subject to} & \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 = 0 \end{array}$$

where $\mathbf{Q}$ is symmetric and positive definite.

The Karush-Kuhn-Tucker condition with the Lagrangian multiplier λ is written as

$$\begin{cases} 2\mathbf{Q}\mathbf{x} + 2\lambda\mathbf{x} = \mathbf{0} \\ \mathbf{x}^{\mathrm{t}}\mathbf{x} - 1 = 0 \end{cases}$$

The first equation $\mathbf{Q}\mathbf{x} = -\lambda\mathbf{x}$ is an eigenvalue problem, that is, solutions are eigenvalues $-\lambda$ and eigenvectors $\mathbf{x}$ of $\mathbf{Q}$. However, in order to satisfy the constraint, normalize the eigenvector so that $\boldsymbol{\xi}^{\mathrm{t}}\boldsymbol{\xi} = 1$; the eigenvalue is adjusted by the normalization constant.

$$\begin{cases} \mathbf{x} = \boldsymbol{\xi}_i & \boldsymbol{\xi}_i \text{ is the corresponding eigenvector, normalized so that } \boldsymbol{\xi}^{\mathrm{t}}\boldsymbol{\xi} = 1. \\ \lambda = -\psi_i, & \psi_i \text{ is an eigenvalue of } \mathbf{Q} \end{cases}$$

Substituting $\mathbf{x} = \boldsymbol{\xi}_i$ into the objective function and using $\boldsymbol{\xi}_i^{\mathrm{t}}\boldsymbol{\xi}_i = 1$, we have

$$f(\boldsymbol{\xi}_i) = \boldsymbol{\xi}_i^{\mathrm{t}}\mathbf{Q}\boldsymbol{\xi}_i = \psi_i\boldsymbol{\xi}_i^{\mathrm{t}}\boldsymbol{\xi}_i = \psi_i$$

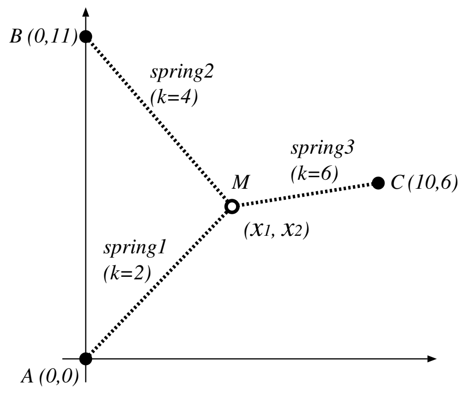
As a result, the minimum value of $f(\mathbf{x})$ under $\mathbf{x}^{\mathrm{t}}\mathbf{x} = 1$ is equal to the minimum eigenvalue of $\mathbf{Q}$, and the minimum point is equal to its corresponding (normalized) eigenvector.

5.7.1 Exercise

1. Using the first order necessary condition (Karush-Kuhn-Tucker condition), find all candidates of local optimum solutions for the following instance of the non-linear optimization problem.

$$\begin{array}{rl} \text{Objective;} & f(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 4)^2 \quad \to \quad min. \\ \text{Constraint;} & g(x_1, x_2) = x_1 + 2x_2 \leq 0 \end{array}$$

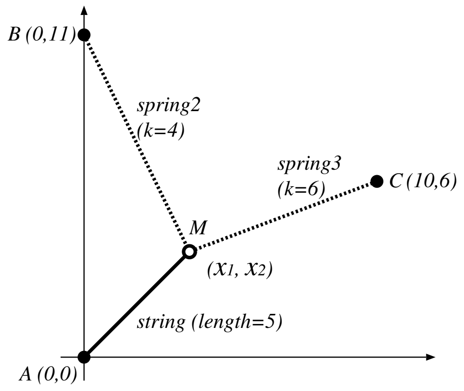
2. There are three ideal springs. One-ends of them are connected to one movable object M, and the other ends are connected to fixed points A at $(0, 0)$, B at $(0, 11)$ and C at $(10, 6)$, respectively. Assume that unloaded spring has its length 0, and the spring constants (k) of these springs are 2, 4, and 6, respectively.



(1) Each spring has the potential energy $k\ell^2/2$ where k is the spring constant, ℓ is the length of the spring (assuming the unloaded spring has the length 0). Let (x_1, x_2) be the position of the object M. Find the overall potential energy stored in these three springs.

(2) If no other force operates on the object M, the object moves to the position where the overall potential energy stored in these three springs becomes minimum and stays there. Find this minimum energy position of the object M.

(3) Now we replace "spring1" (connecting M and A) with a string with its length 5. Let (x_1, x_2) be the position of the object M again. Formulate the problem to minimize the overall potential energy stored in "spring2" and "spring3" as a nonlinear programming problem.



(4) Solve the nonlinear programming problem formulated in (3), and find a stationary position of the object M.

3. Consider the following nonlinear optimization problem.

$$\begin{array}{rl} \text{Objective:} & f(\boldsymbol{x}) = (x_1 - 4)^2 + (x_2 + 2)^2 \to min. \\ \text{Constraint:} & g_1(x_1, x_2) = x_1 - x_2 - 2 \leq 0 \end{array}$$

(1) Write down the first-order necessary condition (Karush-Kuhn-Tucker condition) for an optimum solution for this problem instance.

(2) Solve the first-order necessary condition, and find all candidates of a local optimum solution.

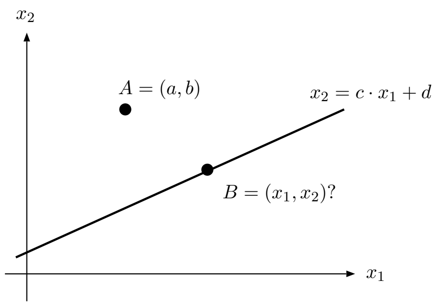
(3) Now we add the second condition (g_2 with a parameter k) in order to control the position of an optimum solution.

$$\begin{array}{rl} \text{Objective:} & f(\boldsymbol{x}) = (x_1 - 4)^2 + (x_2 + 2)^2 \to min. \\ \text{Constraint:} & g_1(x_1, x_2) = x_1 - x_2 - 2 \leq 0 \\ & g_2(x_1, x_2) = (x_1 - 1) + k(x_2 + 1) \leq 0 \end{array}$$

We want to let this problem have an optimum solution at $(x_1, x_2) = (1, -1)$. Find the range of k to realize it.

4. Consider two-dimensional space whose axes are x_1, x_2. There are a point A at $(x_1, x_2) = (a, b)$, and a straight line specified with $x_2 = c \cdot x_1 + d$, where a, b, c and d are fixed real values.

Usually, the distance between the point A and the line is defined by the Euclidean distance between A and a point B on the line, where B is the nearest point to A.



(1) Let (x_1, x_2) be the position of B, and formulate the problem to find the position of B as a Nonlinear Programming Problem.

(2) Find all possible candidates of B by using the first-order necessary condition (Karush-Kuhn-Tucker condition).

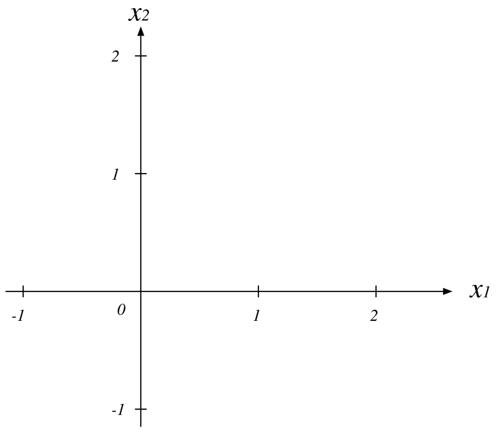
(3) It is well-known that the point B in this problem is the foot-point (cross-point) of the perpendicular drawn from A to the line $x_2 = c \cdot x_1 + d$. This phenomenon can be drawn directly from the Karush-Kuhn-Tucker Condition. Explain it.

5. Consider the following nonlinear programming problem.

$$\begin{aligned} &\text{Objective:} \quad f(x_1, x_2) = {x_1}^2 - x_1 x_2 + 2{x_2}^2 \ \rightarrow \text{min.} \\ &\text{Constraint:} \quad g_1(x_1, x_2) = -x_1 - x_2 + 1 \leq 0 \end{aligned}$$

(1) Using the first-order necessary condition (Karush-Kuhn-Tucker condition), find all candidates of optimum solution.

(2) Draw the following on a two-dimensional x_1-x_2 plane.
   (a) Feasible region of this problem.
   (b) Position (x_1^*, x_2^*) of each candidate of optimum solution obtained in (1).
   (c) Gradient vector (draw it like an arrow which has length and direction) of the objective function at (x_1^*, x_2^*).
   (d) Gradient vector of each active constraint function at (x_1^*, x_2^*).



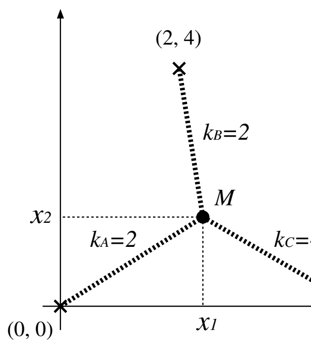
(3) Find the Hessian matrix of the objective function $f(x_1, x_2)$, and decide whether the Hessian matrix is “positive definite”, “positive semi-definite”, or neither of them.

6. There are three ideal springs A, B and C on a plane, and their spring constants are $k_A = 2$, $k_B = 2$ and $k_C = 4$, respectively. One-ends of these sprints are connected to a movable object M, and the other ends are fixed at the positions $(0, 0)$, $(2, 4)$, and $(5, 0)$, respectively.

(1) An ideal spring with its spring constant k stores the potential energy $\frac{k\ell^2}{2}$ when its length is ℓ.
Now we let (x_1, x_2) be the position of the object M. Then the total potential energy E_1 stored in this system is

$$E_1 = \{{x_1}^2 + {x_2}^2\} + \{(x_1 - 2)^2 + (x_2 - 4)^2\} + 2\{(x_1 - 5)^2 + {x_2}^2\}$$

Find the position of M, where the total potential energy E_1 is minimized.
(Note that the position of the object M can be explained by the balance of forces operating to this object M. The stable position of M can be explained also by the minimization of potential energy, and it provides identical solution with the one obtained from “force-balance”.)

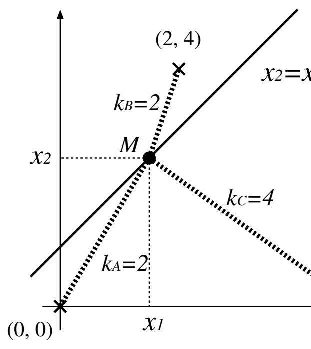


(2) Suppose that there is a straight rail which passes through the points $(0, 1)$ and $(3, 4)$, and M can move freely only on this rail.

The frictional force between M and the rail is assumed to be zero. In this case, the system has the identical total potential energy with E_1 in (1), but the position of M is constrained as

$$x_1 - x_2 + 1 = 0$$

Find the position of M, which makes the total potential energy of this system minimum. (It is enough to show the solution which satisfies the 1st order necessary condition.)



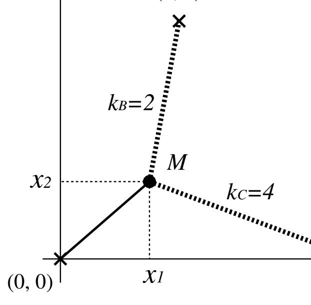
(3) The spring A is replaced with a string whose length is 2.
Since a string cannot store potential energy, the total potential energy of this system E_3 is

$$E_3 = \{(x_1 - 2)^2 + (x_2 - 4)^2\} + 2\{(x_1 - 5)^2 + {x_2}^2\}$$

On the other hand, the string will limit the position of M as

$${x_1}^2 + {x_2}^2 \le 2^2 = 4$$

Find the position of M, which makes the total potential energy E_3 of this system minimum. (It is enough to show the solution which satisfies the 1st order necessary condition.)



7. Advanced problem: Solve the following nonlinear programming problem, where a, b and c are positive real constants.

$$\begin{aligned}\text{Objective:} &= f(x_1,x_2,x_3) = {x_1}^a {x_2}^b {x_3}^c \rightarrow \text{max.}\\ \text{Constraint:} &= x_1+x_2+x_3 \le 1\\ & x_1 > 0\\ & x_2 > 0\\ & x_3 > 0\end{aligned}$$

8. Consider the following optimization problem:

$$\begin{array}{rl}\textbf{Minimize} & -\log(3+x_1)-\log(9+x_2)\\ \textbf{subject to} & x_1 + x_2 = 5\\ & x_1 \geq 0\\ & x_2 \geq 0\end{array}$$

   (a) Write the first-order KKT conditions.
   (b) Simplify the KKT conditions from part (a) by eliminating the μ_i variables.
   (c) Write an expression for x_1 and x_2 as a function of λ. It may be convenient to write $(x)^+ = \max(x,0)$.
   (d) Apply your answer from part (c) to the equality constraint to find λ, x_1^* and x_2^*.

5.8 Lagrange Dual Problem

The Lagrange dual problem associated with an original (primal) optimization problem (minimization problem) is another optimization problem (maximization problem) whose objective value is always less than or equal to the objective value of the primal problem (weak duality). In addition, under some assumption, the minimum objective value of the primal problem is equal to the maximum objective value of the dual problem (strong duality).

5.8.1 Derivation

Consider the following optimization problem (we call it “primal problem”).

$$\begin{array}{lll}\text{(P0)} & \min f(\mathbf{x}) & \\ & \text{subject to} & \mathbf{x}\in\mathbf{x}\\ & & h_i(\mathbf{x})=0,\quad i=1,\cdots,\ell\\ & & g_j(\mathbf{x})\le 0,\quad j=1,\cdots,m\end{array}$$

or, more simply;

$$\begin{array}{ll}\min f(\mathbf{x}) & \\ \text{subject to} & \mathbf{x}\in\mathbf{x}\\ & \boldsymbol{h}(\mathbf{x})=\mathbf{0}\\ & \boldsymbol{g}(\mathbf{x})\le\mathbf{0}\quad \text{(element-wise inequality)}\end{array}$$

At first, the primal problem is re-formulated using the Lagrangian function. Lagrangian function $L : \mathbb{R}^{n+\ell} \times \mathbb{R}^m_+ \to \mathbb{R}$ associated with this primal problem is defined as follows.

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f(\mathbf{x}) + \sum_{i=1}^{\ell} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{m} \mu_j g_j(\mathbf{x}), \quad (\text{where } \mu_j \geq 0,\ j = 1, \cdots, m) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{t}} \cdot \boldsymbol{g}(\mathbf{x}), \quad (\text{where } \boldsymbol{\mu} \geq \mathbf{0}) \end{aligned}$$

Now we try to maximize $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ with respect to λ_i, and μ_j.

$$\begin{aligned} \theta(\mathbf{x}) &= \max_{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^{\ell+m}, \boldsymbol{\mu} \geq \mathbf{0}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= \max_{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^{\ell+m}, \boldsymbol{\mu} \geq \mathbf{0}} \left\{ f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{t}} \cdot \boldsymbol{g}(\mathbf{x}) \right\} \end{aligned}$$

If $h_i(\mathbf{x}) \neq 0$ for some i, we can assign an infinitely large value (its polarity depends on the polarity of $h_i(\mathbf{x})$) to λ_i, and make $\theta(\mathbf{x})$ to have an infinitely large value. Similarly, if $g_j(\mathbf{x}) > 0$ for some j, we can assign an infinitely large positive value to μ_j, and make $\theta(\mathbf{x})$ to have an infinitely large value. Hence, $\theta(\mathbf{x})$ can be rewritten as follows.

$$\theta(\mathbf{x}) = \begin{cases} +\infty, & \text{if } \boldsymbol{h}(\mathbf{x}) \neq \mathbf{0} \text{ or } \boldsymbol{g}(\mathbf{x}) \not\leq \mathbf{0} \\ f(\mathbf{x}), & \text{if } \boldsymbol{h}(\mathbf{x}) = \mathbf{0} \text{ and } \boldsymbol{g}(\mathbf{x}) \leq \mathbf{0} \end{cases}$$

Note that, for $g_j(\mathbf{x}) < 0$, $\mu_j = 0$ is chosen in order to maximize $\mu_j g_{j(\mathbf{x})}$ over $\mu_j \geq 0$. As a result, $\theta(\mathbf{x})$ becomes $f(\mathbf{x})$ when $\boldsymbol{h}(\mathbf{x}) = \mathbf{0}$ and $\boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}$ are feasible.

Next we consider the problem to minimize $\theta(\mathbf{x})$ with respect to $\mathbf{x}$.

$$\min_{\mathbf{x} \in \mathbf{X}} \theta(\mathbf{x}) = \begin{cases} +\infty, & \text{if } \{\boldsymbol{h}(\mathbf{x}) = \mathbf{0},\ \ \boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}\} \text{ is infeasible in } \mathbf{x}. \\ \displaystyle\min_{\mathbf{x} \in \mathbf{X},\ \boldsymbol{h}(\mathbf{x}) = \mathbf{0},\ \boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}} f(\mathbf{x}), & \text{if } \{\boldsymbol{h}(\mathbf{x}) = \mathbf{0},\ \ \boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}\} \text{ is feasible in } \mathbf{x}. \end{cases}$$

As a result, the primal problem is rewritten as follows.

$$\begin{aligned} \text{(P1)} \quad \min_{\mathbf{x} \in \mathbf{X}} \theta(\mathbf{x}) &= \min_{\mathbf{x} \in \mathbf{X}} \left[\max_{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^{\ell+m},\ \boldsymbol{\mu} \geq \mathbf{0}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right] \\ &= \min_{\mathbf{x} \in \mathbf{X}} \left[\max_{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^{\ell+m},\ \boldsymbol{\mu} \geq \mathbf{0}} \left\{ f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{t}} \cdot \boldsymbol{g}(\mathbf{x}) \right\} \right] \end{aligned}$$

The Lagrange dual problem (D) is another optimization problem which is obtained from the primal problem by changing the order of "max" operation and "min" operation.

$$\text{(D)} \qquad \max_{\substack{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^{\ell+m}, \\ \boldsymbol{\mu} \geq \mathbf{0}}} \left[\min_{\mathbf{x} \in \mathbf{X}} \left\{ f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{t}} \cdot \boldsymbol{g}(\mathbf{x}) \right\} \right]$$

If we use

$$\min_{\mathbf{x} \in \mathbf{X}} \left\{ f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{t}} \cdot \boldsymbol{g}(\mathbf{x}) \right\} \triangleq \omega(\boldsymbol{\lambda}, \boldsymbol{\mu})$$

then, the Lagrange dual can be written as,

$$\begin{array}{lll}\text{(D)} & \max\limits_{\boldsymbol{\lambda},\boldsymbol{\mu}} \omega(\boldsymbol{\lambda},\boldsymbol{\mu}) & \\ & \text{subject to} & (\boldsymbol{\lambda},\boldsymbol{\mu}) \in \mathbb{R}^{\ell+m} \\ & & \boldsymbol{\mu} \geq \mathbf{0}\end{array}$$

5.8.2 Example 1

We will construct the Lagrange dual problem for the following primal problem.

$$\begin{array}{lll}\text{(P0)} & \min \boldsymbol{c}^{\mathrm{t}} \cdot \mathbf{x} & \\ & \text{subject to} & \boldsymbol{A}\mathbf{x} = \boldsymbol{b}, \quad \mathbf{x} \geq \mathbf{0}\end{array}$$

First of all, we will rewrite it into an standard form.

$$\begin{array}{lll}\text{(P0')} & \min \boldsymbol{c}^{\mathrm{t}} \cdot \mathbf{x} & \\ & \text{subject to} & \boldsymbol{A}\mathbf{x} - \boldsymbol{b} = \mathbf{0} \\ & & -\mathbf{x} \leq \mathbf{0}\end{array}$$

If we will write this primal problem with the Lagrangian function, we have,

$$\min_{\mathbf{x}} \left[\max_{\boldsymbol{\mu} \geq \mathbf{0}} \{L(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu})\} \right] = \min_{\mathbf{x}} \left[\max_{\boldsymbol{\mu} \geq \mathbf{0}} \left\{ \boldsymbol{c}^{\mathrm{t}} \cdot \mathbf{x} + \boldsymbol{\lambda}^{\mathrm{t}} (\boldsymbol{A}\mathbf{x} - \boldsymbol{b}) + \boldsymbol{\mu}^{\mathrm{t}} (-\mathbf{x}) \right\} \right]$$

The objective function $\omega(\boldsymbol{\lambda},\boldsymbol{\mu})$ for the Lagrange dual problem is given as,

$$\begin{aligned}\omega(\boldsymbol{\lambda},\boldsymbol{\mu}) &= \min_{\mathbf{x}} \left\{ \boldsymbol{c}^{\mathrm{t}} \cdot \mathbf{x} + \boldsymbol{\lambda}^{\mathrm{t}} (\boldsymbol{A}\mathbf{x} - \boldsymbol{b}) + \boldsymbol{\mu}^{\mathrm{t}} (-\mathbf{x}) \right\} \\ &= \min_{\mathbf{x}} \left\{ \left(\boldsymbol{c}^{\mathrm{t}} + \boldsymbol{\lambda}^{\mathrm{t}}\boldsymbol{A} - \boldsymbol{\mu}^{\mathrm{t}} \right) \mathbf{x} - \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{b} \right\} \\ &= \begin{cases} -\infty, & \text{if } \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} - \boldsymbol{\mu} \neq \mathbf{0} \\ -\boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{b}, & \text{if } \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0} \end{cases}\end{aligned}$$

(Note that, if at least one of the elements in $\boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} - \boldsymbol{\mu}$ is non-zero, we can make $\omega = -\infty$ by choosing $x_i = +\infty$ or $x_i = -\infty$.)

So, the Lagrange dual problem is given as following.

$$\begin{aligned}\text{(D)} \quad \max_{\boldsymbol{\mu} \geq \mathbf{0}} \omega(\boldsymbol{\lambda},\boldsymbol{\mu}) &= \max_{\boldsymbol{\mu} \geq \mathbf{0}} \left\{ \begin{array}{ll} -\infty & \text{for } \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}}\boldsymbol{\lambda} - \boldsymbol{\mu} \neq \mathbf{0}, \\ -\boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{b} & \text{for } \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}}\boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0} \end{array} \right\} \\ &= \begin{cases} -\infty & \text{if no } (\boldsymbol{\lambda},\boldsymbol{\mu}) \text{ satisfies } \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}}\boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0} \\ \max\limits_{\boldsymbol{\mu} \geq \mathbf{0}} \left(-\boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{b} \right) & \text{if } \{(\boldsymbol{\lambda},\boldsymbol{\mu}) | \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}}\boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0}\} \neq \emptyset \end{cases}\end{aligned}$$

If we ignore a trivial part, we further have the following.

$$\begin{array}{lll}\text{(D)} & \max\limits_{\boldsymbol{\lambda},\boldsymbol{\mu}} \left(-\boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{b} \right) & \\ & \text{subject to} & \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0} \\ & & \boldsymbol{\mu} \geq \mathbf{0}\end{array}$$

Now we can apply the following modifications on constraints.

$$\boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0}, \quad \boldsymbol{\mu} \geq \mathbf{0}$$
$$\Downarrow$$
$$\boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} = \boldsymbol{\mu} \geq \mathbf{0}$$
$$\Downarrow$$
$$\boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} \geq \mathbf{0}$$

Finally we have the following Lagrange dual problem.

$$\text{(D)} \qquad \max_{\boldsymbol{\lambda}} (-\boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{b}) \qquad \text{subject to} \quad \boldsymbol{c} + \boldsymbol{A}^{\mathrm{t}} \cdot \boldsymbol{\lambda} \geq \mathbf{0}$$

5.8.3 Properties

In the following, we will introduce $(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \boldsymbol{z}$, and use $\boldsymbol{z} \in \boldsymbol{Z}$ instead of $(\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^{\ell+m}$, $\boldsymbol{\mu} \geq \mathbf{0}$ when $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are not necessarily distinguished explicitly.

We consider the primal problem (P);

$$\text{(P)} \quad \min_{\mathbf{x} \in \mathbf{x}} \theta(\mathbf{x}), \quad \text{where} \quad \theta(\mathbf{x}) = \max_{\boldsymbol{z} \in \boldsymbol{Z}} L(\mathbf{x}, \boldsymbol{z})$$

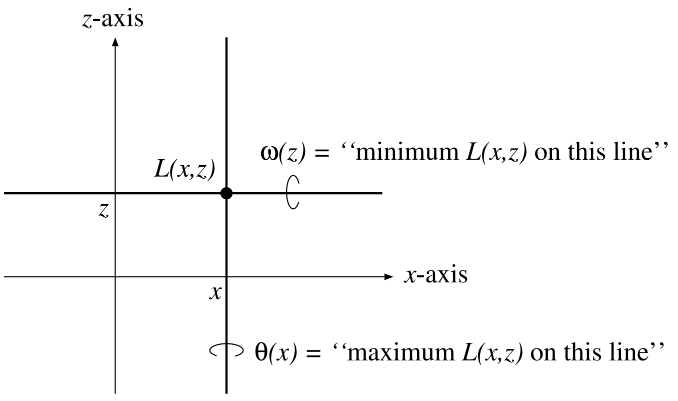
and its Lagrange dual problem (D);

$$\text{(D)} \quad \max_{\boldsymbol{z} \in \boldsymbol{Z}} \omega(\boldsymbol{z}), \quad \text{where} \quad \omega(\boldsymbol{z}) = \min_{\mathbf{x} \in \mathbf{x}} L(\mathbf{x}, \boldsymbol{z})$$

The following is a very basic property on L, θ and ω.

Lemma 3.

$$^{\forall}\mathbf{x} \in \mathbf{x}, \quad ^{\forall}\boldsymbol{z} \in \boldsymbol{Z}, \quad \omega(\boldsymbol{z}) \leq L(\mathbf{x}, \boldsymbol{z}) \leq \theta(\mathbf{x})$$



z-axis
L(x,z)
ω(z) = ‘‘minimum L(x,z) on this line’’
z
x-axis
x
θ(x) = ‘‘maximum L(x,z) on this line’’

Starting with this lemma, we have

$$^{\forall}\mathbf{x} \in \mathbf{x}, \qquad \max_{\boldsymbol{z} \in \boldsymbol{Z}} \omega(\boldsymbol{z}) \leq \theta(\mathbf{x})$$
$$^{\forall}\boldsymbol{z} \in \boldsymbol{Z}, \qquad \omega(\boldsymbol{z}) \leq \min_{\mathbf{x} \in \mathbf{x}} \theta(\mathbf{x})$$

and so the following theorem.

Theorem 5.4 (Weak duality).

$$\max_{\boldsymbol{z}\in\boldsymbol{Z}} \omega(\boldsymbol{z}) \quad \leq \quad \min_{\mathbf{x}\in\mathbf{x}} \theta(\mathbf{x})$$

In addition, for any feasible solution $\overline{\mathbf{x}}$ of the primal problem ($\boldsymbol{h}(\overline{\mathbf{x}}) = \mathbf{0}$, $\boldsymbol{g}(\overline{\mathbf{x}}) \leq \mathbf{0}$), and for $\overline{\boldsymbol{z}} \in \boldsymbol{Z}$,

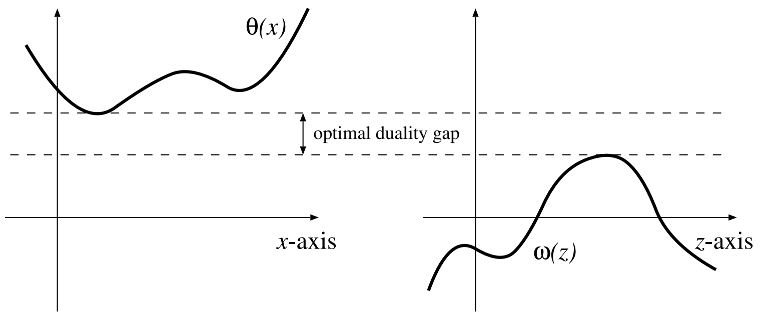
$$\max_{\boldsymbol{z}\in\boldsymbol{Z}} \omega(\boldsymbol{z}) \leq \theta(\overline{\mathbf{x}}) = f(\overline{\mathbf{x}}), \quad \omega(\overline{\boldsymbol{z}}) \leq \min_{\mathbf{x}\in\mathbf{x}} \theta(\mathbf{x})$$

holds.

From the weak duality, the difference of the minimum objective vale of the primal problem minus the maximum objective value of the dual problem;

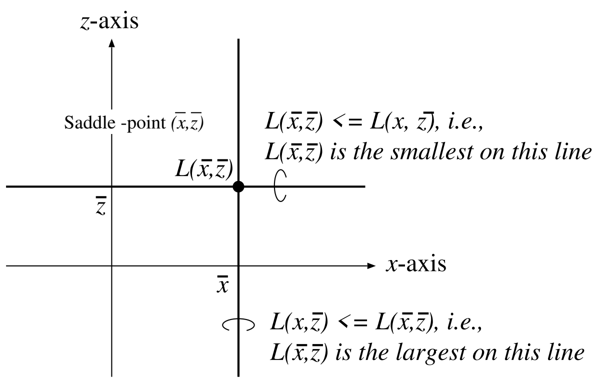
$$\eta \overset{\Delta}{=} \min_{\mathbf{x}\in\mathbf{x}} \theta(\mathbf{x}) - \max_{\boldsymbol{z}\in\boldsymbol{Z}} \omega(\boldsymbol{z})$$

has always non-negative value. This η is called the *optimal duality gap.*



Theorem 5.5. The optimal duality gap becomes zero if and only if the Lagrangian function has a saddle point. †For a point $(\overline{\mathbf{x}}, \overline{\boldsymbol{z}}) \in \mathbf{x} \times \boldsymbol{Z}$ satisfies the following, the point is called a saddle point of the Lagrangian function $L(\mathbf{x}, \boldsymbol{z})$.

$$^\forall\mathbf{x} \in \mathbf{x}, \ ^\forall\boldsymbol{z} \in \boldsymbol{Z}, \ L(\overline{\mathbf{x}}, \boldsymbol{z}) \leq L(\overline{\mathbf{x}}, \overline{\boldsymbol{z}}) \leq L(\mathbf{x}, \overline{\boldsymbol{z}})$$


z-axis
Saddle -point $(\bar{x},\bar{z})$
$L(\bar{x},\bar{z})$ <= $L(x, \bar{z})$, i.e.,
$L(\bar{x},\bar{z})$ is the smallest on this line
$L(\bar{x},\bar{z})$
$\bar{z}$
x-axis
$\bar{x}$
$L(x,\bar{z})$ <= $L(\bar{x},\bar{z})$, i.e.,
$L(\bar{x},\bar{z})$ is the largest on this line

Proof of Theorem 5.5 :

Suppose that $(\overline{\mathbf{x}}, \overline{\boldsymbol{z}})$ is a saddle point of L. From the definition of the saddle point, we have

$$\begin{aligned} L(\overline{\mathbf{x}}, \overline{\boldsymbol{z}}) &= \max_{\boldsymbol{z} \in \boldsymbol{Z}} L(\overline{\mathbf{x}}, \boldsymbol{z}) = \theta(\overline{\mathbf{x}}) \\ L(\overline{\mathbf{x}}, \overline{\boldsymbol{z}}) &= \min_{\mathbf{x} \in \mathbf{x}} L(\mathbf{x}, \overline{\boldsymbol{z}}) = \omega(\overline{\boldsymbol{z}}) \end{aligned}$$

and

$$\min_{\mathbf{x} \in \mathbf{x}} \theta(\mathbf{x}) \leq \theta(\overline{\mathbf{x}}) = \omega(\overline{\boldsymbol{z}}) \leq \max_{\boldsymbol{z} \in \boldsymbol{Z}} \omega(\boldsymbol{z})$$

On the other hand, from the weak duality,

$$\min_{\mathbf{x} \in \mathbf{x}} \theta(\mathbf{x}) \geq \max_{\boldsymbol{z} \in \boldsymbol{Z}} \omega(\boldsymbol{z})$$

holds. So, we have

$$\min_{\mathbf{x} \in \mathbf{x}} \theta(\mathbf{x}) = \max_{\boldsymbol{z} \in \boldsymbol{Z}} \omega(\boldsymbol{z})$$

which means zero optimal duality gap.

On the contrary, let $\mathbf{x}^*$ and $\boldsymbol{z}^*$ be the optimum point for (P) and the one for (D), respectively, and suppose that the optimal duality gap is 0, i.e., $\theta(\mathbf{x}^*) = \omega(\boldsymbol{z}^*)$. Then, immediately we have

$$L(\mathbf{x}^*, \boldsymbol{z}^*) \geq \min_{\mathbf{x} \in \mathbf{x}} L(\mathbf{x}, \boldsymbol{z}^*) = \omega(\boldsymbol{z}^*) = \theta(\mathbf{x}^*) = \max_{\boldsymbol{z} \in \boldsymbol{Z}} L(\mathbf{x}^*, \boldsymbol{z})$$

which means

$$^\forall \boldsymbol{z} \in \boldsymbol{Z},\ L(\mathbf{x}^*, \boldsymbol{z}) \leq L(\mathbf{x}^*, \boldsymbol{z}^*).$$

Similarly,

$$L(\mathbf{x}^*, \boldsymbol{z}^*) \leq \max_{\boldsymbol{z} \in \boldsymbol{Z}} L(\mathbf{x}^*, \boldsymbol{z}) = \theta(\mathbf{x}^*) = \omega(\boldsymbol{z}^*) = \min_{\mathbf{x} \in \mathbf{x}} L(\mathbf{x}, \boldsymbol{z}^*)$$

holds, which means

$$^\forall \mathbf{x} \in \mathbf{x},\ L(\mathbf{x}^*, \boldsymbol{z}^*) \leq L(\mathbf{x}, \boldsymbol{z}^*)$$

The pair of those two inequalities is just the definition of a saddle point.

The following are important theorems relevant to primal-dual problems with zero optimal duality gap.

Theorem 5.6. Consider an original optimization problem (P) with

- the objective function f is a convex function,
- equality constraint functions h_i are all linear functions, and

- inequality constraint functions g_j are all convex,
- $\mathbf{x} = \mathbb{R}^n$

The problem (P) has a saddle point of its Lagrangian function if and only if (P) has a point which satisfies Karush-Kuhn-Tucker condition.

Theorem 5.7 (Strong Duality)**.** Suppose that, for an original optimization problem, the objective function f and inequality constraint function g_j are differentiable and convex functions, and equality constraint function h_i is a linear function. In addition, suppose $\mathbf{x} = \mathbb{R}^n$. If the original (primal) problem has a minimum solution, and Slater's constraint qualification is satisfied, then its Lagrange dual problem has a maximum solution, and the optimal duality gap becomes 0.

†Slater's constraint qualification: Equality constraint functions h_i are all linear functions, and inequality constraint functions are all convex functions. In addition, there exists a point $\mathbf{x}^0$ which satisfies $\boldsymbol{h}(\mathbf{x}^0) = \mathbf{0}$ and $\boldsymbol{g}(\mathbf{x}^0) < \mathbf{0}$.

Theorem 5.8. Let $\boldsymbol{z}^* = (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \in \boldsymbol{Z}$ be a maximum solution for the dual problem, and suppose that the optimal duality gap is zero. If an optimization problem

$$\begin{aligned} &\min L(\mathbf{x}, \boldsymbol{z}^*) \\ &\text{subject to} \quad \mathbf{x} \in \mathbf{x} \end{aligned}$$

has a minimum solution $\mathbf{x}^*$ which satisfies

$$\begin{aligned} \boldsymbol{h}(\mathbf{x}^*) &= \mathbf{0} \\ \boldsymbol{g}(\mathbf{x}^*) &\leq \mathbf{0}, \end{aligned}$$

then $\mathbf{x}^*$ is a globally minimum solution of the original problem.

Dual problem has also the following properties.

(i) The dimension of unknown variables is identical to the number of constraints of the primal problem.

(ii) The feasible region $\boldsymbol{Z}$ for the dual problem has a simple form.

(iii) The dual problem is a convex optimization problem.

(iV) A solution of the dual problem provides the lower bound on the minimum solution for the primal problem.

5.8.4 Example-2

Consider a problem

$$\begin{aligned} &\min \|\boldsymbol{p}\|^2 \\ &\text{subject to} \quad 1 - a_i \left(\boldsymbol{p}^{\mathrm{t}} \boldsymbol{z}_i + b\right) \leq 0 \quad (i = 1, \cdots, m) \end{aligned}$$

with unknown variables $\boldsymbol{p} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, while $\boldsymbol{z}_i \in \mathbb{R}^n$ and $a_i \in \mathbb{R}$ $(i = 1, 2, \cdots, m)$ are fixed constants.

The Lagrangian function of this problem is given as,

$$\begin{aligned}
L(\boldsymbol{p}, b, \boldsymbol{\mu}) &= \|\boldsymbol{p}\|^2 + \sum_{i=1}^{m} \mu_i \left\{1 - a_i \left(\boldsymbol{p}^{\mathrm{t}} \boldsymbol{z}_i + b\right)\right\} \\
&= \boldsymbol{p}^{\mathrm{t}} \cdot \boldsymbol{p} - \boldsymbol{p}^{\mathrm{t}} \left(\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i\right) - \left(\sum_{i=1}^{m} \mu_i a_i\right) b + \sum_{i=1}^{m} \mu_i \\
&= \left\|\boldsymbol{p} - \frac{1}{2}\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i\right\|^2 - \frac{1}{4}\left\|\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i\right\|^2 - \left(\sum_{i=1}^{m} \mu_i a_i\right) b + \sum_{i=1}^{m} \mu_i
\end{aligned}$$

Hence, an objective function ω for the dual problem is

$$\begin{aligned}
\omega(\boldsymbol{\mu}) &= \min_{(\boldsymbol{p}, b) \in \mathbb{R}^{n+1}} L(\boldsymbol{p}, b, \boldsymbol{\mu}) \\
&= \min_{\boldsymbol{p} \in \mathbb{R}^n} \left\|\boldsymbol{p} - \frac{1}{2}\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i\right\|^2 - \min_{b \in \mathbb{R}} \left(\sum_{i=1}^{m} \mu_i a_i\right) b - \frac{1}{4}\left\|\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i\right\|^2 + \sum_{i=1}^{m} \mu_i
\end{aligned}$$

The minimization of the first term in the right hand side achieves the minimum value 0 with

$$\boldsymbol{p} = \frac{1}{2}\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i$$

With respect to the minimization of the second term in the right hand side, if $\sum_{i=1}^{m} \mu_i a_i \neq 0$ holds, then it takes the minimum value $-\infty$ by choosing $b = +\infty$ or $b = -\infty$. Our dual problem is to maximize ω, and hence it is enough to consider such $\boldsymbol{\mu}$ which satisfies $\sum_{i=1}^{m} \mu_i a_i = 0$, which makes the second term 0.

As a result, the third and the fourth terms remain in $\omega(\boldsymbol{\mu})$, and the dual problem can be written as

$$\begin{aligned}
\max \;\; \omega(\boldsymbol{\mu}) &= -\frac{1}{4}\left\|\sum_{i=1}^{m} \mu_i a_i \boldsymbol{z}_i\right\|^2 + \sum_{i=1}^{m} \mu_i \\
\text{subject to} \;\; & \sum_{i=1}^{m} \mu_i a_i = 0 \\
& \boldsymbol{\mu} \geq \mathbf{0}
\end{aligned}$$

This is a convex quadratic optimization problem with unknown variables $\boldsymbol{\mu}$.

5.8.5 Lagrangian Dual Based KKT Condition

We consider a nonlinear programming problem;

$$\begin{aligned}
& f(\mathbf{x}) \to \min \\
\text{subject to} \quad & \\
& \mathbf{x} \in \mathbf{x} \\
& h_i(\mathbf{x}) = 0, \quad i \in \{1, \cdots, \ell\} \\
& g_j(\mathbf{x}) \leq 0, \quad j \in \{1, \cdots, m\}
\end{aligned}$$

and let $\mathbf{x}^*$ be its (local) optimum solution. In spite of the original domain of the problem instance, we will consider a limited domain $\mathbf{x}^*$ which is a ball of $\mathbb{R}^n$ centered at $\mathbf{x}$.

Within the domain $\mathbf{x}^*$, $\mathbf{x}^*$ is the optimum solution, and we suppose that the strong duality holds. That is, there exists a dual optimum solution $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, and $\theta(\mathbf{x}^*) = \omega(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$.

Since $\mathbf{x}^*$ is a feasible solution for the primary problem, we have the following.

$$\begin{aligned}\theta(\mathbf{x}^*) &= \max_{\boldsymbol{\lambda},\boldsymbol{\mu}\geq\mathbf{0}} \left\{ f(\mathbf{x}^*) + \boldsymbol{\lambda}^{\mathrm{t}} \cdot \boldsymbol{h}(\mathbf{x}^*) + \boldsymbol{\mu}^{\mathrm{t}} \cdot \boldsymbol{g}(\mathbf{x}^*) \right\} \\ &= f(\mathbf{x}^*)\end{aligned}$$

We have the last equality from the feasibility of $\mathbf{x}^*$ *(*$\boldsymbol{h}(\mathbf{x}^*) = \mathbf{0}$*, and* $\boldsymbol{g}(\mathbf{x}^*) \leq \mathbf{0}$*) and the maximization operation with respect to* $\boldsymbol{\lambda}$ *and* $\boldsymbol{\mu}$*.*

On the other hand, with respect to the dual problem,

$$\begin{aligned}\omega(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) &= \min_{\mathbf{x}\in\mathbf{x}^*} \left\{ f(\mathbf{x}) + \boldsymbol{\lambda}^{*T} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{*T} \cdot \boldsymbol{g}(\mathbf{x}) \right\} \\ &\leq f(\mathbf{x}^*) + \boldsymbol{\lambda}^{*T} \cdot \boldsymbol{h}(\mathbf{x}^*) + \boldsymbol{\mu}^{*T} \cdot \boldsymbol{g}(\mathbf{x}^*) \\ &\leq f(\mathbf{x}^*)\end{aligned}$$

The last inequality comes from the feasibility of $\mathbf{x}^*$ *(*$\boldsymbol{h}(\mathbf{x}^*) = \mathbf{0}$*, and* $\boldsymbol{g}(\mathbf{x}^*) \leq \mathbf{0}$*), and the nonnegativeness of* $\boldsymbol{\mu}$*.*

From the strong duality $\theta(\mathbf{x}^*) = \omega(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, we have

$$\begin{aligned}&\min_{\mathbf{x}\in\mathbf{x}^*} \left\{ f(\mathbf{x}) + \boldsymbol{\lambda}^{*T} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{*T} \cdot \boldsymbol{g}(\mathbf{x}) \right\} \\ &= f(\mathbf{x}^*) + \boldsymbol{\lambda}^{*T} \cdot \boldsymbol{h}(\mathbf{x}^*) + \boldsymbol{\mu}^{*T} \cdot \boldsymbol{g}(\mathbf{x}^*) \\ &= f(\mathbf{x}^*)\end{aligned}$$

(1) The first equality shows that $\mathbf{x}^*$ is the optimum solution of the first minimization problem. Since $\mathbf{x}^*$ is an interior point of $\mathbf{x}^*$, $\mathbf{x}^*$ must satisfy the 1st order necessary condition for the (unconstraint) minimization problem. Hence, we have

$$\begin{aligned}&\nabla \left(f(\mathbf{x}) + \boldsymbol{\lambda}^{*T} \cdot \boldsymbol{h}(\mathbf{x}) + \boldsymbol{\mu}^{*T} \cdot \boldsymbol{g}(\mathbf{x}) \right) \\ &= \nabla f(\mathbf{x}^*) + \sum_{i=1}^{\ell} \lambda_i^* \cdot \nabla h_i(\mathbf{x}^*) + \sum_{j=1}^{m} \mu_j^* \cdot \nabla g_j(\mathbf{x}^*) \\ &= \mathbf{0} \qquad (5.42)\end{aligned}$$

(2) From the second equality, we have

$$\boldsymbol{\lambda}^{*T} \cdot \boldsymbol{h}(\mathbf{x}^*) + \boldsymbol{\mu}^{*T} \cdot \boldsymbol{g}(\mathbf{x}^*) = 0$$

Since $\mathbf{x}^*$ is feasible in the primary problem, $\boldsymbol{h}(\mathbf{x}^*) = \mathbf{0}$ and $\boldsymbol{g}(\mathbf{x}^*) \leq \mathbf{0}$ hold. Together with $\boldsymbol{\mu} \geq \mathbf{0}$, every individual term $\mu_j \cdot g_j(\mathbf{x}^*)$ should be zero. In conclusion,

$$h_i(\mathbf{x}^*) = 0, \quad i \in \{1, \cdots, \ell\} \qquad (5.43)$$

$$\mu_j \cdot g_j(\mathbf{x}^*) = 0, \quad j \in \{1, \cdots, m\} \qquad (5.44)$$

The set of equations (5.42), (5.43) and (5.44) is no other than KKT condition.

5.9 Appendix A: Constraint Qualification in Optimality Condition

In Karush-Kuhn-Tucker Condition, it is assumed that the gradients of all active constraints at an optimal point are linearly independent, which is called the "constraint qualification". Such constraint qualification guarantees Karush-Kuhn-Tucker condition to be a necessary condition for the optimality.

A primitive necessary condition for the optimality without any constraint qualification is the following "Fritz-John Condition".

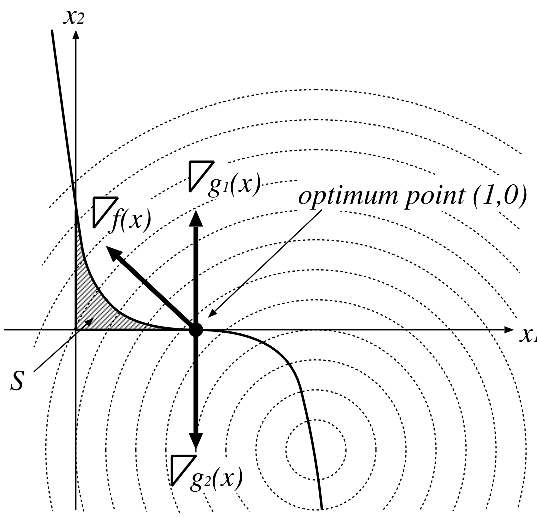
Fritz-John Condition:

$$\begin{aligned}&{}^{\exists}(u_0^*, u_1^*, \cdots, u_m^*) \neq (0, 0, \cdots, 0) \quad \text{such that} \\ &\quad u_0^*\nabla f(\mathbf{x}^*) + u_1^*\nabla g_1(\mathbf{x}^*) + \cdots u_m^*\nabla g_m(\mathbf{x}^*) = \mathbf{0} \\ &\quad g_j(\mathbf{x}^*) \leq 0 \quad (j = 1, 2, \cdots, m) \\ &\quad u_j^* g_j(\mathbf{x}^*) = 0 \quad (j = 1, 2, \cdots, m) \\ &\quad u_j^* \geq 0 \quad (j = 0, 1, 2, \cdots, m)\end{aligned}$$

In this condition, the case of $u_0^* = 0$ is not excluded. However, the optimarity evaluated with $u_0^* = 0$ is independent with the objective function f, and has less meaning for the problem itself. Karush-Kuhn-Tucker condition, however, is a revised version of Fritz-John condition with excluding $u_0 = 0$ at the cost of the introduction of the constraint qualification.

The following shows an example, in which the regularity condition for Karush-Kuhn-Tucker condition is not satisfied, and hence the optimality of the problem instance is not evaluated correctly by Karush-Kuhn-Tucker condition.

$$\begin{aligned}\text{Objective:} \quad & f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 + 1)^2 \quad \rightarrow \ \text{min.} \\ \text{Constraints:} \quad & g_1(\mathbf{x}) = (x_1 - 1)^3 + x_2 \leq 0 \\ & g_2(\mathbf{x}) = -x_1 \leq 0 \\ & g_3(\mathbf{x}) = -x_2 \leq 0\end{aligned}$$



In this example, $u_0\nabla f + u_1\nabla g_1 + u_2\nabla g_2 = \mathbf{0}$ is satisfied with $(u_0, u_1, u_2) = (0, 1, 1) \neq (0, 0, 0)$ (Fritz-John condition). However, no Lagrangian multiplier exists for Karush-Kuhn-Tucker condition.

5.10 Appendix B: Convergence of Steepest Descent Method

Here we discuss the speed of convergence of the steepest descent method. We assume that $\mathbf{x}^*$, the limit of a sequence of points generated by the steepest descent method, satisfies the second order sufficient condition for optimality. That is, the objective function f is twice differentiable, $\nabla f(\mathbf{x}^*) = \mathbf{0}$, and $\nabla^2 f(\mathbf{x}^*) \triangleq \boldsymbol{G}$ is positive definite.

Theorem 5.9. With respect to the sequence of points generated by the steepest descent method, for any $\varepsilon > 0$, there exists a positive integer K such that

$$^\forall k > K, \quad \left\|\mathbf{x}^{k+1} - \mathbf{x}^*\right\|_G \leq \left(\frac{\tau - 1}{\tau + 1} + \varepsilon\right) \left\|\mathbf{x}^k - \mathbf{x}^*\right\|_G$$

where

$$\|\boldsymbol{d}\|_G \triangleq \sqrt{\boldsymbol{d}^{\mathrm{t}}\boldsymbol{G}\boldsymbol{d}},$$

$$\tau = \frac{\psi_{max}}{\psi_{min}},$$

and ψ_{mas} and ψ_{min} are the largest and the smallest eigen values of $\boldsymbol{G}$, respectively.

The recurrence of the steepest descent method;

$$\mathbf{x}^{k+1} \quad = \quad \mathbf{x}^k - \alpha \cdot \nabla f(\mathbf{x}^k)$$

Value of $\nabla f(\mathbf{x}^k)$:

Using the first-order approximation of $\nabla f(\mathbf{x}^k)$, we have,

$$\begin{aligned}\nabla f(\mathbf{x}^k) &= \nabla f(\mathbf{x}^*) + \nabla(\nabla \boldsymbol{f}(\mathbf{x}^*)) \cdot \left(\mathbf{x}^k - \mathbf{x}^*\right) \\ &= \nabla f(\mathbf{x}^*) + \nabla^2 \boldsymbol{f}(\mathbf{x}^*) \cdot \left(\mathbf{x}^k - \mathbf{x}^*\right) \\ &\qquad (\text{since } \nabla f(\mathbf{x}^*) = \mathbf{0}, \text{ and } \nabla^2 f(\mathbf{x}^*) \triangleq \boldsymbol{G},) \\ &= \boldsymbol{G} \cdot \left(\mathbf{x}^k - \mathbf{x}^*\right)\end{aligned}$$

Find the analytic best choice of α;

$$\begin{aligned}f(\mathbf{x}^{k+1}) &= f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*) \cdot \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) + \frac{1}{2}\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^{\mathrm{t}} \boldsymbol{G} \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) \\ &\qquad (\text{since } \nabla f(\mathbf{x}^*) = \mathbf{0}) \\ &= f(\mathbf{x}^*) + \frac{1}{2}\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^{\mathrm{t}} \boldsymbol{G} \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) \\ &\qquad (\text{substitute } \mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \cdot \nabla f(\mathbf{x}^k)) \\ &= f(\mathbf{x}^*) + \frac{1}{2}\left((\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)) - \mathbf{x}^*\right)^{\mathrm{t}} \boldsymbol{G} \left((\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)) - \mathbf{x}^*\right)\end{aligned}$$

By solving $df(\mathbf{x}^{k+1})/d\alpha = 0$, we have,

$$\begin{aligned}\alpha &= \frac{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \left(\mathbf{x}^k - \mathbf{x}^*\right)}{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k)} \\ &\qquad (\text{since } \nabla f(\mathbf{x}^k) = \boldsymbol{G} \cdot \left(\mathbf{x}^k - \mathbf{x}^*\right)\ ) \\ &= \frac{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)}{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k)}\end{aligned}$$

Values of $f(\mathbf{x}^{k+1})$ and $f(\mathbf{x}^k)$:

Using the second-order approximation around $\mathbf{x}^*$, we have,

$$\begin{aligned}f(\mathbf{x}^{k+1}) &= f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^{\mathrm{t}} \cdot \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) + \frac{1}{2}\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) \\ &= f(\mathbf{x}^*) + \frac{1}{2}\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)\end{aligned}$$

Hence, we have,

$$\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) = 2\left(f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*)\right)$$

In the similar way, we have the following for $f(\mathbf{x}^k)$.

$$\left(\mathbf{x}^{k} - \mathbf{x}^*\right)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \left(\mathbf{x}^{k} - \mathbf{x}^*\right) = 2\left(f(\mathbf{x}^{k}) - f(\mathbf{x}^*)\right)$$

Value of $f(\mathbf{x}^{k+1})$:

Using the second-order approximation around $\mathbf{x}^k$, we have,

$$\begin{aligned}f(\mathbf{x}^{k+1}) &= f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^{\mathrm{t}} \left(-\alpha \cdot \nabla f(\mathbf{x}^k)\right) + \frac{1}{2}\alpha^2 \nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k) \\ &= f(\mathbf{x}^k) - \frac{1}{2}\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k)}\end{aligned}$$

Norm computation:

Combining above equations, we have,

$$\begin{aligned}\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)-\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right) &= 2\left(f(\mathbf{x}^{k+1})-f(\mathbf{x}^k)\right)\\ &= -\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k)}\end{aligned}$$

Finally, we have,

$$\begin{aligned}&\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)\\
&= \left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)-\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k)}\\
&= \left\{1-\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k)}\cdot\frac{1}{\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)}\right\}\cdot\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)\\
&= \left\{1-\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k)}\cdot\frac{1}{\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\boldsymbol{G}^{-1}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)}\right\}\cdot\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)\\
&\qquad \text{(note that } \boldsymbol{G} \text{ is symmetric)}\\
&= \left\{1-\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k)}\cdot\frac{1}{\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}^{\mathrm{t}}\boldsymbol{G}^{-1}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)}\right\}\cdot\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)\\
&\qquad \text{(substitute } \boldsymbol{G}(\mathbf{x}^k-\mathbf{x}^*)=\nabla f(\mathbf{x}^k)\text{)}\\
&= \left\{1-\frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k)}\cdot\frac{1}{\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}^{-1}\cdot\nabla f(\mathbf{x}^k)}\right\}\cdot\left(\mathbf{x}^{k}-\mathbf{x}^*\right)^{\mathrm{t}}\boldsymbol{G}\left(\mathbf{x}^{k}-\mathbf{x}^*\right)\end{aligned}$$

Details of the coefficient:

For a symmetric and positive definite matrix $\boldsymbol{G}$, we always have an orthogonal matrix $\boldsymbol{T}$ such that

$$\boldsymbol{T}^{\mathrm{t}}\boldsymbol{G}\boldsymbol{T}=\boldsymbol{\Psi}$$

where $\boldsymbol{\Psi}$ is a diagonal matrix whose diagonal entries are eigen values of $\boldsymbol{G}$, and

$$\boldsymbol{T}^{\mathrm{t}}\cdot\boldsymbol{T}=\mathbf{1}.$$

Without loss of generality, we let

$$\boldsymbol{\Psi}=\mathrm{diag}[\psi_1,\psi_2,\cdots,\psi_n]$$

and,

$$0<\psi_1\le\psi_2\le\cdots\le\psi_n.$$

(1) a part of the coefficient;

$$\begin{aligned}\nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{G}\cdot\nabla f(\mathbf{x}^k) &= \nabla f(\mathbf{x}^k)^{\mathrm{t}}\cdot\boldsymbol{T}\cdot\boldsymbol{T}^{\mathrm{t}}\boldsymbol{G}\cdot\boldsymbol{T}\cdot\boldsymbol{T}^{\mathrm{t}}\cdot\nabla f(\mathbf{x}^k)\\ &= \left(\boldsymbol{T}^{\mathrm{t}}\nabla f(\mathbf{x}^k)\right)^{\mathrm{t}}\cdot\boldsymbol{\Psi}\cdot\left(\boldsymbol{T}^{\mathrm{t}}\nabla f(\mathbf{x}^k)\right)\end{aligned}$$

Here we let

$$\boldsymbol{T}^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k) = (d_1, d_2, \cdots, d_n)^{\mathrm{t}}$$

Then, we have,

$$\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k) = \sum_{i=1}^{n} \psi_i {d_i}^2$$

(2) the second part of the coefficient;

$$\begin{aligned}
\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G}^{-1} \cdot \nabla f(\mathbf{x}^k) &= \nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \left(\boldsymbol{T}\boldsymbol{T}^{\mathrm{t}}\boldsymbol{G}\boldsymbol{T}\boldsymbol{T}^{\mathrm{t}}\right)^{-1} \cdot \nabla f(\mathbf{x}^k) \\
&= \nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{T} \cdot \left(\boldsymbol{T}^{\mathrm{t}}\boldsymbol{G}\boldsymbol{T}\right)^{-1} \cdot \boldsymbol{T}^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k) \\
&= \left(\boldsymbol{T}^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^{\mathrm{t}} \cdot \boldsymbol{\Psi}^{-1} \cdot \left(\boldsymbol{T}^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right) \\
&= \sum_{i=1}^{n} \frac{{d_i}^2}{\psi_i}
\end{aligned}$$

(3) the third part of the coefficient;

$$\begin{aligned}
\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^2 &= \left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{T} \cdot \boldsymbol{T}^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^2 \\
&= \left(\sum_{i=1}^{n} d_i^2\right)^2
\end{aligned}$$

Totally, we have,

$$\begin{aligned}
&\left\{1 - \frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^2}{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k)\right) \cdot \left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G}^{-1} \cdot \nabla f(\mathbf{x}^k)\right)}\right\} \\
&= 1 - \frac{\left(\sum_{i=1}^{n} {d_i}^2\right)^2}{\left(\sum_{i=1}^{n} \psi_i {d_i}^2\right)\left(\sum_{i=1}^{n} \frac{{d_i}^2}{\psi_i}\right)} \\
&= 1 - \frac{1}{\left(\sum_{i=1}^{n} \psi_i \cdot \frac{d_i^2}{\sum_{j=1}^{n} d_j^2}\right)\left(\sum_{i=1}^{n} \frac{1}{\psi_i} \cdot \frac{d_i^2}{\sum_{j=1}^{n} d_j^2}\right)}
\end{aligned}$$

Now we let

$$\frac{{d_i}^2}{\sum_{j=1}^{n} {d_j}^2} \triangleq \xi_i.$$

Note that $\xi_1 + \xi_2 + \cdots + \xi_n = 1$.

By using ξ_is, we have,

$$\begin{aligned}
&\left\{1 - \frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^2}{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k)\right) \cdot \left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G}^{-1} \cdot \nabla f(\mathbf{x}^k)\right)}\right\} \\
&= 1 - \frac{1}{\left(\sum_{i=1}^{n} \psi_i \xi_i\right)\left(\sum_{i=1}^{n} \frac{\xi_i}{\psi_i}\right)}
\end{aligned}$$

We will characterize the value of

$$\left(\sum_{i=1}^{n}\psi_i\xi_i\right)\left(\sum_{i=1}^{n}\frac{\xi_i}{\psi_i}\right)\triangleq\phi(\boldsymbol{\psi})$$

When we look at one eigen value ψ_p, $p\in\{2,3,\cdots,n-1\}$,

$$\begin{aligned}\phi(\boldsymbol{\psi}) &= \left(\psi_p {d_p}^2+A\right)\left(\frac{{d_p}^2}{\psi_p}+B\right)\\ &\leq \max\left\{\left(\psi_1 {d_p}^2+A\right)\left(\frac{{d_p}^2}{\psi_1}+B\right),\left(\psi_n {d_p}^2+A\right)\left(\frac{{d_p}^2}{\psi_n}+B\right)\right\}\end{aligned}$$

Based on this observation, we will truncate each ψ_i into either ψ_1 (the minimum eigen value) or ψ_n (the maximum eigen value) so that the result becomes no less than the original value. That is, let $Q\subseteq\{1,2,\cdots,n-1\}$ be the set of indexes of eigen values which are truncated into ψ_1, and

$$\begin{aligned}\eta_1 &= \sum_{i\in Q}\xi_i\\ \eta_n &= \sum_{i\in\{1,2,\cdots,n\}\setminus Q}\xi_i.\end{aligned}$$

Then,

$$\begin{aligned}\phi(\psi) &= \left(\sum_{i=1}^{n}\psi_i\xi_i\right)\left(\sum_{i=1}^{n}\frac{\xi_i}{\psi_i}\right)\\ &\leq \left(\sum_{i\in Q}\psi_1\xi_i+\sum_{i\in\{1,\cdots,n\}\setminus Q}\psi_n\xi_i\right)\left(\sum_{i\in Q}\frac{\xi_i}{\psi_1}+\sum_{i\in\{1,\cdots,n\}\setminus Q}\frac{\xi_i}{\psi_n}\right)\\ &= \left(\psi_1\eta_1+\psi_n\eta_n\right)\left(\frac{\eta_1}{\psi_1}+\frac{\eta_n}{\psi_n}\right)\\ &= {\eta_1}^2+{\eta_n}^2+\left(\frac{\psi_1}{\psi_n}+\frac{\psi_n}{\psi_1}\right)\eta_1\eta_n\\ &= \left(\eta_1+\eta_2\right)^2+\frac{\left(\psi_n-\psi_1\right)^2}{\psi_1\psi_n}\eta_1\eta_n\\ &\qquad \text{(use } \eta_1\eta_n\leq(\eta_1+\eta_n)^2/4\text{)}\\ &\leq \left(\eta_1+\eta_2\right)^2\left\{1+\frac{\left(\psi_n-\psi_1\right)^2}{4\psi_1\psi_n}\right\}\\ &\qquad \text{(use } \eta_1+\eta_n=\textstyle\sum_{i=1}^{n}\xi_i=1\text{)}\\ &= \frac{\left(\psi_n-\psi_1\right)^2}{4\psi_1\psi_n}\end{aligned}$$

Finally, we have,

$$\begin{aligned}
\left\{1 - \frac{\left(\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \nabla f(\mathbf{x}^k)\right)^2}{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G} \cdot \nabla f(\mathbf{x}^k)} \cdot \frac{1}{\nabla f(\mathbf{x}^k)^{\mathrm{t}} \cdot \boldsymbol{G}^{-1} \cdot \nabla f(\mathbf{x}^k)}\right\} &= 1 - \frac{1}{\phi(\boldsymbol{\psi})} \\
&\leq 1 - \frac{4\psi_1\psi_n}{\left(\psi_n + \psi_1\right)^2} \\
&= \left(\frac{\psi_n - \psi_1}{\psi_n + \psi_1}\right)^2
\end{aligned}$$

So we have the result;

$$\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^{\mathrm{t}} \boldsymbol{G} \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) \quad \leq \quad \left(\frac{\psi_n - \psi_1}{\psi_n + \psi_1}\right)^2 \cdot \left(\mathbf{x}^k - \mathbf{x}^*\right)^{\mathrm{t}} \boldsymbol{G} \left(\mathbf{x}^k - \mathbf{x}^*\right)$$

Chapter 6

Combinatorial Optimization

In *combinatorial optimization*, the feasible set is a **discrete set**, and often a **finite set**. We find the minimum value of an objective function over all discrete values in the feasible set. Wikipedia: Combinatorial Optimization

Since the number of elements in the feasible solutions is finite, we always find an optimum solution by enumerating all feasible solutions and evaluating (and comparing) values of the objective function for those feasible solutions. However it does not mean that any combinatorial optimization is easily solved. For example, a problem is supposed to be formulated with n variables $x_1, \cdots, x_n$ and each variable can be either 0 or 1. Then the number of feasible solutions is 2^n, and it can be too large to evaluate all those solutions with a practical computation time.

Hence we need to develop time and memory efficient algorithms for those combinatorial optimization problems. Unfortunately, there are many combinatorial optimization problems whose efficient algorithms are not known. (For those problems, essentially better algorithms than evaluating all feasible solutions are not known.)

6.1 Greedy Method

A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage.[1] In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time

Most algorithms follow a step-by-step procedure. In a greedy algorithm, at each step we make a locally optimal decision, without considering the global solution. For some problems, a greedy algorithm provides the optimal solution. One example is the simple change-making problem: Suppose you need to make change, say 36 cents. Given the coins have values 1,2, 5, 10, 20, 50 (a large number of each type of coin), how to make 36 cents of change using the minimum

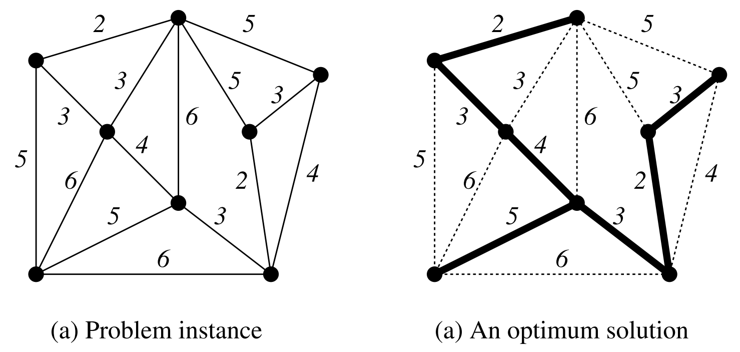


Figure 6.1: An instance of the minimum spanning tree problem, and its one of the optimum solutions.

number of coins? In general, greedy algorithms are not optimal, but in some cases may provide acceptable approximations of the optimal solution.

6.1.1 Minimum Spanning Tree

An example of a problem that can be solved by a greedy algorithm is finding the minimum-spanning tree.

Definition 6.1. Given an undirected graph $(\mathcal{V}, \mathcal{E})$, a *spanning tree* T is a subgraph that is a tree which include all nodes $\mathcal{V}$.

Let $\mathcal{E}_T \subseteq \mathcal{E}$ be the edges in the spanning tree. Then the spanning tree can be represented by $(\mathcal{V}, \mathcal{E}_T)$. In general, a graph may have multiple spanning trees. This chapter assumes that the graph is a *connected graph*, that is, each node is connected to at least one other node.

Definition 6.2. Given a **weighted** undirected graph, a *minimum spanning tree* (MST) is a spanning tree with minimum weight. Wikipedia: Minimum spanning tree

Let $\lambda(e)$ denote the weight of edge e and $\lambda > 0$. The cost or weight of a spanning tree $T(\mathcal{V}, \mathcal{E}_T)$ is:

$$C(T(V, E_T)) = \sum_{e \in \mathcal{E}_T} \lambda(e)$$

Problem 6.1. Given a weighted graph $(\mathcal{V}, \mathcal{E})$ with weights λ, find the minimum spanning tree $(\mathcal{V}, \mathcal{E}_T)$.

The minimum spanning tree problem can be solved exactly by a greedy method. Two algorithms, Kruskal's Algorithm and Prim's Algorithm find the minimum spanning tree.

Kruskal's Algorithm

<u>Kruskal Algorithm</u>

1. Sort edges with their weights, and let its result be $e_{I_1}, e_{I_2}, \cdots, e_{I_m}$. That is,
$$\lambda(e_{I_1}) \leq \lambda(e_{I_2}) \leq \cdots \leq \lambda(e_{I_m})$$
2. $E_T = \{e_{I_1}\}$, $k = 2$.
3. If the set of edges $E_T \cup \{e_{I_k}\}$ does not include cycles, then $E_T \leftarrow E_T \cup \{e_{I_k}\}$.
4. If (V, E_T) is a spanning tree of G, then quit ((V, E_T) is a cost-minimum spanning tree).
   Otherwise, $k \leftarrow k + 1$ and go to Step 3.

Wikipedia: Kruskal's Algorithm

Theorem 6.1. If a given problem instance G is connected, Kruskal Algorithm provides a cost-minimum spanning tree.

<u>**Proof**</u>

Given $G = (V, E)$ as a problem instance, let $T = (V, E_T)$, $E_T = \{e_i | 1 \leq i \leq n-1\}$ be the spanning tree obtained by Kruskal Algorithm, where the suffix of each edge represents the order of being chosen by the algorithm. Hence, we have $\lambda(e_1) \leq \lambda(e_2) \leq \cdots \leq \lambda(e_{n-1})$. Note that n is the number of vertices in G, and it is known that the number of edges in a spanning tree of a connected graph with n vertices is $n - 1$.

On the other hand, let $T^* = (V, E_T^*)$ be an optimum spanning tree of $G = (V, E)$. If there are multiple optimum spanning trees, we will take an optimum spanning tree that has the largest parameter value k defined in the following among all optimum spanning trees.

Now we assume that $E_T \neq E_T^*$, and let e_k is an edge with smallest suffix which satisfies $e_k \in E_T$ and $e_k \notin E_T^*$. That is,

$$\begin{aligned} &e_i \in E_T^* \text{ , for } i < k \\ &e_k \notin E_T^* \end{aligned}$$

Now we can see the following.

- Because of the edge selection rule of Kruskal Algorithm, we have
$$\lambda(e_k) \leq \lambda(e_\ell^*), \text{ for } e_\ell^* \in E_T^* \backslash \{e_1, e_2, \cdots, e_{k-1}\}$$
- Because of the edge selection rule of Kruskal Algorithm, edges $e_1, e_2, \cdots, e_k$ do not form cycles.
- Addition of e_k to T^* forms exactly one cycle, and the cycle includes e_k and at least one edge (let it be e_ℓ^*) from $E_T^* \backslash \{e_1, e_2, \cdots, e_{k-1}\}$ (Fig.6.2(b)).
  $\lambda(e_k^*) \leq \lambda(e_\ell^*)$ holds.

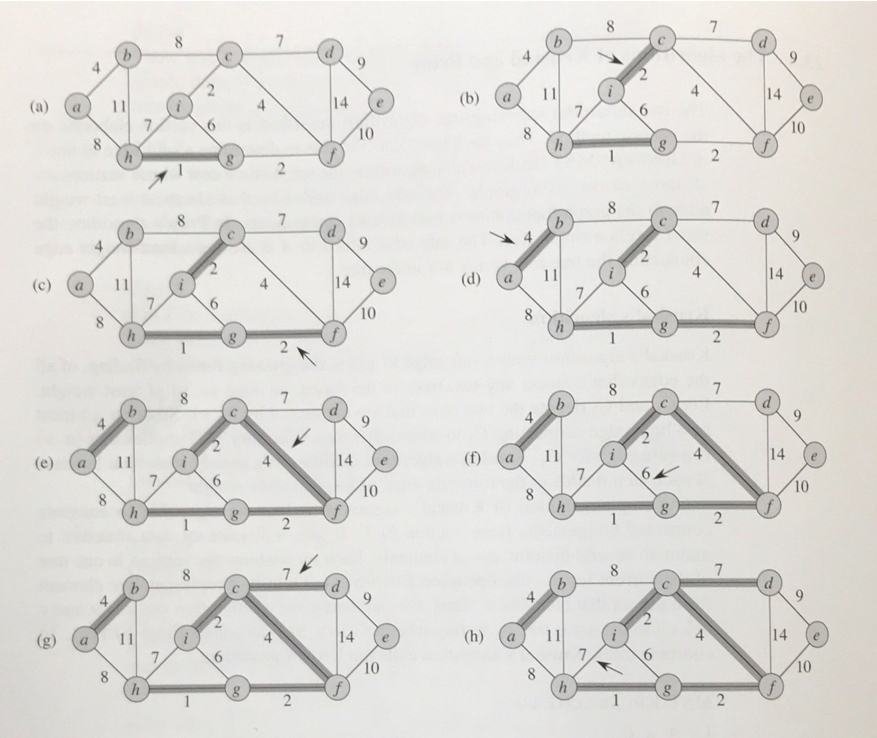


Figure 23.4 The execution of Kruskal's algorithm on the graph from Figure 23.1. Shaded edges belong to the forest A being grown. The algorithm considers each edge in sorted order by weight. An arrow points to the edge under consideration at each step of the algorithm. If the edge joins two distinct trees in the forest, it is added to the forest, thereby merging the two trees.

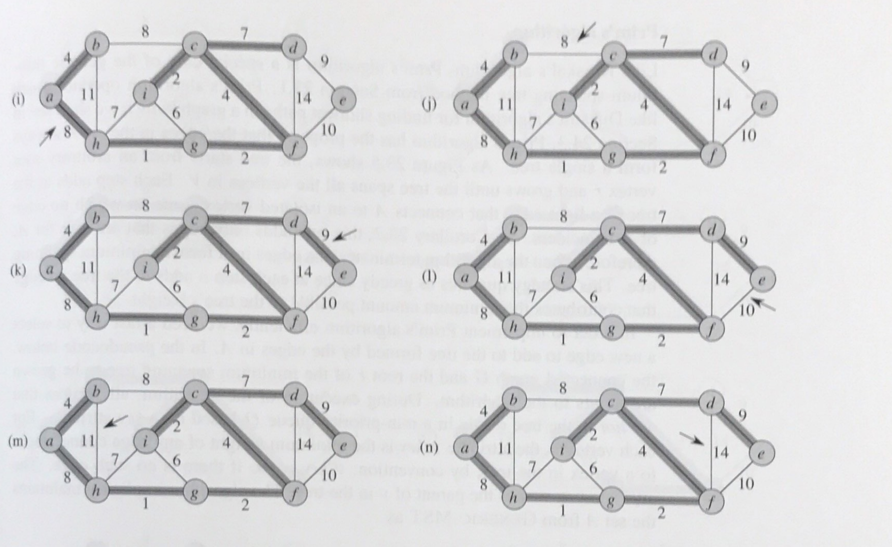


Figure 23.4, continued Further steps in the execution of Kruskal's algorithm.

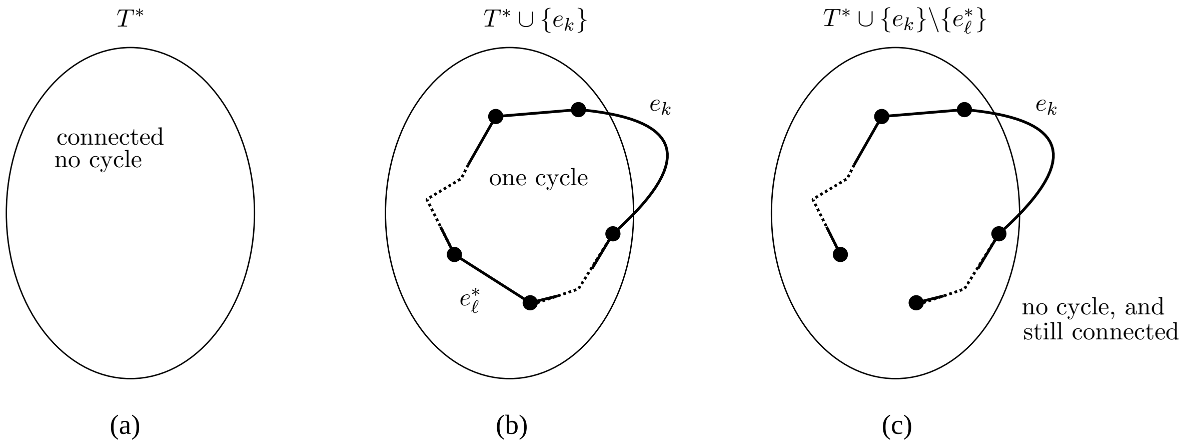


Figure 6.2: Illustration for the proof of Kruskal Algorithm

- $T' = (V, (E^*_T \cup \{e_k\}) \backslash \{e^*_\ell\})$ is again a spanning tree (Fig.6.2(c)), and

$$c(T') \quad = \quad c(T^*) + \lambda(e_k) - \lambda(e^*_\ell)$$

  Since $\lambda(e_k) \leq \lambda(e^*_\ell)$, we have $c(T') \leq c(T^*)$.

As a result,

1. If $c(T') < c(T^*)$, then it contradicts the assumption that T^* is an optimum spanning tree.
2. If $c(T') = c(T^*)$, then T' is also an optimum spanning tree, and $e_i \in E^{'}_T$, for $i \leq k$, which contradicts the choice of T^*.

Hence, $E_T = E^*_T$ holds.

□

Prim's Algorithm

Prim's Algorithm

1. Choose one vertex u arbitrary from a given problem instance $G = (V, E)$.
2. $E_0 = \emptyset$, $V_0 = \{u\}$, $k \leftarrow 1$.
3. Find a weight minimum edge, $\{p, q\}$, $p \in V_{k-1}$, $q \in V \backslash V_{k-1}$, among the cut separating V_{k-1} and $V \backslash V_{k-1}$ on G.
   $E_k = E_{k-1} \cup \{\{p, q\}\}$, $V_k = V_{k-1} \cup \{q\}$
4. If $k = |V| - 1$, then quit $((V_{|V|-1}, E_{|V|-1})$ is an cost-minimum spanning tree).
   Otherwise, $k \leftarrow k + 1$ and go to Step 3.

Theorem 6.2. If a given graph G is connected, Prime's Algorithm provide a cost-minimum spanning tree of G.

Proof

Given $G = (V, E)$ as a problem instance, let $T = (V, E_T)$, $E_T = \{e_i | 1 \le i \le n-1\}$, be the spanning tree obtained by Prime's algorithm, where the suffix of each edge in E_T represents the order of being selected in the execution of Prime's algorithm. Note that n is the number of vertices in G, and it is known that the number of edges of a spanning tree of a connected graph with n vertices is $n-1$. On the other hand, let $T^* = (V, E_T^*)$ be an optimum spanning tree on $G = (V, E)$. If there are multiple optimum spanning trees, we will take an optimum spanning tree that has the largest parameter value k defined in the following among all the optimum spanning trees. Note that $|E_T| = |E_T^*|$.

Now we assume that $E_T \neq E_T^*$, and let e_k be an edge with smallest suffix which satisfies $e_k \in E_T$ and $e_k \notin E_T^*$. That is;

$$\begin{aligned} &e_i \in E_T^* \text{ , for } i < k \\ &e_k \notin E_T^* \end{aligned}$$

In Prime's algorithm, e_k is chosen as an edge having smallest edge cost among the cut (let it be C_k) separating sets of vertices V_{k-1} and $V \backslash V_{k-1}$(Fig.6.3(a),(b)).

Now we can see the following;

- $(V_{k-1}, \{e_1, \cdots, e_{k-1}\})$ is a tree.
- Addition of e_k to T^* forms exactly one cycle, and the cycle includes e_k and one other edge (let it be e'_k) in C_k (Fig.6.3(c)). It is trivial that $e'_k \in E_T^*$.
- $(V, (E_T^* \cup \{e_k\}) \backslash \{e'_k\}) = T'$ is again a spanning tree (Fig.6.3(d)), and

  $$c(T') = c(T^*) + \lambda(e_k) - \lambda(e'_k)$$

  Since $\lambda(e_k) \le \lambda(e'_k)$ holds from the edge selection rule in Prime's algorithm, we have $c(T') \le c(T^*)$.

As a result,

1. If $c(T') < c(T^*)$, then it contradicts the assumption that T^* is an optimum spanning tree.
2. If $c(T') = c(T^*)$, then T' is again an optimum spanning tree which includes at least first k edges chosen in Prime's algorithm (that is, $e_i \in E'_T$, for $i \le k$), which contradicts the choice of T^*.

Hence, $E_T = E_T^*$ is satisfied.

□

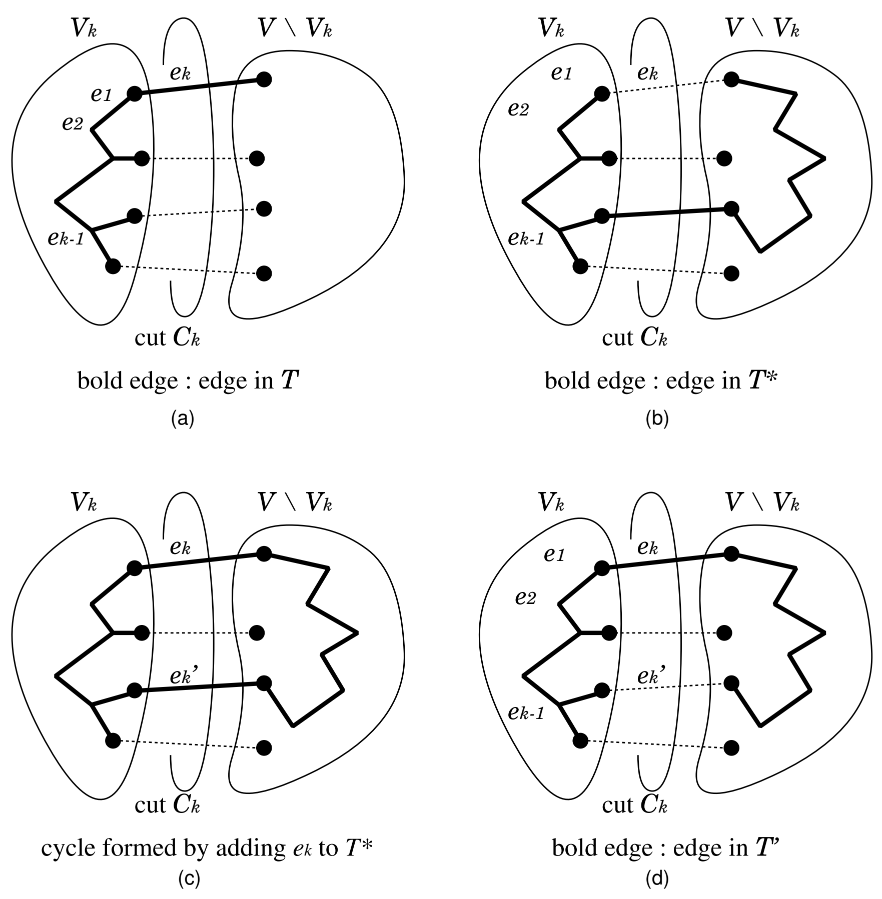


Figure 6.3: Illustration used for proving the correctness of Prime's Algorithm

6.1.2 Knapsack Problem

Problem 6.2. Given a set of n items, each with given weight a_i and utility $c_i \geq 0$, select the items to maximize the total utility while having total weight not greater than W.

In this case, there is only one of each item, that is $x_i \in \{0, 1\}$, where x_i is the number of item i in the knapsack.

Using the following example

$$\begin{aligned} n &= 4 \\ (c_1, c_2, c_3, c_4) &= (7, 8, 1, 2) \\ (a_1, a_2, a_3, a_4) &= (4, 5, 1, 3) \\ W &= 6 \end{aligned}$$

solve the problem using a greedy approach.

We will formulate the knapsack problem based on the framework of 0-1 programming problem. Now we suppose that $x_i = 0$ represents that m_i is not packed into the knapsack, while $x_i = 1$ represents that m_i is packed into the knapsack.

Clearly, the utility per weight $\frac{a_i}{c_i}$ is important for cost. In the absence of the 0-1 restriction, we would choose the items with the highest utility per weight. For convenience, the items have been ordered such that:

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \frac{c_3}{a_3} \geq \frac{c_4}{a_4}$$

holds.

A greedy method can be used to obtain a candidate solution. The greedy algorithm chooses items having larger utility per weight. Optimality is not guaranteed.

1. At first, no item is chosen, and 6 is left as knapsack capacity.
2. Item 1 has the largest "utility"/"weight". Since $a_1 = 4 \leq 6$, we pack item 1 into the knapsack ($x_1 = 1$). The remaining knapsack capacity is $6 - a_1 = 2$.
3. Item 2 has the second largest "utility"/"weight". Since $a_2 = 5 > 2$, we cannot pack m_2 ($x_2 = 0$).
4. Item 3 is next. SInce $a_3 = 1 \leq 2$, we pack item 3 into the knapsack ($x_3 = 1$). The remaining knapsack capacity is $2 - a_3 = 1$.
5. Finally, item 4. Since $a_4 = 3 > 1$, we cannot pack item 4 ($x_4 = 0$).

As a result, we have a candidate solution $\boldsymbol{x} = (1, 0, 1, 0)^{\mathrm{t}}$ which achieves the objective function value $z = 8$. Later, we will use and we will use this solution as the initial tentative solution for branch-and-bound.

6.1.3 Exercise

[1] Knapsack problem is the following problem: Given a set of items, each of which has its own "value" and "weight", and a number which represents the total weight limitation, find a combination of items (a subset of the given set of items) such that their total value is maximized while their total weight is no larger than the given limitation.

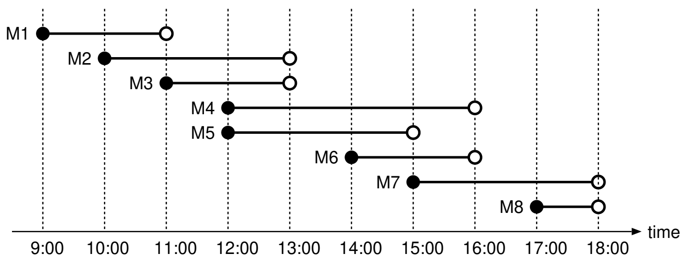
The following shows an instance of Knapsack problem, and we want to find a solution of this instance by applying a greedy method. Choose a priority metric for applying greedy method, and compute a greedy solution for the given instance.

Items	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
Value	5	9	1	3	6	2	5	3	7
Weight	2	3	1	2	3	2	4	2	4

Weight limit $W = 10$

[2] You are an executive of a company, and very busy. Today, as usual, many meetings are scheduled as they are shown in the following time table, where one meeting is denoted with • (start time), ◦ (end time) and a line segment connecting them. If you attend one meeting, you have to be there from the beginning, and you can not quit the meeting before its end. If the start time of one meeting is the same with the end time of another, you can attend both meetings. (Time needed for moving one meeting to another is negligible.)

Since many meetings overlap in time, you have to choose which meetings you attend, and which meetings you are absent.



(1) Assuming that you want to attend **as many meetings as possible**. Design a greedy algorithm to choose meetings you attend under this objective. (Assuming that a greedy algorithm checks meetings one after another, and decide attendance/absence to each meeting, design (1) the order of meetings to be checked, and (2) a criterion of attendance/absence.)

(2) Using the algorithm obtained in (1), decide attendance/absence of each meeting today.

(3) Does your greedy algorithm always compute an optimum solution (that is, the maximum number of non-overlapping meetings) for any instance of meeting schedule?

- · If your algorithm always computes an optimum solution, show its mathematical proof.
- · If not, show one counterexample (an instance of meeting schedule for which your algorithm fails to find an optimum solution). After that, design another greedy algorithm.

[3] We will consider the **relaxed real-valued** Knapsack Problem. There are n items, and each item m_i has its value c_i and its weight a_i. Variable x_i represents how much of m_i is put into the knapsack, that is, the weight $a_i \cdot x_i$ and the value $c_i \cdot x_i$ are put into the knapsack, where $x_i \in \mathbb{R}$ and $0 \le x_i \le 1$. The problem is now formulated as follows.

$$\begin{aligned}
\text{Objective:} \quad & -\sum_{i=1}^{n} c_i \cdot x_i \to \min \\
\text{Constraints:} \quad & \sum_{i=1}^{n} a_i \cdot x_i - W \le 0 \\
& -x_1 \le 0 \\
& \quad \vdots \\
& -x_n \le 0 \\
& x_1 - 1 \le 0 \\
& \quad \vdots \\
& x_n - 1 \le 0
\end{aligned}$$

(1) Show the Karush-Kuhn-Tucker condition for this problem.

(2) Show (prove) that the solution obtained by applying greedy algorithm based on the value/weight metric satisfies this Karush-Kuhn-Tucker condition.

6.2 Branch-and-Bound Method

Since a combinatorial optimization problem has a finite number of feasible solutions, an optimum solution can be found by enumerating and evaluating all feasible solutions. However, a naive enumeration has exponential complexity. In branch-and-bound, the computation time is shortened by skipping hopeless candidate solutions.

Branch-and-bound can be represented on a tree, where each level of the tree corresponds to a variable and each branch its decision. The algorithm explores branches of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and the branch is discarded if it cannot produce a better solution than the best one found so far by the algorithm. Wikipedia: Branch and bound

The algorithm depends on an efficient computation of lower and upper bounds. Selecting good upper and lower bounds is an important aspect of this algorithm.

Branch-and-Bound (Minimization Version)

0. Find a tentative solution by some appropriate method, and let z^* be the value of the objective function for this tentative solution. (The tentative solution and its corresponding tentative minimum value may be updated afterward.)

1. Generate subproblems $\mathcal{P}_1$, $\mathcal{P}_2$, $\cdots$, $\mathcal{P}_m$ from the original problem $\mathcal{P}_0$.
   $\mathcal{A} \leftarrow \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_m\}$

2. Choose one subproblem $\mathcal{P}_i$ from $\mathcal{A}$ and $\mathcal{A} \leftarrow \mathcal{A} \backslash \{\mathcal{P}_i\}$.

   2-1. If $\mathcal{P}_i$ has no feasible solution, then go to Step 4, otherwise go to Step 2-2.

   2-2. If the optimum solution of $\mathcal{P}_i$ is obtained, let z_i be the value of the objective function achieved by this solution, otherwise go to Step 2-3.
   When $z_i < z^*$, then update the tentative solution and $z^* \leftarrow z_i$.
   After that, go to Step 4.

   2-3. A lower bound $\underline{z_i}$ of the value of the objective function for solutions of $\mathcal{P}_i$ is computed.
   If $\underline{z_i} < z^*$, then go to Step 3.
   If $\underline{z_i} \geq z^*$, then go to Step 4.

3. Generate subproblems $\mathcal{P}_j$, $\cdots$, $\mathcal{P}_k$ from the subproblem $\mathcal{P}_i$, update $\mathcal{A}$ as $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{P}_j, \cdots, \mathcal{P}_k\}$, and go to Step 2.

4. If $\mathcal{A} = \emptyset$, then quit (the tentative solution at this time is an optimum (minimum) solution of the original problem $\mathcal{P}_0$).
   Otherwise, go to Step 2.

In an implementation of branch-and-bound method, the set A of subproblems is normally maintained with a stack (last in first out). That is, if we consider a rooted tree such that the root node represents a given problem instance, and children nodes of an internal node (a parent node of children nodes) represent subproblems generated from the (sub)problem corresponding to the parent node, the order of subproblems picked up at Step 2 is just the pre-order of nodes visited by Depth First Search started at the root node. The idea of branch-and-bound search is to eliminate fruitless searches of subtrees in Depth First Search for improving the time cost needed for searching for an optimum solution.

When we practically apply the idea of branch-and-bound to some combinatorial optimization problem, (1) the way to find an initial tentative solution

(Step 0) and (2) the way to compute a lower bound $\underline{z_i}$ of the value of the objective function achieved by solutions of each subproblem $\mathcal{P}_i$ (Step 2-3) must be devised. In addition, when we apply the method to a maximization problem, Step 2-3 must be arranged such as changing "lower bound" to "upper bound" and changing the criteria (larger or smaller) in the decision of continuing/eliminating subtree search.

6.2.1 Knapsack Problem

The branch-and-bound method may be used to solve the knapsack problem.

Upper bound

To find an upper bound, we use *relaxation.* In relaxation, instead of restricting the variables to be discrete, they are allowed to be continuous. For the knapsack problem, instead of $x_i \in \{0,1\}$, we allow $0 \le x_i \le 1$. Clearly, non-integer solutions are possible. But the resulting relaxed integer problem is easily solved, and the optimal value will be an **upper bound** on the solution to the integer-valued problem.

We will continue the example in Subsection 6.1.2. We can write the 0-1 programming problem as:

$$\begin{aligned} \textbf{Maximize}\quad & 7x_1 + 8x_2 + x_3 + 2x_4 \\ \textbf{subject to}\quad & 4x_1 + 5x_2 + x_3 + 3x_4 \le 6 \\ & x_1\ ,\quad x_2\ ,\quad x_3\ ,\quad x_4 \in \{0,1\} \end{aligned} \tag{6.1}$$

If the problem given in (**??**) on **??** is relaxed:

$$\begin{aligned} x_1\ , x_2\ , x_3\ , x_4 \ge 0 \\ x_1\ , x_2\ , x_3\ , x_4 \le 1 \end{aligned} \tag{6.2}$$

the optimal solution is $\boldsymbol{x} = (1, 2/5, 0, 0)^T$, which achieves the objective function equal to $7 + 8 \times 2/5 = 10.2$. Clearly, this is not an integer solution.

The feasible region of the relaxed problem contains all feasible solutions of the original problem. Hence, the optimum value of the objective function in the relaxed continuous problem is always greater than or equal to the optimum value of the objective function in the original problem. Thus, the optimum value for the relaxed continuous problem is used as an upper bound of the value of objective function in the original problem.

Lower Bound

The lower bound is the best solution yet found. If the optimum solution of the relaxed continuous problem satisfies $x_i \in \{0,1\}$ for all variables, then the solution is also the optimum solution for the original problem. If this solution is greater than the current LB, then the LB may be updated.

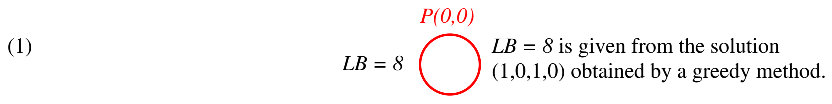
Branch-and-bound for the Knapsack Problem

Now we let J_0 be a set of variables whose values are fixed to 0, and similarly we let J_1 be a set of variables whose values are fixed to 1. Also we let $\mathcal{P}(J_0, J_1)$ be a (combinatorial) optimization problem where some variables are fixed according to J_0 and J_1. Note that $\mathcal{P}(\emptyset, \emptyset)$ represents the original problem (no variable is fixed).

Generation of subproblems: With respect to a problem $\mathcal{P}(J_0, J_1)$, choose one variable x_k from $\boldsymbol{x}\backslash(J_0 \cup J_1)$, and generate two subproblems $\mathcal{P}(J_0 \cup \{x_k\}, J_1)$ and $\mathcal{P}(J_0, J_1 \cup \{x_k\})$.

In the following, the computation process of the branch-and-bound will be briefly demonstrated.

(1) With respect to the original problem $\mathcal{P}(\emptyset, \emptyset)$, compute an initial tentative solution (this example uses the solution $(1, 0, 1, 0)$ computed from a greedy method) and a lower bound $LB = 8$ for the objective value achieved by the optimum solution of $\mathcal{P}(\emptyset, \emptyset)$.


(1)
P(0,0)
LB = 8
LB = 8 is given from the solution
(1,0,1,0) obtained by a greedy method.

(2) Generate a subproblem $\mathcal{P}(\{x_1\}, \emptyset)$ from $\mathcal{P}(\emptyset, \emptyset)$ with fixing $x_1 = 0$.

$$\begin{array}{lrl} \mathcal{P}(\{x_1\}, \emptyset) & \text{Objective:} & 8x_2 + x_3 + 2x_4 \rightarrow \max \\ & \text{Constraints:} & 5x_2 + x_3 + 3x_4 \leq 6 \\ & & x_i = 0, 1 \quad (i = 2, 3, 4) \end{array}$$

In order to find the upper bound on the objective value achieved by the optimum solution for $\mathcal{P}(\{x_1\}, \emptyset)$, we will solve the relaxed continuous version of $\mathcal{P}(\{x_1\}, \emptyset)$.

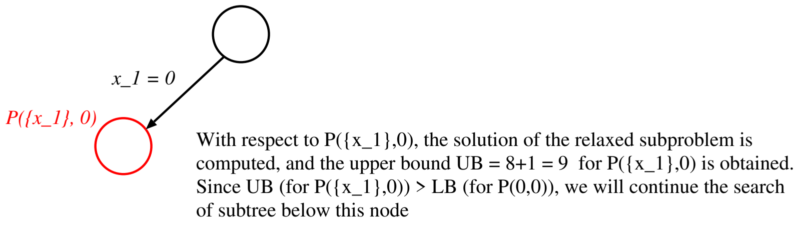
$$\begin{array}{lrl} \mathcal{P}(\{x_1\}, \emptyset)_{relax} & \text{Objective:} & 8x_2 + x_3 + 2x_4 \rightarrow \max \\ & \text{Constraints:} & 5x_2 + x_3 + 3x_4 \leq 6 \\ & & 0 \leq x_i \leq 1 \quad (i = 2, 3, 4) \end{array}$$

The relaxed continuous version can be solved exactly by a greedy method with value per weight as the priority metric, and $(x_2, x_3, x_4)^T = (1, 1, 0)^T$ is obtained.

Comparing the upper bound $UB = 9$ on the objective value for $\mathcal{P}(\{x_1\}, \emptyset)$ with the lower bound $LB = 8$ for $\mathcal{P}(\emptyset, \emptyset)$, $UB > LB$ indicates the possibility of the presence of the optimum solution for $\mathcal{P}(\emptyset, \emptyset)$ in the subproblem $\mathcal{P}(\{x_1\}, \emptyset)$, and the search of the subproblems below this node will follow.

(3) Subproblem $\mathcal{P}(\{x_1, x_2\}, \emptyset)$ is generated from $\mathcal{P}(\{x_1\}, \emptyset)$ with fixing $x_2 = 0$. With respect to this new subproblem, we find the optimum solution $(0, 0, 1, 1)$ for the relaxed continuous problem, which means that the upper

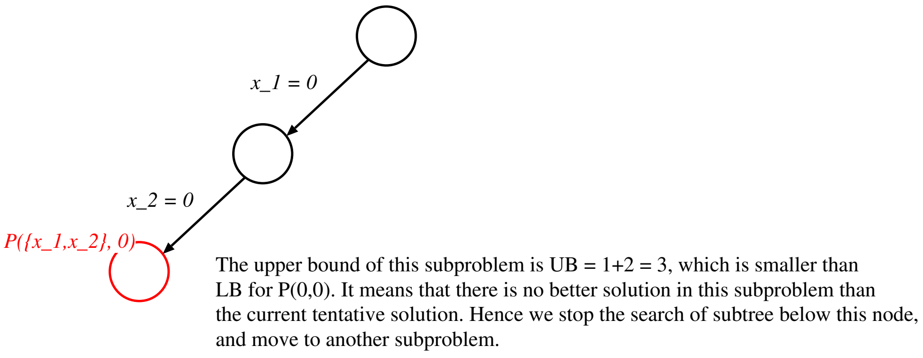
(2)



With respect to P({x_1},0), the solution of the relaxed subproblem is computed, and the upper bound UB = 8+1 = 9 for P({x_1},0) is obtained. Since UB (for P({x_1},0)) > LB (for P(0,0)), we will continue the search of subtree below this node

bound for $\mathcal{P}(\{x_1, x_2\}, \emptyset)$ is $UB = 1 + 2 = 3$. Since $UB = 3 < LB = 8$, we will skip the search of subproblems below this node, and go back to $\mathcal{P}(\{x_1\}, \emptyset)$ (and generate another subproblem).

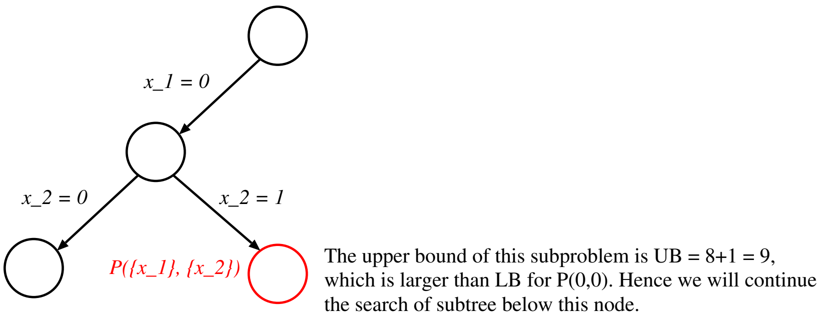
(3)



The upper bound of this subproblem is UB = 1+2 = 3, which is smaller than LB for P(0,0). It means that there is no better solution in this subproblem than the current tentative solution. Hence we stop the search of subtree below this node, and move to another subproblem.

(4) Subproblem $\mathcal{P}(\{x_1\}, \{x_2\})$ is generated from $\mathcal{P}(\{x_1\}, \emptyset)$ with fixing $x_2 = 1$. The upper bound $UB = 9$ on the objective value for this new subproblem is then computed by solving the relaxed continuous problem (the solution is $(0, 1, 1, 0)$). Since $UB = 9 > LB = 8$, the search of subproblems below this node will follow.

(4)



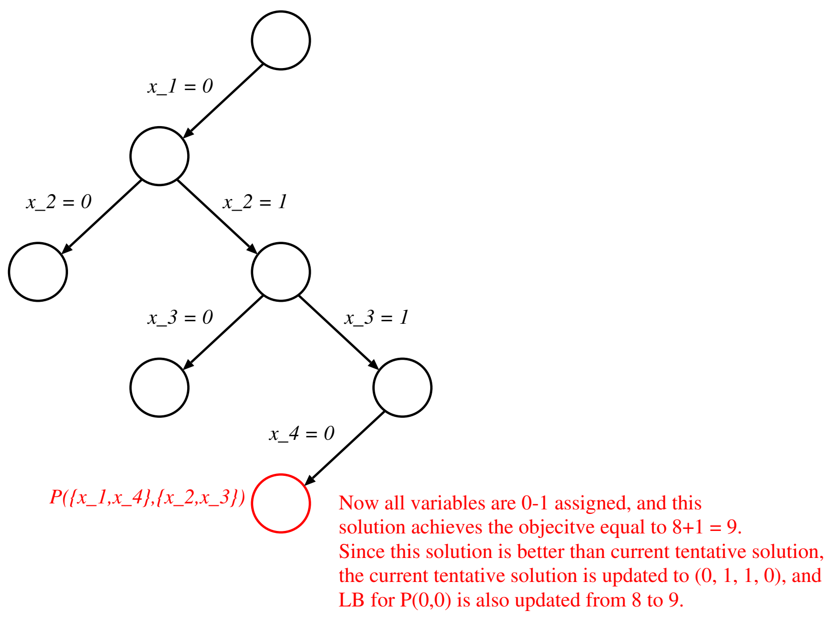
The upper bound of this subproblem is UB = 8+1 = 9, which is larger than LB for P(0,0). Hence we will continue the search of subtree below this node.

(5), (6) are omitted

(7) When we generate a new subproblem from $\mathcal{P}(\{x_1\}, \{x_2, x_3\})$ with fixing $x_4 = 0$, the result $\mathcal{P}(\{x_1, x_4\}, \{x_2, x_3\})$ is a compute 0-1 assignment, i.e., a complete solution, for the original problem $\mathcal{P}(\emptyset, \emptyset)$.

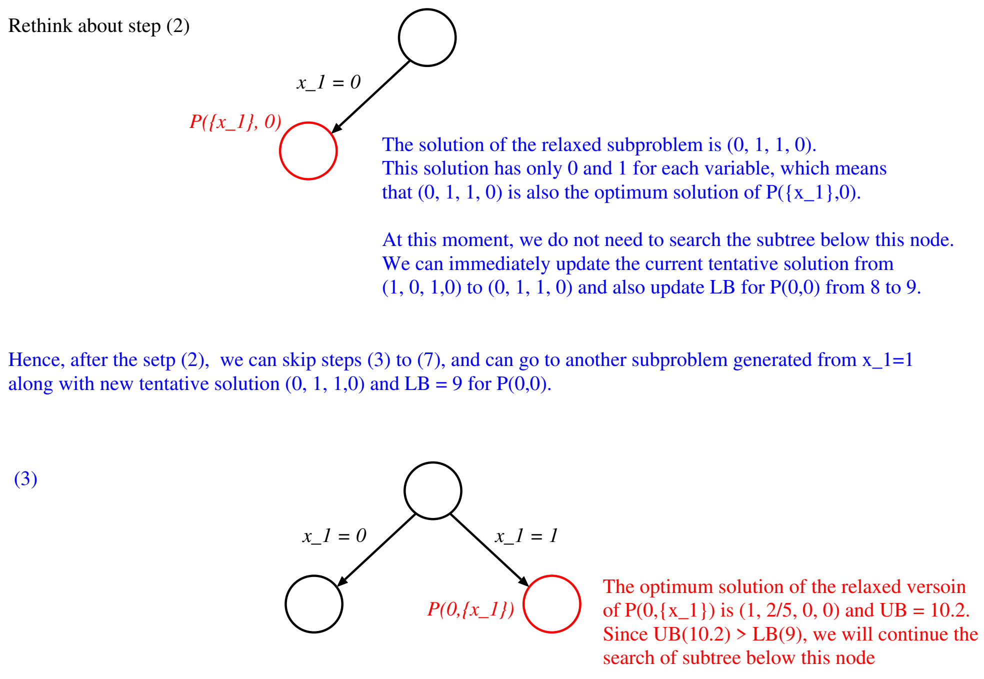
This solution provides the objective value 9, which is larger than the current tentative solution for $\mathcal{P}(\emptyset, \emptyset)$. As a consequence of it, the tentative solution is updated from $(1, 0, 1, 0)$ to $(0, 1, 1, 0)$ and the lower bound is also updated from 8 to 9. In the following steps, $LB = 9$ is used for the decision of skipping/continuing the subproblem search.

(7)


x_1 = 0
x_2 = 0
x_2 = 1
x_3 = 0
x_3 = 1
x_4 = 0
P({x_1,x_4},{x_2,x_3})
Now all variables are 0-1 assigned, and this solution achieves the objecitve equal to 8+1 = 9. Since this solution is better than current tentative solution, the current tentative solution is updated to (0, 1, 1, 0), and LB for P(0,0) is also updated from 8 to 9.

The above is a simple flow (part of the way) of branch-and-bound, but it will be revised a bit. Let's rethink about Step (2).

In Step (2) where the subproblem $\mathcal{P}(\{x_1\}, \emptyset)$ is dealt with, we found $(0, 1, 1, 0)$ as the exact solution of the relaxed continuous version of $\mathcal{P}(\{x_1\}, \emptyset)$. This solution consists of 0 and 1 only, which means that $(0, 1, 1, 0)$ is also the exact solution for the original subproblem $\mathcal{P}(\{x_1\}, \emptyset)$. That is, without searching the subproblems below this node, we find that $(0, 1, 1, 0)$ is the best solution for $\mathcal{P}(\{x_1\}, \emptyset)$, and this solution is better than current tentative solution for $\mathcal{P}(\emptyset, \emptyset)$. So, the update of tentative solution and the update of the lower bound LB, both for $\mathcal{P}(\emptyset, \emptyset)$, can be made immediately, and we can go back $\mathcal{P}(\emptyset, \emptyset)$ for generating another subproblem $\mathcal{P}(\emptyset, \{x_1\})$. Please refer also to Fig.6.4.


Rethink about step (2)

The solution of the relaxed subproblem is (0, 1, 1, 0).
This solution has only 0 and 1 for each variable, which means
that (0, 1, 1, 0) is also the optimum solution of P({x_1},0).

At this moment, we do not need to search the subtree below this node.
We can immediately update the current tentative solution from
(1, 0, 1,0) to (0, 1, 1, 0) and also update LB for P(0,0) from 8 to 9.

Hence, after the setp (2), we can skip steps (3) to (7), and can go to another subproblem generated from x_1=1
along with new tentative solution (0, 1, 1,0) and LB = 9 for P(0,0).

(3)

The optimum solution of the relaxed versoin
of P(0,{x_1}) is (1, 2/5, 0, 0) and UB = 10.2.
Since UB(10.2) > LB(9), we will continue the
search of subtree below this node

Figure 6.4: Rethink of Step (2).

Once the Depth First Search of the tree of subproblems has finished, branch-and-bound algorithm outputs the latest tentative solution as the exact solution of $\mathcal{P}(\emptyset, \emptyset)$, and terminates. Fig.6.5 shows the overall tree of subproblems visited by branch-and-bound algorithm.

6.2.2 Exercise

[1] Assuming that we try to find an optimum solution of a combinatorial “minimization” problem by using “branch-and-bound” method, explain about the mechanism (how to do it, and why it can be done) of “the bounding operation”.

[2] In the branch-and-bound method, it is important to manage the search (branching) so that better solutions can be found as early as possible. Explain why it is.

[3] For an optimization problem, its solution is not always unique. For a problem which may possibly have multiple optimum solutions, we sometimes have to find all optimum solutions, and at other times it is enough to find only one of these solutions. What is the difference between branch-and-bound methods, one for finding all solutions, and the other for finding only one of these solutions.

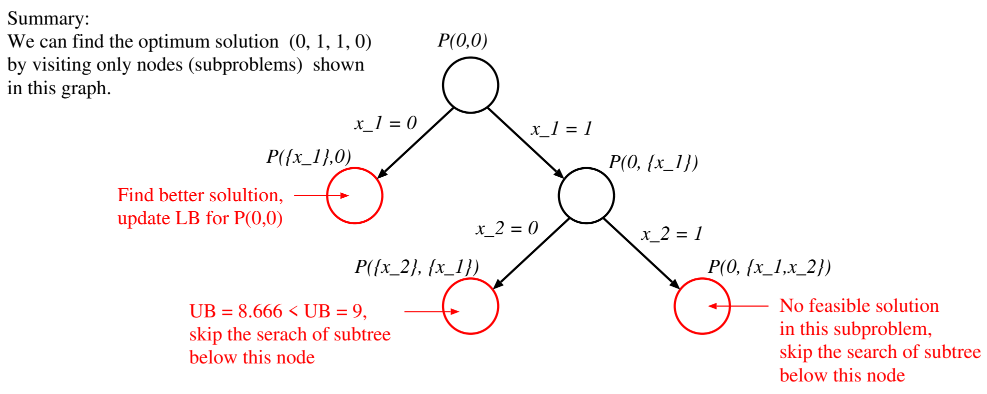

Summary:
We can find the optimum solution (0, 1, 1, 0) by visiting only nodes (subproblems) shown in this graph.
P(0,0)
x_1 = 0
x_1 = 1
P({x_1},0)
P(0, {x_1})
Find better solultion, update LB for P(0,0)
x_2 = 0
x_2 = 1
P({x_2}, {x_1})
P(0, {x_1,x_2})
UB = 8.666 < UB = 9, skip the serach of subtree below this node
No feasible solution in this subproblem, skip the search of subtree below this node

Figure 6.5: Subproblems (nodes in the search tree) visited during the process of branch-and-bound.

6.3 Dynamic Programming

Dynamic programming solves an optimization problem by dividing the main problem into smaller subproblems. If it is possible to find an optimal solution to the subproblems, then this can be used to find the optimal solution of the main problem. In dynamic programming, a problem exhibits optimal substructure if the optimal solution contains optimal solutions to subproblems.

To find a dynamic programming method for solving a specific problem:

1. Characterize the structure of the optimal solution.
2. Recursively define the value of the optimal solution.
3. Compute the optimal value.
4. Construct the optimal solution.

6.3.1 Rod Cutting

The following simple problem[1] illustrates the effectiveness of dynamic programming.

Problem 6.3. The Stone River Supply Company buys long steel rods and cuts them; each cut has no cost. For a rod of length i meters, the income is p_i dollars, given by the following table:

length i	1	2	3	4	5	6	7	8	9	10
income p_i	1	5	8	9	10	17	17	20	24	30

[1]Corman, Leiserson, Rivest, Stein, *Introduction to Algorithms*, MIT Press, 2009, page 360

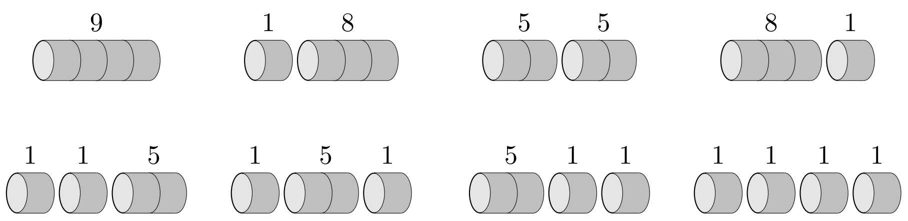


Figure 6.6: All ways to cut a rod of length $n = 4$; the number above the rod gives the income. The most efficient way gives income $p_2 + p_2 = 5 + 5 = 10$.

Given a long rod of length n, the company wants to cut it to maximize the total income r_n.

Rods are only cut to integer lengths i. Consider a long rod of length $n = 4$. The figure shows the various ways to cut the rod, including no cut at all. It can be seen that cutting the length 4 long rod into 2 rod of length 2 each: $4 = 2+2$, gives income of 10, which is the maximum income.

Let r_n be the maximum income from a long rod of length n. By inspection, we can find the following values:

$$\begin{aligned}
r_1 &= 1 && \text{obtained by } 1 = 1 \text{ (no cuts)}\\
r_2 &= 5 && \text{obtained by } 2 = 2 \text{ (no cuts)}\\
r_3 &= 8 && \text{obtained by } 3 = 3 \text{ (no cuts)}\\
r_4 &= 10 && \text{obtained by } 4 = 2 + 2
\end{aligned}$$

For rod cutting, the optimal solution has subproblems which also have optimal solution. For the $n = 4$ example, a cut can be made so that the first piece has length 1, 2, 3 or 4 (4 means no cut). If we know r_1, r_2 and r_3, then we write the problem as:

$$r_4 = \max(p_1 + r_3, p_2 + r_2, p_3 + r_1, p_4),$$

where the last term p_4 corresponds to not cutting the rod. For a general rod of length n, the maximum income is:

$$r_n = \max_{1 \le i \le n} (p_i + r_{n-i}),$$

where $r_0 = 0$. The term $p_i + r_{n-i}$ corresponds to the maximum income for cutting the rod into two pieces of size i and $n-i$. The length of first rod is fixed at i. But the second piece of length $n - i$ may be cut into smaller pieces. The maximum income from this second piece is r_{n-i}. Since we don't know which i is best, we should consider all $i = 1, 2, \ldots, n$ and take the value of i which will maximize income.

For example, we can find r_5 using the previously-found r_1 to r_4 as:

$$\begin{aligned}
r_5 &= \max(1 + 10, 5 + 8, 8 + 5, 9 + 1, 10)\\
&= 13 \quad \text{obtained by } 5 = 2 + 3
\end{aligned}$$

Continuing, more values of the optimal length can be found as:

$$\begin{aligned}
r_6 &= 17 && \text{obtained by } 6 = 6 \text{ (no cuts)}\\
r_7 &= 18 && \text{obtained by } 7 = 1+6 \text{ or } 7 = 2+2+3\\
r_8 &= 22 && \text{obtained by } 8 = 2+6\\
r_9 &= 25 && \text{obtained by } 9 = 3+6\\
r_{10} &= 30 && \text{obtained by } 10 = 10 \text{ (no cuts)}
\end{aligned}$$

6.3.2 Chain of Matrix Multiplications

Multiplication of a $p \times q$ matrix A and $q \times r$ matrix B yields a $p \times r$ matrix. q scalar multiplications are needed to compute one entry of this matrix, and hence pqr multiplications are needed for $A \times B$.

When we compute matrix multiplication of n matrices $A_1 A_2 \cdots A_n$, we repeatedly perform multiplication of two matrices. Now we want to find the order of matrix multiplications (which corresponds to the assignment of parentheses) for minimizing the total number of scalar multiplications. For example, $A_1 A_2 A_3$ could be computed as either:

$$\begin{aligned}
&A_1(A_2A_3) \text{ or}\\
&(A_1A_2)A_3
\end{aligned}$$

Suppose that:

1. A_1 is 10×100
2. A_2 is 100×5
3. A_3 is 5×50

If we compute $(A_1A_2)A_3$, first $10{\cdot}100{\cdot}5 = 5000$ scalar multiplications are needed, followed by $10 \cdot 5 \cdot 50 = 2500$ multiplications, for a total of 7500 multiplications. If we compute $A_1(A_2A_3)$, first $100 \cdot 5 \cdot 50 = 25000$ scalar multiplications are needed, followed by $10 \cdot 100 \cdot 50 = 50000$ multiplications, for a total of 75000 multiplications. Clearly, the first parenthesization is preferred because it uses fewer multiplications.

In the following, we assume that A_i has r_i rows and c_i columns. It is clear that $c_i = r_{i+1}$ for $i = 1, 2, \cdots, n-1$.

Matrix A_i is a $p_{i-1} \times p_i$ matrix.

For four matrices, there are 5 parenthesizations, such as $(A_1A_2)(A_3A_4)$. First, show the complexity of a brute-force approach to demonstrate the advantage of dynamic programming. Let $P(n)$ be the number of different assignments of parentheses for $A_1 A_2 \cdots A_n$.

Parentheses representing the final matrix multiplication have one of the following $n-1$ patterns.

$$\begin{array}{c}(A_1) \times (A_2A_3 \cdots A_{n-1}A_n) \\ (A_1A_2) \times (A_3 \cdots A_{n-1}A_n) \\ \vdots \\ (A_1A_2A_3 \cdots A_{n-1}) \times (A_n)\end{array}$$

Recursively we can consider parentheses representing the final matrix multiplication inside of each pair of parentheses, and we have

$$P(n) = \begin{cases} 1 & \text{for } n = 1 \\ \displaystyle\sum_{k=1}^{n-1} P(k)P(n-k) & \text{for } n \geq 2 \end{cases}$$

The solution of this recursive equation is called Catalan number.

$$P(n) = C(n-1) = \frac{1}{n}\left(\begin{array}{c} 2(n-1) \\ n-1 \end{array}\right)$$

Hence the number of parenthesis patterns grows exponentially with respect to n.

Characterize the structure of the optimal solution

The final matrix multiplication is one of the following.

$$\begin{array}{c}(A_1) \times (A_2A_3 \cdots A_{n-1}A_n) \\ (A_1A_2) \times (A_3 \cdots A_{n-1}A_n) \\ \vdots \\ (A_1A_2A_3 \cdots A_{n-1}) \times (A_n)\end{array}$$

If $(A_1 \cdots A_k) \times (A_{k+1} \cdots A_n)$ is the final matrix multiplication of an optimal assignment of parentheses, the assignment of parentheses to $A_1 \cdots A_k$ is an optimal assignment of parentheses to $A_1 \cdots A_k$, and the assignment of parentheses to $A_{k+1} \cdots A_n$ is also an optimal assignment of parentheses to $A_{k+1} \cdots A_n$.

Recursively define the value of the optimal solution

Let $m[i,j]$ be the minimum number of scalar multiplication for $A_iA_{i+1} \cdots A_j$.

- The optimal assignment of parentheses to A_i is (A_i). No operations are necessary and $m[i,i] = 0$.
- Assume that the last matrix multiplication of $A_iA_{i+1} \cdots A_j$ is given as

  $$(A_i \cdots A_k) \times (A_{k+1} \cdots A_j),$$

  that is, the product is split at k. Now we use the structure of the optimal solution from the previous step by assuming that know $m[i,k]$, the minimum number of scalar multiplications for $A_i \cdots A_k$ as well as $m[k+1, j]$, the minimum number of scalar multiplications for $A_{k+1} \cdots A_j$. Then $(A_i \cdots A_k) \times (A_{k+1} \cdots A_j)$ requires

$$m[i,k] + m[k+1,j] + c_{i-1}c_kc_j$$

scalar multiplications. Thus, the recursive definition of the cost is:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} m[i,k] + m[k+1,j] + c_{i-1}c_kc_j & \text{if } i < j \end{cases}$$

and $m[1, n]$ is the minimum number of scalar multiplications for $A_1 \cdots A_n$.

Compute the optimal costs

Recall $m[i, j]$ is the minimum number of scalar multiplications for $A_i \cdots A_j$. In addition, let $s[i, j]$ be such k that minimizes $\{m[i,k] + m[k+1,j] + c_{i-1}c_kc_j\}$.

Algorithm 6.1 Compute the optimal costs Matrix Multiplication

```
$m[i,i] = 0$, $1 \le i \le n$
**for** $\ell \leftarrow 2$ to $n$ **do**
    **for** $i \leftarrow 1$ to $n - \ell + 1$ **do**
        $j \leftarrow i + \ell - 1$
        $m[i,j] \leftarrow \infty$
        **for** $k \leftarrow i$ to $j - 1$ **do**
            $q \leftarrow m[i,k] + m[k+1,j] + c_{i-1}c_kc_j$
            **if** $q < m[i,j]$ **then**
                $m[i,j] \leftarrow q$
                $s[i,j] \leftarrow k$
            **end if**
        **end for**
    **end for**
**end for**
```

Example 6.1. We consider $A_1A_2\cdots A_6$, where the size of each matrix is given in the table.

matrix	size
A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

As an example, consider the computation of $m[2,5]$, where we assume that all of $m[i,j]$, $j-i \leq 2$, have been computed so far.

$$\begin{aligned}
m[2,5] &= \min\left\{\begin{array}{l} m[2,2]+m[3,5]+c_1c_2c_5 = 0+2500+35\cdot 15\cdot 20 = 13000, \\ m[2,3]+m[4,5]+c_1c_3c_5 = 2625+1000+35\cdot 5\cdot 20 = 7125, \\ m[2,4]+m[5,5]+c_1c_4c_5 = 4375+0+35\cdot 10\cdot 20 = 11375 \end{array}\right\} \\
&= 7125 \\
s[2,5] &= 3
\end{aligned}$$

(See $i=2$, $j=5$ entry of $m[i,j]$ table.)

Construct an optimal solution

Although we have found the minimum number of operations, it is needed to construct the optimal solution.

Example 6.2. From $m[i,j]$ table, we have an optimal assignment of parentheses

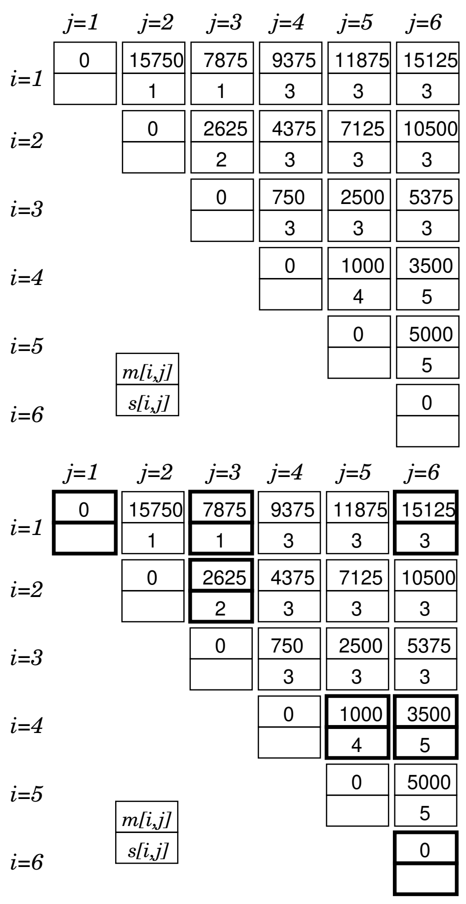
$$(A_1(A_2A_3))((A_4A_5)A_6).$$

6.3.3 Knapsack Problem

As another example of designing a dynamic programming algorithm, consider Knapsack problem, restated for convenience:

Problem 6.4. Given a set of n items, each with given weight a_i and utility $c_i \geq 0$, select the items to maximize the total utility while having total weight not greater than W.

In this case, there is only one of each item, that is $x_i \in \{0,1\}$, where x_i is the number of item i in the knapsack. Let the items be denoted $1,2,\ldots,n$, and c_i and a_i be the value and the weight of an item i. W is the maximum total weight of items to be packed into the knapsack.



		j=1	j=2	j=3	j=4	j=5	j=6
i=1	m[i,j]	0	15750	7875	9375	11875	15125
	s[i,j]		1	1	3	3	3
i=2	m[i,j]		0	2625	4375	7125	10500
	s[i,j]			2	3	3	3
i=3	m[i,j]			0	750	2500	5375
	s[i,j]				3	3	3
i=4	m[i,j]				0	1000	3500
	s[i,j]					4	5
i=5	m[i,j]					0	5000
	s[i,j]						5
i=6	m[i,j]						0
	s[i,j]						

		j=1	j=2	j=3	j=4	j=5	j=6
i=1	m[i,j]	**0**	15750	**7875**	9375	11875	**15125**
	s[i,j]		1	**1**	3	3	**3**
i=2	m[i,j]		0	**2625**	4375	7125	10500
	s[i,j]			**2**	3	3	3
i=3	m[i,j]			0	750	2500	5375
	s[i,j]				3	3	3
i=4	m[i,j]				0	**1000**	**3500**
	s[i,j]					**4**	**5**
i=5	m[i,j]					0	5000
	s[i,j]						5
i=6	m[i,j]						**0**
	s[i,j]						

Figure 6.7: Table of $m[i,j]$, $s[i,j]$.

Characterize the structure of the optimal solution

To apply dynamic programming, find the structure of the subproblem. The subproblem is finding $x_1, \ldots, x_{n-1}$ when the weights are $a_1, \ldots, a_{n-1}$, the values are $c_1, \ldots, c_{n-1}$ and the capacity is some $w \le W$. Suppose that we have an optimal solution $x_1^*, x_2^*, \ldots, x_n^*$. For the original problem, consider the two cases, $x_n^* = 0$ and $x_n^* = 1$. If $x_n^* = 0$, then it must be that $x_1^*, \ldots, x_{n-1}^*$ is an optimal solution to the knapsack problem with $n-1$ items $1, 2, \ldots, n-1$ and capacity W. (otherwise, any better solution would also improve the n-item problem). If $x_n^* = 1$, then it must be that $x_1^*, \ldots, x_{n-1}^*$ is an optimal solution to the knapsack problem with $n-1$ and capacity $W - a_n$.

In this way, we can see the structure of the subproblem: to solve the knapsack problem with n variables, the optimal solution to another knapsack problem with $n-1$ variables is needed. However, we need an solution for a **group** of subproblems — even when the number of variables is fixed at $n-1$, we need solution for arbitrary weights $w \le W$.

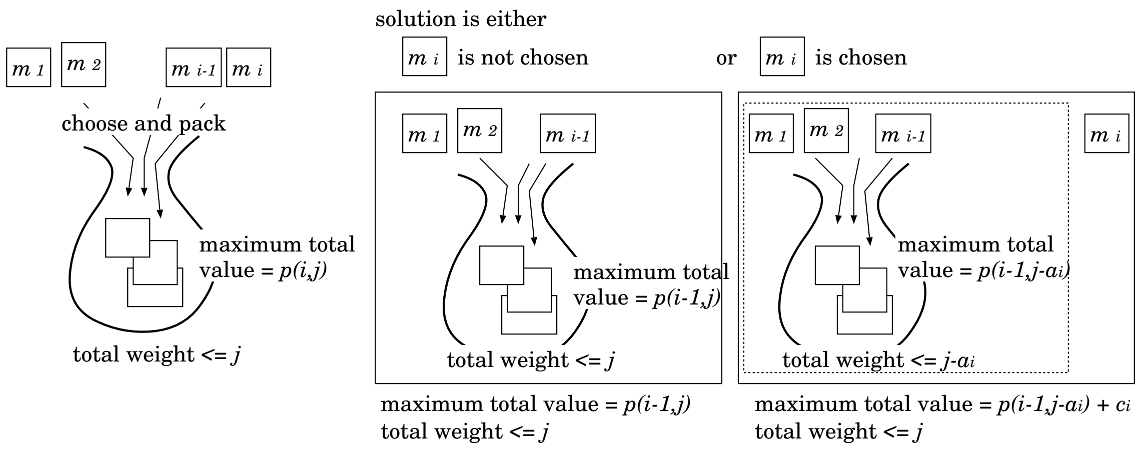


Figure 6.8: Principle of Optimality in Knapsack problem.

Recursively define the value of the optimal solution

Following the structure in the first step, define an objective function. Let $p(i, w)$ be the maximum total value achieved in the problem "choose items among $1, 2, \ldots, i$ for a knapsack with capacity w". (Note that $p(n, W)$ is the maximum total value achieved for the original problem instance.) The essential part is to consider two cases, either $x_i = 0$, in which case we need the optimal value is $p(i-1, w)$, or $x_1 = 1$, in which case we add c_i to the value, but need the optimal value $p(i-1, w-a_i)$ of the problem with capacity $w - a_i$. The optimal value is the greater of the two:

$$\max\big(p(i-1, w), p(i-1, w-a_i) + c_i\big)$$

More precisely, we handle the edge cases and $p(i, w)$ is:

$$p(i, w) = \begin{cases} p(i-1, w) & \text{if } a_i > w \\ \max\big(p(i-1, w), p(i-1, w-a_i) + c_i\big) & \text{if } a_i \le w \\ 0 & \text{if } i = 0 \end{cases}$$

Compute the optimal costs

Using the above recursive definition, compute $p(i, w)$. First fix $i = 0$ as the smallest value and compute all $w = 1, 2, 3, \ldots, W$. Then $i = 1$, and again compute all w. This continues until all values $p(i, w)$ are found. See the example below.

Construct an optimal solution

Finding the $p(i,w)$ only provides the optimal value. Now define $x(i,w)$ for recording partial solution corresponding to $p(i,w)$. That is,

$$x(i,w) = \begin{cases} 0 & \text{if } p(i,w) \text{ is achieved by discarding } i \\ 1 & \text{if } p(i,w) \text{ is achieved by accepting } i \end{cases}$$

In particular, the value of $x(i,w)$ depends on how we obtained the maximum:

$$\max\big(\underbrace{p(i-1,w)}_{x(i,w)=0}, \underbrace{p(i-1,w-a_i)+c_i}_{x(i,w)=1}\big)$$

The values of $x(i,w)$ are computed recursively along with $p(i,w)$.

Example 6.3. Use dynamic programming to solve the $n=4$ knapsack problem with

$$(c_1,c_2,c_3,c_4) = (7,8,1,2)$$
$$(a_1,a_2,a_3,a_4) = (4,5,1,3)$$

and $W=6$.

We represent the values $p(i,w)$ and $x(i,w)$ in a table. The first row (the row with $i=0$) is trivial since $i=0$. The computation proceeds first in columns, then in rows. Set $i=1$ and compute all w from left to right. Then proceed to row $i=2$.

$p(i,w)$ / $x(i,w)$	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$
$i=0$	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
$i=1$	0 / 0	0 / 0	0 / 0	0 / 0	7 / 1	7 / 1	7 / 1
$i=2$	0 / 0	0 / 0	0 / 0	0 / 0	7 / 0	8 / 1	8 / 1
$i=3$	0 / 0	1 / 1	1 / 1	1 / 1	7 / 0	8 / 0	9 / 1
$i=4$	0 / 0	1 / 0	1 / 0	2 / 1	7 / 0	8 / 0	9 / 0

For example, entries in row $i=2$ are computed as:

$$\begin{aligned}
p(2,1) &= p(1,1)=0, \quad x(2,1)=0 \quad (\text{since } a_2=5>1)\\
p(2,2) &= p(1,2)=0, \quad x(2,2)=0 \quad (\text{since } a_2=5>2)\\
p(2,3) &= p(1,3)=0, \quad x(2,3)=0 \quad (\text{since } a_2=5>3)\\
p(2,4) &= p(1,4)=7, \quad x(2,4)=0 \quad (\text{since } a_2=5>4)\\
p(2,5) &= max\{p(1,5), p(1,5-a_2)+c_2\} = max\{7,8\}=8, \quad x(2,5)=1\\
p(2,6) &= max\{p(1,6), p(1,6-a_2)+c_2\} = max\{7,8\}=8, \quad x(2,6)=1
\end{aligned}$$

Entries in row $i = 3$ are:

$$\begin{aligned}
p(3,1) &= max\{p(2,1), p(2,1-a_3)+c_3\} = max\{0,1\} = 1, \quad x(3,1) = 1\\
p(3,2) &= max\{p(2,2), p(2,2-a_3)+c_3\} = max\{0,1\} = 1, \quad x(3,2) = 1\\
p(3,3) &= max\{p(2,3), p(2,3-a_3)+c_3\} = max\{0,1\} = 1, \quad x(3,3) = 1\\
p(3,4) &= max\{p(2,4), p(2,4-a_3)+c_3\} = max\{7,1\} = 7, \quad x(3,4) = 0\\
p(3,5) &= max\{p(2,5), p(2,5-a_3)+c_3\} = max\{8,8\} = 8, \quad x(3,5) = 0\\
p(3,6) &= max\{p(2,6), p(2,6-a_3)+c_3\} = max\{8,9\} = 9, \quad x(3,6) = 1
\end{aligned}$$

Once the table is completed, and $p(4,6) = 9$ is obtained, we find out that the maximum total value achieved by an optimum solution is 9.

Next, we will construct an optimum solution in top-down fashion from the entry $(i = 4, w = 6)$ to entries with smaller i and/or smaller w. In the above case, a solution is constructed as follows.

1. Since $x(4,6) = 0$, $x_4 = 0$ (item 4 is discarded).
   Next entry we need to check is $(3,6)$, since $p(4,6) = p(3,6)$.
2. Since $x(3,6) = 1$, $x_3 = 1$ (item 3 is accepted).
   Next entry we move to is $(2, 6-a_3) = (2,5)$, since $p(3,6) = p(2,6-a_3)+c_3$.
3. Since $x(2,5) = 1$, $x_2 = 1$ (item 2 is accepted).
   Next entry is $(1, 5-a_2) = (1,0)$, since $p(2,5) = p(1,5-a_2)+c_2$.
4. Since $x(1,0) = 0$, $x_1 = 0$ (item 1 is discarded).

So, we have an optimum solution $(x_1, x_2, x_3, x_4) = (0,1,1,0)$.

$p(i,w)$ / $x(i,w)$	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$
$i=0$	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
$i=1$	**0 / 0**	0 / 0	0 / 0	0 / 0	7 / 1	7 / 1	7 / 1
$i=2$	0 / 0	0 / 0	0 / 0	0 / 0	7 / 0	**8 / 1**	8 / 1
$i=3$	0 / 0	1 / 1	1 / 1	1 / 1	7 / 0	8 / 0	**9 / 1**
$i=4$	0 / 0	1 / 0	1 / 0	2 / 1	7 / 0	8 / 0	**9 / 0**

6.3.4 Exercise

[1] Consider the following problem instance of matrix multiplications, and we are going to solve it with the dynamic programming.

$A_1A_2A_3A_4$ where

matrix		row$\times$ column
A_1	:	8×10
A_2	:	10×3
A_3	:	3×9
A_4	:	9×7

(1) Construct the table of $m[i,j]$ (and $s[i,j]$).

(2) Show the optimum assignment of parentheses.

[2] Using the dynamic programming method, find an optimum solution of the following instance of Knapsack problem.

Items

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
Value	25	19	27	34	7	21	12
Weight	5	4	6	8	2	6	4

Weight limitation of the knapsack $W = 12$

[3] Consider the following problem.

Given n weights $m_1, m_2, \cdots, m_n$, where the weight of m_i is w_i (an integer), and an integer W is also given, decide whether the weight W is realized using only these weights or not.

We will use A to represent a subset of weights $\{m_1, m_2, \cdots, m_n\}$. The most light weight which can be realized by those weights is 0 ($A = \emptyset$), and the most heavy weight is $\sum_{i=1}^{n} w_i$ ($A = \{m_1, m_2, \cdots, m_n\}$).

For example, given four weights, where $w_1 = 2, w_2 = 3, w_3 = 2, and w_4 = 4$, and given also $W = 7$, the answer of the problem is "yes", and $A = \{m_1, m_2, m_3\}$ and $A = \{m_2, m_4\}$ are realizations of $W = 7$. On the other hand, $W = 10$ is given, the answer is "no", that is, $W = 10$ can not be realized by these four weights.

The most primitive way to solve this problem is to enumerate combinations of weights and check one by one whether W is realized or not. Clearly, it needs $\mathcal{O}(2^n)$ computation time.

The following shows another computation algorithm based on Dynamic programming, where the function $t : \{1, 2, \cdots, n\} \times \mathbb{Z} \to \{T(rue), F(alse)\}$ plays an important role to represents the recursive structure of optimum solutions.

$$t(i,j) = \begin{cases} T & : \text{ Using weights } \{m_1, m_2, \cdots, m_i\}, \text{ weight } j \text{ is realized.} \\ F & : \text{ Using weights } \{m_1, m_2, \cdots, m_i\}, \text{ weight } j \text{ can not be realized.} \end{cases}$$

It is clear that $t(i, 0) = T$ (total weight of the subset $\emptyset$ of $\{m_1, m_2, \cdots, m_i\}$ is 0). On the other hand, $t(n, W)$ is equivalent to the original problem.

(1) For the problem instance $n = 4$, $w_1 = 2$, $w_2 = 3$, $w_3 = 2$, and $w_4 = 4$, compute $t(1, 1)$, $t(1, 2)$, $t(1, 3)$, $t(1, 4)$, $t(1, 5)$, and $t(1, 6)$.
Show a general form of $t(1, j)$ without depending to some specific problem instance.

(2) Find out the recursive from of the function $t(i, j)$. (That is, $t(i, j)$ is represented using $t(i', j)$, $t(i, j')$, or $t(i', j')$ with $i' < i$, $j' < j$.)

(3) Consider the problem instance; $n = 4$, $w_1 = 2, w_2 = 3, w_3 = 2, w_4 = 4$, $W = 6$. Complete the table of $t(i, j), 1 \leq i \leq 4, 0 \leq j \leq 6$, and find the solution.

$t(i, j)$

$i \backslash j$	0	1	2	3	4	5	6
1	T						
2	T						
3	T						
4	T						

[4] Knapsack problem is the problem to find articles, from a given set of articles, so that total value is maximized under the limitation of the total weight. Now we will consider the following "Another version of Knapsack problem".

[Another Version of Knapsack problem (AVKS)]

Given n articles $M = \{m_1, \cdots, m_n\}$, each of which has integer value and integer weight, and also given an integer C, choose articles from M so that total weight is minimized under the constraint that the total value is no smaller than C.

(1) Develop (your own) greedy method for this AVKS. After that, apply it to the following problem instance of AVKS, and find a solution.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
Value	2	3	3	1	2	4	1
Weight	4	10	11	4	9	19	5

Required minimum total value $C = 7$

(2) We want to develop a dynamic programming method for this AVKS. Find out a recursive structure of the optimum solution, which can be used to develop a dynamic programming method.

(3) With respect to the problem instance shown in [3]-(1), find an optimum solution using the dynamic programming method.

[5] Let $\mathcal{S}$ be a finite set of alphabets, and L_1, L_2 be two strings of these alphabets. We can find a sequence of alphabets which are commonly appear in both L_1 and L_2, and we call it a "common subexpression" of L_1 and L_2.

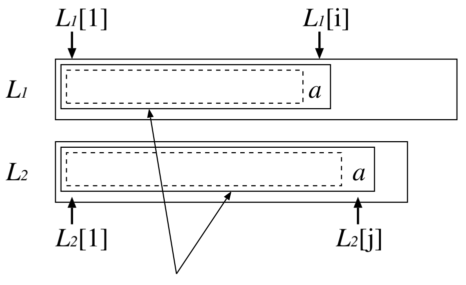
For example, for $\mathcal{S} = \{a,\ b,\ c\}$, and given two strings $L_1 = a\ b\ b\ a\ c\ c\ b\ a\ c$ and $L_2 = b\ a\ c\ b\ a\ a\ b\ c\ a$, we can find "$a\ b\ a\ b\ c$" as one of common subexpressions for L_1 and L_2.

"$a\ b\ a\ b\ c$" is one of	$L_1 = \underline{a}\ \underline{b}\ b\ \underline{a}\ c\ c\ \underline{b}\ a\ \underline{c}$
common subexpressions:	$L_2 = b\ \underline{a}\ c\ \underline{b}\ \underline{a}\ a\ \underline{b}\ \underline{c}\ a$

For simplicity, $L_1[i]$ ($L_2[i]$) denotes ith alphabet in L_1 (L_2). For example, for the previous example, $L_1[1] = a$, $L_2[3] = c$, etc.

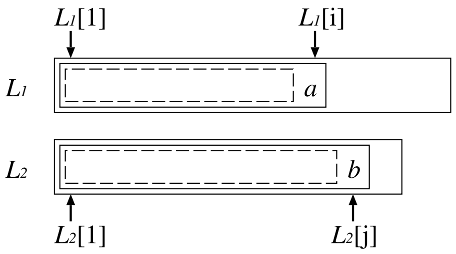
Our problem considered here is to find a longest common subexpression for given two strings. (For the above example, $b\ a\ c\ b\ a\ c$ is a longer (maybe longest) common subexpression.) We will approach this **longest common subexpression problem** with a **dynamic programming method**.

(1) Let $p(i, j)$ be the length of the longest common subexpression of two strings $L_1[1]\ L_1[2] \cdots L_1[i]$ and $L_2[1]\ L_2[2] \cdots L_2[j]$.
If $L_1[i] = L_2[j]$, how can $p(i, j)$ be expressed using the length of the longest common subexpression of shorter sub-strings?



$p(i,j)$; the length of the longest common subexpression for these two sequences

(2) If $L_1[i] \neq L_2[j]$, how can $p(i, j)$ be expressed using the length of the longest common subexpression of shorter sub-strings?



(3) It is trivial that $p(0, j) = 0$ and $p(i, 0) = 0$. Using this fact and the results of (1) and (2), find a dynamic programming solution method for the longest common subexpression problem.

(4) With respect to the following problem instance, apply the dynamic programming method (obtained in (3)) to find the longest common subexpression.

$$L_1 = a\ b\ c\ b\ a\ c$$
$$L_2 = b\ c\ a\ b\ c\ a$$

[6] "Knapsack Problem" is a problem to choose items which will be packed into a knapsack. Here we think about the problem to choose items for two knapsacks, named "2-Knapsack Problem".

2-Knapsack Problem:

A set of items $\{m_1, m_2, \cdots, m_n\}$ is given, and each item has its weight $a_i \in \mathbb{Z}_+$ and its value $c_i \in \mathbb{Z}_+$. We now have two knapsacks B_1 and B_2

which have weight limit W_1 and W_2, respectively. Choose items for the knapsack B_1 and those for the knapsack B_2, so that the total value is maximized.

(Remark) If we prepare two binary variables $x_i \in \{0, 1\}$ and $y_i \in \{0, 1\}$ for each item m_i, and we assume that $x_i = 1$ means that m_i is packed into B_1, and $y_i = 1$ means that m_i is packed into B_2, "2-Knapsack Problem" can be formulated as follows.

$$\begin{array}{lll}
\text{Objective:} & \displaystyle\sum_{i=1}^{n} c_i\,(x_i + y_i) \ \rightarrow \max & \\
\text{Constraints:} & x_i \in \{0,1\} & \text{for } i = 1, 2, \cdots, n \\
 & y_i \in \{0,1\} & \text{for } i = 1, 2, \cdots, n \\
 & x_i + y_i \le 1 & \text{for } i = 1, 2, \cdots, n \\
 & & \text{(Each } m_i \text{ is never packed into } B_1 \text{ and } B_2 \text{ simultaneously.)} \\
 & \displaystyle\sum_{i=1}^{n} a_i \cdot x_i \le W_1 & \text{(Weight limitation for } B_1) \\
 & \displaystyle\sum_{i=1}^{n} a_i \cdot y_i \le W_2 & \text{(Weight limitation for } B_2)
\end{array}$$

(1) Consider the following instance of 2-Knapsack Problem.

	m_1	m_2	m_3	m_4	m_5
c_i	3	3	2	2	1
a_i	1	1	2	2	3

$W_1 = 3$
$W_2 = 3$

Find (by some "ad hoc" manner) the optimum solution as the 2-Knapsack Problem.

(2) From now, we are going to construct a procedure to compute a solution of 2-Knapsack Problem. First of all, we will try to use the solution of conventional Knapsack (one knapsack) Problem to compute a solution of 2-Knapsack Problem.

Approach 1:

Step 1: Consider only B_1, and solve the conventional Knapsack Problem for B_1. (That is, choose items which can be packed into B_1 so that the total value in B_1 is maximized.)

Step 2: With respect to the remained items and the second knapsack B_2, solve the conventional Knapsack Problem for B_2. (That is, choose items from the remained items so that the total value in B_2 is maximized.)

Step 3: Combine the solution in Step 1 and the one in Step 2 to form a solution for 2-Knapsack Problem.

According to this Approach 1, find a solution of 2-Knapsack Problem for the problem instance given in (1).

(3) As the **second approach**, develop a dynamic-programming-based algorithm which can compute an exact solution for 2-Knapsack Problem.

The following issues should be included in the explanation.

· The principle of optimality for 2-Knapsack Problem: an optimum solution of a problem instance includes an optimum solution of a smaller problem instance
· How to construct a table used in the dynamic programming
· The other necessary issues (if any)

Hint 1: In a dynamic programming approach to the conventional Knapsack Problem, $p(i, j)$ is used for representing an optimum solution (optimum total value) of a problem instance which consists of items from m_1 through m_i and weight limitation j. In the 2-Knapsack Problem, we need to treat B_1 and B_2 separately, and to do it, $p(i, j, k)$ is used for representing an optimum solution (optimum total value) of a problem instance which consists of items from m_1 through m_i, weight limitation j for B_1, and weight limitation k for B_2. Then, the problem is to find the way how to compute $p(i, j, k)$ by using those values of smaller instances.

Hint 2: With respect to each item m_i, an available choice is (1) m_i is put into B_1, (2) m_i is put into B_2, or (3) m_i is not put into B_1 nor B_2.

(4) Using the dynamic programming proposed in (3), compute the solution of the problem instance given in (1). (Note that not only the final solution, but also the table used in the dynamic programming and the process to construct the solution using this table should be presented.)

[7] We consider a modified version of Knapsack Problem. In this modified problem, m_i and m_{i+1}, $i \in \{1, 3, 5, \cdots, n-1\}$, have similar functions, and hence we do not need to put both of them into the knapsack. That is, for each $i \in \{1, 3, 5, \cdots, n-1\}$, our choice is (a) m_i is put into the knapsack, but m_{i+1} is not, (b) m_{i+1} is put into the knapsack, but m_i is not, or (c) neither m_i nor m_{i+1} is put into the knapsack.

Find a way to compute the exact optimum solution of this modified Knapsack Problem based on Dynamic Programming, and explain it in detail.

6.4 Approximation Algorithm

Formally, a combinatorial optimization problem Π is either a minimization problem or a maximization problem and consists of the following three parts:

(1) a set D_Π of instances;

(2) for each instance $I \in D_\Pi$, a finite set $S_\Pi(I)$ of candidate solutions for I; and

(3) a function (objective function) m_Π that assigns to each instance $I \in D_\Pi$ and each candidate solution $\sigma \in S_\Pi(I)$ a positive rational number $m_\Pi(I, \sigma)$, called the solution value for σ.

If Π is a minimization (maximization) problem, then an optimal solution for an instance $I \in D_\Pi$ is a candidate solution $\sigma^* \in S_\Pi(I)$ such that, for all $\sigma \in S_\Pi(I)$, $m_\Pi(I, \sigma^*) \leq m_\Pi(I, \sigma)$ $(m_\Pi(I, \sigma^*) \geq m_\Pi(I, \sigma))$.

An algorithm A is an approximation algorithm for Π if, given any instance $I \in D_\Pi$, it finds a candidate solution $\sigma \in S_\Pi(I)$.

The value $m_\Pi(I, \sigma)$ of the candidate solution σ found by A when applied to I will be denoted by $A(I)$, while the value $m_\Pi(I, \sigma^*)$ of an optimal solution σ^* for I will be denoted by $OPT(I)$.

If $A(I) = OPT(I)$ for all $I \in D_\Pi$, then A is called an optimization algorithm for Π.

If the optimization problem is NP–hard, then we know that a polynomial time optimization algorithm cannot be found unless $P = NP$. A more reasonable goal is that of finding an approximation algorithm A that runs in low–order polynomial time and that has the property that, for all instances I, $A(I)$ is "close" to $OPT(I)$.

If Π is a minimization (maximization) problem, and I is any instance in D_Π, we define the ratio $R_A(I)$ by

$$R_A(I) = \frac{A(I)}{OPT(I)} \qquad \left(R_A(I) = \frac{OPT(I)}{A(I)}\right)$$

<u>Absolute performance ratio R_A</u> for an approximation algorithm A for Π;

$$R_A = \inf\{r \geq 1 \,|\, R_A(I) \leq r \text{ for all instance } I \in D_\Pi\}$$

<u>Asymptotic performance ratio R_A^∞</u> for$A$;

$$R_A^\infty = \inf\left\{ r \geq 1 \;\middle|\; \begin{array}{l} \text{for some } N \in \mathbb{Z}^+,\ R_A(I) \leq r \text{ for all} \\ I \in D_\Pi \text{ satisfying } OPT(I) \geq N \end{array} \right\}$$

It is clear that a ratio that is closer to 1 indicates better performance.

(For some (many) approximation algorithms, it may be very difficult to evaluate R_A or R_A^∞ exactly. In such cases, its upper bound may be still informative.)

6.4.1 Knapsack Problem

We consider the following approximation algorithm (greedy algorithm) for Knapsack Problem.

Algorithm A**:** (S1) Sort elements in $\{m_1, m_2, \cdots, m_n\}$ in decreasing order of value/weight ratio. Starting with U' empty, proceed sequentially through this list, each time adding m_i to U' whenever the sum of weights of the items already in U' does not exceed $W - a_i$. (S2) Find another solution consisting solely of the maximum value item. (S3) Compare total values for the solutions found in S1 and S2, and output the better of the two.

Theorem 6.3. Let $\boldsymbol{x}_{opt}$ be an optimum solution of an instance of the Knapsack problem, and let $\boldsymbol{x}_A$ be a solution for the same instance obtained by the algorithm A. Then the following holds for all instances of Knapsack problem.

$$\frac{C(\boldsymbol{x}_{opt})}{C(\boldsymbol{x}_A)} < 2$$

where $C(\boldsymbol{x})$ is a solution value for a feasible solution $\boldsymbol{x}$.

Proof

Without loss of generality, we assume that $c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n$. Let $\boldsymbol{x}_A$ be the solution obtained by Algorithm A, and let m_k be the first item which is not chosen. That is,

$$\boldsymbol{x}_A = \begin{array}{ccccccccc} & x_1, & \cdots, & x_{k-1}, & x_k, & x_{k+1}, & \cdots, & x_n & \\ (& 1, & \cdots, & 1, & 0, & -, & \cdots, & - &) \end{array}$$

- With respect to the solution value obtained by the Algorithm A, we have,

$$C(\boldsymbol{x}_A) \geq \sum_{i=1}^{k-1} c_i$$

  and

$$C(\boldsymbol{x}_A) \geq c_j, \quad \text{for } 1 \leq j \leq n$$

- From the procedure of the Algorithm A,

$$a_k > W - \sum_{i=1}^{k-1} a_i.$$

- Relaxed continuous version can be solved by the greedy method based on value/weight ratio, and let $\boldsymbol{x}^*$ be the solution of this relaxed version. Then $\boldsymbol{x}^*$ has the form as,

$$\boldsymbol{x}^* = \begin{array}{ccccccccc} & x_1, & \cdots, & x_{k-1}, & x_k, & x_{k+1}, & \cdots, & x_n & \\ (& 1, & \cdots, & 1, & \frac{W-\sum_{i=1}^{k-1} a_i}{a_k}, & 0, & \cdots, & 0 &) \end{array}$$

  and the solution value $C(\boldsymbol{x}^*)$ is given as,

$$C(\boldsymbol{x}^*) = \sum_{i=1}^{k-1} c_i + c_k \cdot \frac{W - \sum_{i=1}^{k-1} a_i}{a_k} < \sum_{i=1}^{k-1} c_i + c_k \cdot \frac{a_k}{a_k} = \sum_{i=1}^{k-1} c_i + c_k$$

- Since the original problem is considered as a constrained version of the relaxed continuous version, we have,

$$C(\boldsymbol{x}_{opt}) \leq C(\boldsymbol{x}^*)$$

As a result, we have,

$$C(\boldsymbol{x}_{opt}) \leq C(\boldsymbol{x}^*) < \sum_{i=1}^{k-1} c_i + c_k \leq C(\boldsymbol{x}_A) + C(\boldsymbol{x}_A) = 2C(\boldsymbol{x}_A)$$

□

Now we consider the following problem instance.

$$\begin{aligned}
(c_1, c_2, c_3) &= \left(p, p - \frac{4p\varepsilon}{W/2 + 2\varepsilon}, p - \frac{5p\varepsilon}{W/2 + 2\varepsilon}\right) \\
(a_1, a_2, a_3) &= (W/2 + 2\varepsilon, W/2 - \varepsilon, W/2 - \varepsilon) \\
W &: \text{Knapsack capacity}
\end{aligned}$$

For this problem instance, the Algorithm A chooses m_1 only, and $C(x_A) = p$. On the other hand, the optimum solution contains m_2 and m_3, and $C(x_{opt}) = 2p - \dfrac{9p\varepsilon}{W/2 + 2\varepsilon}$ Hence, we have

$$\frac{C(x_{opt})}{C(x_A)} = 2 - \frac{9\varepsilon}{W/2 + 2\varepsilon}$$

Decreasing ε, $\frac{C(x_{opt})}{C(x_A)}$ approaches 2. This shows that the upper bound 2 of $\frac{C(x_{opt})}{C(x_A)}$, shown in Theorem 6.3, is not improved.

6.4.2 Minimum Steiner Tree Problem

Given a graph $G = (V, E)$ and a subset of V $R \subseteq V$, a subgraph of $G = (V, E)$, where it is a tree (a node having degree 1 is always a vertex in R) and connects all vertices in R, are called Steiner tree. A vertex in a Steiner tree, whose degree is greater than 2 and is not a vertex included in R, is called a "Steiner vertex".

Let $\lambda : E \to \boldsymbol{R}_+$ be an edge weighting function, and consider the following minimum Steiner Tree Problem.

Minimum Steiner Tree Problem

Instance :	$G = (V, E)$: connected graph
	$\lambda : E \to \boldsymbol{R}_+$
	$R \subseteq V$
Feasible solution:	Steiner tree $T(R_T, E_T)$ of G and a set of required vertices R
Objective function:	$C(T) = \sum_{e \in E_T} \lambda(e) \quad \to \min$

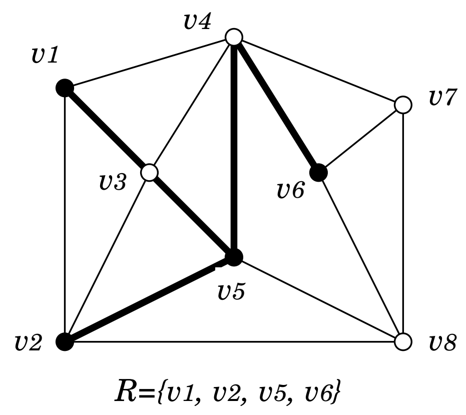


Figure 6.9: An example of Steiner Tree.

Minimum Steiner Tree Problem with $|R| = 2$ is reduced to the Shortest Path Problem, and the one with $R = V$ is reduced to the Minimum Spanning Tree Problem, and each of them has polynomial time algorithm to solve it. However, the Minimum Steiner Tree Problem in general is NP–hard.

Now we show an approximation algorithm for the problem.

Approximation Algorithm

Step 1: Construct a complete graph with its vertex set R, and let it be $G_1(R, E_1)$. We also consider the edge–weighting function, where the weight of $(v, w) \in E_1$ corresponds to the shortest path length between v and w on G.

Step 2: Compute a minimum spanning tree $G_2(R, E_2)$ of $G_1(R, E_1)$.

Step 3: Corresponding to each edge $(v, w) \in E_2$ in $G_2(R, E_2)$, we restore a shortest path $p(v, w)$ between v and w on G, and construct $G_3(V_3, E_3)$ as follows.

$$G_3(V_3, E_3) = \bigcup_{(v,w)\in E_2} p(v, w)$$

Step 4: Compute a minimum spanning tree $G_4(V_3, E_4)$ of $G_3(V_3.E_3)$.

Step 5: On G_4, find a vertex whose degree is 1 and is not included in R, remove the vertex and the associated edge. Let the resultant tree be $G_5(V_5, E_5)$, and output $G_5(V_5, E_5)$ as an approximated solution (Approximated Minimum Steiner Tree) of the Minimum Steiner Tree Problem.

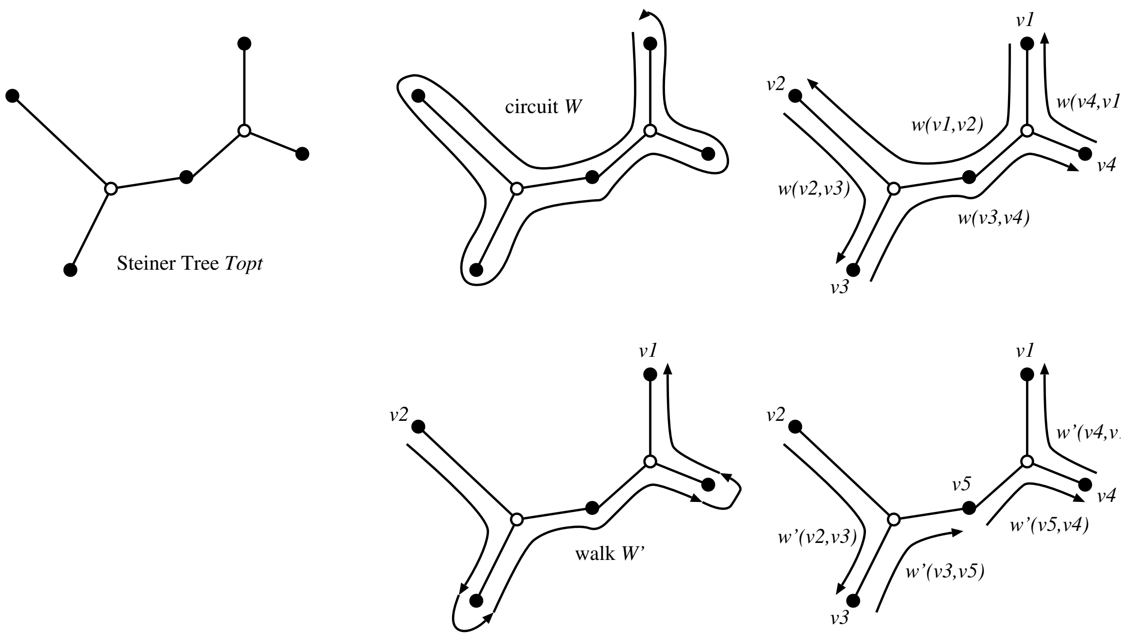
Theorem 6.4. Let T_{opt} and G_5 be a Minimum Steiner Tree (exact solution) and an approximated solution, respectively. Then the following holds for any problem instance.

$$C(G_5) \leq 2(1 - \frac{1}{\ell}) \cdot C(T_{opt})$$

where ℓ is the minimum number of leaves over all minimum Steiner trees.

Proof

1. Let T_{opt} be a minimum Steiner tree of G with respect to the set of required vertices R.
2. Visiting vertices in the depth first search fashion, we construct a circuit W passing every edge twice (in the opposite directions). The total cost of this circuit becomes $2C(T_{opt})$.



3. Separate the circuit W into paths, each from one leaf to another leaf. Labeling vertices appropriately, these paths can be written as

$$w(v_1, v_2), w(v_2, v_3), \cdots, w(v_{\ell-1}, v_\ell), w(v_\ell, v_1)$$

4. We can always find a longest path $w(v_i, v_{i+1})$ among those paths. With respect to the path length, we have

$$C(w(v_i, v_{i+1})) \geq \frac{1}{\ell} \times 2C(T_{opt})$$

Next we let W' be the walk constructed from W by removing $w(v_i, v_{i+1})$. Then

$$C(W') \leq 2(1 - \frac{1}{\ell})C(T_{opt})$$

and W' connects all vertices in R.

5. Next we separate the walk W' into paths (R-paths), each from one vertex in R to another vertex in R. Let $w'(v, w)$ be one of such paths, and let $p(v, w)$ be a shortest path (one of shortest paths) between v and w on G. Then we have,

$$C(p(v, w)) \leq C(w'(v, w))$$

6. On $G_1(R, E_1)$, we choose edges corresponding to R-paths of W', and construct $G_1'(R, E_1')$;

$$G_1'(R, E_1'), E_1' = \{(v, w) | \text{R-path } w'(v, w) \text{ of } W'\}$$

Then, $G_1'(R, E_1')$ includes all vertices of G_1, and is connected subgraph of G_1. Since G_2 is a minimum spanning tree of G_1, we have

$$C(G_2) \leq C(G_1') \leq C(W')$$

7. The construction procedure (Step 3 to 5) reduces cost.

$$C(G_5) \leq C(G_4) \leq C(G_3) \leq C(G_2)$$

8. As a result, we have

$$C(G_5) \leq C(W') \leq 2(1 - \frac{1}{\ell})C(T_{opt})$$

$\square$

6.4.3 Exercise

[1] The traveling salesperson problem (optimization version) can be described as follows: Given a set of cities $\boldsymbol{C} = \{c_1, c_2, \cdots, c_n\}$ and the cost of moving from one city to another $d(c_i, c_j) \in Z^+$, find a minimum cost tour of all the cities in $\boldsymbol{C}$, which visits each city just once.

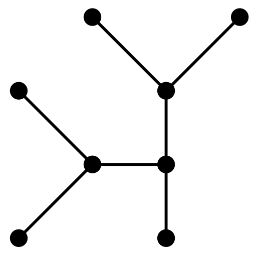
In the following, we assume that the cost is (1) finite, (2) symmetric $({}^\forall c_i, c_j \in \boldsymbol{C},\ d(c_i, c_j) = d(c_j, c_i))$, and (3) it obeys the triangle inequality, $({}^\forall c_i, c_j, c_k \in \boldsymbol{C},\ \ d(c_i, c_j) + d(c_j, c_k) \geq d(c_i, c_k))$.

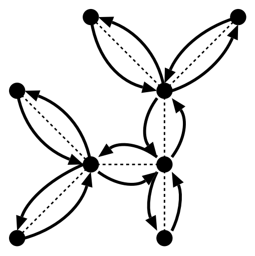
The following shows one simple approximation algorithm for the traveling salesperson problem.

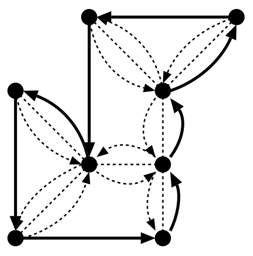
step 1: Find a minimum spanning tree T of the graph having $d(c_i, c_j)$ as the edge weight.

step 2: Replace each edge of T into two arcs having different directions.

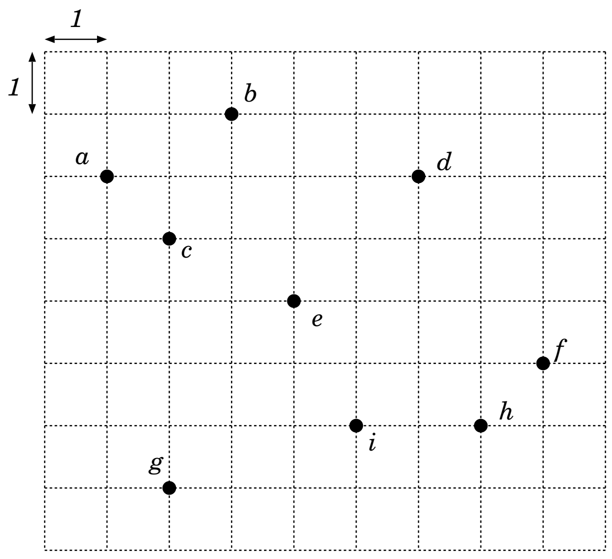
step 3: Choose one city arbitrary, and visit cities one by one along arcs obtained in step 2. If the destination city c_i of an arc has been visited before, then skip the city c_i and find next city using the arc originated from c_i.


A spanning tree


Twice around the tree


A tour, with shortcuts

With respect to the following instance of the traveling salesperson problem, find a solution by using the above approximation algorithm. Note that the cost of moving from one city to another is assumed to be evaluated by the Euclidean distance between two cities.



[2] Let P_{MST} be the cost of the tour obtained by the approximation algorithm given in the previous problem, and P_{OPT} be the cost of the optimum tour. Prove that P_{MST} is bounded above as follows.

$$P_{MST} < 2P_{OPT}$$

(Hint: Use the cost of minimum spanning tree for associating P_{MST} with P_{OPT}.)

6.5 Search Methods

The principle of search method is to subsequently visit a number of feasible solutions. In some problems, we can find an optimal solution after some iterations.

6.5.1 Configuration Graph

Given a problem instance p, let S_p be the set of feasible solutions. The configuration graph for p is a mapping $N : S_p \to 2^{S_p}$.

- $s_j \in N(s_i)$ is called a neighbor solution of s_i.

- Configuration graph is usually represented as a directed graph $C_p = (S_p, A_p)$ with

$$A_p \quad = \quad \{(s_i, s_j) | s_i \in S_p, s_j \in N(s_i)\}$$

- Let $c(s)$ be the value of the objective function for s. If $c(s_i) \leq c(s), ^\forall s \in N(s_i)$ for $s_i \in S_p$, then s_i is called a local optimal (minimum) solution. If $c(s_i) \leq c(s), ^\forall s \in S_p$, then s is called a global optimal (minimum) solution.
- If the configuration graph is strongly connected, and any local optimal solution is also a global optimal solution, then the configuration graph is called convex.

Fig **??** shows the configuration graph of a problem with $S_p = \{s_1, s_2, s_3, s_4, s_5, s_6\}$. Value associated with each vertex represents the value of objective function for the solution. In this example, only s_1 is the global minimum solution.

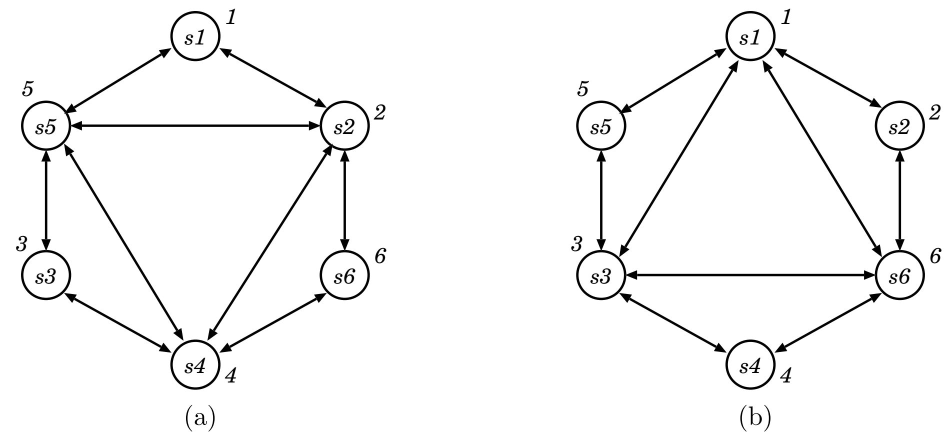


Figure 6.10: Example of a configuration graph.

In Fig **??**(a), $N(s_3) = \{s_4, s_5\}$ and s_3 is a local minimum solution. On the other hand, in Fig **??**(b), $N(s_3) = \{s_1, s_4, s_5, s_6\}$ and s_3 is not a local minimum solution. (The configuration graph Fig **??**(b) is convex.)

6.5.2 Search method

Search of a solution space is a directed walk $w : s \rightsquigarrow s_*$ on $C_p = (S_p, A_p)$. s_* is the solution obtained of this search.

- In most cases, it is too much time consuming to construct a configuration graph completely.
- An individual search algorithm is defined by (1) an algorithm of generating neighbor solutions, and (2) rule to select one among neighbor solutions.

- In search method, it is important to guarantee the existence of a walk from each solution in S_p to an optimal solution. However, since we do not know an optimal solution in advance, it is replaced by the guarantee on the existence of a walk form each vertex to any other vertex (that is, the configuration graph is strongly connected).
- If the value of objective function is decreasing along the walk, the search is called greedy. Otherwise, it is called non-greedy search.
- If the configuration graph is convex, we can find a global optimal solution by a greedy search.