Shu Lu *and* Quoc Tran-Dinh

Introduction to Optimization

From Linear Programming to Nonlinear Programming

December 5, 2019

STOR-UNC-Chapel Hill

Contents

Chapter 1

Introduction

1.1 Introduction

The process of solving a real-world problem is often complicated and consists of several stages. One often needs to collect and analyze data before proceeding to develop a mathematical model. Figure 1.1 illustrates the three main stages of such a process.

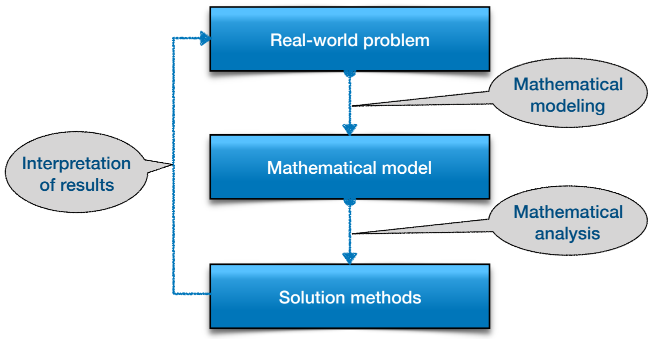


Fig. 1.1 Three main stages of a process of solving real-world problems using mathematical tools

In the mathematical modeling stage, a real-world problem is modeled into a mathematical problem using mathematical concepts such as variables, parameters, equations, constraints and objective functions. Once such a model is in place, we can use mathematical tools to develop algorithms or apply computer software to solve it. This stage is seen as the mathematical analysis stage. The results/solutions of the mathematical model are then analyzed and interpreted in the original real-world setting. If we find a solution that achieves the original goal, then we can recommend implementing it in practice. Otherwise, we should go back to identify issues within the process and fix them, and repeat the process.

In this course, we will cover basic concepts in optimization, fundamental theory in linear programing, brief introductions to nonlinear programming and integer programming, as well as representative solution methods for linear and integer programming. We will also use examples to show how to build optimization models for practical problems in production planning, inventory management, network flow optimization, machine learning, and so on.

1.2 What is mathematical optimization?

A mathematical optimization problem is also called a mathematical programming problem (or a mathematical program). It is the problem of finding the best solutions among all feasible solutions.

Example 1.1. As a very simple example, the problem of finding a rectangle with the maximum area among all rectangles of a fixed perimeter can be modeled as a mathematical optimization problem.

1.2.1 Definitions

An optimization problem must consist of three components:

- **Decision variables** (shortly, variables): Decision variables are the unknown variables which describe the decisions to be made.
- **The objective function:** The objective function is a function of the decision variables that reflects the objective of the decision making. Each optimization problem considered in this course has a single objective function. (Optimization problems with more than one objective functions are called multi-objective optimization problems and are beyond the scope of this course.)
- **Constraints:** These are requirements on values of the decision variables.

Formally, an *optimization problem* **seeks** *values* of the *decision variables* that **optimize** (maximize or minimize) an *objective function* while **satisfying** all the given *constraints*.

Mathematical formulation: Mathematically, an optimization problem can be written as follows:

$$\begin{cases} \min\limits_{x_1,\cdots,x_n} \ (\text{or} \ \max\limits_{x_1,\cdots,x_n}) & f(x_1,\cdots,x_n) \\ \text{subject to} & (x_1,\cdots,x_n) \in \mathscr{X}. \end{cases} \tag{1.1}$$

In the above formulation, $x_1, x_2, \cdots, x_n$ are n decision variables, f is the objective function that returns a numerical value for any input $(x_1, \cdots, x_n)$, and $\mathscr{X}$ is a subset in $\mathbb{R}^n$ and it contains all values of the vector $(x_1, \cdots, x_n)$ that satisfy the constraints. The notation $\min_{x_1,\cdots,x_n}$ ($\max_{x_1,\cdots,x_n}$) means that the goal is to minimize (maximize) the value of $f(x_1, \cdots, x_n)$ among all vectors $(x_1, \cdots, x_n)$ in the set $\mathscr{X}$.

The following basic concepts in optimization will be used frequently in this course.

- **Feasible solutions:** A feasible solution is an assignment of values to the decision variables such that all the constraints are simultaneously satisfied. If there are n decision variables, then each feasible solution is an n-dimensional vector.

  Example 1.2. Consider the following example:

  $$\min_{x \in \mathbb{R}^2} \left\{ x_1^2 + (x_2 - 1)^2 \mid x_1 + x_2 = 1,\ 0 \le x_1 \le 1,\ 0 \le x_2 \le 1 \right\}. \tag{1.2}$$

  Any point $(x_1, x_2) \in \mathbb{R}^2$ that satisfies $x_1 + x_2 = 1$, $0 \le x_1 \le 1$ and $0 \le x_2 \le 1$ simultaneously is a feasible solution. For instance, $x = (0, 1)$ is feasible, while $x = (\frac{1}{3}, \frac{1}{3})$ is not feasible as it does not satisfy $x_1 + x_2 = 1$.

- **The feasible set:** The feasible set (or the feasible region) is the set of all feasible solutions. When there are n decision variables, the feasible set is a subset of $\mathbb{R}^n$. In the formulation (1.1), $\mathscr{X}$ is the feasible set. In the above example, $\mathscr{X} := \left\{ x \in \mathbb{R}^2 \mid x_1 + x_2 = 1,\ 0 \le x_1 \le 1, 0 \le x_2 \le 1 \right\}$ is the feasible set of (1.2).
- **Optimal solutions:** An optimal solution, if any, is a feasible solution that achieves the best objective value among all feasible solutions. For example, the point $x^* = (0, 1)$ is an optimal solution of (1.2) since $f(x^*) = 0 \le f(x) = x_1^2 + (x_2 - 1)^2$ for all $x = (x_1, x_2) \in \mathscr{X}$. In fact, x^* is the unique optimal solution of (1.2).
- **The optimal value:** The optimal value is the objective value achieved at an optimal solution. For example, the optimal value of (1.2) is 0.

We note that an optimization problem can have more than one optimal solution or no optimal solution at all. For instance, consider the following problem with three decision variables:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^3} & x_1^2 + 2x_2 + x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \le 3, \\ & 4x_2 + 2x_3 \ge 1, \\ & x_1, x_2, x_3 \ge 0, \end{cases}$$

The vector $x^* = (0, \frac{1}{4}, \frac{1}{2})$ is an optimal solution with the optimal value $f(x^*) = \frac{1}{2}$. The vector $x^* = (0, 0, \frac{1}{2})$ is another optimal solution with the same optimal value $f(x^*) = \frac{1}{2}$. In fact, the set of optimal solutions is $\mathscr{X}^* = \left\{x \in \mathbb{R}^3 \mid x_1 = 0,\ 2x_2 + x_3 = \frac{1}{2},\ x_2 \ge 0,\ x_3 \ge 0\right\}$. Therefore, this problem has an infinite number of optimal solutions.

The following example shows that an optimization problem may not have any optimal solution:

$$\min_{x\in\mathbb{R}^2} \{x_1 + x_2 \mid x_1 + x_2 \le 0,\ \ 2x_1 + x_2 \le 3\}.$$

This problem has infinitely many feasible solutions, but it does not have an optimal solution. For any feasible solution x, we can find another feasible solution $\bar{x}$ such that $f(\bar{x}) < f(x)$.

For the maximum area rectangle problem above, we can define x as the width and y as the length of a rectangle. Hence, x and y are referred to as decision variables. We have two constraints as $x, y \ge 0$, and $2(x + y) = d$, where d is the given perimeter of the rectangle. Hence, we can write $\mathscr{X} = \left\{(x,y) \in \mathbb{R}^2 \mid x, y \ge 0,\ x + y = \frac{d}{2}\right\}$ as the feasible set. The objective function is $f(x,y) = xy$, which we need to maximize. By putting the objective function and the constraints together, we obtain an optimization problem of the form (1.1).

Continuous vs. discrete: In a continuous optimization problem, values of the decision variables are allowed to be continuous numbers (real numbers with no gaps) in a given subset of $\mathbb{R}^n$. In contrast, the values of decision variables are discrete (finite, or countably infinite) in a discrete optimization problem. The main focus of this course is on continuous optimization problems, with a brief introduction to discrete problems.

1.2.2 More examples

Let us provide more motivating examples of optimization problems used in different fields.

*Example 1.3 (**Optimizing resources**).* We are making an open-topped box from some material so that it can contain 15 cubic inches. We want to make the bottom thicker than the sides of the box. The material for the bottom costs \$0.5 per square inch, and the material for the sides costs \$0.35 per square inch. Our goal is to find the dimensions (width, length, and height) of the box to minimize the total material cost.

Mathematical model: To model this problem, we denote by x, y and z the width, length and height of the box, respectively. Here, $x \geq 0$, $y \geq 0$ and $z \geq 0$. The volume of the box is $V = xyz = 15$, while the total material cost is $C = 0.5xy + 0.7(xz + yz)$. Hence, the problem becomes

$$\min_{x,y,z} \{C = 0.5xy + 0.7(xz + yz)\} \quad \text{subject to} \quad xyz = 15, \ \ x \geq 0, \ y \geq 0, \ z \geq 0.$$

This problem can be solved analytically to obtain a unique optimal solution.

*Example 1.4 (**The transportation problem**).* There are three warehouses WH_1, WH_2 and WH_3 for some commodity located in nearby cities. The minimum demand at each warehouse is given in the last row of Table 1.1. Two factories F_1 and F_2 can supply this commodity to the warehouses, and the capacities of their supply are given in the last column of Table 1.1. The cost to transport each unit of commodity from each factory to each warehouse is also given in Table 1.1. The goal is

	WH_1	WH_2	WH_3	Supply
F_1	4	3	7	15
F_2	5	4	6	25
Demand	8	12	20	40

Table 1.1 Demands, supply capacities, and transportation costs.

to find a transportation plan to minimize the total shipping cost, while fulfilling all the demand and supply requirements.

Mathematical model: To formulate this into an optimization problem, we denote by x_{ij} the units of commodity to transport from F_i to WH_j for $i = 1,2$ and $j = 1,2,3$. We must have $x_{ij} \geq 0$. The total cost of shipping is

$$c(x) = 4x_{11} + 3x_{12} + 7x_{13} + 5x_{21} + 4x_{22} + 6x_{23}.$$

To obey the capacities of supply, we have two constraints $x_{11}+x_{12}+x_{13} \leq 15$ and $x_{21}+x_{22}+x_{23} \leq 40$. To meet the demands, we write three constraints $x_{11}+x_{21} \geq 8$, $x_{12}+x_{22} \geq 12$ and $x_{13}+x_{23} \geq 20$.

Finally, we can write the overall problem as

$$\min_{x} \; 4x_{11}+3x_{12}+7x_{13}+5x_{21}+4x_{22}+6x_{23}$$

subject to

$$\left.\begin{aligned} x_{11}+x_{12}+x_{13} &\leq 15 \\ x_{21}+x_{22}+x_{23} &\leq 25 \end{aligned}\right\} \text{(supply constraints)}$$

$$\left.\begin{aligned} x_{11}+x_{21} &\geq 8 \\ x_{12}+x_{22} &\geq 12 \\ x_{13}+x_{23} &\geq 20 \end{aligned}\right\} \text{(demand constraints)}$$

$$x_{ij} \geq 0, \;\; i = 1,2; \;\; j = 1,2,3.$$

*Example 1.5 (**The knapsack problem**).* There are n objects that have different weights $w_1, w_2, \ldots, w_n$ in certain weight units (e.g., in pounds), and different values $v_1, v_2, \ldots, v_n$ (e.g., in dollars). One needs to select a number of objects to pack into a box that can hold a maximum of W weight units. The goal is to select the right objects such that total value is maximized.

Mathematical model: Let x_i be a binary variable defined as follows:

- $x_i = 1$ if the i-th object is selected, and
- $x_i = 0$ if the i-th object is not selected.

Then, the total value is computed as $V = \sum_{i=1}^n x_i v_i$. We also have one constraint on the maximum weight capacity of the box as $\sum_{i=1}^n x_i w_i \leq W$.

To this end, the knapsack problem can be formulated into the following optimization problem:

$$\begin{cases} \max\limits_{x} & \sum\limits_{i=1}^{n} v_i x_i \\ \text{s.t.} & \sum\limits_{i=1}^{n} w_i x_i \leq W, \\ & x_i \in \{0,1\}. \end{cases}$$

Since the values of x_i are binary, i.e., $x_i \in \{0,1\}$, this problem is a discrete optimization problem.

1.2.3 Optimization in operations research

Operations research is a discipline dealing with the application of advanced analytical methods, such as optimization, simulation and stochastic models, to make better decisions. Operations research techniques are widely used in areas such as

- Computing and information technologies
- Environment, energy, and natural resources
- Financial engineering
- Manufacturing, service science, and supply chain management
- Marketing science
- Policy modeling and public sector work
- Revenue management
- Transportation.

1.3 Linear programming and nonlinear programming

Continuous optimization problems can be further classified into linear optimization (linear programming) and nonlinear optimization (nonlinear programming) problems. This classification is somewhat unbalanced, because nonlinear programming covers many more problems than linear programming. Nonetheless, understanding of the mathematical properties and solution algorithms for linear programming problems is very important for understanding the more complicated nonlinear programming problems.

1.3.1 Linear programming

Linear programming (LP) is a special class of optimization problems, that has one linear objective function and possibly many linear constraints. More specifically, suppose that there are n decision variables, $x_1, \cdots, x_n$. A linear objective function is of the form

$$f(x_1, x_2, \cdots, x_n) \overset{\text{def}}{=} c_1x_1 + c_2x_2 + \cdots + c_nx_n.$$

A linear constraint is of the form

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i,$$

or

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i,$$

or

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i,$$

where $a_{i1}, \ldots, a_{in}$, $c_1, \cdots, c_n$ and b_i are constants for all i.

- We can have three different types of constraints: equality (E) constraint ($=$), "less than or equal to" (LE) constraint ($\leq$), or "great than or equal to" (GE) constraint ($\geq$).
- We can also have three different types of variables: nonnegative variables ($x_i \geq 0$), non-positive variables ($x_i \leq 0$), or free variables (i.e., variables not subject to sign constraints).

We can write an LP problem as follows:

$$\boxed{\left\{\begin{array}{ll} \min\limits_{x_1,\cdots,x_n} & z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to} & \\ & a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq (\geq \text{ or } =) b_i, \text{ for } i = 1, \cdots, m, \\ & x_i \geq 0 \ \ (x_i \leq 0, \text{ or } x_i \text{ is free}). \end{array}\right. \qquad \text{(LP)}}$$

Here, the notation $\min\limits_{x_1,\cdots,x_n}$ means that the goal is to minimize the objective z over the variables $x_1, \cdots, x_n$ subject to the constraints. By noting that $\min z = -\max(-z)$, we can convert any min problem into a max problem, and vice versa. The objective of a minimization problem is often called the cost/risk function, while the objective of the maximization problem is often referred to as the profit or utility function.

As a simple example, consider the following problem:

$$\left\{\begin{array}{lll} \min\limits_{x_1,x_2,x_3} & 2x_1 + x_2 + 3x_3 & \\ \text{subject to} & x_1 + x_2 + x_3 = 10 & \text{(E)}, \\ & 2x_1 + 3x_2 - x_3 \leq 8 & \text{(LE)}, \\ & 3x_1 + 2x_2 + x_3 \geq 5 & \text{(GE)}, \\ & x_1 \geq 0, \ x_2 \leq 0. & \end{array}\right.$$

Since the objective function and the constraints are all linear, the problem is an LP. It has three types (E, LE, and GE) of constraints, and three types of variables: x_1 is nonnegative, x_2 is nonpositive, and x_3 is free.

Example 1.6. A dietician wishes to plan a meal using ground beef, potatoes, and spinach to satisfy minimum daily requirements on protein, carbohydrates, and iron. The nutritional makeup of each ounce of foodstuff (in the appropriate nutritional units) is given below.

	Nutrient		
Food	protein	carbohydrates	iron
beef	20	10	5
potatoes	10	30	6
spinach	6	7	20

The minimum daily requirements of protein, carbohydrates, and iron in the diet are 500, 400, and 75, respectively, and the cost of beef, potatoes, and spinach are \$0.35, \$0.20, and \$0.15 per ounce, respectively.

Goal: How should the dietician plan the menu to satisfy the minimum requirements with the lowest total cost?

Mathematical model: To model this problem as an optimization problem, we first choose the decision variables x_1, x_2, and x_3 to denote the amount in ounces of beef, potatoes, and spinach to put in the meal, respectively. The objective function is the total cost to be minimized. The minimum requirements on the nutritional makeup need to be considered as constraints. In addition, all three decision variables should be nonnegative. This leads to the following optimization problem, which is a linear program:

$$\begin{cases} \min\limits_{x_1,x_2,x_3} \; z = 0.35x_1 + 0.20x_2 + 0.15x_3 & \\ \text{subject to} & \\ \quad 20x_1 + 10x_2 + 6x_3 & \geq 500, \\ \quad 10x_1 + 30x_2 + 7x_3 & \geq 400, \\ \quad 5x_1 + 6x_2 + 20x_3 & \geq 75, \\ \quad x_1 & \geq 0, \\ \quad x_2 & \geq 0, \\ \quad x_3 & \geq 0. \end{cases}$$

1.3.2 Nonlinear programming

A nonlinear program (NLP) is an optimization problem that has a nonlinear objective function, or some nonlinear constraints. Below is a list of sub-classes of problems that belong to NLP:

- quadratic programs (QPs);
- second order cone programs (SOCPs);
- semidefinite programs (SDPs);
- convex programs, and
- nonconvex programs.

We give a few examples.

Example 1.7. Consider the following optimization problem:

$$\begin{cases} \min\limits_{x_1,x_2,x_3} & \frac{1}{2}x_1^2 + x_2^2 + 2x_3^2 + x_2x_3 - x_1 + x_2 - 2x_3 \\ \text{subject to} & x_1 + x_2 + x_3 = 1 \\ & x_1 - 2x_2 \le 0 \\ & x_1, x_2, x_3 \ge 0. \end{cases}$$

The objective function of this problem is $\frac{1}{2}x_1^2 + x_2^2 + 2x_3^2 + x_2x_3 - x_1 + x_2 - 2x_3$, which is nonlinear (indeed, it is quadratic). Hence, the problem is nonlinear.

Example 1.8. We consider the following problem

$$\begin{cases} \min\limits_{x\in\mathbb{R}^3} & x_1 + x_2 + x_3 \\ \text{subject to} & x_1^2 + x_2^2 + x_3^2 \le 1. \end{cases}$$

While the objective function is linear, the constraint is nonlinear. Hence, this is a nonlinear program (indeed, it is a second order cone program).

Example 1.9 (Weber problem). Given m points $z^{(1)} = (x_1, y_1), \ldots, z^{(m)} = (x_m, y_m)$ in the xy-plane, find a point $\bar{z} = (\bar{x}, \bar{y})$ such that the total distance from $\bar{z}$ to all the points $z^{(i)}$ for $i = 1, \ldots, m$ is minimized. This problem can be formulated as follows:

$$\min_{\bar{z}\in\mathbb{R}^2} \sum_{i=1}^{m} d(\bar{z}, z^{(i)}),$$

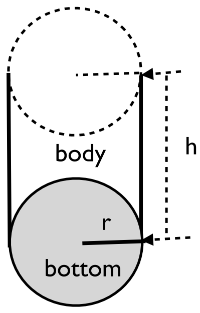
where $d(\bar{z}, z^{(i)}) \overset{\text{def}}{=} \sqrt{(\bar{x}-x_i)^2 + (\bar{y}-y_i)^2}$ is the distance from $\bar{z}$ to $z^{(i)}$ for $i = 1, \cdots, m$. Clearly, this problem is nonlinear since the objective function is nonlinear, even though it does not have any constraint.

1.4 Exercises

Exercise 1.1. Furnco manufactures desks and chairs. Each desk uses 4 units of wood, and each chair uses 3 units. A desk contributes \$40 to profit, and a chair contributes \$25. Marketing restrictions require that the number of chairs produced be at least twice the number of desks produced. There are totally 20 units of wood available.

1. Assume that all decision variables are continuous, formulate an LP to maximize Furnco's profit: define the variables, write down the objective function, and all constraints.
2. Is $(0,0)$ a feasible solution?
3. Is $(-3,2)$ a feasible solution?
4. Is $(2,4)$ a feasible solution?
5. Graphically solve the LP problem obtained from Item 1. Label the vertices (i.e., corners) of the feasible set with their coordinates. Write down the optimal solution(s) and the optimal value.

Exercise 1.2. A factory is manufacturing open-top cylinder containers using new polymeric material. Each square inch of the material costs \$0.4 to make the bottom, and costs \$0.25 to make the sides (body). The factory wants to find suitable radius r and height h of each container.



(a) If the volume of each container is fixed at 15 cubic inches, then find r and h to minimize the total material cost of each container. Model this problem into an optimization problem and solve it.

(b) If the total material cost is fixed at \$7.5 per container, then find r and h to maximize the volume V, and compute the value of V in this case. Model this problem into an optimization problem. Solve the resulting problem.

Exercise 1.3. U.S. Labs manufactures mechanical heart valves from the heart valves of pigs. Different heart operations require valves of different sizes. U.S. Labs purchases pig valves from three different suppliers. The cost and size mix of the valves purchased from each supplier are given in the following table.

Supplier	Cost per Valve (\$)	Percent Large	Percent Medium	Percent Small
1	5	40	40	20
2	4	30	35	35
3	3	20	20	60

For example, any K pig valves from supplier 1 will contain $0.4K$ large, $0.4K$ medium and $0.2K$ small ones; customers must purchase the mix.

Each month, U.S. Labs needs at least 500 large, 300 medium, and 300 small valves. Because of limited availability of pig valves, at most 700 valves per month can be purchased from each supplier.

Formulate an LP that can be used to minimize the cost of acquiring the needed valves: define the variables, write down the objective function, and all constraints. Do not try to solve the obtained LP.

Exercise 1.4 (Road lighting). [3] A road is divided into n segments that are illuminated by m lamps. Let us denote by p_j the power of the j-th lamp, and I_i the illumination of the i-th segment, which is assumed to be $I_i = \sum_{j=1}^m a_{ij} p_j$, where a_{ij} are known coefficients. Let I_i^* be the desired illumination of the segment i of the road, which is given. The goal is to choose the lamp power p_j so that the illuminations I_i are close to the desired illumination I_i^*.

Model this problem into a reasonable LP problem. Note that the wording of the problem is loose and there may be more than one possible formulation.

Hints: We can measure the difference between I_i and I_i^* as $|I_i - I_i^*| \leq t_i$, and try to minimize the sum of upper bound errors $\sum_{i=1}^m t_i$.

Exercise 1.5. A factory is manufacturing and selling two types of commodity, called C1 and C2. Every unit of C1 requires 3 machine hours, and every unit of C2 requires 4 machine hours to manufacture. The material cost of C1 is \$3 per unit, but it can be sold at the price of \$6 per unit when it is completed. The material cost of C2 is \$2 per unit, and it can be sold at the price of \$5.4 per unit.

Due to the limitation of resources, the factory has at most $20,000$ machine hours per week, and at most \$4,000 of cost per week. Moreover, 45% of the sales revenues from C1 and 30% of the sales revenues from C2 will be made available to financial operations during the current week. The aim of the factory is to maximize the net income subject to the availability of resources.

(a) Model this problem into an optimization problem. Is it a linear program?

(b) Graph the feasible set of this problem and compute all the vertices of the feasible set.

Exercise 1.6. We are going to make two different types of beer: light beer and dark beer using malt, hops and yeast. Currently, we have the following raw material in our inventory:

Malt	75 units,
Hops	60 units,
Yeast	50 units.

Each type of beer requires different amount of malt, hops and yeast as given in the following table:

Requirement per gallon			
	Malt	Hops	Yeast
Light beer	2	3	2
Dark beer	3	1	5/3

If we sell the beers, then the light beer brings \$2.00/gallon profit, and the dark beer brings \$1.00/gallon profit. Given that our customers are willing to buy whatever is made available. Our goal is to make a production plan so that we can maximize our profit while satisfying all the requirements on our available material in our inventory.

(a) Model this problem into an optimization problem. Is it a linear program? Be sure to define all of your variables.

(b) Graph the feasible set of this problem and find all the vertices of this feasible set. Check the objective value at these vertices and show the one that gives the maximum profit.

Exercise 1.7. Given a system of four machines that are configured in a given layout in a two-dimensional plane. These four machines are located at a given coordinate (x_i, y_i) as $(3,0)$, $(0,-3)$, $(-2,1)$ and $(1,4)$ for $i = 1, \cdots, 4$, respectively. Assume that we want to add a new machine to the system at a location (x, y). Our goal is to find the location of this new machine so that the total of distances from this new machine to other four machines is minimized. Here, the distance between two points $P(x,y)$ and $Q(\hat{x},\hat{y})$ is measured as $d(P,Q) = |x-\hat{x}| + |y-\hat{y}|$ (which is different from the length between P and Q).

(a) Model this problem as an optimization problem. Define variables, objective functions, and constraints (if any).
(b) Can you geometrically solve this problem? If yes, please solve it.
(c) Transform the resulting problem into a linear program.

Exercise 1.8. A store has hired n staffs to work for a short period during Christmas time. The manager of the store plans to assign each staff one job among n different available jobs. Since each staff can do each job with different cost, he wants to assign each job to the right person to minimize the total cost. Formulate this problem as an optimization problem such that the following constraints are satisfied:

- if the i-th staff is assigned to the j-th job, then it costs c_{ij} dollars for $i, j = 1, \cdot, n$;
- each staff must be only assigned exactly to one job.

Chapter 2

Overview of Linear Algebra

In this chapter, we review some concepts and tools from linear algebra that will be used in this course. We focus on the definitions and properties of vectors and matrices, linear systems, matrix inverses, elementary row operations, and the Gauss-Jordan method.

2.1 Vectors and vector operations

2.1.1 Definitions and examples

An n-dimensional row vector is an array of numbers arranged in one row as

$$x = (x_1, x_2, \cdots, x_n).$$

For each $i = 1, \cdots, n$, x_i is the ith entry (or element, component, coordinate) of the row vector x. Alternatively, an n-dimensional column vector is an array of numbers arranged in one column as

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

The notation $\mathbb{R}^n$ stands for the space of all n-dimensional column vectors (or row vectors, depending on the context). For example, $x = (2, 3, -1)$ is a row vector in $\mathbb{R}^3$, and $x = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$ is a column vector in $\mathbb{R}^2$.

The transpose operator, denoted by $(\cdot)^T$, is used to transform a row vector into a column vector, and vice versa. In other words, if x is an n-dimensional row (or column) vector, then x^T is the n-dimensional column (or row) vector with the same elements placed in the same order.

Vectors we use in this course will mostly be column vectors. We often write an n-dimensional column vector as $x = (x_1, \cdots, x_n)^T$, the transpose of a row vector.

The vector of all zeros is called the **zero vector**. That is

$$0 = (0, 0, \cdots, 0)^T.$$

The vector of all ones is denoted by

$$1 = (1, 1, \cdots, 1)^T.$$

The vector

$$\mathbf{e}_i = (0, 0, \cdots, 0, \underbrace{1}_{\text{the } i\text{-th entry}}, 0, \cdots, 0)^T$$

is called the i-th unit vector of $\mathbb{R}^n$. The space $\mathbb{R}^n$ has n unit vectors $\mathbf{e}_1, \cdots, \mathbf{e}_n$. For example, $\mathbb{R}^3$ has three unit vectors $\mathbf{e}_1 = (1,0,0)^T$, $\mathbf{e}_2 = (0,1,0)^T$ and $\mathbf{e}_3 = (0,0,1)^T$.

2.1.2 Basic operations

Given two n-dimensional column vectors $x, y \in \mathbb{R}^n$, and a number $c \in \mathbb{R}$, the following operations can be defined.

Addition. The sum of x and y, denoted by $x + y$, is a vector $z = (z_1, \cdots, z_n)^T$ such that $z_i = x_i + y_i$ for $i = 1, \cdots, n$.

Subtraction. The difference of x and y, denoted by $x - y$, is a vector $z = (z_1, \cdots, z_n)^T$ such that $z_i = x_i - y_i$ for $i = 1, \cdots, n$.

Scalar multiplication. The product of the scalar c and the vector x, denoted by cx, is a vector $z = (z_1, \cdots, z_n)^T$ such that $z_i = cx_i$ for $i = 1, \cdots, n$.

Inner product. The inner product of x and y, denoted by $x^T y$, is a number computed by

$$x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

Norm. The norm (also called the Euclidean norm) of a vector x, denoted by $\|x\|_2$, is a number computed as

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Euclidean distance. The Euclidean distance between x and y is given by

$$\|x-y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}.$$

The following properties are trivial:

$$\begin{aligned}
&x-y = x+(-1)y,\\
&x+y = y+x,\\
&c(x+y) = cx+cy, \text{ and}\\
&\|x\|_2 \geq 0, \text{ and } \|x\|_2 = 0 \text{ iff } x = 0.\\
&\|x\|_2 = \sqrt{x^Tx}, \text{ or } \|x\|_2^2 = x^Tx.
\end{aligned}$$

The norm of x is also the Euclidean distance between x and the origin. For example, given $x \in \mathbb{R}^2$, and the origin $0 = (0,0)^T$, then $\|x\| = \sqrt{x_1^2+x_2^2} = \sqrt{(x_1-0)^2+(x_2-0)^2}$, which is exactly the Euclidean distance between x and 0. Note that we do not have the division operator (or quotients) between two vectors.

Example 2.1. Here are some examples. Given $x = (1,2)^T \in \mathbb{R}^2$, $y = (-3,2)^T \in \mathbb{R}^2$ and $c = 3$. We have

$$x+y = \begin{pmatrix}-2\\4\end{pmatrix}, \ x-y = \begin{pmatrix}4\\0\end{pmatrix}, \ 3y = \begin{pmatrix}-9\\6\end{pmatrix}, \ x^Ty = -3+4 = 1, \text{ and } \|x\|_2 = \sqrt{1^2+2^2} = \sqrt{5}.$$

2.2 Matrices and matrix operations

2.2.1 Definitions and special cases

An $m \times n$ **matrix** is a rectangular array of numbers arranged in m rows and n columns as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n}\\ A_{21} & A_{22} & \cdots & A_{2n}\\ \vdots & \vdots & & \vdots\\ A_{m1} & A_{m2} & \cdots & A_{mn}\end{bmatrix}.$$

The entry that appears at the intersection of the i-th row and j-th column of A is called the (i,j) entry of A. It is written as A_{ij} or sometimes as a_{ij}. The notation $A \in \mathbb{R}^{m\times n}$ or $(a_{ij})_{m\times n}$ means that A is an $m \times n$ matrix. We conventionally use square brackets to present matrices in an explicit form as above.

Example 2.2. The following matrices are of the dimensions 2×2, 2×3, 2×1 and 1×3, respectively, with entries such as $A_{11} = 1$, $A_{12} = 2$, $A_{21} = 3$, $A_{22} = 4$ and

$B_{21} = 4$, etc.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad D = \begin{bmatrix} 3 & 0 & 4 \end{bmatrix}.$$

Clearly, if a matrix has one row (or one column), it becomes a row vector (or a column vector). In other words, an $m \times 1$ matrix is an m-dimensional column vector, while a $1 \times n$ matrix is an n-dimensional row vector. In the above example, C is a 2-dimensional column vector, and D is a 3-dimensional row vector. Hence, an $m \times n$ matrix A is formed from n column vectors A_j of size m, or from m row vectors a_i^T of size n. Here, for each $i = 1, \cdots, m$ and each $j = 1, \cdots, n$,

$$A_j = \begin{pmatrix} A_{1j} \\ \vdots \\ A_{mj} \end{pmatrix}, \quad a_i = \begin{pmatrix} A_{i1} \\ \vdots \\ A_{in} \end{pmatrix}, \quad \text{and } a_i^T = (A_{i1}, \cdots, A_{in}).$$

Example 2.3. $A = \begin{bmatrix} 1 & 2 & -3 \\ 2 & 1 & 4 \\ 4 & -2 & 3 \end{bmatrix}$ is formed from 3 column vectors $A_1 = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$, $A_2 = \begin{pmatrix} 2 \\ 1 \\ -2 \end{pmatrix}$, and $A_3 = \begin{pmatrix} -3 \\ 4 \\ 3 \end{pmatrix}$, or from 3 row vectors $a_1^T = (1, 2, -3)$, $a_2^T = (2, 1, 4)$ and $a_3^T = (4, -2, 3)$.

A **zero** matrix is a matrix where all elements are zeros. For example, $O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ is the 2×2 zero matrix.

If the number of rows in a matrix equals the number of its columns, then this matrix is called a **square matrix**. For example, $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ is a 2×2 square matrix. An $n \times n$ square matrix is called a **diagonal matrix** if all of its off-diagonal entries are 0's. We often use $\mathrm{diag}(d_1, \cdots, d_n)$ to denote the diagonal matrix

$$\begin{bmatrix} d_1 & 0 & 0 & \cdots & 0 \\ 0 & d_2 & 0 & \cdots & 0 \\ 0 & 0 & d_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & d_n \end{bmatrix}.$$

A square matrix is called an **identity matrix** if its diagonal entries are all 1's, and its off-diagonal entries are all 0's. That is, the identity matrix is a special diagonal matrix with unit diagonal entries. We use $\mathbb{I}_m$ to denote the $m \times m$ identity matrix. For instance,

$$\mathbb{I}_1 = \begin{bmatrix} 1 \end{bmatrix}, \quad \mathbb{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbb{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.2.2 Basic matrix operations

There are a number of basic operations that can be applied to matrices.

Scalar-matrix multiplication: Given a matrix A and a scalar c, the product cA is computed by multiplying every entry of A by c. That is

$$cA = \begin{bmatrix} cA_{11} & cA_{12} & \cdots & cA_{1n} \\ cA_{21} & cA_{22} & \cdots & cA_{2n} \\ \vdots & \vdots & & \vdots \\ cA_{m1} & cA_{m2} & \cdots & cA_{mn} \end{bmatrix}.$$

Example 2.4. If $A = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}$, then $3A = \begin{bmatrix} 3 & 6 \\ -3 & 0 \end{bmatrix}$, and $-2A = \begin{bmatrix} -2 & -4 \\ 2 & 0 \end{bmatrix}$.

Addition: Two matrices of the same dimension can be added. Given two $m \times n$ matrices A and B, their sum $A + B$ is an $m \times n$ matrix computed by summing up the corresponding entries of A and B. That is

$$A + B = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} & \cdots & A_{1n} + B_{1n} \\ A_{21} + B_{21} & A_{22} + B_{22} & \cdots & A_{2n} + B_{2n} \\ \vdots & \vdots & & \vdots \\ A_{m1} + B_{m1} & A_{m2} + B_{m2} & \cdots & A_{mn} + B_{mn} \end{bmatrix}.$$

Example 2.5. If $A = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}$, $B = \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix}$, then $A+B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$.

Subtraction: Two matrices of the same dimension can be subtracted. Given two $m \times n$ matrices A and B, their difference $A - B$ is an $m \times n$ matrix computed by subtracting the entries of B from the corresponding entries of A. That is

$$A - B = \begin{bmatrix} A_{11} - B_{11} & A_{12} - B_{12} & \cdots & A_{1n} - B_{1n} \\ A_{21} - B_{21} & A_{22} - B_{22} & \cdots & A_{2n} - B_{2n} \\ \vdots & \vdots & & \vdots \\ A_{m1} - B_{m1} & A_{m2} - B_{m2} & \cdots & A_{mn} - B_{mn} \end{bmatrix}.$$

Example 2.6. If $A = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}$, $B = \begin{bmatrix} -1 & -2 \\ 2 & 1 \end{bmatrix}$, then $A-B = \begin{bmatrix} 2 & 4 \\ -3 & -1 \end{bmatrix}$.

Only matrices of the same dimension can be added or subtracted. It is obvious that $A - B = A + (-1)B$.

Transpose: Given an $m \times n$ matrix A, its transpose A^T is an $n \times m$ matrix obtained by turning rows of A into columns (and columns of A into rows). That is

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}^T = \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{m1} \\ A_{12} & A_{22} & \cdots & A_{m2} \\ \vdots & \vdots & & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{mn} \end{bmatrix}.$$

Example 2.7. If $A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 2 \end{bmatrix}$, then $A^T = \begin{bmatrix} 1 & -1 \\ 2 & 0 \\ 3 & 2 \end{bmatrix}$.

The transpose of an n-dimensional column vector is an n-dimensional row vector, and vice versa, as we have seen before.

Example 2.8. If $B = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, then $B^T = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$.

Matrix multiplication: Given two matrices $A = (a_{ij})_{m \times n} \in \mathbb{R}^{m \times n}$ and $B = (b_{ij})_{p \times q} \in \mathbb{R}^{p \times q}$, the product AB is well defined if $n = p$. When it is well defined, we denote the product AB by $C = (c_{ij})_{m\times q}$, which is an $m \times q$ matrix such that its entries are computed by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \quad \forall i = 1,\cdots m, \ \ j = 1,\cdots q.$$

Condition of matrix multiplication: Two matrices $A = (a_{ij})_{m\times n} \in \mathbb{R}^{m\times n}$ and $B = (b_{ij})_{p\times q} \in \mathbb{R}^{p\times q}$ can be multiplied as AB if the number of columns of A is equal to the number of rows of B, i.e., $n = p$.

We note that the fact AB is well defined does not mean that the product BA is also well defined.

Example 2.9. In the following calculations, the products AB are well defined.

If $A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 1 & 2 \end{bmatrix}$, then

$$AB = \begin{bmatrix} 1(1)+2(2)+3(1) & 1(3)+2(0)+3(2) \\ (-1)(1)+0(2)+2(1) & (-1)(3)+0(0)+2(2) \end{bmatrix} = \begin{bmatrix} 8 & 9 \\ 1 & 1 \end{bmatrix}.$$

If $A = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 3 & 0 \end{bmatrix}$, then

$$AB = \begin{bmatrix} 3(1) & 3(3) & 3(0) \\ 2(1) & 2(3) & 2(0) \\ 1(1) & 1(3) & 1(0) \end{bmatrix} = \begin{bmatrix} 3 & 9 & 0 \\ 2 & 6 & 0 \\ 1 & 3 & 0 \end{bmatrix}.$$

This product is also called the outer product of two vectors A and B.

If $A = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$, then

$$AB = \begin{bmatrix} 3(1)+2(3)+1(0) \end{bmatrix} = \begin{bmatrix} 9 \end{bmatrix}.$$

This becomes the inner product of two column vectors A^T and B.

2.2.3 Properties of matrix operations

Similar to operations on real numbers, matrix operations enjoy properties such as associativity and distributivity. The following lists basic properties satisfied by matrix operations. In this list, we use c to denote an arbitrary real number, and use A, B, C to denote matrices.

The dimensions of A, B and C need to be consistent for the involved operations to be well defined. For example, the first part of the third item below says that when the dimensions of A, B C are such that $A+B$, AC and BC are well defined, then computing $(A+B)C$ can be equivalently done by multiplying A and B by C separately and then summing the products up.

- *Associativity of scalar-matrix multiplication:*

$$(cA)(B) = A(cB) = c(AB).$$

- *Associativity of matrix multiplication:*

$$(AB)C = A(BC).$$

- *Distributivity of matrix multiplication:*

$$(A+B)C = AC + BC \quad \text{and} \quad C(A+B) = CA + CB.$$

- *Transposes of matrix product/sum:*

$$(AB)^T = B^T A^T \quad \text{and} \quad (A+B)^T = A^T + B^T.$$

- *Multiplication with identity matrices:* for any $m \times n$ matrix A, we have

$$\mathbb{I}_m A = A \mathbb{I}_n = A,$$

  where $\mathbb{I}_m$ and $\mathbb{I}_n$ are two identity matrices of sizes m and n respectively.

It is important to keep in mind that matrix multiplication is NOT commutative in general. Namely, in general,

$$AB \neq BA.$$

For example, if $A \in \mathbb{R}^{3\times 4}$ and $B \in \mathbb{R}^{4\times 5}$, then AB is a 3×5 matrix, but BA is not even well defined!

Example 2.10. If $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $C = \begin{bmatrix} 2 & 0 & 3 & 0 \end{bmatrix}$, then $AB = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ and $BC = \begin{bmatrix} 2 & 0 & 3 & 0 \\ -2 & 0 & -3 & 0 \end{bmatrix}$. Consequently,

$$(AB)C = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 3 & 0 \end{bmatrix} = \begin{bmatrix} -2 & 0 & -3 & 0 \\ -2 & 0 & -3 & 0 \end{bmatrix}$$

and

$$A(BC) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 2 & 0 & 3 & 0 \\ -2 & 0 & -3 & 0 \end{bmatrix} = \begin{bmatrix} -2 & 0 & -3 & 0 \\ -2 & 0 & -3 & 0 \end{bmatrix}.$$

2.2.4 Block matrices

A **block matrix** or a **partitioned matrix** is a partition of a matrix into rectangular smaller matrices called blocks. The matrix is split up by horizontal and vertical lines that go all the way across.

Example 2.11. The matrix

$$P = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix}$$

can be partitioned into 4 blocks

$$P = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

with

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 7 & 8 \end{bmatrix}, B = \begin{bmatrix} 4 & 5 \\ 9 & 10 \end{bmatrix}, C = \begin{bmatrix} 11 & 12 & 13 \end{bmatrix}, \text{ and } D = \begin{bmatrix} 14 & 15 \end{bmatrix}.$$

If A and B are two matrices of the same dimension, and they are partitioned in the same way, then $A+B$ can be computed block by block. For example, if

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \text{ and } B = \begin{bmatrix} B_1 & B_2 \end{bmatrix},$$

and the dimensions of A_1 (respectively, A_2) are the same as those of B_1 (respectively, B_2), then

$$A+B=\begin{bmatrix}A_1+B_1 & A_2+B_2\end{bmatrix}.$$

If M and N are two matrices of dimensions $m\times n$ and $n\times k$ respectively, and the columns of M are partitioned in the same way as rows of N, then we can compute MN using block-by-block multiplication. More precisely, if we have

$$M=\begin{bmatrix}A_1 & B_1\\ C_1 & D_1\end{bmatrix}\ \text{and}\ N=\begin{bmatrix}A_2 & B_2\\ C_2 & D_2\end{bmatrix},$$

then the equality

$$MN=\begin{bmatrix}A_1A_2+B_1C_2 & A_1B_2+B_1D_2\\ C_1A_2+D_1C_2 & C_1B_2+D_1D_2\end{bmatrix}$$

holds as long as all the product matrices, such as A_1A_2, are well defined. In other words, the number of columns in A_1 needs to equal the number of rows in A_2, and the number of columns in B_1 needs to equal the number of rows in C_2.

Example 2.12. Given two matrices

$$A=\left[\begin{array}{cc|cc}1 & 3 & 2 & -1\\ 3 & 2 & 1 & 0\\ -2 & 4 & 0 & 5\\ \hline -1 & -2 & 3 & 1\\ 5 & 4 & 2 & -1\end{array}\right]_{5\times 4}\quad \text{and}\quad B=\left[\begin{array}{cc|cc}2 & 1 & 3 & 4\\ -1 & 0 & 2 & 1\\ \hline 1 & -1 & 0 & 3\\ 4 & 5 & -2 & -3\end{array}\right]_{4\times 4},$$

then, the product of A and B can be computed as

$$AB=\begin{bmatrix}\begin{bmatrix}1 & 3\\ 3 & 2\\ -2 & 4\end{bmatrix}\begin{bmatrix}2 & 1\\ -1 & 0\end{bmatrix}+\begin{bmatrix}2 & -1\\ 1 & 0\\ 0 & 5\end{bmatrix}\begin{bmatrix}1 & -1\\ 4 & 5\end{bmatrix} & \begin{bmatrix}1 & 3\\ 3 & 2\\ -2 & 4\end{bmatrix}\begin{bmatrix}3 & 4\\ 2 & 1\end{bmatrix}+\begin{bmatrix}2 & -1\\ 1 & 0\\ 0 & 5\end{bmatrix}\begin{bmatrix}0 & 3\\ -2 & -3\end{bmatrix}\\ \begin{bmatrix}-1 & -2\\ 5 & 4\end{bmatrix}\begin{bmatrix}2 & 1\\ -1 & 0\end{bmatrix}+\begin{bmatrix}3 & 1\\ 2 & -1\end{bmatrix}\begin{bmatrix}1 & -1\\ 4 & 5\end{bmatrix} & \begin{bmatrix}-1 & -2\\ 5 & 4\end{bmatrix}\begin{bmatrix}3 & 4\\ 2 & 1\end{bmatrix}+\begin{bmatrix}3 & 1\\ 2 & -1\end{bmatrix}\begin{bmatrix}0 & 3\\ -2 & -3\end{bmatrix}\end{bmatrix}$$

$$=\left[\begin{array}{cc|cc}-3 & -6 & 11 & 16\\ 5 & 2 & 13 & 17\\ 12 & 23 & -8 & -19\\ \hline 7 & 1 & -9 & 0\\ 4 & -2 & 25 & 33\end{array}\right].$$

For an $m \times n$ matrix A, recall that A_j denotes its j-th column, and a_i denotes the transpose of the i-th row of A. The matrix A can be partitioned by columns or by rows:

$$A = \begin{bmatrix} A_1 \; A_2 \; \cdots \; A_n \end{bmatrix} = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{bmatrix}.$$

The product Ax of a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$ can be written as

$$Ax = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}$$

by partitioning A by rows, or as

$$Ax = \sum_{i=1}^{n} x_i A_i$$

by partitioning A by columns.

2.2.5 Invertible matrices, dependence and independence

For each **square** matrix $A \in \mathbb{R}^{m \times m}$, one can compute its **determinant**, denoted by $\det(A)$, which is a real number. For a matrix $A \in \mathbb{R}^{1 \times 1}$, $\det(A)$ is simply defined as $\det A = a_{11}$. For a matrix $A \in \mathbb{R}^{2 \times 2}$, the determinant is given by the formula

$$\det(A) = a_{11}a_{22} - a_{12}a_{21}.$$

For a matrix $A \in \mathbb{R}^{3 \times 3}$, we can compute it by

$$\begin{aligned}\det(A) &= a_{11} \det\left(\begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}\right) - a_{12} \det\left(\begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix}\right) + a_{13} \det\left(\begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{332} \end{bmatrix}\right) \\ &= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{21}a_{13}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{33}a_{21} - a_{11}a_{23}a_{32}.\end{aligned}$$

For higher dimensions, the computation of $\det A$ is much more complicated, and we shall not discuss it in this course. It is enough to know that $\det(\mathbb{I}_n) = 1$ for any identity matrix $\mathbb{I}_n$, and that $\det(AB) = \det(A)\det(B)$ for any square matrices A and B of the same dimension. Students can read, e.g., [21] for further details.

- If a matrix A satisfies $\det A = 0$, then it is called a *singular matrix*.
- If a matrix A satisfies $\det A \neq 0$, then it is called a *nonsingular matrix* or *an invertible matrix*.

Example 2.13. For example, given $A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{bmatrix}$, then

$$\det(A) = 1\times1\times1+2\times3\times3+2\times2\times(-1)-3\times1\times(-1)-2\times2\times1-2\times3\times1 = 1+18-4+3-4-6 = 8.$$

Hence, A is a nonsingular matrix, or A is invertible.

Given a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, there exists a unique $n\times n$ matrix, denoted by A^{-1}, that satisfies

$$A(A^{-1}) = \mathbb{I}_n.$$

Here $\mathbb{I}_n$ is the $n \times n$ identity matrix. The matrix A^{-1} is called the inverse of A, which also satisfies $(A^{-1})A = \mathbb{I}_n$.

For instance, the inverse of $A = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}$ is $A^{-1} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} \end{bmatrix}$. To check this, first calculate $\det(A) = -3$ to find A to be nonsingular, and then compute $AA^{-1} = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. This suffices to verify that the given matrix is the inverse of A. One can further check that $A^{-1}A = \mathbb{I}_2$ as well.

While it is possible for two non-square matrices A and B to satisfy $AB = \mathbb{I}$ or $BA = \mathbb{I}$, we do not call them inverses of each other since they are non-square.

2.2.5.1 Facts about invertible matrices

- If A is invertible, then A^T is also invertible, with

$$(A^T)^{-1} = (A^{-1})^T.$$

- If both A and B are invertible, and they have the same dimension, then AB is invertible, with

$$(AB)^{-1} = B^{-1}A^{-1}.$$

2.2.5.2 Linear combinations

Let $v^{(1)}, v^{(2)}, \cdots, v^{(k)}$ be k column vectors of the same dimension. That is, each $v^{(i)} \in \mathbb{R}^m$ for $i = 1, \cdots, k$.

- A **linear combination** of these k vectors is any vector $u \in \mathbb{R}^m$ that can be written as

$$u = c_1 v^{(1)} + c_2 v^{(2)} + \cdots c_k v^{(k)}$$

  for some real numbers $c_1, c_2, \cdots, c_k$. The numbers $c_1, \cdots, c_k$ are called coefficients of the linear combination.
- The vectors $v^{(1)}, v^{(2)}, \cdots, v^{(k)}$ are **linearly dependent** if there exist k coefficients $c_1, c_2, \cdots, c_k$, not all being zeros, with

$$c_1 v^{(1)} + c_2 v^{(2)} + \cdots c_k v^{(k)} = 0.$$

  Here $0 = (0, 0, \cdots, 0)^T$ is the zero vector in $\mathbb{R}^m$.
- The vectors $v^{(1)}, v^{(2)}, \cdots, v^{(k)}$ are **linearly independent** if they are not linearly dependent. In other words, it is linearly independent if the following equality

$$c_1 v^{(1)} + c_2 v^{(2)} + \cdots c_k v^{(k)} = 0$$

  holds if and only if $c_1 = c_2 = \cdots = c_k = 0$.

The above definitions apply to row vectors as well.

Example 2.14. Consider the following 4 vectors in $\mathbb{R}^3$:

$$v^{(1)} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \ v^{(2)} = \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}, \ v^{(3)} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \text{ and } v^{(4)} = \begin{pmatrix} -1 \\ 3 \\ 1 \end{pmatrix}.$$

- The vectors $v^{(1)}, v^{(2)}, v^{(3)}$ are linearly independent. To see this, consider a linear combination $c_1 v^{(1)} + c_2 v^{(2)} + c_3 v^{(3)} = 0$ and write this explicitly as

$$\begin{cases} c_1 & -2c_2 & +2c_3 & = 0 \\ 2c_1 & +c_2 & & = 0 \\ c_1 & & & = 0. \end{cases}$$

  The last equation shows that $c_1 = 0$. Substitute $c_1 = 0$ into the second equation, we have $c_2 = 0$. Substitute both $c_1 = 0$ and $c_2 = 0$ into the first equation, we get

$c_3 = 0$. This shows that $(c_1, c_2, c_3) = 0$ is the only coefficient vector that satisfies $c_1 v^{(1)} + c_2 v^{(2)} + c_3 v^{(3)} = 0$. Hence, $v^{(1)}, v^{(2)}, v^{(3)}$ is linearly independent.

- The vectors $v^{(1)}, v^{(2)}, v^{(4)}$ are linearly dependent. Indeed, we can easily observe that $v^{(1)} + v^{(2)} = v^{(4)}$. This is equivalent to $v^{(1)} + v^{(1)} - v^{(4)} = 0$. Hence, the coefficients $c_1 = 1$, $c_2 = 1$ and $c_4 = -1$ satisfy $c_1 v^{(1)} + c_2 v^{(2)} + c_4 v^{(4)} = 0$.

The following theorem gives the relation between independence of vectors and invertibility of matrices. The proof is omitted. Two statements are said to be equivalent if they imply each other.

Theorem 2.1. *Let A be a square matrix. The following statements are equivalent.*

1. *A is invertible.*
2. *The rows of A are linearly independent.*
3. *The columns of A are linearly independent.*

When A is not a square matrix, we have the following result. A **submatrix** of a matrix is obtained by deleting any collection of rows and/or columns.

Theorem 2.2. *Let $A \in \mathbb{R}^{m \times n}$ where $m \le n$. The following statements are equivalents.*

1. *The rows of A are linearly independent.*
2. *There exist m columns of A that are linearly independent.*
3. *A has a nonsingular $m \times m$ submatrix.*

Proofs of these two theorems can be found in standard linear algebra books, e.g., [21].

Example 2.15. Consider the following 4×7 matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

One can readily check that the rows of A are linearly independent. It follows from the above theorem that one must be able to find 4 columns of A that are linearly independent. Indeed, the first 4 columns are linearly independent. This does not mean any 4 columns of A are linearly independent; for example you cannot pick columns 1, 5, 6, and 7.

2.3 Systems of linear equations

2.3.1 Definitions and examples

Consider a system of linear equations

$$Ax = b,$$

where $A \in \mathbb{R}^{m\times n}$ is a given left-hand-side coefficient matrix and $b \in \mathbb{R}^m$ is a given right-hand-side vector. We can write this system in the explicit form as

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & +\cdots & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\cdots & a_{2n}x_n & = & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1}x_1 & +a_{m2}x_2 & +\cdots & a_{mn}x_n & = & b_m. \end{cases}$$

This system can also be represented in the augmented matrix form $[A \mid b]$. The *solution set* of this system is the set of all of its solutions, that is, the set of all vectors $x \in \mathbb{R}^n$ that satisfy $Ax = b$.

Depending on its solution set, a linear system may belong to one of the following three types:

1. *A linear system with infinitely many solutions:* This happens when b is a linear combination of columns of A, and columns of A are linearly dependent.
2. *A linear system with a unique solution:* This happens when b is a linear combination of columns of A, and columns of A are linearly independent.
3. *A linear system with no solution:* This happens when b is not a linear combination of columns of A.

In general, if $\bar{x}$ is a known solution to the system $Ax = b$, then the solution set of this system is

$$\{\bar{x} + d \mid Ad = 0\}.$$

If A is an invertible square matrix, then the system $Ax = b$ has a unique solution x for any $b \in \mathbb{R}^n$, and this solution is $x = A^{-1}b$.

We say that a system of linear equations $Ax = b$ is equivalent to a system of linear equations $Cx = d$ if any solution of $Ax = b$ is also a solution of $Cx = d$, and vice versa.

Example 2.16. Consider two systems of linear equations:

$$\begin{cases} x_1 + x_2 + x_3 &= 3 \\ x_1 + 2x_2 + 3x_3 &= 6, \end{cases} \quad \text{and} \quad \begin{cases} x_1 + 3x_2 + 5x_3 &= 9 \\ 2x_1 + 3x_2 + 4x_3 &= 9. \end{cases}$$

Let (x_1, x_2, x_3) be a solution of the first system. Then, $x_1 + x_2 + x_3 = 3$ and $x_1 + 2x_2 + 3x_3 = 6$. Summing up these two equations, we get $2x_1 + 3x_2 + 4x_3 = 9$, which is the second equation of the second system. Moreover, if we compute $2 \times (x_1 + 2x_2 + 3x_3 = 6) - (x_1 + x_2 + x_3 = 3)$, we obtain the first equation of the second system. Similarly, we can obtain the first system from the second system by combining its equations. Hence, these two systems are equivalent.

2.3.2 Elementary row operations

When we apply the following elementary row operations on a system of linear equations, they do NOT change the solution set of this system.

Type 1: Multiplying an equation by a nonzero number.

Type 2: Adding a multiple of one equation to another equation.

Type 3: Switching two equations.

More generally, consider a system of linear equations represented by

$$Ax = b,$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We can pre-multiply both sides of it by an invertible matrix $M \in \mathbb{R}^{m \times m}$, to obtain a new system $MAx = Mb$. Note that

$$Ax = b \quad \Leftrightarrow \quad MAx = Mb$$

so the two systems $Ax = b$ and $MAx = Mb$ have exactly the same solution set.

Pre-multiplying both sides of a system of linear equations by an invertible matrix does not change the solution set of this system.

In fact, conducting an elementary row operation on a system of equations is equivalent to pre-multiplying the system by a special matrix.

Example 2.17. Consider the following system of linear equations $Ax = b$:

$$\begin{cases} x_1 + x_2 + x_3 &= 3 \\ x_1 + 2x_2 + 3x_3 &= 6, \end{cases} \quad \text{or in the matrix form} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{pmatrix} 3 \\ 6 \end{pmatrix}.$$

Now, let $M = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ be an invertible matrix $\det(M) = -3 \neq 0$). Then, the system $MAx = Mb$ becomes

$$\begin{cases} 3x_1 + 5x_2 + 7x_3 &= 15 \\ 3x_1 + 4x_2 + 5x_3 &= 12. \end{cases}$$

In fact, the first equation of the latter system is the sum of the first equation and two times the second equation of the former system, and the second equation of the latter system is the sum of two times the first equation and the second equation of the former system.

2.3.3 *Gauss-Jordan's method for solving systems of linear equations*

Consider a system of m linear equations in the augmented-matrix form $[A \mid b]$. The Gauss-Jordan (GJ) method proceeds as follows.

1. Initialization: set the counter $k = 1$.
2. Find the leftmost column in A that is not all zeros from rows k to m. If no such a column exists, stop the procedure right away.

   The following illustrates this step for the case in which $k = 1$.

$$\left[\begin{array}{cccccc|c} 0 & \cdots & 0 & * & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & * & \cdots & * & * \end{array}\right]$$
$$\Uparrow$$

3. Use elementary row operations, we transform this column to the unit vector $\mathbf{e}_k$ (i.e., a column with 1 as the kth entry, and 0 elsewhere). In order to do so, we carry out the following steps:

a. If the kth entry of this column is zero, switch the kth row with another row so that the kth entry of this column becomes $a \neq 0$.

b. Multiply the kth row by $1/a$.

c. Add multiples of the kth row to other rows so that all other entries of this column become zeros.

The following illustrates these steps for the case in which $k = 1$.

$$\left[\begin{array}{cccccc|c} 0 & \cdots & 0 & 0 & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & * & \cdots & * & * \end{array}\right] \xrightarrow{\text{After step (a)}} \left[\begin{array}{cccccc|c} 0 & \cdots & 0 & a & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & * & \cdots & * & * \end{array}\right] \xrightarrow{\text{After step (c)}} \left[\begin{array}{cccccc|c} 0 & \cdots & 0 & 1 & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & * & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & * & * \end{array}\right].$$

4. If $k = m$, stop the procedure. Otherwise, increase k by 1, and go back to Step 2.

At the end of the Gauss-Jordan method, we obtain a system of linear equations in the so-called “reduced row-echelon form”, from which we can read off its solution set.

Example 2.18. The linear system

$$\begin{cases} 2x_1 + 2x_2 + x_3 = 9 \\ 2x_1 - x_2 + 2x_3 = 6 \\ x_1 - x_2 + 2x_3 = 5 \end{cases}$$

can be written into the matrix form $Ax = b$ as:

$$\begin{bmatrix} 2 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & -1 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 9 \\ 6 \\ 5 \end{pmatrix}.$$

Hence, its augmented-matrix form becomes

$$[A \mid b] = \left[\begin{array}{ccc|c} 2 & 2 & 1 & 9 \\ 2 & -1 & 2 & 6 \\ 1 & -1 & 2 & 5 \end{array}\right]$$

The Gauss-Jordan method: We apply Gauss-Jordan's method to solve this system, which consists of the following steps:

Step 1: Multiply row 1 by $1/2$ to get

$$\left[\begin{array}{ccc|c} 1 & 1 & 1/2 & 9/2 \\ 2 & -1 & 2 & 6 \\ 1 & -1 & 2 & 5 \end{array}\right].$$

Step 2: Add -2(row 1) to row 2 to get

$$\left[\begin{array}{ccc|c} 1 & 1 & 1/2 & 9/2 \\ 0 & -3 & 1 & -3 \\ 1 & -1 & 2 & 5 \end{array}\right].$$

Step 3: Add -1(row 1) to row 3 to get

$$\left[\begin{array}{ccc|c} 1 & 1 & 1/2 & 9/2 \\ 0 & -3 & 1 & -3 \\ 0 & -2 & 3/2 & 1/2 \end{array}\right].$$

Step 4: Multiply row 2 by $-1/3$ to get

$$\left[\begin{array}{ccc|c} 1 & 1 & 1/2 & 9/2 \\ 0 & 1 & -1/3 & 1 \\ 0 & -2 & 3/2 & 1/2 \end{array}\right].$$

Step 5: Add -1(row 2) to row 1 to get

$$\left[\begin{array}{ccc|c} 1 & 0 & 5/6 & 7/2 \\ 0 & 1 & -1/3 & 1 \\ 0 & -2 & 3/2 & 1/2 \end{array}\right].$$

Step 6: Add 2(row 2) to row 3 to get

$$\left[\begin{array}{ccc|c} 1 & 0 & 5/6 & 7/2 \\ 0 & 1 & -1/3 & 1 \\ 0 & 0 & 5/6 & 5/2 \end{array}\right].$$

Step 7: Multiply row 3 by $6/5$ to get

$$\left[\begin{array}{ccc|c} 1 & 0 & 5/6 & 7/2 \\ 0 & 1 & -1/3 & 1 \\ 0 & 0 & 1 & 3 \end{array}\right].$$

Step 8: Add -5/6(row 3) to row 1 to get

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & -1/3 & 1 \\ 0 & 0 & 1 & 3 \end{array}\right].$$

Step 9: Add 1/3(row 3) to row 2 to get

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right].$$

Since each operation does not change the solution set of the system, the last system has the same solution set as the original system. Here, the last system has a unique solution $x = (1,2,3)^T$, which is also the unique solution of the original system.

Example 2.19. Consider the following simple system:

$$\begin{cases} x_1 + 2x_2 &= 3 \\ 2x_1 + 4x_2 &= 4. \end{cases}$$

If we apply Gauss-Jordan's method to this system, then we have

$$\left[\begin{array}{cc|c} 1 & 2 & 3 \\ 2 & 4 & 4 \end{array}\right] \quad \Rightarrow \quad \left[\begin{array}{cc|c} 1 & 2 & 3 \\ 0 & 0 & -2 \end{array}\right].$$

The second row in the last system says $0x_1 + 0x_2 = -2$, which can never be true. Therefore, this system has no solution.

Example 2.20. Consider the linear system:

$$\begin{cases} x_1 + x_2 &= 1 \\ \quad\;\; x_2 + x_3 &= 3 \\ x_1 + 2x_2 + x_3 &= 4. \end{cases}$$

If we apply Gauss-Jordan's method to this system, then we have

$$\left[\begin{array}{ccc|c} 1&1&0&1\\ 0&1&1&3\\ 1&2&1&4 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1&1&0&1\\ 0&1&1&3\\ 0&1&1&3 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1&0&-1&-2\\ 0&1&1&3\\ 0&1&1&3 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1&0&-1&-2\\ 0&1&1&3\\ 0&0&0&0 \end{array}\right].$$

We can rewrite the last system as

$$\begin{cases} x_1 - x_3 = -2 \\ x_2 + x_3 = 3. \end{cases}$$

This system has infinitely many solutions: for any arbitrary value of $x_3 \in \mathbb{R}$, $x_1 = -2 + x_3$ and $x_2 = 3 - x_3$. Its solution set can be represented as

$$\{x = (-2+t, 3-t, t) \mid t \in \mathbb{R}\}.$$

Example 2.21. Solve the following system of equations with three variables and three equations:

$$\begin{cases} x_1 + x_2 + 2x_3 = 9 \\ x_1 + x_2 + x_3 = 6 \\ \quad - x_2 + 2x_3 = 5. \end{cases}$$

First, we write this system in the matrix form $Ax = b$ with

$$[A \mid b] = \left[\begin{array}{ccc|c} 1&1&2&9\\ 1&1&1&6\\ 0&-1&2&5 \end{array}\right].$$

Then, we apply the GJ method to transform this matrix into the row-echelon form as:

$$[A \mid b] = \left[\begin{array}{ccc|c} 1&1&2&9\\ 1&1&1&6\\ 0&-1&2&5 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1&1&2&9\\ 0&0&-1&-3\\ 0&-1&2&5 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1&1&2&9\\ 0&-1&2&5\\ 0&0&-1&-3 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1&1&2&9\\ 0&1&-2&-5\\ 0&0&-1&-3 \end{array}\right]$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1&0&4&14\\ 0&1&-2&-5\\ 0&0&-1&-3 \end{array}\right] \Rightarrow \cdots \Rightarrow \left[\begin{array}{ccc|c} 1&0&0&2\\ 0&1&0&1\\ 0&0&1&3 \end{array}\right].$$

We finally obtain the solution of this system as $x = (2, 1, 3)^T$.

2.3.3.1 Back substitution

There is an alternative way to apply Gauss-Jordan's method. Instead of transforming $[A \mid b]$ until A contains an identity submatrix, we can apply Gauss-Jordan's method to transform $[A \mid b]$ until A becomes an upper trapezoidal matrix (a matrix is an upper trapezoidal matrix if nonzero elements are found only in the upper triangle of the matrix, including the main diagonal). Following that, we apply the **back substitution method** to solve the resulting system.

Below we briefly describe this method for the special case in which the number of variables equals the number of equations in the system. This method can readily be applied to general systems.

Given a system of linear equations $Ax = b$ with $A \in \mathbb{R}^{n\times n}$, we perform two steps:

1. Apply the GJ method to $[A \mid b]$ to transform it into a row echelon form $[\tilde{A} \mid \tilde{b}]$ (or upper triangular):
$$\left[\begin{array}{cccc|c} \tilde{a}_{11} & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} & \tilde{b}_1 \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} & \tilde{b}_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \tilde{a}_{nn} & \tilde{b}_n \end{array}\right]$$
2. Then apply the back substitution method to solve the linear system $\tilde{A}x = \tilde{b}$. If all diagonal elements of $\tilde{A}$ are nonzero, this method performs backward steps from the bottom to the top as follows:
   - Compute x_n from $\tilde{a}_{nn}x_n = \tilde{b}_n$ as $x_n = \frac{\tilde{b}_n}{\tilde{a}_{nn}}$.
   - Once $x_{i+1}, \cdots, x_n$ are computed, we can compute x_i recursively as $x_i = \frac{\tilde{b}_i - \sum_{j=i+1}^{n} \tilde{a}_{ij}x_j}{\tilde{a}_{ii}}$ for $i = n-1$ down to $i = 1$.

   When some diagonal elements are zero, the system either has no solution or infinitely many solutions. In the latter case, the solution set can be written by expressing some variables in terms of other variables.

Example 2.22. To illustrate this method, we reconsider Example 2.21 by solving
$$\begin{cases} x_1 + x_2 + 2x_3 = 9 \\ x_1 + x_2 + \ x_3 = 6 \\ \quad\ - x_2 + 2x_3 = 5. \end{cases}$$

We have $[A \mid b] = \left[\begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 1 & 1 & 1 & 6 \\ 0 & -1 & 2 & 5 \end{array}\right]$.

Step 1: After applying the GJ method we can transform this matrix into an upper triangular matrix:

$$[\tilde{A} \mid \tilde{b}] = \left[\begin{array}{ccc|c} 1 & 1 & 2 & 9 \\ 0 & 1 & -2 & -5 \\ 0 & 0 & -1 & -3 \end{array}\right].$$

Hence, we can write this system as

$$\begin{cases} x_1 + x_2 + 2x_3 = 9 \\ \quad\ \ x_2 - 2x_3 = -5 \\ \qquad\quad -x_3 = -3. \end{cases}$$

Step 2: Now, we apply the back substitution step as

– From the last equation, we can compute $x_3 = \frac{-3}{-1} = 3$.

– From the second equation, we have $x_2 = -5 + 2x_3 = -5 + 2 \times 3 = 1$.

– From the first equation, we have $x_1 = 9 - x_2 - 2x_3 = 9 - 1 - 2 \times 3 = 2$.

Hence, the solution of the original linear system is $x = (2, 1, 3)^T$.

2.3.4 Gauss-Jordan's method for computing the inverse of a nonsingular matrix

We can use the Gauss-Jordan method to compute the inverse of an $n \times n$ nonsingular matrix A. This method works as follows:

- First, we write the matrix A and the identity matrix $\mathbb{I}$ side by side, as $[A \mid \mathbb{I}]$.
- Second, we use elementary row operations to transform the matrix A into the row-echelon form (or upper triangular), so that its left-bottom corner below the diagonal is all zero.
- Third, we further use elementary row operations to transform the matrix A into a diagonal matrix.
- Fourth, we normalize the diagonal elements of the diagonal matrix, to obtain a matrix of the form $[\mathbb{I} \mid B]$, where B is exactly the inverse of A.

Example 2.23 (Inverse matrix). Using the Gauss-Jordan method to compute the inverse of $A = \begin{bmatrix} 1 & 2 & -4 \\ 2 & 1 & -1 \\ 3 & 0 & 1 \end{bmatrix}$.

Our method performs as follows:

- First, we write

$$[A \mid \mathbb{I}] = \left[\begin{array}{ccc|ccc} 1 & 2 & -4 & 1 & 0 & 0 \\ 2 & 1 & -1 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

- Next, we transform this matrix into the "row-echelon form". The following steps are performed:

  1. Keep row 1 unchanged.
  2. Replace row 2 by row 2 subtracting two times of row 1 from row 2. That is, $R_2 \leftarrow R_2 - 2 \times R_1$.
  3. Replace row 3 by row 3 subtracting three times of row 1 from row 3. That is, $R_3 \leftarrow R_3 - 3 \times R_1$.
  4. Then, replace row 3 by row 3 subtracting two times of row 2 from row 3. That is, $R_3 \leftarrow R_3 - 2 \times R_2$.

$$\left[\begin{array}{ccc|ccc} 1 & 2 & -4 & 1 & 0 & 0 \\ 0 & -3 & 7 & -2 & 1 & 0 \\ 0 & 0 & -1 & 1 & -2 & 1 \end{array}\right].$$

- The next step is to transform this matrix to a "row-diagonal form".

  1. Keep the last row unchanged.
  2. Replace row 2 by row 2 adding to row 3 after multiplying it by 7. That is, $R_2 \leftarrow R_2 + 7 \times R_3$.
  3. Replace row 1 by row 1 subtracting four times of row 3 from row 1. That is, $R_1 \leftarrow R_1 - 4 \times R_3$.
  4. Then, replace row 1 by row 1 adding to row 2 after multiplying it by $\frac{2}{3}$. That is, $R_1 \leftarrow R_1 + \frac{2}{3} \times R_2$.

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{1}{3} & \frac{-2}{3} & \frac{2}{3} \\ 0 & -3 & 0 & 5 & -13 & 7 \\ 0 & 0 & -1 & 1 & -2 & 1 \end{array}\right].$$

- Finally, we normalize this matrix to get the identity matrix on the left by dividing the second row by -3, and the last row by -1.

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{1}{3} & \frac{-2}{3} & \frac{2}{3} \\ 0 & 1 & 0 & \frac{-5}{3} & \frac{13}{3} & \frac{-7}{3} \\ 0 & 0 & 1 & -1 & 2 & -1 \end{array}\right] = [\mathbb{I} \mid B].$$

- This matrix is of the form $[\mathbb{I} \mid B]$, we conclude that $B = A^{-1} = \begin{bmatrix} \frac{1}{3} & \frac{-2}{3} & \frac{2}{3} \\ \frac{-5}{3} & \frac{13}{3} & \frac{-7}{3} \\ -1 & 2 & -1 \end{bmatrix}$.

2.3.5 *Checking linearly independence of vectors*

Given n column vectors $\{A_1, A_2, \cdots, A_n\}$ in $\mathbb{R}^m$, we can use Gauss-Jordan method to check if they are linearly independent. If we let

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

be the $m \times n$ matrix formed by these vectors, then the equation

$$x_1A_1 + x_2A_2 + \cdots x_nA_n = 0$$

can be equivalently written as

$$Ax = 0,$$

where $x \in \mathbb{R}^n$ is the column vector with entries given by the coefficients $x_1, \cdots, x_n$. If the above equation has nonzero solutions, then the vectors $A_1, \cdots, A_n$ are linearly dependent. Otherwise, if the only solution to the above system is zero, then these vectors are linearly independent.

2.4 Exercises

Exercise 2.1. Given two matrices $A = \begin{bmatrix} 4 & 3 & 2 \\ 2 & 3 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -2 \end{bmatrix}$. Answer the following questions:

1. Compute the product AB using the definition of matrix multiplication.
2. Partition A into 2 blocks by a vertical line between its 2nd and 3rd columns, and partition B into 2 blocks by a horizontal line between its 2nd and 3rd rows. Compute AB using block operations.

Exercise 2.2. Use the Gauss-Jordan method to solve the following linear systems. Show each step of your calculations.

1.
$$\begin{cases} \quad\;\; x_2 + 2x_3 = 3 \\ x_1 + 2x_2 + x_3 = 4 \\ x_1 + x_2 - 2x_3 = 0. \end{cases}$$

2.
$$\begin{cases} x_1 - 4x_2 + 2x_3 = -4 \\ \quad\; 2x_2 - x_3 = 1 \\ -x_1 + 2x_2 - x_3 = 3 \\ -2x_1 + 6x_2 - 3x_3 = 7. \end{cases}$$

3.
$$\begin{cases} x_1 \qquad\quad + x_4 = 5 \\ \quad x_2 \qquad + 2x_4 = 5 \\ \qquad x_3 + 0.5x_4 = 1 \\ \qquad 2x_3 + x_4 = 3. \end{cases}$$

Exercise 2.3. Given the following four linear equation systems of the form $Ax = b$, where their augmented matrix $\bar{A} = [A \mid b]$ are respectively given by

$$\left[\begin{array}{cccc|c} 1&0&0&2&4 \\ 0&1&0&0&-2 \\ 0&0&1&0&2 \end{array}\right], \quad \left[\begin{array}{cccc|c} 1&0&1&9&3 \\ 0&1&-1&3&2 \\ 0&0&0&0&0 \end{array}\right], \quad \left[\begin{array}{cccc|c} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&1&0 \end{array}\right], \quad \left[\begin{array}{cccc|c} 1&0&1&9&0 \\ 0&-1&1&-3&1 \\ 1&-1&2&6&-2 \end{array}\right].$$

For each system, determine if it has a unique solution, multiple solutions, or no solution. Write down the solution set of each system when it exists.

Exercise 2.4. Use the Gauss-Jordan method to compute the inverse matrix of the following matrices:

$$(a)\ A = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \qquad (b)\ A = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ 2 & 3 & -2 \end{bmatrix}.$$

Show the steps of your Gauss-Jordan transformation. Verify the result by computing AA^{-1}.

To show a square matrix A is not invertible (that is A^{-1} does not exist), we can also use the Gauss-Jordan method. First, we apply the Gauss-Jordan method to transform $[A \mid \mathbb{I}]$ into a trapezoidal form $[\tilde{A} \mid \tilde{B}]$, where $\tilde{A}$ is a lower triangular matrix. Then, if there exists a zero entry on the diagonal of $\tilde{A}$, then we can conclude that A is not invertible.

Now, use this theory and the Gauss-Jordan method to show that $A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & -2 \\ 5 & 2 & -3 \end{bmatrix}$ is not invertible.

Exercise 2.5. Determine if each of the following statements is true or false. Justify your answer by a proof or a counter example. Proofs should be based on the material covered in this course.

1. The equality $x^T y = y^T x$ holds for any two n-dimensional column vectors x and y.
2. If A, B and C are invertible matrices of the same dimension, then the product of their transposes, $A^T B^T C^T$, is invertible.
3. If A, B and C are invertible matrices of the same dimension, then their sum $A+B+C$ is invertible.
4. Suppose that A is an invertible matrix. Switch its top two rows to obtain a new matrix B. The matrix B must be invertible as well.
5. Let $v^1, v^2, \cdots, v^k$ be column vectors of the same dimension, and let $v = v^1 + v^2 + \cdots + v^k$. The vectors $v^1, v^2, \cdots, v^k, v$ are linearly dependent.
6. The system $Ax = b$ has a solution, if b is a linear combination of columns of A.
7. For a system of linear equations that contains 2 equations and 3 variables (i.e., $A \in \mathbb{R}^{2\times 3}$), there are always infinitely many solutions.

Exercise 2.6. 1. Given k column vectors $v^i \in \mathbb{R}^3$ for $i = 1, \cdots k$ as follows:

$$v^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad v^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad v^3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad v^4 = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}, \quad v^5 = \begin{pmatrix} 1 \\ 1 \\ -3 \end{pmatrix}.$$

Find all the possible groups of 3 vectors such that each group is linearly independent. Is the group $\{v^1, v^2, v^3, v^4\}$ linearly independent? If it is not, please explain.

2. Given a matrix A

$$A = \begin{bmatrix} -1 & 2 & 3 & 4 \\ 5 & 0 & -1 & -1 \\ 8 & -6 & -10 & -13 \end{bmatrix}.$$

   a. Show that the last row of A is a linear combination of the first two rows.
   b. Let B be a 3×3 matrix formed by three columns of A. Is B invertible? Does your answer depend on which three columns of A are included in B?

Exercise 2.7. Given two vector $x, y \in \mathbb{R}^n$, we say that x and y are orthogonal if $x^T y = 0$. If, in addition, $\|x\|_2 = \|y\|_2 = 1$, then we say that x and y are orthonormal. Find an orthonormal vector b for each of the following vectors a:

$$a = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)^T, \quad \text{and } a = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{6}}\right)^T.$$

Find a vector c such that it is simultaneously orthonormal with two vectors $a = \frac{1}{\sqrt{14}}(1,2,3)^T$ and $b = \frac{1}{\sqrt{5}}(2,-1,0)^T$. Is c unique?

Exercise 2.8. An $m \times n$ matrix A is called column orthogonal if each column of A is orthogonal to other columns in A. If, in addition, the length of each column A_j is unit (i.e., $\|A_j\|_2 = 1$), then A is called a column orthonormal matrix. Given an orthogonal matrix A, we can always orthonormalize it by dividing each column by its norm. Check if the following matrices are column orthogonal/orthonormal.

$$A = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad A = \frac{1}{4}\begin{bmatrix} 1 & 1 & -1 \\ 2 & -2 & 0 \\ -2 & 1 & 0 \end{bmatrix} \text{ and } A = \begin{bmatrix} -\frac{1}{2} & \frac{1}{\sqrt{2}} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{\sqrt{2}} & -\frac{1}{2} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

If A is orthonormal, compute A^{-1} and compare it to A^T.

Chapter 3
Foundations of Linear Programming

3.1 Graphical methods for solving LPs with two variables

In practice, we use computer programs to implement some optimization algorithms to solve linear programs to find their optimal solutions, or to find them to be infeasible or unbounded. We will discuss some common algorithms in detail later. Any linear program with only two decision variables can be graphically solved.

Let us illustrate this method with the following example:

$$\begin{cases} \max\limits_{x} & z = 2.25x_1 + 2.60x_2 & \\ \text{subject to} & 2x_1 + x_2 & \leq 4000, \\ & x_1 + 2x_2 & \leq 5000, \\ & x_1 & \geq \quad 0, \\ & x_2 & \geq \quad 0. \end{cases}$$

With only two decision variables, it can be graphically solved by the following procedure.

1. **Plot the feasible set:** We first plot the line $2x_1 + x_2 = 4000$ by connecting the two intercepts $(2000, 0)$ and $(0, 4000)$. This line divides the whole plane into two sides, and we need to decide which side satisfies $2x_1 + x_2 \leq 4000$. It is often convenient to use the origin to check: since $(0, 0)$ satisfies the constraint, points on the same side of the line as the origin all satisfy the constraint. After handling other constraints in a similar manner, we take the intersection to obtain the feasible set, see Figure 3.1. In this example, the quadrilateral with four vertices $(0, 0)$, $(2000, 0)$, $(1000, 2000)$, and $(0, 2500)$ is the feasible set.

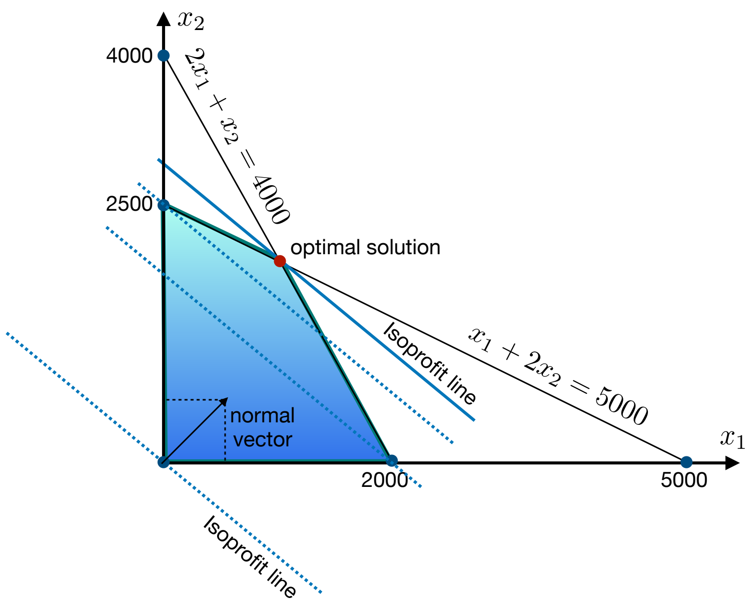


Fig. 3.1 Solving an LP graphically.

2. **Plot the objective line:** Plot lines of constant objective values (these lines are called isoprofit lines for maximization problems as the present example, and isocost lines in minimization problems). For this example, the isoprofit lines are all of the form

$$2.25x_1 + 2.60x_2 = P$$

for a given constant P. Note that isoprofit lines are all parallel. The vector of coefficients of the objective function is called the normal vector. In this example, $\vec{n} = (2.25, 2.6)$ is the normal vector. We can therefore choose an arbitrary value for P to plot the first isoprofit line, and then shift it along the normal vector $\vec{n} = (2.25, 2.6)$ to generate other isoprofit lines.

3. **Shift the isoprofit line:** Push the isoprofit line in the direction of the normal vector $\vec{n}$ to increase the objective value if we are solving a max problem. For a min problem, we need to push the isocost line in the opposite direction of $\vec{n}$ to decrease the objective value. We stop pushing when we would leave the feasible set. The point(s) that lie(s) on the intersection of the last isoprofit line and the feasible set are the optimal solution(s). In the example, the last isoprofit line meets the feasible set at the point where the two lines $2x_1 + x_2 = 4000$ and $x_1 + 2x_2 = 5000$ intersect.

4. **Determine optimal solutions:** Compute the optimal solution and the optimal value. For the example, we can solve the two equations $2x_1 + x_2 = 4000$ and $x_1 + 2x_2 = 5000$, to find the optimal solution to be $(1000, 2000)$. By plugging these values into the objective function, we find the optimal value to be 7450.

3.2 Types of linear programs

Depending on properties of their feasible sets and optimal solutions, linear programs can be classified into the following four types. We will discuss how to identify the type of any linear program in the subsequent sections.

1. **Infeasible linear programs.** These are LPs with no feasible solutions, or equivalently, LPs with empty feasible sets. For instance, consider the following problem:

$$\begin{cases} \min\limits_{x} & x_1 + x_2 \\ \text{subject to} & x_1 + x_2 \leq -1, \\ & x_1 \geq 0,\ x_2 \geq 0. \end{cases}$$

   This LP is infeasible because its feasible set $\mathscr{X} = \left\{x \in \mathbb{R}^2 \mid x_1 + x_2 \leq -1,\ x_1 \geq 0,\ x_2 \geq 0\right\}$ is empty.

2. **Linear programs with unique optimal solutions.** These are linear programs that have unique optimal solutions. The example in Figure 3.1 is of this type, because the last isoprofit line meets its feasible set at a single point, which is its only optimal solution.
3. **Linear programs with multiple optimal solutions.** These are linear programs that have more than one optimal solution. Whenever an LP has two different optimal solutions, it has infinitely many optimal solutions. Why? (Exercise.)
4. **Unbounded linear programs.** An LP is said to be unbounded, if its objective value can be made arbitrarily *good* by selecting appropriate feasible solutions. In other words, a maximization LP is unbounded if for any given number M we can find a feasible solution with its objective value bigger than M. When this is the case, we say that the optimal value of this problem is $+\infty$. Conversely, a minimization LP is unbounded if for any given number M we can find a feasible solution with its objective value less than M. In this case, we say that the optimal value of this problem is $-\infty$.

We should distinguish the difference between the unboundedness of an LP and the unboundedness of its feasible set. In general, a set in $\mathbb{R}^n$ is said to be bounded, if there exists a positive number M such that the norm of all elements in this set is no more than M. Hence, the feasible set of an LP is unbounded, if for any positive number M there exists a feasible solution whose norm is bigger than M.

Note that, if an LP is unbounded, then its feasible set must be unbounded. On the other hand, it is possible for an LP with an unbounded feasible set to have either unique or multiple optimal solutions. This is illustrated in the following example.

Example 3.1. In this example, x_1 and x_2 are variables, and c_1 and c_2 are constants.

$$\begin{cases} \min\limits_{x} \ c_1x_1 + c_2x_2 \\ \text{subject to } -x_1 + x_2 \le 1, \\ \qquad x_1 \ge 0, \\ \qquad x_2 \ge 0. \end{cases} \tag{3.1}$$

Different values of c_1 and c_2 lead to different linear programs with a common feasible set shown in Figure 3.2. We consider three cases, with different values of

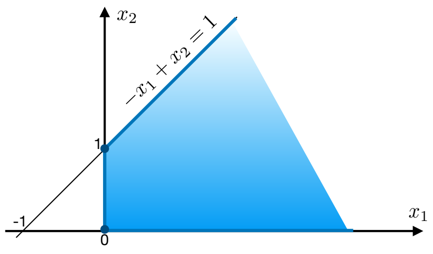


Fig. 3.2 The feasible set in the problem (3.1).

c_1 and c_2.

Case 1: Let us choose $c_1 = 1$ and $c_2 = 1$. Then $c = (1,1)$ is the normal vector of the isocost line. The isocost lines in this case are dashed lines in Figure 3.3. The last isocost line meets the feasible set at the unique optimal solution $x = (0,0)$. The optimal value is 0. The LP has a unique optimal solution.

Case 2: Let us choose $c_1 = 1, c_2 = 0$. The isocost lines in this case are dashed lines in Figure 3.4. The last isocost line that meets the feasible set intersects with it at

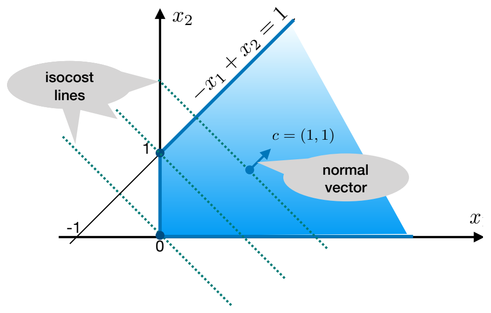


Fig. 3.3 **Case 1** of the LP (3.1) with $c = (1,1)$.

its left edge. Each point on that edge is an optimal solution. For example, $(0,0)$

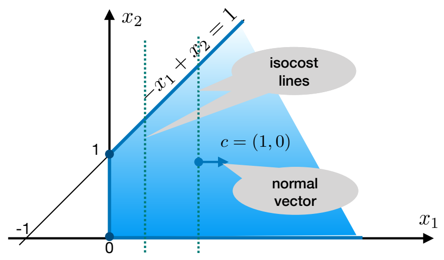


Fig. 3.4 **Case 2** of the LP (3.1) with $c = (1,0)$.

and $(0,1)$ are both optimal solutions. Hence, in this case the LP has multiple optimal solutions. Indeed, the set of optimal solutions can be written as

$$\{(0,x_2) \mid 0 \le x_2 \le 1\},$$

which is a bounded set.

Case 3: Let us choose $c_1 = -1$ and $c_2 = -1$. The isocost lines in this case are dashed lines in Figure 3.5. Note that the isocost lines in this case are the same as those in **Case 1**, but we need to push in the opposite direction to decrease the objective value. We can keep pushing the isocost line further and further, without ever leaving the feasible set. Thus, the LP is unbounded, which means that the objective value can be made arbitrarily negative for a minimization problem.

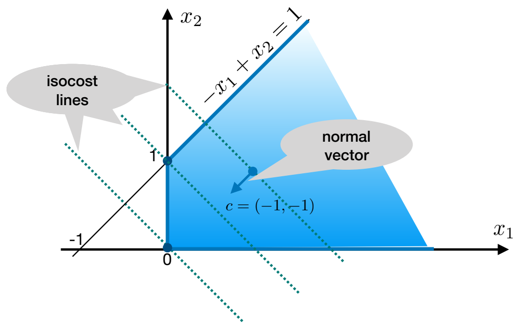


Fig. 3.5 **Case 3** in the LP (3.1) with $c = (-1,-1)$.

3.3 Forms of linear programs

3.3.1 Forms of LPs

An LP can be written in one of the three forms: general form, standard form, or canonical form.

3.3.1.1 The general form

By definition, any linear program can be written into the following **general form**:

$$\begin{cases} \min_{x\in\mathbb{R}^n} \ (\text{or } \max_{x\in\mathbb{R}^n}) \ z = \sum_{j=1}^n c_j x_j & \\ \text{subject to} \quad \sum_{j=1}^n a_{ij}x_j \le b_i, & i = 1,\cdots,m \\ \qquad\qquad \sum_{j=1}^n a_{ij}x_j \ge b_i, & i = m+1,\cdots,m+p \\ \qquad\qquad \sum_{j=1}^n a_{ij}x_j = b_i, & i = m+p+1,\cdots,m+p+q \\ \qquad\qquad x_j \ge 0, & j = 1,\cdots,k \\ \qquad\qquad x_j \le 0, & j = k+1,\cdots,k+l. \end{cases}$$

For a general LP:

- Its linear objective function $z = c^T x$ can be **maximized** or **minimized**.
- Its constraints can be a mixture of the three types:

  "$\le$": less than or equal to (LE),

  "$\ge$": great than or equal to (GE), and

  "$=$": equal to (E).

- Its variables can be any of the three types:

"≥ 0": nonnegative,

"≤ 0": nonpositive, and

"free": with no sign constraint.

- We note that linear programs may have bound constraints as $l_i \leq x_i \leq u_i$ for some $i \in \{1, \cdots, n\}$, where $l_i, u_i \in \mathbb{R}$ and $l_i \leq u_i$. If $l_i = u_i$, then x_i is called a fixed variable. If $l_i = -\infty$, then x_i does not have a lower bound, and if $u_i = +\infty$, then x_i does not have an upper bound. Bound constraints can be treated as LE and GE constraints, and sometimes it helps to handle them in special ways to improve algorithm efficiency.

Example 3.2. Here is an example of a general LP:

$$\begin{cases} \min_x \ z = x_1 - 2x_2 + x_3 - 3x_4 & \\ \text{s.t.} \ \ 2x_1 + x_2 - x_3 + x_4 = 3 & (E) \\ \qquad x_1 - 5x_2 + x_3 \leq 4 & (LE) \\ \qquad 3x_2 + 2x_2 - 3x_4 \geq -2 & (GE) \\ \qquad x_1 \geq 0, \ x_2 \leq 0, \ -2 \leq x_3 \leq 4. & \end{cases}$$

Here, x_4 is a free variable without any sign constraint, and x_3 has a bound constraint with $l_3 = -2$ and $u_3 = 4$.

3.3.1.2 The standard form

An LP is said to be in the **standard form** if it satisfies the following conditions simultaneously:

1. Each variable x_i in the LP is subject to the nonnegativity sign restriction (i.e., $x_i \geq 0$ for $i = 1, \cdots, n$).
2. Each other constraint in addition to the nonnegativity constraints is an equality constraint of the form $\sum_{j=1}^n a_{ij}x_j = b_i$ for some $i = 1, \cdots, m$.

Note: Some books restrict standard form LPs to either minimization or maximization problems. Our definition for standard form allows both minimization and maximization LPs, and the requirement is only on the format of constraints.

A standard LP can be represented as follows:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} \ (\text{or} \ \max\limits_{x\in\mathbb{R}^n}) \ z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{s.t.} \qquad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ \qquad\quad a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \qquad\quad \vdots \qquad\qquad\qquad \vdots \ \ \vdots \\ \qquad\quad a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m, \\ \qquad\quad x_i \geq 0, \quad i = 1, 2, \cdots, n. \end{cases}$$

Matrix representation of the standard form: If we define

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \ \text{and} \ b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix},$$

then we can write the above standard LP as

$$\begin{cases} \min\limits_{x} \ (\text{or} \ \max\limits_{x}) \ z = c^T x \\ \text{s.t.} \qquad Ax = b, \\ \qquad\quad x \geq 0. \end{cases} \tag{3.2}$$

Example 3.3. The following LP is in the standard form:

$$\begin{cases} \min\limits_{x} z = 2x_1 + x_2 - 3x_3 \\ \text{s.t.} \ \ 2x_1 + x_2 + x_3 = 6 \\ \qquad x_1 - 2x_2 + 3x_3 = 12 \\ \qquad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{cases}$$

Let us define

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -2 & 3 \end{bmatrix}, \quad b = \begin{pmatrix} 6 \\ 12 \end{pmatrix}, \quad c = \begin{pmatrix} 2 \\ 1 \\ -3 \end{pmatrix}, \ \text{and} \ x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Then, we can write this LP into the matrix form (3.2).

Example 3.4. Consider the following LP:

$$\begin{cases} \max\limits_x & z = x_1 + x_2 \\ \text{s.t} & 2x_1 + x_2 \le 60, \\ & x_1 + x_2 \le 20, \\ & x_1 \ge 0. \end{cases}$$

This LP is in general form but not in the standard form, because (i) the first two constraints are not equalities; and (ii) there is no nonnegativity requirement on the variable x_2.

3.3.1.3 The canonical form

We have seen that the standard form of an LP is a special case of the general form. There is a special case of the standard form, called the **canonical form**. That means

$$\text{canonical form} \quad \subset \quad \text{standard form} \quad \subset \quad \text{general form}.$$

Here, $A \subset B$ means that A is a subset of B.

An LP is said to be in the canonical form if it simultaneously satisfies the following conditions:

1. It is in the standard form.
2. The entries of the right-hand-side vector of the constraints are all nonnegative.
3. Each equality constraint **isolates** a variable. That is
   - This variable has a coefficient of 1 in this constraint;
   - This variable has a zero coefficient in all other equality constraints;
   - This variable has a zero coefficient in the objective function.

Example 3.5. The following LP is in a canonical form as it satisfies all above conditions.

$$\begin{cases} \max\limits_x z = x_1 + x_2 \\ \text{s.t} \quad 2x_1 + x_2 + x_3 = 60, \\ \qquad x_1 + x_2 + x_4 = 20, \\ \qquad x_1, x_2, x_3, x_4 \ge 0 \end{cases}$$

Why?

1. It is in the standard form.
2. The right-hand side vector of its equality constraints is $(60,20)^T$, which is nonnegative.
3. The coefficient of x_3 is 1 in the first equality constraint, and is 0 in the other equality constraint and the objective function, so x_3 is isolated by the first equality constraint. Similarly, x_4 is isolated by the second equality constraint.

If we define

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \quad b = \begin{pmatrix} 60 \\ 20 \end{pmatrix}, \quad c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \text{and} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix},$$

then we can rewrite this problem into the matrix form (3.2).

For a canonical LP with m equality constraints, we use the following notation.

- $x_{B(1)}$: the variable isolated by the 1^{st} equality constraint,
- $x_{B(2)}$: the variable isolated by the 2^{nd} equality constraint,
- $\cdots$
- $x_{B(m)}$: the variable isolated by the m^{th} equality constraint.

Then, from the definition, it follows that

$$c_{B(1)} = c_{B(2)} = \cdots = c_{B(m)} = 0,$$

and

$$A_{B(1)} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad A_{B(2)} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \cdots \quad A_{B(m)} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

Later, we will denote $c_B = (c_{B(1)}, \cdots, c_{B(m)})^T$, a column vector whose entries are $c_{B(1)}, \cdots, c_{B(m)}$, and $A_B = (A_{B(1)}, \cdots, A_{B(m)})$, an $m \times m$ submatrix of A whose columns are formed from m columns $A_{B(1)}, \cdots, A_{B(m)}$.

Note: If an LP is given by

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} \text{ (or } \max\limits_{x\in\mathbb{R}^n}\text{)} & z = c^T x \\ \text{s.t.} & Ax \le b, \\ & x \ge 0, \end{cases}$$

where b is a vector of nonnegative entries, then we can always convert it into the canonical form by introducing m slack variables $x_{n+1},\dots,x_{n+m}$ as we will see in the next subsection.

3.3.2 Converting from the general form to the standard form

Two maximization (minimization) linear programs are said to be *equivalent* if for every feasible solution of one problem there exists a feasible solution of the other problem with the same objective value. (In fact, as long as the difference between the two objective values is a constant for any such pair of feasible solutions, we can say the two LPs are equivalent.) Any LP in the general form can be converted into an *equivalent* LP in the standard form, by the following procedure. It is a step of the preprocessing stage in linear programming.

- **The objective function:** We note that $\max z = -\min(-z)$. Therefore, to convert from the maximization form to the minimization form, we simply need to change the values c_i in the objective to $-c_i$, and vice versa.
- **Nonpositive variables:** For a nonpositive variable $x_j \le 0$, we conduct a change of variable as $\hat{x}_j = -x_j$, where $\hat{x}_j$ is a new variable on which we impose the sign restriction $\hat{x}_j \ge 0$. We then substitute x_j by $-\hat{x}_j$ in the objective functions and all constraints.
- **Free variables:** For a free variable x_j, we introduce two new variables x_j^+ and x_j^- and write $x_j = x_j^+ - x_j^-$. In this case, x_j^+ and x_j^- are both required to be nonnegative, i.e., $x_j^+ \ge 0$ and $x_j^- \ge 0$. We then substitute x_j by $x_j^+ - x_j^-$ in the objective function and all constraints.
- **LE constraints:** For an inequality constraint

$$\sum_{j=1}^n a_{ij}x_j \le b_i,$$

we introduce a new variable s_i and transform this constraint into

$$\begin{aligned} \textstyle\sum_{j=1}^n a_{ij}x_j + s_i &= b_i, \\ s_i &\ge 0. \end{aligned}$$

Such a variable s_i is called a **slack** variable.

- **GE constraints:** For an inequality constraint

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i,$$

we introduce a new variable s_i and transform it into

$$\begin{aligned} \textstyle\sum_{j=1}^{n} a_{ij}x_j - s_i &= b_i, \\ s_i &\geq 0. \end{aligned}$$

Such a variable s_i is called a **surplus** variable.

There are other procedures for converting an LP in the general form into the standard form, see, e.g., [23]. We omit details of those procedures, but mention two common cases:

- **Fixed variables:** If x_j is a fixed variable, i.e., $x_j = u_j$ for a given number u_j, then we eliminate this variable by moving it to the right-hand side of the constraint, and remove it from the objective function.
- **Bound constraints:** For a bound constraint $l_j \leq x_j \leq u_j$, we can process as follows:
  - Introduce two new variables s_j^+ and s_j^- and split the constraint into two constraints as

$$x_j - s_j^- = l_i, \quad x_j + s_j^+ = u_j, \quad \text{where } s_j^+ \geq 0,\ s_j^- \geq 0.$$

  - Or we can do a change of variable $\hat{x}_j = x_j - l_j$ and introduce one slack variable s_j to write it as

$$\hat{x}_j + s_j = u_j - l_j, \ \text{ where } \hat{x}_j \geq 0,\ s_j \geq 0.$$

  Then, substitute x_j by $\hat{x}_j + l_j$ into the objective function and all the constraints. Finally, rearrange the resulting problem to remove constant terms.

- **Redundant constraints:** The row vectors of the constraint matrix A may be linearly dependent. In this case, we say that the LP has redundant constraints. One can remove such a redundant row by applying elementary row operations as discussed in the previous chapter. We skip the details.

Example 3.6. The following LP is in the general form, but not in the standard form. Convert it into the standard form.

$$\begin{cases} \max\limits_{x} & z = x_2 - 2x_3 \\ \text{s.t.} & x_1 + x_2 - 2x_3 \leq 1, \\ & 2x_1 - x_2 = 0, \\ & x_1 + x_3 \geq 0, \\ & 2 \leq x_2 \leq 5. \end{cases}$$

Solution: Applying the above procedure, we process as follows:

- Since x_1 is free, we introduce x_1^+ and x_1^- to write $x_1 = x_1^+ - x_1^-$ with $x_1^+ \geq 0$ and $x_1^- \geq 0$.
- Since x_2 has a bound constraint, we define $\hat{x}_2 = x_2 - 2$, and write $2 \leq x_2 \leq 5$ as $0 \leq \hat{x}_2 \leq 3$.
- Since x_3 is free, we introduce x_3^+ and x_3^- to write $x_3 = x_3^+ - x_3^-$ with $x_3^+ \geq 0$ and $x_3^- \geq 0$.

To process the objective, we note that $x_2 = \hat{x}_2 + 2$. Hence, we have $z = x_2 - 2x_3 = \hat{x}_2 + 2 - 2(x_3^+ - x_3^-) = \hat{x}_2 - 2x_3^+ + 2x_3^- + 2$. We define $w = \hat{x}_2 - 2x_3^+ + 2x_3^-$ by ignoring the constant 2.

Now, we process the constraints:

- For the first constraint $x_1 + x_2 - 2x_3 \leq 1$, we need a slack variable $s_1 \geq 0$, and write it as $x_1^+ - x_1^- + \hat{x}_2 + 2 - 2x_3^+ + 2x_3^- + s_1 = 1$, or equivalent to $x_1^+ - x_1^- + \hat{x}_2 - 2x_3^+ + 2x_3^- + s_1 = -1$.
- For the second constraint $2x_1 - x_2 = 0$, we write it as $2x_1^+ - 2x_1^- - \hat{x}_2 - 2 = 0$, or equivalent to $2x_1^+ - 2x_1^- - \hat{x}_2 = 2$.
- For the third constraint $x_1 + x_3 \geq 0$, we introduce a surplus variable $s_2 \geq 0$, and write it as $x_1^+ - x_1^- + x_3^+ - x_3^- - s_2 = 0$.
- For the last constraint $2 \leq x_2 \leq 5$, we already write it as $\hat{x}_2 \leq 3$ and $\hat{x}_2 \geq 0$. Hence, we need a slack variable $s_3 \geq 0$ to write it as $\hat{x}_2 + s_3 = 3$.

Finally, we write the entire standard LP as

$$\begin{cases} \max\limits_{x,s} & w = \hat{x}_2 - 2x_3^+ + 2x_3^- \\ \text{s.t.} & x_1^+ - x_1^- + \hat{x}_2 - 2x_3^+ + 2x_3^- + s_1 = -1, \\ & 2x_1^+ - 2x_1^- - \hat{x}_2 = 2, \\ & x_1^+ - x_1^- + x_3^+ - x_3^- - s_2 = 0, \\ & \hat{x}_2 + s_3 = 3, \\ & x_1^+, x_1^-, \hat{x}_2, x_3^+, x_3^-, s_1, s_2, s_3 \geq 0. \end{cases} \tag{3.3}$$

For notational simplicity, we can redefine $x = (x_1, x_2, \cdots, x_8)^T$ for $(x_1^+, x_1^-, \hat{x}_2, x_3^+, x_3^-, s_1, s_2, s_3)^T$.

3.3.3 Equivalence between the two LPs

To prove that the original LP in general form and the transformed LP in standard form (shortly, the standard LP) are equivalent, we show they satisfy the condition given in the definition of equivalent LPs at the beginning of the preceding subsection.

Given a feasible solution x of the original LP, we can perform the following steps to construct a corresponding feasible solution of the standard LP:

- For a free variable x_j, since $x_j = x_j^+ - x_j^-$, with $x_j^+ \geq 0$ and $x_j^- \geq 0$, we take $x_j^+ = \max\{0, x_j\} \geq 0$ and $x_j^- = \max\{0, -x_j\} = -\min\{0, x_j\} \geq 0$.
- For a slack variable s_i in the constraint $a_i^T x + s_i = b_i$, we define $s_i = b_i - a_i^T x \geq 0$.
- For a surplus variable s_i in the constraint $a_i^T x - s_i = b_i$, we define $s_i = a_i^T x - b_i \geq 0$.

Conversely, given a feasible solution of the standard LP, we can also construct the corresponding feasible solution of the original LP using similar rules.

Example 3.7. In Example 3.6, $x = (1,2,1)^T$ is a feasible solution to the first LP with objective value $z = 0$. We construct a feasible solution of the standard LP (3.3) as follows:

- given any feasible solution $(x_1, x_2, x_3) = (1,2,1)$ to the first LP, we can construct a feasible solution to the second LP as

$$\big(\max(x_1,0), -\min(0,x_1), x_2 - 2, \max(x_3,0), -\min(0,x_3), 1 - x_1 - x_2 + 2x_3, x_1 + x_3, 5 - x_2\big)^T$$
$$= (1,0,0,1,0,0,2,3)^T.$$

In this case, the objective value is $w = \hat{x}_2 - 2x_3^+ + 2x_3^- = -2$. Hence, $z = w + 2 = 0$, which shows the difference between two objective functions is a fixed constant 2.

- Conversely, given any feasible solution

$$(x_1^+, x_1^-, \hat{x}_2, x_3^+, x_3^-, s_1, s_2, s_3)^T$$

to the second LP (3.3), we can construct a feasible solution to the first LP as $(x_1^+ - x_1^-, \hat{x}_2 + 2, x_3^+ - x_3^-)$, which has the same objective value for the first LP as the given feasible solution for the second LP. For example, if $(x_1^+, x_1^-, \hat{x}_2, x_3^+, x_3^-, s_1, s_2, s_3)^T = (1,0,0,1,0,0,2,3)^T$, then $x = (1,2,1)^T$.

3.4 Basic solutions and basic feasible solutions

Let us consider a standard form LP written into the following matrix form:

$$\begin{cases} \max\limits_{x\in\mathbb{R}^n} & z \ = c^T x \\ \text{s.t.} & Ax = \ b, \\ & x \ \geq \ 0. \end{cases} \tag{3.4}$$

Throughout this section, we assume that A is an $m \times n$ matrix whose rows are linearly independent (which implies $m \leq n$).

3.4.1 Basic solutions

By linear algebra theory (see the preceding chapter), the above assumption implies that there exist m columns of A that are linearly independent. One can find a **basic solution** by the following procedure:

1. Choose m linearly independent columns $A_{B(1)}, \cdots, A_{B(m)}$. These columns are called **basic columns**. Let variables $x_{B(1)}, \cdots, x_{B(m)}$ be called **basic variables**, and the remaining variables be **nonbasic variables**. The collection of all basic variables, $\{x_{B(1)}, \cdots, x_{B(m)}\}$, is called a **basis**.
2. Set the nonbasic variables to zero. That is, let $x_i = 0$ for $i \notin \{B(1), \cdots, B(m)\}$. Then solve the system of m equations $Ax = b$ for values of the basic variables. When nonbasic variables are fixed to be zero, the system $Ax = b$ reduces to a system of m equations with m variables:

$$A_{B(1)}x_{B(1)} + \cdots + A_{B(m)}x_{B(m)} = b, \tag{3.5}$$

which has a unique solution because its coefficient matrix is invertible.

3. Put values of the nonbasic variables and basic variables together to obtain a **basic solution**.

Example 3.8. Consider the system of linear constraints of a standard LP:

$$\begin{cases} x_1 + 2x_2 + x_3 - x_4 &= 4 \\ 2x_1 + x_2 + 2x_3 + x_4 &= 5. \end{cases} \quad \text{or equivalent} \quad \begin{bmatrix} 1 & 2 & 1 & -1 \\ 2 & 1 & 2 & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}.$$

Here $m = 2$ and $n = 4$. It is easy to check that the row vectors of A are linearly independent. Note that the two first columns of A as $A_1 = (1,2)^T$ and $A_2 = (2,1)^T$ are linearly independent. By letting $B(1) = 1$ and $B(2) = 2$, we can form a basic solution $x = (x_1, x_2, 0, 0)^T$ where x_1 and x_2 are the solutions of the following square system:

$$\begin{cases} x_1 + 2x_2 &= 4 \\ 2x_1 + x_2 &= 5. \end{cases}$$

Solving this system, we have $x_1 = 2$ and $x_2 = 1$. This gives a basic solution $x = (2,1,0,0)^T$. □

From the above construction, one can see that a vector $x^* \in \mathbb{R}^n$ is a basic solution for the LP (3.4) if and only if it satisfies the following conditions:

1. $Ax^* = b$.
2. Columns of A corresponding to nonzero entries of x^* are linearly independent.

3.4.2 Basic feasible solutions

If a **basic solution** is also a feasible solution of the LP (3.4), then it is called a **basic feasible solution**. From its construction, the basic solution must satisfy the constraint $Ax = b$, so the additional requirement for it to be a basic feasible solution is the nonnegativity of all its components. Equivalently, we say that:

A vector $x^* \in \mathbb{R}^n$ is a basic feasible solution of (3.4) if and only if it satisfies the following conditions:

1. $Ax^* = b$ and $x^* \geq 0$.

2. Columns of A corresponding to positive entries of x^* are linearly independent.

For convenience, we define $B = \{B(1), \cdots, B(m)\}$ the index set of the basic columns of A; A_B is an $m \times m$ matrix formed from $A_{B(i)}$ for $i = 1, \ldots, m$. The nonbasic index set is denoted by $N = \{1, 2, \cdots, n\} \backslash B$. Hence, we can also define the nonbasic submatrix as A_N. In this case, the matrix A can be decomposed as $A = [A_B, A_N]$, and vector c can also be decomposed as $c = [c_B, c_N]$. [1]

Example 3.9. Consider the following LP:

$$\begin{cases} \min\limits_{x} & z = x_1 - x_2 + x_3 \\ \text{s.t.} & \quad x_1 - 2x_2 - x_3 = 2, \\ & \quad x_1 + x_2 - x_3 = 2, \\ & \quad x \geq 0. \end{cases} \tag{3.6}$$

There are 3 possible ways to choose basic columns:

- **Choice 1:** $B(1) = 1, B(2) = 2$: We have $A_B = \begin{bmatrix} 1 & -2 \\ 1 & 1 \end{bmatrix}$. Hence, a basic solution $x = (x_1, x_2, x_3) = (2, 0, 0)$. This is also a basic feasible solution since $x \geq 0$.
- **Choice 2:** $B(1) = 2, B(2) = 3$: We have $A_B = \begin{bmatrix} -2 & -1 \\ 1 & -1 \end{bmatrix}$. A basic solution $x = (x_1, x_2, x_3) = (0, 0, -2)$. This is not a basic feasible solution since $x_3 < 0$.
- **Choice 3:** $B(1) = 1, B(2) = 3$: We have $A_B = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$. These two columns are linearly dependent. Accordingly, this choice does not provide a basic solution.

A basic feasible solution of (3.4) is said to be **nondegenerate**, if exactly $n - m$ of its components are zero. It is **degenerate** if more than $n - m$ of its components are zero.

Since every basic feasible solution is a basic solution, the nonbasic variables in it are all zero. The nondegeneracy requires all the basic variables in a basic feasible solution to be strictly positive.

[1] Here, we need to re-order the variables so that we can concatenate both submatrices to form A.

Example 3.10. In the above example, we have $n = 3$ and $m = 2$. The basic solution $(2,0,0)$ is degenerate since x_2, one of its basic variables, is 0.

Basic optimal solutions:

If an optimal solution of the LP (3.4) is also a **basic feasible solution**, then we say that it is a basic optimal solution. The following is an important fact for linear programming, whose proof is out of scope of this course.

Fact 3.4.1 *If a linear program of the form* (3.4) *has an optimal solution, then it has a* ***basic optimal solution****.*

3.4.3 Geometric interpretation of basic feasible solutions

The feasible set $\mathscr{X} := \{x \mid Ax = b, x \geq 0\}$ of the standard LP (3.4) is a **polyhedron**. In general, a polyhedron in $\mathbb{R}^n$ is a set defined by finitely many linear constraints. A point x in a polyhedron is called an **extreme point** of this set, if there do not exist two distinct points y and z in this set, such that $x = \frac{1}{2}y + \frac{1}{2}z$ (i.e., x is not the middle point of any two points y and z in this set). Geometrically, extreme points of a polyhedron are its vertices (or corner points). Below, we show that basic feasible solutions of (3.4) are exactly the extreme points of the feasible set $\mathscr{X}$.

Theorem 3.1. *A vector $x^* \in \mathbb{R}^n$ is a basic feasible solution of* (3.4) *if and only if it is an extreme point of $\mathscr{X}$.*

Proof. First, we prove that any basic feasible solution is an extreme point. Suppose x^* is a BFS, and let $\{x_{B(1)}, \cdots, x_{B(m)}\}$ be a basis that can be used to construct x^*. This means that $x_i^* = 0$ for each $i \notin \{B(1), \cdots, B(m)\}$ and that columns $A_{B(1)}, \cdots, A_{B(m)}$ are linearly independent. Suppose that $y \in \mathscr{X}$ and $z \in \mathscr{X}$ satisfy $x^* = \frac{y+z}{2}$; we will show that $y = z = x^*$.

Consider an index $i \notin \{B(1), \cdots, B(m)\}$. For such an index, we have $x_i^* = 0$, so that

$$y_i + z_i = 2x_i^* = 0.$$

Since y and z both belong to $\mathscr{X}$, they satisfy $y_i \geq 0$ and $z_i \geq 0$. The only possibility for two nonnegative numbers to sum to 0 is that both numbers are exactly 0. We have therefore shown that $y_i = z_i = 0 = x_i^*$ for each index $i \notin \{B(1), \cdots, B(m)\}$.

The two points y and z also satisfy $Ay = Az = b$. Eliminating the zero components gives

$$A_{B(1)}y_{B(1)} + A_{B(2)}y_{B(2)} + \cdots + A_{B(m)}y_{B(m)} = b$$

and

$$A_{B(1)}z_{B(1)} + A_{B(2)}z_{B(2)} + \cdots + A_{B(m)}z_{B(m)} = b.$$

However, the columns $A_{B(1)}, \cdots, A_{B(m)}$ are known to be linearly independent, so the matrix A_B that consists of those columns is an invertible matrix. Thus,

$$\begin{bmatrix} y_{B(1)} \\ y_{B(2)} \\ \vdots \\ y_{B(m)} \end{bmatrix} = \begin{bmatrix} z_{B(1)} \\ z_{B(2)} \\ \vdots \\ z_{B(m)} \end{bmatrix} = A_B^{-1}b = \begin{bmatrix} x^*_{B(1)} \\ x^*_{B(2)} \\ \vdots \\ x^*_{B(m)} \end{bmatrix}.$$

This proves that $y = z = x^*$. We have thereby shown that any BFS is an extreme point.

Next, we prove that any extreme point is a basic feasible solution. Suppose x^* is an extreme point. To prove that x^* is a BFS, it suffices to show that columns of A corresponding to positive entries of x^* are linearly independent. Suppose without loss of generality that $x^*_i > 0$ for $i = 1, \cdots, k$ and $x^*_i = 0$ for $i = k+1, \cdots, n$. We need to show that $A_1, \cdots, A_k$ are linearly independent. Suppose they are not linearly independent. Then there exist scalars $\lambda_1, \cdots, \lambda_k$, not all zero, such that $\lambda_1 A_1 + \cdots \lambda_k A_k = 0$. Let t be a small positive number. Define two points $y \in \mathbb{R}^n$ and $z \in \mathbb{R}^n$ as follows:

$$y_i = \begin{cases} x^*_i + t\lambda_i & i = 1, \cdots, k \\ 0 & i = k+1, \cdots, n \end{cases}$$

and

$$z_i = \begin{cases} x^*_i - t\lambda_i & i = 1, \cdots, k \\ 0 & i = k+1, \cdots, n \end{cases}$$

We have

$$Ay = A_1y_1 + A_2y_2 + \cdots + A_ky_k = A_1(x^*_1 + t\lambda_1) + A_2(x^*_2 + t\lambda_2) + \cdots + A_k(x^*_k + t\lambda_k) = b$$

because $A_1x^*_1 + A_2x^*_2 + \cdots + A_kx^*_k = b$ and $\lambda_1 A_1 + \cdots \lambda_k A_k = 0$. Similarly we have $Az = b$. By choosing t to be sufficiently small, it can be guaranteed that $y_i \geq 0$ and $z_i \geq 0$ for all $i = 1, \cdots, k$, so that $y \in \mathscr{X}$ and $z \in \mathscr{X}$. Moreover, $x^* = \frac{y+z}{2}$. The fact that λ_i's are not all zero implies that y and z are two different points in $\mathscr{X}$. We have thereby written x^* as the midpoint of two distinct points in $\mathscr{X}$, which contradicts with the initial assumption that x^* is an extreme point. This contradiction is caused by assuming the columns $A_1, \cdots, A_k$ to be linearly dependent.

We can also "project" the feasible set $\mathcal{X}$ of the standard LP (3.4) in $\mathbb{R}^n$ onto a set in the smaller space $\mathbb{R}^{n-m}$ in the following way. Let A_B be an $m \times m$ invertible submatrix of A, write $A = [A_B, A_N]$ and decompose $x = (x_B, x_N)$ correspondingly. The constraint $Ax = b$ can be written as $A_B x_B + A_N x_N = b$. In this case, we can write

$$x_B + A_B^{-1} A_N x_N = A_B^{-1} b \quad \text{or equivalent to} \quad x_B = A_B^{-1} b - A_B^{-1} A_N x_N.$$

The constraint $x_B \geq 0$ then translates to the condition that $A_B^{-1} A_N \bar{x}_N \leq A_B^{-1} b$. Hence, if we define $\bar{A}_N = A_B^{-1} A_N$ and $\bar{b} := A_B^{-1} b$, then the feasible set $\mathcal{X}$ of the standard LP (3.4) in $\mathbb{R}^n$ can be projected to the subspace $\mathbb{R}^{n-m}$ as

$$\bar{\mathcal{X}} := \left\{ x_N \in \mathbb{R}^{n-m} \mid \bar{A}_N x_N \leq \bar{b}, \ \ x_N \geq 0 \right\}.$$

This set is again a polyhedron in $\mathbb{R}^{n-m}$. Each point x_N of this polyhedron can be extended into a feasible solution of (3.4) as (x_B, x_N) with $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$, which projects back to x_N. The two functions (the extension and the projection) together define a one-to-one correspondence between points in $\mathcal{X}$ and $\bar{\mathcal{X}}$. In particular, each extreme point of $\bar{\mathcal{X}}$ corresponds to a basic feasible solution of (3.4).

Example 3.11. To illustrate the geometric interpretation of basic feasible solutions in LPs, let us consider the standard LP (3.4) in $\mathbb{R}^4$ with the following feasible set

$$\mathcal{X} = \left\{ x \in \mathbb{R}^4 \mid x_1 + x_2 + x_3 + x_4 = 4, \ \ x_1 + 2x_2 - 2x_3 + 4x_4 = 6, \ \ x \geq 0 \right\}.$$

This problem has 4 variables ($n = 4$) and 2 constraints ($m = 2$). It is easy to check that $\bar{x} = (2, 2, 0, 0)^T$ is a basic feasible solution of this problem. Hence, we can let $B = \{1, 2\}$, $N = \{3, 4\}$, and form submatrices $A_B = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$ and $A_N = \begin{bmatrix} 1 & 1 \\ -2 & 4 \end{bmatrix}$. We then express x_B as a function of x_N as

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{pmatrix} 4 \\ 6 \end{pmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 \\ -2 & 4 \end{bmatrix} x_N = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - \begin{bmatrix} 4 & -2 \\ -3 & 3 \end{bmatrix} \begin{pmatrix} x_3 \\ x_4 \end{pmatrix}.$$

Now, if we define $\bar{A} = \begin{bmatrix} 4 & -2 \\ -3 & 3 \end{bmatrix}$ and $\bar{b} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, then the projected feasible set of this problem becomes

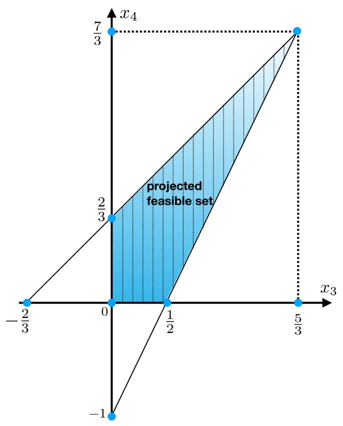


Fig. 3.6 The projected feasible set onto $\mathbb{R}^2$ of the standard LP

$$\bar{\mathscr{X}} = \left\{x_N = (x_3,x_4)^T \in \mathbb{R}^2 \mid 4x_3 - 2x_4 \leq 2, \;\; -3x_3 + 3x_4 \leq 2, \;\; x_3 \geq 0, \; x_4 \geq 0\right\}.$$

This is a subset in $\mathbb{R}^2$. We can plot this projected feasible set as in Figure 3.6, which is a quadrilateral.

This quadrilateral has 4 vertices at $x_N = (0,0)^T$, $x_N = (\frac{1}{2},0)^T$, $x_N = (\frac{5}{3},\frac{7}{3})^T$ and $x_N = (0,\frac{2}{3})^T$. For each x_N, we can compute the corresponding x_B as $x_B = (2,2)^T$, $x_B = (0,\frac{7}{2})^T$, $x_B = (0,0)^T$ and $x_B = (\frac{10}{3},0)^T$, respectively. Hence, we obtain 4 basic feasible solutions of the original problem as

$$\bar{x}^{(1)} = (2,2,0,0)^T, \;\; \bar{x}^{(2)} = (0,\tfrac{7}{2},\tfrac{1}{2},0)^T, \;\; \bar{x}^{(3)} = (0,0,\tfrac{5}{3},\tfrac{7}{3})^T, \;\; \text{and } \bar{x}^{(4)} = (\tfrac{10}{3},0,0,\tfrac{2}{3})^T.$$

These basic feasible solutions are all nondegenerate.

3.4.4 Finding a basic feasible solution

To find a basic feasible solution for an LP in the standard form, one would need to try various choices of basic columns, compute the basic solutions and check if they are feasible. We can perform two steps:

- Choose m basic columns A_B of A, and compute $\bar{x}_B = A_B^{-1}b$. If $\bar{x}_B \geq 0$, then let $\bar{x}_N = 0$ and form $\bar{x} = (\bar{x}_B, \bar{x}_N)$ as a basic feasible solution.
- Otherwise, repeat this step by choosing another set of basic columns A_B.

This procedure may require many iterations, but it terminates after at most $\frac{n!}{m!(n-m)!}$ iterations, a very big number when n is large.

Example 3.12. In Example 3.11, we can choose $B = \{1,2\}$ and $A_B = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$, which is invertible. We compute $\bar{x}_B = A_B^{-1} b = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{pmatrix} 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \geq 0$. Hence, we can form a basic feasible solution $\bar{x} = (2,2,0,0)^T$.

If the LP (3.4) is not only in the standard form but also in the canonical form, then it is much easier to find a basic feasible solution of it. As in Section 3.3, let $x_{B(i)}$ be the variable isolated by the i-th equality constraint for each $i = 1, \cdots, m$. Their corresponding columns satisfy

$$A_{B(1)} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad A_{B(2)} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \cdots, \quad A_{B(m)} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix},$$

which are linearly independent. Let $x_{B(1)}, \cdots, x_{B(m)}$ be the basic variables, and set other variables to zero. Since columns $A_{B(1)}, \cdots, A_{B(m)}$ form the identity matrix $\mathbb{I}_m$ (i.e., $A_B = \mathbb{I}_m$), the system (3.5) has a unique solution given by $x_{B(i)} = b_i$ for each i. Since the LP is in the canonical form, its right-hand-side constants b_i are all nonnegative. The basic solution obtained by this way is therefore a basic feasible solution, and is called *the basic feasible solution naturally associated with the LP in canonical form*, or simply the *natural basic feasible solution*.

Example 3.13. In Example 3.5, the basic feasible solution naturally associated with the LP in the canonical form is $x = (0,0,60,20)^T$, with the isolated variables x_3 and x_4 being basic variables.

3.4.5 Existence of optimal solutions

The following theorem summarizes conditions for existence of an optimal solution (or a basic optimal solution) in linear programs. The phrase "has a" here means "has at least one." The condition "the objective value is bounded from above on its feasible set" means that you can find a fixed constant M such that the objective value achieved by any feasible solution is no more than M. The first statement of this theorem justifies the completeness of the four-type classification of linear programs in Section 3.2. The second statement restates Fact 3.4.1. The proof can be found in [7, 3], for example.

Theorem 3.2. *A maximization linear program has an optimal solution if and only if its feasible set is nonempty and the objective value is bounded from above on its feasible set. For an LP in standard form* (3.4)*, if it has an optimal solution, then it has a basic optimal solution.*

The second statement of Theorem 3.2 has the following extension, that is not restrictive to standard LPs. Here we use the definition of extreme points given in Subsection 3.4.3. The proof can be found in [3].

If an LP has an optimal solution and its feasible set has an extreme point, then there exists an extreme point that is optimal.

Based on the above result, if we know that the feasible set of an LP has some extreme points (vertices), and that this LP is not unbounded, then the extreme point with the best objective value is guaranteed to be an optimal solution.

3.5 Exercises

Exercise 3.1. Megabuck Hospital Corporation is going to build a subsidized nursing home catering to homeless and high-income patients.

- The overall capacity of the new hospital should be 2100 patients.
- State regulations require such hospitals to house a minimum of 1000 homeless patients and no more than 750 high-income patients in order to receive the subsidy.
- The board of directors, under pressure from a neighborhood group, insists that the number of homeless patients will not exceed twice the number of high-income patients.
- The hospital will make a profit of $\$10,000$ per month for each homeless patient, and $\$8,000$ per month for each high-income patient.

1. Formulate an LP problem to find how many beds should be reserved for each type of patient in order to maximize the hospital's profit.

2. Solve the LP problem graphically. Label the vertices (i.e., corners) of the feasible set with their coordinates, and then find the optimal solution(s) and the optimal value. You can use the method of graphing isoprofit lines as did in class, or you can compare the objective function value at each vertex and choose the best.
3. Interpret your solution.

Exercise 3.2. Consider the following LP problem:

$$\begin{cases} \min\limits_{x_1,x_2} & z = 3x_1 + x_2 \\ \text{subject to} & x_1 \qquad \geq 3, \\ & x_1 + x_2 \geq 4, \\ & x_2 \qquad \geq 0. \end{cases}$$

1. Graphically show the feasible set. Label the vertices (i.e., corners) of the feasible set with their coordinates.
2. Find the optimal solution(s) and the optimal value. What type of LP is this?
3. Suppose now the objective function changes to min $x_1 + x_2$. What type of LP is this? Find the optimal solution(s) and the optimal value if they exist.
4. Suppose now the objective function changes to min $x_1 - x_2$. What type of LP is this? Find the optimal solution(s) and the optimal value if they exist.
5. Suppose now that we add an additional constraint $x_1 + 2x_2 \leq 3$ to the original LP. What type of LP is this? Find the optimal solution(s) and the optimal value if they exist.

Exercise 3.3. Consider the following linear programming problem of two variables:

$$\begin{cases} \max\limits_{x_1,x_2} z = 2x_1 + x_2 \\ \text{s.t.} \quad a_1x_1 + a_2x_2 \leq b \\ \qquad x_1 \geq 0,\ x_2 \geq 0. \end{cases} \tag{3.7}$$

where a_1, a_2 and b are given

1. Find one value of a_1, a_2, and b such that the feasible set is empty (infeasible LP problem).
2. Find one value of a_1, a_2, and b such that the feasible set is unbounded.
3. Find one value of a_1, a_2, and b such that the problem is unbounded.

4. Find one value of a_1, a_2, and b such that the problem has multiple solutions.
5. Is there any value of a_1, a_2, and b such that the feasible set is unbounded, but the problem has an optimal solution?

Show your results graphically on an x_1x_2-plane.

Make sure that you understand the difference between the unbounded feasible set and the unbounded problem. The unbounded problem must have an unbounded feasible set, and its objective function goes to infinity on such a set. But an LP problem has an unbounded feasible set, it may still have optimal solution with finite optimal value.

Exercise 3.4. Given the following linear programming problem with three variables:

$$\begin{cases} \min\limits_{x_1,x_2,x_3} & z = c_1x_1 + c_2x_2 + c_3x_3 \\ \text{s.t.} & -x_1 + x_2 + x_3 = 1 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0. \end{cases} \tag{3.8}$$

where the normal vector $c = (c_1, c_2, c_3)$ of the objective function is given.

1. Reformulate this problem into a two variables linear program (we call it a *reduced LP problem*).
2. Plot the feasible domain of the reduced problem in a plane.
3. Is there any triple (c_1, c_2, c_3) such that the LP problem (3.8) is unbounded? If yes, find one triple.
4. Is there any triple (c_1, c_2, c_3) such that the LP problem (3.8) has multiple solutions? If yes, find one triple.

Exercise 3.5. You are given the following linear programming problems:

1.
$$\begin{cases} \min\limits_{x} & z = 2x_1 + 5x_2 \\ \text{s.t.} & x_1 \geq 5, \\ & x_1 + x_2 \leq 9, \\ & -2x_1 + x_2 = -4, \\ & x_1, x_2 \geq 0. \end{cases}$$

2.

$$\begin{cases} \max\limits_{x} & z = x_1 + 2x_2 + 2x_3 + 3x_4 \\ \text{s.t.} & 2x_1 + x_2 + x_3 + x_4 \geq 15, \\ & x_1 + x_2 \leq 6, \\ & x_1 \geq 0,\ x_2 \leq 0,\ x_4 \geq 0. \end{cases}$$

3.

$$\begin{cases} \max\limits_{x} & z = x_1 + 3x_3 \\ \text{s.t.} & x_1 + x_3 + x_4 = 12, \\ & x_1 + x_2 + x_3 = 8, \\ & x_1, x_2, x_3, x_4 \geq 0. \end{cases}$$

4.

$$\begin{cases} \max\limits_{x} & z = x_1 - 2x_2 - x_3 + x_4 \\ \text{s.t.} & x_1 + x_3 + x_4 + x_5 = 14, \\ & x_1 + x_2 + x_3 - x_6 = -12, \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{cases}$$

Solve the following questions for each problem:

(a) Determine if each of the above linear programs is in standard form or canonical form. For linear programs in canonical form, specify the isolated variables.

(b) For linear programs not in standard form, convert them into standard form. Check if the converted problem is canonical.

(c) If the problem is feasible, then given a feasible solution $\bar{x}$ of each problem above, how can you form a feasible solution for the corresponding problem that you convert to?

(d) If the problem is feasible, then find a specific feasible solution for each problem above. Then, form a corresponding feasible solution of the converted one from this feasible solution.

Exercise 3.6. Consider the following LP problem:

$$\begin{cases} \max\limits_{x} & z = 2x_1 + 3x_2 \\ \text{s.t.} & x_1 + x_2 + x_3 - x_4 = 2 \\ & x_1 - x_2 + x_4 = 0 \\ & x \geq 0. \end{cases} \tag{3.9}$$

1. List all the basic solutions. Identify all the nondegnerate basic feasible solutions and degenerate basic feasible solutions.

2. Project the feasible set onto the (x_1, x_2) plane: first express x_3 and x_4 as functions of x_1 and x_2, then convert the nonnegativity constraints on x_3 and x_4 into constraints on x_1 and x_2, and finally write down the set of (x_1, x_2) that satisfies the nonnegativity constraints on themselves and the constraints transformed from the nonnegativity of x_3 and x_4.
3. Graph the set you obtain in question 2 in the (x_1, x_2) plane, and label all of its vertices.
4. Does the LP in (3.9) have an optimal solution? If yes, what are the solutions, and the optimal value?

Exercise 3.7. Consider the following LP problem in standard form, with a single equality constraint:

$$\begin{cases} \max\limits_{x} \ \sum\limits_{i=1}^{n} c_i x_i \\ \quad \text{s.t.} \ \sum\limits_{i=1}^{n} a_i x_i = b \\ \quad\quad x_i \geq 0, \ i = 1, \cdots, n \end{cases} \tag{3.10}$$

Suppose $b > 0$, and $a_i > 0$ for each $i = 1, \cdots, n$.

1. Write down all the basic feasible solutions.
2. For each $i = 1, \cdots, n$, find a number M_i, which may depend on b and a_i, such that any feasible solution x satisfies $x_i \leq M_i$.
3. If $c_i \geq 0$ for all $i = 1, \cdots, n$, then does the LP in (3.10) have an optimal solution? If yes, what is it? and what is the optimal value?
4. If there is no condition on c, then does the LP in (3.10) have an optimal solution? If yes, what is it? and what is the optimal value?
5. Is the feasible set of (3.10) bounded? (This is an extra question without any credit, just for you to understand LPs).

Exercise 3.8. The bartender of your local pub asks you to assist him in finding the combination of mixed drinks to maximize his revenue. He has the following bottles available:

- 1 quart (32 oz.) Old Cambridge (a fine whiskey)
- 1 quart Joy Juice (another fine whiskey)
- 1 quart Ma's Wicked Vermouth

- 2 quarts Gil-boy's Gin

Since he is new to the business, his knowledge is limited to the following drinks:

- *Whiskey Sour*. Each serving uses 2 oz. whiskey and sells for \$1.
- *Manhattan*. Each serving uses 2 oz. whiskey and 1 oz. vermouth, and sells for \$2.
- *Martini*. Each serving uses 2 oz. gin and 1 oz. vermouth, and sells for \$2.
- *Pub Special*. Each serving uses 2 oz. gin and 2 oz. whiskey, and sells for \$3.

First, formulate an LP model to maximize the bar's profit. Ignore the fact that the numbers of servings have to be integers. Second, if the problem is not canonical, convert it into a standard form, and you will see the resulting problem is canonical.

Chapter 4
Simplex Methods for LPs

This chapter introduces the simplex method for solving linear programs. The simplex method was invented by G. Dantzig in 1947 [7] when he was working at the US Army Air Force. This method is still one of the most efficient methods for solving LPs, and is available in most software packages for LPs.

Before applying the simplex method, we need to convert a general LP into standard form using techniques discussed in the preceding chapter. The complete simplex method consists of two phases. The goal of the first phase is to decide if the LP is feasible, eliminate any redundant equality constraints, and find an initial basic feasible solution when the LP is feasible. Following that, at the second phase, we will start with that initial basic feasible solution, and carry out iterations until we find an optimal basic solution, or find the LP to be unbounded.

For reasons that will become clear, we will start by introducing the *second* phase of the complete simplex method. For brevity, people sometimes just use **the simplex method** to refer to the second phase of the complete simplex method, and use **the two-phase simplex algorithm** to refer to the complete simplex method that consists of the two phases. We will adopt this convention in the sequel. Below, Section 4.1 shows how to perform the simplex method in the matrix form, assuming that an initial basic feasible solution is known. Following that, Section 4.2 shows how to perform the same method in the format of simplex tableaux, assuming that the LP is given in canonical form (which implies the existence of a natural basic feasible solution). Finally, Section 4.3 presents the complete two-phase simplex algorithm, adding the first phase to the second phase to handle any LP.

4.1 The simplex method in matrix form

Let us consider the following standard LP

$$\begin{cases} \max\limits_{x\in\mathbb{R}^n} z = c^T x \\ \text{s.t.}\quad Ax = b \\ \qquad\quad x \geq 0, \end{cases} \tag{4.1}$$

where $A \in \mathbb{R}^{m\times n}$. We assume that rows of A are linearly independent, and that we already have a basic feasible solution $\hat{x} = (\hat{x}_B, \hat{x}_N)$, where $\hat{x}_B$ is the subvector consisting of values of the basic variables $x_{B(1)}, \cdots, x_{B(m)}$, and $\hat{x}_N = 0$ is the subvector consisting of values (all zeros) of the nonbasic variables $x_{N(1)}, \cdots, x_{N(n-m)}$. The corresponding decomposition of A is $A = [A_B, A_N]$, where A_B is an $m\times m$ invertible matrix consisting of the basic columns, and A_N is an $m \times (n-m)$ matrix consisting of the nonbasic columns. With $A\hat{x} = A_B\hat{x}_B + A_N\hat{x}_N = b$ and $\hat{x}_N = 0$, we have $A_B\hat{x}_B = b$, which implies $\hat{x}_B = A_B^{-1}b$. Using this decomposition, we can rewrite (4.1) as

$$\begin{cases} \max\limits_{x} z = c_B^T x_B + c_N^T x_N \\ \text{s.t.}\quad A_B x_B + A_N x_N = b, \\ \qquad\quad x_B \geq 0,\ x_N \geq 0, \end{cases}$$

where $x_B = (x_{B(1)}, \cdots, x_{B(m)})$ consists of all basic variables and $x_N = (x_{N(1)}, \cdots, x_{N(n-m)})$ consists of all nonbasic variables.

4.1.1 The idea of the simplex method

The idea behind the simplex method is as follows:

- Solve x_B from $A_B x_B + A_N x_N = b$ as $x_B = A_B^{-1}(b - A_N x_N) = A_B^{-1}b - A_B^{-1}A_N x_N$.
- Write the above LP equivalently as

$$\begin{cases} \max\limits_{x} z = c_B^T A_B^{-1} b + (c_N^T - c_B^T A_B^{-1} A_N) x_N \\ \text{s.t.}\quad x_B + A_B^{-1} A_N x_N = A_B^{-1} b, \\ \qquad\quad x_B \geq 0,\ x_N \geq 0. \end{cases} \tag{4.2}$$

- Denote $\bar{c}_N^T = c_B^T A_B^{-1} A_N - c_N^T$, the vector of **reduced costs** for the nonbasic variables. We can then write the objective function as $z = c_B^T A_B^{-1} b - \bar{c}_N^T x_N$. If $\bar{c}_N \geq 0$, then for any feasible solution x we have $z \leq c_B^T A_B^{-1} b$, while the z value achieved by the current basic feasible solution $\hat{x} = (\hat{x}_B, \hat{x}_N) = (A_B^{-1}b, 0)$ is exactly $c_B^T A_B^{-1} b$. Hence, the current basic feasible solution $\hat{x}$ is already optimal. We therefore claim that an optimal solution is found and terminate the method.

- **Determine an entering variable:** Otherwise, if the jth element of $\bar{c}_N$ is < 0 for some j, let $j_p = N(j)$. Any increase in the value of x_{j_p} will cause an increase in the objective value z, when the values of other nonbasic variables are fixed at their current value of zero. If there are more than one choice for such an index j_p, we can choose any of them. The variable x_{j_p} is called the **entering variable**, and the index j_p is called the index of the **pivot column**.
- Now, the idea is to try to increase the value of x_{j_p}, whose value in the current basic feasible solution $\hat{x}$ is zero. In the mean time, we keep values of other nonbasic variables at their current value of zero. This will affect values of the basic variables x_B through the equation $x_B + A_B^{-1} A_N x_N = A_B^{-1} b$, which can now be written as

$$x_B + x_{j_p} A_B^{-1} A_{j_p} = A_B^{-1} b \quad \Leftrightarrow \quad x_B = A_B^{-1} b - x_{j_p} A_B^{-1} A_{j_p}.$$

Let us denote by $\bar{b} = A_B^{-1} b$ and the column vector $d_{j_p} = A_B^{-1} A_{j_p}$. Then, the above constraint becomes $x_B = \bar{b} - x_{j_p} d_{j_p}$. Recall the nonnegativity constraint $x_B \geq 0$. Clearly, if $d_{j_p} = A_B^{-1} A_{j_p} \leq 0$, then $x_B \geq 0$ will always hold no matter how much we increase the value of x_{i_p}. In this case, we can keep increasing the value of x_{j_p} with values of other nonbasic variables fixed at zero, without ever violating any nonnegativity sign restrictions on x_B. Hence, the value of z can be made arbitrarily large. We therefore conclude that the LP is unbounded, and terminate the method.

- **Determine a leaving variable:** Otherwise, some entries of d_{j_p} are positive. We choose a row $i_p \in \{1, \cdots, m\}$ from

$$\operatorname*{arg\,min}_{i \in \{1,\cdots,m\}:\ d_{ij_p} > 0} \left\{ \frac{\bar{b}_i}{d_{ij_p}} \right\}.$$

That is: we choose the index that has the minimum ratio $\frac{\bar{b}_i}{d_{ij_p}}$ provided that $d_{ij_p} > 0$. If there are many indices that achieve the minimum ratio, we arbitrarily choose one of them. The idea is then to increase the value of x_{j_p} to

$$\min_{i \in \{1,\cdots,m\}:\ d_{ij_p} > 0} \left\{ \frac{\bar{b}_i}{d_{ij_p}} \right\}$$

with values of other nonbasic variables fixed at zero, which will drive the value of $x_{B(i_p)}$ to 0 and keep $x_{B(i)} \geq 0$ for all other indices $i = 1, \cdots, m$. This will give a feasible solution with a better (at least not worse) objective value, which is in fact also a basic feasible solution because replacing the column $A_{B(i_p)}$ by A_{j_p} in the basic columns preserves the required linear independency [1]. By replacing the variable $x_{B(i_p)}$ by x_{j_p} in the basis, we can obtain this basic feasible solution. We name $x_{B(i_p)}$ as the **leaving variable**, as it is the variable to leave the basis. The row i_p is called the **pivot row**.

We repeat this procedure until one of the two termination conditions is met.

4.1.2 The procedure of the simplex method

The simplex method is an iterative method, each iteration of which computes a new basic feasible solution. We algorithmically present the simplex method here. We have the following remarks on the implementation of this method.

- If the standard LP (4.1) is in canonical form, then we can let $B(i)$ be the index of the variable isolated by the ith equation. In this way, $A_{B(i)}$ is the ith unit vector, and $A_B = \mathbb{I}_m$, the identity matrix. Hence, at the beginning of the first iteration, we have $A_B^{-1} = \mathbb{I}_m$, which implies $\bar{c}_N^T = c_B^T A_N - c_N^T$, $\bar{b} = b$, and $d_{j_p} = A_{j_p}$.
- At the intermediate iterations, instead of directly inverting the newly obtained A_B to obtain A_B^{-1}, we perform elementary row operations on the previous A_B^{-1} to obtain the new A_B^{-1}. More specifically, let $A_B = \left[A_{B(1)} \cdots A_{B(m)}\right]$ be the old matrix, and $A_{B^*} = \left[A_{B(1)} \cdots A_{B(i_p-1)}\ A_{j_p}\ A_{B(i_p+1)} \cdots A_{B(m)}\right]$ be the new matrix. Because $A_B^{-1} A_{B(i)} = e_i$ for each $i = 1, \cdots, m$, we have

  $$A_B^{-1} A_{B^*} = \left[e_1 \cdots e_{l-1}\ A_B^{-1} A_{j_p}\ e_{l+1} \cdots e_m\right].$$

  Now put the matrix $A_B^{-1} A_{B^*}$ side by side with A_B^{-1}, and conduct elementary row operations to change $A_B^{-1} A_{B^*}$ to the identity matrix $\mathbb{I}_m$. By conducting the same elementary row operations to the matrix A_B^{-1}, we obtain $A_{B^*}^{-1}$.

[1] Note that $A_B^{-1} A_B = \mathbb{I}_m$, and $A_B^{-1} A_{B(i_p)} = e_{i_p}$ is the i_p'th unit vector. On the other hand, the i_p'th element of $d_{j_p} = A_B^{-1} A_{j_p}$ is > 0, by the definition of i_p. Hence, replacing the column e_{i_p} in the identity matrix $\mathbb{I}_m$ with d_{j_p} preserves the nonsingularity of the matrix. This implies that replacing the column $A_{B(i_p)}$ in the matrix A_B by A_{j_p} preserves the nonsingularity of A_B.

Algorithm 1 (*Simplex method in matrix form*)

```
1: Input:
2:   Given a basic feasible solution $\hat{x} = [\hat{x}_B, \hat{x}_N] = [\hat{x}_B, 0]$, where $B = \{B(1), \cdots, B(m)\}$ is the (ordered) index set of basic variables, and $N = \{1, \cdots, n\} \setminus B$
3: Main loop:
4:   Compute the reduced costs $\bar{c}_N^T = c_B^T A_B^{-1} A_N - c_N^T$, and write $\bar{b} = A_B^{-1} b$ (the values of $\hat{x}_B$).
5:   If $\bar{c}_N \geq 0$, then terminate, and return $\hat{x} = [\hat{x}_B, \hat{x}_N]$ as an optimal solution.
6:   Select $j$ such that the $j$th element of $\bar{c}_N$ is $< 0$, let $j_p = N(j)$, and compute the vector $d_{j_p} = A_B^{-1} A_{j_p}$.
7:   If $d_{j_p} \leq 0$, then terminate, and claim that the LP is unbounded.
8:   Choose which index to leave the index set $B$:
         $i_p = \arg\min_{i \in \{1, \cdots, m\}:\, d_{ij_p} > 0} \left\{ \frac{\bar{b}_i}{d_{ij_p}} \right\}.$
9:   Update the index set $B$ by replacing $B(i_p)$ by $j_p$, and update $N$ accordingly:
         $B(i_p) = j_p$ and $N \leftarrow \{1, \cdots, n\} \setminus B.$
10:  Update $A_B^{-1}$ and compute a new basic feasible solution $\hat{x} = [\hat{x}_B, \hat{x}_N] = [A_B^{-1} b, 0]$.
11: Repeat the Main loop
```

Example 4.1. Let us apply Algorithm 1 to solve the following canonical LP:

$$
\begin{cases}
\max\limits_{x} \ z = & x_1 & +x_2 & & & \\
\text{s.t.} \ \ x_1 & +2x_2 & +x_3 & & = 20 \\
\qquad 2x_1 & +x_2 & & +x_4 & = 20 \\
\qquad x_1 \geq 0, \ x_2 \geq 0, \ x_3 \geq 0, \ x_4 \geq 0. & & & &
\end{cases}
\tag{4.3}
$$

We can write

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \quad A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}, \quad b = \begin{pmatrix} 20 \\ 20 \end{pmatrix}, \quad \text{and} \quad c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

We have $x^0 = (0,0,20,20)^T$ as a basic feasible solution. In this case, $B = \{3,4\}$ and $N = \{1,2\}$. We can write $x^0 = [x^0_B, x^0_B]$ with $x^0_B = (20,20)^T$ and $x^0_N = (0,0)^T$. We also have $A_B = \mathbb{I}_2$ and $A_N = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$.

Iteration 0: We start with $B = \{3,4\}$ and $N = \{1,2\}$, $x^0_B = (20,20)^T$ and $x^0_N = (0,0)^T$, and $A_B = \mathbb{I}_2$.

1. We compute $\bar{c}_N^T = c_B^T A_B^{-1} A_N - c_N^T = c_B^T A_N - c_N^T = -c_N^T = (-1,-1)$, and $\bar{b} = A_B^{-1} b = b = \begin{pmatrix} 20 \\ 20 \end{pmatrix}$.
2. The first element of $\bar{c}_N$ is $-1 < 0$, we can let $j_p = N(1)$. The variable x_1 is an **entering variable**, and the pivot column is $j_p = 1$.
3. We compute $d_1 = A_B^{-1} A_1 = A_1 = (1,2)^T$.
4. Since there exists $i \in B = \{3,4\}$ such that $d_{i1} > 0$, we compare the two ratios $\left\{\frac{20}{1}, \frac{20}{2}\right\}$ and find the second to be smaller. This means $i_p = 2$. Because $B(2) = 4$, the **leaving variable** is x_4, and the pivot row is the second row.
5. Finally, we form a new B and new N as $B = \{3,1\}$ and $N = \{4,2\}$. As an **optional** step, we can also reorder indices in B and N so that they are in the increasing order, to obtain $B = \{1,3\}$ and $N = \{2,4\}$.

Iteration 1: We have $A_B = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$, $A_N = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}$. (Note: if we choose not to reorder indices in B and N at the end of last iteration, then with $B = \{3,1\}$ and $N = \{4,2\}$ we will have $A_B = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$, $A_N = \begin{bmatrix} 0 & 2 \\ 1 & 1 \end{bmatrix}$. This apparent difference will not affect the basic feasible solution.) We find $A_B^{-1} = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix}$, and obtain a new basic feasible solution $x^1 = [x^1_B, x^1_N]$ such that $x^1_B = A_B^{-1} b = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix} \begin{pmatrix} 20 \\ 20 \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \end{pmatrix}$. More precisely, we have $x^1 = (10,0,10,0)^T$.

1. We compute $\bar{c}_N^T = c_B^T A_B^{-1} A_N - c_N^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$, and $\bar{b} = A_B^{-1} b = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix} \begin{pmatrix} 20 \\ 20 \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \end{pmatrix}$.
2. With the first element of $\bar{c}_N$ being $-\frac{1}{2} < 0$, we let $j_p = N(1) = 2$, and choose the variable x_2 as the entering **entering variable**.
3. We compute $d_2 = A_B^{-1} A_2 = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{3}{2} \end{pmatrix}$.
4. Since there exists $i \in B$ such that $d_{i2} > 0$, we compare the two $\left\{ \frac{10}{1/2}, \frac{10}{3/2} \right\}$ and find the second to be smaller, which gives $i_p = 2$ and $B(i_p) = 3$. The **leaving variable** is x_3.
5. Finally, we form a new B and a new N as $B = \{1,2\}$ and $N = \{3,4\}$, respectively.

Iteration 2: We have $A_B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$, $A_N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. We can compute $A_B^{-1} = \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix}$. Hence, we can find a new basic feasible soluion $x^2 = [x_B^2, x_N^2]$ with $x_N^2 = 0$ and $x_B^2 = A_B^{-1} b = \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix} \begin{pmatrix} 20 \\ 20 \end{pmatrix} = \begin{pmatrix} \frac{20}{3} \\ \frac{20}{3} \end{pmatrix}$.

1. We compute $\bar{c}_N^T = c_B^T A_B^{-1} A_N - c_N^T = \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix}^T = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} \end{pmatrix}$.
2. Since $\bar{c}_N \geq 0$, we conclude that x^2 is optimal. We can write it as $x^* = x^2 = [x_B^2, x_N^2] = \left(\frac{20}{3}, \frac{20}{3}, 0, 0 \right)^T$. The corresponding optimal value is $z^* = x_1^* + x_2^* = \frac{40}{3}$.

To show how does the simplex method geometrically work, we eliminate variables x_3 and x_4 and rewrite problem (4.3) as

$$\begin{cases} \max\limits_{x} \ z = & x_1 & +x_2 & & \\ \text{s.t.} \ \ x_1 & & +2x_2 & \leq & 20 \\ \quad\ \ 2x_1 & & +x_2 & \leq & 20 \\ \quad\ \ x_1 \geq 0, \ x_2 \geq 0. & & & & \end{cases}$$

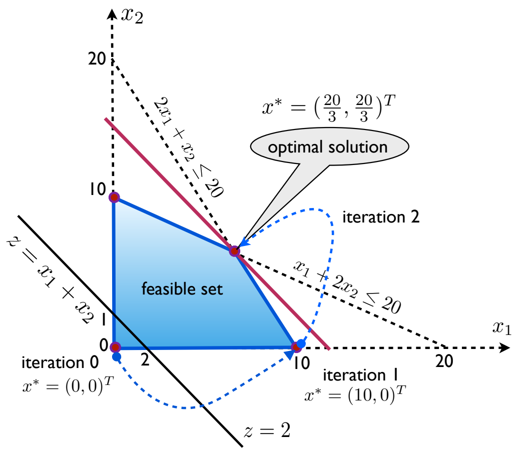


Fig. 4.1 An illustration of the simplex method for solving a simple LP

Figure 4.1 shows the feasible set and the objective line of this problem.

It also shows that we start from the vertex $x^0 = (0,0)^T$ at **iteration 0** with the objective value $z = c^Tx^0 = 0$. Then, at **iteration 1**, we move to the next vertex $x^1 = (10,0)^T$ with the objective value $z = c^Tx^1 = 10$. At **iteration 2**, we move to the vertex $x^* = x^2 = (20/3, 20/3)^T$, which is optimal, and the optimal value is $z^* = 40/3$. $\square$

4.2 The simplex method with simplex tableaux

We consider the following canonical linear program:

$$
\begin{cases}
\max\limits_{x} & z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{s.t.} & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\
& \qquad \vdots \qquad\qquad\qquad \vdots \quad \vdots \\
& a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i, \\
& \qquad \vdots \qquad\qquad\qquad \vdots \quad \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m, \\
& x_j \ge 0, \qquad j = 1, 2, \cdots, n.
\end{cases}
$$

In order to implement the simplex method, we can use simplex tableaux to store data and operate the iterations. At each iteration, we move from one simplex tableau to another by carrying out elementary row operations. We now describe this method in details.

4.2.1 Simplex tabeaux

The initial simplex tableau is given as in Table 4.1. This table has $m+1$ rows

Table 4.1 The initial simplex tableau for the simplex method

z	x_1	$\cdots$	x_n	RHS	Basic var	Ratio test
1	$-c_1$	$\cdots$	$-c_n$	0	$z=0$	
0	a_{11}	$\cdots$	a_{1n}	b_1	$x_{B(1)} = b_1$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
0	a_{i1}	$\cdots$	a_{in}	b_i	$x_{B(i)} = b_i$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
0	a_{m1}	$\cdots$	a_{mn}	b_m	$x_{B(m)} = b_m$	

max z; $x \geq 0$

excluding the title row. The first column consists of coefficients for the objective z, the next n columns include coefficients for variables x_1 to x_n, and the RHS column shows the right-hand side vector b. Moreover:

- The first row in the data section of Table 4.1 represents the equation $z - c_1x_1 - \cdots - c_nx_n = 0$. By convention this row is called "row 0".
- The remaining m rows represent the m equality constraints, and are called "row 1" $\cdots$, up to "row m".
- The "Basic var" column gives information on the natural basic feasible solution associated with the canonical LP, where $x_{B(i)}$ is the variable isolated by the ith equation.
- The last column will be used for the ratio test.
- Entries in row 0 of Table 4.1 are called the **reduced costs**. For example the reduced costs of z, x_1 are x_n are 1, $-c_1$ and $-c_n$ respectively in this tableau. (This definition of reduced costs only applies to simplex tableaus for a maximization LP. If the simplex tableau shows a minimization problem, then the reduced costs are the negative entries of row 0. In this course, we only work with simplex tableaus for maximization problems. The name "reduced cost" was chosen for historical reasons and is not particularly useful in understanding the general method.)

4.2.2 Simplex iterations

Starting from the initial simplex tableau, Table 4.1, one can apply the simplex method. Each iteration of this simplex method for solving a canonical LP contains the following steps:

Step 1. (**Choose the pivot column.**) Choose **one** nonbasic variable with a negative (< 0) **reduced cost** to be the **entering variable**. The column under the entering variable is the **pivot column**. If there are more than one negative reduced costs, then we can pick any one of them.

If no entering variable can be selected, we **termninate** the algorithm. The current BFS is an **optimal solution**, and the current simplex tableau is called an **optimal tableau**.

Step 2. (**Choose the pivot row.**) Excluding row 0, compare the ratios

$$\frac{\text{RHS}}{\text{entry in the pivot column}}$$

for each row with a positive (> 0) entry in the pivot column. The row with the minimal ratio is the **pivot row**. If more than one row gives the minimal ratio, then we can pick any of such rows to be the pivot row. The current basic variable in this row is the **leaving variable**. The intersection of the pivot column and the pivot row is the **pivot element**.

If no leaving variable can be selected, we **terminate** the algorithm. The LP is **unbounded**.

Step 3. (**Elementary row operations.**) Multiply the pivot row by the inverse of the pivot element, so that the pivot element becomes one; add a multiple of the pivot row to each other row, so that all other entries of the pivot column become zeros.

Step 4. (**Update the column about basic variables.**) In the pivot row, the basic variable is now the entering variable (it was the leaving variable before this iteration). The basic variables of all other rows remain the same. The values of the basic variables are given by the RHS constants of corresponding rows (which change in different tableaux).

4.2.3 Examples

Let us apply the simplex method in simplex tableaux to solve an LP.

Example 4.2. Consider the following linear program:

$$\begin{cases} \min\limits_{x} & -x_1 - x_2 - x_3 \\ \text{s.t} & -x_1 + x_2 - x_3 \leq 2, \\ & x_1 - x_2 - x_3 \quad \leq 3, \\ & -x_1 - x_2 + x_3 \leq 1, \\ & x_1 + x_2 + x_3 \quad \leq 4, \\ & x_1, x_2, x_3 \qquad \geq 0. \end{cases}$$

We first convert this problem into the maximization problem as

$$\begin{cases} \max\limits_{x} & z = x_1 + x_2 + x_3 \\ \text{s.t} & \qquad -x_1 + x_2 - x_3 \leq 2, \\ & \qquad x_1 - x_2 - x_3 \quad \leq 3, \\ & \qquad -x_1 - x_2 + x_3 \leq 1, \\ & \qquad x_1 + x_2 + x_3 \quad \leq 4, \\ & \qquad x_1, x_2, x_3 \qquad \geq 0. \end{cases}$$

We then add slack variables x_4, x_5, x_6, x_7 to convert the latter problem into standard form:

$$\begin{cases} \max\limits_{x} & z = x_1 + x_2 + x_3 \\ \text{s.t} & \qquad -x_1 + x_2 - x_3 + x_4 \qquad\qquad\qquad = 2, \\ & \qquad x_1 - x_2 - x_3 \qquad + x_5 \qquad\qquad = 3, \\ & \qquad -x_1 - x_2 + x_3 \qquad\qquad + x_6 \qquad = 1, \\ & \qquad x_1 + x_2 + x_3 \qquad\qquad\qquad + x_7 = 4, \\ & \qquad x_1, x_2, x_3, x_4, x_5, x_6, x_7 \qquad\quad \geq 0. \end{cases}$$

This problem has $m = 3$ constraints and $n = 7$ variables. Moreover, it is easy to check that the last LP is canonical.

Let us write down the initial simplex tableau as follows:

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	-1	-1	-1	0	0	0	0	0	$z = 0$	
0	-1	1	-1	1	0	0	0	2	$x_4 = 2$	
0	1	-1	-1	0	1	0	0	3	$x_5 = 3$	
0	-1	-1	1	0	0	1	0	1	$x_6 = 1$	
0	1	1	1	0	0	0	1	4	$x_7 = 4$	

max z; $x \geq 0$

Iteration 0: Now, let us start the simplex method from the initial tableau. Since the reduced cost of x_3 is $-1 < 0$, we can choose $j_p = 3$ to be a pivot column. We can of course choose $j_p = 1$ or $j_p = 2$ as well.

z	x_1	x_2	▼ x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	-1	-1	-1	0	0	0	0	0	$z = 0$	
0	-1	1	-1	1	0	0	0	2	$x_4 = 2$	N/A
0	1	-1	-1	0	1	0	0	3	$x_5 = 3$	N/A
0	-1	-1	$\boxed{1}$	0	0	1	0	1	$x_6 = 1$	$1/1 = 1$
0	1	1	1	0	0	0	1	4	$x_7 = 4$	$4/1 = 4$

max z; $x \geq 0$

1. Entering variable: x_3 (again, it is fine to choose either x_1 or x_2).
2. Ratio test: There are two rows $i = 3$ and $i = 4$ in the tableau with positive entries in the pivot column. For these two rows, we compute the ratio between the RHS and the entry in the pivot column. Row 3 wins, and x_6 is the leaving variable.
3. The pivot element is at the location $(i_p, j_p) = (3, 3)$, with a value of 1.
4. Conduct the following elementary row operations:
   - Add row 3 to row 0, row 3 to row 1, and row 3 to row 2.
   - Subtract row 3 from row 4.

We obtain the second tableau below.

z	x_1 ▼	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio
1	-2	-2	0	0	0	1	0	1	$z = 1$	
0	-2	0	0	1	0	1	0	3	$x_4 = 3$	N/A
0	0	-2	0	0	1	1	0	4	$x_5 = 4$	N/A
0	-1	-1	1	0	0	1	0	1	$x_3 = 1$	N/A
0	$\boxed{2}$	2	0	0	0	-1	1	3	$x_7 = 3$	$3/2 = 1.5$

max z; $x \geq 0$

Iteration 1: We repeat the simplex iteration as follows.

1. Entering variable: Since the reduced cost of x_1 is $-2 < 0$, we choose x_1 as an entering variable (it is fine to choose x_2).
2. Ratio test: row 4 is the only row with positive entry in the pivot column, so we choose it as the pivot row and x_7 as the leaving variable.
3. Conduct elementary row operations:

   - Divide row 4 by 2.
   - Add twice row 4 to row 0, twice row 4 to row 1, and row 4 to row 3.
   - Keep row 2 unchanged.

We obtain the third tableau below.

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var
1	0	0	0	0	0	0	1	4	$z = 4$
0	0	2	0	1	0	0	1	6	$x_4 = 6$
0	0	-2	0	0	1	1	0	4	$x_5 = 4$
0	0	0	1	0	0	0.5	0.5	2.5	$x_3 = 2.5$
0	1	1	0	0	0	-0.5	0.5	1.5	$x_1 = 1.5$

max z; $x \geq 0$

We see that the first $n+1$ columns of row 0 do not include any negative values. Hence, there is no entering variable to select, and we can conclude that the current BFS

$$x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)^T = (1.5, 0, 2.5, 6, 4, 0, 0)^T$$

is optimal, and the objective value attained by this BFS, $z = 4$, is the optimal value. The algorithm is terminated with the above tableau being an optimal tableau. By removing the slack variables, we obtain $\hat{x}^* = (1.5, 0, 2.5)^T$ as an optimal solution of the original problem, with the optimal value $\hat{z} = -4$.

4.2.4 Features of simplex tableaux

With a closer look at the simplex tableaux, we observe the following features:

- The data section of any simplex tableau contains columns of the form

$$\begin{pmatrix}1\\0\\0\\\vdots\\0\end{pmatrix}, \quad \begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix}, \quad \cdots, \quad \begin{pmatrix}0\\0\\0\\\vdots\\1\end{pmatrix},$$

  where the first column is the column under z, and the others are columns associated with the current basic variables. The basic variable of each row is the variable isolated by this row (having a coefficient of 1 in this row and a coefficient of 0 in any other row), and its value is exactly the current RHS of this row.
- All the entries (except the one in row 0) of the RHS column of any simplex tableau are nonnegative numbers, which are values of the basic variables.
- Each simplex tableau represents a transformation of the original LP. All these LPs are equivalent to each other, and they are all in canonical form, with different isolated variables.

4.2.5 Uniqueness of the optimal solution

An LP may have more than one optimal solution. To illustrate this, we consider LPs of two variables. Figure 4.2 shows two different LPs. The goal in both cases are to maximize the objective functions. By shifting the iso-profit line along the normal vector c until it would leave the feasible set, we obtain the last iso-profit line. The intersection between the last iso-proft line and the feasible set gives the optimal solutions. In both cases, that intersection consists of an entire edge of the feasible set, so any point on that edge is an optimal solution of the LP. The two cases are different in the boundedness of the set of the optimal solutions:

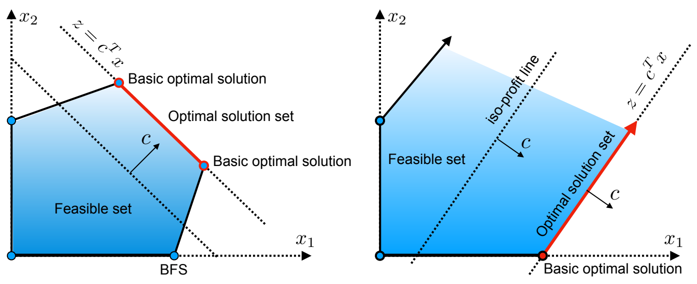


Fig. 4.2 Multiple optimal solutions in LPs. Left: **Case 1:** Bounded solution set, Right: **Case 2:** Unbounded solution set

- **Case 1:** The optimal edge is of finite length. In other words, the set of optimal solutions is bounded.
- **Case 2:** The optimal edge is of infinite length. In other words, the set of optimal solutions is unbounded.

Next, we consider how to treat these two situations in the simplex method with simplex tableaux.

Consider the optimal tableau obtained from applying the simplex method to an LP. All reduced costs in such a tableau are nonnegative.

- If all reduced costs associated with the nonbasic variables are strictly positive, then the LP has a unique optimal solution.
- If there exists a nonbasic variable x_j with zero reduced cost, then the LP can have multiple solutions. In this case, we can increase this variable from zero to a positive number without decreasing the objective value, to obtain a new optimal solution. This can be done by performing an extra simplex iteration . In this simplex iteration, x_j is selected as the **entering variable**. In trying to determine the **leaving variable**, there are two situations:
  - **Case 1:** If there exists a positive entry in the j-th column, then we can choose the leaving variable by using the ratio test rule. Hence, by performing an extra simplex iteration on the optimal simplex tableau, we can move from one basic optimal solution to another one.
  - **Cases 2:** If all the entries of the j-th column are nonpositive, then we can increase the value of x_j from zero to any positive number without making other basic variables to be negative. Hence, we obtain an unbounded

halfline starting from the current basic optimal solution, with each point on this halfline being an optimal solution.

Note: if x^* and $\hat{x}^*$ are two optimal solutions of an LP, then for any $\alpha \in (0,1)$, the point $x = \alpha x^* + (1-\alpha)\hat{x}^*$ is also an optimal solution of this problem.

Let us consider three examples that illustrate three situations described above.

Example 4.3. Suppose that we obtain the following optimal simplex tableau when solving a maximization LP of 4 variables and 2 equality constraints:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	0	0	2	3	10	$z = 10$
0	1	0	3	2	4	$x_1 = 4$
0	0	1	1	1	3	$x_2 = 3$

The BFS $x^* = (4,3,0,0)^T$ is the unique optimal solution, since the reduced costs of both nonbasic variables x_3 and x_4 are strictly positive.

Example 4.4. Suppose that we obtain the following optimal simplex tableau when solving a maximization LP of 4 variables and 2 equality constraints:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	2	0	0	0	2	$z = 2$
0	1	0	1	1	2	$x_4 = 2$
0	3	1	-2	0	3	$x_2 = 3$

The BFS $x^* = (0,3,0,2)^T$ is optimal, but there can be other optimal solutions, because the reduced cost of the nonbasic variable x_3 is zero. To find other optimal solutions, we select x_3 as the entering variable, and conduct an extra simplex iteration to find a different basic optimal solution $\hat{x}^* = (0,7,2,0)^T$ (more details about how to conduct such an extra iteration are provided in Example 4.6 below). Then, any vector of the format $x = \alpha x^* + (1-\alpha)\hat{x}^* = \alpha(0,3,0,2)^T + (1-\alpha)(0,7,2,0)^T$ with some $0 \leq \alpha \leq 1$ is an optimal solution.

Alternatively, one can let $x_3 = t \geq 0$, fix $x_1 = 0$, and let x_2 and x_4 be expressed as functions of t as $x_2 = 3 + 2t$ and $x_4 = 2 - t$ respectively. The non-negativity requirements on x_2, x_3 and x_4 restrict t to the range $[0,2]$. Any vector $x^* = (0, 3 + 2t, t, 2-t)^T$ is an optimal solution of the LP for any $t \in [0,2]$.

Example 4.5. Suppose that we obtain the following optimal simplex tableau when solving a maximization LP of 4 variables and 2 equality constraints:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	2	0	0	0	2	$z=2$
0	1	0	-1	1	2	$x_4=2$
0	3	1	-2	0	3	$x_2=3$

The BFS $x^*=(0,3,0,2)^T$ is optimal, but there can be other optimal solutions because the reduced cost of the nonbasic variable x_3 is zero. To find other optimal solutions, let $x_3=t$, and fix $x_1=0$, and let x_2 and x_4 be expressed as functions of t as $3+2t$ and $2+t$ respectively. Here, the nonnegativity requirements on x_2 and x_4 are automatically satisfied for any $t\geq 0$, so $x=(0,3+2t,t,2+t)$ is an optimal solution for any $t\geq 0$. In this example, if one would select x_3 as the entering variable in the extra simplex iteration, then the ratio test would be void and one would not be able to find a different basic optimal solution.

Example 4.6. In Example 4.2, we note that the objective function of the LP in the optimal simplex tableau is

$$z=x_7-4,$$

which holds for each feasible solution. If we increase the nonbasic variable x_6 from zero to a positive number while keeping the other nonbasic variables x_2 and x_7 at zero, z will remain to be 4. Hence, for the extra pivot, we can choose

1. Entering variable: x_6
2. Ratio test: Row 2 wins, and x_5 is the leaving variable.

By elementary row operations we obtain the following tableau:

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	0	0	0	0	0	0	1	4	$z=4$	
0	0	2	0	1	0	0	1	6	$x_4=6$	
0	0	-2	0	0	1	1	0	4	$x_6=4$	
0	0	1	1	0	-0.5	0	0.5	0.5	$x_3=0.5$	
0	1	0	0	0	0.5	0	0.5	3.5	$x_1=3.5$	

max z; $x\geq 0$

Again no entering variable to select: the above tableau is another optimal tableau. The current BFS is

$$(x_1,x_2,x_3,x_4,x_5,x_6,x_7)^T = (3.5,0,0.5,6,0,4,0)^T,$$

which is an alternative optimal solution. The optimal value at this optimal solution is still $z = 4$.

We have found 2 **basic optimal solutions** so far. In fact, there are 4 more **basic optimal solutions**. We can find them by continuing extra pivots from the present optimal simplex tableau. We may also find them by selecting different entering variables in the process of the algorithm. In total, there are 6 **basic optimal solutions**, and infinitely many optimal solutions!

4.2.6 Unbounded linear programs

In the previous chapter, we have illustrated the situation in which an LP is unbounded. A maximization (minimization) LP is unbounded, when its feasible set is unbounded and there exists a halfline $\{x+td \mid t \geq 0\}$ entirely included in the feasible set, with x being the starting point and d being the direction, such that the objective value of $x+td$ goes to ∞ $(-\infty)$ as t increases to ∞. Such a direction d is called the **direction of unboundedness** of the LP. While an unbounded LP must have an unbounded feasible set, an LP with an unbounded feasible set may have unique or multiple optimal solutions, and the set of its optimal solutions can be bounded or unbounded.

Below, we give details on how to identify unbounded LPs from simplex tableaux. Consider a simplex tableau obtained by solving a maximization LP. If the tableau contains a column that has a negative entry in row 0 and nonpositive entries in the all remaining rows, then the LP is unbounded. In other words, if we can select an entering variable but not a leaving variable, then the LP is unbounded. With such a tableau, one can find feasible solutions with arbitrarily large objective values, by letting the entering variable take any nonnegative value, fixing other nonbasic variables at zero, and letting basic variables be expressed as functions of the entering variable.

Example 4.7. Suppose that we are solving a maximization LP of 4 variables and 2 equality constraints, and obtain the following simplex tableau:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	-3	-2	0	0	0	$z = 0$
0	1	-5	1	0	3	$x_3 = 3$
0	2	0	0	1	4	$x_4 = 4$

The LP is unbounded, because the nonbasic variable x_2 has a negative reduced cost, and all entries in the column under x_2 are nonpositive.

To find the direction of unboundedness, let $x_2 = t \geq 0$, and fix the other nonbasic variable $x_1 = 0$, and let x_3 and x_4 be expressed as functions of t as $x_3 = 3+5t$ and $x_4 = 4$, respectively. In summary, $x = (0,t,3+5t,4)^T$ is a feasible solution for any $t \geq 0$, with the objective value $z = 2t$. For example, with $t = 1000$, we have a feasible solution $(0,1000,5003,4)^T$ that gives $z = 2000$. Furthermore, by a decomposition, we can write any feasible solution x as

$$\begin{pmatrix} 0 \\ t \\ 3+5t \\ 4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 3 \\ 4 \end{pmatrix} + t \begin{pmatrix} 0 \\ 1 \\ 5 \\ 0 \end{pmatrix}.$$

Hence, we have found a starting point $x^0 = (0,0,3,4)^T$ and a direction $d = (0,1,5,0)^T$, so that the half-line $x = x^0 + td$ from the starting point x^0 along the direction d is entirely contained in the feasible set and the objective value $z = 2t$ increases along the direction d as $t \to +\infty$.

4.2.7 Finite termination of the simplex method

The following theorem shows that the simplex method is guaranteed to terminate in finitely many steps given that all basic feasible solutions are nondegenerate.

Theorem 4.1. *Consider a max LP in the canonical form. Suppose all the basic feasible solutions of the feasible set are nondegenerate, then the simplex method will stop in finitely many iterations.*

Proof. First, nondegeneracy means that the RHS entries in any simplex tableau to be encountered in the simplex method are strictly positive, excluding row 0.

Second, in each iteration of the simplex method, the pivot element is strictly positive, the reduced cost of the entering variable is strictly negative, and a positive multiple of the pivot row is added to row 0 to cancel out the reduced cost. As a result, the z value will be strictly increased in each iteration. Thus, no simplex tableau appears twice in the simplex method.

Without the nondegeneracy assumption, some anti-cycling rules are needed to guarantee finite termination. One example is Bland's Rule:

- Among all qualified nonbasic variables, choose the nonbasic var with the smallest index as the entering var.
- In case of a tie in the ratio test, choose the row whose current basic var has the smallest index as the pivot row,

It can be shown that the simplex method implemented using Bland's rule is guaranteed to terminate in finitely many iterations, without the above nondegeneracy assumption. The proof is rather complicated and is omitted. See [3] for a more detailed discussion.

4.3 The two-phase simplex algorithm

In real-world problems, we often obtain a general linear program instead of a canonical LP. To solve such a general LP, we need to find an initial basic feasible solution to apply the simplex method as described in the preceding sections to solve the LP by starting from this basic feasible solution. However, for a general LP, finding a basic feasible solution is almost as hard as finding an optimal solution given a initial basic feasible solution. In this section, we describe a two-phase simplex method to solve a general LP. First, we convert a general LP into a standard LP, which can always be done. If the resulting LP is canonical, we can easily obtain a natural BFS and directly apply the simplex method described in Section 4.2. Otherwise, we apply the two-phase simplex method as described below, to solve the standard LP. The two phases of this method are discussed in the following two subsections respectively.

4.3.1 Phase 1: Solving the auxiliary LP

Let us consider the following standard LP:

$$\begin{cases} \max\limits_{x} & z = c^T x \\ \text{s.t.} & \quad Ax \ = b, \\ & \quad\ \ x \ \geq 0. \end{cases} \tag{4.4}$$

Here, without loss of generality, we assume that the right-hand side vector b is nonnegative, i.e., $b \geq 0$. (If $b_i < 0$ for some index i, then we can simply multiply the i-th constraint by -1 on both sides to obtain a positive right-hand side.) We also assume that rows of A are linearly independent.

Step 1: Our next step is to form an auxiliary LP for (4.4) as

$$\begin{cases} \max\limits_{x,y} & t = -y_1 - y_2 - \cdots - y_m \\ \text{s.t.} & Ax + y = b, \\ & x \geq 0, \\ & y \geq 0. \end{cases} \tag{4.5}$$

Here, we add an **artificial variable** y_i to the i-th equality constraint of the original LP (4.4). We ignore the objective function of (4.4), and define a new objective function to be maximizing the sum of additive inverses of all artificial variables y, i.e., $t = -\sum_{i=1}^{n} y_i$. The resulting LP has $n+m$ variables and m constraints, and is called the **Phase I LP** or the **auxiliary LP**.

Fact 4.3.1 *The auxiliary LP* (4.5) *always has an optimal solution.*

Indeed, the feasible set of this problem is nonempty since $(x,y) = (0,b)$ is a feasible solution. Moreover, the objective function $t = -\sum_{i=1}^{m} y_i \leq 0$ for all feasible solutions $y \geq 0$, which shows that t is bounded from above. It follows that (4.5) must have an optimal solution.

Step 2: To transform (4.5) into canonical form, we note from the constraints of (4.5) that $y = b - Ax$. Hence, we can rewrite the objective of (4.5) as

$$t = -\sum_{i=1}^{m} y_i = -\sum_{i=1}^{m} b_i + \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}x_j.$$

If we denote by $1 = (1,1,\cdots,1)^T$ the vector of all ones in $\mathbb{R}^m$, then $t = (A^T 1)^T x - b^T 1 = \bar{c}^T x - b^T 1$, where $\bar{c} = A^T 1$. Hence, the auxiliary LP (4.5) is equivalent to the following canonical LP:

$$\begin{cases} \max\limits_{x,y} & t = \bar{c}^T x - b^T 1 \\ \text{s.t.} & Ax + y = b, \\ & x \geq 0, \\ & y \geq 0, \end{cases} \tag{4.6}$$

in which y_i is the variable isolated by the ith equation. Accordingly, $(x, y) = (0, b)$ is the basic feasible solution naturally associated with (4.6).

Fact 4.3.2 *The standard LP* (4.4) *has a feasible solution if and only if the optimal value t^* of the auxiliary problem* (4.5) *is zero.*

Indeed, if (4.4) has a feasible solution $\bar{x}$, then $A\bar{x} = b$ and $\bar{x} \geq 0$. Then $(x, y) = (\bar{x}, 0)$ is a feasible solution for (4.5) with $t = 0$. On the other hand, any feasible solution (x, y) of (4.5) must have $y \geq 0$ and therefore $t \leq 0$. Hence, the optimal value of (4.5) is $t = 0$.

Conversely, let (x^*, y^*) be an optimal solution of (4.5) with $t^* = 0$. Then, since $y^* \geq 0$, the fact $t^* = -\sum_{i=1}^{m} y_i^* = 0$ implies $y^* = 0$. This means $Ax^* = b$, which together with $x^* \geq 0$ implies that x^* is a feasible solution to (4.4).

Remark 4.1. If the ith equality constraint of the standard LP (4.4) already isolates a variable, then we do not have to add an artificial variable y_i into this constraint. The objective function of the auxiliary problem (4.5) is always to maximize the sum of the additive inverses of all artificial variables. (See the example below for such a situation.)

Step 3: Our final step is to apply the simplex method to solve the canonical auxiliary LP (4.6), to find a basic optimal solution (x^*, y^*) with the optimal value $t^* \leq 0$. Depending on the value of t^*, there are the following two cases to consider:

Case 1: The optimal value t^* of (4.6) is negative ($t^* < 0$). In this case, we conclude that the standard LP (4.4) is infeasible. Hence, the original general LP is also infeasible.

Case 2: The optimal value $t^* = 0$. In this case, $y^* = 0$ and the standard LP (4.4) is feasible. Since $(x^*, 0)$ is a BFS of (4.6), x^* is a BFS of (4.4). (This can be seen, e.g., from the fact that basic feasible solutions are exactly extreme points.) We will then apply the simplex method to solve (4.4) using this BFS as the initial point. This phase is called Phase 2, as presented below.

4.3.2 Phase 2: Performing the simplex method on the standard LP

If we directly apply the simplex method to solve the standard LP (4.4) using the BFS x^* obtained at the end of Phase 1, then we will need to initialize a simplex tableau so that all the basic variables defining x^* are isolated variables in the tableau. In other words, we will need to identity the matrix A_B that consists of columns for the basic variables defining x^*, and then multiply both sides of the equation $Ax = b$ by A_B^{-1} to transform the LP from (4.4) to the canonical form (4.2). However, by exploiting the optimal simplex tableau from Phase 1, we can construct the initial simplex tableau for the original LP (4.4) without directly computing A_B^{-1}. Below we explain how to proceed.

Examining the optimal simplex tableau at Phase 1, we can encounter the following two cases:

Case 1: None of the artificial variables is a basic variable in the optimal tableau of Phase 1. In this case, we eliminate the artificial variables and the columns under these variables from the tableau, and express the objective function of the original LP (4.4) using the remaining nonbasic variables. This yields a canonical form of the original LP. From there, one can apply the simplex method to solve (4.4).

More precisely, after removing all artificial variables from the optimal tableau of Phase 1, we can write all basic variables x_B as $x_B = \bar{b} - \bar{A}_N x_N$, where $\bar{A}_N$ is the matrix formed from the nonbasic columns in that tableau, and $\bar{b}$ is the right-hand side vector in that tableau. Substituting x_B into the objective function

$$z = c^T x = c_B^T(\bar{b} - \bar{A}_N x_N) + c_N^T x_N = c_B^T \bar{b} + (c_N - \bar{A}_N^T c_B)^T x_N,$$

we obtain the reduced costs for nonbasic variables in the initial simplex tableau for (4.4) as $\bar{c}_N = \bar{A}_N^T c_B - c_N$.

Case 2: If there exists an artificial variable y_i that is a basic variable in the optimal tableau of Phase 1, then we can change this basic variable to an non-artificial variable by elementary row operations. More specifically, suppose that y_i is the basic variable in row l. Because the optimal value is $t^* = 0$, all artificial variables are zero in this BFS, which implies that the right hand side of row l is zero. By the linear independency of rows of A, there is at least one nonzero entry of row l under some non-artificial variable. Suppose the entry in row l under a non-artificial variable x_j is nonzero. Then, x_j must be a nonbasic variable in this tableau. Choose this entry as the pivot element, and conduct elementary row operations so that x_j becomes the basic variable in row l (i.e., use ERO's to make the entry in row l under x_j to become 1, and other entries in the x_j column to become zeros). During those ERO's, the right hand side coefficients in the tableau do not change because the right hand side of row l is zero. We repeat this procedure until we remove all artificial variables from the basis.

4.3.3 Examples

Let us consider some examples to illustrate how the two-phase simplex method works.

Example 4.8. Consider the following standard LP:

$$\begin{cases} \max\limits_{x} \ z = -2x_1 - 4x_2 + 2x_3 \\ \text{s.t} \quad x_1 - 2x_2 + x_3 = 27, \\ \quad 2x_1 + x_2 + 2x_3 = 50, \\ \quad x_1 - x_2 - x_3 + x_4 = 18, \\ \quad x_1, \cdots, x_4 \geq 0. \end{cases}$$

Since the last constraint isolates the variable x_4, we do not need an artificial variable for this constraint. We can write the auxiliary LP for Phase 1 by adding two artificial variables y_1 and y_2 as:

$$\begin{cases} \max\limits_{x,y} \ t = 3x_1 - x_2 + 3x_3 - 77 \\ \text{s.t} \quad x_1 - 2x_2 + x_3 + y_1 = 27, \\ \quad 2x_1 + x_2 + 2x_3 + y_2 = 50, \\ \quad x_1 - x_2 - x_3 + x_4 = 18, \\ \quad x_1, \cdots, x_4, y_1, y_2 \geq 0. \end{cases}$$

Here, the objective function $t = -y_1 - y_2$ is equivalently written as $t = -77 + 3x_1 - x_2 + 3x_3$, because summing up the two first constraints gives $-y_1 - y_2 = -77 + 3x_1 - x_2 + 3x_3$.

Phase 1: Now, we apply the simplex method to solve the auxiliary problem. The initial tableau is as follows.

▼

t	x_1	x_2	x_3	x_4	y_1	y_2	RHS	Basic var	Ratio test
1	-3	1	-3	0	0	0	-77	$t = -77$	
0	1	-2	1	0	1	0	27	$y_1 = 27$	$\frac{27}{1} = 27$
0	2	1	$\boxed{2}$	0	0	1	50	$y_2 = 50$	$\frac{50}{2} = 25$
0	1	-1	-1	1	0	0	18	$x_4 = 18$	N/A

Iteration 0: x_3 is the entering variable, and y_2 is the leaving variable.

▼

t	x_1	x_2	x_3	x_4	y_1	y_2	RHS	Basic var	Ratio test
1	0	5/2	0	3	0	3/2	-2	$t = -2$	
0	0	-5/2	0	0	1	-1/2	2	$y_1 = 2$	
0	1	1/2	1	0	0	1/2	25	$x_3 = 25$	
0	2	-1/2	0	1	0	1/2	43	$x_4 = 43$	

This simplex tableau is already optimal. Here, we have $t^* = -2 < 0$. We conclude that the original problem is infeasible.

Example 4.9. We consider the following standard LP:

$$\begin{cases} \max\limits_{x} \ z = -x_1 - x_2 - x_3 \\ \text{s.t} \qquad x_1 + 2x_2 + 3x_3 = 3, \\ \qquad -x_1 + 2x_2 + 6x_3 = 2, \\ \qquad 3x_3 + x_4 = 1, \\ \qquad x_1, \cdots, x_4 \geq 0. \end{cases} \tag{4.7}$$

Because x_4 has coefficient 1 in the third equation and zero coefficients in the other two equations, we do not need to add an artificial variable in the third equation. We only add two artificial variables x_5 and x_6 to obtain the following auxiliary LP:

$$\begin{cases} \max\limits_{x} \ t = -x_5 - x_6 \\ \text{s.t} \qquad x_1 + 2x_2 + 3x_3 + x_5 = 3, \\ \qquad -x_1 + 2x_2 + 6x_3 + x_6 = 2, \\ \qquad 3x_3 + x_4 = 1, \\ \qquad x_1, \cdots, x_6 \geq 0. \end{cases}$$

Using the first two constraints one can express x_5 and x_6 as functions of x_1, x_2 and x_3 as

$$x_5 = 3 - x_1 + 2x_2 + 3x_3 \quad \text{and} \quad x_6 = 2 + x_1 - 2x_2 - 6x_3.$$

Substitute these two expressions into t to transform the auxiliary LP into:

$$\begin{cases} \max_x \ t = \quad 4x_2 + 9x_3 - 5 \\ \text{s.t} \quad x_1 + 2x_2 + 3x_3 + x_5 = 3, \\ \quad -x_1 + 2x_2 + 6x_3 + x_6 = 2, \\ \quad 3x_3 + x_4 = 1, \\ \quad x_1, \cdots, x_6 \geq 0. \end{cases}$$

This problem is canonical. The initial tableau is as follows.

t	x_1	x_2	x_3	x_4	x_5	x_6	RHS	Basic var	Ratio test
1	0	-4	-9	0	0	0	-5	$t = -5$	
0	1	2	3	0	1	0	3	$x_5 = 3$	$\frac{3}{2}$
0	-1	2	6	0	0	1	2	$x_6 = 2$	$\frac{2}{2}$
0	0	0	3	1	0	0	1	$x_4 = 1$	

We apply the simplex iterations to operate on this simplex tableau.

Iteration 0: We can identify that x_2 is the entering variable and x_6 is the leaving variable.

t	x_1	x_2	x_3	x_4	x_5	x_6	RHS	Basic var	Ratio test
1	-2	0	3	0	0	2	-1	$t = -1$	
0	2	0	-3	0	1	-1	1	$x_5 = 1$	$\frac{1}{2}$
0	-1/2	1	3	0	0	1/2	1	$x_2 = 1$	
0	0	0	3	1	0	0	1	$x_4 = 1$	

Iteration 1: x_1 is the entering variable, and x_5 is the leaving variable.

t	x_1	x_2	x_3	x_4	x_5	x_6	RHS	Basic var	Ratio test
1	0	0	0	0	1	1	0	$t = 0$	
0	1	0	-3/2	0	1/2	-1/2	1/2	$x_1 = 1/2$	
0	0	1	9/4	0	1/4	1/4	5/4	$x_2 = 5/4$	
0	0	0	3	1	0	0	1	$x_4 = 1$	

The above simplex tableau is the optimal simplex tableau we obtain for the auxiliary LP. It shows that the optimal value of the auxiliary LP is $t^* = 0$. Moreover, in this optimal tableau, the basic variables are x_1, x_2, x_4, none of which is an artificial variable. This means that we are in Case 1. By removing columns under the artificial variables, we obtain a tableau for the original LP, where we leave row 0 undetermined for now.

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	*	*	*	*	*	$z = *$
0	1	0	-3/2	0	1/2	$x_1 = 1/2$
0	0	1	9/4	0	5/4	$x_2 = 5/4$
0	0	0	3	1	1	$x_4 = 1$

To complete row 0, we need to express the objective function z as a function of the nonbasic variables (because the reduced costs for basic variables have to be zero in order for the tableau to be a valid simplex tableau). To this end, we use information from rows 1 to 3 in the above tableau, which provides expressions of x_1, x_2 and x_4 as functions of x_3:

$$x_1 = 1/2 + 3/2x_3, \quad x_2 = 5/4 - 9/4x_3, \quad \text{and} \quad x_4 = 1 - 3x_3.$$

By substitution, we have

$$z = -x_1 - x_2 - x_3 = -1/4x_3 - 7/4.$$

We now have a complete simplex tableau for the original LP:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	0	0	1/4	0	-7/4	$z = -7/4$
0	1	0	-3/2	0	1/2	$x_1 = 1/2$
0	0	1	9/4	0	5/4	$x_2 = 5/4$
0	0	0	3	1	1	$x_4 = 1$

Luckily, this simplex tableau is already optimal since all reduced costs are nonnegative. We obtain the optimal solution of this LP as $x^* = (1/2, 5/4, 0, 1)^T$ and the optimal value as $z = -7/4$.

In general, at the end of Phase I, one will obtain an initial simplex tableau for the original LP (when the original LP is feasible) and will then need to continue iterations from there.

Example 4.10. Let us consider the following standard LP:

$$\begin{cases} \max_x \ z = -3x_1 + x_3 \\ \text{s.t} \quad x_1 + x_2 + x_3 + x_4 = 4, \\ -2x_1 + x_2 - x_3 = 1, \\ 3x_2 + x_3 + x_4 = 9, \\ x_1, \cdots, x_4 \geq 0. \end{cases}$$

With the same procedure as above, we can form the following auxiliary LP:

$$\begin{cases} \max_x \ t = -x_1 + 5x_2 + x_3 + 2x_4 - 14 \\ \text{s.t} \quad x_1 + x_2 + x_3 + x_4 + x_5 = 4, \\ -2x_1 + x_2 - x_3 + x_6 = 1, \\ 3x_2 + x_3 + x_4 + x_7 = 9, \\ x_1, \cdots, x_7 \geq 0. \end{cases}$$

Here, x_5, x_6 and x_7 are three artificial variables. The objective function is given by $t = -x_5 - x_6 - x_7$, which can be equivalently written as $t = -14 - x_1 + 5x_2 + x_3 + 2x_4$ by summing up the three constraints.

Phase 1: Using the simplex method to solve the auxiliary LP, we have the initial tableau as

t	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	1	-5	-1	-2	0	0	0	-14	$t = -14$	
0	1	1	1	1	1	0	0	4	$x_5 = 4$	$\frac{4}{1} = 4$
0	-2	$\boxed{1}$	-1	0	0	1	0	1	$x_6 = 1$	$\frac{1}{1} = 1$
0	0	3	1	1	0	0	1	9	$x_7 = 9$	$\frac{9}{3} = 3$

Iteration 0: x_2 is the entering variable, and x_6 is the leaving variable.

t	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	-9	0	-6	-2	5	0	0	-9	$t = -9$	
0	$\boxed{3}$	0	2	1	1	0	0	3	$x_5 = 3$	$\frac{3}{3} = 1$
0	-2	1	-1	0	0	1	0	1	$x_2 = 1$	N/A
0	6	0	4	1	0	-3	1	6	$x_7 = 6$	$\frac{6}{6} = 1$

Iteration 1: x_1 is the entering variable, and x_5 is the leaving variable.

t	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	0	0	0	1	8	0	0	0	$t = 0$	
0	1	0	2/3	1/3	1/3	0	0	1	$x_1 = 1$	
0	0	1	1/3	2/3	2/3	1	0	3	$x_2 = 3$	
0	0	0	0	-1	-2	-3	1	0	$x_7 = 0$	

This tableau is optimal for the auxiliary LP, and it shows that the optimal value of the auxiliary LP is 0. This means that the original LP is feasible. However, in the basic optimal solution shown in this tableau, x_7, an artificial variable, is in the basis. We replace x_7 with x_4 as the basic variable in row 3, by performing the following extra step. The elementary row operations in this step do not change the right hand sides because the right hand side of row 3 (which is also the value of x_7) is zero.

t	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic var	Ratio test
1	0	0	0	0	6	-3	1	0	$t = 0$	
0	1	0	2/3	0	-1/3	-1	1/3	1	$x_1 = 1$	
0	0	1	1/3	0	-2/3	-1	2/3	3	$x_2 = 3$	
0	0	0	0	1	2	3	-1	0	$x_4 = 0$	

Now, we can remove all artificial variables x_5, x_6 and x_7 to obtain a basic feasible solution $x = (1,3,0)^T$ to start Phase II.

Phase 2: The initial simplex tableau of Phase II is as below, where the objective is $z = -3x_1 + x_3 = -3(1 - \frac{2}{3}x_3) + x_3 = -3 + 3x_3$.

z	x_1	x_2	x_3	x_4	RHS	Basic var	Ratio test
1	0	0	-3	0	0	$z = -3$	
0	1	0	$\boxed{2/3}$	0	1	$x_1 = 1$	$\frac{1}{2/3} = 3/2$
0	0	1	1/3	0	3	$x_2 = 3$	$\frac{3}{1/3} = 9$
0	0	0	0	1	0	$x_4 = 0$	N/A

Iteration 1: The entering variable is x_3 and the leaving variable is x_1.

z	x_1	x_2	x_3	x_4	RHS	Basic var	Ratio test
1	9/2	0	0	0	0	$z = 3/2$	
0	-3/2	0	1	0	3/2	$x_3 = 3/2$	
0	-1/2	1	0	0	5/2	$x_2 = 5/2$	
0	0	0	0	1	0	$x_4 = 0$	

Luckily, this simplex tableau is already optimal. The optimal solution of the original problem is $x^* = (0, 5/2, 3/2, 0)^T$, and the optimal value is $z^* = 3/2$.

4.4 Exercises

Exercise 4.1. Solve the following problems using the simplex method in tableau form. If one is not canonical, transform it into the canonical form first. Describe your steps in detail and give explicit simplex tableau for each iteration.

$$(a)\ \begin{cases} \max\limits_{x}\ z = x_1 + x_2 \\ \text{s.t.}\quad 2x_1 + x_2 + x_3 = 5 \\ \qquad\ \ x_1 + 2x_2 + x_4 = 6 \\ \qquad\ \ x_1, x_2, x_3, x_4 \geq 0. \end{cases} \qquad (b)\ \begin{cases} \min\limits_{x}\ z = 2x_1 - 3x_2 + x_3 \\ \text{s.t.}\quad x_1 + x_2 + x_3 \leq 10 \\ \qquad\ \ x_1 + 2x_2 - x_3 \leq 8 \\ \qquad\ \ x_1, x_2, x_3 \geq 0. \end{cases}$$

Exercise 4.2. The bartender of your local pub asks you to assist him in finding the combination of mixed drinks to maximize his revenue. He has the following bottles available:

- 1 quart (32 oz.) Old Cambridge (a fine whiskey)
- 1 quart Joy Juice (another fine whiskey)
- 1 quart Ma's Wicked Vermouth
- 2 quarts Gil-boy's Gin

Since he is new to the business, his knowledge is limited to the following drinks:

- *Whiskey Sour*. Each serving uses 2 oz. whiskey and sells for \$1.
- *Manhattan*. Each serving uses 2 oz. whiskey and 1 oz. vermouth, and sells for \$2.
- *Martini*. Each serving uses 2 oz. gin and 1 oz. vermouth, and sells for \$2.
- *Pub Special*. Each serving uses 2 oz. gin and 2 oz. whiskey, and sells for \$3.

First, formulate an LP model to maximize the bar's profit. Ignore the fact that the numbers of servings have to be integers. Second, if the problem is not canonical, convert it into a standard form, and you will see the resulting problem is canonical. Next, solve the LP problem using the simplex method. Is there a unique optimal solution, or multiple optimal solutions? Finally, convert the solution to the solution of the original problem and interpret the result.

Exercise 4.3. Suppose that, after applying the simplex method to solve a maximization LP problem, you obtain the tableau below, where $a_{13}, a_{14}, a_{23}, a_{24}, a_{33}, a_{34}, b_1, b_2, b_3, c_3, c_4, d$ are real numbers.

z	x_1	x_2	x_3	x_4	x_5	RHS	Basic var
1	0	0	c_3	c_4	0	d	$z = d$
0	0	1	a_{13}	a_{14}	0	b_1	$x_2 = b_1$
0	0	0	a_{23}	a_{24}	1	b_2	$x_5 = b_2$
0	1	0	a_{33}	a_{34}	0	b_3	$x_1 = b_3$

max z; $x \geq 0$

Suppose $b_1 \geq 0$, $b_2 \geq 0$ and $b_3 \geq 0$. For each of the scenarios below, describe the next action to take in applying the simplex method:

1. $c_3 < 0$, $c_4 > 0$, $a_{13} < 0$, $a_{23} < 0$ and $a_{33} > 0$.
2. $c_3 < 0$, $c_4 > 0$, $a_{13} < 0$, $a_{23} < 0$ and $a_{33} = 0$.
3. $c_3 > 0$, $c_4 > 0$.
4. $c_3 > 0$, $c_4 = 0$.
5. $c_3 > 0$, $c_4 < 0$, $a_{14} > 0$, $a_{24} > 0$ and $a_{34} < 0$.

Note that

- If you conclude that an optimal solution is found, write down an optimal solution.
- If you conclude that the problem has multiple solutions, then can you conclude if the solution set is bounded or unbounded?
- If you conclude that the LP is unbounded, it suffices to state that.
- If the next action is about another iteration, describe the entering variable and the way to choose the leaving variable, but you do not have to conduct the actual row operations to write down the next tableau.

Exercise 4.4. Follow the following steps for each of the problems given below.

- First, convert any minimization problem into a maximization problem by changing the signs of coefficients in the objective function.
- Next, convert the problem into standard form; you will find the resulted LP to be in canonical form.
- Then, apply the simplex method, and decide if the problem is unbounded or has unique or multiple optimal solutions.
- Visualize the result in the x_1x_2-plane (that is, solve these problems graphically) and compare with the result you obtained from the simplex methods.

1.

$$\begin{cases} \max\limits_{x} & z = 2x_1 + 3x_2 \\ \text{s.t.} & x_1 + 2x_2 \le 6, \\ & 2x_1 + x_2 \le 8, \\ & x_1, x_2 \ge 0 \end{cases}$$

2.

$$\begin{cases} \max\limits_{x} & z = -3x_1 + 6x_2 \\ \text{s.t.} & 5x_1 + 7x_2 \le 35, \\ & -x_1 + 2x_2 \le 2, \\ & x_1, x_2 \ge 0 \end{cases}$$

3.

$$\begin{cases} \min\limits_{x} & z = -x_1 - 3x_2 \\ \text{s.t.} & x_1 - 2x_2 \le 4, \\ & -x_1 + x_2 \le 3, \\ & x_1, x_2 \ge 0 \end{cases}$$

Exercise 4.5. Determine the feasibility of each LP problem below by constructing an auxiliary LP problem and perform Phase 1 of the two-phase simplex method. If an LP problem is not in standard form, first convert it into standard form. For each LP problem, determined as feasible, write down its feasible solution obtained from the last tableau of Phase 1, but you do not need to find its optimal solution. To avoid unnecessary calculations, do not define artificial variables for equations that already isolate variables.

1.

$$\begin{cases} \max\limits_{x} & z = 2x_1 + 3x_2 - x_4 \\ \text{s.t.} & 2x_1 + x_2 + 2x_3 \le 16, \\ & x_1 + x_2 - x_3 \ge 15, \\ & x_1 \quad + x_3 - x_4 = -10, \\ & x_i \ge 0, \ i = 1, \cdots, 4. \end{cases}$$

2.

$$\begin{cases} \max\limits_{x} z = x_8 \\ \text{s.t. } -x_1 \quad -x_3 + x_4 + x_5 \qquad = -2, \\ \quad x_1 \quad -2x_3 - 3x_4 \quad + x_6 \quad - x_8 = 4, \\ \quad x_1 \quad + x_3 - 0.5x_4 \quad + x_7 \quad = 1, \\ \quad 2x_1 + x_2 \quad - 5x_4 \qquad - x_8 = 6, \\ \quad x_i \geq 0, \quad i = 1, \cdots, 8. \end{cases}$$

Exercise 4.6. Consider the following LP problem:

$$\begin{cases} \min\limits_{x,y} \; z = y_1 + 2y_2 + 3y_3 \cdots + my_m \\ \text{s.t.} \;\; Ax + y = 1 \\ \qquad x \geq 0, \;\; y \geq 0, \end{cases}$$

where A is some $m \times n$ matrix, $1 = (1, 1, \cdots, 1)^T$ is the vector of all ones in $\mathbb{R}^m$, and $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are the variables.

1. Is this LP problem always feasible? Why?
2. Is it possible for this LP problem to be unbounded? Why?

Please explain your answers in detail.

Exercise 4.7. Consider the following linear program:

$$\begin{cases} \min\limits_{x} z = x_1 - 2x_2 - x_4 + x_5 - x_6 \\ \text{s.t.} \;\; x_1 + x_2 - x_3 + 2x_4 = 10 \\ \qquad 2x_2 + 2x_3 + x_5 = 12 \\ \qquad -2x_2 + 3x_3 - x_4 + x_6 = 8 \\ \qquad x_1, \cdots, x_6 \geq 0. \end{cases}$$

1. Is this LP canonical? Is this LP always feasible? If yes, find a feasible solution. Is it a basic feasible solution. Explain in detail your answers.
2. If you find a basic feasible solution in part 1, using it as a starting point and solve the problem by using simplex method.

Exercise 4.8. Given the following linear program:

$$\begin{cases} \min\limits_{x} \ z = 3x_1 + x_2 - 3x_3 \\ \text{s.t.} \ \ x_1 + 2x_2 - x_3 = 2 \\ \qquad 10x_2 - 5x_3 = -5 \\ \qquad -3x_2 + 2x_3 = 4, \\ \qquad x_1, x_2, x_3 \geq 0. \end{cases}$$

Answer the following questions:

1. First, convert this problem into a standard maximization LP problem.
2. Next, form the corresponding auxiliary problem.
3. Using simplex method to solve this auxiliary problem as Phase 1.
4. If the standard LP problem is feasible, and all the artificial variables are non-basic, then move to Phase 2 to solve this standard LP problem.
5. Finally, if you answer part 4, then convert the solution you obtain back to the original problem.

Chapter 5

Duality Theory and Sensitivity Analysis

5.1 The primal and dual pair

Each linear program is naturally paired with another linear program. One in this pair is called the **primal LP**, and the other is called the **dual LP**. These two LPs are defined by the same data: the coefficients and constants in constraints and the objective function of one LP are used in the other LP, but in a transposed manner. One LP in each primal-dual pair is a minimization problem, and the other is a maximization problem.

Given an LP, we will construct its dual LP by identifying the following attributes:

- The number of dual variables, and the number of dual constraints.
- The direction of optimization: to minimize or to maximize.
- The coefficients in the dual objective function.
- The coefficient matrix of the constraints and the right-hand side constants of the constraints.
- The types of constraints: LE ($\leq$), GE ($\geq$), or equality constraints.
- The types of sign restrictions on the variables: nonnegative, nonpositive, or free.

In this procedure, it will be convenient to classify constraints (not the sign restrictions) into three types: *natural*, *reversed*, or *equality*, as shown in Table 5.1. The class of a constraint depends on its own type ($\leq$, $\geq$, or $=$) as well as the direction of the optimization problem it belongs to. For example, a constraint of the type $\leq$ in a maximization LP is a natural constraint, and a constraint of the type $\leq$ in a minimization LP is a reversed constraint. Also note that Table 5.1 should not be

Types of constraints	in a max LP	in a min LP
Natural constraints	$\leq$ (LE)	$\geq$ (GE)
Reversed constraints	$\geq$ (GE)	$\leq$ (LE)
Equality constraints	$=$ (E)	$=$ (E)

Table 5.1 Natural, reversed, and equality constraints

applied to sign restrictions of variables, which are classified into the three types, nonnegative, nonpositive and free, as before.

The following steps give us a generic procedure to construct the dual LP from a given primal LP. Here, the term "constraint" is used in a restrictive way and does not include sign restrictions on variables.

Step 1: For each constraint i of the primal LP, define a dual variable, e.g., y_i.

- Again, **sign restrictions** will be treated differently. Do not define dual variables for them.

Step 2: Determine the dual objective function.

- If the primal is a max problem, then the dual is a min problem. Conversely, if the primal is a min problem, then the dual is a max problem.
- Use the right-hand side constants of the primal constraints as coefficients in the dual objective function.

Step 3: Determine data in the dual constraints.

- For each primal variable, e.g., x_j, define a corresponding dual constraint j.
- The right-hand side constant of this dual constraint is given by the coefficient of this primal variable in the primal objective function.
- The left-hand side coefficients of this dual constraint are given by the coefficients of the primal variable in primal constraints. In other words, the j-th column of the coefficient matrix for the primal constraints becomes the j-th row of the coefficient matrix in the dual constraints.

Step 4: Determine types of the dual constraints ($\leq$, $\geq$ or $=$).

- By construction, each dual constraint j corresponds to a primal variable x_j. The type of this dual constraint depends on the sign restriction of the primal variable x_j.
  - If its corresponding primal variable is **nonnegative**, then the corresponding dual constraint is **natural**. *Hence, based on Table 5.1, the dual constraint is of type $\leq$ if the dual LP is a max problem, and it is of type $\geq$ if the dual LP is a min problem.*
  - If its corresponding primal variable is **nonpositive**, then the dual constraint is **reversed**. *Again, based on Table 5.1, the dual constraint is of type $\geq$ if the dual LP is a max problem, and it is of type $\leq$ if the dual LP is a min problem.*
  - If its corresponding primal variable is **free**, then the dual constraint is an **equality constraint**.

Step 5: Determine sign restrictions of the dual variables.

- By construction, each dual variable, e.g., y_i, corresponds to a primal constraint i, the one used to define the dual variable.
  - If its corresponding primal constraint is **natural**, then the dual variable is **nonnegative**. *Again, whether the constraint is natural depends on both its type and the optimization direction of the primal LP. Based on Table 5.1, the primal constraint is natural, if it is of type $\leq$ when the primal LP is a max problem, or if it is of type $\geq$ when the primal LP is a min problem.*
  - If its corresponding primal constraint is **reversed**, then the dual variable is **nonpositive**. *Based on Table 5.1, the primal constraint is reversed, if it is of type $\geq$ when the primal LP is a max problem, or if it is of type $\leq$ when the primal LP is a min problem.*
  - If its corresponding primal constraint is an equality, then the dual variable is **free**.

Let us provide some examples to illustrate this procedure.

Example 5.1. Given a primal LP:

$$\begin{cases} \qquad\qquad \underline{\text{Primal}} \\ \min\limits_{x} \quad z = 2x_1 + 3x_2 \\ \text{s.t.} \qquad\qquad x_1 + x_2 \geq 5 \\ \text{and } x_1 \geq 0,\ x_2 \geq 0. \end{cases} \tag{5.1}$$

By **Step 3**, the dual problem is as follows:

$$\begin{cases} \qquad \underline{\text{Dual}} \\ \max\limits_{y} \ v = 5y \\ \text{s.t.} \qquad y \ \square\ 2 \\ \qquad\qquad y \ \square\ 3 \\ \text{and} \quad y \ \square \end{cases}$$

Here, we have yet to decide constraint types and sign restrictions of the dual variable.

Dual constraint types:

- The primal variable $x_1 \geq 0$: the 1st dual constraint should be natural. A natural constraint in a max problem is of the type $\leq$. So the 1st dual constraint is $y \leq 2$.
- The primal variable $x_2 \geq 0$: the 2nd dual constraint should be natural: $y \leq 3$.

The sign restrictions of the dual variable:

- The primal constraint is of type $\geq$ in a min problem, so it is natural. The dual variable corresponding to a natural constraint is always nonnegative, so the sign restriction is $y \geq 0$.

The dual problem of (5.1) becomes

$$\begin{cases} \max\limits_{y} \ v = 5y \\ \text{s.t.} \qquad y \ \leq \ 2 \\ \qquad\qquad y \ \leq \ 3 \\ \qquad\qquad y \geq 0. \end{cases}$$

Example 5.2. Consider the following primal LP:

$$
\textbf{(Primal)}\ \begin{cases}
\min\limits_{x}\ z = & 2x_1 & +5x_2 & -4x_3 & \\
\text{s.t.} & 2x_1 & +x_2 & -x_3 & \geq 5 \\
& 4x_1 & -2x_2 & +3x_3 & \leq 7 \\
& x_1 & +3x_2 & -2x_3 & = 12 \\
& x_1 & \leq 0, & x_3 & \geq 0.
\end{cases}
$$

This problem has three different types of constraints, and three different types of variables. Following the above procedure, we can determine:

- The number of dual variables $m = 3$, and the number of dual constraints $n = 3$.
- The dual LP is a maximization problem.
- If we denote the dual variables by y_1, y_2, and y_3, then $y_1 \geq 0$ (since the first constraint is natural), $y_2 \leq 0$ (since the second constraint is reversed), and y_3 is free.
- The first dual constraint is GE (reversed), the second is E (equality), and the last is LE (natural).

In summary, we can write the entire dual problem as

$$
\textbf{(Dual)}\ \begin{cases}
\max\ v = & 5y_1 & +7y_2 & +12y_3 & \\
\text{s.t.} & 2y_1 & +4y_2 & +y_3 & \geq & 2 \\
& y_1 & -2y_2 & +3y_3 & = & 5 \\
& -y_1 & +3y_2 & -2y_3 & \leq & -4 \\
& y_1 & \geq 0, & y_2 & \leq 0.
\end{cases}
$$

We note that if we form the dual problem of the dual LP, then we obtain again the primal LP. That is $\textbf{Dual}\big(\textbf{Dual}\big) = \textbf{Primal}$.

5.2 Weak and strong duality, and complementary slackness

Next, we investigate the relation between the primal and dual linear programs. First, if x is a feasible solution of the primal problem, and y is feasible for the dual problem, is there a relation between them? Second, if x^* is an optimal solution of the primal problem, can we say anything about optimal solutions of the dual problem? Third, given a primal optimal solution x^*, is there a way to compute optimal solutions of the dual problem, or check the optimality of a given dual feasible solution?

5.2.1 Weak and strong duality

We answer the first question by proving the following theorem. This theorem is called the **weak duality theorem** for linear programs.

Theorem 5.1 (Weak duality). *Suppose for a pair of primal-dual LPs that the objective of the primal LP is to maximize $c^T x$, and the objective of the dual LP is to minimize $b^T y$. If x is feasible for the primal LP, and y is feasible for the dual LP, then*

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

Proof. Let us suppose the primal LP is of the following form:

$$\begin{cases} \max\limits_{x} & z = c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0, \end{cases}$$

where A is an $m \times n$ matrix. The corresponding dual problem becomes:

$$\begin{cases} \min\limits_{y} & v = b^T y \\ \text{s.t.} & A^T y \geq c \\ & y \geq 0. \end{cases}$$

Suppose that x is feasible to the primal problem, and y is feasible to the dual problem. We define

$$u_i = y_i(b_i - \sum_{j=1}^{n} A_{ij} x_j),\ i = 1, \cdots, m, \text{ and } v_j = (\sum_{i=1}^{m} A_{ij} y_i - c_j) x_j,\ j = 1, \cdots, n.$$

Then, it follows from the relationship between the primal and dual LP's that

- for each $i = 1, \cdots, m$, $y_i \geq 0$ and $b_i - \sum_{j=1}^{n} A_{ij} x_j \geq 0$ and
- for each $j = 1, \cdots, n$, $x_j \geq 0$ and $\sum_{i=1}^{m} A_{ij} y_i - c_j \geq 0$.

Hence, u_i and v_j are all nonnegative. Moreover, we can easily show that

$$\sum_{i=1}^{m} u_i = y^T b - y^T A x \quad \text{and} \quad \sum_{j=1}^{n} v_j = y^T A x - c^T x.$$

These expressions imply

$$y^T b - c^T x = \sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j \geq 0.$$

Hence, $b^T y \geq c^T x$, and we complete the proof. □

We note that the proof of Theorem 5.1 is based on the assumption that the constraints of the primal problem are natural, and all the variables are nonnegative. However, this proof can be extended to general primal-dual pairs, because y_i and $b_i - \sum_{j=1}^n A_{ij}x_j$ always have the same sign in any general primal-dual pair. This means that we always have $u_i = y_i(b_i - \sum_{j=1}^n A_{ij}x_j) \geq 0$. Similarly, x_j and $\sum_{i=1}^m A_{ij}y_i - c_j$ also have the same sign, which implies $v_j = (\sum_{i=1}^m A_{ij}y_i - c_j)x_j \geq 0$ (see Exercise below). The proof of this theorem is therefore valid in general.

Suppose that a primal LP and its dual both have optimal solutions. Then, based on the weak duality theorem, the optimal value of the maximization LP in this primal-dual pair is no more than the optimal value of the minimization LP in this pair (exercise). Suppose, on the other hand, that one LP in this pair is unbounded; then the weak duality theorem tells us that the other LP must be infeasible. In other words:

(a) If a minimization LP is unbounded, then its optimal value is $-\infty$, and its dual is infeasible.

(b) If a maximization LP is unbounded, then its optimal value is $+\infty$, and its dual is infeasible.

The following theorem gives a sharper result on the relation between the optimal values of the primal and dual LPs.

Theorem 5.2 (Strong duality). *In a pair of primal-dual LPs, if one problem has an optimal solution, then the other also has an optimal solution, with the same optimal value.*

Proof. Without loss of generality, we can prove this theorem for a canonical LP. Otherwise, we can always transform it into a canonical form that is equivalent to the original problem as long as the problem has an optimal solution. Our proof is based on the simplex method.

Write the primal LP as

$$\begin{cases} \max\limits_{x} & z = c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0. \end{cases}$$

The corresponding dual problem becomes:

$$\begin{cases} \min\limits_{y} \ v = b^T y \\ \text{s.t.} \ \ A^T y \geq c. \end{cases}$$

Suppose that the primal LP is in canonical form and has an optimal solution. Then, if we apply the simplex method to solve it, we will end up with an optimal tableau that shows a basic optimal solution (see the previous chapter). Let us write the initial simplex tableau as

z	x	RHS
1	$-c^T$	0
0	A	b

and the final optimal tableau that displays a basic optimal solution x^* as

z	x	RHS	Basic var
1	$\bar{c}^T$	z^*	$z = z^*$
0	$\bar{A}$	$\bar{b}$	$x_B = \bar{b}$

The elementary row operations performed at iterations of the simplex method have led us to the final optimal tableau, starting from the initial simplex tableau. At each such iteration, we have added a multiple of the pivot row to the top row. The pivot row of a tableau is the linear combination of rows (excluding the top row) of its previous tableau. Rows (excluding the top row) of the previous tableau are in turn linear combinations of rows (excluding the top row) of the tableau before it. Hence, each row (excluding the top row) of each tableau is a linear combination of rows (excluding the top row) in the initial tableau. As a result, the reduced cost vector $\bar{c}$ that we obtain in the optimal tableau can be written as $\bar{c}^T = (y^*)^T A - c^T$, where y^* collects all the multiples that we have added up in this procedure (more precisely, $y^* = (A_B^{-1})^T c_B$). Clearly, since x^* is optimal, $\bar{c}^T \geq 0$. This implies $A^T y^* \geq c$.

Since the same elementary row operations have been conducted for the RHS column, the RHS entry in the top row of the optimal tableau is given by $z^* = b^T y^*$. But this is also optimal value of the primal problem, so we have $z^* = b^T y^* = c^T x^*$. The fact $A^T y^* \geq c$ shows that y^* is a feasible solution to the dual problem, and the fact $b^T y^* = c^T x^* \leq b^T y$ shows that $b^T y^* \leq b^T y$ for any feasible solution y due to

Theorem 5.1. Hence, we conclude that y^* is an optimal solution of the dual problem, and $b^T y^* = c^T x^*$. □

5.2.2 The complementary slackness theorem

As in Theorem 5.1, we will use the following primal-dual pair

$$\text{Primal problem:} \begin{cases} \max\limits_{x} \ z = c^T x \\ \text{s.t.} \quad Ax \leq b \\ \qquad\ x \geq 0, \end{cases} \tag{P}$$

$$\text{Dual problem:} \begin{cases} \min\limits_{x} \ t = b^T y \\ \text{s.t.} \quad y \geq 0 \\ \qquad\ A^T y \geq c \end{cases} \tag{D}$$

to illustrate the proof of the complementary slackness theorem below.

Theorem 5.3. *Vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are optimal solutions to the primal LP* (P) *and the dual LP* (D) *respectively, if and only if they simultaneously satisfy the following three conditions:*

1. *x is a primal feasible solution.*
2. *y is a dual feasible solution.*
3. *x and y satisfy the following $m+n$ equalities:*

$$\begin{cases} y_i \left(b_i - \sum_{j=1}^{n} A_{ij} x_j \right) = 0, \quad i = 1, \cdots, m \\ \left(\sum_{i=1}^{m} A_{ij} y_i - c_j \right) x_j = 0, \quad j = 1, \cdots, n. \end{cases} \tag{5.2}$$

Remark 5.1. Equations in (5.2) are called the **complementary slackness condition**. These are not linear equations due to the cross product terms between x_j and y_i.

Proof. Suppose that x and y are feasible solutions to the primal LP (P) and the dual LP (D) respectively. We will show that they are optimal solutions to the primal LP (P) and the dual LP (D), if and only if the complementary slackness condition hold.

Indeed, we define

$$u_i = y_i (b_i - \sum_{j=1}^{n} A_{ij} x_j), \ i = 1, \cdots, m \quad \text{and} \quad v_j = (\sum_{i=1}^{m} A_{ij} y_i - c_j) x_j, \ j = 1, \cdots, n.$$

It was shown in the proof of Theorem 5.1 that u_i and v_j are all nonnegative, with

$$\sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j = y^T b - c^T x. \tag{5.3}$$

Now, suppose that x and y are optimal solutions to the primal LP (P) and the dual LP (D), respectively. By the strong duality theorem (Theorem 5.2), we have $c^T x = y^T b$. This and (5.3) imply that all of the u_i and v_j are zeros. This means that the conditions in (5.2) hold.

Conversely, suppose all conditions in (5.2) hold. Then all components of the u_i and v_j are zeros. By (5.3) we have $c^T x = y^T b$. The weak duality theorem now tells us that x and y are optimal solutions to the primal LP (P) and the dual LP (D). Indeed, if x is not an optimal solution of (P), there exists a feasible solution $\hat{x}$ such that $c^T x < c^T \hat{x}$. By Theorem 5.1, we have $c^T \hat{x} \leq b^T y$ since y is a feasible solution to (D). Hence, $c^T x < c^T \hat{x} \leq b^T y$, which contradicts $c^T x = b^T y$. Hence, x must be an optimal solution to (P). Similarly, y must be an optimal solution to (D). □

To see the meaning of the equalities in (5.3), pair the primal constraints $Ax \leq b$ one by one with the corresponding sign restrictions $y \geq 0$, and pair the sign restrictions $x \geq 0$ with corresponding dual constraints $A^T y \geq c$. Every pair consists of two inequalities, and the equalities in (5.3) say that at least one of those two inequalities is **satisfied as an equality**. An inequality is said to be satisfied as an equality if both sides of it are equal. For example, the inequality $2x_1 + x_2 \geq 8$ is satisfied as an equality at $x = (3,2)$, and $y_1 \leq 0$ is satisfied as an equality at $y_1 = 0$.

When $\sum_{j=1}^{n} A_{ij} x_j \leq b_i$ is satisfied with an equality at a given point, we say that the i-th primal constraint is **active** (also called **binding**) at this point. Otherwise, it is said to be **inactive**. Similarly, if $\sum_{i=1}^{m} A_{ij} y_i = c_j$ at a given point, then we say that the j-th dual constraint is **active** there. Otherwise, it is **inactive**.

Although the statement of Theorem 5.3 only covers primal and dual LPs in the above special formats, it can be readily extended to general primal and dual LPs. For a general primal LP and its dual LP, pair the primal constraints one by one with the corresponding sign restrictions for the dual variables, and pair the primal sign restrictions one by one with corresponding dual constraints. Then, the general complementary slackness theorem states:

Let x and y be a pair of feasible solutions to the primal LP and the dual LP respectively. Then, they are optimal solutions to the primal LP and the dual LP respectively, if and only if at least one in each pair of primal-dual constraints is active at these solutions.

Again, the above condition that "at least one in each pair of primal-dual constraints is active at these solutions" is called the complementary slackness condition. If, in a pair of primal-dual constraints, the primal constraint is an equality, then the corresponding dual sign restriction will be free. The complementary slackness condition for such a pair is automatically satisfied whenever the considered primal solution is feasible, because the equality constraint is satisfied at any primal feasible solution.

Example 5.3. Consider the following primal-dual LP pair:

$$
\text{(P)}\ \begin{cases} & \underline{\text{Primal}} \\ \min\limits_{x} & 13x_1 + 10x_2 + 6x_3 \\ \text{s.t.} & 5x_1 + x_2 + 3x_3 = 8 \\ & 3x_1 + x_2 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{cases}
\qquad
\text{(D)}\ \begin{cases} & \underline{\text{Dual}} \\ \max\limits_{y} & 8y_1 + 3y_2 \\ \text{s.t.} & 5y_1 + 3y_2 \leq 13 \\ & y_1 + y_2 \leq 10 \\ & 3y_1 \leq 6 \end{cases}
$$

- Is $x^\star = (1,0,1)$ an optimal solution to the primal?

  Solution: First, note that $x^\star$ is a primal feasible solution by directly checking all the constraints. To check its optimality, we need to check if there exists a dual feasible solution $y^\star$ such that $(x^\star, y^\star)$ satisfies the complementary slackness condition. Since $x_1^\star = 1 > 0$ and $x_3^\star = 1 > 0$, we need the first and third dual constraints to be satisfied as equalities at $y^\star$. That is, both these constraints need to be **active** at $y^\star$. Hence, the conditions on $y^\star$ are:

$$
\begin{cases} 5y_1 + 3y_2 = 13 \\ y_1 + y_2 \leq 10 \\ 3y_1 = 6. \end{cases} \tag{5.4}
$$

  Solving the system of two equations $5y_1 + 3y_2 = 13$ and $3y_1 = 6$ in y_1 and y_2 we obtain $y^* = (2,1)$, which satisfies $y_1 + y_2 \leq 10$. Clearly, y^* is feasible to the dual problem (D), and $b^T y^* = c^T x^\star = 19$. Hence, $x^\star$ is a primal optimal solution.

- What is the set of optimal solutions to the dual problem?

  Solution: Since $y^\star = (2,1)$ is dual feasible, and satisfies the complementary slackness condition with $x^\star$, it is a dual optimal solution. Also note that no other vector in $\mathbb{R}^2$ satisfies the three conditions in (5.4) simultaneously. Hence $y^\star = (2,1)$ is the unique dual optimal solution.

Example 5.4. Consider the following primal and dual LP pair:

$$(\mathrm{P})\quad \begin{cases} \max\limits_x z = x_1 + 2x_2 + 2x_3 + 3x_4 \\ \text{s.t.} \quad 2x_1 + 2x_2 + 2x_4 \le 64 \\ \qquad\quad x_2 + x_3 \le 32 \\ \qquad\quad 2x_3 + 2x_4 \le 64 \\ \qquad\quad x_1, x_2, x_3, x_4 \ge 0, \end{cases}$$

and

$$(\mathrm{D})\quad \begin{cases} \min\limits_y v = 64y_1 + 32y_2 + 64y_3 \\ \text{s.t.} \quad 2y_1 \ge 1 \\ \qquad\quad 2y_1 + y_2 \ge 2 \\ \qquad\quad y_2 + 2y_3 \ge 2 \\ \qquad\quad 2y_1 + 2y_3 \ge 3 \\ \qquad\quad y_1, y_2, y_3 \ge 0. \end{cases}$$

Verify that $y^* = (\frac{3}{4}, \frac{1}{2}, \frac{3}{4})^T$ is an optimal solution of (D). Compute an optimal solution x^* of (P).

Solution: Let us assume that y^* is optimal to (D). We denote x^* an optimal solution to (P). By substituting y^* into the constraints of (D), we can see that the last three constraints are active (i.e., satisfied as equalities). The first dual constraint is not active at y^*, so $x_1^* = 0$ must hold. Moreover, since y_1^*, y_2^* and y_3^* are all strictly positive, their corresponding constraints in (P) need to be active at x^*. This means that x^* is a solution of the following three equations:

$$\begin{cases} 2x_2 + 2x_4 = 64 \\ x_2 + x_3 = 32 \\ 2x_3 + 2x_4 = 64. \end{cases}$$

The only solution to the above equations is $x_2 = x_3 = x_4 = 16$, so we find $x^* = (0, 16, 16, 16)^T$. We can check that x^* is feasible to (P). Hence, y^* is optimal to the dual problem (D), and x^* is the only optimal solution to the primal problem (P).

5.3 Sensitivity Analysis

Sensitivity analysis in linear programming analyzes the effects of perturbation of the input data on solutions of LPs. This section introduces sensitivity analysis techniques for LPs. More details can be found in linear programming textbooks, e.g., [3, 7, 23].

5.3.1 Introduction

We consider the following LP in standard form:

$$\begin{cases} \max\limits_{x} & z = c^T x \\ \text{s.t.} & Ax = b, \\ & x \geq 0. \end{cases} \tag{5.5}$$

Here, the matrix $A \in \mathbb{R}^{m \times n}$, vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are input data for this LP. We can put them into a triple (A, b, c). In many situations the input data (A, b, c) may not be accurate and may change over time.

Suppose that this LP has an optimal solution x^* with the optimal value z^*. The following questions can be asked:

1. How do the optimal solution x^* and/or the optimal value z^* change if
   - the objective coefficient vector c changes?
   - the right-hand side vector b changes?
   - the constraint coefficient matrix A changes?
2. How much can we change the input data (A, b, c) so that x^* remains an optimal solution?

When the right hand side constant of the ith constraint of (5.5) changes from b_i to $b_i + \delta$, the optimal value of (5.5) changes accordingly. Under some conditions, the change in the optimal value of (5.5) can be represented as a linear function of δ, for δ belonging to a certain interval that includes 0. The slope of such a linear function is called the **shadow price** on this constraint, which represents the rate of change in the optimal value as the right-hand side of this constraint changes from its current value b_i.

The following theorem makes precise the above claim. A proof can be found in [3]. Again, this theorem can be extended to general linear programs.

Theorem 5.4. *Suppose that the LP* (5.5) *has a nondegenerate optimal basic solution x^*. Then its dual has a unique optimal solution y^*. Moreover, for each $i = 1, \cdots, m$, there exists an interval $[l_i, u_i]$ with $l_i < 0 < u_i$ such that for each $\delta \in [l_i, u_i]$ the optimal value of the LP* (5.5) *with b_i replaced by $b_i + \delta$ is given by $z^* + y_i^* \delta$.*

Below, we consider the following example for a simple illustration.

$$\begin{cases} \max\limits_{x} \ z = 500x_1 + 450x_2 \\ \text{s.t.} \quad 6x_1 + 5x_2 \leq 60 \\ \qquad 10x_1 + 20x_2 \leq 150 \\ \qquad 0 \leq x_1 \leq 8, \ x_2 \geq 0. \end{cases} \tag{5.6}$$

This LP has a unique optimal solution $x^* = (\frac{45}{7}, \frac{30}{7})^T$ with the optimal value $z^* = 5142\frac{6}{7}$.

If we increase the RHS of the first constraint by 1 unit, so that the first constraint becomes $6x_1 + 5x_2 \leq 61$, how much does the optimal value change? If we solve this LP graphically, we can find that the optimal solution after the change is still determined by the following equations

$$\begin{cases} 6x_1 + 5x_2 = 61 \\ 10x_1 + 20x_2 = 150. \end{cases}$$

The new optimal solution is $\hat{x}^* = (6\frac{5}{7}, 4\frac{1}{7})$ and the new optimal value is $\hat{z}^* = 5221\frac{3}{7}$. The change in the optimal value is $\hat{z}^* - z^* = 5221\frac{3}{7} - 5142\frac{6}{7} = 78\frac{4}{7}$.

To use this example to verify the above theorem, we can write the dual LP of (5.6) as

$$\begin{cases} \min\limits_{y} \ t = 60y_1 + 150y_2 + 8y_3 \\ \text{s.t.} \quad 6y_1 + 10y_2 + y_3 \geq 500 \\ \qquad 5y_1 + 20y_2 \geq 450 \\ \qquad y \geq 0. \end{cases}$$

Solving the dual LP we obtain $y_1^* = 78\frac{4}{7}$, exactly the same value as $\hat{z}^* - z^*$.

In the subsequent sections, we use the following example to describe a method that uses information in the optimal tableau to conduct sensitivity analysis. This method works whenever an optimal tableau for the original LP is available.

Example 5.5. Consider the following LP:

$$\begin{cases} \max_{x}\ z = 41x_1 + 25x_2 \\ \quad \text{s.t.}\ \ 3x_1 + 2x_2 + x_3 = 10 \\ \quad\quad\ \ 5x_1 + 3x_2 + x_4 = 16 \\ \quad\quad\ x_1, \cdots, x_4 \geq 0. \end{cases} \tag{5.7}$$

Since (5.7) is canonical, we can apply the simplex method to solve it. The initial tableau (called Tableau 1) in solving (5.7) is:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	-41	-25	0	0	0	$z = 0$
0	3	2	1	0	10	$x_3 = 10$
0	5	3	0	1	16	$x_4 = 16$

After several simplex iterations, we arrive at the optimal tableau below (called Tableau 2), which shows the optimal basic solution in which x_1 and x_2 are basic variables. We say that $\{x_1, x_2\}$ is an **optimal basis**.

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	0	0	2	7	132	$z = 132$
0	1	0	-3	2	2	$x_1 = 2$
0	0	1	5	-3	2	$x_2 = 2$

5.3.2 Changes in the objective coefficients

We consider two cases:

- **Case 1:** The coefficient is associated with a nonbasic variable.
- **Case 2:** The coefficient is associated with a basic variable.

Case 1: Consider changing the coefficient for x_3 in the objective function from 0 to δ. Note that x_3 is a nonbasic variable in Tableau 2. Tableau 1 becomes Tableau 1a below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	-41	-25	$-\delta$	0	0	$z=0$
0	3	2	1	0	10	$x_3=10$
0	5	3	0	1	16	$x_4=16$

Starting from Tableau 1a, we conduct the same EROs (elementary row operations) that bring Tableau 1 to Tableau 2, to arrive at Tableau 2a below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	0	0	$2-\delta$	7	132	$z=132$
0	1	0	-3	2	2	$x_1=2$
0	0	1	5	-3	2	$x_2=2$

Clearly, if $\delta \leq 2$, the current BFS (basic feasible solution) $x=(2,2,0,0)^T$ remains optimal, and the optimal value is $z=132$. If $\delta > 2$, then extra simplex iterations are needed to find the new optimal solution.

<u>Case 2:</u> Now, consider changing the coefficient in the objective function for x_1 from 41 to $41+\delta$. Note that x_1 is a basic variable in Tableau 2. Tableau 1 becomes Tableau 1b below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	$-41-\delta$	-25	0	0	0	$z=0$
0	3	2	1	0	10	$x_3=10$
0	5	3	0	1	16	$x_4=16$

Starting from Tableau 1b, we conduct the same EROs that bring Tableau 1 to Tableau 2, to arrive at Tableau 2b below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	$-\delta$	0	2	7	132	$z=132$
0	1	0	-3	2	2	$x_1=2$
0	0	1	5	-3	2	$x_2=2$

Tableau 2b is not a valid simplex tableau, because the coefficient for x_1 in row 0 is nonzero. It needs to be zero since x_1 is a basic variable in this tableau. We add δ times row 1 to row 0 to obtain Tableau 3b below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	0	0	$2-3\delta$	$7+2\delta$	$132+2\delta$	$z = 132+2\delta$
0	1	0	-3	2	2	$x_1 = 2$
0	0	1	5	-3	2	$x_2 = 2$

In order for the current BFS to remain optimal, we need $2-3\delta$ and $7+2\delta$ to be nonnegative. Hence, when $-7/2 \le \delta \le 2/3$, the current BFS $x = (2,2,0,0)$ remains optimal, and the optimal value is $z = 132+2\delta$. If δ does not belong to the above range, then extra iterations need to be conducted.

5.3.3 Changes in the right hand side coefficients

Now, consider changing the right hand side constant for the first constraint from 10 to $10+\delta$. Tableau 1 becomes Tableau 1c below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	-41	-25	0	0	0	$z = 0$
0	3	2	1	0	$10+\delta$	$x_3 = 10+\delta$
0	5	3	0	1	16	$x_4 = 16$

Starting from Tableau 1c, we conduct the same EROs that bring Tableau 1 to Tableau 2, to arrive at Tableau 2c below:

z	x_1	x_2	x_3	x_4	RHS	Basic var
1	0	0	2	7	132+2δ	$z = 132+2\delta$
0	1	0	-3	2	2$-$3δ	$x_1 = 2-3\delta$
0	0	1	5	-3	2+5δ	$x_2 = 2+5\delta$

The right hand side column in Tableau 2c is obtained by noting that the column under x_3 changes from

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{to} \quad \begin{pmatrix} 2 \\ -3 \\ 5 \end{pmatrix}$$

after the elementary row operations. So a column that starts with $(0,\delta,0)^T$ will become $(2\delta,-3\delta,5\delta)^T$ after the same EROs.

Consequently, whenever $2-3\delta \ge 0$ and $2+5\delta \ge 0$ hold simultaneously, Tableau 2c shows an optimal solution. This happens when $-2/5 \le \delta \le 2/3$. For each δ belonging to this range, $\{x_1, x_2\}$ is an optimal basis, and an optimal solution is given by $x = (2-3\delta, 2+5\delta, 0, 0)$, with the optimal value $132+2\delta$.

5.4 Exercises

Exercise 5.1. Consider a primal-dual pair of LPs, where the primal problem is a maximization problem. Using primal-dual rules to show that the vectors u and v computed by $u = y^T(A^Tx - c)$ and $v = y^T(b - Ax)$ are always nonnegative, where x is the vector of primal variables and y is the vector of dual variables.

Exercise 5.2. For each LP problem below, write down the corresponding dual LP problem. Check if the dual problem is in a standard or in a canonical form (Explain why?). Explain how do you conduct the sign of the variables and the constraints in the dual problem?

1.
$$\begin{cases} \min\limits_{x} & z = x_1 + 2x_2 - 3x_3 + x_4 \\ \text{s.t.} & x_1 - 2x_2 + 3x_3 + x_4 \leq 3, \\ & x_2 + 2x_3 + 2x_4 \geq -5, \\ & 2x_1 - 3x_2 - 7x_3 - 4x_4 = 2, \\ & x_1 \geq 0, \ x_4 \leq 0. \end{cases}$$

2.
$$\begin{cases} \max\limits_{x} & z = -x_1 + 2x_3 \\ \text{s.t.} & x_1 + x_2 \leq 1, \\ & -x_1 + x_3 = 2, \\ & x_1 \leq 0, \ x_2 \geq 0 \end{cases}$$

3.
$$\begin{cases} \max\limits_{x} & z = c^T x \\ \text{s.t.} & Ax = b, \\ & Bx \geq d, \\ & x \geq 0, \end{cases}$$

where $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times n}$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $d \in \mathbb{R}^p$.

4.
$$\begin{cases} \min\limits_{x} & z = c^T x \\ \text{s.t.} & a \leq Ax \leq b, \\ & 0 \leq x \leq u, \end{cases}$$

where $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $a, b \in \mathbb{R}^m$ $(a \leq b)$ and $u \in \mathbb{R}^n$ $(u \geq 0)$.

Exercise 5.3. Using the duality theorem, verify if the following LP problems are unbounded:

$$(a)\ \begin{cases} \max_x z = 2x_1 + x_2 \\ \text{s.t.}\ \ x_1 - x_2 \leq 4, \\ \qquad x_1 - x_2 \leq 2 \\ \qquad x_1 \geq 0, x_2 \geq 0. \end{cases} \qquad (b)\ \begin{cases} \min_x z = -4x_1 + 2x_2 \\ \text{s.t.}\ \ -x_1 + x_2 \geq 2, \\ \qquad -x_1 + x_2 \geq 1 \\ \qquad x_1 \geq 0, x_2 \geq 0. \end{cases}$$

Explain your answer using the duality theory we studied in class, do not use simplex methods or graphs.

Exercise 5.4. When modeling a portfolio selection problem, we obtain the following LP problem:

$$\max_x z = 0.043x_1 + 0.027x_2 + 0.025x_3 + 0.022x_4 + 0.045x_5$$
$$\text{s.t.}\ \begin{cases} \text{Cash:} & x_1 + x_2 + x_3 + x_4 + x_5 & \leq 10, \\ \text{Government:} & x_2 + x_3 + x_4 & \geq 4, \\ \text{Quality:} & 0.6x_1 + 0.6x_2 - 0.4x_3 - 0.4x_4 + 3.6x_5 & \leq 0, \\ \text{Maturity:} & 4x_1 + 10x_2 - x_3 - 2x_4 - 3x_5 & \leq 0, \\ & x \geq 0. \end{cases}$$

(a) Write down the dual problem of this LP.

(b) Verify that $x^* = (2.1818, 0, 7.3636, 0, 0.4545)^T$ is a feasible solution of the primal problem rounding up to 4 digits after the decimal point.

(c) Using the complementarity slackness theory to check if x^* is an optimal solution of the primal LP. If x^* is optimal, compute an optimal solution for the dual problem.

Note that all the computation must be done approximately up to 4 digits after the decimal point.

Exercise 5.5. Consider the following pair of primal and dual LPs:

$$(\text{Primal})\ \begin{cases} \min_x & z = 35x_1 + 30x_2 + 60x_3 + 50x_4 + 27x_5 + 22x_6 \\ \text{s.t} & x_1 \qquad\quad + 2x_3 + 2x_4 + x_5 + 2x_6 \geq 9, \\ & \qquad x_2 + 3x_3 + x_4 + 3x_5 + 2x_6 \geq 19, \\ & \qquad\qquad\qquad\qquad x \geq 0. \end{cases}$$

$$\text{(Dual)}\quad\begin{cases}\displaystyle\max_{y} & v = 9y_1 + 19y_2\\ \text{s.t} & y_1 \le 35,\\ & y_2 \le 30,\\ & 2y_1 + 3y_2 \le 60,\\ & 2y_1 + y_2 \le 50,\\ & y_1 + 3y_2 \le 27,\\ & 2y_1 + 2y_2 \le 22,\\ & y \ge 0.\end{cases}$$

(a) Suppose that $x^* = (0,0,0,0,5,2)^T$ is feasible solution of the primal problem, verify that this vector x^* is an optimal solution to the primal LP using the complementary slackness condition.

(b) Find the set of all optimal solutions of the dual LP. Does the primal LP have a unique optimal solution, or multiple optimal solutions?

(c) Now suppose that the right hand side constants of the first two primal constraints are changed to 11 and 23 respectively (from the current values of 9 and 19). Does the dual optimal solution(s) you found in (b) remain optimal after the change? If so, use it to find the set of all optimal solutions to the primal LP.

Exercise 5.6. Consider the following LP problem:

$$\begin{cases}\displaystyle\max_{x} & z = 3x_1 + 7x_2 + 5x_3\\ \text{s.t} & x_1 + x_2 + x_3 + s_1 = 50,\\ & 2x_1 + 3x_2 + x_3 + s_2 = 100,\\ & x_1,\ x_2,\ x_3,\ s_1,\ s_2 \ge 0.\end{cases}$$

After using the simplex method to solve it, we obtain the following tableau, in which x_3 and x_2 are basic variables. For each question below, you should briefly write down the details of your answer to support your claim.

z	x_1	x_2	x_3	s_1	s_2	RHS	Basic var
1	3	0	0	4	1	300	$z = 300$
0	0.5	0	1	1.5	-0.5	25	$x_3 = 25$
0	0.5	1	0	-0.5	0.5	25	$x_2 = 25$

$\max z; x, s \ge 0$

1. Write down the basic optimal solution obtained from this simplex tableau, and the corresponding optimal value of this LP. Explain if this LP has a unique or multiple solutions.
2. Suppose that the objective function changes to $\max z = (3+\Delta)x_1 + 7x_2 + 5x_3$. How does this change affect the simplex tableau in which x_3 and x_2 are basic variables? For what range of Δ does that tableau show an optimal solution? Write down an optimal solution and the optimal value for all Δ belonging to that range. Does $\Delta = 4$ belong to that range? Find an optimal solution and the optimal value for $\Delta = 4$.
3. Suppose that the objective function changes to $\max z = 3x_1 + (7+\Delta)x_2 + 5x_3$. Write down the new simplex tableau in which x_3 and x_2 are basic variables. For what range of Δ does that tableau show an optimal solution? Write down an optimal solution and a formula for the optimal value for Δ in that range. Does $\Delta = 4$ belong to that range? Find an optimal solution and the optimal value for $\Delta = 4$.
4. Suppose that the objective function changes to $\max z = 3x_1 + 7x_2 + (5+\Delta)x_3 + \Theta s_1$. Write down the new simplex tableau in which x_3 and x_2 are basic variables. For which Δ and Θ does that tableau show an optimal solution? Write down an optimal solution and a formula for the optimal value for Δ belonging to that range. Does $(\Delta, \Theta) = (2, 2)$ belong to that range?
5. Let the objective function be the original function. Suppose that the right hand side constant of the first constraint changes from 50 to 50+Δ. For what range of Δ does the basis $\{x_3, x_2\}$ continue to be optimal? What is the optimal value when Δ belongs to this range?
6. Let the objective function be the original function. Suppose that the right hand side constant of the second constraint changes from 100 to 100+Δ. For what range of Δ does the basis $\{x_3, x_2\}$ continue to be optimal? What is the optimal value when Δ belongs to this range?

Exercise 5.7. Consider the following linear program:

$$\begin{cases} \max_x z = 3x_1 + x_2 + 4x_3 + x_4 \\ 6x_1 + 3x_2 + 5x_3 + 4x_4 \le 25 \\ 3x_1 + 2x_2 + 3x_3 + x_4 \le 15 \\ 3x_1 + 4x_2 + 5x_3 + 2x_4 \le 20 \\ x_1 \ge 0\ x_2 \ge 0\ x_3 \ge 0\ x_4 \ge 0\,. \end{cases}$$

By adding three slack variables x_5, x_6 and x_7 to the first, second, and third constraints, we convert the above LP into canonical form. Then we apply the simplex method to it and obtain the following simplex tableau, in which the value in the RHS entry of row 0 is missing. For each question below, you should briefly write down the details of your answer to support your claim.

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS	Basic Var
1	0	2	0	1	$1/5$	0	$3/5$	?	$z = ?$
0	1	$-1/3$	0	$2/3$	$1/3$	0	$-1/3$	$5/3$	$x_1 = 5/3$
0	0	0	0	-1	$-2/5$	1	$-1/5$	1	$x_6 = 1$
0	0	1	1	0	$-1/5$	0	$2/5$	3	$x_3 = 3$

1. What is the value of the RHS entry in row 0 of the above tableau? Justify your answer.
2. Suppose that the objective coefficient of x_3 (currently 4) changes to $4 + \Delta$. For what range of Δ does the current optimal basis $\{x_1, x_6, x_3\}$ continue to be optimal? What is the optimal value for Δ in this range?
3. Let the objective function be the original function. Suppose that the right hand side constants for the first and second constraints change to $25 + \Delta$ and $15 + \Delta$ simultaneously (for example if $\Delta = 1$ then the RHS constants in the first two constraints become 26 and 16 respectively). For what range of Δ does the current optimal basis $\{x_1, x_6, x_3\}$ continue to be optimal? What is the optimal value for Δ in this range?

Chapter 6
Software for Linear Programming

6.1 Overview

Computer software for linear programming is perhaps the most well developed area in optimization software, and is substantially reliable in practice. This chapter briefly presents an overview on computer software for LPs. We distinguish between two types of computer software.

- **LP solvers:** Roughly speaking, an LP solver is a computer program that implements one or many solution methods to solve LPs in a given form. There are at least two kinds of methods that are often implemented in an LP solver: simplex methods and interior-point methods. Nowadays, the list of LP solvers is getting longer and longer. For instance, CPLEX or `linprog` are LP solvers.
- **Optimization modeling languages:** Optimization modeling languages provide interfaces between model developers and optimization solvers. Modeling languages are "developed with the aim of allowing users to express LP and other optimization problems in a natural, algebraic form similar to the original mathematical expressions," as put in [8]. Data and models can be separated, making it easy to audit and adapt the models. The model developers do not have to arrange data into matrices or vectors, as this task will be done by the compiler. The compiler will translate the model into an acceptable format and pass it to a solver, which receives the problem and solves it. An optimization modeling software is often connected to various solvers ((either built-in or third-party solvers)) that handle different types of problems.

Modeling languages are good choices for model developers who deal with optimization problems of moderate to large-scale sizes and do not intend to write more codes than just the models. By using a modeling language in this course, we can focus on how to build the mathematical models for various types of problems, rather than spending too much effort on coding-specific issues. We do need to master the basics of a modeling language to be able to use it at an introductory level.

Let us start discussing some common LP solvers before using GAMS as a modeling language to illustrate the various applications. While many LP solvers are available, we focus on LP solvers that are readily accessible to most users.

6.2 LP solvers

We discuss Excel in Microsoft Office, `linprog` in Matlab, CPLEX, and some other LP solvers.

6.2.1 Excel

Let us start with Excel, a Microsoft's software program. The Solver add-in of Microsoft Excel is an easy-to-use tool and is widely used in business worldwide. We illustrate how to use this tool with a small example.

Problem description: A company produces two types of picture frames: each type-1 frame uses 2 hours of labor and 1 ounce of metal and brings \$2.25 in profit, and each type-2 frame uses 1 hour of labor and 2 ounces of metal and brings \$2.60 in profit. Each week 4000 labor hours and 5000 ounces of metal are available, and the company aims to maximize its weekly profit with an optimal production plan.

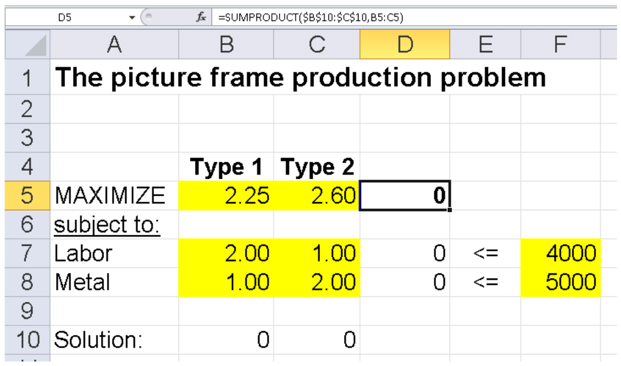
Problem modeling: To model the problem as a linear program, we use x_i to denote the number of type i picture frames produced weekly for $i = 1, 2$, and formulate the following LP:

$$\begin{cases} \max\limits_{x} & z = 2.25x_1 + 2.60x_2 \\ \text{s.t.} & 2x_1 + x_2 \leq 4000, \\ & x_1 + 2x_2 \leq 5000, \\ & x \geq 0. \end{cases} \tag{6.1}$$

Solving this LP by Excel: When using Excel Solver for the first time, it is often necessary to activate it. The procedure depends on the version of Excel, and it often involves accessing the "Add-Ins" menu in Excel "Options" to select the "Solver

Add-In" to be active. Once activated, one can set up the problem in Excel as shown in Figure 6.1.

We enter data of the problem into the cells B5, C5, B7, C7, B8, C8, F7 and F8, and use cells B10 and C10 to store an arbitrarily chosen starting solution $(0,0)$. We use cell D5 to store the objective value, which is given by the formula `=SUMPRODUCT(B5:C5,$B$10:$C$10)`. The \$ signs are to anchor locations of cells B10 and C10 when copying this formula to cells below. We use cells D7 and D8 for values of the left hand sides of the labor and metal constraints respectively, given by formulas `=SUMPRODUCT(B7:C7,$B$10:$C$10)` and `=SUMPRODUCT(B8:C8,$B$10:$C$10)` respectively. The $\leq$ signs in E7 and E8 are for display purposes and will be ignored by Excel Solver.



D5 =SUMPRODUCT($B$10:$C$10,B5:C5)

	A	B	C	D	E	F
1	**The picture frame production problem**					
2						
3						
4		**Type 1**	**Type 2**			
5	MAXIMIZE	2.25	2.60	**0**		
6	subject to:					
7	Labor	2.00	1.00	0	<=	4000
8	Metal	1.00	2.00	0	<=	5000
9						
10	Solution:	0	0			

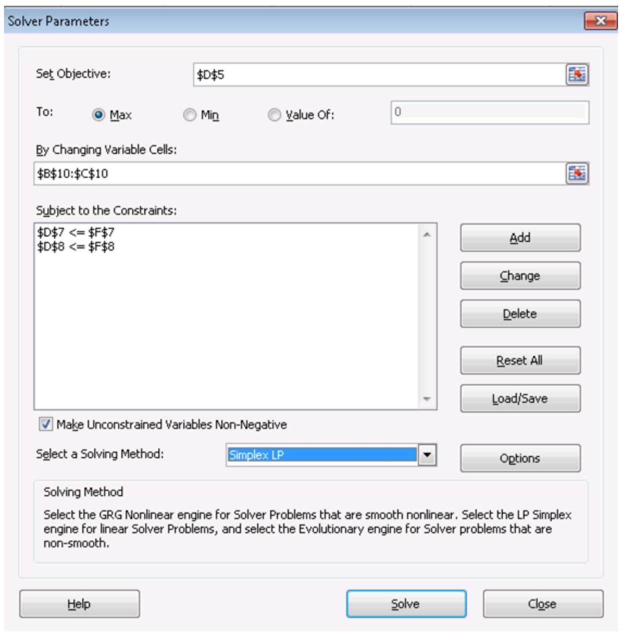
Fig. 6.1 Model and solve the **picture frame production problem** in Excel.

To start using the Excel Solver in Excel 2010, click **Solver** under the **Data** menu to open the Solver Parameters window.

- In the **Set Objective** box, enter the cell which contains the formula for computing the objective value (cell D5 for the example).
- Select **max** in the line below since we want to maximize the profit. For the **By Changing Variable Cells**, select the cells which contain the values of the decision variables (cells B10:C10 for the example).
- Next, add constraints one by one: select the cell that contains the left hand side formula as the **Cell Reference**, choose $<=$, $>=$, or $=$ based on the type of constraint, and enter the cell that contains the right hand side in the **Constraint** box.

- Following that, check the box of making unconstrained variables non-negative, and select Simplex LP as the solving method.
- Finally, click **Solve**, and the Solver Results box will tell us if Solver finds a solution. If Solver finds a solution, ensure that "Keep Solver Solution" is marked and click **OK**.

The optimal solutions for the decision variables will be placed in the Changing Cells.


Solver Parameters

Set Objective: $D$5

To: [✓ Max] Min Value Of: 0

By Changing Variable Cells: $B$10:$C$10

Subject to the Constraints:
$D$7 <= $F$7
$D$8 <= $F$8

[✓] Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP

Solving Method
Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Fig. 6.2 Completed Excel Solver dialog box.

Final remarks: As we can see, the Excel Solver is easy and intuitive. With the wide availability of Excel, the Solver add-in is extensively used for simple optimization tasks. However, in general, it is not convenient to use Excel Solver for large-scale optimization models. It is hard to model complicated algebraic constraints using Excel Solver, and its computational performance is not as competitive as software specialized in optimization. Moreover, any dimensional or structural changes in the data would demand a major revision of the Solver parameters.

6.2.2 Matlab's LP solver

Matlab provides several routines for optimization in the Optimization Toolbox. It is often integrated in Matlab when we install it. More information about Matlab can be found at https://www.mathworks.com. The LP solver, called `linprog`, in the Optimization Toolbox of Matlab can solve relatively large-scale LPs. It accepts matrices and vectors as inputs. To see how we can input data of an LP, from the Command Window of Matlab, type

```
>> help linprog
```

for instructions on how to use this function. Below, we use `linprog` to solve the above picture frame example. We can use the following commands in the Command Window of Matlab:

```
>> c = [-2.25; -2.60];
>> A = [2 1; 1 2];
>> b = [4000;5000];
>> lb = zeros(2,1);
>> [x, z] = linprog(c, A, b, [], [], lb)
```

The first to the third lines enter the input data for c, A and b, respectively. The fourth line give a lower bound on x, which is 0. This imposes that $x \geq 0$. The last command calls `linprog` to minimize the linear objective function specified by the vector c, subject to linear constraints defined by the matrix A and the right hand side vector b and the lower bound lb on the variables. The empty matrices [] in the inputs indicate that there are no equality constraints. The outputs will be $x = (1000, 2000)^T$ with the optimal value $z = -7450$ as we can see below:

```
x =
   1.0e+03 *
   0.999999982545405
   2.000000004206820
z =
    -7.449999971664894e+03
```

We note that `linprog` also provides several options so that we can choose different methods, accuracy and monitor outputs to meet our expectation.

As illustrated by the above example, in order to use Matlab for solving LPs, one needs to arrange the input data (A,b,c) into matrices and vectors. For nonlinear optimization problems that involve nonlinear objective functions or constraints, one needs to define Matlab functions and pass them as function arguments in the inputs. For large-scale problems with complicated constraints, specifying the matrices and vectors can become a cumbersome task that makes it inconvenient to audit the codes or adapt the codes in response to changes in the model. However, this can be done by providing an interface on top of the solver to do this preprocessing step. Matlab provides several tools and routines to do such a task.

6.2.3 Other LP solvers

There are many other solvers for LPs. Below is a non-exhaustive list of LP solvers due to H. Mittelmann (see http://plato.asu.edu/ftp/lpsimp.html).

- **CPLEX:** CPLEX is a high-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming. The name came from Simplex method implemented in C. This software program is currently owned by IBM, and is still active. CPLEX has a long history of development. It was founded by Robert E. Bixby and was sold commercially in 1988 by CPLEX Optimization Inc. More information about this solver can be found at http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html. CPLEX is available free of charge for academic use and for students.
- **Gurobi:** The Gurobi Optimizer is a new mathematical programming solver and is able to handle all major problem types, including linear programming. More information can be found here: http://www.gurobi.com. Gurobi is available free of charge for academic use and for students. Gurobi is named from its founders: Zonghao **Gu**, Edward **Ro**thberg and Robert **Bi**xby (also the father of CPLEX). Gurobi is very fast and can solve really big problems.
- **Mosek:** MOSEK is a tool for solving mathematical optimization problems such as LPs, quadratic programming, conic problems and mixed integer programming. MOSEK ApS was established in 1997 by Erling D. Andersen and Knud D. Andersen from Denmark. This software program is available at http://www.mosek.com and is free-of-charge for academic use.

- **Xpress:** XPRESS-Optimizer is a commercial optimization solver for many mathematical programming problems including linear programming (LP). More information about this package can be found at http://www.fico.com/en/products/fico-xpress-optimization-suite.
- **CLP:** CLP (Coin-or linear programming) is an open-source linear programming solver written in C++. More information can be found at https://projects.coin-or.org/Clp.
- **Google-GLOP:** This is also an LP solver developed by Google. See https://developers.google.com/optimization/lp/ for more detail.
- **SoPlex:** SoPlex is an optimization package for solving linear programs based on an advanced implementation of the primal and dual revised simplex algorithm. This software program is available at http://soplex.zib.de.
- **LP_SOLVE:** Lp_solve is a Mixed Integer Linear Programming (MILP) solver. It is an open-source package implemented in C. This package was originally developed by Michel Berkelaar at Eindhoven University of Technology. The code and information are available at http://lpsolve.sourceforge.net.
- **GLPK:** The GNU Linear Programming Kit (GLPK) is a software package intended for solving large-scale linear programs and other problems. It is an open-source package, and can be found at http://www.gnu.org/software/glpk/glpk.html. This package was developed by Andrew O. Makhorin, and remains active.

6.3 Algebraic modeling languages

AMLs (Algebraic modeling languages) are high-level computer programming languages for expressing and solving complex, large-scale mathematical optimization problems. The syntax adopted by an AML is often close to the natural language syntax used to describe a problem. There exist many optimization modeling languages including LINGO, AIMMS, AMPL, GAMS, OPL and Mosel. These software programs often support multiple optimization solvers including those discussed above. Below are more detailed information about some of these languages.

- **LINGO:** According to its website, LINGO is a comprehensive tool designed to build and solve linear, nonlinear (convex and nonconvex/global), quadratic, quadratically constrained, second-order cone, semidefinite, stochastic, and Integer optimization models. LINGO provides a integrated package that includes a language for expressing optimization models, a full featured environment for building and editing problems, and a set of built-in solvers.

- **AIMMS:** According to the website https://aimms.com, the AIMMS (Publishing and Remote Optimization) platform is a collaboration platform in the field of optimization. It enables the use of AIMMS optimization applications through a web browser and offers a personal Enterprise Optimization App Store.
- **AMPL:** A Mathematical Programming Language (AMPL) is an algebraic modeling language to describe and solve highly complex, large-scale optimization problems. It first appeared in 1985 and was developed by Robert Fourer, David Gay, and Brian Kernighan at Bell Laboratories. It is still widely used in many areas of operations research. More information can be found at http://www.ampl.com.
- **GAMS:** GAMS (General Algebraic Modeling System) is a high-level modeling system for mathematical optimization. It is designed for modeling and solving linear, nonlinear, and mixed-integer optimization problems. More information can be founded at https://www.gams.com. We will use this software program extensively in this course.
- **OPL:** OPL was developed by IBM (http://www-01.ibm.com/software/commerce/optimization/modeling/index.htm It provides a natural mathematical description of optimization models. Its high-level syntax supports expressions needed to model and solve problems using both mathematical programming and constraint programming.
- **Mosel:** Similar to OPL, Mosel is a modeling language developed by a company called FICO (the owner of XPRESS). More information about this software can be found at http://www.fico.com/en/products/fico-xpress-optimization-suite.

Other computer software packages for solving optimization problems include APMonitor, ASCEND, OPTMODEL from SAS, Pyomo, and OptimJ, and so on.

6.4 GAMS: A General Algebraic Modeling System

GAMS was initially developed in the mid-1970s by Alex Meeraus and Jan Bisschop working at the World Bank. It later became a commercial product owned by the GAMS Development Corporation. From its website http://www.gams.com one can download free versions of GAMS.

In this section, we use the **picture frame example** described at the beginning of Section 6.2 to illustrate how to solve an optimization model in GAMS. We have two files, `picframe1.gms` and `picframe2.gms`, representing two different modeling styles that solve the same LP problem described in the problem formulation (6.1).

Each GAMS model is a collection of GAMS statements. Every statement is terminated with a semicolon ‘;’ (experienced users can omit the semicolon when the next statement begins with a keyword). The most basic components in a GAMS model are:

- Variables
- Equations
- Model and solve statements.

Let us describe them in detail in the following subsections.

6.4.1 Variables: declaration and type assignment

When using GAMS to solve an optimization problem, we need to declare variables of the optimization problem in GAMS. In particular, we need to explicitly declare a variable to denote the quantity to be optimized. The last variable is called the **objective variable**.

Declaration: To declare a variable in GAMS, start with the keyword **variable** followed by the name for the variable. We can also add some descriptive text after the name. The following statements declare three variables with names `x1`, `x2` and `profit` respectively, where `profit` is the objective variable. The words between the quotation marks are descriptive texts that serve as comments. Experienced users can omit those quotation marks.

```
variable x1 "type 1 picture frames";
variable x2 "type 2 picture frames";
variable profit "total profit";
```

Alternatively, we can declare multiple variables in a single statement:

```
variables
     x1 "type 1 picture frames",
     x2 "type 1 picture frames",
            profit "total profit";
```

Types of variables: Every GAMS variable is associated with a `type`. In this course we will use variables of the following five types.

- **Free** (the default type): a variable that can take any real value.
- **Positive**: a variable that can take any nonnegative value (GAMS uses the word “positive” to mean “nonnegative,” so a positive variable in GAMS is allowed to be zero).
- **Negative**: a variable that can take any nonpositive value (a negative variable in GAMS is allowed to be zero).
- **Binary**: a variable that is either 0 or 1.
- **Integer**: a variable that can only take integer values between specified bounds (the bounds are 0 and 100 by default and can be changed).

We can declare the variables first and then assign types to them in a separate statement:

```
        positive variables x1, x2;
```

We can also declare variables and assign their type in one statement. Note that the objective variable **must** be a free variable in a GAMS optimization model.

6.4.2 Equations: declaration and definition

To solve an optimization problem in GAMS, we need to specify the constraints and the objective function by **equations**. The keyword **equation** has a broad meaning in GAMS; it can mean either an equality or an inequality (see the example below). A GAMS equation defined with a common name can also stand for a family of equations of the same structure.

A GAMS equation needs to be declared first, and then defined in a separate statement. To declare an equation is to give it a name. To define an equation is to specify the relations between variables in this equation.

Declaration: To declare an equation, start with the keyword **equation** followed by the name for the equation. As with the declaration of variables, we can add some optional descriptive text after the name. The following statement declares three equations named `obj`, `labor` and `metal`.

```
        equations
```

```
obj    "max total profit",
labor  "labor hours",
metal  "metal in oz";
```

Define equations: To define an equation, start with the name of the equation declared in a preceding statement, put two dots '..' after the name, write down the left-hand-side expression and the right-hand-side expression, and put a relational operator between the two expressions. A relational operator can be one of the following three types:

- **=g= Greater than**: LHS (the left-hand side) must be greater than or equal to RHS (the right-hand side).
- **=e= Equality**: RHS must equal LHS.
- **=l= Less than**: LHS must be less than or equal to RHS.

Keep in mind that we **must** use one of the above operators in defining GAMS equations instead of the usual symbols $\leq$, $=$, and $\geq$ in mathematical formulations. It is fine to use upper case letters (=G=, =E=, and =L=).

The following three statements define the three equations declared above, with the equation `obj` specifying the objective function, and equations `labor` and `metal` specifying the two constraints.

```
obj.. profit =e= 2.25*x1 + 2.60*x2;
labor.. 2*x1 + x2 =l= 4000;
metal.. 1*x1 +2*x2 =l= 5000;
```

6.4.3 Model and solve statements

The model statement is used to collect equations into groups and to label them so that they can be solved (advanced users may create several models in one GAMS file).

Declaration: To declare a model, start with the keyword **model** followed by the name for the model, followed by a list of equation names enclosed in slashes. For example, the statement below declares a model named `picframe`, which collects the three equations declared before.

```
model picframe /obj,labor,metal/;
```

In fact, if all previously defined equations are to be included in a model, we can use `/all/` in place of the explicit list. That is

```
model picframe /all/;
```

Call solver: Once a model has been declared and assigned equations, we are ready to call the solver. This is done with a solve statement. A "solve statement" starts with the keyword **solve** followed by the name of the model to be solved, then another keyword **using** followed by the model type, then the keyword **minimizing** or **maximizing**, and ends with the name of the objective variable. That is

```
solve Model_Name using Model_Type maximizing [minimizing] Objective_variable;
```

Below is the solve statement in the example:

```
solve picframe using lp maximizing profit;
```

where `picframe` is the name of the model just declared, "lp" is the model type, and `profit` is the name of the objective variable declared earlier.

Types of models: The model type "lp" stands for linear programming. GAMS can be used to solve various types of models, including

- **lp** (or LP): for linear programming;
- **qcp** (or QCP): for quadratic constraint programming;
- **nlp** (or NLP): for nonlinear programming;
- **mip** (or MIP): for mixed integer programming, etc.

In summary, we obtain the following gams model, and save it into a `*.gms` file called `picframe1.gms`:

```
*picframe1.gms: This is a GAMS model for example (1) above.
*Variable declaration
variables
    x1 "type 1 picture frames",
    x2 "type 2 picture frames",
    profit "total profit";
*Assign type of variables
positive variables x1, x2;
*Equation declaration
equations
```

```
      obj    "max total profit",
      labor  "labor hours",
      metal  "metal in oz";
  *Equation definition
    obj..profit =e= 2.25*x1 + 2.60*x2;
    labor..2*x1 + x2 =l= 4000;
    metal..1*x1 +2*x2 =l= 5000;
  *Model and solve statements
  model picframe /all/;
  solve picframe using lp maximizing profit;
```

6.4.4 GAMS outputs

To compile and execute a GAMS file, select “Run” in “File” menu in GAMS-IDE, or click the “Run” button on the toolbar. In response to that command, GAMS will compile the file and call a solver to solve the model if it does not find any syntax errors. GAMS has integrated many solvers developed by various people in industry and academia. We can see the list of solvers in the “File $\rightarrow$ Options $\rightarrow$ Solvers” window. Each solver handles different types of models, and each type of models has a default solver. For example, the system default solver to solving LP models is CPLEX.

During the compilation / execution and solution phases, a process window will pop up to show the progress of a GAMS job. If the file contains syntax errors, error messages will be shown in red with an indication where the error occurred.

After closing the process window, we will find the “`lst`” file opened automatically in GAMS-IDE. The `*.lst` file contains output produced from the GAMS run. It is saved in the currently active **project directory**. GAMS-IDE has a default project directory under which all output files will be saved. We can reset the directory by creating a new project at a convenient location. Information we can find in the `*.lst` file include:

(a) Error messages (if there is any error).

(b) The solution report:

- Solver status: the desired status is 1: normal completion; any other solver status indicates something wrong with the solution process.

- Model status: for an LP model there are three possible model statuses: 1 - optimal, 3 - unbounded, 4 - infeasible.
- Objective value: the optimal value found by the solver.
- The **levels** of variables: This indicates the values of the **optimal solution** found by the solver. A single dot '.' in the `*.lst` file represents a zero value.
- The "marginal" column for equations: This indicates the values of dual variables.

6.4.5 Language items

This subsection collects small tips for using the GAMS language correctly. Detailed instructions and discussion can be found in Section 3.4 of `GAMS User Guide`, which is accessible from the "Help" menu in GAMS-IDE.

- **Names:** Names given to GAMS variables, equations, models, etc are all called **identifiers**. There are certain rules one needs to follow in picking an identifier. For example, it must start with a letter followed by more letters or digits. It is worth mentioning that GAMS is case-insensitive. Hence, `x1` and `X1` can be used interchangeably in GAMS.
- **Keywords:** Keywords (also called reserved words) in GAMS are words with predefined meanings. It is not permitted to use a keyword as an identifier. For example, we cannot name a variable just as "variable" or "VARIABLE", because the words "variable" and "VARIABLE" are reserved as keywords by GAMS. The complete list of reserved words is in table 3.3 of the Guide.
- **Separation:** As we know semicolons are used to separate GAMS statements. Inside a statement, one can use commas as delimiters to separate names of different variables or equations.
- **Comments:** Finally, we can add a line of comments by putting the symbol "*" as the first character of that line. We can also create a paragraph of comments between the options `$ontext` and `$offtext`. Comments are ignored when the file is compiled.

6.4.6 Set declaration

An important common feature of optimization modeling languages is the use of sets to collect indices. This feature allows the model to be succinctly stated and easily read, making it convenient to model large-scale problems.

To declare a set, start with the keyword **set** or **SET** followed by the name of the set, followed by members (also called elements) of the set between two slashes. The following statement in `picframe2.gms` declares a set named `I` that consists of two elements "type-1" and "type-2".

```
        set I /type-1, type-2/;
```

Note that elements of a set are labels but not numbers. To emphasize this, we can put the labels in quotes in the above statement:

```
        set I /'type-1', 'type-2'/;
```

We note that we can define multiple sets at the same time. For example,

```
  SETS I "variables" /type-1, type-2/, J "constraints" /labor, metal/;
```

6.4.7 Scalars and parameters

To declare scalars and parameters in GAMS is to give names to the various numbers that appear in a problem formulation. By using scalars and parameters instead of specific numbers in the definitions of GAMS equations, we can separate data from the equations, making the model easier to audit, adapt and manage.

(a) Scalars: Scalars are used for single data entries that are not associated with any set. We can declare a scalar by using the keyword **scalar** followed by the name of the scalar, followed by the value of the scalar between two slashes. As with other GAMS entities, we can put optional descriptive text after the name of the scalar. The following statement in `picframe2.gms` declares two scalars, named `NumLabor` and `NumMetal` respectively, with values 4000 and 5000 respectively.

```
        scalar NumLabor "Number of labor hours available" /4000/,
               NumMetal "Metal (oz) available" /5000/;
```

(b) Parameters: Parameters are used for data over index sets. The set over which a parameter is indexed is called the **domain** of the parameter. To declare a parameter, start with the keyword **parameter** followed by the name of the parameter, put the domain of the parameter between parentheses, then assign values of the parameter for each element in the domain between two slashes. The following statement declares parameters named `sellingPrice`, `laborUse` and `metalUse` over the set `I`, and assigns their values. For example, `sellingPrice("type-1")` takes the value 2.25 and `metalUse("type-2")` takes the value 2.

```
parameters sellingPrice(I) /type-1 2.25, type-2 2.60/,
           laborUse(I) /type-1 2, type-2 1/,
           metalUse(I) /type-1 1, type-2 2/;
```

6.4.8 Variables over sets

One can declare a variable over a set, by putting the name of the set in parentheses after the name of the variable in the statement that declares the variable. The set is called the domain of that variable. To declare a variable over a set is to name a group of variables using elements of the set as indices. The following statement from `picframe2.gms` declares a variable named `x` over the set `I`. With this declaration, we have two variables `x("type-1")` and `x("type-2")` to be used in equations.

```
positive variables x(I) "number of picture frames";
```

If we declare the variable `x` first and then assign type to it in a separate statement, then we should not put its domain in the second statement. Example:

```
variables x(I) "number of picture frames";
positive variable x;
```

6.4.9 Equations with scalars, parameters and variables over sets

We can use scalars and parameters in defining equations. Recall the equations defined in `picframe1.gms`:

```
obj.. profit =e= 2.25*x1 + 2.60*x2;
labor.. 2*x1 + x2 =l= 4000;
metal.. 1*x1 +2*x2 =l= 5000;
```

After replacing the numbers by their names and replacing variables `x1` and `x2` by `x("type-1")` and `x("type-2")`, we obtain the following definitions for the equations:

```
obj.. profit =e= sellingPrice("type-1")*x("type-1") + sellingPrice("type-2")*x("type-2");
labor.. laborUsed("type-1")*x("type-1") + laborUsed("type-2")*x("type-2") =l= NumLabor;
metal.. metalUsed("type-1")*x("type-1") +  metalUsed("type-2")*x("type-2") =l= NumMetal;
```

The summations in above equations can be written compactly using the GAMS summation notation. The basic format is `Sum(index of summation, summand).` With this notation the above equations can be written as

```
obj.. profit =e=   sum(I, sellingPrice(I)*x(I));
labor.. sum(I, laborUsed(I)*x(I)) =l= NumLabor;
metal.. sum(I, metalUsed(I)*x(I)) =l= NumMetal;
```

In the above statements, the set `I` is the index of summation.
The expression `sum(I, sellingPrice(I)*x(I))` gives the sum of `sellingPrice(I)*x(I)` over all indices in `I`, and is therefore an equivalent way to express the sum

```
sellingPrice("type-1")*x("type-1") + sellingPrice("type-2")*x("type-2")
```

Clearly, if the number of elements in `I` is large, it is much more convenient to use the command `sum`.

A complete script of the file `picframe3.gms` in GAMS is given below, including comments.

```
$ontext
  A company produces two picture frame types.
  Each Type 1 frame uses 2 hr labor and 1 oz metal, and brings $2.25 of profit.
  Each Type 2 frame uses 1 hr labor and 2 oz metal, and brings $2.60 of profit
  There are 4000 hours labor and 5000 oz metal available each week. How many
  frames of each type should this company produce to maximize weekly
  profit?
$offtext
*set declaration
set I /type-1, type-2/;
*scalars and parameters
scalar
    NumLabor /4000/,
    NumMetal /5000/;
parameters
    price(I)         /type-1 2.25, type-2 2.60/,
    laborUse(I)  /type-1 2,       type-2 1/,
    metalUse(I) /type-1 1,       type-2 2/;
*variable declaration and type assignment
free variable
    profit "total profit";
positive variables
    x(I) "number of picture frames";
*equation declaration
equations
    obj    "max total profit",
    labor  "labor hours",
    metal  "metal in oz";
*equation definition
    obj.. profit =e=   sum(I, price(I)*x(I));
    labor.. sum(I, laborUse(I)*x(I)) =l= NumLabor;
    metal.. sum(I, metalUse(I)*x(I)) =l= NumMetal;
model picframe /all/;
solve picframe using lp maximizing profit;
```

If we call GAMS to solve this problem, we will obtain an output file called `picframe3.lst` as below.

```
  S O L V E      S U M M A R Y
     MODEL   picframe            OBJECTIVE  profit
     TYPE    LP                  DIRECTION  MAXIMIZE
     SOLVER  CPLEX               FROM LINE  40
**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      1 Optimal
**** OBJECTIVE VALUE             7450.0000
 RESOURCE USAGE, LIMIT          0.016      1000.000
 ITERATION COUNT, LIMIT         2    2000000000
IBM ILOG CPLEX   24.4.6 r52609 Released Jun 26, 2015 WEI x86 64bit/MS Windows
Cplex 12.6.2.0
Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.01 ticks)
Optimal solution found.
Objective :        7450.000000
                       LOWER     LEVEL     UPPER    MARGINAL
---- EQU obj             .         .         .        1.000
---- EQU labor          -INF   4000.000  4000.000     0.633
---- EQU metal          -INF   5000.000  5000.000     0.983
  obj  max total profit
  labor  labor hours
  metal  metal in oz
                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR profit         -INF   7450.000     +INF       .
  profit  total profit
---- VAR x  number of picture frames
          LOWER     LEVEL     UPPER    MARGINAL
type-1      .     1000.000     +INF       .
```

```
type-2      .     2000.000     +INF       .
**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
EXECUTION TIME       =        0.000 SECONDS      2 MB  24.4.6 r52609 WEX-WEI
USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000
```

6.5 Exercises

Exercise 6.1. Young MBA Erica Cudahy plans to invest up to \$1000 to a financial market. She can invest her money in stocks and loans. Each dollar invested in stocks yields 10 cents of profit, and each dollar invested in a loan yields 15 cents of profit. At least 30% of all money invested must be in stocks, and at least \$400 must be in loans. Formulate an LP problem to maximize total profit earned from Erica's investment.

Create a GAMS file named `Erica.gms`. In the GAMS code, declare two positive variables, *stock* and *loan*, to denote the amount of money to put in stock and loan respectively. Also declare a free variable called *profit* to denote the total profit. Model and solve as an LP.

Exercise 6.2. Cuppa Coffee Company mixes specialty coffee blends to sell to SmartBux, a small chain of coffee shops. The ingredients for their specialty coffee are the following beans:

Bean Type	Cost per pound	Available amount (pounds)
Columbian	\$1.00	550
Brazilian	\$0.85	450
Sumatran	\$1.55	650

Cuppa's products are:

- **Robust Joe** must consist of 60%–75% Sumatran beans and at least 10% Columbian beans. Each pound of Robust Joe can be sold to SmartBux for \$4.25.
- **Light Joe** must consist of 50%–60% Brazilian beans and no more than 20% Sumatran beans. Each pound of Light Joe can be sold to SmartBux for \$3.95.

Formulate an LP problem to maximize total profit. To check the correctness of your solution, we provide the optimal value, which is the maximum profit, to be \$4902.5.

Create a GAMS file named `cuppacoffee.gms`. In the GAMS code, declare

```
SETS I /Columbian, Brazilian, Sumatran/, J /Robust, Light/;
```

as two sets of indices, and then declare a variable `x` over `(I,J)`. Declare the bean supply, purchase prices and coffee prices as parameters over `I` or `J`. For simplicity, you can use numbers for percentages in the equations (without declaring them as scalars). Example:

```
equations  RS1 "Robust has at least 60 percent of sumatran";
    RS1.. x("Sumatran","Robust")  =g= 0.6* sum(I, x(I,"Robust"));
```

Exercise 6.3. James Beerd bakes two types of cakes: cheesecakes and black forest cakes. During any month, he can bake at most 65 cakes in total. The costs per cake and the demands for cakes, which must be met in time, are given in the following table.

	Month 1		Month 2		Month 3	
Item	Demand	Cost/cake($)	Demand	Cost/cake($)	Demand	Cost/cake($)
Cheesecake	40	3.00	30	3.40	20	3.80
Black Forest	20	2.50	30	2.80	10	3.40

We assume that cakes baked during a month can be used to meet demand for this month. At the end of each month (after all cakes have been baked and the current month's demand has been satisfied), a holding cost of 50 cents per cheesecake and 40 cents per black forest cake is incurred for cakes left in inventory. Those cakes can be used to satisfy future demand.

Formulate an LP problem to minimize the total cost of meeting the next three months' demands. Solve it using GAMS. Create a GAMS file named `cake.gms`. Declare two sets in the GAMS code:

```
SETS i /cc, bf/, t /1*3/;
```

Declare the holding costs at a parameter over `i`, and the demands and production costs as tables over `(i,t)`. Then declare positive variables `x(i,t)` and `a(i,t)`, to represent numbers of each type of cakes to make each month, and numbers of each type of cakes left in inventory at the end of each month respectively. To check the correctness of your solution: the optimal value is \$464.5.

Chapter 7
Applications of Linear Programming

7.1 Overview

Linear programming has many applications in fields such as computer science, operations research, engineering, data science, economics, and finance. We can classify LP applications into two categories: direct and indirect applications. Chebyshev

7.1.1 Direct applications

In direct applications, we model the practical problem into an LP or a sequence of LPs. Although these LP models only use linear functions to represent the objective function and all the constraints, they are still widely used in various applications. Below is a non-exhaustive list of applications of LPs and possibly mixed integer programs (due to R. E. Bixby [4]):

- **Classical problems in applied mathematics:** Weber's problem, Chebyshev approximation, Chebyshev center, and the ℓ_1-norm approximation.
- **Engineering:** Synthesis of filters, trust topology design, sparse signal recovery, and model predictive control (MPC) [1].
- **Statistics and machine learning:** Classification, data fitting, robust LASSO, support vector machine, and sparse/structural optimization [6].
- **Transportation:** Vehicle routing, freight vehicle scheduling and assignment, depot/warehouse allocation, public transportation system operation, maintenance scheduling, and gate scheduling.
- **Industry and manufacturing:** Plant production scheduling and logistic, gasoline and chemical blending, coal blending, mining operations management,

product mix planning, manufacturing scheduling, inventory management, personnel scheduling, oil and gas production scheduling, food blending, food transportation logistics, pattern layout and cutting, production scheduling, inventory planning, and waste water recycling.

- **Finance:** Portfolio selection and optimization, cash management, lease analysis, capital budgeting and rationing, bank financial planning, accounting allocation, etc.
- **Agriculture and forestry:** Farm land management, agricultural pricing models, crop and product mix decision models, product distribution, forest land management, and planting and harvesting models
- **Public utilities and natural resources:** Power generator scheduling, natural gas distribution planning, water resource management, and public water transportation models.
- **Communications and computing:** Circuit board (VLSI) layout, logical circuit design, complex computer graphics, curve fitting, computer system capacity planning, multiprocessor scheduling, telecommunications scheduling, and telephone operator scheduling.
- **Health care:** Staff scheduling, hospital layout, health cost reimbursement, and ambulance scheduling.
- **Government and military:** Post office scheduling and planning, military logistics, target assignment, and manpower deployment.

Some applications mentioned above may not be directly formulated into a LP, but rather a mixed-integer program. In this case, LPs can be used as relaxations of such mixed-integer programs.

7.1.2 Indirect applications

Indirect applications use LPs as subproblems, intermediate steps, or auxiliary problems in other computational methods. For instance, the sequential linear programming (SLP) approach uses LPs as subproblems in solving nonlinear programs [9]. Frank-Wolfe method also uses LPs as subproblems to generate iterates [10]. This method was invented since 1950, but has recently gained attention due to its applications in machine learning and other large-scale convex programs. Branch-and-bounds methods for integer programming also use LPs as subproblems to compute lower-bounds for each branch-and-bound iteration [14].

In this chapter, we focus on some direct applications of LPs, including the Chebyshev center problem, robust sparse signal recovery, blending problems, inventory problems, transportation problems, minimum cost network flow problems, assignment problems, and shortest path problems.

7.2 Applications of LPs

In this section, we discuss using linear programming to solve the Chebyshev center problem, robust sparse signal recovery, the blending problem, and the inventory problem.

7.2.1 Chebyshev center of a polyhedron

We consider a problem of finding the largest ball that is contained in a polyhedron described by the following linear inequalities

$$\mathscr{P} := \left\{x \in \mathbb{R}^n \mid a_i^T x \leq b_i,\ i = 1, \cdots, m\right\}.$$

The center of this largest ball is called the *Chebyshev center* of the polyhedron.

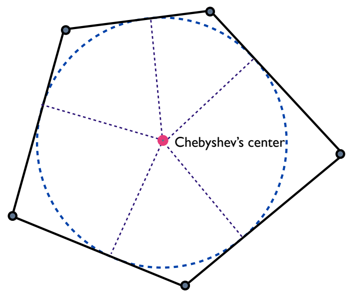


Fig. 7.1 An illustration of the Chebyshev center problem in the two dimensional space $\mathbb{R}^2$

Mathematical model: To model this problem, we present the ball centered at $\bar{x}$ of the radius r as

$$\mathscr{B}_r(\bar{x}) = \left\{x \in \mathbb{R}^n \mid \sum_{i=1}^n (x_i - \bar{x}_i)^2 \leq r^2\right\}.$$

If we define $u = x - \bar{x}$, then $x = \bar{x} + u$ and $\sum_{i=1}^n (x_i - \bar{x}_i)^2 = \sum_{i=1}^n u_i^2$. For any point $x \in \mathscr{B}_r(\bar{x})$ to belong to the polyhedron $\mathscr{P}$, we need $a_i^T x = a_i^T \bar{x} + a_i^T u \leq b_i$ to hold for $i = 1, \cdots, m$. By the Cauchy-Schwarz inequality, we have $a_i^T u \leq \|a_i\|\|u\| \leq r\|a_i\|$, where $\|a\| = \sqrt{\sum_{i=1}^n a_i^2}$ is the norm of a. Hence, the condition $a_i^T \bar{x} + a_i^T u \leq b_i$

holds if $a_i^T\bar{x} + r\|a_i\| \leq b_i$. The ball $\mathscr{B}_r(\bar{x})$ is contained in the polyhedron $\mathscr{P}$ when the following m inequalities are simultaneously satisfied:

$$a_i^T\bar{x} + r\|a_i\| \leq b_i,\ i = 1,\cdots,m.$$

Our goal is to maximize the radius r while satisfying m constraints above. Hence, we can formulate this problem into the following linear program:

$$\begin{cases} \max\limits_{\bar{x},r}\ r \\ \text{s.t.}\quad a_i^T\bar{x} + r\|a_i\| \leq b_i,\ i = 1,\cdots,m \\ \qquad\ \ r \geq 0. \end{cases}$$

If we define $y = (\bar{x}, r)$, $c = (0,0,\cdots,0,1)^T \in \mathbb{R}^{n+1}$, $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} & \|a_1\| \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} & \|a_m\| \end{bmatrix}$, then we can write the above problem into the following compact form:

$$\begin{cases} \max\limits_{y\in\mathbb{R}^{n+1}}\ z = c^Ty \\ \text{s.t.}\qquad By \leq b, \\ \qquad\qquad y_{n+1} \geq 0. \end{cases}$$

Example 7.1. We consider a polyhedron $\mathscr{P}$ in $\mathbb{R}^2$ generated by the following six inequalities:

$$\begin{cases} -x_1 - x_2 \leq -2 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \\ -2x_1 + 3x_2 \leq 12 \\ 3x_1 - 2x_2 \leq 15 \\ 4x_1 + 5x_2 \leq 50. \end{cases}$$

Let us formulate this problem in Matlab (Excel or GAMS) using the input data as

$$B = \begin{bmatrix} -1 & -1 & \sqrt{2} \\ -1 & 0 & 1 \\ 0 & -1 & 1 \\ -2 & 3 & \sqrt{13} \\ 3 & -2 & \sqrt{13} \\ 4 & 5 & \sqrt{41} \end{bmatrix}, \quad b = \begin{pmatrix} -2 \\ 0 \\ 0 \\ 12 \\ 15 \\ 50 \end{pmatrix}, \quad \text{and } c = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Then, we solve this problem to obtain $r = 2.8661$, and $\bar{x} = (3.4661, 2.8661)^T$. Figure 7.2 illustrates the result of this example. The ball is only tangents to three edges

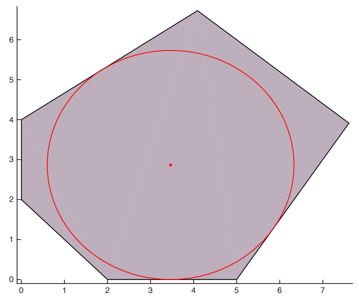


Fig. 7.2 An illustration of the Chebyshev center in $\mathbb{R}^2$ for Example 7.1

over five edges of the polyhedron.

Modeling in GAMS: If we use GAMS, then the following code will solve the above problem:

```
* Chebyshev_center.gms
SETS J /x1, x2, r/, I /c1*c6/;
PARAMETERS b /c1 -2, c2 0, c3 0, c4 12, c5 15, c6 50/;
TABLE A(I,J)
           x1     x2          r
      c1   -1     -1      1.4142
      c2   -1      0      1
      c3    0     -1      1
      c4   -2      3      3.6056
      c5    3     -2      3.6056
      c6    4      5      6.4031;
VARIABLES x(J), z;
```

```
EQUATIONS constraints(I), radius, positive_radius;
radius..z =E= x("r");
constraints(I).. sum(J, A(I,J)*x(J)) =L=b(I);
positive_radius..x("r") =G= 0;
MODEL ChebyshevCenter /all/;
SOLVE ChebyshevCenter USING LP MAXIMIZING z;
```

Note that the values in `TABLE A(I,J)` are the corresponding coefficients of the variables $\bar{x}_1$, $\bar{x}_2$ and r.

7.2.2 Robust sparse signal recovery

Problem description: Assume that we have a signal $s(t)$ that varies over time t. Via a discrete Fourier transformation, this signal is transformed into a sparse vector x in a high dimensional space $\mathbb{R}^n$, meaning that the vector x has few nonzero entries while the others are zero. The signal $s(t)$ can (approximately) be reconstructed by using the inverse of the Fourier transformation F as $s = Fx$. In signal processing, we are not able to collect the full signal s, but we rather sample it at a given sampling rate to obtain a small number of measurements y via a linear measurement operator (or a dictionary) M. Hence, we obtain the output $\bar{b} = Ms = MFx$. However, due to noise, we may obtain $b = MFx + \varepsilon$, where ε is an additive noise vector. Therefore, at the end, we obtain the following model

$$b = Ax + \varepsilon,$$

where $A = MF$ is a given matrix of size $m \times n$, where m is much smaller than n. Our goal is to minimize the mismatch between the actual output b and the clean

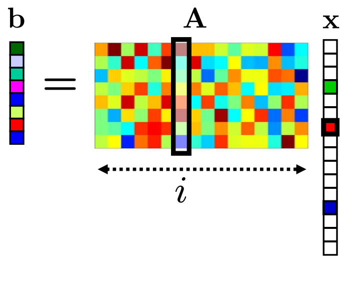


Fig. 7.3 The illustration of a sparse signal recovery model

output $\bar{b}$. We can measure the differences between these vectors by $\|b - \bar{b}\|_1 =$

$\sum_{i=1}^{m} |b_i - \bar{b}_i|$. At the same time, we also want to keep the signal vector x in the Fourier domain to be sparse. Hence, we minimize the following objective function

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} |b_i - (Ax)_i| + \lambda \sum_{j=1}^{n} |x_i|\,.$$

Here, $(Ax)_i$ is the i-th component of the vector $\bar{y} = Ax$, and $\lambda > 0$ is a fixed parameter to trade-off the sparsity level in x. That is if λ is large, then we get a more sparse solution; otherwise, we get a less sparse solution.

LP model: Using the fact that $|z| \leq s$ is equivalent to $-s \leq z \leq s$, the above problem can be reformulated into a linear program by introducing intermediate variables u and v as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n,u,v} & \sum_{i=1}^{m} u_i + \lambda \sum_{j=1}^{n} v_i, \\ \text{s.t.} & -u_i \leq b_i - \sum_{j=1}^{n} A_{ij}x_j \leq u_i,\ i = 1,\cdots,m \\ & -v_j \leq x_j \leq v_j,\ \ j = 1,\cdots,n. \end{array}$$

Each constraint in the first line can be split into two distinct constraints: $\sum_{j=1}^{n} A_{ij}x_j - u_i \leq b_i$ and $\sum_{j=1}^{n} A_{ij}x_j + u_i \geq b_i$ for $i = 1,\cdots,m$. Each constraint in the second line can also be written as $x_j + v_j \geq 0$ and $x_j - v_j \leq 0$ for $j = 1,\cdots,n$.

Example 7.2. We consider an example as follows. First, we generate a sparse vector $x^\natural$ as a ground-truth (which is generally unknown in reality) that has s nonzero elements. Then, we generate a measurement operator A of the size $m \times n$. We form the output $b = Ax^\natural + \varepsilon$, with a sparse and random noise ε (with 20% of nonzero elements). We solve the above problem using the two different data sets as shown in the following figure, Figure 7.4. We use a Matlab code to solve this problem, and figure 7.4 shows that the recovery signal fits very well the original signal $x^\natural$ in both the small and medium size problems. As an exercise, students are asked to model this problem in Matlab using `linprog` to solve it.

7.2.3 Blending problems

Blending problems are a typical application of mathematical optimization. They involve blending several resources or materials to create one or more products corresponding to a demand. The goal is often to maximize the total profit/net sales or to minimize the cost, while satisfying all the constraints on available resources, product quality, or certain regulations, etc. Such problems arise in domains such

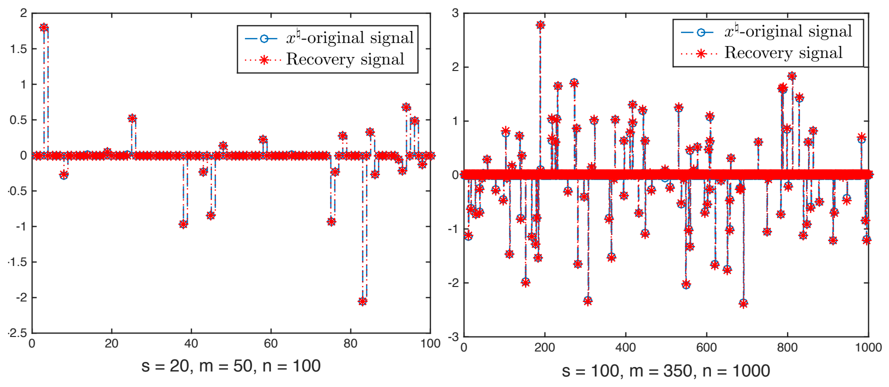


Fig. 7.4 An illustration of sparse signal recovery. Left: small problem, Right: medium problem

as metal, oil, paint, and food-processing industries. Some documents refer to this problem as an assembly problem. Since the form of this problem varies from one application to another, which is very hard to give a general formulation, we therefore prefer to describe it via a concrete example as follows.

7.2.3.1 A concrete example

We illustrate a way to solve a blending problem by performing the following steps.
Problem description: Sunco Oil manufactures three types of gasolines (gas 1, gas 2, gas 3). Each type is produced by blending three types of crude oils (crude 1, crude 2 and crude 3). The prices of selling gasolines and of purchase crude oils are given in the following table:

Gas	Sales price (\$/Barrel)	Crude	Purchase price (\$/Barrel)
1	70	1	45
2	60	2	35
3	50	3	25

Sunco can purchase up to 5000 barrels of each type of crude oil daily. It costs \$4 to transform one barrel of oil into one barrel of gasoline, and Sunco's refinery can produce up to 14000 barrels of gasoline daily.

The three types of crude oil differ in their octane levels and sulfur content, see the table below.

Crude	Octane level	Sulfur content (%)
1	12	0.5
2	6	2.0
3	8	3.0

In a mixture of different types of crude oil, the octane levels and sulfur content blend linearly. The mixture in gas 1 (or gas 2, gas 3) must have an average octane level of at least 10 (8 for gas 2 and 6 for gas 3) and contain at most 1% sulfur (2% for gas 2 and 1% for gas 3).

The demand of each type of gasoline is as follows:

gas 1 – 3000 barrels per day
gas 2 – 2000 barrels per day
gas 3 – 1000 barrels per day

Sunco considers it an obligation to meet these demands. Assume that the gasoline cannot be stored, so it does not bring any revenue if not sold on the day it is produced. How does Sunco maximize its daily profit (The daily profit is defined as the revenue subtracting the cost)?

7.2.3.2 Mathematical model

To model Sunco's problem as a linear program, we let x_{ij} denote the number of barrels of crude oil i used daily to produce gas j, for $i = 1,2,3,\ j = 1,2,3$. We let s_j be the sales price for gas j and p_i be the purchase price of crude oil i, and let d_j be the demand of gas j.

Objective function: The objective function is to maximize the daily profit:

$$z = \underbrace{\sum_{j=1}^{3} s_j \sum_{i=1}^{3} x_{ij}}_{\text{total sales of gasolines}} - \underbrace{\sum_{i=1}^{3} p_i \sum_{j=1}^{3} x_{ij}}_{\text{total cost of crude oils}} - \underbrace{4\sum_{i=1}^{3}\sum_{j=1}^{3} x_{ij}}_{\text{total cost of transformation}} = \sum_{i=1}^{3}\sum_{j=1}^{3}(s_j - p_i - 4)x_{ij}.$$

Constraints: There are different types of constraints:

- **Daily purchase limit of the crude oil:** The purchase limit is 5000 barrels, which requires $\sum_{j=1}^{3} x_{ij} \leq 5000$ for $i = 1,2,3$.
- **Daily production limit of the gasoline:** The total production limit is 14000 barrels, which requires $\sum_{i=1}^{3}\sum_{i=1}^{3} x_{ij} \leq 14000$.
- **Demand constraints:** Each type of gasoline should meet the demand, which is $\sum_{i=1}^{3} x_{ij} \geq d_j$ for $j = 1,2,3$.

- **Octane level constraints:** The octane level in the gasoline and the crude, which has 3 constraints.
- **Sulfur content limit constraints:** The sulfur content in the gasoline and the crude, which also needs three constraints.

In a summary, the mathematical formulation of the LP problem for this blending process is as follows:

$$\begin{cases} \max\limits_{x_{ij}} \ z = \sum_{j=1}^{3} s_j \sum_{i=1}^{3} x_{ij} - \sum_{i=1}^{3} p_i \sum_{j=1}^{3} x_{ij} - 4\sum_{i=1}^{3}\sum_{j=1}^{3} x_{ij} & \text{(total profit)} \\ \text{s.t.} \ \sum_{j=1}^{3} x_{ij} \le 5000, \ i = 1,2,3 & \text{(daily purchase limits of crude oils)} \\ \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ij} \le 14000 & \text{(daily production limit of gasolines)} \\ \sum_{i=1}^{3} x_{ij} \ge d_j, \ j = 1,2,3 & \text{(demand constraints)} \\ \left.\begin{aligned} 12x_{11} + 6x_{21} + 8x_{31} &\ge 10(x_{11} + x_{21} + x_{31}) \\ 12x_{12} + 6x_{22} + 8x_{32} &\ge 8(x_{12} + x_{22} + x_{32}) \\ 12x_{13} + 6x_{23} + 8x_{33} &\ge 6(x_{13} + x_{23} + x_{33}) \end{aligned}\right\} & \text{(octane level constraints)} \\ \left.\begin{aligned} 0.005x_{11} + 0.02x_{21} + 0.03x_{31} &\le 0.01(x_{11} + x_{21} + x_{31}) \\ 0.005x_{12} + 0.02x_{22} + 0.03x_{32} &\le 0.02(x_{12} + x_{22} + x_{32}) \\ 0.005x_{13} + 0.02x_{23} + 0.03x_{33} &\le 0.01(x_{13} + x_{23} + x_{33}) \end{aligned}\right\} & \text{(sulfur content limits)} \\ x_{ij} \ge 0, \ i = 1,2,3, \ j = 1,2,3. \end{cases}$$

7.2.3.3 GAMS model

In the file `blendingoil.gms`, we define two sets `I` and `J` to be index sets of the three types of crude oil and the three types of gas respectively. The asterisk "*" is used to omit intermediate elements in a set. The set `I` as defined in this example contains three labels "crude-1", "crude-2" and "crude-3". That is

```
SET  I /crude-1*crude-3/, J /gas-1*gas-3/;
```

We then define a variable `x` over `(I, J)`; this defines nine variables, one for each pair that contains a label in `I` and a label in `J`. For example, the GAMS variable `x("crude-1","gas-1")` represents the variable x_{11} in the mathematical formulation.

In the definition of the equation `obj` we use the expression

```
profit =E= sum((I,J), (sellingPrice(J)-purchasePrice(I)-unitPCost)*x(I,J))
```

to represent the sum of `sellingPrice(J)-purchasePrice(I)-unitPCost)*x(I,J)` over all `(I,J)` pairs. While the equation `obj` uses sets `I` and `J` in its definition, it is declared to be a single equation. In contrast, the equation `proDem` is declared over the set `J`. In other words, `J` is the domain of the equation `proDem`. In the definition statement of the equation,

```
proDem(J).. sum(I, x(I,J))  =g= demand(J);
```

the set `J` appears in the parentheses after the equation name `proDem` before the two dots, and works as the "controlling set" or "controlling index" of the equation. (In general, if an equation is declared over a domain, then this domain needs to be used as the controlling set in its definition statement. More experienced users can also use subsets of the domain as the controlling set.) Since the controlling set `J` consists of three labels, GAMS compiler will generate three constraints, one for each label in `J`. In contrast, `I` is used as the index of summation in the equation. The expression `sum(I, x(I,J))` gives the sum of `x(I,J))` over all indices `I`, for a fixed label `J`. After compiling `blendingoil.gms`, we can find the following excerpt from the output file `blendingoil.lst`:

```
proDem(gas-1).. x(crude-1,gas-1) + x(crude-2,gas-1) + x(crude-3,gas-1) =G= 3000;
proDem(gas-2).. x(crude-1,gas-2) + x(crude-2,gas-2) + x(crude-3,gas-2) =G= 2000;
proDem(gas-3).. x(crude-1,gas-3) + x(crude-2,gas-3) + x(crude-3,gas-3) =G= 1000;
```

Note that the above lines are extracted from the GAMS output and should not be used as GAMS statements.

As we now see, by declaring an equation over a domain, we can create multiple equations under a common name, indexed by labels in the domain. Equations `limCrude`, `octaneGas`, and `sulfurGas` are similarly declared over domains. For example, `limCrude` is declared over the set `I`, so GAMS will create one constraint for each label in `I` as can be found in `blendingoil.lst`.

Finally, we provide a complete GAMS code for this example as below:

```
*blendingoil.gms
SET
       I /crude-1*crude-3/, J /gas-1*gas-3/;
SCALAR upCrude "Crude oil available (barrels)" /5000/,
       upTotalGas "Upper limit on gasoline (barrels)" /14000/,
       unitPCost "Production cost (dollar per barrel)" /4/;
```

```
PARAMETERS
       sellingPrice(J)   /gas-1 70, gas-2 60, gas-3 50/,
       purchasePrice(I)  /crude-1 45, crude-2 35, crude-3 25/,
       octane(I)         /crude-1 12, crude-2 6, crude-3 8/,
       sulfur(I)         /crude-1 0.5, crude-2 2, crude-3 3/,
       loOctane(J)       /gas-1 10, gas-2 8, gas-3 6/,
       upSulfur(J)       /gas-1 1, gas-2 2, gas-3 1/,
       demand(J)         /gas-1 3000, gas-2 2000, gas-3 1000/;
FREE VARIABLE
       profit "total profit";
POSITIVE VARIABLES
       x(I,J) "barrels of crude I used to produce gas J";
EQUATIONS
       obj            "max total profit",
       proDem(J)      "production meets demand",
       limCrude(I)    "Limit on each type of crude oil",
       limGas         "Limit on total gasoline",
       octaneGas(J)   "Gasoline octane level",
       sulfurGas(J)   "Gasoline sulfur content";
       obj..          sum((I,J), (sellingPrice(J)-purchasePrice(I)
                      -unitPCost)*x(I,J)) =E= profit;
       proDem(J)..    sum(I, x(I,J))  =G= demand(J);
       limCrude(I)..  sum(J,x(I,J)) =L= upCrude;
       limGas..       sum((I,J),x(I,J)) =L= upTotalGas;
       octaneGas(J).. sum(I,octane(I)*x(I,J)) =G= loOctane(J)*sum(I,x(I,J));
       sulfurGas(J).. sum(I,sulfur(I)*x(I,J)) =L= upSulfur(J)*sum(I,x(I,J));
MODEL blending /ALL/;
SOLVE blending USING LP MAXIMIZING profit;
```

7.2.4 The inventory problem

Managing and maintaining inventories (goods and materials held for sale or use) are necessary for every business that handles physical products. Many companies use mathematical models built with operations research techniques to improve their inventory management procedure [12].

Inventory models can be divided into two broad categories: deterministic models and stochastic models, depending on whether the demand is predictable. They can also be classified into continuous-review models and periodic-review models. Continuous-review models treat demand as continuous flow and replenish inventory whenever the inventory level drops sufficiently low, while periodic-review models divide the time span into periods and plan for how much to produce or order during each period to satisfy the demand of each period. A third approach to classify inventory models is based on whether the model considers a single location (single-echelon) or a network of locations (multiple-echelon).

Due to the scope of this course, we only present this problem through a simple example. The example we present is a deterministic periodic-review single-echelon model.

7.2.4.1 A concrete example: Sailboat inventory problem.

(a) Problem description: Sailco Corporation must determine how many sailboats to produce for each of the next four quarters (one quarter = three months). The demand d_t during quarter t ($t = 1,2,3,4$) is as given below, and the demand must be met on time.

First quarter	$d_1 = 40$ sailboats
Second quarter	$d_2 = 60$ sailboats
Third quarter	$d_3 = 75$ sailboats
Fourth quarter	$d_4 = 25$ sailboats

At the beginning of the first quarter, Sailco has an inventory of 10 sailboats. During each quarter, Sailco can produce up to 40 sailboats with regular-time labor at a cost of \$400 per sailboat. By having employees work overtime during a quarter, Sailco can produce additional sailboats with overtime labor at a cost of \$450 per sailboat.

For simplicity, we assume that sailboats manufactured during a quarter can be used to meet demand for that quarter. At the end of each quarter (after production has occurred and the current quarter's demand has been satisfied), a carrying or holding cost of \$20 per sailboat is incurred for sailboats remaining in stock.

Goal: Sailco's objective is to minimize the total cost. How many sailboats should be produced during each quarter?

(b) Mathematical model: We define the following variables for each quarter t ($t = 1,2,3,4$):

- x_t = number of sailboats produced by regular-time labor (at \$400/boat) during quarter t;

- y_t = number of sailboats produced by overtime labor (at \$450/boat) during quarter t;
- i_t = number of sailboats at hand at the end of the quarter t (will be in an inventory).

If we specify a constant $i_0 = 10$, then the number of sailboats at the end of the quarter i is

$$i_t = i_{t-1} + x_t + y_t - d_t, \quad \text{for } t = 1,2,3,4.$$

The total cost is

$$z = \sum_{t=1}^{4} (400x_t + 450y_t + 20i_t),$$

which includes the labor cost of manufacturing in regular-time and during overtime, and the incurred cost of holding the boats. We also have a constraint on the maximum production capacity: $x_t \leq 40$ for $t = 1,2,3,4$.

Putting these components together, we can write the mathematical formulation of this inventory problem as follows:

$$\begin{cases} \min\limits_{x_t,y_t,i_t} \sum\limits_{t=1}^{4} (400x_t + 450y_t + 20i_t) \\ \text{s.t. } i_0 = 10, \;\; i_t = i_{t-1} + x_t + y_t - d_t, \;\; t = 1,2,3,4, \\ \qquad x_t \leq 40, \;\; t = 1,2,3,4, \\ \qquad i_t, x_t, y_t \geq 0, \;\; t = 1,2,3,4. \end{cases}$$

In the above formulation, we use separate variables for sailboats produced during regular time and those produced overtime, in order to express the total cost as a linear function of variables. The irrational choices such that $(x_t, y_t) = (35, 25)$ will not appear in an optimal solution, since it can be replaced by $(x_t, y_t) = (40, 10)$ for lower cost without affecting values of other variables.

This problem is an LP problem with $n = 12$ variables, 4 equality constraints, and 4 bound constraints on x_t. All variables are nonegative.

7.2.4.2 GAMS model

To model this problem in GAMS, as before, we use the keyword `SET` to define the index set t for the quarters. For example, with the following declaration

```
SET t /1*4/;
```

the labels in the set `t` will be ordered as “1,” “2,” “3” and “4.”. The main component is the inventory equation $i_t = i_{t-1} + x_t + y_t - d_t$. The equation `invRel` declared on the domain `t` is used to represent the constraints this constraint. To understand how the equation `invRel(t)` is defined, we need to understand the use of **ordered sets**, **logical conditions** and **conditional expressions** in GAMS.

As we know, a GAMS set is a collection of labels. For modeling convenience, GAMS puts the labels in an order. Interested readers can find more details about ordered sets in Section 13.2 of `GAMS User Guide`. For this course it suffices to know that if the labels in a (static) set are not used as labels in another set, then their order is determined by their order in the set declaration statement. The function `ord()` returns the ordinal number associated with a given label. For the above set one has `ord("1") = 1`, `ord("2") = 2`, etc, if the set `t` is defined as `SET t /1*4/;` as above. If we instead declare the set `t` as

```
SET t /4, 1, 3, 2/;
```

then the labels in the set `t` will be ordered as “4”, “1”, “3” and “2” with `ord("4") = 1`, `ord("1") = 2`, etc.

While elements of a set are labels instead of numbers, the expression `t-1` is recognized by GAMS as the lag operator for order sets. For an equation declared over the set `t`, the expression `t-1` in the equation means the previous element (of the current element). In contrast, `t+1` in the equation definition refers to the next element.

If we would define the equation `invRel(t)` as

```
invRel(t)..  i(t) =e=  i(t-1) + x(t)+ y(t) - d(t);
```

then GAMS would attempt to generate the following four equations, one for each element of `t`, as follows:

```
t="1": i("1") =e= i("0") + x("1") + y("1") - d("1");
t="2": i("2") =e= i("1") + x("2") + y("2") - d("2");
t="3": i("3") =e= i("2") + x("3") + y("3") - d("3");
t="4": i("4") =e= i("3") + x("4") + y("4") - d("4");
```

The problem with the above definition is that `i("0")` is undefined in GAMS, and will be treated as zero instead of the desired value 10. To overcome this problem, we modify the equation definition as follows:

```
invRel(t).. i(t) =e= i(t-1)$(ord(t) > 1) + iniInv$(ord(t)=1)+ x(t)+ y(t) - d(t);
```

The expression `i(t-1)$(ord(t) > 1)` is a conditional expression. When GAMS creates the equation for a fixed element of `t`, it will check if the condition `ord(t) > 1` holds. If that conditions holds, then the conditional expression takes the value of `i(t-1)`; otherwise, the conditional expression takes the value of zero. Similarly, the expression `iniInv$(ord(t)=1)` is a conditional expression that takes the value of `iniInv` if `ord(t) = 1` holds and zero otherwise. With the modified definition, GAMS will generate the following four individual equations:

```
t="1": i("1") =e= iniInv + x("1") + y("1") - d("1");
t="2": i("2") =e= i("1") + x("2") + y("2") - d("2");
t="3": i("3") =e= i("2") + x("3") + y("3") - d("3");
t="4": i("4") =e= i("3") + x("4") + y("4") - d("4");
```

The conditions `ord(t) > 1` and `ord(t) = 1` are examples of logical conditions, that are heavily used in GAMS. Logical conditions are special expressions that evaluate to a value of `True` or `False`. Numerical comparisons and logical operators can be used in logical conditions.

A numerical comparison compares the values of two numerical expressions, with one of the following operators:

$$<, \ =, \ >, \ <=, \ >= \text{ and } <> .$$

For example `(2 < 1)` is a numerical comparison that is always `False`, and `ord(t) > 1` is `True` if the element of `t` considered is not the first label. A logical operator can be one of the following:

`not, and, or.`

These single expressions can be combined in a more complicated expression. For example,

- the logical condition `(2 < 1) and (3 < 4)` is `False`,
- the logical condition `not(2 < 1)` is `True`,
- the logical condition `(2 < 1) or (3 < 4)` is `True`.

Another operator that can be used in logical conditions is the function `sameas(A,B)`. This function compares if two strings `A` and `B` are exactly the same. For example, `sameas(i,"3")` evaluates to be `True` if the element of `i` considered here is the same as the string `"3"`. Using this function, we can rewrite the definition of `invRel(t)` as

```
invRel(t).. i(t) =e= i(t-1)$ (not sameas(t,'1')) + iniInv$(sameas (t,'1'))+ x(t)+ y(t) - d(t);
```

Note that it is wrong to write the above equation as

```
invRel(t).. i(t)=e=i(t-1)$(t>1)+iniInv$(t=1)+ x(t)+ y(t)-d(t);
```

because elements of the set `t` are not numbers and cannot be compared using the numerical comparators.

In general, a conditional expression is of the form

```
numerical_expression$logical_condition
```

where we put a dollar sign ($) after a numerical expression `numerical_expression`, followed by a logical condition `logical_condition`. If this logical condition is true, then the entire expression takes the value of the numerical expression. Otherwise the entire expression takes the value of zero. We will see more ways of using the dollar sign for conditional assignments and control of the domain.

The constraints $x_t \leq 40$ for $t = 1,2,3,4$ in the mathematical model are treated in GAMS by the following statement

```
x.up(t)=upRegProd;
```

which assigns the upper bound `upRegProd` for each $x(t)$. Note that the above statement is an assignment statement in GAMS, instead of a definition statement for an equation. For this reason, we use the equal sign instead of `"=e="` between the two sides of the equality. Note that the above statement assigns a common upper bound for `x("1"), x("2"), x("3")` and `x("4")` simultaneously. If one would need to assign an upper bound to `x("1")` only, then one should write

```
x.up("1")=upRegProd;
```

For lower bounds, one must use `.lo` instead of `.up`.

There is a potential benefit with assigning upper bounds to variables, instead of declaring and defining equations for them. Some LP solvers in GAMS handle bound constraints in special ways, different from other (general) constraints. This can bring much more efficiency in some large-scale problems.

To this end, we can summarize the entire GAMS code for the above inventory problem.

```
*sailcoinv.gms
SET
    t "indices of quarters" /1*4/;
SCALAR
    iniInv "Inventory at the beginning of first quarter"     /10/,
    upRegProd "Upper limit of regular-time production"       /40/,
    regUnitCost "Regular time unit cost (dollar per boat)"   /400/,
    overUnitCost "Overtime unit cost (dollar per boat)"      /450/,
    holdCost "Holding cost (dollar per boat)"                /20/;
PARAMETERS
    d(t) /1 40, 2 60, 3 75, 4 25/;
FREE VARIABLE cost "total cost";
POSITIVE VARIABLES
    x(t),
    y(t),
    i(t);
EQUATIONS
    obj        "min total cost",
    invRel(t)  "Relation between inventory from different periods";
  obj.. cost =E= SUM(t, regUnitCost*x(t) + overUnitCost*y(t) + holdCost*i(t));
  invRel(t).. i(t) =E= i(t-1)$(ord(t)>1) + iniInv$(ord(t) = 1) + x(t) + y(t) - d(t);
  x.up(t) = upRegProd;
MODEL sailboat /ALL/;
SOLVE sailboat USING LP MINIMIZING cost;
```

We note that the command line in this GAMS code

```
invRel(t).. i(t) =E=  i(t-1)$(ord(t)>1) + iniInv$(ord(t) = 1) + x(t) + y(t) - d(t);
```

can be replaced by

```
invRel(t).. i(t) =e=i(t-1)$ (not sameas(t,'1')) + iniInv$(sameas (t,'1'))+ x(t)+ y(t) - d(t);
```

7.3 Transportation problems

In operations research, the transportation problem is often reformulated into an LP problem. However, this LP formulation has special structure that allows us to treat it individually, and to develop different simplex algorithmic variants to solve it efficiently. Due to time limit, we only focus in this section the mathematical modeling aspects of this problem. Methods for solving this problem can be found in many textbooks including [3, 7, 23].

7.3.1 Problem description and mathematical formulation

Problem description: The transportation problem is a type of problems that seek a minimum-cost distribution of a given commodity/product from a group of supply points to a group of demand points.

- Each supply point $i = 1, \cdots, m$ has a fixed supply s_i that is ready to distribute.
- Each demand point $j = 1, \cdots, n$ requires a demand d_j that must be satisfied.
- It costs c_{ij} to ship each unit of the commodity from the supply point i to the demand point j.

The goal is to find a transportation plan that minimizes the total shipping cost, while it satisfies the demand at each demand point, and does not exceed the supply limitation at each supply point.

If we consider supply points and demand points as nodes of a graph (network), then we can visualize the transportation problem as a directed bipartite graph as in Figure 7.5. Here, each supply node i with a supply capacity s_i is connected to each

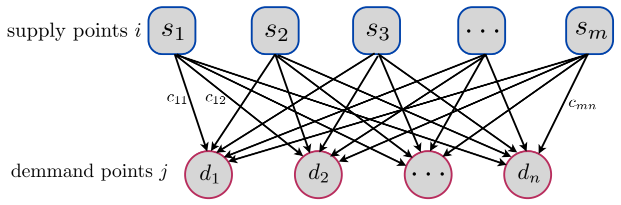


Fig. 7.5 An illustration of a transportation network.

demand node j with a demand d_j by a link.

Mathematical formulation: Let us define x_{ij} to be the amount to ship from supply point i to demand point j. These variables form a matrix $X = (x_{ij})$ called the transportation plan. Then, we can formulate the problem as follows:

$$\begin{cases} \min\limits_{x_{ij}} \ z = \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad \sum\limits_{j=1}^{n} x_{ij} \le s_i, \ i = 1, \cdots, m \ \text{(supply constraints)} \\ \qquad \sum\limits_{i=1}^{m} x_{ij} \ge d_j, \ j = 1, \cdots, n \ \text{(demand constraints)} \\ \qquad x_{ij} \ge 0, \ i = 1, \cdots, m, \ j = 1, \cdots, n \end{cases} \tag{7.1}$$

Let us demonstrate how to formula this problem using GAMS via the following example.

Example 7.3. Powerco has three electric power plants that supply the needs of four cities. The supply capacity of each power plant and the demand in each city (in million kwh/hour) are given in the last column and the last row, respectively of the table below.

From/to	City 1	City 2	City 3	City 4	Supply
Plant 1	\$8	\$6	\$10	\$9	35
Plant 2	\$9	\$12	\$13	\$7	50
Plant 3	\$14	\$9	\$16	\$5	40
Demand	45	20	30	30	

The cost of sending 1 million kwh of electricity from each plant to each city is also given in the above table. How does Powerco minimize the total cost of meeting all cities' demands?

GAMS model: To model this problem, we use a two-dimensional table `unitCost(I, J)` to represent the costs c_{ij} between the plant i to the city j. The domain of this table is the set of ordered pairs in the Cartesian product of sets `I` and `J`. When using a table one needs to line up the entries properly so that the values are under the appropriate headings. If there are blank entries in the table, they are interpreted as zero. The table declaration statement does not use two slashes to enclose data, as the declaration statement of parameters, but it still needs to end with a semicolon. Experienced GAMS users may sometimes prefer to omit the comma separating different items in a statement when starting each item with a new line, as at some places in `transportation.gms`.

We can model this problem with the following complete GAMS model.

```
*transportation.gms
SETS
    I "plants" /p1*p3/
    J "cities" /c1*c4/;
PARAMETERS
    s(I)  "Supply of each plant"
         /p1 35
          p2 50
          p3 40/,
    d(J) "Demand of each city"
         /c1    45
          c2    20
          c3    30
          c4    30/;
TABLE
    unitCost(I, J)
                   c1       c2      c3      c4
        p1       8        6       10      9
        p2       9        12      13      7
        p3       14       9       16      5;
VARIABLES z, x(I, J);
POSITIVE VARIABLE x;
EQUATIONS cost, supply(I), demand(J);
  cost..      SUM( (I, J), unitCost(I, J)*x(I, J) ) =E= z;
  supply(I).. SUM(J, x(I, J)) =L=  s(I);
  demand(J).. SUM(I, x(I, J)) =G=  d(J);
MODEL transport /ALL/;
SOLVE transport USING LP MINIMIZING z;
```

7.3.2 Balanced transportation problems

We say that a transportation problem is **balanced** if the total supply equals total demand, i.e.:

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j.$$

In this case, we call it a **balanced transportation problem**. For example, the Powerco example above is such a balanced transportation problem.

Theorem 7.1. *In a balanced transportation problem, any feasible solution x satisfies*

$$\sum_{j=1}^{n} x_{ij} = s_i \text{ for each } i = 1,\cdots,m,$$

and

$$\sum_{i=1}^{m} x_{ij} = d_j \text{ for each } j = 1,\cdots,n.$$

Proof. If $X = (x_{ij})$ is a feasible solution in a balanced transportation problem, then

$$\sum_{j=1}^{n} d_j \leq \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \leq \sum_{i=1}^{m} s_i,$$

where the first item equals the last item by assumption. So

$$\sum_{j=1}^{n} d_j = \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{m} s_i.$$

Since $d_j \le \sum_{i=1}^m x_{ij}$ for each $j = 1,\cdots,n$, from the equality $\sum_{j=1}^n d_j = \sum_{j=1}^n \sum_{i=1}^m x_{ij}$ we can deduce $d_j = \sum_{i=1}^m x_{ij}$ for each $j = 1,\cdots,n$. Similarly, since $\sum_{j=1}^n x_{ij} \le s_i$ for each $i = 1,\cdots,m$, from the equality $\sum_{i=1}^m \sum_{j=1}^n x_{ij} = \sum_{i=1}^m s_i$ we can deduce $\sum_{j=1}^n x_{ij} = s_i$ for each $i = 1,\cdots,m$. □

According to Theorem 7.1, a balanced transportation problems can alternatively be written as:

$$\begin{cases} \min\limits_{x_{ij}} \; z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = s_i, \; i = 1,\cdots,m \quad \text{(supply constraints)} \\ \qquad \sum_{i=1}^{m} x_{ij} = d_j, \; j = 1,\cdots,n \;\text{(demand constraints)} \\ \qquad x_{ij} \ge 0, \; i = 1,\cdots,m, j = 1,\cdots,n \end{cases} \tag{7.2}$$

The above new formulation is much more efficient to solve than the original formulation (7.1). More explanation will be given in the subsequent sections on network flows.

Now, we investigate more properties of the balanced transportation problem (7.2). First we arrange the variables x_{ii} into a vector $x \in \mathbb{R}^{mn}$ of the form

$$x = (x_{11}, x_{12}, \cdots, x_{1n}, x_{21}, x_{22}, \cdots, x_{2n}, \cdots, x_{m1}, x_{m2}, \cdots, x_{mn})^T.$$

Next, we write the input data (A,b,c) of the balanced transportation problem (7.2) as

$$A = \begin{bmatrix} 1 & 1 & \cdots 1 & 0 & 0 & \cdots 0 & \cdots 0 & 0 & \cdots 0 \\ 0 & 0 & \cdots 0 & 1 & 1 & \cdots 1 & \cdots 0 & 0 & \cdots 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots 0 & 0 & 0 & \cdots 0 & \cdots 1 & 1 & \cdots 1 \\ 1 & 0 & \cdots 0 & 1 & 0 & \cdots 0 & \cdots 1 & 0 & \cdots 0 \\ 0 & 1 & \cdots 0 & 0 & 1 & \cdots 0 & \cdots 0 & 1 & \cdots 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots 1 & 0 & 0 & \cdots 1 & \cdots 0 & 0 & \cdots 1 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} m \text{ rows} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} n \text{ rows} \end{matrix} \quad b = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \\ d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}, \text{ and } c = \begin{pmatrix} c_{11} \\ \vdots \\ c_{1n} \\ c_{21} \\ \vdots \\ c_{2n} \\ \vdots \\ c_{m1} \\ \vdots \\ c_{mn} \end{pmatrix}.$$

Matrix A is of the size $(m+n) \times mn$. Moreover, from the form of A, we can see that each column of A has exactly two entries 1, and other entries are zero. Therefore, if we sum up each column of A, we obtain 2. This shows that A has less than $m+n$ linear independent rows. On the other hand, from the last n rows, we can form an $n \times n$ identity matrix by taking the left-bottom $n \times n$-submatrix. From the second to the m-th rows, we can form another $(m-1) \times (m-1)$ identity submatrix by taking the entries at the columns $n+1$, $2n+1, \cdots, (m-1)n+1$. Putting these submatrices together, we obtain an identity submatrix of the size $(m+n-1) \times (m+n-1)$. Hence, we can conclude that A has exactly $m+n-1$ independent rows.

Theorem 7.2. *The balanced condition $\sum_{i=1}^m s_i = \sum_{j=1}^n d_j$ is necessary and sufficient for* (7.2) *to have an optimal solution.*

Proof. If (7.2) has an optimal solution $X^* = (x^*_{ij})$, then we have

$$\sum_{j=1}^n x^*_{ij} = s_i, \ \ i = 1, \cdots, m, \quad \text{and} \quad \sum_{i=1}^m x^*_{ij} = d_j, \ \ j = 1, \cdots, n.$$

Summing up the first expression from $i = 1$ to m and the second one from $j = 1$ to n, we get

$$\sum_{i=1}^m s_i = \sum_{i=1}^m \sum_{j=1}^n x^*_{ij} = \sum_{j=1}^n \sum_{i=1}^m x^*_{ij} = \sum_{j=1}^n d_j.$$

Hence, we obtain the balanced condition $\sum_{i=1}^m s_i = \sum_{j=1}^n d_j$.

Conversely, assume that $\sum_{i=1}^m s_i = \sum_{j=1}^n d_j$. We define $M := \sum_{i=1}^m s_i = \sum_{j=1}^n d_j > 0$ and $\bar{X} = (\bar{x}_{ij})$ with $\bar{x}_{ij} = \frac{s_i d_j}{M}$. Clearly, we can check that $\bar{x}_{ij} \geq 0$ for $i = 1, \cdots, m$ and $j = 1, \cdots, n$, Moreover, we have

$$\sum_{j=1}^{n} \bar{x}_{ij} = \sum_{j=1}^{n} \frac{s_i d_j}{M} = \frac{s_i}{M} \sum_{j=1}^{n} d_j = s_i, \quad i = 1, \cdots, m$$
$$\sum_{i=1}^{m} \bar{x}_{ij} = \sum_{n=1}^{m} \frac{s_i d_j}{M} = \frac{d_j}{M} \sum_{i=1}^{m} s_i = d_j, \quad j = 1, \cdots, n.$$

Hence, $\bar{X}$ is a feasible solution to (7.2). Now, we consider an arbitrary feasible solution $X = (x_{ij})$ of (7.2). It must satisfy all the constraints of (7.2). Hence, we have

$$0 \leq x_{ij} \leq \sum_{j=1}^{n} x_{ij} = s_i, \quad i = 1, \cdots, m, \quad j = 1, \cdots, n.$$

Therefore, the feasible set of (7.2) is nonempty and bounded. We can show that

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \geq \sum_{i=1}^{m} \sum_{j=1}^{n} \min\left\{0, c_{ij} s_i\right\} = \text{constant}.$$

This shows that the objective function of (7.2) is bounded from below. By Theorem 3.2, problem (7.2) must have an optimal solution. □

7.3.3 *Transportation tableaux and solution methods*

Since a transportation problem is an LP problem, in general, we can apply any simplex method in the previous chapter to solve this problem. However, due to its special structure, the simplex method has been specialized to solve transportation problems in a much simpler manner and efficient for large-scale problems. In this subsection, we describe one method for solving transportation problems based on transportation tableaux as a special case of the simplex method.

(a) Transportation tableaux: We can present the input data of a transportation problem in a table called **transportation tableau**. This tableau can also be used to perform a simplex method for solving the given transportation problem. A transportation tableau is a table of $m+2$ rows and $n+2$ columns as below.

	D_1	D_2	D_3	...	D_n	Supply
S_1	c_{11} x_{11}	c_{12} x_{12}	c_{13} x_{13}	... ...	c_{1n} x_{1n}	s_1
S_2	c_{21} x_{21}	c_{22} x_{22}	c_{23} x_{23}	... ...	c_{2n} x_{2n}	s_2
S_3	c_{31} x_{31}	c_{32} x_{32}	c_{33} x_{33}	... ...	c_{3n} x_{3n}	s_3
...	... ...	... ...	... ...	... ...	... ...	...
S_m	c_{m1} x_{m1}	c_{m2} x_{m2}	c_{m3} x_{m3}	... ...	c_{mn} x_{mn}	s_m
Demand	d_1	d_2	d_3	...	d_n	

The first row (called row 0) and the first column (called column 0) mark the labels of the demand nodes and supply nodes, respectively. The last column $(n+1)$ provides the supply capacity of each supply node, and the last row $(m+1)$ shows the demand of each demand node. The cells from 2nd to mth rows (row 1 to row m) and 2nd to nth columns (column 1 to column n) contains the cost c_{ij} and a feasible solution x_{ij}. The cost c_{ij} is given on the left-top conner, and the component x_{ij} is on the right-bottom conner of the cell.

(b) Finding a basic feasible solution: In order to start the simplex method for solving the transportation problem (7.2), we need to find a basic feasible solution $X = (x_{ij})$ first. There are at least two methods to find such a BFS X: The **north-west corner** method, and the **least-cost** method (also called **least-remaining cost** method). Let us describe the north-west corner method as follows:

Step 1: Start from the left-top corner cell $(i, j) = (1, 1)$ (also called the north-west corner). Allocate x_{ij} with $x_{ij} = \min\left\{s_i, d_j\right\}$. There are two situations:

- If $s_i > d_j$, then we cross column j, and substitute s_i by $s_i - d_j$. We obtain a new transportation tableau with the same number of rows, but less than one column.
- If $s_i < d_j$, then we cross row i, and substitute d_j by $d_j - s_i$. We obtain a new transportation tableau with the same number of column, but less than one row.

In any case, we remove either one column or one row.

Step 2: Repeat **Step 1** with the new transportation tableau as a part of the whole transportation tableau.

The algorithm terminates after we cross all rows and columns. Record all the entries x_{ij} corresponding at each step of the algorithm, we obtain a basic feasible solution $X = (x_{ij})$.

For the least-cost method, instead of starting from the left-top conner, we start at the cell with minimum cost c_{ij}. If there are many cells, we can choose any of them. The theory of such a method can be found in several textbooks including [3, 7, 23]. We do not go into detail in this course.

Example 7.4. In order to see how the north-west corner method works, we consider Example 7.3 above. The following steps are carried out:

1. Allocate $x_{11} = \min\{35,45\} = 35$. Since $s_1 < d_1$, we cross row 1 and replace d_1 with $d_1 \to d_1 - s_1 = 45 - 35 = 10$. The new tableau has 2 rows and 4 columns in the main part.
2. Allocate $x_{21} = \min\{50,10\} = 10$. Since $s_2 > d_1$, we cross column 1 in the new tableau, and replace s_2 with $s_2 - d_1 = 40$. The new tableau has 2 rows and 3 columns in the main part.
3. Allocate $x_{22} = \min\{40,20\} = 20$. Since $s_2 > d_2$, we cross column 2 in the new tableau, and replace s_2 with $s_2 - d_2 = 20$. The new tableau has 2 rows and 2 columns in the main part.
4. Allocate $x_{23} = \min\{20,30\} = 20$. Since $s_2 < d_3$, we cross row 2 in the new tableau, and replace d_3 with $d_3 - s_2 = 10$. The new tableau has 1 rows and 2 columns in the main part.
5. Allocate $x_{33} = \min\{40,10\} = 10$. Since $s_3 > d_3$, we cross column 3 in the new tableau, and replace s_3 with $s_3 - d_3 = 30$. The new tableau has 1 rows and 1 columns in the main part with $s_3 = 30$ and $d_4 = 30$.
6. Allocate $x_{34} = 30$, and the algorithm is completed.

From/To	City 1	City 2	City 3	City 4	Supply
Plant 1	8 $\boxed{35}$	6	10	9	35
Plant 2	9 $\boxed{10}$	12 $\boxed{20}$	13 $\boxed{20}$	7	50
Plant 3	14	9	16 $\boxed{10}$	5 $\boxed{30}$	40
Demand	45	20	30	30	125

We finally obtain a basic feasible solution as

$$X = \begin{bmatrix} 35 & 0 & 0 & 0 \\ 10 & 20 & 20 & 0 \\ 0 & 0 & 10 & 30 \end{bmatrix}.$$

This BFS has exactly $m + n - 1 = 3 + 4 - 1 = 6$ positive entries. Hence, it is a non-degenerate basic feasible solution.

(c) The simplex method: We describe one simplex method to solve the balanced transportation problem (7.2). We first define a concept so-called **closed loop** of a BFS on a transportation tableau: A set of cells $\mathscr{L} = \{(i_k, j_k) \mid 1 \leq k \leq s\}$ on a transportation tableau is called a **closed loop** if

- every consecutive cells (i_k, j_k) and (i_{k+1}, j_{k+1}) are on the same row or the same column;
- no three cells are on the same row or column;
- the first and last cells are on the same row or column.

The basis $\mathscr{U}$ of any non-degenerated BFS X of (7.2) does not contain any closed loop. If we add a new cell to $\mathscr{U}$, then $\mathscr{U}$ contains a closed loop.

This method is described in detail as follows:

Initialization: Apply either the north-west corner method or the least-cost method to find a BFS $X = (x_i j)$. We denote by $\mathscr{U} = \big\{(i, j) \mid x_{ij} > 0\big\}$ the basis of the BFS X. This set has $m+n-1$ entries whenever X is a non-degenerated BFS.

Step 1: Find u_i $(i = 1, \cdots, m)$ and v_j $(j = 1, \cdots, n)$ from the following system of linear equations:

$$u_i + v_j = c_{ij}, \quad (i, j) \in \mathscr{U}.$$

This is a system of $m+n-1$ equations with $m+n$ unknowns u_i, v_j. We can set one unknown to be zero and solve for $m+n-1$ other remaining unknowns, e.g., by a substitution method. Once u_i and v_j are computed, we compute $\bar{c}_{ij}$ as $\bar{c}_{ij} = u_i + v_j - c_{ij}$ for $(i, j) \in \mathscr{U}$.

Step 2: If $c_{ij} \leq 0$ for all $(i, j) \in \mathscr{U}$, we terminate the algorithm, and conclude that X is an optimal solution.

Step 3: (*Choose entering variable*) If $\bar{c}_{ij} > 0$ for some $(i, j) \in \mathscr{U}$, then choose $x_{i_* j_*}$ as the entering variable from the rule

$$\bar{c}_{i_* j_*} = \max\big\{\bar{c}_{ij} \mid \bar{c}_{ij} > 0, (i, j) \in \mathscr{U}\big\}.$$

Step 4: (*Find a closed loop*) The set $\mathscr{U} \cup \{(i_*, j_*)\}$ contains a closed loop $\mathscr{L}$ so that $(i_*, j_*) \in \mathscr{L}$. Find this closed loop. Then, alternatively mark all cells of $\mathscr{L}$ with a "+" or "−" sign starting from (i_*, j_*) with a "+" such that any two consecutive cells must have different signs. We split $\mathscr{L}$ into two subsets $\mathscr{L}_+$ and $\mathscr{L}_-$ that contain all cells of $\mathscr{L}$ with "+" and "−" signs, respectively.

Step 5: (*Choose leaving variable*) Choose $x_{i_0 j_0}$ as a leaving variable from the rule

$$x_{i_0 j_0} = \min\left\{x_{ij} \mid (i,j) \in \mathscr{L}_-\right\}.$$

Step 6: (*Construct a new BFS*) We construct a new BFS $\bar{X} = (\bar{x}_{ij})$ as

$$\bar{x}_{ij} = \begin{cases} x_{ij} - x_{i_0 j_0} & \text{if } (i,j) \in \mathscr{L}_- \\ x_{ij} + x_{i_0 j_0} & \text{if } (i,j) \in \mathscr{L}_+ \\ x_{ij} & \text{if } (i,j) \notin \mathscr{L}. \end{cases}$$

We repeat **Step 1** to **Step 6** until the termination condition at **Step 2** is satisfied. Clearly, $\bar{X}$ constructed at **Step 6** does not contain a closed loop, and its becomes a new basic feasible solution. We note that $\bar{c}_{ij}$ computed at **Step 2** can be considered as the negative values of the reduced cost in the simplex method from the previous chapter since we are solving the minimization problem.

Example 7.5. We consider a balanced transportation problem with the input data given in the following tableau ($m = 3$ supply points and $n = 3$ demand points):

D_j / S_i	D_1	D_2	D_3	Supply
S_1	5	4	1	50
S_2	3	2	6	40
S_3	7	9	11	70
Demand	80	20	60	160

Let us apply the above simplex method to solve this problem:

- **Initialization:** We can check that the following matrix is a BFS to this transportation problem:

$$X = \begin{bmatrix} 0 & 0 & 50 \\ 20 & 20 & 0 \\ 60 & 0 & 10 \end{bmatrix} \quad \text{with} \quad f(X) = 680.$$

We locate this BFS into the transportation tableau to obtain an initial tableau as

D_j / S_i	D_1	D_2	D_3	Supply	
S_1	5	4	1 $\boxed{50}$	50	$u_1 = 6$
S_2	3 $\boxed{20}$	2 $\boxed{20}$	6	40	$u_2 = 0$
S_3	7 $\boxed{60}$	9	11 $\boxed{10}$	70	$u_3 = 4$
Demand	80	20	60	160	
	$v_1 = 3$	$v_2 = 2$	$v_3 = 7$		

• **Iteration 0:** Perform the following steps:

1. Compute u_1, u_2, u_3 and v_1, v_2, v_3 from the following linear system:

$$u_1 + v_3 = 1, \;\; u_2 + v_1 = 3, \;\; u_2 + v_2 = 2, \;\; u_3 + v_1 = 7, \;\; u_3 + v_3 = 11.$$

   Choose $u_2 = 0$, and solve this system, we obtain $v_1 = 3$, $v_2 = 2$, $v_3 = 7$, $u_1 = 6$, $u_2 = 0$ and $u_3 = 4$. Compute $\bar{c}_{ij} = u_i + v_j - c_{ij}$ and store them into the tableau:

$$\bar{c}_{11} = -8, \;\; \bar{c}_{12} = -8, \;\; \bar{c}_{13} = 0, \;\; \bar{c}_{21} = 0, \;\; \bar{c}_{22} = 0, \;\; \bar{c}_{23} = 1, \;\; \bar{c}_{31} = 0, \;\; \bar{c}_{32} = -3, \;\; \bar{c}_{33} = 0.$$

2. Since $\bar{c}_{23} > 1$ (the unique value). This BFS X is not yet optimal, we continue.
3. We choose x_{23} as the entering variable.
4. We can form a closed loop as

$$\mathscr{L} = \left\{(2,3)^+, \; (3,3)^-, \; (3,1)^+, \; (2,1)^-\right\}.$$

   It is easy to find

$$\mathscr{L}_+ = \{(2,3), (3,1)\} \quad \text{and} \quad \mathscr{L}_- = \{(3,3), (2,1)\}.$$

5. We choose $x_{i_0 j_0} = \min\{x_{33}, x_{21}\} = \min\{10, 20\} = 10 = x_{33}$ to be the leaving variable.

6. We construct the new basic feasible solution $\bar{X} = (\bar{x}_{ij})$ as

$$\begin{cases} \bar{x}_{21} = x_{21} - x_{33} = 20 - 10 = 10 \\ \bar{x}_{33} = 0 \\ \bar{x}_{23} = 0 + x_{33} = 10 \\ \bar{x}_{31} = x_{31} + x_{33} = 60 + 10 = 70 \end{cases} \quad \text{or } \bar{X} = \begin{bmatrix} 0 & 0 & 50 \\ 10 & 20 & 10 \\ 70 & 0 & 0 \end{bmatrix} \text{ with } f(\bar{X}) = 670.$$

In this case, we obtain a new transportation tableau:

D_j / S_i	D_1	D_2	D_3	Supply	
S_1	5	4	1 $\boxed{50}$	50	$u_1 = -5$
S_2	3 $\boxed{10}$	2 $\boxed{20}$	6 $\boxed{10}$	40	$u_2 = 0$
S_3	7 $\boxed{70}$	9	11	70	$u_3 = 4$
Demand	80	20	60	160	
	$v_1 = 3$	$v_2 = 2$	$v_3 = 6$		

• **Iteration 1:** Perform the following steps:

1. Compute new $u_i, (i = 1,2,3)$ and new $v_j, (j = 1,2,3)$ by solving the following linear system:

$$u_1 + v_3 = 1, \;\; u_2 + v_1 = 3, \;\; u_2 + v_2 = 2, \;\; u_2 + v_3 = 6, \;\; u_3 + v_1 = 7.$$

Choose $u_2 = 0$ we can easily obtain $v_1 = 3, v_2 = 2, v_3 = 6, u_1 = -5$, and $u_3 = 4$. Next, we compute $\bar{c}_{ij} = u_j + v_j - c_{ij}$ and obtain

$$\bar{c}_{11} = -7, \; \bar{c}_{12} = -7, \; \bar{c}_{13} = 0, \; \bar{c}_{21} = 0, \; \bar{c}_{22} = 0, \; \bar{c}_{23} = 0, \; \bar{c}_{31} = 0, \; \bar{c}_{32} = -3, \; \bar{c}_{33} = -1.$$

2. Since $\bar{c}_{ij} \leq 0$ for all $(i,j) \in \mathscr{U}$, the current basic feasible solution $\bar{X}$ is optimal. We terminate the algorithm.

Finally, we obtain an optimal solution $X^* = \begin{bmatrix} 0 & 0 & 50 \\ 10 & 20 & 10 \\ 70 & 0 & 0 \end{bmatrix}$ with the optimal value $f(X^*) = 670$.

7.3.4 Unbalanced transportation problems

A transportation problem in which the total supply does not equal the total demand is called an **unbalanced transportation** problem. There are two types of unbalanced transportation problems. The first type consists of transportation problems in which the **total supply exceeds the total demand**.

An example can be obtained if we reduce the demand for city 1 in the Powerco example to 40 million kwh and keep other data unchanged. The formulation in (7.1) is still valid and feasible in this case, but formulation (7.2) becomes infeasible. To take advantage of the computational efficiency of the second formulation, we can transform the problem into a balanced problem, by adding a dummy demand point that has a demand equal to the amount of excess supply, and assigning a zero unit cost of shipments to the dummy demand point. In some situations, any unused supply is charged a holding fee. In those situations, let the unit cost to ship to the dummy demand point be the unit holding cost. In the optimal solution for the resulting balanced problem, the amount of shipments to the dummy demand point is exactly the amount of unused supply.

Example 7.6. For the Powerco example with the demand for city 1 being 40 million kwh, adding the dummy demand point results in a change in its data:

From/to	City 1	City 2	City 3	City 4	Dummy City 5	Supply
Plant 1	\$8	\$6	\$10	\$9	\$0	35
Plant 2	\$9	\$12	\$13	\$7	\$0	50
Plant 3	\$14	\$9	\$16	\$5	\$0	40
Demand	40	20	30	30	5	

By solving the balanced transportation problem with the above data, we can obtain an optimal solution to the unbalanced problem.

In the second type of unbalanced transportation problems, **the total demand exceeds the total supply**. We can construct an example of this type by reducing the supply at plant 1 in the Powerco example to 30 million kwh and keeping other data as in the original Example 7.3. In this case, both formulations in (7.1) and (7.2) become infeasible, as it is impossible to meet all the demand with the available supply. A common approach to modeling such situations is to impose a penalty for unmet demand at each demand point. To formulate the problem as a balanced transportation problem, we add a dummy supply point that has a supply equal to the amount of excess demand, and let the unit cost for shipments from the dummy supply point be the unit penalty for unmet demand. In the optimal solution for the resulting balanced problem, the amount of shipments from the dummy supply point is exactly the amount of unmet demand.

Example 7.7. Consider the Powerco example with the supply at plant 1 being 30 million kwh. Suppose that other data are as given in Example 7.3, and that the penalty for each million kwh of unmet demand is \$2 for city 1, \$1 for city 2, \$1 for city 3 and \$4 for city 4. After adding the dummy demand point, the data becomes

From/to	City 1	City 2	City 3	City 4	Supply
Plant 1	\$8	\$6	\$10	\$9	30
Plant 2	\$9	\$12	\$13	\$7	50
Plant 3	\$14	\$9	\$16	\$5	40
Dummy Plant 4	\$2	\$1	\$1	\$4	5
Demand	45	20	30	30	

The above data lead to a balanced transportation problem, which we can solve to find an optimal solution to the unbalanced problem with excess demand.

7.4 The minimum cost network flow problem and its extensions

The minimum cost network flow problem (MCNFP) is to find a feasible flow in a network that achieves the minimum total cost.

7.4.1 Problem description and mathematical formulation

Problem description: A network $\mathscr{N} = (V,A)$ consists of a set V of **nodes** (or vertices), and a set A of **arcs** (or edges). Each arc is an ordered pair of nodes; the arc from node i to node j is denoted by (i,j). For any arc (i,j), we say that i is the start node and j is the end node. The arc (i,j) is said to be outgoing from node i, incoming to node j. In the type of networks we consider, there is at most one arc from each node to each other node. It is fine to have an arc from node i to j, and another arc from j to i. If we use $V \times V$ to denote the Cartesian product of V and V (i.e., the set of all ordered pairs of elements of V), then A is a subset of $V \times V$.

In an MCNFP, each node $i \in V$ of the network has a **net supply** s_i, which may be positive, zero, or negative. Each arc (i,j) is associated with a unit cost c_{ij}, an

upper bound of the flow on this arc denoted by u_{ij}, and a **lower bound** for flow on this arc denoted by l_{ij}. It is possible for u_{ij} to be ∞, and the lower bound l_{ij} is usually 0 unless explicitly stated otherwise.

Given the network $\mathcal{N} = (V,A)$ as described above, the goal of the MCNFP is to assign a number x_{ij} to each arc (i,j), which is the **flow** on that arc, such that:

(1) the flow bounds on all arcs are observed;
(2) the **flow conservation constraints** for all nodes are satisfied; and
(3) the total cost is minimized.

The flow conservation constraint for each node is also called the **balance equation** for each node. The balance equation for a given node i requires the difference between the total flow out of node k and the total flow into node k to be equal to the net supply s_i. This constraint can be written as

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{k:(k,i)\in A} x_{ki} = s_i.$$

Note that i is a fixed index in the above constraint. The expression $\sum_{j:(i,j)\in A} x_{ij}$ is the sum of x_{ij} over all nodes j such that (i,j) is an arc, and therefore gives the total amount of flow out of node i. Similarly, the expression $\sum_{k:(k,i)\in A} x_{ki}$ is the sum of x_{ki} over all nodes k such that (k,i) is an arc, and gives the total amount of flow into node i.

Example 7.8. Consider a network shown in Figure 7.6 below: Here, $V = \{1,2,3,4\}$

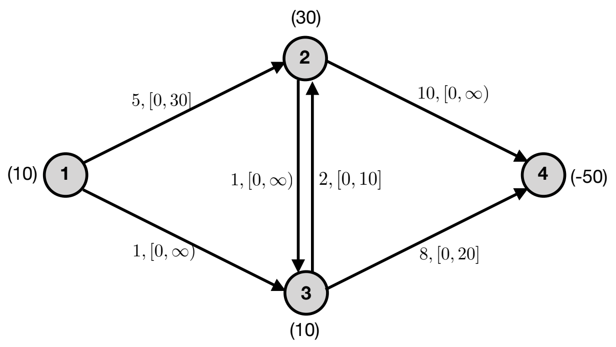


Fig. 7.6 A network representation of an MCNFP.

is the set of nodes, and $A = \{(1,2),(1,3),(2,3),(3,2),(2,4),(3,4)\}$ is the set of arcs. The numbers next to the nodes are net supplies (e.g., the net supply for node 1 is 10, and net supply for node 4 is -50).

On each of the arcs, the first number is the unit cost and the interval gives flow bounds. For example on arc $(1,3)$ the unit cost is 1, the lower bound of flow is 0, and the upper bound is ∞.

Mathematical model: Based on the problem described above, the following is a general formulation of an MCNFP, in which $\sum_{(i,j)\in A} c_{ij}x_{ij}$ is the sum of cost over all arcs (i,j).

$$\begin{cases} \min\limits_{x_{ij}} \sum\limits_{(i,j)\in A} c_{ij}x_{ij} \\ \text{s.t.} \quad \sum\limits_{j:(i,j)\in A} x_{ij} - \sum\limits_{k:(k,i)\in A} x_{ki} = s_i \quad \text{for each } i \in V \quad \text{(balance equation)} \\ \quad\quad l_{ij} \le x_{ij} \le u_{ij} \text{ for each } (i,j) \in A \quad\quad\quad \text{(flow bounds)} \end{cases} \tag{7.3}$$

Summing up all the balance equations will result in an equality $\sum_{i\in N} s_i = 0$ as the left hand sides of all balance equations be canceled out. This implies that $\sum_{i\in N} s_i = 0$ is a necessary condition for the MCNFP to be feasible.

GAMS model: We can model this problem into a GAMS file, named by `mcnfp.gms`. In this file, the set `A` is declared as a subset of the Cartesian product set $V \times V$. The latter set contains 16 elements, each being an ordered pair of nodes, among which 6 belong to `A`. In general, the following statement

```
SET set1(set2) ;
```

declares `set1` as a subset of the larger set `set2`. We will use `A` as a filter in conditional index operations and conditional equation assignments, and as the index of summation in the equation that defines the objective function.

One caution about using the set `A` in `mcnfp.gms` is to avoid declaring parameters, variables, or equations over it. In general, one can use a one-dimensional set (a set that contains a sequence of elements) or a product of multiple one-dimensional sets, but not a subset of the product of two sets, as the domain of a parameter, variable or equation. In `mcnfp.gms`, we declare parameters `c` and `u` over (V,V), even though we will only use values of those parameters for (i,j) in the set `A`. The values of parameters `c` and `u` that are not explicitly assigned in the declaration statement are zero by default; for example, `c('1','1')` and `u('1','4')` are both zero by default.

The variable `x` is declared over `(V, V)`, so it contains 16 components, one for each ordered pair of nodes. However, as will be clear from subsequent statements, only `x(i, j)` for `(i,j)` in the set `A` will appear in the equations. The statement

```
alias (V, j);
```

declares `j` as an alias of the set `V`, i.e., an alternative name for `V`. Aliases are useful when a set has to be referred to by more than one name. For example, in the definition of the balance equation,

```
balance(V)..sum(j$A(V, j), x(V, j)) - sum(j$A(j,V), x(j,V)) =e= S(V);
```

`V` and `j` serve very different roles. The set `V` appears in the parentheses after the equation name `balance` before the two dots, and is the "controlling set" or "controlling index" of the equation. GAMS will generate an equation for each element of `N`. In each such equation, `V` is fixed. In contrast, `j` is used as the index of summation in the equation. The expression `sum(j$A(V, j), x(V,j))` in the equation gives the sum of `x(V, j)` over all indices `j` such that `(V, j)` is in the set `A`. Compiling `mcnfp.gms` will generate the following excerpt from `mcnfp.lst`:

```
balance(1)..  x(1,2) + x(1,3) =E= 10 ;
balance(2)..  - x(1,2) + x(2,3) + x(2,4) - x(3,2) =E= 30 ;
balance(3)..  - x(1,3) - x(2,3) + x(3,2) + x(3,4) =E= 10 ;
```

Note that one cannot use the same name of a set as both the controlling set and the index of summation.

In the expression `sum(j$A(V, j), x(V, j))`, a dollar condition is used to control the domain of the summation operator. With the dollar condition the summation is done over all indices `j` for which `(V, j)` belongs to the set `A`. Without that dollar condition, the expression `sum(j, x(V, j))` would give the sum of `x(V, j)` over all `j` for the fixed `V`, which is incorrect. The statement

```
x.up(A) = u(A);
```

specifies `u(i,j)` as the upper bound for `x(i, j)` for each `(i, j)` in `A`.

In summary, we can write the complete GAMS code for this example as follows:

```
*mcnfp.gms
SETS
        V       "nodes" /1*4/,
        A(V, V) "arcs"  /1.2, 1.3, 2.3, 3.2, 2.4, 3.4/;
ALIAS (V, j);
PARAMETERS
        S(V)    "net supplies at nodes"
          /1 10, 2 30, 3 10, 4 -50/
        c(V, V) "unit cost on arcs"  /
          1.2   5
          1.3   1
          2.3   1
          3.2   2
          2.4   10
          3.4   8/,
        u(V, V) "upper bounds" /
          1.2   30
          1.3   100
          2.3   100
          3.2   10
          2.4   100
          3.4   20/;
VARIABLES totalcost, x(V, V);
POSITIVE VARIABLE x;
EQUATIONS balance(V), obj;
    balance(V).. sum(j$A(V, j), x(V, j)) - sum(j$A(j, V), x(j, V)) =E= S(V);
    obj..         totalcost =E= sum(A, c(A)*x(A));
    x.up(A) = u(A);
MODEL MCNFP /ALL/;
SOLVE MCNFP USING LP MINIMIZING totalcost;
```

7.4.2 Integrality of MCNFP optimal solutions

The MCNFP as formulated in (7.3) is a special case of linear programming. Any algorithm for linear programming, including the simplex method introduced earlier in this course, can be directly applied to solve the MCNFP. On the other hand, the special structure of the MCNFP makes it possible to substantially simplify the simplex method. The resulting simplified algorithm is called the **network simplex** method, which is more efficient than the general simplex method when applied to an MCNFP. The detailed procedure of the network simplex method is beyond the scope of this course. As in the general simplex method, the network simplex method updates a basic feasible solution in each iteration. Instead of maintaining a simplex tableau, the network simplex method updates the basic feasible solution and the reduced costs based on the network structure.

An important feature of the MCNFP is the integrality of its solutions, as stated in the following theorem. The proof of the theorem is omitted due to the scope of this course.

Theorem 7.3. *If an MCNFP has an optimal solution, and all the net supplies and all the flow bounds are integers, then the MCNFP has an integer optimal solution (i.e., an optimal solution with all components being integer).*

It is possible for an MCNFP to have more than one optimal solutions. According to the above theorem, at least one optimal solution is entirely integer. In addition to its computational efficiency, the network simplex method guarantees finding an integer optimal solution for an MCNFP satisfying conditions in the above theorem.

To inform GAMS to use the "network simplex method" to solve a model, one needs to place the following statements with the standard model and solve statements:

```
MODEL MCNFP /ALL/;
OPTION LP = cplex;
$onecho > cplex.opt
lpmethod 3
$offecho
MCNFP.optfile = 1;
SOLVE MCNFP USING LP MINIMIZING totalcost;
```

The word `MCNFP` that appears in the above statements is the name of the model chosen by the modeler. The statement

```
OPTION LP = cplex;
```

is an option statement; it sets CPLEX as the LP solver. The three lines from `$onecho` to `$offecho` creates a file named `cplex.opt` in the GAMS project folder and writes a line "`lpmethod 3`" into the file. The statement

```
        MCNFP.optfile = 1;
```

informs the solver to read the option file `cplex.opt`. After reading that file, the solver will set the parameter `lpmethod` to 3. The CPLEX solver is equipped with several alternative algorithms to solve linear programs, including the primal simplex method that we studied in this course, the dual simplex method, the network simplex method, the interior point method (also called the barrier method), and others. Setting `lpmethod` to 3 specifies the network simplex method as the algorithm to be used to solve the model `MCNFP`.

7.4.3 Balanced transportation problems as minimum cost network flow problems

Balanced transportation problems can be treated as a special case of the minimum cost network flow problems (MCNFP). It is natural to consider the supply points and demand points as nodes in the graph, and connect each supply point and each demand point by an arc that points from the supply point to the demand point. For each supply point, let its net supply be its supply capacity. For each demand point, let its net supply be the negative of its demand.

Example 7.9. Consider the transportation problem discussed in the previous lecture.

From/to	City 1	City 2	City 3	City 4	Supply
Plant 1	\$8	\$6	\$10	\$9	35
Plant 2	\$9	\$12	\$13	\$7	50
Plant 3	\$14	\$9	\$16	\$5	40
Demand	45	20	30	30	

To formulate it as an MCNFP, we create two sets

$$V = \{p_1, p_2, p_3, c_1, c_2, c_3, c_4\},$$

$$A = \{(p_i, c_j),\ i = 1,2,3,\ j = 1,2,3,4\},$$

and assign the net supplies of p_1, p_2, p_3 to be $35, 50, 40$, and the net supplies of c_1, c_2, c_3, c_4 to be $-45, -20, -30, -30$ respectively. The unit cost of each arc is the unit shipping cost between the two endpoints of this arc, as given in the table. Since the problem does not specify bounds on the amount of traffic for any arcs, we let $l_{ij} = 0$ and $u_{ij} = \infty$ for all $(i, j) \in A$.

Note that only a balanced transportation problem can be treated as a special MCNFP. An unbalanced transportation problem needs to be transformed into a balanced problem before being modeled as an MCNFP. Also note that the graph generated by a balanced transportation problem has a special structure: each node in this graph is either a supply point or a demand point, and each arc is from a supply point to a demand point. There are no arcs between two supply points or arcs between two demand points. Such a graph is called a bipartite graph.

7.4.4 Shortest path problems as minimum cost network flow problems

Problem description: A shortest path problem seeks a shortest path from a given node to another given node in a network. In such a problem, a network consists of a set V of nodes, and a set A of (directed) arcs, each of which is an ordered pair of nodes. Each arc (i, j) has a cost $c_{ij} > 0$. A path from a node s to another node y is a sequence of consecutive arcs joining s to y: the first arc in such a path starts from s, each arc in the sequence starts from where the previous arc ends, and the last arc ends with y. The cost of a path is the sum of costs of all arcs in it.

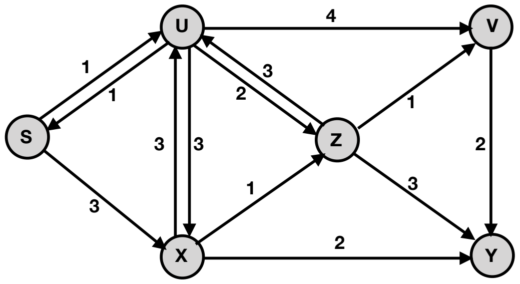


Fig. 7.7 The network for a shortest path problem. This graph has 6 notes: s, u, v, x, y, and z, and 13 arcs. The numbers on arcs are costs (or the lengths of the arcs).

There are specific algorithms for finding shortest paths, such as the Dijkstra algorithm. Details of those algorithms are beyond the scope of this course. Below, we describe how to formulate and solve a shortest path problem as a special MCNFP.

Mathematical model: Suppose we want to find a shortest path from node s to node y. To model the problem as an MCNFP, we specify the net supply of s to be 1, the net supply of y to be -1, and the net supply to each other node to be 0. We specify the lower bounds and upper bounds on all arcs to be 0 and ∞ respectively. Note that the properties about net supplies and flow bounds are not given in the original network in which a shortest path is to be found. We add such properties to the network arbitrarily in order to model it as an MCNFP. In addition, let the unit cost on each arc (i,j) be the cost c_{ij} as given in the original network.

With the above information we can solve the resulting MCNFP using the network simplex method to find an integer optimal solution, provided that an optimal solution exists. In such an integer optimal solution, the flow on each arc is either 1 or 0, and the arcs with one unit of flow are along a shortest path from s to y. To understand why this is true, first consider the node s. Because the net supply for s is 1, there must exist an arc, say (s,x), that starts with s and carries positive flow. If node x is not y, there must be an arc out of x with positive flow, because the net supply of x is 0. We can repeat this procedure to follow a sequence of consecutive arcs with positive flow, starting from node s. Since the flow has minimum cost, the arcs that carry positive flow do not form any cycle. For this reason, we will never revisit any node and will eventually reach the node y. This gives a path from s to y along which all arcs have positive flow. By the integrality of the flow, the amount of flow on each arc along this path is at least 1. Indeed, the flow on each arc along this path must be exactly 1, since any extra flow will increase the total cost and therefore will not be present in a minimum cost flow. Similarly, the flow on any arc not belonging to this path must be zero, for otherwise we would be able to remove flow from those arcs to further reduce the cost. This path that carries one unit of flow from s to y must be a shortest path from s to y.

A complete GAMS code: We now give a complete GAMS code for this example.

```
*shortestpath.gms
SETS
        V /s,u,x,v,z,y/,
        A(V, V) /s.(u,x), u.(s,x,z,v), x.(u,z,y), z.(u,v,y), v.y/;
ALIAS (V, j);
PARAMETERS
        S(V)    "net supplies for nodes" /s 1, y -1/,
        c(V, V) "unit cost on arcs"
         /s.u   1
          s.x   3
```

```
            u.s   1
            u.x   3
            u.z   2
            u.v   4
            x.u   3
            x.z   1
            x.y   2
            z.u   3
            z.v   1
            z.y   3
            v.y   2/;
  VARIABLES totalcost, x(V, V);
  POSITIVE VARIABLE  x;
  EQUATIONS balance(V), obj;
      balance(V).. sum(j$A(V, j), x(V, j)) - sum(j$A(j, V), x(j, V)) =E= s(V);
      obj..        totalcost =E= sum(A, c(A)*x(A));
  MODEL shortestpath /balance, obj/;
  OPTION LP = cplex;
  $onecho > cplex.opt
      lpmethod 3
  $offecho
  shortestpath.optfile = 1;
  SOLVE shortestpath USING LP MINIMIZING totalcost;
```

7.4.5 The assignment problem

The assignment problem is an important type of problem that arises from many practical contexts. The goal of an assignment problem is to assign members of one group to members of another group to maximize the total “value” of assignments, or to minimize the total cost. Typical examples include assigning employees to jobs, machines to tasks, and so on.

Example 7.10. In this subsection, we present one example to illustrate this type of problems.

Problem description: Machineco needs to assign four jobs to four machines, with each machine handling a different job. The time required to set up each machine for completing each job is shown below. Help Machineco to minimize the total setup time needed to complete the four jobs.

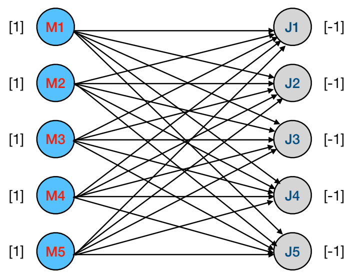


Fig. 7.8 A bipartite graph illustrates an assignment problem: 5 machines are assigned to perform 5 jobs

	Time (Hours)			
Machine	Job 1	Job 2	Job 3	Job 4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

Mathematical model: Binary linear programming and its LP relaxation. Let c_{ij} denote the time to set up machine j to do job i. Since Machineco's problem is to decide which job to assign to each machine, we can use a binary variable x_{ij} to represent the decision on whether to assign job i to machine j. A binary variable is a variable that takes the value either 0 or 1. We use $x_{ij} = 1$ to represent the decision of assigning job i to machine j, and $x_{ij} = 0$ to represent not assigning job i to machine j. With these variables, we formulate the following problem:

$$\begin{cases} \min\limits_{x_{ij}} \sum_{i=1}^{4}\sum_{j=1}^{4} c_{ij}x_{ij} & \\ \text{s.t.} \ \sum_{j=1}^{4} x_{ij} = 1, \ \ i = 1,2,3,4 & \text{(Each job assigned to a machine)} \\ \qquad \sum_{i=1}^{4} x_{ij} = 1, \ \ j = 1,2,3,4 & \text{(Each machine assigned a job)} \\ \qquad x_{ij} \in \{0,1\}, \ \ i = 1,2,3,4, \ \ j = 1,2,3,4 & \end{cases} \tag{7.4}$$

It is important to note that the problem in (7.4) is NOT a linear program, because the constraints that require x_{ij} to be either 0 or 1 are not linear constraints. Indeed, the problem (7.4) is an integer programming problem, and we will introduce techniques for general integer programming problems later. However, as we explain below, the special structure of (7.4) makes it possible to solve it without using integer programming techniques.

Consider the following linear programming problem:

$$\begin{cases} \min\limits_{x_{ij}} \sum\limits_{i=1}^{4}\sum\limits_{j=1}^{4} c_{ij}x_{ij} & \\ \text{s.t.} \ \ \sum\limits_{j=1}^{4} x_{ij} = 1, \ \ i = 1,2,3,4 & \text{(Each job assigned to a machine)} \\ \qquad \sum\limits_{i=1}^{4} x_{ij} = 1, \ \ j = 1,2,3,4 & \text{(Each machine assigned a job)} \\ \qquad 0 \le x_{ij} \le 1, \ \ i = 1,2,3,4, \ \ j = 1,2,3,4 & \text{(A relaxation of the binary constraint).} \end{cases} \tag{7.5}$$

The difference between (7.4) and (7.5) is about their last group of constraints. The constraint $x_{ij} \in \{0,1\}$ for each (i,j) pair can be equivalently stated as $0 \le x_{ij} \le 1$ and x_{ij} is integer. If we can find an integer optimal solution to (7.5), then that solution will be an optimal solution to (7.4). In general, a linear program with an optimal solution does not necessarily have an integer optimal solution. However, as stated in Theorem 7.3, an MCNFP with integer data always has an integer optimal solution provided it has an optimal solution.

Reformulation as an MCNFP: Next, we observe that the problem in (7.5) is a special MCNFP. To this end, consider a graph with 8 nodes, with 4 nodes representing jobs and the other 4 representing machines. Connect each job node with each machine node by an arc that points from the job to the machine, and let each job node have net supply 1 and each machine node have net supply -1. Let c_{ij} be the unit cost on the arc (i,j) that points from job i to machine j. Finally, let the lower bounds and upper bounds for flow on each arc to be 0 and 1, respectively. The MCNFP associated with such a graph can be written as

$$\begin{cases} \min\limits_{x_{ij}} \sum\limits_{i=1}^{4}\sum\limits_{j=1}^{4} c_{ij}x_{ij} & \\ \text{s.t.} \ \ \sum\limits_{j=1}^{4} x_{ij} = 1, \ \ i = 1,2,3,4 & \text{(Balance equation for job } i\text{)} \\ \qquad -\sum\limits_{i=1}^{4} x_{ij} = -1, \ \ j = 1,2,3,4 & \text{(Balance equation for machine } j\text{)} \\ \qquad 0 \le x_{ij} \le 1, \ \ i = 1,2,3,4, \ \ j = 1,2,3,4. & \end{cases} \tag{7.6}$$

The MCNFP (7.6) is clearly equivalent to (7.5). Accordingly, one can solve (7.5) using the network simplex method to find an integer optimal solution, which will be an optimal solution to the integer program (7.4).

A complete GAMS model: The model `assignment` formulated in assignment.gms consists of all constraints and the objective function of (7.5), except the constraints $x_{ij} \leq 1$ for each (i, j). It is fine to omit those constraints, because the balance equations $\sum_{j=1}^{4} x_{ij} = 1$ for each i and the non-negativity constraints on x_{ij} automatically imply $x_{ij} \leq 1$. With the statement `lpmethod 3` in the option file, the model is solved by the CPLEX solver with the network simplex method, and an integer optimal solution is found by the solver.

```
*assignment.gms
SETS
        i  "jobs"     / job1*job4 /,
        j  "machines" / m1*m4 /;
TABLE
        c(i,j)
                      m1       m2       m3     m4
          job1        14        2       7      2
          job2        5        12       8      4
          job3        8         6       3      6
          job4        7         5       9      10;
VARIABLES
        x(i,j), z;
POSITIVE VARIABLE x;
EQUATIONS
        cost, jobEq(i), machine(j);
    cost ..        SUM((i, j), c(i, j)*x(i, j)) =E= z;
    jobEq(i) ..    SUM(j, x(i, j))              =E= 1;
    machine(j) .. SUM(i, x(i, j))               =E= 1;
MODEL assignment /ALL/ ;
OPTIONS LP = cplex;
$onecho > cplex.opt
   lpmethod 3
$offecho
assignment.optfile = 1;
SOLVE assignment USING LP MINIMIZING z;
```

7.5 Exercises

Exercise 7.1. A company supplies goods to three customers, who require 40, 50, and 40 units respectively. The company has three warehouses, each of which has 30 units available. The costs of shipping 1 unit from each warehouse to each customer are shown in the table below.

	To		
From	Customer 1	Customer 2	Customer 3
Warehouse 1	\$15	\$35	\$25
Warehouse 2	\$10	\$50	\$40
Warehouse 3	\$20	\$40	\$30

There is a penalty for unmet demand: With customer 1, a penalty cost of \$70 per unit is incurred; with customer 2, \$75 per unit; and with customer 3, \$65 per unit. The company needs to minimize the total cost.

1. Does the following formulation correctly model the problem? Why? (Here, x_{ij} denotes the amount to ship from warehouse i to customer j, and c_{ij} is the corresponding unit shipping cost.)

$$\min \textstyle\sum_{i=1}^3 \sum_{j=1}^3 c_{ij}x_{ij} + 70(40 - \sum_{i=1}^3 x_{i1}) + 75(50 - \sum_{i=1}^3 x_{i2}) + 65(40 - \sum_{i=1}^3 x_{i3})$$

$$\text{s.t.} \quad \textstyle\sum_{j=1}^3 x_{ij} \leq 30, \quad i = 1, \cdots, 3 \text{ (supply constraints)}$$

$$x_{ij} \geq 0, \quad i = 1, \cdots, 3, j = 1, \cdots, 3$$

2. Formulate the problem as a balanced transportation problem. Create a GAMS file named *company.gms* to solve the problem. (The optimal value is \$4950.)

Exercise 7.2. Given the following two balanced transportation problems:

From/to	D_1	D_2	D_3	Supply
S_1	\$3	\$3	\$4	30
S_2	\$2	\$2	\$2	60
S_3	\$5	\$5	\$3	10
Demand	25	35	40	100

From/to	D_1	D_2	D_3	D_4	Supply
S_1	\$4	\$3	\$3	\$4	20
S_2	\$2	\$4	\$5	\$2	30
S_3	\$4	\$2	\$1	\$6	50
Demand	25	25	15	35	100

Carry out the following tasks:

(a) Use the north-west corner method to find a BFS for each problem. Is that BFS degenerated?

(b) Use the least-cost method to find a BFS for each problem. Is that BFS degenerated?

(c) Apply the simplex method for transportation problems to solve these problems.

Exercise 7.3. Each year, Data Corporal produces 400 computers in Boston and 300 computers in Raleigh. Los Angeles customers must receive 400 computers, and Austin customers must receive 300 computers. Data Corporal can ship computers from each production city to each demand city directly, or through Chicago. The costs of sending a computer among pairs of cities are shown below.

	To		
From	Chicago	Austin	Los Angeles
Boston	\$80	\$220	\$280
Raleigh	\$100	\$140	\$170
Chicago	-	\$40	\$50

1. Suppose that no more than 200 computers can be shipped between each pair of cities. Formulate an MCNFP to minimize the total shipping cost, and provide a graph to include all information about the MCNFP. Create a GAMS file named *datacorporal.gms* to solve the problem. (The optimal value is \$118000.)
2. Suppose now the total amount of computers shipped through Chicago cannot exceed 100 (that is, the total amount of computers entering Chicago cannot exceed 100). How do you formulate the problem as an MCNFP? Note that an MCNFP is only allowed to have two types of constraints: bound constraints on arcs, and balance equations for nodes. If you add a constraint that does not belong to one of these two types, then the new model is not a MCNFP. (The optimal value is \$133000.)

For part 2, do not hand in your GAMS code. Just explain how to modify the model (or the graph), and interpret the optimal solution and optimal value you find.

Exercise 7.4. Oilco has oil fields in San Diego and Los Angeles. The San Diego field can produce 500,000 barrels per day, and the Los Angeles field can produce 400,000 barrels per day.

Oil is sent from the fields to a refinery, in either Dallas or Houston (assume each refinery has unlimited capacity). To refine 100,000 barrels costs \$700 at Dallas and \$900 at Houston.

Refined oil is shipped to customers in Chicago and New York. Chicago customers require 400,000 barrels per day, and New York customers require 300,000 barrels per day.

The costs of shipping 100,000 barrels of oil (refined or unrefined) between cities are shown below.

	To(\$)			
From	Dallas	Houston	New York	Chicago
Los Angels	300	110	-	-
San Diego	420	100	-	-
Dallas	-	-	450	550
Houston	-	-	470	530

Formulate an MCNFP to minimize the total cost of meeting all demands; provide a graph to include all information about the MCNFP. Create a GAMS file *oilco.gms* to solve the problem (the optimal value is \$10,470).

Exercise 7.5. Consider a transportation problem where the input data is given by the following table:

From/To	Demand point 1	Demand point 2	Demand point 3	Demand point 4	Supply
Supply point 1	2	3	5	6	5
Supply point 2	2	1	3	5	10
Supply point 3	3	8	4	5	15
Demand	12	8	4	6	30

Here, we have three supply points with the supply capacities are given in the last column, and four demand points with the demand values are given in the last row. The value in the cell (i, j) provides the unit cost to ship from the supply point i to the demand point j.

(a) Code this problem in GAMS. Then, do not submit the code in Sakai, but run the code and present the result. Interpret the solution in the context of this transportation problem.

(b) Using one of the methods we described in class to find a basic solution for this problem (e.g., the northwest conner rule, or the minimum cost rule).

(c) Due to the lack of warehouse, we have to transfer some supply value Δ from Supply point 3 to Supply point 1 so that they become $15 - \Delta$ and $5 + \Delta$, respectively. Use the sensitivity analysis theory of LPs to compute the value of Δ so that we do not need to change the transportation route obtained in Question (a) (That is, it does not change the basis of this optimal solution).

(d) Now, assume that the unit cost given in the cells of the above table is the price which a transportation company will charge when it does transportation services. Modify the GAMS code in Question (a) to solve the problem of maximizing the total revenue this company makes. Interpret the resulting optimal solution in this case.

Exercise 7.6. A company produces cars in Atlanta, Boston, Chicago and Los Angeles. The cars are then shipped to warehouses in Memphis, Milwaukee, New York City, Denver and San Francisco. The number of cars available at each plant is given in the last column, and the capacity of each warehouse is given in the last row of the following table. In addition, each cell of this table presents a distance (in miles) between these cities.

	Memphis	Milwaukee	New York	Denver	San Francisco	Plants
Atlanta	371	761	841	1398	2496	5000
Boston	1296	1050	206	1949	3095	6000
Chicago	530	87	802	996	2142	4000
Los Angeles	1817	2012	2786	1059	379	3000
Warehouses	6000	4000	4000	2000	2000	18000

(a) Assume that the cost (in dollars) of shipping a car from plants to warehouses is given by the function $c(x) = \sqrt{x}$, where x is the distance in miles between two cities. Formulate this problem as a transportation problem. Then, model it in GAMS. Do not submit your code on Sakai, but run your code and interpret your result in detail.

(b) Assume that due to long distance, the company does not want to ship cars through two routes: from Boston to San Francisco, and from Los Angeles to New York. Formulate this problem as a minimum cost network flow problem with the cost function as in Question (a). Then, model it in GAMS. Do not submit your code in Sakai, but run your code and interpret your result in detail. Compare this result with the one in Question (a).

Exercise 7.7. The data given in the following table are distances of arcs between pairs of nodes i and j in a network, where $i, j = 1, \cdots, 11$. An empty cell indicates that there is no arc between that pair.

	To										
From	1	2	3	4	5	6	7	8	9	10	11
1		5	2	4							
2					3	4					
3				3	4	8					
4						6	3				
5						2		9			10
6								12	9		
7									9	11	
8									6		10
9											4
10											7

(a) Formulate an MCNFP, whose solution can be used to find the shortest path from node 1 to node 11.

(b) Code this MCNFP in GAMS. Do not submit the code in Sakai, but run your code and interpret your result in detail.

(c) Assume that for each route on arc (i, j), we need to pay a cost $c(x) = \sqrt{x}$, where x is the length of this arc (the distance between node i and node j). Reformulate this problem as an MCNFP, and code it in GAMS. Do not submit the code in Sakai, but run your code and interpret your result in detail. Compare this result with the one in Question (b).

Chapter 8

Introduction to Nonlinear Programming

8.1 Introduction

So far, we have focused on linear programming and its applications. In this chapter, we discuss nonlinear programming (NLP), which provides a more general framework to capture underlying structures of practical problems, to model complex applications in engineering, artificial intelligence, computer science, and so on. While theory and methods for LPs are well understood and developed, theory and methods for NLPs are still under intensive ongoing research. Here, we give an overview on some nonlinear programming problems at an introductory level.

8.1.1 Mathematical formulation

A general optimization problem in finite dimensions can be formulated as follows:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & f_0(x) \\ \text{s.t.} & f_i(x) \le 0, \;\; i = 1,\cdots,m, \\ & x \in \mathscr{X}. \end{cases} \tag{8.1}$$

Here, f_i, $i = 0,\cdots,m$ are $m+1$ functions from $\mathbb{R}^n$ to $\mathbb{R}$, and $\mathscr{X}$ is a nonempty, closed subset in $\mathbb{R}^n$. Clearly, maximization problems or problems with constraints of the form $f_i(x) = 0$ or $f_i(x) \ge 0$ can be easily converted into (8.1). We define

$$\mathscr{F} := \{x \in \mathbb{R}^n \mid x \in \mathscr{X},\; f_i(x) \le 0, \;\; i = 1,\cdots,m\} \tag{8.2}$$

as the feasible set of (8.1). In many problems, the feasible set $\mathscr{F}$ is explicitly expressed through constraint functions. In some other cases, $\mathscr{F}$ is given implicitly.

For instance, $\mathscr{F}$ can be the solution set of another problem. When $\mathscr{F} = \mathbb{R}^n$, we say that (8.1) is an unconstrained problem.

There are more than one way to classify optimization problems. Here, we clarify the difference between linear and nonlinear programming problems:

- **A linear programming problem** minimizes or maximizes a linear objective function of finitely many variables (say, $x_1, x_2, \cdots, x_n$) under finitely many linear constraints. In other words, if $f_0(x) = c^T x$ is a linear function, $f_i(x) = a_i^T x - b_i$ is an affine function for each $i = 1, \cdots, m$, and $\mathscr{X}$ is the whole space $\mathbb{R}^n$, the orthant $\mathbb{R}^n_+$, or more generally a polyhedron, then (8.1) is a linear program.
- If $\mathscr{X}$ in (8.1) is specified by constraint functions, and at least some of the constraints or the objective are nonlinear, continuous functions, then (8.1) is a **nonlinear programming problem**.

If $\mathscr{X}$ is specified by some integrality constraints, i.e., if some variables x_i are required to be integers, then (8.1) becomes a discrete optimization problem, or integer programming problem. We will give a basic introduction to integer programming in the next chapter. To read more about different classes of optimization problems, see, e.g., [17].

8.1.2 Applications and examples

Nonlinear programming have many direct applications. Below are some examples:

- **Engineering:** Nonlinear programming can be used to control and stabilize a system; optimize time, energy and material; design optimal structures in civil engineering, and so on.
- **Computer science:** computer vision, machine learning, image processing, graph theory.
- **Economics and finance:** equilibrium price, risk minimization, utility maximization, portfolio selection, expenditure minimization problems, etc.

Here are two specific examples.

1. A simple quadratic program:

$$\begin{cases} \min_x \sum_{j=1}^n (x_j - c_j)^2 \\ \text{s.t.} \ \sum_{j=1}^n A_{ij}x_j \le b_i, \ i = 1, \cdots, m. \end{cases}$$

The optimal solution of the above problem is the Euclidean projection of the point $c = (c_1, c_2, \cdots, c_n)^T$ onto the polyhedral set $\mathscr{P} := \{x \in \mathbb{R}^n \mid Ax \le b\}$.

2. A sparse solution recovery of a linear system (a related problem has been considered in the previous chapter):

$$\begin{cases} \min_x \sum_{j=1}^n |x_j| \\ \text{s.t.} \ \sum_{j=1}^n A_{ij}x_j = b_i, \ i = 1, \cdots, m. \end{cases}$$

This problem in the above form is an NLP because the objective function is nonlinear, but it can be converted into an LP by introducing more variables. A related problem is

$$\min_x \frac{1}{2} \sum_{i=1}^m \left(\sum_{j=1}^n A_{ij}x_j - b_i \right)^2 + \lambda \sum_{j=1}^n |x_j|,$$

where $\lambda > 0$ is a regularization parameter. The latter problem cannot be converted into an LP. These problems are commonly used in compressive sensing, a signal processing technique.

8.1.3 The focus of this chapter

Nonlinear programming is a very broad class of optimization problems. Here, we focus on the following two subclasses of problems.

- **Convex quadratic programs (QP).** Quadratic programming is a natural extension of linear programming. We will discuss applications of QP in portfolio selection and support vector machines, as well as some basic properties and solution methods.
- **Convex programs (CP).** Convex programs are nonlinear programs with convex objective functions and convex feasible sets. They cover linear programs and convex quadratic programs as special cases. Many problems in compressive sensing, signal and image processing, machine learning, statistics, and data science are convex programs.

8.2 Multivariable calculus, linear algebra, and convexity

First, we recall some basic concepts of multivariable calculus. Then, we review symmetric positive semidefinite matrices. Finally, we provide definitions and basic properties of convex sets and convex functions. See, e.g., [20] for details on multivariable calculus.

8.2.1 Multivariable functions: A review

A real-valued function f of n variables $x_1, \cdots, x_n$ is a mapping from the n variables to a scalar, denoted as

$$f : \mathbb{R}^n \to \mathbb{R} \quad \text{or} \quad x \mapsto y = f(x).$$

Here are some examples.

1. The function $f(x_1, x_2) = 4x_1 + 2x_2 + 4$ is a linear function of two variables x_1 and x_2.
2. The function $f(x_1, x_2) = x_1^2 + 2x_2^2 + 3x_1x_2 + 2x_1 + 4x_2$ is a quadratic function of two variables x_1 and x_2.
3. The function $f(x_1, x_2) = \log(1 + x_1^2 + x_2^2)$ is a nonlinear function of two variables x_1 and x_2.
4. The function $f(x) = \sum_{i=1}^n x_i \log(x_i)$ is a nonlinear function of n variables $x_1, x_2, \cdots, x_n$.

The set of points x for which $f(x)$ is defined is called the domain of f, denoted by $\mathrm{dom}(f)$. For instance, the domain of the first three functions above is the whole space, while the domain of the last function is $\mathbb{R}^n_{++}$, the set of $x \in \mathbb{R}^n$ with strictly positive entries.

Gradients and Hessian matrices of multivariable functions. The gradient of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at a given point $x \in \mathbb{R}^n$ is defined as

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \cdots, \frac{\partial f(x)}{\partial x_n} \right)^T,$$

which is the vector of all partial derivatives of f with respect to all variables $x_1, x_2, \cdots, x_n$. Here, $\frac{\partial f(x)}{\partial x_n}$ stands for the partial derivative of f with respect to x_n at x. For example, if $f(x) = 2x_1 + 3x_2 + 4$, then $\nabla f(x) = (2, 3)^T$. If $f(x) = 2x_1^2 + 4x_2^2 + 3x_1x_2 + 2x_1 + x_2$, then $\nabla f(x) = (4x_1 + 3x_2 + 2, 8x_2 + 3x_1 + 1)^T$. The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ can be considered as a mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$, called the gradient mapping: $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$.

The Hessian matrix of a twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at x is defined as

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1{}^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n{}^2} \end{bmatrix},$$

which is a square matrix formed by all the second order partial derivatives of f. When the second order partial derivatives of f are continuous, the Hessian matrix $\nabla^2 f(x)$ is a symmetric matrix, i.e., $\nabla^2 f(x) = \nabla^2 f(x)^T$.

Below are some examples:

1. If $f(x) = a^T x$ then $\nabla f(x) = a$ and $\nabla^2 f(x) = 0$, the zero matrix.
2. If $f(x) = \frac{1}{2} x^T Q x + q^T x$ a quadratic function, where Q is symmetric, then
$$\nabla f(x) = Qx + q \quad \text{and} \quad \nabla^2 f(x) = Q.$$
3. If $f(x) = e^{x_1^2 + x_2^2}$, then
$$\nabla f(x) = \begin{pmatrix} 2x_1 e^{x_1^2+x_2^2} \\ 2x_2 e^{x_1^2+x_2^2} \end{pmatrix}, \text{ and } \nabla^2 f(x) = \begin{bmatrix} 2e^{x_1^2+x_2^2}(1+2x_1^2) & 4x_2 x_1 e^{x_1^2+x_2^2} \\ 4x_1 x_2 e^{x_1^2+x_2^2} & 2e^{x_1^2+x_2^2}(1+2x_2^2) \end{bmatrix}.$$

The Hessian matrix of a function $f : \mathbb{R}^n \to \mathbb{R}$ can also be considered as a mapping from $\mathbb{R}^n$ to $\mathbb{R}^{n \times n}$, called the Hessian mapping: $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n \times n}$.

For a differentiable function $F : \mathbb{R}^n \to \mathbb{R}^m$ that takes the input of n independent variables and returns the values of m dependent variables, we write $F = (F_1, F_2, \cdots, F_m)$ with each $F_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \cdots, m$, being a real-valued function. The gradients of all the component functions F_i form an $m \times n$ matrix as

$$F'(x) = \begin{bmatrix} \nabla F_1(x)^T \\ \nabla F_2(x)^T \\ \vdots \\ \nabla F_1(x)^T \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1}(x) & \frac{\partial F_1}{\partial x_2}(x) & \cdots & \frac{\partial F_1}{\partial x_n}(x) \\ \frac{\partial F_2}{\partial x_1}(x) & \frac{\partial F_2}{\partial x_2}(x) & \cdots & \frac{\partial F_2}{\partial x_n}(x) \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial F_m}{\partial x_1}(x) & \frac{\partial F_m}{\partial x_2}(x) & \cdots & \frac{\partial F_m}{\partial x_n}(x) \end{bmatrix}.$$

This matrix is called the Jacobian matrix of F at x, and $F'(\cdot) : \mathbb{R}^n \to \mathbb{R}^{m \times n}$ is called the Jacobian mapping.

Example 8.1. Consider a function $F : \mathbb{R}^3 \to \mathbb{R}^2$ defined as $F(x) = (2x_1^2 + x_1x_2 + x_3^2,\ x_1x_2 + x_2x_3 + x_3x_1)$. Here, $F_1(x) = 2x_1^2 + x_1x_2 + x_3^2$ and $F_2(x) = x_1x_2 + x_2x_3 + x_3x_1)$. Then, the Jacobian mapping $F' : \mathbb{R}^3 \to \mathbb{R}^{2\times 3}$ is

$$F'(x) = \begin{bmatrix} 4x_1 + x_2 & x_1 & 2x_3 \\ x_2 + x_3 & x_1 + x_3 & x_2 + x_1 \end{bmatrix}.$$

The second derivatives of such vector functions are relatively complicated, and we omit them in this course.

8.2.2 Symmetric and positive semidefinite matrices

Given a square matrix Q in $\mathbb{R}^{n\times n}$, we say that Q is symmetric if $Q = Q^T$. For example, the identity matrix or any diagonal matrix is symmetric. The matrix $Q = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & -2 \\ 3 & -2 & 5 \end{bmatrix}$ is symmetric.

We say that Q is positive semidefinite if $x^TQx \geq 0$ for any vector $x \in \mathbb{R}^n$. For example, $Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $Q = \begin{bmatrix} 4 & 2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 4 \end{bmatrix}$ are both symmetric and positive semidefinite. To verify the positive semidefiniteness, note that

$$x^TQx = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1^2 \geq 0.$$

Similarly, we have

$$\begin{aligned} x^TQx = & \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^T \begin{bmatrix} 4 & 2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ =& 4x_1^2 + 2x_2^2 + 4x_3^2 + 4x_1x_2 + 2x_2x_3 = (2x_1 + x_2)^2 + (x_2 + x_3)^2 + 3x_3^2 \geq 0, \end{aligned}$$

which shows that Q is positive semidefinite. For another example, consider the matrix $Q = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$. For any $x \in \mathbb{R}^2$, we have $x^TQx = x_1^2 + 4x_1x_2 + 3x_2^2$. If we let $x_2 = -\frac{x_1}{2} \neq 0$, then $x^TQx = x_1^2 - 2x_1^2 + \frac{3}{4}x_1^2 = -\frac{x_1^2}{4} < 0$. Hence, this matrix Q is not positive semidefinite. Similarly, $Q = \begin{bmatrix} 4 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix}$ is not positive semidefinite. Indeed, for any $x \in \mathbb{R}^3$ we have $x^T Qx = 4x_1^2 + 2x_2^2 + x_3^2 + 4x_1x_2 + 6x_1x_3 + 2x_2x_3$. If we let $x_2 = -x_1$ and $x_3 = -x_1$ for any $x_1 \neq 0$, then we have $x^T Qx = 4x_1^2 + 2x_1^2 + x_1^2 - 4x_1^2 - 6x_1^2 + 2x_1^2 = -x_1^2 < 0$. Hence, Q is not positive semidefinite.

We say that Q is positive definite if $x^T Qx > 0$ for any nonzero vector $x \in \mathbb{R}^n$. For example, $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ is symmetric and positive definite, because $x^T Qx = x_1^2 + x_2^2 > 0$ whenever $x \neq 0$. Similarly, $Q = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ is also symmetric and positive definite because $x^T Qx = 4x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_2x_3 = (x_1 + x_2)^2 + (x_2 + x_3)^2 + 3x_1^2 > 0$ for any $x \neq 0$.

Recall that any symmetric matrix $Q \in \mathbb{R}^{n\times n}$ can be decomposed as $Q = V\Lambda V^{-1}$, where Λ is a diagonal matrix whose diagonal elements are exactly the n eigenvalues λ_i of Q, and V is an $n \times n$ orthogonal matrix that satisfies $VV^T = I$. Then, Q is positive semidefinite if and only if $\min_i\{\lambda_i\} \geq 0$, and is positive definite if and only if $\min_i\{\lambda_i\} > 0$. For example, $Q = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$ has an eigenvalue decomposition as

$$Q = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

Here, $\lambda_1 = 4$ and $\lambda_2 = 2$ are both real and positive. Hence, Q is positive definite.

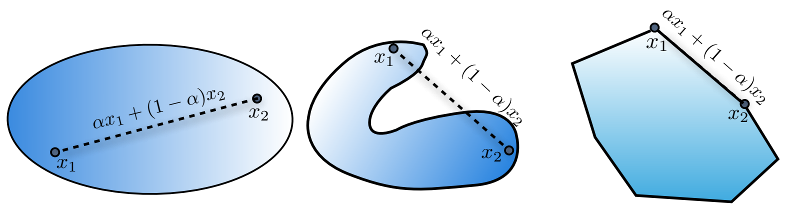
8.2.3 Convex sets, convex functions and convex optimization

We recall the definition of convex sets and convex functions in this subsection.

8.2.3.1 Convex sets

Definition 8.1. A set $\mathscr{C}$ in $\mathbb{R}^n$ is said to be convex if $(1-\alpha)x_1 + \alpha x_2 \in \mathscr{C}$ for any points $x_1, x_2 \in \mathscr{C}$ and any $\alpha \in [0,1]$.

In other words, $\mathscr{C}$ is convex if the line segment that connects x_1 and x_2 for any two points $x_1, x_2 \in \mathscr{C}$ is entirely contained in $\mathscr{C}$. Geometrically, the figure below shows three sets, where the left and the right are convex, while the middle one is nonconvex.


$\alpha x_1 + (1-\alpha)x_2$
x_1
x_2
$\alpha x_1 + (1-\alpha)x_2$
x_1
x_2
$\alpha x_1 + (1-\alpha)x_2$
x_1
x_2

Definition 8.2. A set $\mathscr{P}$ in $\mathbb{R}^n$ is said to be polyhedral (or a polyhedron) if it is formed as the intersection of finitely many half-spaces, that is

$$\mathscr{P} = \left\{x \in \mathbb{R}^n \mid a_i^T x \le b_i,\ i = 1, \cdots, m\right\}. \tag{8.3}$$

Here are some examples of convex sets.

- Any polyhedral set $\mathscr{P} = \{x \in \mathbb{R}^n \mid Ax \le b\}$ is convex. Indeed, if $x, y \in \mathscr{P}$, we have $Ax \le b$ and $Ay \le b$. Now, for any $\alpha \in [0,1]$, we have $A((1-\alpha)x + \alpha y) = (1-\alpha)Ax + \alpha Ay \le (1-\alpha)b + \alpha b = b$. Hence, $(1-\alpha)x + \alpha y \in \mathscr{P}$. We have verified that $\mathscr{P}$ is convex. Note that half-spaces and hyperplanes are special polyhedra, and therefore are convex. (In some fields such as geometry and some other books, polyhedra are defined as the union of finitely many sets of the form (8.3) and are allowed to be nonconvex. For example, stars with straight edges are special polyhedrons according to that definition. Our definition here follows standard books in convex analysis and convex optimization, such as [5, 6].)
- The whole space $\mathbb{R}^n$ and the empty set are both convex.
- The ball $\mathscr{B} = \{x \in \mathbb{R}^n \mid \sum_{j=1}^n (x_i - a_i)^2 \le r^2\}$ is convex. Indeed, we can write this ball using Euclidean norm as $\mathscr{B} = \{x \in \mathbb{R}^n \mid \|x - a\| \le r\}$. For any $x, y \in \mathscr{B}$, we have $\|x - a\| \le r$ and $\|y - a\| \le r$. Choose any $\alpha \in [0,1]$, and consider the point $z = (1-\alpha)x + \alpha y$. We have $\|z - a\| = \|(1-\alpha)x + \alpha y - a\| = \|(1-\alpha)(x-a) + \alpha(y-a)\| \le \|(1-\alpha)(x-a)\| + \|\alpha(y-a)\| = (1-\alpha)\|x - a\| + \alpha\|y - a\| \le (1-\alpha)r + \alpha r = r$. Here, the first inequality follows from the triangle inequality. We conclude that $z \in \mathscr{B}$, and have thereby verified that $\mathscr{B}$ is convex.

8.2.3.2 Convex functions

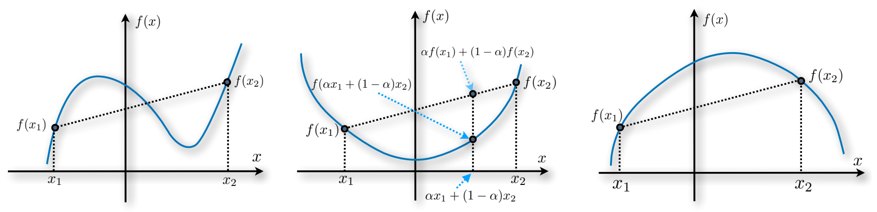
Definition 8.3. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if

$$f((1-\alpha)x_1 + \alpha x_2) \le (1-\alpha)f(x_1) + \alpha f(x_2)$$

for any $x_1, x_2 \in \mathbb{R}^n$ and any $\alpha \in [0,1]$.

In other words, f is convex if the secant line that connects any two points on the graph of f lies above the graph of f between those two points. If a convex function f is differentiable at x, then the tangent line to the graph of f at the point $(x, f(x))$ lies below the graph of f.

A function f is said to be concave if $-f$ is convex. The figure below illustrates several functions, where the left function is nonconvex, the middle one is convex, and the right one is concave.



Below are some examples of convex functions.

1. Constant functions $f(x) = c$ and linear functions $f(x) = a^T x$ are convex.
2. Consider a quadratic function $f(x) = \frac{1}{2}x^T Qx + q^T x$, where Q is an $n \times n$ symmetric matrix, and $q \in \mathbb{R}^n$. This function is convex if and only if Q is positive semidefinite.
3. The norm function $f(x) = \|x\|$ is convex. Indeed, for any points $x, y \in \mathbb{R}^n$, and any $\alpha \in [0,1]$, we consider $z = (1-\alpha)x + \alpha y$. By the triangle inequality, we have $\|z\| = \|(1-\alpha)x + \alpha y\| \le \|(1-\alpha)x\| + \|\alpha y\| = (1-\alpha)\|x\| + \alpha\|y\|$. Hence, $f((1-\alpha)x + \alpha y) \le (1-\alpha)f(x) + \alpha f(y)$, which shows that f is convex.

The following basic properties can be used to check if a given function is convex or not. See, e.g., [6, 19] for proofs of these properties.

- If f and g are convex, and $\beta, \gamma \ge 0$, then $\beta f + \gamma g$ is convex.
- If $f : \mathbb{R} \to \mathbb{R}$ is differentiable, then f is convex if and only if $f'(x_0) \le f'(x_1)$ when $x_0 < x_1$.

- If $f : \mathbb{R} \to \mathbb{R}$ is differentiable, then f is convex if and only if $f(y) \geq f(x) + f'(x)(y-x)$ for all x and y in $\mathbb{R}$.
- If $f : \mathbb{R} \to \mathbb{R}$ is twice differentiable, then f is convex if and only if $f''(x) \geq 0$ for all x in $\mathbb{R}$.
- If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, then f is convex if and only if $\langle x_1 - x_0,\ \nabla f(x_1) - \nabla f(x_0)\rangle \geq 0$ for all x_0 and x_1 in $\mathbb{R}^n$.
- If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, then f is convex if and only if $f(y) \geq f(x) + \langle \nabla f(x),\ y-x\rangle$ for all x and y in $\mathbb{R}^n$.
- If $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, then f is convex if and only if the Hessian $\nabla^2 f(x)$ is positive semidefinite for all x in $\mathbb{R}^n$.

A property for functions from $\mathbb{R}^n$ to $\mathbb{R}$ that is stronger than convexity is strict convexity.

Definition 8.4. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be strictly convex if

$$f((1-\alpha)x_1 + \alpha x_2) < (1-\alpha)f(x_1) + \alpha f(x_2)$$

for any $x_1, x_2 \in \mathbb{R}^n$ with $x_1 \neq x_2$ and any $\alpha \in (0,1)$.

Graphically, f is strictly convex if the secant line that connects any two points on the graph of f lies strictly above the graph of f between those two points. There are analogous properties as above that can be used to check if a given function is strictly convex. For simplicity, we list one of them:

- If $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, then f is strictly convex if the Hessian $\nabla^2 f(x)$ is positive definite for all x in $\mathbb{R}^n$.

8.2.3.3 Basic properties of convex optimization

For a general optimization problem of the form

$$\min_x f(x) \ \text{ s.t. } x \in \mathscr{X} \subset \mathbb{R}^n, \tag{8.4}$$

we say that $x^* \in \mathscr{X}$ is a local optimal solution if there exists a neighborhood $\mathscr{V}$ of x^* in $\mathbb{R}^n$ such that

$$f(x^*) \leq f(x), \ \ \forall x \in \mathscr{V} \cap \mathscr{X}.$$

We say that $x^* \in \mathscr{X}$ is a global optimal solution of this problem if

$$f(x^*) \leq f(x), \ \ \forall x \in \mathscr{X}.$$

A convex optimization problem is an optimization problem that minimizes a convex objective function over a convex feasible set, that is, a problem of the form (8.4) in which $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $\mathscr{X}$ is a nonempty, closed and convex set in $\mathbb{R}^n$. The following theorem is a fundamental result for convex optimization. It says that any local optimal solution is also a global optimal solution in convex optimization. For this reason, for convex optimization problems we can use the term "optimal solutions" directly, as with linear programming.

Theorem 8.1. *For a convex optimization problem of the form* (8.4)*, every local optimal solution (if one exists) is a global optimal solution. The set of all global solutions to* (8.4) *is convex. Furthermore, if f is strictly convex, there can be at most one global solution.*

Proof. Suppose that x_0 and x_1 are both global solutions. Let $\lambda \in (0,1)$. We have

$$f(\lambda x_0 + (1-\lambda)x_1) \leq \lambda f(x_0) + (1-\lambda)f(x_1) = f(x_0)$$

because $f(x_0) = f(x_1)$. This proves that the set of global solutions is convex. Moreover, if f is strictly convex, then x_0 and x_1 cannot be different, so that there can be at most one global solution.

Suppose now that x_0 is a local solution. Let $x_1 \in \mathscr{X}$. We need to show that $f(x_1) \geq f(x_0)$. The definition of local solution implies the existence of a neighborhood $\mathscr{V}$ of x_0 such that $f(x_0) \leq f(x)$ for all $x \in \mathscr{V} \cap \mathscr{X}$. We can choose a small positive real number $\lambda \in (0,1)$ such that $x_\lambda = x_0 + \lambda(x_1 - x_0) \in \mathscr{V}$. We have $f(x_\lambda) \geq f(x_0)$, which implies

$$(1-\lambda)f(x_0) + \lambda f(x_1) \geq f(x_\lambda) \geq f(x_0)$$

and therefore $f(x_0) \leq f(x_1)$. $\square$

We know from calculus that if x^* is a local minimum or local maximum of a function $f : \mathbb{R} \to \mathbb{R}$, then the derivative $f'(x^*) = 0$. For example, $x^* = 2$ is a global minimum of $f(x) = x^2 - 4x$, and we can easily verify that $f'(x^*) = 2x^* - 4 = 0$. When the function f is also convex, then the condition $f'(x^*) = 0$ is sufficient for x^* to be a global optimal solution. This result can be generalized to functions $f : \mathbb{R}^n \to \mathbb{R}$ as follows.

Theorem 8.2. *Consider the unconstrained problem defined by a function $f : \mathbb{R}^n \to \mathbb{R}$:*

$$\min_{x \in \mathbb{R}^n} f(x). \tag{8.5}$$

If x^* is a local solution and f is differentiable at x^*, then

$$\nabla f(x^*) = 0. \tag{8.6}$$

Furthermore, if f is convex, and is differentiable at a point x^* and satisfies (8.6)*, then x^* is a global solution.*

Proof. To prove the first statement, suppose for the purpose of contradiction that $\nabla f(x^*) \neq 0$. Define the vector $p = -\nabla f(x^*)$ and note that $\langle p, \nabla f(x^*) \rangle = -\|\nabla f(x^*)\|^2 < 0$. Consider the function $g(t) = f(x^* + tp)$, which is a function from $\mathbb{R}$ to $\mathbb{R}$. Here $g'(0) = \langle p,\ \nabla f(x^*) \rangle < 0$. By the definition of derivative,

$$\lim_{t \to 0} \frac{g(t) - g(0)}{t} = g'(0) < 0,$$

so $\frac{g(t)-g(0)}{t} < 0$ for t sufficiently small. Therefore, $g(t) - g(0) < 0$ for $t > 0$ sufficiently small. This contradicts with the fact that x^* is a local solution. This shows that (8.6) must hold at the local solution x^*.

To prove the second statement, note that the convexity of f and its differentiable at x^* implies

$$f(x) \geq f(x^*) + \langle \nabla f(x^*),\ x - x^* \rangle \text{ for all } x \in \mathbb{R}^n.$$

With $\nabla f(x^*) = 0$, this implies $f(x) \geq f(x^*)$ for all $x \in \mathbb{R}^n$. Hence, x^* is a global solution. □

The condition (8.6) in the above theorem is referred to as the first order optimality condition for unconstrained optimization problems. The above theorem is also called Fermat's rule.

Example 8.2. Let us consider the following convex optimization problem:

$$\min_{x \in \mathbb{R}^2} f(x) = x_1^2 - 4x_1x_2 + 6x_2^2 - 2x_1.$$

Since f is convex and differentiable, using Theorem 8.2, we can show that $x^* \in \mathbb{R}^2$ is an optimal solution of this problem iff $\nabla f(x^*) = 0$. That is,

$$\nabla f(x^*) = \begin{pmatrix} 2x_1^* - 4x_2^* - 2 \\ -4x_1^* + 12x_2^* \end{pmatrix} = 0.$$

This is a linear system of x_1^* and x_2^*. Solve this system we obtain $(x_1^*, x_2^*)^T = (3,1)^T$, which is an optimal solution of the above problem with the optimal value $f(x^*) = -3$.

The optimality condition for a constrained convex optimization problem (8.4) is much more complicated, see below for one such condition.

Theorem 8.3. *Consider a constrained optimization problem* (8.4) *in which* $f : \mathbb{R}^n \to \mathbb{R}$ *is a convex, differentiable function and* $\mathscr{X}$ *is a nonempty, closed and convex set in* $\mathbb{R}^n$. *Then,* $x^* \in \mathscr{X}$ *is an optimal solution of* (8.4) *if and only if*

$$\nabla f(x^*)^T (y - x^*) \geq 0, \ \ \forall y \in \mathscr{X}. \tag{8.7}$$

Proof. First, suppose (8.7) holds for $x^* \in \mathscr{X}$. Then, for any $y \in \mathscr{X}$,

$$f(y) \geq f(x) + \nabla f(x^*)^T (y - x^*) \geq f(x),$$

so x^* is an optimal solution to (8.4).

Conversely, suppose that $x^* \in \mathscr{X}$ is an optimal solution of (8.4), and suppose for the purpose of contradiction that (8.7) does not hold. This means for some $y \in \mathscr{X}$ we have

$$\nabla f(x^*)^T (y - x^*) < 0.$$

Define the vector $p = y - x^*$ and note that $\langle p, \nabla f(x^*) \rangle = \nabla f(x^*)^T (y - x^*) < 0$. Consider the function $g(t) = f(x^* + tp)$, which is a function from $\mathbb{R}$ to $\mathbb{R}$. Here $g'(0) = \langle p, \ \nabla f(x^*) \rangle < 0$. By the definition of derivative,

$$\lim_{t \to 0} \frac{g(t) - g(0)}{t} = g'(0) < 0,$$

so $\frac{g(t)-g(0)}{t} < 0$ for t sufficiently small. Therefore, $g(t) - g(0) < 0$ for $t > 0$ sufficiently small. In other words, for $t > 0$ sufficiently small, we have

$$g(t) = f(x^* + t(y - x^*)) < f(x^*) = g(0).$$

Since x^* and y both belong to the convex set $\mathscr{X}$, the point $x^* + t(y - x^*) = (1 - t)x^* + ty$ also belongs to $\mathscr{X}$. The above inequality therefore contradicts with the fact that x^* is an optimal solution of (8.4). This shows that (8.6) must hold at x^*. □

The condition (8.7) is no longer an equation, and is referred as a variational inequality, as it consists of a family of inequalities, one for each $y \in \mathscr{X}$. By further exploring the structure of the set $\mathscr{X}$, it is possible to rewrite (8.7) in other forms. We will return to optimality conditions in subsequent sections.

8.3 Convex quadratic programming

Mathematically, a quadratic program is a problem that minimizes (or maximizes) a quadratic function $\frac{1}{2}x^TQx + q^Tx$ over a polyhedron $\mathscr{P}$:

$$\begin{cases} \min \ \frac{1}{2}x^TQx + q^Tx \\ \text{s.t.} \ \ x \in \mathscr{P}. \end{cases} \tag{8.8}$$

Quadratic programming is a natural extension of linear programing: when $Q = 0$, the above problem reduces to a linear program. By the same technique that converts general linear programs into standard form, we can convert any quadratic program into the the following standard form:

$$\begin{cases} \min\limits_{x} \ \frac{1}{2}x^TQx + q^Tx \\ \text{s.t.} \ \ Ax = b, \\ \qquad x \geq 0, \end{cases} \tag{8.9}$$

where Q is an $n \times n$ symmetric matrix, A is an $m \times n$ matrix, $b \in \mathbb{R}^m$ and $q \in \mathbb{R}^n$ are two given vectors.

The objective function $f(x) = \frac{1}{2}x^TQx + q^Tx$ can be convex or nonconvex. Because $\nabla^2 f(x) = Q$ for any x, f is convex (strictly convex) if and only if Q is positive semidefinite (positive definite). In this course, we only consider convex quadratic programs, so Q is assumed to be symmetric and positive semidefinite. (If Q in a given problem is non-symmetric, we can first symmetrize it using the fact that $x^TQx = \frac{1}{2}(x^T(Q+Q^T)x.)$

As an example, consider the following least-squares problem:

$$\min_{x\in\mathbb{R}^n} \left\{ \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}\sum_{i=1}^m \Big(\sum_{j=1}^n A_{ij}x_j - b_i \Big)^2 \right\}$$

By defining $Q = A^T A$ (which is symmetric and positive semidefinite) and $q = -A^T b$, we can rewrite this problem as

$$\min_{x} \frac{1}{2} x^T Q x + q^T x.$$

This problem is a convex quadratic program without constraints.

Properties of convex quadratic programming:

1. Any (local) optimal solution x^* of the convex quadratic problem (8.9) is also a global optimal solution, i.e., it satisfies $f(x^*) \leq f(x)$ for all feasible solutions x.
2. The set of all optimal solutions to (8.9) is convex.
3. If Q is symmetric and positive definite and the problem is feasible, then it must have a unique solution.

The first two properties are straightforward applications of Theorem 8.1. Solution uniqueness in the last property also follows from Theorem 8.1, because the objective function is strictly convex when Q is positive definite. Solution existence in the last property is harder to prove and is a consequence of the Frank-Wolfe Theorem for quadratic programming, see, e.g., [2].

Below, we first consider two special quadratic programs, unconstrained QPs and equality constrained QPs, and then discuss an algorithm to solve the standard QP (8.9).

8.3.1 Unconstrained quadratic programs

Consider the following unconstrained convex QP:

$$\min_{x \in \mathbb{R}^n} f(x) = \tfrac{1}{2} x^T Q x + q^T x,$$

where $Q \in \mathbb{R}^{n \times n}$ is symmetric and positive semidefinite, and $q \in \mathbb{R}^n$.

By Theorem 8.2, a point $x^* \in \mathbb{R}^n$ is an optimal solution to the above problem if and only if it satisfies

$$\nabla f(x^*) = Q x^* + q = 0.$$

If Q is invertible, then we can compute the solution x^* as $x^* = -Q^{-1} q$. Since Q is symmetric and positive definite, we can solve $Q x^* = -q$ by using Cholesky decomposition of Q, or using a conjugate gradient method [11]. If Q is positive semidefinite but not positive definite, then other methods such as QR factorization should be conducted to solve $Qx^* + q = 0$. For details on different methods on linear equations, see, e.g., [22].

Example 8.3. Consider the unconstrained convex QP problem

$$\min_{x\in\mathbb{R}^3}\left\{ f(x) = \frac{1}{2}(x_1^2 + 4x_2^2 + 3x_3^2 - 2x_1x_2 + 2x_2x_3) + 3x_1 + 2x_2 - x_3 \right\}.$$

In this case, we have $Q = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 4 & 1 \\ 0 & 1 & 3 \end{bmatrix}$ and $q = (3,2,-1)^T$. Hence, the solution of this problem can be computed by solving the following linear system:

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 4 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{pmatrix} x_1^* \\ x_2^* \\ x_3^* \end{pmatrix} = - \begin{pmatrix} 3 \\ 2 \\ -1 \end{pmatrix}.$$

Solving this system, we obtain $x^* = (-4,-2,1)^T$ as the unique optimal solution of the QP problem, with the optimal value being $f(x^*) = 22.5$.

8.3.2 Quadratic programs with equality constraints

Consider the following QP with equality constraints:

$$\min_x f(x) = \tfrac{1}{2}x^TQx + q^Tx \quad \text{subject to } Ax = b, \tag{8.10}$$

where Q is an $n\times n$ symmetric and positive semidefinite matrix, A is an $m\times n$ matrix, $b\in\mathbb{R}^m$ and $q\in\mathbb{R}^n$ are two given vectors. Assume that $m\le n$ and that the m rows of A are linearly independent.

The feasible set of (8.10) can be written as $\mathscr{X} = \{x\in\mathbb{R}^n \mid Ax = b\}$, which is an *affine set*. It has the following property: if y and x^* both belong to $\mathscr{X}$, then $2x^* - y$ also belongs to $\mathscr{X}$. Using the latter property, we can rewrite the optimality condition (8.7) as

$$\nabla f(x^*)^T(y - x^*) = 0, \ \ \forall y\in\mathscr{X} \tag{8.11}$$

for the problem (8.10). The vectors of the form $y - x^*$ for $y\in\mathscr{X}$ are precisely elements of the subspace $\{x\in\mathbb{R}^n \mid Ax = 0\}$, and the above condition says that $\nabla f(x^*)$ is orthogonal to all elements of the latter subspace. By the decomposition theorem in linear algebra on null and range spaces, this means that $\nabla f(x^*)$ belongs to the range space of A^T. We have thereby proved the following theorem:

Theorem 8.4. *A point x^* is an optimal solution to* (8.10) *and only if there exists a vector $y^* \in \mathbb{R}^m$ such that*

$$\begin{cases} Qx^* + q + A^T y^* = 0 \\ Ax^* - b \qquad\quad = 0, \end{cases}$$

namely,

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}.$$

The vector y^* here is called the vector of Lagrange multipliers.

If Q is positive definite (so that Q^{-1} exists), and rows of A are linearly independent, then we can solve the above linear system explicitly as follows. From $Qx^* + A^T y^* + q = 0$ we have $x^* = -Q^{-1}(A^T y^* + q)$. Substituting this into the second equation, we have $-AQ^{-1}A^T y^* - AQ^{-1}q = b$, which implies $AQ^{-1}A^T y^* = -AQ^{-1}q - b$. Hence, we obtain $y^* = -(AQ^{-1}A^T)^{-1}(AQ^{-1}q + b)$. (Exercise: show that $AQ^{-1}A^T$ is invertible.) Plug this expression into $x^* = -Q^{-1}(A^T y^* + q)$, we obtain

$$x^* = Q^{-1}A^T(AQ^{-1}A^T)^{-1}(AQ^{-1}q + b) - Q^{-1}q.$$

In summary, the solution of (8.10) can explicitly be written as

$$x^* = Q^{-1}A^T(AQ^{-1}A^T)^{-1}(AQ^{-1}q + b) - Q^{-1}q,$$

with the vector of Lagrange multipliers being

$$y^* = -(AQ^{-1}A^T)^{-1}(AQ^{-1}q + b).$$

Computing the above solution requires finding the inverse Q^{-1} of Q, and $(AQ^{-1}A^T)^{-1}$, and some matrix-vector multiplications. In practice, other efficient methods are often used to solve the linear system directly.

Example 8.4. Consider the following quadratic program with one equality constraint:

$$\min_{x_1, x_2} \left\{ f(x) = \frac{1}{2}(x_1^2 + 2x_2^2 - 2x_1x_2) + 2x_1 - 3x_2 \right\} \quad \text{s.t. } x_1 + x_2 = 4.$$

With $Q = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$, $q = (2, -3)^T$, $A = [1, 1]$ and $b = 4$, we can write the optimality condition as

$$\begin{cases} x_1^* - x_2^* + 2 + y^* & = 0, \\ -x_1^* + 2x_2^* - 3 + y^* & = 0, \\ x_1^* + x_2^* & = 4, \end{cases}$$

where y is the Lagrange multiplier. Solving this system, we obtain the optimal solution, $x^* = (1.4, 2.6)^T$ and the multiplier $y^* = -0.8$.

8.3.3 Solution methods

There are two common methods to solve standard convex quadratic programs (8.9).

- *Active set methods:* These methods can be considered as extensions of simplex methods for linear programs.
- *Interior-point methods:* These methods are iterative methods based on logarithmic barriers.

Both methods are widely used, and have been implemented in several software packages including `quadprog` in Matlab. Below, we introduce a standard primal-dual interior-point method to solve (8.9).

8.3.3.1 A primal-dual interior-point method for convex QPs

We first define a barrier problem associated with (8.9) as follows:

$$\min_x \left\{ \mathscr{B}_\mu(x) = \frac{1}{2} x^T Q x + q^T x - \mu \sum_{j=1}^n \log(x_j) \mid Ax = b \right\}, \tag{8.12}$$

where $\mu > 0$ is a penalty parameter. The solution of this problem $x^*(\mu)$ generates a set $\{x^*(\mu) \mid \mu > 0\}$ called the central path of the barrier method. We can write the optimality condition of this barrier problem as follows:

$$\begin{cases} Qx + q + A^T y - \mu \mathrm{diag}(X^{-1}) & = 0 \\ Ax - b & = 0. \end{cases}$$

Here, $X = \mathrm{diag}(x)$ is the diagonal matrix with x_i being the ith diagonal element. Let us define $S := \mu X^{-1}$ as the slack variable. Then, $s = \mathrm{diag}(S)$ and $x \otimes s = \mu \mathbf{e}$, where $\mathbf{e} = (1, 1, \cdots, 1)^T$ is the vector of all ones, and $\otimes$ is component-wise product between two vectors. We write this optimality condition as

$$\begin{cases} Qx + A^T y - s = -q \\ Ax \qquad\qquad\quad = b \\ x \otimes s \qquad\quad\;\; = \mu \mathbf{e}. \end{cases}$$

Clearly, this is a nonlinear system of equations due to the equation $x \otimes s = XS\mathbf{e} = \mu\mathbf{e}$.

In the infeasible primal-dual interior-point methods, we linearize the nonlinear system $x \otimes s = \mu\mathbf{e}$ to obtain $X\Delta s + S\Delta x = \mu\mathbf{e} - XS\mathbf{e}$. Hence, the linear system for computing a Newton direction is defined as

$$\begin{bmatrix} Q & A^T & -\mathbb{I} \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} r^x \\ r^y \\ r^s \end{pmatrix} \quad \text{where} \quad \begin{cases} r^x & := -Qx - q - A^T y \\ r^y & := b - Ax \\ r^s & := \mu\mathbf{e} - XS\mathbf{e}. \end{cases}$$

Then, the next iteration is updated as

$$x_+ := x + \alpha_p \Delta x, \;\; y_+ := y + \alpha_p \Delta y, \;\; s_+ := s + \alpha_d \Delta s,$$

where $\alpha_p \in (0,1]$ and $\alpha_d \in (0,1]$ are given step-sizes chosen such that $x_+ > 0$ and $s_+ > 0$. Very often, the parameter μ is replaced by $\sigma\mu$ for a given $\sigma \in (0,1]$ and $\mu = \frac{s^T x}{n}$ measuring the duality gap.

Now, we can describe a primal-dual interior point method as follows: Algorithm 2 is called an infeasible primal-dual interior-point method since (x_k, y_k, s_k) may not be feasible to the primal and dual problems. If we start from a feasible point (x_0, y_0, s_0) such that $Qx_0 + A^T y_0 - s_0 = -q$, $Ax_0 - b = 0$, $x_0 > 0$ and $s_0 > 0$, then we can maintain $r_k^x = 0$ and $r_k^y = 0$ at each iteration. In this case, we obtain a feasible primal-dual IP method. We note that practical primal-dual IP methods are often more sophisticated than Algorithm 2 in order to accelerate their performance. A well-known variant is perhaps the Mehrotra predictor-corrector IP method [25]. More details about interior-point methods can be found in [16, 18, 25].

8.3.3.2 Implementation details

The main step of Algorithm 2 is the solution of the linear system at Step 7. This system is often solved by a substitution method combining with a Cholesky decomposition. Let us describe this method as follows: From the last equation $X_k \Delta s + S_k \Delta_x = r_k^s$, we compute Δs as $\Delta s = X_k^{-1}(r_k^s - S_k \Delta x)$. Substituting Δs into

Algorithm 2 (*Primal-dual interior-point algorithm for QPs*)

```
1: Input: A tolerance $\varepsilon > 0$ and $\sigma \in (0,1]$.
2: Initialization: Find an initial point $(x_0, y_0, s_0) \in \mathbb{R}^n_{++} \times \mathbb{R}^m \times \mathbb{R}^n_{++}$.
3: Main loop:
4:    Compute the duality gap $\mu_k := \frac{x_k^T s_k}{n}$.
5:    If $\mu_k \le \varepsilon$, then terminate, and $(x_k, y_k, s_k)$ is an approximate solution.
6:    Compute the residual $r_k^x := Qx_k - q - A^T y_k$, $r_k^y := b - Ax_k$ and $r_k^s := \sigma\mu_k \mathbf{e} - X_k S_k \mathbf{e}$.
7:    Solve the linear system
      $$\begin{bmatrix} Q & A^T & -\mathbb{I} \\ A & 0 & 0 \\ S_k & 0 & X_k \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} r_k^x \\ r_k^y \\ r_k^s \end{pmatrix}$$
      to obtain $\Delta x_k$, $\Delta y_k$ and $\Delta s_k$.
8:    Find step-sizes $\alpha_k^p, \alpha_k^d \in (0,1]$ from the conditions $x_k + \alpha_k^p \Delta x_k > 0$ and $s_k + \alpha_k^d \Delta s_k > 0$.
9:    Update
      $$x_{k+1} := x_k + \alpha_k^p \Delta x_k, \;\; y_{k+1} := y_k + \alpha_k^p \Delta y_k, \;\; s_{k+1} := s_k + \alpha_k^d \Delta s_k.$$
10: End
```

the first equation $Q\Delta x + A^T\Delta y - \Delta s = r_k^x$, we have

$$(Q + X_k^{-1}S_k)\Delta x + A^T\Delta y = r_k^x + X_k^{-1}r_k^s.$$

This implies that $\Delta x = (Q + X_k^{-1}S_k)^{-1}\left(r_k^x + X_k^{-1}r_k^s - A^T\Delta y\right)$. Substituting Δx into the second equation $A\Delta x = r_k^y$, we get $A(Q + X_k^{-1}S_k)^{-1}\left(r_k^x + X_k^{-1}r_k^s - A^T\Delta y\right) = r_k^y$, which leads to

$$A(Q + X_k^{-1}S_k)^{-1}A^T\Delta y = (Q + X_k^{-1}S_k)^{-1}\left(r_k^x + X_k^{-1}r_k^s\right) - r_k^y.$$

Let us define $H_k := A(Q + X_k^{-1}S_k)^{-1}A^T$ and $h_k := (Q + X_k^{-1}S_k)^{-1}\left(r_k^x + X_k^{-1}r_k^s\right) - r_k^y$. Since matrix H_k is symmetric positive definite, we can compute its Cholesky decomposition as $H_k := L_k L_k^T$, where L_k is a lower triangular matrix. Then the last linear system becomes

$$L_k L_k^T \Delta y = h_k.$$

This system can be solved by both back-substitution and forward-substitution methods. Since Q is symmetric positive semidefinite, we can compute its eigen-decomposition once in advance as $Q = U\Lambda U^T$, and apply a change of variable to convert the QP problem (8.9) into a form so that Q is a diagonal matrix. Without loss of generality, we can assume that Q is a diagonal matrix. Hence, computing $(Q + X_k^{-1}S_k)^{-1}$ now can be done efficiently, since this matrix is in fact a diagonal matrix.

To compute the stepsizes α_k^p and α_k^d, we can apply the following rules:

$$\begin{aligned} \alpha_k^p &:= 0.99 \min\left\{1, \min\left\{-\tfrac{(x_k)_i}{(\Delta x_k)_i} \mid (\Delta x_k)_i < 0\right\}\right\}, \\ \alpha_k^d &:= 0.99 \min\left\{1, \min\left\{-\tfrac{(s_k)_i}{(\Delta s_k)_i} \mid (\Delta s_k)_i < 0\right\}\right\}. \end{aligned}$$

Here, we clip these step-sizes by a factor 0.99 to make sure that $x_{k+1} > 0$ and $s_{k+1} > 0$.

8.3.4 Applications

We present three applications of convex quadratic programs in this section: the least-squares problem, portfolio selection, and the support vector machine.

8.3.4.1 Application 1: The least squares problem

Problem description: Consider a linear model

$$b = x_0 + x_1 a_1 + \cdots + x_n a_n + \varepsilon,$$

where $a = (1, a_1, \cdots, a_n)^T \in \mathbb{R}^{n+1}$ is an input vector, $b \in \mathbb{R}$ is the response, $x = (x_0, \cdots, x_n)^T \in \mathbb{R}^{n+1}$ is a parameter vector we need to estimate, and ε is an additive Gaussian white noise. Suppose that we have a collection of m data points $(a^{(i)}, b_i)$, with $a^{(i)} \in \mathbb{R}^{n+1}$ and $b_i \in \mathbb{R}$ for $i = 1, \cdots, m$, generated from the above linear model. The noise vector ε consists of components $\varepsilon_i = b_i - (x_0 + x_1 a_1^{(i)} + \cdots + x_n a_n^{(i)})$ for $i = 1, \cdots, m$. Our goal is to minimize the sum of squared errors $\sum_{i=1}^m \varepsilon_i^2$ over all possible parameter vectors $x \in \mathbb{R}^{n+1}$.

Problem formulation. By minimizing the sum of squares of components of the noise vector $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_m)^T$, we obtain the following least-squares problem:

$$\min_{x=(x_0,\cdots,x_n)^T} \frac{1}{2}\sum_{i=1}^{m}\left(b_i - x_0 - \sum_{j=1}^{n} a_j^{(i)} x_j\right)^2.$$

With

$$x = \begin{pmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{pmatrix}, \quad \text{and } A = \begin{bmatrix} 1 & a_1^{(1)} & \cdots & a_n^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & a_1^{(m)} & \cdots & a_n^{(m)} \end{bmatrix},$$

we can write the above problem into matrix form as

$$\min_x \frac{1}{2}\|Ax - b\|_2^2, \tag{8.13}$$

where $\|u\|_2^2 = \sum_{i=1}^m u_i^2$ is the square of the Euclidean norm of u.

The optimal solution. By Theorem 8.2, a vector $x^* \in \mathbb{R}^{n+1}$ is an optimal solution of (8.13) if and only if it solves the following equation (called the normal equation):

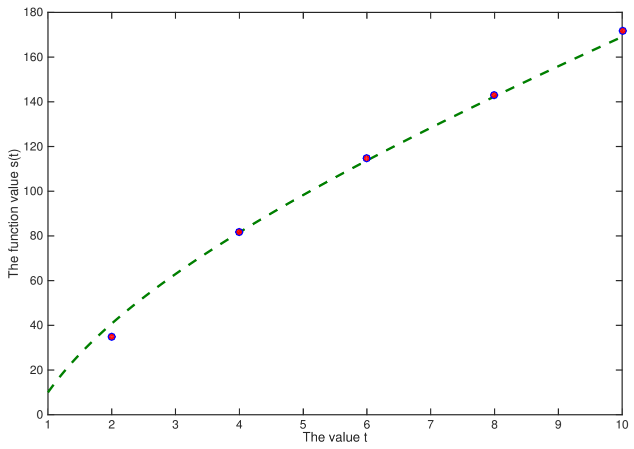
$$A^T A x = A^T b.$$

This is a special square linear system of $n+1$ variables, and the matrix $A^T A$ is symmetric and positive semidefinite (why?).

- If $A^T A$ is invertible, then the normal equation has a unique solution $x^* = (A^T A)^{-1} A^T b$. This is often the case when $m \geq n+1$, i.e., when there are more than n data points.
- If $A^T A$ is not invertible, then the normal equation has infinitely many solutions. In this case, we need to regularize the problem or to use the pseudo-inverse of A, see, e.g., [24].

An example. We consider a basic example as follows. Suppose that we are to estimate the two parameters x_1 and x_2 in a function $s(t) = x_1 t + x_2 \log(t)$ for $t \in [1, 10]$. By experiments, we obtain measured values $\hat{s}(t)$ for some different values of t, as shown in the following table. The values of $\hat{s}(t)$ can be different from the exact values of $s(t)$ due to measurement errors and other reasons. We can formulate

t	2	4	6	8	10
$\hat{s}(t)$	35	82	115	143	172



the problem of estimating the parameters x_1 and x_2 as a least-squares problem as follows.

Write the error between $\hat{s}(t)$ and $s(t)$ as $\varepsilon(t) = s(t) - \hat{s}(t)$. Using five data points given in the table, we minimize the sum of squared errors as $\sum_{i=1}^{5} \varepsilon(t_i)^2$. Since $s(t) = x_1 t + x_2 \log(t)$, we can write this problem as a least-squares problem:

$$\min_{x_1, x_2} \frac{1}{2} \sum_{i=1}^{5} (x_1 t_i + x_2 \log(t_i) - \hat{s}(t_i))^2.$$

If we define

$$A = \begin{bmatrix} 2 & \log(2) \\ 4 & \log(4) \\ 6 & \log(6) \\ 8 & \log(8) \\ 10 & \log(10) \end{bmatrix} = \begin{bmatrix} 2 & 0.6931 \\ 4 & 1.3863 \\ 6 & 1.7918 \\ 8 & 2.0794 \\ 10 & 2.3026 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 35 \\ 82 \\ 115 \\ 143 \\ 172 \end{bmatrix},$$

then we can write the above problem as $\min_{x \in \mathbb{R}^2} \frac{1}{2} \|Ax - b\|^2$. In this case, we can solve this problem to obtain the unique solution $x^* = (A^T A)^{-1} A^T b = (11.4489, 24.9939)^T$. We can then use the obtained function $s(t) = 11.4489t + 24.9939 \log(t)$ to evaluate the value of $s(t)$ at other points of t, such as $s(1) = 11.4489$, $s(3) = 61.8053$, and $s(5) = 97.4706$.

8.3.4.2 Application 2: Markowitz's portfolio selection

Problem description: Suppose that n stocks (or assets) are being considered by an investor for inclusion in a portfolio to be held through a period. Let the decision variables x_i $(i = 1, \cdots, n)$ be the amount (in dollars) of the investor's wealth invested in stock i at the beginning of the period. Hence, the number of shares of stock i held by the investor is given by x_i divided by the initial price of stock i.

Let p_i be the return rate of asset i over the period, which is the ratio of the price change over the period (the difference between the end price and the initial price) to the initial price. Because the end price is random, $p = (p_1, \cdots, p_n)$ is an n-dim random vector. Suppose that we can estimate its mean $\bar{p}$ and covariance matrix Σ. Then, with $x = (x_1, \cdots, x_n)$, the overall return of the portfolio is $r = p^T x$ (in dollars), which is a 1-dim random variable with mean $\bar{p}^T x$ and variance $x^T \Sigma x$. The classical Markowitz model tries to minimize the variance $x^T \Sigma x$, subject to a lower bound on the expect return, a budget constraint, as well as some bound constraints on x. This leads to the following formulation.

Problem formulation: We can formulate Markowitz's model as

$$\begin{cases} \min\limits_{x} \ x^T \Sigma x \\ \text{s.t.} \ \ \bar{p}^T x \geq r_{\min} \\ \qquad \sum_{i=1}^n x_i = B, \\ \qquad 0 \leq x_i \leq u_i, \ i = 1, \cdots, n, \end{cases}$$

where $r_{\min}$ is a lower bound on the expect return, B is the total budget (the amount of money to be invested at the beginning of the period), and u_i is an upper bound on x_i. This problem is a constrained convex QP. If we are not required to use the entire budget, we can replace the budget constraint by $\sum_{i=1}^n x_i \leq B$. The problem remains a constrained convex QP.

A numerical example: We consider three assets: AS1, AS2 and AS3. Suppose that the covariance matrix of the return p is

$$\Sigma = \begin{bmatrix} 3 & 1 & -0.5 \\ 1 & 2 & -0.4 \\ -0.5 & -0.4 & 1 \end{bmatrix},$$

the mean of p is $\bar{p} = (1.3, 1.2, 1.08)^T$, the budget is $B = 1$, and the upper bound is $u_i = 0.75, i = 1, 2, 3$, and the lower bound on the expected return is $r_{\min} = 1.2$. We can model this problem either in GAMS or in Matlab. For example, here is a small Matlab script with three lines.

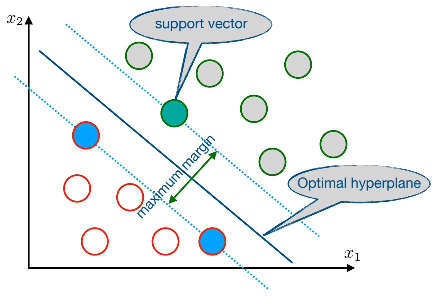
```
>> S = [3 1 -0.5; 1 2 -0.4; -0.5 -0.4 1];
>> A = -[1.3, 1.2, 1.08]; b = -1.2; C = [1, 1, 1]; d = 1;
>> [x, sig] = quadprog(S, zeros(3,1), A, b, C, d,
                              zeros(3,1), 0.75*ones(3,1));
```

Running the above code, we get $x = (0.4202, 0.2296, 0.3502)^T$, and the optimal variance is $\sigma = 0.3696$.

8.3.4.3 Application 3: Support vector machine

Problem description. We are given a set of data points $\left\{(\mathbf{x}^{(i)}, y_i)\right\}_{i=1}^m$, where $\mathbf{x}^{(i)} \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ for each $i = 1, \cdots, m$. The value of y_i is called the label of the point $\mathbf{x}^{(i)}$.

Suppose that we plot all m points $\left\{\mathbf{x}^{(i)}\right\}$ in the $\mathbb{R}^n$ space. The goal is to find a hyperplane $\mathscr{H} : \mathbf{w}^T\mathbf{x} + b = 0$ so that points associated with $y_i = +1$ belong to one halfspace and the other points (i.e., points with $y_i = -1$) belong to the other halfspace. This hyperplane is called the separating hyperplane, $\mathbf{w} \in \mathbb{R}^n$ is its normal vector (called the support vector), and $b \in \mathbb{R}$ is its intercept.



Problem formulation in the case of perfect classification. Suppose that there exists a hyperplane that provides perfect classification, namely, all the points associated with $y_i = +1$ belong to one halfspace and the other points (i.e., points with $y_i = -1$) belong to the other halfspace. Because points that lie exactly on the separating hyperplane cannot be classified correctly, we hope that the hyperplane can provide the maximum margin between points of opposite labels. To clarify this purpose, we use two hyperplanes $\mathbf{w}^T\mathbf{x} + b = 1$ and $\mathbf{w}^T\mathbf{x} + b = -1$ that are parallel to $\mathscr{H}$ to separate points of opposite labels, and try to maximize the distance between these two hyperplanes. In other words, we hope each $(\mathbf{x}^{(i)}, y_i), i = 1, \cdots, m$ satisfies the following condition:

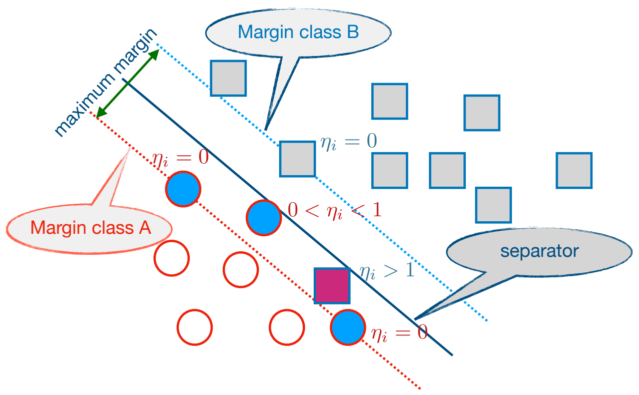
$$\begin{cases} \mathbf{w}^T\mathbf{x}^{(i)} + b \geq 1 & \text{if } y_i = 1, \\ \mathbf{w}^T\mathbf{x}^{(i)} + b \leq -1 & \text{if } y_i = -1 \end{cases} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1.$$

The distance between the two hyperplanes $\mathbf{w}^T\mathbf{x} + b = 1$ and $\mathbf{w}^T\mathbf{x} + b = -1$ is given by $\frac{2}{\|\mathbf{w}\|}$, which we aim to maximize. Equivalently, we can minimize $\|\mathbf{w}\|_2$, or minimize $\|\mathbf{w}\|_2^2$. Hence, we can formulate the support vector machine problem as below:

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}\|_2^2 \text{ subject to } y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \ \forall i = 1, \cdots, m. \tag{8.14}$$

This is a convex quadratic program.

Problem formulation in the case of imperfect classification. In reality, it is hard to expect that perfect classification exists for big datasets, so we have to accept misclassification of a few points. Accordingly, we relax (8.14) into the following



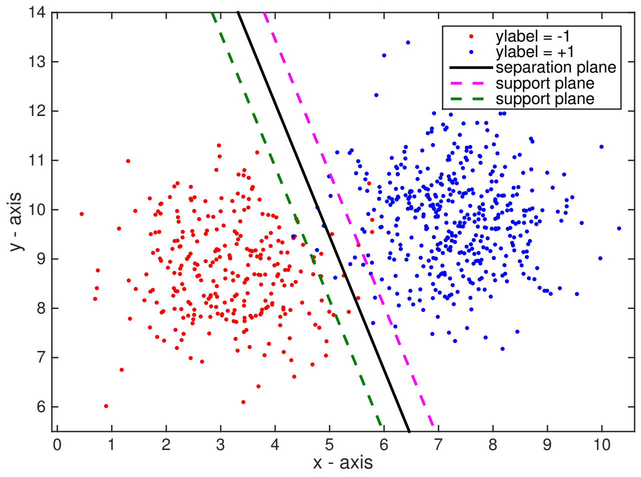
form, where $C > 0$ is a tuning parameter:

$$\min_{\mathbf{w}\in\mathbb{R}^n, \eta\in\mathbb{R}^m_+} \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^m \eta_i \ \middle| \ y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \eta_i, \ \forall i = 1, \cdots, m \right\}. \tag{8.15}$$

This problem is again a quadratic program with the joint variable $(\mathbf{w}, \eta) \in \mathbb{R}^{n+m}$.

Numerical example 1. We consider a dataset of (X, y) where $X \in \mathbb{R}^{711\times 2}$ represents 771 observed measurements and $y \in \{-1, 1\}^{711}$ provides the labels. By applying the SVM model (8.15) with $C = 1000$, we obtain the optimal separating hyperplane defined by $b = -17.7893$ and $w = (2.0934, 0.7745)^T$.

Numerical example 2. In the folder classification on the GAMS-Jupyter website, the file *traindata.csv* contains data extracted from the Wisconsin Diagnosis Breast



Cancer Dataset [13] publicly available at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic). We use 290 data points (samples) from the dataset. The "diagnosis" column in traindata.csv indicates whether the sample is malignant (+1) or benign (-1). The "Attr1" and "Attr2" columns in *traindata.csv* are taken from attributes 23 and 24 from the original dataset. The goal is to find an affine function of the two attributes to be a good predictor for the sample type. Here, $n = 2$, $m = 290$. The first data point is $(x^{(1)}, y_1) = (25.38, 17.33, 1)$, and so on. We choose $C = \frac{1}{m} = \frac{1}{290}$.

The GAMS codes in *CancerClassificationQP.ipynb* includes data in the csv file into a table in GAMS and solves the quadratic program using QCP. The variables $w1$, $w2$ and b defines a linear function $w1 \cdot attr1 + w2 \cdot attr2 + b$ which is used to classify maglinant (function value > 0) versus benign (function value < 0). The effect of the division line obtained from this method is shown in *testdata.xls*.

8.4 Convex optimization

In this section, we discuss optimality conditions, representative applications, and basic solution methods for general convex optimization problems.

Consider the following nonlinear program:

$$\begin{aligned} \min_x \ & f_0(x) \\ \text{s.t.} \ & Ax = b \\ & f_i(x) \le 0, \ i = 1, \cdots, l, \end{aligned} \tag{8.16}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and f_i for each $i = 0, \cdots, l$ is a convex function from $\mathbb{R}^n$ to $\mathbb{R}$. If we write the feasible set $\mathscr{D} = \{x \in \mathbb{R}^n \mid Ax = b, \ f_i(x) \le 0, \ i = 1, \cdots n\}$, then it is easy to check that $\mathscr{D}$ is a convex set in $\mathbb{R}^n$. The above problem, which can be equivalently written as

$$\min_{x \in \mathscr{D}} f_0(x),$$

is therefore a convex optimization problem. If there is no constraint, then $\mathscr{D} = \mathbb{R}^n$ and this problem reduces to an unconstrained convex optimization problem

$$\min_{x \in \mathbb{R}^n} f_0(x).$$

8.4.1 Optimality conditions

Let us consider the general convex optimization problem (8.16). We can write the Lagrange function associated with this problem as

$$\mathscr{L}(x,y,z) = f_0(x) + y^T(Ax - b) + \sum_{i=1}^{l} z_i f_i(x),$$

where $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^l$ are two Lagrange multiplier vectors. The dual function is defined as

$$d(y,z) = \min_x \left\{ \mathscr{L}(x,y,z) = f_0(x) + y^T(Ax - b) + \sum_{i=1}^{l} z_i f_i(x) \right\}. \tag{8.17}$$

It can be shown that this dual function is concave regardless of convexity of functions f_i, $i = 0, \cdots, l$. The dual problem is define as

$$\max_{y \in \mathbb{R}^m, z \in \mathbb{R}^l_+} d(y,z). \tag{8.18}$$

A point (x^*, y^*, z^*) is called a saddle point of $\mathscr{L}$ if

$$\mathscr{L}(x^*, y, z) \le \mathscr{L}(x^*, y^*, z^*) \le \mathscr{L}(x, y^*, z^*), \;\; \forall x \in \mathbb{R}^n,\; y \in \mathbb{R}^m, z \in \mathbb{R}^l.$$

If all functions f_i are differentiable, then under some conditions called *constraint qualifications*, the optimality condition in Theorem 8.3 can be rewritten as below:

$$\begin{cases} \nabla f_0(x^*) + A^T y^* + \sum_{i=1}^{l} z_i^* \nabla f_i(x^*) & = 0 \\ Ax^* & = b \\ f_i(x^*) \le 0, \;\; z_i^* \ge 0, \;\; f_i(x^*) z_i^* & = 0, \; i = 1, \cdots, l. \end{cases} \tag{8.19}$$

The above condition is called the KKT (Karush-Kuhn-Tucker) condition. The equations $f_i(x) z_i = 0$, $i = 1, \cdots, l$ included in the KKT condition are called the complementarity slackness conditions, as in linear programming. While solving the KKT condition numerically is still a challenging problem, it provides a tool to study properties of optimal solutions.

Under the aforementioned constraint qualifications, x^* is an optimal solution of the convex optimization problem (8.16) if and only if there exists $(y^*, z^*) \in \mathbb{R}^m \times \mathbb{R}^l$ such that (x^*, y^*, z^*) satisfies the above KKT condition. One such constraint qualification is called the Slater condition, which requires

$$\{x \in \mathbb{R}^n \mid Ax = b,\ f_i(x) < 0,\ \ i = 1, \cdots, l\} \neq \emptyset. \tag{8.20}$$

For example, when $Ax = b$ is absent, then the Slater condition means that there exists $x \in \mathbb{R}^n$ such that $f_i(x) < 0$ for all $i = 1, \cdots, l$.

8.4.2 Example applications

8.4.2.1 Application 1: LASSO problem in statistics

Problem description: As in the least squares problem, consider a linear regression model that generates the response vector b using a linear model of the form $b = a^T x + \varepsilon$. Here, a is a regressor, and x is a vector of parameters in $\mathbb{R}^n$ (which is unknown), and ε is a Gaussian noise. Suppose that we have a set of data points $\{(a_i, b_i)\}$ with the input $a_i \in \mathbb{R}^n$ and the response $b_i \in \mathbb{R}$ for $i = 1, \cdots, m$. The sum of squares of errors is computed by

$$E_n = \sum_{i=1}^{m} (a_i^T x - b_i)^2 = \|Ax - b\|^2,$$

where A is a matrix formed by rows $a_1^T, \cdots, a_m^T$, and $b = (b_1, \cdots, b_m)^T$. In contrast to the least squares problem whose single goal is to minimize E_n , the LASSO problem aims to both

- minimize this error, and
- select as few as possible the number of parameters x_j for $j = 1, \cdots, n$.

Penalized LASSO (least absolute shrinkage and selection operator): This problem can be formulated as

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda \sum_{j=1}^{n} |x_j|,$$

where $\lambda > 0$ is a tuning parameter that controls a trade-off between the two terms in the objective function.

8.4.2.2 Application 2: Robust linear programming

Problem description: Consider the following linear program:

$$\min_x c^T x \quad \text{s.t. } a_i^T x \le b_i, \text{ for } i = 1, \cdots, m.$$

Now, suppose that, for each $i = 1, \cdots, m$, the coefficient vector a_i is allowed to vary whinin a unit ball centered at a given vector $\bar{a}_i$. That is,

$$a_i \in B_1(\bar{a}_i) := \{a_i \in \mathbb{R}^n \mid \|a_i - \bar{a}_i\| \le 1\}.$$

The goal is to find an optimal solution x that minimizes the objective $c^T x$, while satisfying the constraints $a_i^T x \le b_i$ for all $a_i \in B_1(\bar{a}_i)$ and all $i = 1, \cdots, m$.
Problem formulation: The robust linear constraint $a_i^T x \le b_i$ for all $a_i \in B_1(\bar{a}_i)$ becomes $\max_{a_i \in B_1(\bar{a}_i)} a_i^T x \le b_i$. Solving the right-hand side directly, we get

$$\bar{a}_i^T x + \|x\| \le b_i, \quad i = 1, \cdots, m.$$

Hence, we can formulate the robust linear program as follows:

$$\min_x c^T x \quad \text{s.t. } a_i^T x + \|x\| \le b_i, \text{ for } i = 1, \cdots, m.$$

One can check that this is a convex optimization problem (exercise).

8.4.3 Solution methods

8.4.3.1 Unconstrained problems with smooth objective functions

Consider the following unconstrained convex program:

$$f^* = \min_{x \in \mathbb{R}^n} f(x), \tag{8.21}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and smooth (i.e., differentiable with a continuous gradient).
Gradient method: One of the simplest methods to solve (8.21) is called the gradient method, which is an iterative method widely used in machine learning. It generates a sequence of vectors $\{x^k\}$ in $\mathbb{R}^n$ such that $\{f(x^k)\}$ converges to f^*, the optimal value of (8.21). More specifically, starting from an initial point $x^0 \in \mathbb{R}^n$, it

updates x^{k+1} from the current x^k at each iteration k as

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k),$$

where $\alpha_k \in \mathbb{R}$ is called the stepsize. The stepsize α_k can be pre-specified, or chosen by a subroutine inside each iteration. The simplest strategy is to use a constant stepsize for all iterations. For example, if the gradient $\nabla f(x)$ is Lipschitz continuous, meaning that there exists a constant $L_f \in [0, +\infty)$ such that $\|\nabla f(x) - \nabla f(y)\| \leq L_f \|x - y\|$ for all $x, y \in \operatorname{dom}(f)$, then one can choose α_k to be a constant in the interval $(0, \frac{2}{L_f})$, and the optimal choice is $\alpha_k = \frac{1}{L_f}$ in terms of the fastest guaranteed convergence rate. See [15] for more details on gradient methods and other methods on convex optimization.

Example - *Least-squares problem*: We consider the following least-squares problem

$$\min_x \frac{1}{2}\|Ax - b\|^2.$$

The objective function $f(x) = \frac{1}{2}\|Ax - b\|^2$ is convex and smooth. Its gradient is $\nabla f(x) = A^T(Ax - b)$, which is Lipschitz continuous with $L_f = \lambda_{\max}(A^T A)$, the maximum eigenvalue of $A^T A$.

If we apply the above gradient method to solve this problem, we have

$$x^{k+1} = x^k - \frac{1}{\lambda_{\max}(A^T A)} A^T (Ax^k - b).$$

Example 8.5. Consider an example with

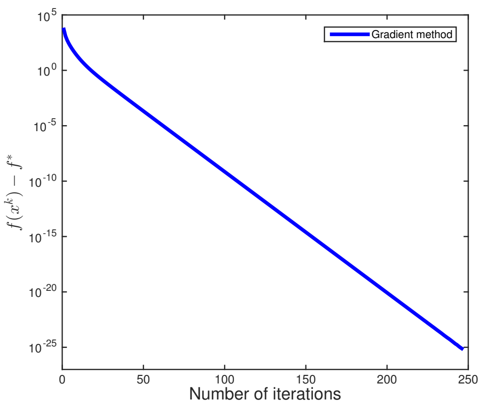
- $m = 50$, $n = 200$.
- `A = randn(m, n)`
- $b = Ax^\natural + \texttt{randn}(0.01)$, where $x^\natural$ is given.

The plot shows the decrease of the objective residual $f(x^k) - f^*$ along the iteration k.

8.4.3.2 Methods for general convex programs

Methods for general convex optimization, unconstrained or constrained, vary from one class of problems to another. Below are some classes of methods.

- **Nonsmooth optimization methods:** Subgradient, bundle, mirror descent, and proximal methods are usually used to solve nonsmooth convex programs.



- **First-order methods:** Gradient, fast gradient, conditional gradient, coordinate descent, stochastic gradient descent methods using only function values and gradients are called first order methods. These methods are widely used in large-scale applications nowadays (e.g., machine learning, deep learning, and data sciences).
- **Second order methods:** Newton and quasi-Newton methods are also used to solve convex programs which often have fast convergence speed. These methods are recently combined with stochastic methods to obtain scalable methods for solving large-scale applications in machine learning and statistical learning.
- **Interior-point methods:** Interior-point methods can be used to solve different classes of convex problems such as conic programming or geometric programming.

8.4.4 Software

Solvers for convex programming: There are several software packages for certain classes of convex programming problems. Below are some examples.

- **Conic programming:** LPs, QPs, conic programs can be solved by several interior-point solvers including SeDuMi, SDPT3, SDPA, and Mosek.
- **First-order methods:** TFOCS is an open-source software package based on first-order methods to solve different convex optimization problems. This package was developed by S. Becker using Matlab, and is available at http://cvxr.com/tfocs/http://cvxr.com/tfocs/http://cvxr.com/tfocs/.

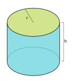
Modeling languages for convex programming:

There exist several modeling languages for convex optimization. The following two packages are widely used:

- **CVX:** This software package is implemented in Matlab by M. Grant and S. Boyd. Initially, it was open-source and is available for users with free-of-charge. It can be downloaded from http://cvxr.comhttp://cvxr.comhttp://cvxr.com.
- **YALMIP:** This software package is also implemented in Matlab by Johan Löfberg. It is also open-source and is available for users with free-of-charge. It can be downloaded from https://yalmip.github.iohttps://yalmip.github.iohttps://yalmip.github.io.

8.5 Exercises

Exercise 8.1. We are going to make an open-topped cylinder as given in the figure below. Here $r \geq 0$ is the radius of the bottom, and $h \geq 0$ is the height of the cylinder



(in inches). The material using to make the bottom and the side area costs \$2 and \$1.5 per square inch, respectively. Assume that we would like to keep the volume of this cylinder at least 100 cubic inches.

(a) Formulate this problem as an optimization problem for finding r and h such that it minimizes the total material cost.

(b) Solve the optimization problem obtained in part (a) assuming that the volume is fixed at 100 cubic inches.

Exercise 8.2. (a) Given the following symmetric matrices:

$$Q_1 = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 1 & 4 \\ 4 & 16 \end{bmatrix}, \quad Q_3 = \begin{bmatrix} 1 & 4 \\ 4 & 17 \end{bmatrix}, \quad Q_4 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}, \quad Q_5 = \begin{bmatrix} 6 & 8 & 8 \\ 8 & 6 & 2 \\ 8 & 2 & 10 \end{bmatrix}$$

Which of these matrices is: (i) positive semidefinite, (ii) positive definite, and (iii) not positive semidefinite? Prove your answer.

(b) Prove that for any matrix A in $\mathbb{R}^{m\times n}$, the matrix $Q = A^T A$ is symmetric and positive semidefinite.

(c) A function $f : \mathbb{R}^3 \to \mathbb{R}$ is defined as $f(x) = x_1^2 + 2x_2^2 + x_3^2 - 2x_1x_2 + x_1x_3 + 3x_2x_3 + 2x_1 + x_2 + 3x_3$. Show that we can write $f(x)$ as $f(x) = \frac{1}{2}x^T Qx + q^T x$, where Q is a 3×3 symmetric matrix, and q is a vector in $\mathbb{R}^3$. Is Q positive definite or positive semidefinite? Prove your answer.

Exercise 8.3. The following data is obtained as the rounded values $\bar{f}(t_i)$ from a cubic function $f(t) = a_0 + a_1t + a_2t^2 + a_3t^3$ at given time periods t_i for $i = 1, 2, \cdots, 10$. Since we do not know the coefficients a_0, a_1, a_2 and a_3, and we want to evaluate

Time t_i [in seconds]	1	2	3	4	5	6	7	8	9	10
$\bar{f}(t_i)$	120	140	160	180	200	220	240	260	280	301

them using the above data. Because the the data is rounded, we no longer have $f(t_i) = \bar{f}(t_i)$, but we instead have $f(t_i) \approx \bar{f}(t_i)$.

(a) Formulate this problem as an optimization problem by minimizing the sum of square of all the differences $f(t_i) - \bar{f}(t_i)$ for $i = 1, \cdots, 10$. Here, the vector of variables $x = (a_0, a_1, a_2, a_3)^T$ is in $\mathbb{R}^4$.
(b) Is it a quadratic program? If so, write down the matrix Q in the form $\frac{1}{2}x^T Qx + q^T x$, and show that Q is symmetric positive semidefinite.
(c) Solve this problem using Fermat's rule.

Exercise 8.4. A communication company plans to place 10 broadcast stations at given locations $a^{(1)}, \cdots, a^{(10)}$ in 10 regions. Each station has its own maximum *signal coverage area* of radius r_i (in feet). The radius r_i of each station i is at least $\underline{r}$ feet and at most $\bar{r}$ feet ($i = 1, \cdots, 10$). To cover each square ft of the *signal coverage area* of the i-th station, a cost of \$$c_i$ is incurred. The company can remotely control these stations by placing a central station at a location x. This location should be covered by all stations $a^{(i)}$ in their *signal coverage area*, but it must be at most r_0 feet far from the company's headquarter located at a given place $a^{(0)}$. The goal is to find the location x of the central station and the radius r_i of the i-th station to minimize the total cost. The input data is given as follows:

- The location of all stations is $a^{(1)} = (0,4)$, $a^{(2)} = (1,5)$, $a^{(3)} = (2,3)$, $a^{(4)} = (2,1)$, $a^{(5)} = (3,6)$, $a^{(6)} = (4,5)$, $a^{(7)} = (4,1)$, $a^{(8)} = (5,2)$, $a^{(9)} = (6,5)$ and $a^{(10)} = (7,4)$.
- The location of the headquarter is $a^{(0)} = (4,4)$, the upper and lower bounds of the radius are $r_0 = 1$, $\bar{r} = 5$, and $\underline{r} = 0.02$.
- The unit costs are $c = [1;2;1.2;2.5;2.1;1.1;1.8;1.4;1.35;1.82]^T$.

The unit of the distances r_i, $\underline{r}$, $\bar{r}$, r_0 and the coordinates of $a^{(i)}$ are in 10^5 feet, and the unit cost c_i is in dollars. The distance is measured by the Euclidean distance, which is defined as $\|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ for any two points $x = (x_1, x_2)$ and $y = (y_1, y_2)$ in $\mathbb{R}^2$.

(a) Formulate this problem into an optimization problem where the variables are the radius r_i and the location x of the central station. This cost excludes the cost of building the central station.

(b) Now, we replace each *signal coverage area* from a circle to a square, where the center of this square is $a^{(i)}$ and the length of its edges is $2r_i$ for $i = 0, \cdots, 10$. Reformulate the problem in part (a) into an optimization problem? Is it a quadratic program or a linear program? Show your result in detail.

Chapter 9

Introduction to Integer Programming

9.1 Modeling with integer variables

Like linear programming problems, integer programming (IP) problems are a special class of optimization problems. In an integer program, some or all of the decision variables are required to be integers. In any integer program considered in this course, the objective function and all constraints other than the integrality requirement must be linear. Accordingly, if we drop the integrality requirement in an integer program, we will obtain a linear program, which is called the **LP relaxation** of the integer program.

Integer programs can be classified into **pure** integer programs and **mixed** integer programs. An integer program with all of its variables required to be integers is a pure integer program. An integer program with some variables not subject to the integrality constraint is a **mixed integer program**.

Example 9.1. Woody's Furniture Company produces chairs and tables. Chairs are made entirely out of pine, and each chair uses 14 linear feet of pine. Tables are made of pine and mahogany, and each table uses 26 linear feet of pine and 15 linear feet of mahogany. Each chair requires 8 hours of labor to produce, and each table requires 3 hours of labor. The profit from each chair is $35, and the profit from each tables is $60.

Woody has 190 linear feet of pine and 60 linear feet of mahogany available each day, and has a work force of 92 labor hours each day. How should Woody use his resources to achieve the maximum daily profit, if the numbers of chairs and tables he makes daily have to be integers?

IP model: Let x_1 and x_2 denote the numbers of chairs and tables to be made each day respectively. We formulate the following IP problem:

$$\begin{cases} \max_x \ z = 35x_1 + 60x_2 & \text{profit} \\ 14x_1 + 26x_2 \le 190 & \text{pine} \\ 15x_2 \le \ 60 & \text{mahogany} \\ 8x_1 + \ 3x_2 \le \ 92 & \text{labor} \\ x_1, x_2 \ge 0 \text{ and are integer} \end{cases} \tag{9.1}$$

LP relaxation: By dropping the integrality constraint on x, we obtain the following LP relaxation:

$$\begin{cases} \max_x \ z = 35x_1 + 60x_2 & \text{profit} \\ 14x_1 + 26x_2 \le 190 & \text{pine} \\ 15x_2 \le \ 60 & \text{mahogany} \\ 8x_1 + \ 3x_2 \le \ 92 & \text{labor} \\ x_1, x_2 \ge 0 \end{cases} \tag{9.2}$$

Solving the LP problem (9.2) graphically as shown in Figure 9.1, we find its optimal solution to be $(10.976, 1.398)$. Rounding the solution to nearest integers, we obtain

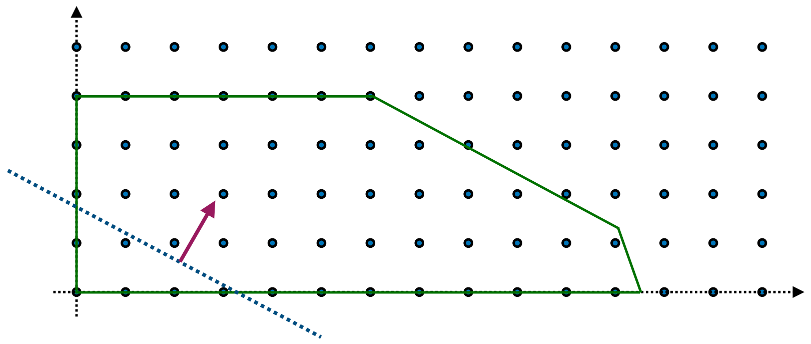


Fig. 9.1 Graphical illustration of Woody's IP and the LP relaxation

an integer solution $(11,1)$. However, the optimal solution for (9.1) is in fact $(8,3)$ instead of (11,1). In general, one cannot find an optimal solution of an integer program by rounding an optimal solution of its LP relaxation to integer. A basic method to solve integer programs is called the **branch and bound method**, the topic of next section.

GAMS model: To solve the integer program (9.1) using GAMS, we declare x_1 and x_2 as integer variables. This is done in furniture.gms by the following statement

```
integer variables x(I);
```

where `I` is the set that contains two labels “chair” and “tab.” By default, GAMS sets a lower bound of zero to integer variables, so it is not necessary to include the nonnegativity constraint of `x(I)` in the model. In the solve statement,

```
solve woody using mip maximizing profit;
```

the word “mip” is the model type that stands for mixed integer programming.

```
*furniture.gms
SET
        I "Type of products" /chair, tab/
        J "Type of resource" /pine, mahogany, labor/;
PARAMETERS
        sellingPrice(I) /chair 35, tab 60/,
        resLimit(J) /pine 190, mahogany 60, labor 92/;
TABLE  a(I,J)
                pine    mahogany    labor
        chair    14      0            8
        tab      26      15           3;
FREE VARIABLE profit;
INTEGER VARIABLES x(I);
EQUATIONS obj, resLim(J);
    obj.. profit =e=   sum(I, sellingPrice(I)*x(I));
    resLim(J).. sum(I, a(I,J)*x(I)) =l= resLimit(J);
MODEL Woody /all/;
    option mip = cplex;
    $onecho > cplex.opt
    epgap=0.01
    $offecho
    woody.optfile = 1;
SOLVE Woody USING mip MAXIMIZING profit;
```

In (9.1), the integrality constraints on x_1 and x_2 arise from the explicit requirement that only integer values are acceptable as numbers of products. Integer variables are also widely used to represent yes/no decisions in optimization models. The following example is a typical capital budgeting problem. In such a problem, we use binary variables to represent decisions on whether or not to make investments.

Example 9.2. Stockco is considering four investments. Investment 1 will yield a net present value (NPV) of \$16000; investment 2, an NPV of \$22000; investment 3, an NPV of \$12000; and investment 4, an NPV of \$8000. Each investment requires a certain cash outflow at the present time: investment 1: \$5000; investment 2: \$7000; investment 3: \$4000; investment 4: \$3000. Stockco has a total of \$14000 to invest. How does Stockco maximize the total NPV?

IP model: We begin by defining a variable for each decision Stockco needs to make. This results in four variables x_j, $j = 1,2,3,4$. Each x_j is a binary variable that represents the decision on investment j, with $x_j = 1$ corresponding to the decision of making investment j, and $x_j = 0$ corresponding to the decision of not making investment j. The following IP is what Stockco needs to solve.

$$\begin{cases} \max\limits_{x} \ 16x_1 + 22x_2 + 12x_3 + 8x_4 \\ \text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 \quad \leq 14 \\ \qquad x_j = 0 \text{ or } 1, \quad j = 1,2,3,4. \end{cases} \tag{9.3}$$

The expression $16x_1 + 22x_2 + 12x_3 + 8x_4$ is the total NPV obtained by Stockco under the decision represented by the variables x_j. If $x_j = 1$, then the NPV of investment j is included in this expression; if $x_j = 0$ then the NPV of investment j is not included. Whatever combination of investments is undertaken, the expression gives the total NPV for that combination. Similarly, the expression $5x_1 + 7x_2 + 4x_3 + 3x_4$ is the total cash outflow to be paid by Stockco under the decision represented by variables x_j.

Next, we show how to express logical conditions on decisions by linear constraints on the corresponding binary variables. Consider the following requirements on the investment decisions of Stockco.

1. Stockco can invest in at most two investments.
2. If Stockco invests in investment 2, they must also invest in investment 1.

3. If Stockco invests in investment 2, they cannot invest in investment 4.

To account for the first requirement, add the constraint

$$x_1 + x_2 + x_3 + x_4 \leq 2$$

to (9.3). This eliminates any choice of three or four investments.

In terms of x_1 and x_2, the second requirement states that if $x_2 = 1$, then x_1 must also equal 1. If we add the constraint

$$x_2 \leq x_1$$

to (9.3), then we have eliminated the case in which $x_1 = 0$ and $x_2 = 1$. The other cases are (1) $x_1 = 0$ and $x_2 = 0$, (2) $x_1 = 1$ and $x_2 = 0$, (3) $x_1 = 1$ and $x_2 = 1$. Those cases satisfy both the requirement and the constraints. In summary, the constraint $x_2 \leq x_1$ is consistent with the second requirement in all cases regarding values of x_1 and x_2.

In terms of x_2 and x_4, the third requirement states that if $x_2 = 1$, then x_4 must equal 0. If we add the constraint

$$x_2 + x_4 \leq 1$$

to (9.3), then we have eliminated the case in which $x_2 = 1$ and $x_4 = 1$. The other cases are (1) $x_2 = 1$ and $x_4 = 0$, (2) $x_2 = 0$ and $x_4 = 0$, (3) $x_2 = 0$ and $x_4 = 1$. Those cases satisfy both the requirement and the constraint. In summary, the constraint $x_2 + x_4 \leq 1$ is consistent with the third requirement in all cases regarding values of x_2 and x_4.

It is important to note that the added constraints must be linear. For example, another constraint consistent with the third requirement would be $x_2 x_4 = 0$. However, the latter constraint is nonlinear, and therefore should not be included in an IP. Recall that an IP studied in this course cannot have nonlinear constraints except the integrality requirement.

GAMS model: Now, we provide a GAMS model to solve this problem:

```
*capitalB.gms
SET I investments /1*4/;
SCALAR cashLim /14/;
```

```
PARAMETERS
    NPV(I) in thousands /1 16, 2 22, 3 12, 4 8/,
    outflow(I) in thousands /1 5, 2 7, 3 4, 4 3/;
FREE VARIABLE profit;
BINARY VARIABLES x(I);
EQUATIONS obj, obCashLim;
    obj.. profit =e=   sum(I, NPV(I)*x(I));
    obCashLim.. sum(I, outflow(I)*x(I)) =l= cashLim;
MODEL Stockco /ALL/;
SOLVE Stockco USING mip MAXIMIZING profit;
```

9.2 The branch and bound method

The branch and bound method is the most widely used method for solving integer programs. Given an IP, the method proceeds as follows.

1. Remove the integrality constraints from the problem to obtain its LP relaxation. Find an optimal solution and the optimal value of the LP relaxation.
2. If all components of the optimal solution of the LP relaxation are integers, stop the algorithm: we already find the integer optimal solution for the original IP. Otherwise, branch on a variable with a fractional value to create two subproblems. (If more than one variable has a fractional value, arbitrarily choose any of these variables to branch on.)
3. Continue solving these subproblems, branching to further subproblems by adding appropriate constraints. Each time we create a new problem, we add a node to the branching-and-bound tree.
4. There are three situations in which we **fathom** a node, that is, stop branching from a subproblem.

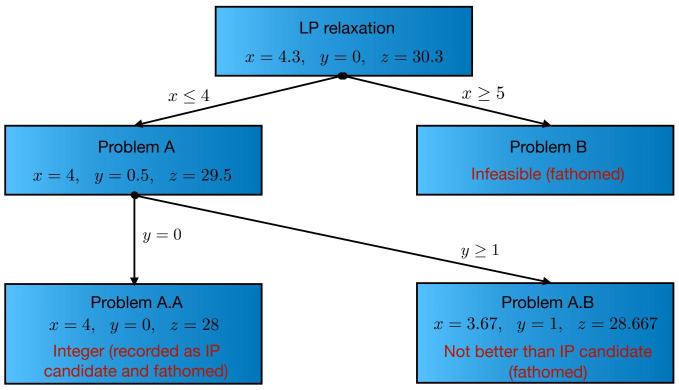
    a. This subproblem is **infeasible**.
    b. This subproblem has an **integer optimal solution**. If this integer solution achieves a better objective value than all the previously obtained integer solutions, then it becomes the **candidate IP solution** obtained so far.
    c. The optimal value of this subproblem is not better than the objective value achieved by the current candidate IP solution.

5. When there are no nodes to further branch on, stop the algorithm, with the current candidate IP solution being the IP optimal solution.

Example 9.3. Consider the following IP.

$$\begin{cases} \max_{x,y} & z = 7x + 3y \\ & 2x + y \le 9 \\ & 3x + 2y \le 13 \\ & x, y \ge 0, \text{ and are integers.} \end{cases}$$

1. Solve the LP relaxation, to find its optimal solution is $(x, y) = (4.3, 0)$, with $z = 30.3$.
2. Since x is fractional, branch on x. Add a constraint $x \le 4$ to the LP relaxation to obtain problem A, and another constraint $x \ge 5$ to the LP relaxation to obtain problem B. The optimal solution for problem A is $(4, 0.5)$, with $z = 29.5$. Problem B is infeasible, so it is fathomed.
3. Starting from problem A, branch on y because the y value of its optimal solution is fractional. Add a constraint $y \le 0$ to problem A to obtain problem AA, and another constraint $y \ge 1$ to problem A to obtain problem AB. The optimal solution for problem AA is (4,0), with $z = 28$. This is the first integer solution we find so far, so we record this as the candidate IP solution, and fathom problem AA. The optimal solution for problem AB is (3.67, 1), with $z = 28.667$. Because any integer feasible solution for problem AB will have an optimal value no more than 28, and hence no better than the current candidate IP solution, we fathom problem AB as well.
4. We do not have a node to further branch on, so we stop. The current candidate IP solution, (4,0), is an optimal solution for the original IP.


LP relaxation
$x = 4.3, \quad y = 0, \quad z = 30.3$
$x \leq 4$
$x \geq 5$
Problem A
$x = 4, \quad y = 0.5, \quad z = 29.5$
Problem B
Infeasible (fathomed)
$y = 0$
$y \geq 1$
Problem A.A
$x = 4, \quad y = 0, \quad z = 28$
Integer (recorded as IP candidate and fathomed)
Problem A.B
$x = 3.67, \quad y = 1, \quad z = 28.667$
Not better than IP candidate (fathomed)

9.3 Fixed charge problems

A fixed charge is an expense that does not depend on the level of goods or services produced by a business. This is in contrast to variable costs, which are volume-related (and are paid per quantity produced). To model a fixed charge in an optimization problem often requires using a binary variable.

Example 9.4. Gandhi Cloth Company makes three types of clothing: shirts, shorts, and pants. Gandhi needs to rent the appropriate machinery to make each type of clothing. The machinery can only be rented weekly. The weekly machinery rental cost, unit production cost, unit sales price, unit labor usage and unit cloth usage for each type of clothing are given below.

	Costs($)		Price($)	Unit resource usage	
Type	Rental	Production		Labor(hr)	Cloth(sq. yd.)
Shirt	200	6	12	3	4
Shorts	150	4	8	2	3
Pants	100	8	15	6	4

Each week, 150 labor hours and 160 sq. yds. of cloth are available. How should Gandhi plan weekly production to maximize his profit?

As in linear programming formulations, we define a variable for each decision that Gandhi must make. Since Gandhi needs to decide how many of each type of clothing should be manufactured each week, we define x_1, x_2, and x_3 to be the numbers of shirts, shorts and pants produced each week respectively. To account for the renting cost in the objective function, we define three binary variables y_i, $i = 1,2,3$, to represent whether or not to rent each type of machinery (shirts, shorts, pants). For example, $y_1 = 1$ represents the decision of renting shirt machinery, and $y_1 = 0$ represents the decision of not renting shirt machinery.

The IP model is as follows:

$$\begin{cases} \max\limits_{x} & (12x_1 + 8x_2 + 15x_3) - (6x_1 + 4x_2 + 8x_3) \\ & -(200y_1 + 150y_2 + 100y_3) \\ \text{s.t.} & 3x_1 + 2x_2 + 6x_3 \leq 150, \text{ (labor constraint)} \\ & 4x_1 + 3x_2 + 4x_3 \leq 160, \text{ (cloth constraint)} \\ & x_i \geq 0, i = 1,2,3, \\ & y_i = 0 \text{ or } 1, \ i = 1,2,3, \\ & x_i \leq M_i y_i, \ i = 1,2,3 \text{ (relation between } x_i \text{ and } y_i) \end{cases}$$

The parameter M_i that appears in the above model needs to be a large number that x_i does not exceed in any feasible solution. For example, the cloth constraint implies that $x_1 \leq \frac{160}{4} = 40$ in any feasible solution, so we can safely choose M_1 to be 40. Similarly, we can choose $M_2 = \frac{160}{3}$ and $M_3 = \frac{160}{4} = 40$. With such choices of M_i, the constraint $x_i \leq M_i y_i$ models the relation "if $x_i > 0$ then $y_i = 1$" (if any shirts are to be produced, then the shirts machinery must be rented) without imposing any unnecessary restriction on values of x_i. If M_1 were not chosen large (say $M_1 = 10$), then this constraint would unnecessarily restrict the value of x_1.

Other than the requirement that M_i needs to be an upper bound for x_i, the choice of M_i is flexible. For example the model in fixecharge.gms chooses a common M for all i. Finding an upper bound for the decision variables are usually not a problem in realistic models.

```
*fixedcharge.gms
SET
    I /shirt, shorts, pants/
    J /labor,cloth/;
SCALAR M /60/;
PARAMETERS
    sellingPrice(I) /shirt 12, shorts 8, pants 15/,
    varCost(I)  /shirt 6, shorts 4, pants 8/,
    fixCost(I) /shirt 200, shorts 150, pants 100/,
    resLimit(J) /labor 150, cloth 160/;
TABLE  a(I,J)
              labor    cloth
    shirt     3        4
    shorts    2        3
    pants     6        4;
FREE VARIABLE profit;
POSITVE VARIABLES x(I);
BINARY VARIABLES y(I);
EQUATIONS obj, resLim(J), bigM(I);
    obj.. profit =e= sum(I, sellingPrice(I)*x(I)) - sum(I, varCost(I)*x(I))
                  - sum(I,fixCost(I)*y(I));
    resLim(J).. sum(I, a(I,J)*x(I)) =l= resLimit(J);
    bigM(I).. x(I) =l= M * y(I);
MODEL Fixcharge /ALL/;
SOLVE Fixcharge USING mip MAXIMIZING profit;
```

9.4 Exercises

Exercise 9.1. For each $j = 1,2,\cdots,7$, let x_j be a binary variable which equals 1 if we invest in project j, and 0 if we do not. Consider the following statements:

F. We can invest in at most two projects, among projects 1, 2, and 3.
G. If we invest in project 4, we have to invest in project 5 as well.
H. We must invest in either project 6 or project 7, but not both.

Consider the following constraints:

I. $x_1+x_2+x_3 \geq 2$ II. $x_1+x_2+x_3 \leq 2$
III. $x_4-x_5 \geq 0$ IV. $x_5-x_4 \geq 0$
V. $x_6+x_7 \geq 1$ VI. $x_6+x_7 = 1$

Which of the following correctly matches the statements to their corresponding constraints?

(a) F-I, G-III, H-V (b) F-II, G-III, H-VI
(c) F-II, G-IV, H-V (d) F-II, G-IV, H-VI

Exercise 9.2. A manufacturer produces two types of products. The unit selling prices are \$2,000 for product 1 and \$5,000 for product 2. The production uses a type of material. Each unit of product 1 uses 3 units of material, and each unit of product 2 uses 6 units of material. A total of 120 units of material are available. If any positive amount of product 1 needs to be produced, a setup cost of \$10,000 is incurred. If any positive amount of product 2 needs to be produced, a setup cost of \$20,000 is incurred.

Formulate a mixed integer program to maximize the total profit, by using binary variables to model the setup cost. Create a GAMS file *products.gms* to solve the problem. We assume that the manufacturer can produce fractional amounts, so it is not necessary to use integer variables to represent the amounts of products (but there are other integer variables in the formulation). The optimal value is \$80,000.

Exercise 9.3. The Lotus Point Condo Project will contain both homes and apartments. The site can accommodate up to 1000 dwelling units. (That is, the sum of numbers of homes and apartments cannot exceed 1000.) The project must also contain a recreation project: either a swimming-tennis complex or a sailboat marina, but not both. If a marina is built, then the number of homes in the project must be at least triple the number of apartments in the project.

A swimming-tennis complex will cost \$2.8 million to build, and a marina will cost \$1.2 million. The developers believe that each home will yield revenues with an NPV of \$46,000, and each apartment will yield revenues with an NPV of \$48,000. Each home (or apartment) costs \$40,000 to build.

Formulate an IP to help Lotus Point maximize profits. Create a GAMS file *condo.gms* to solve the problem. For simplicity, define the numbers of homes or apartments as nonnegative variables instead of integer variables (but there will be other integer variables in the formulation). The optimal value is \$5,300,000.

References

1. A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. *SIAM J. Optim.*, 12:79–108, 2001.
2. D.P. Bertsekas, Angelia Nedic, and A.E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.
3. Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.
4. Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.

5. J. Borwein and A. Lewis. *Convex analysis and nonlinear optimization*. CMS Books in Mathematics. Springer, second edition, 2006.
6. S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.
7. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
8. Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.
9. R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, 2nd edition, 1987.
10. M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.
11. G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 3rd edition, 1996.
12. S Hillier Frederick and J Lieberman Gerald. Introduction to operations research, 2005.
13. Olvi L Mangasarian, W Nick Street, and William H Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.
14. George L Nemhauser and Laurence A Wolsey. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988.
15. Y. Nesterov. *Lectures on convex optimization*, volume 137 of *Springer Optimization and Its Applications*. Springer, 2018.
16. Y. Nesterov and A. Nemirovski. *Interior-point Polynomial Algorithms in Convex Programming*. Society for Industrial Mathematics, 1994.
17. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.
18. J. Renegar. *A mathematical view of interior-point methods in convex optimization*, volume 2. SIAM, 2001.
19. R. T. Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series*. Princeton University Press, 1970.
20. Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976.
21. Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
22. L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.
23. R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2015.
24. H. Wendland. *Numerical Linear Algebra: An Introduction*. Cambridge University Press, 2018.
25. S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997.