MA231

Operational Research Methods
Lecture Notes

Lent Term[1]

2018/19

Dr Victor Verdugo

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

[1]Based on the notes of previous versions by Dr Xue Lu.

Contents

Part III

Optimisation and Simulation Modelling

Chapter 19

Queueing Theory

19.1 Introduction

What is it? Queueing (also spelled "queuing") theory is the study of queues. These are situations in which units (possibly customers) arrive for a service and must wait if the service facility is occupied. When the service rate is too low or the arrival rate is too high, excess waiting results; if the service capacity is high relative to the arrival rate, idle capacity results. In order to balance the costs associated with undercapacity and overcapacity, it is necessary to understand the behaviour of the queueing system. We will be interested in questions such as the expected number of customers in the system, the expected time that a customer will spend in the system, or the fraction of time that a server will be idle (i.e., not serving). Queueing theory is widely applicable, and queues and queueing models can take many forms.

A simple queueing system is sketched in Figure 19.1. A queue will be characterised by one or more servers, customers who arrive in some known pattern, and a service-time distribution. The servers will usually form parallel channels, some or all of which may be specialised, for example a "cash payment only" or "fewer than ten items" in a supermarket. Queues may also be linked in series, such as when a customer must pass through several servers before completing service, or in a manufacturing assembly line. Different queues may interfere with one other, as at traffic lights.

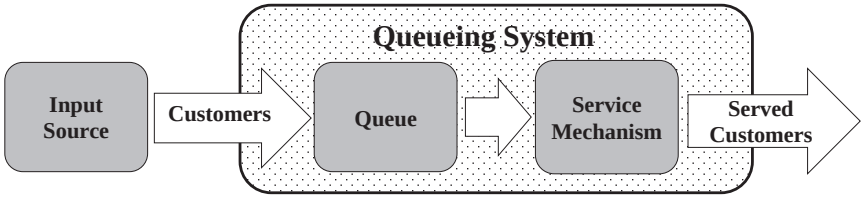


Figure 19.1: A simple queueing system schematic diagram.

There will usually be a known interarrival time distribution for the customers, but the arrival intervals may or may not be independent. Where one queue feeds into another, the

interdeparture pattern from the first is of interest, because it is the interarrival pattern for the second. Arrivals may also occur in batches, to enter a theatre for example. Service times may also be dependent, such as when the server is a machine which has to be set up for each job, and the setup time depends on both the last job completed and the job which is about to be started. In the real world there may be a system for reducing service times in emergencies or if the queue gets too long, and there may be a priority rule applying in the queue.

The customer population may be finite or infinite, and may be split into different categories, for example with separate airport border-control queues for "UK or EU citizens" and "other nationals". Customers may queue as expected, but may also vary their behavior in different ways: balking (not join the queue), reneging (leave the queue), jockeying (look for a shorter queue), or colluding (have one person hold a spot in line for his partners). Queueing systems typically have some initial, transient period in which the starting conditions are relevant, but then exhibit a steady-state behavior independent of the initial condition (much like Markov chains). Queueing theory usually focuses on the steady state.

In many real life situations queueing systems are very complex, and therefore simplifications are required in order to understand it. Sometimes, it is possible to obtain exact and analytical expressions for the quantities we aim to understand of queueing systems; even when not directly applicable, it can give general insight in the more complex situation. *Simulation* techniques can be used to estimate properties of the system when this is complex, and it has become widely used as growing computing power has made this more feasible and software has improved. We will revisit this briefly in the last part of the course.

19.2 Queueing systems

A *queueing system*, also referred to simply as a *system*, is comprised of the customers queueing to be served, and the customers being served. The *state of the system* is equal to the number of customers in the queueing system. That is, the number of customers queueing plus the number being served.

Example 19.1. *Suppose there are currently ten people in a shop: nine customers and the shopkeeper. The shopkeeper is serving one customer; two others are queueing, waiting to be served; and six are wandering around the shop, choosing goods to buy. In this case the queueing system consists of only three people: the customer who is being served and the two who are queueing. The system does not include the shopkeeper (the server, or servers) nor the customers who are still shopping.*

There are three key factors influencing the life cycles of individual queues: The *arrival process* of the customers (or items), the *rules* of the queue behavior and the *characteristics of the service* facility. In the following we detail each of this elements determining the queue.

19.2.1 Arrival process

The input source of the queueing system is the set of potential customers who *may* enter the system. In many cases the number of potential customers is very large, and therefore we will assume that the number of potential customers is unbounded.

Poisson process. The arrival time for each of the customer corresponds to a random variable and therefore, the total number of arrivals over time defines a stochastic process. A common assumption, which will be made throughout these notes, is that the stochastic process is a *Poisson process*. It is conventional to denote by $\lambda > 0$ the Poisson *arrival rate*, that is, the expected number of arrivals per unit time (e.g. an hour, a day). More specifically, we denote by $N(t)$ the total number of arrivals up to time t, and let $N(0) = 0$. There are many equivalent definitions for this process, but we will use a definition based on the following two assumptions:

(a) The process exhibits *independent increments*, that is, arrivals in non-overlapping time intervals are independent. In particular, this implies the process to be *memoryless*: the arrival process after time t is independent of the arrivals before time t.

(b) For every interval of length t, the number of arrivals is distributed according the Poisson distribution with mean λt. That is, for every $s, t \geq 0$ and for every $k \in \mathbb{Z}_+$,

$$\mathbb{P}(N(t+s) - N(s) = k) = e^{-\lambda t} \frac{(\lambda t)^k}{k!}.$$

We say that a process exhibits *stationary increments* if for every $s, t \geq 0$, the distribution of $N(t) - N(s)$ depends on $t - s$. That is, the number of arrivals in intervals of equal length are equally distributed. Equivalently, $(N(t))_{t \geq 0}$ defines a Poisson process if it exhibits independent and stationary increments, and for every $t \geq 0$, $\mathbb{P}(N(t) = 1) = \lambda t + o(t)$ and[1] $\mathbb{P}(N(t) \geq 2) = o(t)$. Furthermore, it can be shown that the interarrival times are independent and identically distributed according to an exponential distribution with mean $1/\lambda$. More specifically, let X_1 be the time of the first arrival, and let X_n be the time between the $(n-1)$-th and the n-th arrivals. Then, the variables $\{X_n\}_{n \in \mathbb{N}}$ are independent and for every $n \in \mathbb{N}$, $\mathbb{P}(X_n > t) = e^{-\lambda t}$. A proof of these statements is given in Appendix A.1.

Example 19.2. *If between 9am and midnight your mobile receives messages randomly at a typical rate of 3 messages per hour, this could be modeled by a Poisson process with rate $\lambda = 3$ (when time units are hours), with expected interarrival time $1/\lambda = 1/3$, i.e., 20 minutes.*

The Poisson process is very useful to model situations where arrivals can be considered independent of previous arrivals, and at most one customer arrives at any instant. This is often a good model, and admits variants such as time-dependent arrival rates, which is called the *non-homogeneous* Poisson process. In real life situations there can be bulk arrivals, such as groups of people coming to a restaurant, or arrivals can depend on previous ones through the number of people already in the queue, for example with people balking if the queue is too long. Even more, arrivals can depend on the time since the last arrival, for example with trains running on a regular schedule rather than coming randomly. These are examples where the assumptions made for constructing the Poisson process are no longer valid, although it can provide a good approximation, under some simplifications, to the system behavior.

[1] We say that a function f is $o(t)$ if $\lim_{t \to 0} f(t)/t = 0$.

19.2.2 Queue disciplines

The queue discipline determines the order in which customers are served. Below we list a few common examples.

- **FIFO**: *First-in first-out*, also known as *first-come first-served*. That is, the order in which customers are served coincides with the arrival order.
- **LIFO**: *Last-in first-out*. This is also thought of as a *stack*, that is, the last item put on top of the stack, that is, into the queue, is the first to be taken off.
- **Random**: The next customer in the queue to be served is selected at random. This happens, for example, when passengers are put on hold at the airport.
- **Priority queueing**: Customers are classified at arrival into different classes, with different priority levels. Within each category, customers are served, for example, according to FIFO. This is routinely done in hospital A&E departments.

The simplest queue discipline is FIFO, but it is worth mentioning that the discipline does not affect many quantities of interest, including the distribution of the queue length, the throughput, or the server idle time, since all these measures does not depend on the individual customers arrival or service time. On the other hand, the discipline *does affect* the waiting-time distribution. For example, according to FIFO the waiting times should be relatively equal, but in LIFO, you might get served almost immediately from the top of the stack, but also you might have a terribly long wait if you wind up buried in the stack.

19.2.3 Service Characteristics

We denote by μ the *service rate*. We will assume that service time is distributed according to an exponential distribution[2] with parameter μ, that is, with density equal to $s(t) = \mu e^{-\mu t}$. A consequence of having exponentially distributed service times is the following.

> The minimum of k independent random variables distributed according to exponential distributions of parameters $\mu_1, \ldots, \mu_k$ is an exponential random variable with parameter $\mu_1 + \mu_2 + \cdots + \mu_k$.

This property has direct a implication when analysing queueing systems with multiple servers. Suppose we have multiple independent servers all having service times exponentially distributed with the same parameter μ. In this case, if k people are being served simultaneously, the *service time of the system* is the minimum of the service times among the k busy servers, it is exponentially distributed thanks to the property above, and the service rate is the sum of the service rates, that is, $k\mu$.

19.2.4 Queue Development

Let us denote by $p_n(t)$ the probability that there are n customers in the system at time t, where[3] $n \in \mathbb{Z}_+$. Typically, there is a *transient phase* (e.g., just after a shop opens, before many

[2] While mathematically convenient, the assumption of exponentially distributed service times is often unrealistic, for example if the service time is roughly the same for every customer.

[3] We denote by $\mathbb{Z}_+ = \{0, 1, 2, \ldots, \}$ the set of non-negative integers.

customers have entered) where the distributions $p_n(t)$ vary considerably over time t, but, as in Markov chains, this leads on to a *steady state* where limit probabilities $\lim_{t\to+\infty} p_n(t) = p_n$ exist for all $n \in \mathbb{Z}_+$. The probability distribution p_n is the *steady state distribution* for n. In general, arrival and service rates may depend on the current state of the system. We use the following notation,

λ_n = arrival rate, or expected number of arrivals per unit time, at state $n \in \mathbb{Z}_+$.
μ_n = service rate, or expected number of services completed per unit time, at state $n \in \mathbb{N}$.

Note that μ_0 is not defined, because no customer is served if the system is empty. Common scenarios are ones where λ_n is increasing in n (e.g., customers attracted to a successful restaurant) or decreasing in n (e.g., customers balking if they see long queues).

19.3 Little's law

Before we proceed, we present Little's law. Named for John D.C. Little, who proved a version of it in 1961, the law holds in great generality for arrival-departure processes. We consider any system where customers arrive over time, spend some time in the system and then leave the system. Let $L(t)$ be the number of customers in the system at time t. We denote by W_n the amount of time that the n-th customer to arrive spends in the system. We make no assumption on the probability distributions of the interarrival times nor on the time spent in the system. Little's law gives a general relationship between the following three quantities, assuming they exist,

- The average number of arrivals over time $\lambda = \lim_{t\to+\infty} \dfrac{N(t)}{t}$,
- The average time spent in the system, $W = \lim_{n\to+\infty} \dfrac{1}{n} \sum_{j=1}^{n} W_j$,
- The average number of customers in the system, $L = \lim_{t\to+\infty} \dfrac{1}{t} \int_0^t L(\tau)\, d\tau$.

Little's law: If λ and W exist and are finite, then $L = \lambda W$.

The law is uninteresting if there is only a finite number of arrivals, since then $\lambda = 0$ and $L = 0$. The law also says nothing if the arrival rate exceeds the service rate, since then the servers cannot keep up with the customers joining the system, the queue grows infinitely, and both L and W are infinite.

Example 19.3. *Suppose that a shop opens at 9:00 in the morning and closes at 20:00. After closing, the manager observes the numbers of the day, and noticed that the average number of arrivals during the day was 30 customers per hour, and that the average time spent by the customers in the system is 90 minutes. Therefore, by Little's law the average number of customers in the system is* $30 \cdot 3/2 = 45$.

19.4 The birth-and-death process

In these notes we assume that the arrivals and departures of the queueing model behave according to a *birth-and-death process*, where a birth corresponds to an arrival in the system, and a death corresponds to a served customer. As shown in Figure 19.2, transitions are only from state n to $n+1$ (birth), and from state n to $n-1$ (death). In the following we assume the following.

(*i*) For every $n \in \mathbb{Z}_+$, interarrival times (births) are exponentially-distributed with rate λ_n (mean $1/\lambda_n$).

(*ii*) For every $n \in \mathbb{Z}_+$, service times (deaths) are exponentially-distributed with rate μ_n (mean $1/\mu_n$).

(*iii*) The interarrival times and service times are all mutually independent.

The exponential is memoryless. The exponential distribution satisfies the following property: If X is an exponentially-distributed random variable with rate λ, then $X - \tau$ conditioned on $X \geq \tau$ is also exponentially-distributed, and with the same rate. For example, if light bulb lifetimes were exponentially-distributed with mean 1 year, then the expected remaining lifetime of an 8 months old bulb would still be 1 year. Since the density of an exponential random variable of rate λ is $\lambda e^{-\lambda t}$, nearly $t = 0$ is $\approx \lambda$. That is, for a small value Δ, given that the event has not occurred by time t, the probability it occurs in the interval $(t, t+\Delta)$ is $\lambda\Delta + o(\Delta)$. We will use this property to study birth-and-death processes in the next section.

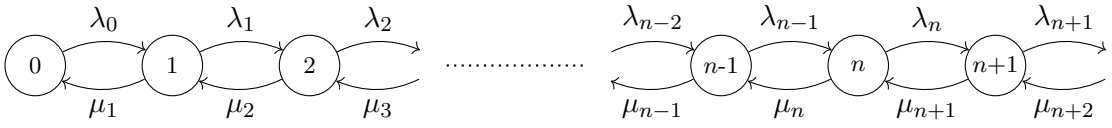


Figure 19.2: Representation of the birth-death process.

19.4.1 Steady state conditions: Kolmogorov equations

Thanks to the memoryless property of the exponential distribution, we have that

$$p_0(t+\Delta) = (1-\lambda_0\Delta)p_0(t) + \mu_1\Delta p_1(t) + o(\Delta),$$
$$p_n(t+\Delta) = \lambda_{n-1}\Delta p_{n-1}(t) + (1-(\lambda_n+\mu_n)\Delta)p_n(t) + \mu_{n+1}\Delta p_{n+1}(t) + o(\Delta),$$

for every $n \in \mathbb{Z}_+$. These expressions are obtained by conditioning on the state at time t. Therefore, by subtracting $p_n(t)$ and dividing by Δ, when $t \to 0$ we obtain the following set of differential equations,

$$p_0'(t) = -\lambda_0 p_0(t) + \mu_1 p_1(t),$$
$$p_n'(t) = \lambda_{n-1} p_{n-1}(t) - (\lambda_n + \mu_n) p_n(t) + \mu_{n+1} p_{n+1}(t), \text{ for every } n \in \mathbb{Z}_+,$$

that are known as *forward Kolmogorov differential equations*. In steady state, we assume that $p_n(t)$ converges to a constant p_n independent of the time t. In particular, in the limit we have that $p_n' = 0$ for every $n \in \mathbb{N}$, and therefore in steady state the left hand side of the equations above is equal to zero. For example, when $n = 0$ we obtain

$$\mu_1 p_1 = \lambda_0 p_0.$$

Similarly, for every state $n \in \mathbb{Z}_+$, we obtain that

$$\lambda_{n-1} p_{n-1} + \mu_{n+1} p_{n+1} = (\lambda_n + \mu_n) p_n.$$

State	Balance equation
$n = 0$	$p_1 \mu_1 = p_0 \lambda_0$
$n = 1$	$p_0 \lambda_0 + p_2 \mu_2 = p_1 \lambda_1 + p_1 \mu_1$
$n = 2$	$p_1 \lambda_1 + p_3 \mu_3 = p_2 \lambda_2 + p_2 \mu_2$
$\vdots$	$\vdots$
n	$p_{n-1} \lambda_{n-1} + p_{n+1} \mu_{n+1} = p_n \lambda_n + p_n \mu_n$
$\vdots$	$\vdots$

Table 19.1: Balance equations for the steady state of the birth-death process.

Table 19.1 shows the collection of all the balance equations. How can we solve for the stationary state probabilities p_n? It can be done straightforwardly, but there is an observation that makes it easier and more intuitive. The rates into and out of state 0 must balance, but these are the same as two of the rates into and out of state 1 (see Figure 19.2). Therefore, of the four rates into and out of state 1 that must balance, the pair to and from state 0 must balance, and therefore the pair to and from state 2 must also balance. Likewise, of the four rates to and from state 2 that must balance, we just showed that the pair between 2 and 1 must balance, thus the pair between 2 and 3 must also balance. We conclude that there is not just balanced flow into and out of each state, but balanced flow between each *pair* of states, a property often called *detailed balance*. Therefore, for every $n \in \mathbb{N}$, we obtain that

$$\mu_{n+1} p_{n+1} = \lambda_n p_n.$$

The state $n = 0$ balance equation gives $p_1 = (\lambda_0/\mu_1) p_0$; the $n = 1$ detailed balance equation gives $p_2 = (\lambda_1/\mu_2) p_1 = (\lambda_0 \lambda_1 / \mu_1 \mu_2) p_0$, and so forth. In general, we obtain the following.

In steady state, for $n \in \mathbb{Z}_+$, the probability that n customers are in the system is

$$p_n = \left(\prod_{j=0}^{n-1} \frac{\lambda_j}{\mu_{j+1}} \right) p_0. \tag{19.1}$$

Since $\sum_{n=1}^{\infty} p_n = 1$, it follows that in the steady state, the probability that no customers are in the system is

$$p_0 = \left(1 + \sum_{n=1}^{\infty} \prod_{j=0}^{n-1} \frac{\lambda_j}{\mu_{j+1}} \right)^{-1}. \tag{19.2}$$

Once the values of the arrival and departure rates λ_n and μ_n are known, assuming the steady state exists, p_0 can in principle be calculated using (19.2), whereupon the values of the steady state probabilities are determined by (19.5.1). Many queueing systems are based on the birth-and-death process. We will consider several cases, depending on whether there is only one server or multiple servers and on whether or not the arrival rates depend on the number of customers in the system (the state).

19.5 Arrival rates independent of the state

In this section we assume that the arrival rates are independent of the state of the system, namely,

$$\lambda_n \equiv \lambda, \text{ for every } n \in \mathbb{N}.$$

This assumption implies that the queueing system has infinite capacity: if capacity were capped, then the arrival rate would have to drop to 0 if the system was filled to capacity, as no more customers are allowed in.

We further assume that all servers are "identical", in the sense that each server has the same rate μ. Taking as an example two servers, when there is no customer in the system there is of course no service; when there is one customer, only one server can be at work (this is an assumption) and therefore service progresses at rate μ; when there are two or more customers, both servers can act and service progresses at rate 2μ.

19.5.1 Single server

We first consider the case of a single server. In this case, the service rates are also independent of the system state, that is,

$$\mu_n \equiv \mu, \text{ for every } n \in \mathbb{N}.$$

It is convenient to define the *traffic intensity* as $\rho = \lambda/\mu$. The traffic intensity ρ is the ratio of the mean number of arrivals and mean number of customers served per unit time. We assume that $\rho < 1$, otherwise the arrival rate is greater than the service rate and the system will never reach a steady state because the queue will increase unbounded.

In this case, the coefficients defined in in the previous section become $(\lambda/\mu)^n = \rho^n$. Then, (19.2) gives

$$p_0 = \left(1 + \sum_{n=1}^{\infty} \rho^n\right)^{-1} = 1 - \rho.$$

The last equality follows from the well known geometric series formula $\sum_{n=0}^{\infty} \rho^n = 1/(1-\rho)$; see equation (A.1) in Appendix A.2. We conclude the following.

The steady-state probability that n customers are in the system, for $n \in \mathbb{N}$, is

$$p_n = \rho^n(1-\rho). \tag{19.3}$$

In particular, the steady state probability of no customers in the system is

$$p_0 = 1 - \rho. \tag{19.4}$$

We are now set up to derive several indicators of the queue in this single-server setting.

1. The probability that the server is *idle* is $p_0 = 1 - \rho = 1 - \lambda/\mu$.

2. The probability that the server is busy is $1 - p_0 = \rho = \lambda/\mu$.

3. Let S denote the expected number of customers *being served*. If X is a random variable representing the number of customers being served at any given moment (so X equals either 0 or 1), then
$$S = \mathbb{E}(X) = 0 \cdot p_0 + 1 \cdot (1 - p_0) = \rho.$$

4. Let L_S denote the expected number of customers *in the system*. Then, using (19.3),
$$L_S = \sum_{n=0}^{\infty} n p_n = \sum_{n=0}^{\infty} n(1-\rho)\rho^n = (1-\rho)\sum_{n=0}^{\infty} n\rho^n = \frac{\rho}{1-\rho}.$$
The last equality follows from the arithmetic-geometric series formula $\sum_{n=0}^{\infty} n\rho^n = \rho/(1-\rho)^2$; see Appendix A.3. Alternatively, substituting $\rho = \lambda/\mu$, we express L_S as
$$L_S = \frac{\lambda}{\mu - \lambda}.$$

5. Let W_S denote the expected time *in the system*. By Little's law,
$$W_S = \frac{L_S}{\lambda} = \frac{1}{\mu - \lambda}.$$

6. Let L_q denote the expected number of customers *in the queue*. Recall that this does *not* include the one customer, if any, being served. Then
$$L_q = L_S - S = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho} = \frac{\lambda^2}{\mu(\mu-\lambda)}.$$

7. Let W_q denote the *expected queueing time*. Consider the queue itself as a system; departing from the queue means starting to get service. Applying Little's law to the queue,

$$W_q = \frac{L_q}{\lambda} = \frac{\lambda}{\mu(\mu - \lambda)}.$$

An alternative derivation is from the expression for W_S previously obtained, using that $W_S = \mathbb{E}(\text{queueing time} + \text{service time}) = W_q + 1/\mu$. This gives

$$W_q = W_S - \frac{1}{\mu} = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)}.$$

8. Let w_q denote the *conditional expected queueing time*, the expected time a customer has to queue given that there is already a queue when he or she enters the system. Then,

$$\begin{aligned} W_q &= \mathbb{P}(\text{no queue})\mathbb{E}(\text{queue length} \mid \text{no queue}) + \mathbb{P}(\text{queue})\mathbb{E}(\text{queue length} \mid \text{queue}) \\ &= p_0 \cdot 0 + (1 - p_0 - p_1) \cdot w_q, \end{aligned}$$

that is, $w_q = \dfrac{W_q}{1 - p_0 - p_1} = \dfrac{1}{\mu - \lambda}$.

Example 19.4. *Fred runs a small newsagents shop. At around midday, when the local school has its lunch hour, his only customers are school children. All the goods are on or behind the counter, so a child is either queueing or being served. On average a child enters the shop every 40 seconds during this period and can be served in 30 seconds. The arrival rate is* $\lambda = 90$ *customers per hour, and the service rate is* $\mu = 120$ *customers per hour, so the traffic density is* $\rho = 0.75$.

1. *The probability that the Fred is idle is* $p_0 = 1 - \rho = 0.25$.

2. *The proportion of time that Fred is busy is* 0.75.

3. *The expected number of children being served is* $\rho = 0.75$.

4. *The expected number of children in the shop is* $L_S = \frac{0.75}{1-0.75} = 3$.

5. *A typical child's expected time spent in the shop is* $W_S = \frac{L_s}{\lambda} = \frac{3}{90} = \frac{1}{30}$ *hours, that is, 2 minutes. (Alternatively,* $W_S = \frac{1}{120-90} = \frac{1}{30}$.*)*

6. *The expected queue length is* $L_q = 3 - 0.75 = 2.25$.

7. *A typical child's expected time spent in the queue is* $W_q = \frac{L_q}{\lambda} = \frac{2{,}25}{90} = \frac{1}{40}$ *hours, that is, 90 seconds.*

8. *A typical child's expected time spent queueing if someone is already in the shop when she or he arrives is* $w_q = \frac{1}{\mu-\lambda} = \frac{1}{30}$ *hours, that is, 2 minutes.*

19.5.2 Multiple identical servers

Suppose we have now s servers, all of them having the same service rate μ. Even though each has the same service rate μ, the service rate μ_n of the whole system depends on the number n of customers in the system. Indeed, when $n \leq s$ only n servers are busy, so the service rate is $n\mu$, whereas when $n > s$ all s servers are busy, so the service rate is $s\mu$. That is,

$$\mu_n = \begin{cases} n\mu, & n \in \{1, 2, \ldots, s\}, \\ s\mu, & n > s. \end{cases}$$

To compute the steady state distribution we need to compute the coefficients appearing in equation (19.5.1). For $n \leq s$, it equals to

$$\prod_{j=0}^{n-1} \frac{\lambda_j}{\mu_{j+1}} = \frac{\lambda^n}{\mu \cdot (2\mu) \cdots (n\mu)} = \frac{1}{n!} \left(\frac{\lambda}{\mu}\right)^n.$$

For $n > s$, we have

$$\prod_{j=0}^{n-1} \frac{\lambda_j}{\mu_{j+1}} = \frac{\lambda^n}{\mu \cdot (2\mu) \cdots (s\mu) \underbrace{(s\mu) \cdots (s\mu)}_{n-s \text{ times}}} = \frac{1}{s!} \left(\frac{1}{s^{n-s}}\right) \left(\frac{\lambda}{\mu}\right)^n.$$

In this case, the *traffic intensity* is given by

$$\rho = \frac{\lambda}{s\mu},$$

so that the system is stable if $\rho < 1$, and the queue increases indefinitely otherwise.

In steady state, the probability that n customers are in the system is $p_n = C_n p_0$, where

$$C_n = \begin{cases} \dfrac{(s\rho)^n}{n!} & n \in \{1, 2, \ldots, s\}, \\ \dfrac{(s\rho)^s}{s!} \rho^{n-s} & n > s. \end{cases} \tag{19.5}$$

From equation (19.2) we can calculate the value of p_0. In order to do this, we need to compute

$$\sum_{n=1}^{s} \frac{(s\rho)^n}{n!} + \frac{(s\rho)^s}{s!} \sum_{n=s+1}^{\infty} \rho^{n-s}.$$

Little can be done to simplify the first sum, but it is simply the finite sum of s terms. For the second sum, we have

$$\sum_{n=s+1}^{\infty} \rho^{n-s} = \sum_{i=1}^{\infty} \rho^i = \rho \sum_{i=0}^{\infty} \rho^i = \frac{\rho}{1-\rho}.$$

Therefore, we have shown the following.

The steady state probability that there are no customers in the system is

$$p_0 = \left(1 + \sum_{n=1}^{s} \frac{(s\rho)^n}{n!} + \frac{(s\rho)^s}{s!}\frac{\rho}{1-\rho}\right)^{-1}. \tag{19.6}$$

We now get the indicators of the queue in this multiple-server setting.

1. *Expected proportion of idle time per server.* When there are $n < s$ customers in the system, only n of the s servers will be busy, so the average proportion of idle servers is $1 - \frac{n}{s}$. Thus the expected proportion of idle time per server is

$$\sum_{n=0}^{s-1} \left(1 - \frac{n}{s}\right) p_n.$$

   This is simply a sum of s terms, hence it can be computed explicitly for any given problem.

2. *Expected number S of customers being served.* If X is a random variable representing the number of customers being served, then it is supported over $\{0, 1, \ldots, s\}$. Then,

$$S = \mathbb{E}(X) = \sum_{n=1}^{s} n \cdot p_n + s \sum_{n=s+1}^{\infty} p_n = p_0 \left(\sum_{n=1}^{s} \frac{(s\rho)^n}{(n-1)!} + \frac{(s\rho)^s}{s!} \sum_{n=s+1}^{\infty} s\rho^{n-s}\right).$$

3. *Expected number L_S of customers in the system.* We can compute it by

$$L_S = \sum_{n=1}^{\infty} n \cdot p_n = p_0 \left(\sum_{n=1}^{s} \frac{(s\rho)^n}{(n-1)!} + \frac{(s\rho)^s}{s!} \sum_{n=s+1}^{\infty} n\rho^{n-s}\right).$$

4. *Expected time W_S in the system.* By Little's law, this is equal to L_S/λ.

5. *Expected number L_q of customers in the queue.* Recall that this does *not* include the customers, if any, being served. Then, $L_q = L_S - S$, which equals to

$$p_0 \frac{(s\rho)^s}{s!} \sum_{n=s+1}^{\infty} (n-s)\rho^{n-s} = p_0 \frac{(s\rho)^s}{s!} \sum_{n=1}^{\infty} n\rho^n = p_0 \frac{(s\rho)^s}{s!} \cdot \frac{\rho}{(1-\rho)^2}.$$

   Observe that when $s = 1$ we recover the value obtained for a single server system.

6. The expected waiting time W_q in the queue. By Little's law, can be computed by $W_q = L_q/\lambda$, with the value of L_q found above.

Example 19.5. *Fred Smith has hired an assistant to help him in his shop when children are waiting to buy. As before, Fred can serve a customer in 30 seconds, and his new assistant is equally fast on average. As before, children enter the shop every 40 seconds, on average.In*

this example, the number of servers is $s = 2$*, the arrival rate is* $\lambda = 90$ *customers per hour, as before, and service rates are* $\mu_1 = 120$ *customers per hour if there is only one customer in the system, and* $\mu_n = 240$ *customers an hour if there are* $n \geq 2$ *customers in the system. The traffic density is* $\rho = \frac{90}{240} = \frac{3}{8} = 0.375$*. According to* (19.6)*,*

$$p_0 = \left(1 + \left[0.75 + \frac{(0.75)^2}{2!}\right] + \left[\frac{(0.75)^2}{2} \cdot \frac{0.375}{1 - 0.375}\right]\right)^{-1} = (2.2)^{-1} = \frac{5}{11}.$$

1. *The expected proportion of idle time per server is*

$$p_0 + (1 - 1/2)p_1 = p_0 + p_1/2 = p_0 \cdot 1.375 = 0.625.$$

2. *The expected number of people in the queue is*

$$L_q = \frac{5}{11} \cdot \frac{(2 \cdot 3/8)^2}{2!} \cdot \frac{3/8}{(1 - 3/8)^2} \approx 0.1227.$$

3. *The expected waiting time is*

$$W_q = \frac{L_q}{90} \approx 0.0013$$

   hours, that is, about 4.9 *seconds.*

4. *The expected time in the system is* $W_S = W_q + \frac{1}{\mu} \approx 4.9 + 30 = 34.9$ *seconds.*
   $S = 90/120 = 0.75$*.*

19.6 Arrival and/or service rates dependent on the state

In the following, we give examples of systems where the arrival and service rates depend on the number of customers in the system. In all these examples, we will assume that the queueing system has finite capacity.

19.6.1 Single server

We extend Example 19.4 as follows (here we assume that Fred is the only server). Fred Smith is becoming concerned about the volume of shoplifting going on in his shop, and decides to limit the number queueing or being served to 3. Because the shop is more attractive to these larcenous children when Fred is busy serving a child, the arrival rate increases to 120 per hour when someone is already in the shop, and stays at 90 when the shop is empty. For what fraction of his time is Fred now idle? How many customers will he expect to lose because of this strategy? How long is the queue, on average?

Here there is a "finite waiting room" situation because at most 3 children are allowed in the system (and no others are allowed to be in the shop browsing). The value of n, the number in the system, is thus one of 0, 1, 2 or 3 only. We have that $\lambda_0 = 90$, whereas $\lambda_n = 120$ children/hour, for $n = 1, 2$, but $\lambda_n = 0$ for $n \geq 3$. For every state n, the service rate is $\mu = 120$ as in Example 19.4.

We must calculate p_0, p_1, p_2 and p_3; we know that $p_n = 0$ for $n \geq 4$. By Equation 19.5.1, we have that,

$$\begin{aligned} p_1 &= \frac{\lambda_0}{\mu_1} p_0 = \frac{90}{120} p_0 = \frac{3}{4} p_0, \\ p_2 &= \frac{\lambda_0 \lambda_1}{\mu_1 \mu_2} p_0 = \frac{90 \cdot 120}{120 \cdot 120} p_0 = \frac{3}{4} p_0, \\ p_3 &= \frac{\lambda_0 \lambda_1 \lambda_2}{\mu_1 \mu_2 \mu_3} p_0 = \frac{90 \cdot 120 \cdot 120}{120 \cdot 120 \cdot 120} = \frac{3}{4} p_0. \end{aligned}$$

Since we know that the sum of the probabilities must be 1, we have

$$p_0 \left(1 + \frac{3}{4} + \frac{3}{4} + \frac{3}{4} \right) = 1,$$

which gives

$$p_0 = \frac{1}{1 + \frac{9}{4}} = \frac{4}{13}, \quad p_1 = p_2 = p_3 = \frac{3}{4} \cdot \frac{4}{13} = \frac{3}{13}.$$

Fred is now idle for just over 30 percent of his time ($p_0 = \frac{4}{13} \approx 0.31$), whereas previously he was idle for only 25 percent of his time. In one hour he expects to serve $(1 - 4/13) \cdot 120 \approx 83.08$ children. His increase in idle time is approximately equal to $(\frac{4}{13} - 0.25) = \frac{3}{52}$ of an hour each hour (approximately 3.46 minutes per hour), during which time he would expect to serve $\frac{3}{52} \cdot 120 \approx 6.92$ children. So his "limited waiting room" strategy costs him 6.92 customers per hour on average. He should ask himself how this compares with the money he loses from shoplifting.

For the last part we need to compute the expected number of children in the queue, L_q. The queue length will be 1 or 2, with probabilities p_2 and p_3 respectively. Hence

$$L_q = 1 \cdot p_2 + 2 \cdot p_3 = 1 \cdot \frac{3}{13} + 2 \cdot \frac{3}{13} = \frac{9}{13} \approx 0.6923.$$

19.6.2 Multiple servers

Previous example is modified as follows. Fred Smith has hired an assistant to help him in his shop during the busiest hour. As before, Fred can serve a customer in 30 seconds, and his new assistant is equally fast on average. Whenever the shop contains at most one child a child will usually enter after 15 seconds on average, which is faster than before because now children expect to be served promptly if there are two servers, but this rate drops to one every 30 seconds if there are two children in the shop and to only one a minute if there are already 3 in the shop. Fred will allow this extra (fourth) child to enter the shop in comparison to the single server situation, because he feels with two people at work he has a better chance of spotting a shoplifter, but this is his limit: no more than 4 children are allowed in the shop.

For what proportion of the day does Fred expect to be idle? How many customers will now be served per hour? If he makes an average of 15p gross profit per child and he pays his assistant £9 for his hour's work each school day, is Fred going to find this more profitable than when he limited the number of children to just 3 at most and didn't have an assistant?

The number of customers in the shop, n, is now between 0 and 4. When $n = 1$, we have that $\mu_1 = 120$ children per hour are served by either Fred of his assistant, but when $n = 2, 3, 4$, we have $\mu_n = 240$ per hour. The values of the arrival rates are $\lambda_0 = \lambda_1 = 240$ per hour, $\lambda_2 = 120$ per hour, $\lambda_3 = 60$ per hour, and $\lambda_n = 0$ for $n \geq 4$. Substituting into the equations (19.5.1), we have

$$\begin{aligned} p_1 &= \frac{240}{120}\, p_0 = 2\, p_0 \\ p_2 &= \frac{240 \cdot 240}{120 \cdot 240}\, p_0 = 2\, p_0 \\ p_3 &= \frac{240 \cdot 240 \cdot 120}{120 \cdot 240^2}\, p_0 = p_0 \\ p_4 &= \frac{240 \cdot 240 \cdot 120 \cdot 60}{120 \cdot 240^3}\, p_0 = \frac{1}{4}\, p_0. \end{aligned}$$

Thus $p_0(1 + 2 + 2 + 1 + 1/4) = 1$. Hence $p_0 = 4/25$, $p_1 = 8/25$, $p_2 = 8/25$, $p_3 = 4/25$, and $p_3 = 1/25$. Fred is idle whenever the number in the system is $n = 0$, and on average half the time when $n = 1$ (because when there is only one child in the shop Fred and the assistant will be equally likely to be serving the child), so that the expected proportion of time Fred is idle is $p_0 + \frac{1}{2}p_1 = 8/25$ of the hour. Hence the number of children Fred expects to serve during the hour is $(1 - 8/25) \cdot 120 \approx 81.6$ children and, since the assistant is expected to serve the same number on average, approximately 163.2 children will be served during the lunch hour.

This is a large increase on the corresponding figure, 83.08, in previous example. The expected extra 80.12 children served contribute an expected $80.12 \cdot 0.15 = £12.02$ additional gross profit. Since the wages of the assistant are only £9, Fred can expect on average that the strategy of letting up to four children in the shop will generate an extra £3.02 profit per school day peak hour.

19.7 Exercises

Exercise 19.1.

(a) Calculate the sum of the 100 numbers $3, 3.5, 4, 4.5, \ldots, 52.5$.

(b) Calculate the sum of the infinite series $5 + 5/3 + 5/9 + 5/27 + 5/81 + \cdots$

(c) Calculate the sum of the infinite series $2 + 6(1/4) + 10(1/16) + 14(1/64) + \cdots$

Exercise 19.2. A TV repairman finds that the time spent on his jobs has an exponential distribution with mean value 30 minutes. If he repairs sets in the order in which they come in, and if the arrival of sets is approximately Poisson distributed with an average rate of 10 per 8-hour day, how much idle time will he expect to have each day? What is the probability at any time that there will be two or more sets waiting to be repaired? If a set has just been brought in, how many sets on average will be ahead of it in his workshop?

Exercise 19.3. The island of St Michael's Mount near Penzance is connected to the mainland by a causeway across which it is possible to walk for a period of several hours, twice a day.

At other times the causeway is covered by the tidal water, and visitors to the island have to be transported by a motor launch, or get very wet! The launch collects a "group" of passengers (either a single individual or two or more family members or friends — a group never contains people who are strangers to each other) from the mainland quayside, takes them to the island, and disembarks them. It next collects anybody who is waiting to return, and returns to the mainland, where it disembarks the returnees (if any). If there are no people waiting to return, it leaves immediately anyway. Note that for the trip back to the mainland, the launch may carry passengers or groups who don't know one another, subject only to health and safety capacity limits, and may even include people who had walked across the causeway when the tide was out.

If there is no group waiting at the mainland quayside for a passage to the island, then the launch will wait there until one comes, even if it is clear that there are people on the island waiting to return to the mainland. On average around 10 groups arrive at the quayside per hour, wishing to use the launch, and they pay £5 per group, although this is reduced to only £2 if, on arrival, there is at least one other group in front of them at the quayside waiting for the launch. Returning groups from the island pay nothing for the privilege.

The average time for a complete cycle is 4 minutes: picking up one group at the mainland, travelling to the island, collecting returners (if appropriate), travelling back to the mainland, and disembarking any returners.

Typically the launch operates twice a day (i.e., in two shifts) for around seven and a half hours each time, until the tide next recedes sufficiently for the causeway to become accessible. The launch operators pay the boatman £75 per shift, and the daily cost of fuelling, maintaining and insuring the launch is £80. How much profit can the launch owners expect to earn per day? What assumptions have you made? How reasonable are these assumptions?

Exercise 19.4. A bank employs three tellers. Each can serve a customer in 2 minutes and 24 seconds on average. Customers arrive individually and at random at a rate of 50 per hour, following a Poisson distribution. Answer the following questions about the steady state:

(a) Calculate p_0 and the probability there are n customers in the bank (for $n \geq 1$).

(b) For what proportion of the day will each teller be busy on average?

(c) Calculate the expected number of customers in the queue.

(d) If, for security reasons, at most 5 customers are allowed in the bank at any time, calculate the percentage decrease in the expected number of customers in the queue. (Assume that customers arriving when the bank is full go away and do not return.)

Exercise 19.5. Show that, in the case $s = 2$ and with independent arrival rates,

1. $p_0 = \dfrac{1-\rho}{1+\rho}$;
2. $L_q = \dfrac{2\rho^3}{(1-\rho^2)}$;
3. $W_q = \dfrac{\rho^2}{\mu(1-\rho^2)}$;

4. $W_S = \dfrac{1}{\mu(1-\rho^2)}$;

5. $L_S = \dfrac{2\rho}{(1-\rho^2)}$.

Exercise 19.6. Lorries arrive at an unloading bay according to a Poisson process with an average rate of 1 every 40 minutes. They can be unloaded at one of 10 bays, provided the bay is in service, and the unloading time is exponential with a mean of 1 hour. It costs £20 per hour to have a lorry idle in the depot, and the cost of having any specific bay in service is £30 per hour. What is the number of bays in service that will minimise the total cost per lorry unloaded?

Exercise 19.7. Ratty is a farm dog. His master keeps him because he is very good at catching rats if they enter the hen house, where the hens lay eggs. Ratty prefers to chase a rat and play with it before killing it, so that when there is a single rat in the hen house Ratty takes 5 minutes on average to (catch and) kill it. If there are two live rats in the hen house Ratty takes on average 3 minutes to kill the first of them, but if there are more than two rats running around in the hen house Ratty gets confused and each kill takes him 10 minutes. Rats enter the hen house randomly and independently at an average rate of once every 10 minutes, but some shared instinct means that there will never be more than four live rats in the hen house at any time. On the other hand there is no way out for a rat except as a dead rat!

(a) For what fraction of his time in the hen house is Ratty able to rest?

(b) Suppose a rat has just entered the hen house and observes with horror that Ratty is in the hen house and is currently occupied with another rat. On average how long can this rat expect to have to wait before starting to receive Ratty's attentions?

Exercise 19.8. Bill and Ben run an emergency drop-in dental clinic, and share the profits from it equally. Typically they can each treat a patient in 20 minutes. When prospective patients arrive at the entrance to the clinic they are more likely to balk and leave the clinic without waiting to get treatment if the clinic appears to be relatively full. Specifically, if there are n patients already in the clinic, either receiving treatment or in the waiting room, the entry rate into the clinic becomes $10 - 2n$ patients per hour, where $n \leq 5$. Whenever $n > 5$ all prospective patients will balk. Patients who receive treatment without having to wait until either Bill or Ben is free are charged £50 per treatment, but if they have to queue, even for one second, this charge is reduced to £20. In addition, half the patients buy on average £10 worth of dental-related goods, such as toothbrushes or mouthwash, the mark-up on these goods being 100 percent. The sales of these goods are handled by the receptionist, who costs Bill and Ben a total of £100 per day in wages, national insurance and related costs. Note that the nominal operational day is six hours, but any patients still in the clinic at the end of the day will receive their completed treatment. The other costs of running the clinic, excluding Bills and Ben's fees, are £100 per day. Answer the following questions:

(a) Calculate the steady state probabilities that there are n patients in the clinic, for $0 \leq n \leq 5$.

(b) For how long in total during the nominal operational day can Bill expect to be free?

(c) On average how many patients will be treated at the clinic each day?

(d) What will be Bill's share of the daily pre-tax profit?

Chapter 20

Multi-objective optimization

In this chapter we introduce a sample of standard methods to handle multiple objectives in linear programming, as well as detection of multiple optimal solutions. All methods are demonstrated on the following ground Linear Program (LP),

$$\max\Big\{c^\top x : Ax \le b\Big\}. \tag{P}$$

where $c \in \mathbb{R}^n$, x is the variables vector, A is an $m \times n$ matrix of the coefficients of the constraints, $b \in \mathbb{R}^m$ is the right-hand-sides of the constraints.

20.1 Deciding the existence of multiple optima

When using a solver, it tells us if the problem is infeasible or unbounded. But if it returns an optimal solution, we will not know whether it is the unique one or the problem has multiple optimal solutions (unless we know so immediately from the way in which our model is built). Note that if there are many optimal solutions, then there are infinitely many of them. This is because if a and b are both optimal, then every point on the line segment connecting them, that is, for every $\lambda \in (0,1)$ the vector $\lambda a + (1-\lambda)b$ is also optimal. This is guaranteed by the feasibility of the feasible region and the linearity of the objective. In Chapter 12, we discussed the necessary (but not sufficient) condition for multiple optimal solutions. In this section, we will see how to decide the uniqueness by solving a second LP.

Linear programming formulation. Assume x^* is an optimal solution to (P) with optimal objective value $\gamma = c^\top x^*$. Then the set of optimal solutions is exactly the set of nodes x with $c^\top x = \gamma$, that is, the feasible solutions of the program

$$\Big\{x \in \mathbb{R}^n : Ax \le b,\ c^\top x = \gamma\Big\}.$$

Hence, the set optimal solutions are in correspondence with the solution of the linear program above. Observe that x^* is clearly a feasible solution; we need to decide whether it is the **unique**. For this aim, let $I \subseteq \{1,\dots,m\}$ be the subset of active constraints for x^*. Since x^*

is a basis feasible solution, x^* is the unique solution to the linear system given by

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \text{ for every } i \in I. \tag{20.1}$$

Consider the linear program over the set of optimal solutions, given by

$$\max \left\{ \sum_{i \in I} \left(b_i - \sum_{j=1}^{n} a_{ij}x_j \right) : Ax \leq b,\ c^\top x = \gamma \right\}. \tag{$\mathrm{P}_{\mathrm{opt}}$}$$

Proposition 20.1. *The point x^* is the unique optimal solution to (P) if and only if the optimal objective value to the program (P_{opt}) is 0. In particular, if it is greater than 0 its optimal solution gives a different optimal solution to (P).*

Proof. To verify this claim observe that the objective value of ($\mathrm{P}_{\mathrm{opt}}$) is always non-negative since $b_i \geq \sum_{j=1}^{n} a_{ij}x_j$ holds for every $i \in I$ and every feasible solution. On the other hand, the objective value for x^* is exactly zero, since the objective corresponds to the total slack of the set of active constraints for x^*. Therefore, the objective value of ($\mathrm{P}_{\mathrm{opt}}$) is strictly positive if and only if there exists an optimal solution $\hat{x}$ of ($\mathrm{P}_{\mathrm{opt}}$) such that there exists $i \in I$ for which $b_i > \sum_{j=1}^{n} a_{ij}\hat{x}_j$. In particular, $\hat{x}$ is also an optimal solution of (P) and it is different from x^*. □

Example 20.2. *Consider the following LP*

$$\begin{aligned} \textit{maximise } & 2x_1 + 10x_2 \\ \textit{subject to } & 2x_1 + 10x_2 \leq 61 \\ & 2x_1 + x_2 \leq 28 \\ & -x_1 + 4x_2 \leq 24.4 \\ & 4x_1 - x_2 \leq 38 \\ & 2x_1 + 3x_2 \leq 54 \\ & x_1, x_2 \geq 0. \end{aligned}$$

For this example, $x^* = (0, 6.1)$ is an optimal solution. This is certified by the dual solution $y = (1, 0, 0, 0, 0, 0, 0)$ and weak duality. We formulate now the program (P_{opt}) deciding the existence of multiple optimal solutions. The active constraints in x^* are

$$\begin{aligned} 2x_1 + 10x_2 &\leq 61 \\ -x_1 + 4x_2 &\leq 24.4 \\ -x_1 &\leq 0. \end{aligned}$$

Accordingly, the objective of this program is given by

$$\textit{maximise } (61 - 2x_1 - 10x_2) + (24.4 + x_1 - 4x_2) + (0 + x_1) = 85.4 - 14x_2$$

Solving the modified linear program returns the optimal solution $\hat{x} = (10.5, 4)$ with optimal objective value 29.4. This is other optimal solution to the original linear program.

20.2 Soft constraints

Consider the constraint $a^i x \leq b_i$ in (P), where a^i is the i-th row of A. This excludes any solution with the left hand side exceeding b_i. In some circumstances this may be realistic; for example, in a product mix this might be a quality stipulation which is legally binding; or it might represent the capacity of a piece of hardware which cannot be expanded. But in many other circumstances this may represent a capacity limitation or availability of raw material which may be increased at a price. Such constraints are sometimes called *soft constraints*. The idea is the following: We allow violation of the i-th constraint but on cost δ_i per unit, and we account it in the objective. Assume the first k constraints are soft in (P). We introduce *slack variables* $z_1, \ldots, z_k \geq 0$ representing the violation on the respective constraint, as follows

$$\max \left\{ c^\top x - \sum_{i=1}^{k} \delta_i z_i : Ax - T^k z \leq b,\ z_1, z_2, \ldots, z_k \geq 0 \right\},$$

where T^k is such that $T^k_{ii} = 1$ for every $i \in \{1, \ldots, k\}$ and zero otherwise. In some cases, we might allow violation, but only up to a certain limit M_i. This can be easily achieved by adding constraints $z_i \leq M_i$ for every $i \in \{1, \ldots, k\}$. We may analogously introduce soft $\geq$ constraints. In addition, soft equality constraints can be simulated by two soft inequalities. Assume that we have an equality constraint $a^\top x = b$, which might be violated in both directions at different penalty rates. Let δ^+ and δ^- be the penalties per unit for positive and negative violations. This can be done by introducing two variables $z^+, z^- \geq 0$, replacing the equality by

$$a^i x - z_i^+ + z_i^- = b_i,$$

and amending the objective by

$$-\delta^+ z^+ - \delta^- z^-.$$

Example 20.3 (Quiz)**.** *Verify that there is always an optimal solution with either $z^+ = 0$ or $z^- = 0$. Under what circumstances can an optimal solution contain both $z^+, z^- > 0$?*

The above demonstrate that a soft constraint is essentially a hybrid of an objective and a constraint. Next, we present an example of modelling using soft constraints. Consider the example of a company that manufactures a particular product on n machines, $M = \{1, 2, ..., n\}$, where each machine produces an end-product. Not all machines are of the latest technology, so their productivity differs. For each unit of raw material used in machine $j \in M$, a_j units of the final product are produced. The total target production is T units and, ideally this should be met exactly, as there is a penalty p per under-produced unit and a penalty q per over-produced unit. The company wishes to minimise the total penalty costs. We use x_j to denote the amount of raw material used in machine j. Further, we use non-negative variables z^- and z^+ to denote the total under-production and total over-production, respectively. The

problem is now formulated as follows

$$\begin{aligned} \text{minimise} \quad & pz^- + qz^+ \\ \text{subject to} \quad & \sum_{j \in M} a_j x_j + z^- - z^+ = T, \\ & z^- \geq 0, \\ & z^+ \geq 0, \\ & x_i \geq 0, \text{ for every } i \in M. \end{aligned}$$

20.3 Multiple objectives

Many real-world problems involve pursuing more than one objective. For instance, in designing its production schedule, a company may want to reduce costs but also reduce production time. More often than not, it is not possible to optimise all objectives simultaneously for a given problem. In other words, making optimal decisions with respect to multiple objectives involves having to trade off the objectives. For instance, a public policy making that differentially benefit different population groups will have to trade off the total benefits provided to different populations.

Multi-Objective Programming. We wish to maximise k different objectives $c^1x, c^2x, \ldots, c^kx$ over the feasible region $Ax \leq b$. This is not a well-defined problem yet, and there are several possible approaches. A simple possibility is to solve an optimisation problem for each of the k objectives and compare the solutions. This is usually inefficient: There might not be a *best* solution among them that meets our original expectations. Below we introduce a number of common approaches to multi-objective programming.

(a) **Hierarchical preferences.** Assume we have a total order over the k objectives, representing priorities of the decision taker. For example, a (fictitious) company does not compromise quality. Between options producing best quality products, they choose the one guaranteeing highest employee safety. Finally, if this can be achieved in multiple ways, they choose the cheapest option.

Formally, for every $j \in \{0, 1, \ldots, k-1\}$, we consider the objective $c^{j+1}x$ only among solutions optimal or near-optimal to the first j objectives. Assume we have flexibility terms $\varepsilon_1, \ldots, \varepsilon_k$, expressing how much we are willing to give up from the optimal value for each objective.

Then we can obtain the multi-objective solution by solving a sequence of k linear programs. The first is simply

$$\gamma_1 = \max\Big\{c^1x : Ax \leq b\Big\}.$$

For $1 < j \leq k$, assume we have already solved the first $j-1$ linear programs, with optimum values $\gamma_1, \ldots, \gamma_{j-1}$. Then the j-th linear program is given by

$$\gamma_j = \max\Big\{c^jx : Ax \leq b,\ c^\ell x \geq \gamma_\ell - \varepsilon_\ell \text{ for every } \ell \in \{1, \ldots, j-1\}\Big\}.$$

The final solution will be then given by the k-th linear program, with value γ_k.

(b) **Weighted average.** A common approach to handle multiple objectives is to create a composite objective by taking a weighted average of the k objective functions. This is applicable if we can define the importance ratio of different objectives: We assign nonnegative weights w_i for every $i \in \{1, \ldots, k\}$, such that the ratio w_i/w_j expresses the relative importance of objectives i and j. Then our problem can be written as a single linear program as follows,

$$\max \left\{ \sum_{i=1}^{k} w_i c^i x : Ax \leq b \right\}.$$

In practice, it can be difficult to find the appropriate weights. This problem is particularly acute if the different objectives are measured in different units. For instance, what should be the exchange ratio between 'cost' and 'quality'?

(c) **Max-min/min-max objectives.** Another approach is to guarantee that even the worst objective value is as good as possible. This involves maximising the minimum of the objectives. For minimisation problem we may wish to minimise the maximum of the objectives. Consider the following example. A non-profit organisation has k volunteers, who can engage in n possible teaching activities for children. There are several linear constraints on these activities, e.g. budget, availability of resources. Other linear constraints give lower bounds on the necessary amount of different combinations of activities the organisation wishes to pursue. The volunteers have different skills and capabilities and cannot be replaced by each other. To perform one hour of activity i, volunteer j needs to spend t_i^j time units on it, for every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$. Let $t^j = (t_1^j, \ldots, t_n^j)$. If we denote by $x = (x_1, \ldots, x_n)$ the vector of times needed for each activity, then the time spend by volunteer j is $t^j x$. The organisation would not like to overload any of its volunteers, and hence their objective is to minimise the maximum amount of time needed by any of them. Hence they need to solve the following optimisation problem

$$\min \left\{ \max_{j \in \{1,\ldots,k\}} t^j x : Ax \leq b \right\}. \tag{20.2}$$

Such problem can be reduced to an equivalent linear program. Let us introduce a new variable z. Then we construct a linear program as follows

$$\min \left\{ z : Ax \leq b,\ z \geq t^j x \text{ for every } j \in \{1, \ldots, k\} \right\}.$$

This linear program is equivalent to the system (20.2). Indeed, the last k constraints imply $z \geq \max\{t^1 x, t^2 x, \ldots, t^k x\}$, and since the objective is to minimise z, it cannot be strictly larger. Similarly, if we have k different objectives $c^1 x, c^2 x, \ldots, c^k x$ to be *maximised* over the feasible region $Ax \leq b$, we can solve the problem

$$\max \left\{ \min_{j \in \{1,\ldots,k\}} c^j x : Ax \leq b \right\}. \tag{20.3}$$

which can be converted to a linear program in a similar way to the example above.

(d) **Goal programming.** For each $j \in \{1, \ldots, k\}$, there is an objective $c^j x$ together with a 'target level' T_j. We wish the objective to be at least T_j, but we allow it to drop below. However, we penalize smaller objective values, on the rate of δ_j per unit violation. This can be achieved by adding a soft constraint for each objective as in Section 20.2. We introduce new variables $z_1, \ldots, z_k$ and formulate the following linear program,

$$\min \left\{ \sum_{i=1}^{k} \delta_j z_k : Ax \leq b,\ c^j x + z_j \geq T_j \text{ for every } j \in \{1, \ldots, k\} \right\}.$$

Analogously to the case in (c), other possibility is to minimise the maximum violation. This can be achieved with only one additional variable,

$$\min \left\{ z : Ax \leq b,\ c^j x + z \geq T_j \text{ for every } j \in \{1, \ldots, k\} \right\}.$$

On some of the objectives, we might wish to achieve the exact target value. For example, we wish to produce the exact ordered amount of a perishable product. This can be done by adding a soft equality constraint by introducing two new variables.

(e) **Any combination of the above.** The above methods are only a few possible approaches to deal with multi-objective problems. It might be the case that the best solution is some combination of these. For example, our first priority is to maximise the average of $c^1 x$ and $c^2 x$, the second priority is that $c^3 x$ reaches a certain minimum level, and the third priority is that the minimum of $c^4 x$ and $c^5 x$ is the largest possible. This can be formulated by combining all methods (a), (b), (c) and (d). *Quiz: formulate the above problem.*

Example 20.4. *Consider the following constraints with two objectives of maximising x_1 and maximising x_2,*

$$\begin{aligned} x_1, x_2 &\leq 6 \\ x_1 + x_2 &\leq 20 \\ 3x_1 + x_2 &\leq 20 \\ x_1, x_2 &\geq 0. \end{aligned}$$

Let us start by considering the objectives separately. Solving with only the first objective provides the optimal solution $x_A = (6, 0)$, whereas solving with only the second objective provides the optimal solution $x_B = (0, 6)$ (Note that there exist multiple optima in both cases and because of this different software may provide different optimal solutions). A decision maker would now have to consider choosing between these two solutions according to their preferences.

Now let us consider a priority order for the objectives as in (a), namely that the first is more important than the second. Solving with the first objective provides the solution x_A as before. We now restrict the solution set by adding the constraint $x_1 \geq 6 - 0.5 = 5.5$ (i.e.

flexibility term = 0.5) Then, we solve the following problem

$$\begin{aligned} \textit{maximise} \quad & x_2 \\ \textit{subject to} \quad & x_1 \geq 5.5 \\ & x_1, x_2 \leq 6 \\ & x_1 + x_2 \leq 20 \\ & 3x_1 + x_2 \leq 20 \\ & x_1, x_2 \geq 0. \end{aligned}$$

This provides a new solution $x_C = (5.5, 3.5)$, which is the best solution associated with the particular priority order for the objectives. Note here that if we had specified that the second objective is more important than the first then we would have ended up choosing a different solution, namely $x_D = (4.833, 5.5)$.

Next, let us consider composing the objectives using a weighted average as in (b). Using weights equal to 0.5 for both objectives and solving the new LP problem provides yet another solution, namely $x_E = (5, 5)$.

Now let us consider the approach of maximising the minimum of the two objectives. As in (c), the formulation is

$$\begin{aligned} \textit{maximise} \quad & z \\ \textit{subject to} \quad & z \leq x_1 \\ & z \leq x_2 \\ & x_1, x_2 \leq 6 \\ & x_1 + x_2 \leq 20 \\ & 3x_1 + x_2 \leq 20 \\ & x_1, x_2, z \geq 0, \end{aligned}$$

which has solution $x_E = (5, 5)$.

20.4 Exercises

Exercise 20.1. Consider the following LP

$$\begin{aligned} \text{minimise} \quad & 2x_2 - 5x_3 \\ \text{subject to} \quad & 2x_1 + x_2 + x_3 \leq 15 \\ & -2x_1 - 3x_2 + 4x_3 \leq -10 \\ & x_1 - x_2 + x_3 \leq 9 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

(a) Solve the problem with Excel Solver.

(b) Decide whether it has multiple optimal solutions.

Exercise 20.2. A student has 10 weeks to prepare for an examination in 3 papers. His "grade average", which he wishes to maximise, is the average of his marks (percentages) on the 3 papers, but he will fail the examination altogether if he has been less than 40% on any one paper.

He estimates that without any further study he would obtain 20% on paper I, 60% on paper II, and 30% on paper III. He also estimates that he can improve his marks by devoting time to the various papers at the rate of 5 marks per week on paper I, 10 marks per week on II, and 8 marks per week on III. But he is certain that no amount of study can raise his mark on any paper above 80%.

(a) Advise the student on how to distribute his remaining time before the exam. What is his expected grade?

(b) The rules for assessment of the exam have been tightened up by the additional restriction that the student's grade will not be more than 15 marks higher than the lowest mark of the three papers. Revise your advice to the student.

Exercise 20.3. A tourism agency organizes a tour for k travellers. There are n possible activities, and agency has to define how much time to spend with each. There are several linear constraints, e.g. budget, travel times, availability. Each tourist has a different subjective evaluations of the activities: for $1 \leq j \leq k$, and $1 \leq i \leq n$, c_i^j expresses the pleasure of 1 hour of activity i for tourist j. Spending x_i hours on activity i, the total pleasure of tourist j is thus

$$\sum_{i=1}^{n} c_i^j x_i.$$

The agency would like every single tourist to enjoy the tour: their objective is that even the least unhappy among them should be as satisfied as possible. Formulate this as a LP.

Chapter 21

Data envelopment analysis

21.1 Introduction

Data envelopment analysis (*DEA*) is a technique for comparing the efficiency of several similar *decision making units* (*DMU*s) such as local authorities, post offices, universities or schools. In particular, it is used to compare DMUs in cases where there are several inputs and several outputs. For example, when comparing comprehensive schools the inputs could be funding received from the government, proportion of children not receiving free school meals, average years of experience of the teachers, and so on; the outputs could be proportion of children receiving at least certain grades, the proportion of children from the school who go to university each year, and so on. It is not clear how to best compare several schools with very different inputs, and DEA provides us with a way of doing this.

21.2 Example

Consider the following problem of comparing the efficiency of 7 DMUs. There are two inputs and one output, as given in the table below.

DMU	1	2	3	4	5	6	7
Input 1	3	10	21	15	20	17	14
Input 2	15	14	15	21	32	7	2
Output	36	72	108	96	90	48	24

We wish to decide which of the 7 DMUs is the most efficient. We make the assumption of constant returns to scale and divisibility of resources so that, for example, DMU1 could produce an output of 12 given an input of 1 of Input 1 and 5 of Input 2. Recall that constant returns to scale means that for every input (x_1, x_2), the output $f(x_1, x_2)$ satisfies that for every $\lambda > 0$, $f(\lambda x_1, \lambda x_2) = \lambda f(x_1, x_2)$. For example, the function $c_1 x_1 + c_2 x_2$ has constant returns to scale. Other example is the classic Cobb-Douglas function, $f(x_1, x_2) = \alpha x_1^{\beta} x_2^{1-\beta}$, for some value $\beta \in (0, 1)$. By normalising the DMUs so that the outputs are equal to 12, we obtain the following modification of the table.

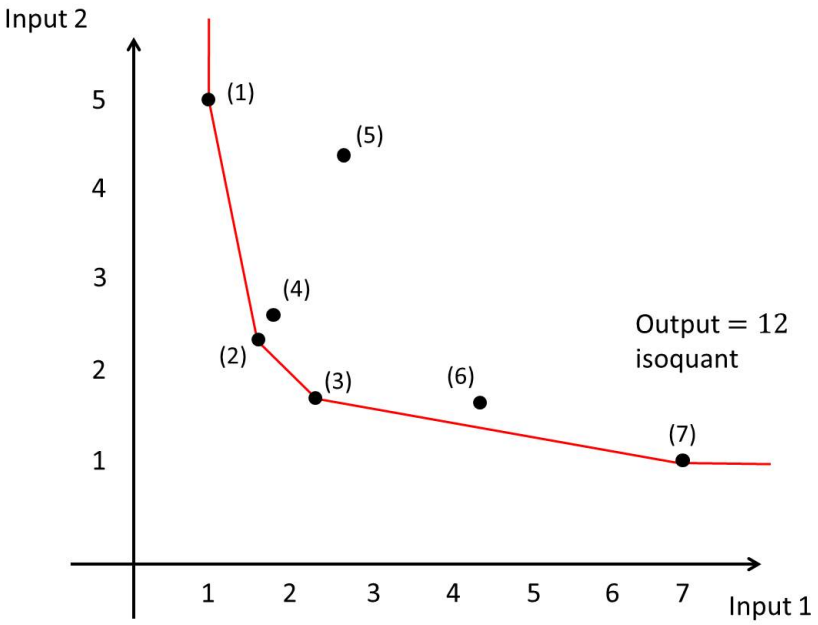


Figure 21.1: Production frontier

DMU	1	2	3	4	5	6	7
Input 1	1	1 2/3	2 1/3	1 7/8	2 2/3	4 1/4	7
Input 2	5	2 1/3	1 2/3	2 5/8	4 4/15	1 3/4	1
Output	12	12	12	12	12	12	12

We can now plot the combinations of the inputs that produce an output of 12 in two dimensions: one point for each DMU, as shown in Figure 21.1. These points are called the *isoquants* for the level of output equal to 12. The result is a piecewise linear *production frontier*. We can produce an output of 12 by any 'combination' of inputs in the convex hull of the 7 points depicted. For example, the point $(4, 3)$ can be written as

$$(4, 3) = \frac{1}{2}(1, 5) + \frac{1}{2}(7, 1).$$

We can think of this as producing an output of 12 by using a combination of DMU1 and DMU7. We are interested in the production frontier as depicted in Figure 21.1. Any DMU off this frontier is *inefficient*, as its output could be achieved by using a combination of other DMUs. For example, DMU4 is inefficient since it is dominated by DMU2. That is, DMU4 requires an input of (15/8, 21/8) to produce an output of 12 whereas DMU 2 requires an input of only $(5/3, 7/3)$ to produce an output of 12. In fact, DMU2 requires precisely 8/9 of the inputs of DMU4 to produce the same output of 12, since

$$\frac{1\frac{2}{3}}{1\frac{7}{8}} = \frac{2\frac{1}{3}}{2\frac{5}{8}} = 8/9.$$

This rather neat calculation arises from the fact that DMU2 is on a *ray* from the origin to DMU4. The number 8/9 is called the *efficiency number* of DMU4, and DMU2 is known as the *comparator* of DMU4. Thinking about this in terms of the original set-up, we observe that DMU4 produced an output of 96 from inputs of (15,21) and given the same inputs, DMU2 would produce an output of $72(3/2) = 108$.

To take another example, DMU5 is dominated by a combination of DMU1 and DMU2. To see this, note that we need $(8/3, 64/15)$ units inputs to produce an output of 12 from DMU5, but we can achieve the same output from

- 5/56 of DMU1, needing $\frac{5}{56}(1,5) = (5/56, 25/56)$ units of inputs to give $\frac{5}{56}(12) = 15/14$ units of output, plus
- 51/56 of DMU2, needing $\frac{51}{56}(5/3, 7/3) = (85/56, 17/8)$ units of inputs to give $\frac{51}{56}(12) = (153/14)$ units of output.

Therefore a total of 12 units of output is produced from $(5/56, 25/56) + (85/56, 17/8) = (1\frac{17}{28}, 2\frac{4}{7})$ of the inputs. This clearly dominates (uses less of each input) DMU5. This combination of DMU1 and DMU2 has been carefully chosen so that it lies on the 'ray' from the origin to DMU5. The efficiency number is calculated as

$$\text{Efficiency number } = \frac{1\frac{17}{28}}{2\frac{2}{3}} = \frac{2\frac{4}{7}}{4\frac{4}{15}} = 0.6027.$$

DMU1 and DMU2 are the comparators of DMU5. Again, thinking about this in term of the original set-up, it should be possible to divide the inputs of DMU5 between DMU1 and DMU2 to achieve a better output, although this time it is not clear how to do this. How can we calculate these numbers and find the results in general? The answer lies in Linear Programming.

21.3 LP approach

Given inputs a_{i1} and a_{i2} for DMUi, we wish to know how to maximise the output. If it is on the production frontier we can use DMUi alone, otherwise we may use a combination of other DMUs to achieve a higher output. Let x_i be the proportion of DMUi that we use. For example $x_3 = 1/3$ means we use 1/3 of DMU3, meaning that we use inputs of $\frac{1}{3}(21, 15) = (7, 5)$ to achieve an output of $\frac{1}{3} \cdot 108 = 36$. We wish to divide up the inputs a_{i1} and a_{i2} between all the DMUs to maximise the output. So the LP formulation is

$$\begin{aligned} \text{maximise } & 36x_1 + 72x_2 + 108x_3 + 96x_4 + 90x_5 + 48x_6 + 24x_7 \\ \text{subject to } & 3x_1 + 10x_2 + 21x_3 + 15x_4 + 20x_5 + 17x_6 + 14x_7 \leq a_{i1} \\ & 15x_1 + 14x_2 + 15x_3 + 21x_4 + 32x_5 + 7x_6 + 2x_7 \leq a_{i2} \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0. \end{aligned}$$

For example if $i = 1$ then $a_{i1} = 3$ and $a_{i2} = 15$ and we obtain the optimal solution

$$\begin{aligned} &x_1 = 1, x_2 = x_3 = \ldots = x_7 = 0, \\ &\text{Objective} = 36, \\ &\text{Dual values } 5\frac{1}{3}, 1\frac{1}{3}. \end{aligned}$$

This says the the best output we can achieve using the inputs of DMU1 is by using DMU1 itself. In other words, DMU1 is efficient. Note that for this approach we did not need to scale the DMUs to have the same output. The significance of the dual values is very important and will be discussed later. To take another example, let us turn to $i = 4$, so that $a_{i1} = 15$ and $a_{i2} = 21$. The optimal solution is

$$\begin{aligned} &x_1 = 0, x_2 = 1.5, x_3 = x_4 = \ldots = x_7 = 0 \\ &\text{Objective} = 108. \end{aligned}$$

This says we can use 1.5 of DMU2 to produce a higher output than DMU4 using the *same* inputs. So DMU4 is *inefficient*. The efficiency number is $96/108 = 0.89$ and the dual values are $3, 3$. For $i = 5$, the inputs are $a_{i1} = 20$ and $a_{i2} = 32$. The optimal solution is

$$\begin{aligned} &x_1 = 0.370, x_2 = 1.889, x_3 = x_4 = \ldots = x_7 = 0 \\ &\text{Objective} = 149.33. \end{aligned}$$

So it is more efficient to use a combination of DMU1 and DMU2 to produce a higher output (149.33) using the same inputs. The efficiency number of DMU5 is $90/149.33 = 0.603$ and the dual values are 5.33 and 1.33.

21.4 An alternative formulation (which generalises)

We now give a different but equivalent formulation which has the advantage of generalising to instances where there are several different outputs. Taking the example from the previous section, the new formulation is

$$\begin{aligned} & & \text{maximise } \lambda \\ \text{subject to } & 3x_1 + 10x_2 + 21x_3 + 15x_4 + 20x_5 + 17x_6 + 14x_7 \leq a_{i1} \\ & 15x_1 + 14x_2 + 15x_3 + 21x_4 + 32x_5 + 7x_6 + 2x_7 \leq a_{i2} \\ & 36x_1 + 72x_2 + 108x_3 + 96x_4 + 90x_5 + 48x_6 + 24x_7 \geq c_i\lambda \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0, \end{aligned}$$

where c_i is the *output* of DMUi (given its inputs are a_{i1} and a_{i2}). The variable λ can be interpreted as the proportion of the output of DMUi that can be achieved with the same inputs as DMUi but using different DMUs. We know that $\lambda = 1$ is feasible, but simply using DMUi, but if in the optimal solution $\lambda > 1$ then DMUi must be inefficient. The efficiency number is now simply given by $1/\lambda$. Solving this LP gives the same results as before.

21.4.1 A problem with 2 inputs and 2 outputs

Consider now a situation in which there are 2 inputs and 2 outputs, as given in the table below.

DMU	1	2	3	4	5	6	7
Input 1	3	10	21	15	20	17	14
Input 2	15	14	15	21	32	7	2
Output1	36	72	108	96	90	48	24
Output1	40	65	100	90	95	40	30

This cannot be formulated in the same way as our original formulation of the first example, since we do not wish to maximise just one output but two. The approach we take is to maximise the minimum of the proportion of the two outputs of DMUi that can be achieved using the inputs of the other DMUs. In other words, the linear program is

$$\begin{aligned} & \text{maximise } \lambda \\ \text{subject to } & 3x_1 + 10x_2 + 21x_3 + 15x_4 + 20x_5 + 17x_6 + 14x_7 \leq a_{i1} \\ & 15x_1 + 14x_2 + 15x_3 + 21x_4 + 32x_5 + 7x_6 + 2x_7 \leq a_{i2} \\ & 36x_1 + 72x_2 + 108x_3 + 96x_4 + 90x_5 + 48x_6 + 24x_7 \geq c_{i1}\lambda \\ & 40x_1 + 65x_2 + 100x_3 + 90x_4 + 95x_5 + 40x_6 + 30x_7 \geq c_{i2}\lambda \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0, \end{aligned}$$

where c_{i1} and c_{i2} are the outputs of DMUi (given its inputs are a_{i1} and a_{i2}). Again, $1/\lambda$ is the efficiency number of DMUi. If it is not possible to achieve more than c_{i1} units of output for DMU1, for example, then the efficiency number of DMU1 will be 1. The solution of this LP is such that DMUi is efficient for every $i \in \{1, 2, 3, 7\}$, and

$i = 4$ Inefficient
$x_2 = 1.5, w = 1.0833$
Efficiency number $= 0.923$
Dual values:
$0.0427, 0.0211$ (inputs)
$0, 0.0111$ (outputs)

$i = 5$ Inefficient
$x_1 = 0.370, x_2 = 1.889, w = 1.448$
Efficiency number $= 0.690$
Dual values:
$0.0404, 0.02$ (inputs)
$0, 0.0105$ (outputs)

$i = 6$ Inefficient
$x_3 = 0.381, x_7 = 0.643, w = 1.179$
Efficiency number $= 0.848$
Dual values:
$0.0179, 0.125$ (inputs)
$0.021, 0$ (outputs)

Remarks

1. DEA is particularly useful when inputs and outputs are measured in different units. Unlike other methods DEA allows us to compare 'chalk and cheese'.

2. For inefficient DMUs, the dual values tell us immediately which inputs would have to be lowered or which outputs would have to be lowered to improve the efficiency. For example, for $i = 5$ the largest dual value is that of the first input constraint. This tells us that if DMU5 could achieve the same output while lowering Input 1 by 1 then w would decrease by 0.0404 and the efficiency would increase from 0.690 to $1/(1.448 - 0.0404) = 0.710$.

3. To solve the problem we need to solve several LPs, one for each DMU.

4. Many DMUs will have an efficiency number of 1, i.e. they will be on the efficient frontier. This is especially true if there are many inputs and outputs.

21.5 Alternative approach

We now discuss an alternative approach to judge the efficiency of DMUs. The approach is actually equivalent to solving the *dual problem*. We continue to consider the problem in the context of our example with 2 inputs and 2 outputs. The approach is to let DMUi choose its own weights $s_1, s_2, t_1, t_2 \geq 0$ for the inputs and outputs to minimise the ratio of inputs to outputs subject to the constraint that the ratio of all other DMUs' inputs to outputs (using the same weights) cannot fall below some arbitrarily chosen level (usually 1). The formulation looks like this

$$\begin{aligned} \text{minimise } & \frac{a_{i1}s_1 + a_{i2}s_2}{c_{i1}t_1 + c_{i2}t_2} \\ \text{subject to } & \frac{a_{j1}s_1 + a_{j2}s_2}{c_{j1}t_1 + c_{j2}t_2} \geq 1, \text{ for every } j \in \{1, \ldots, m\} \setminus \{i\}, \\ & s_1, s_2, t_1, t_2 \geq 0. \end{aligned}$$

Different DMUs will choose different weights (so as to bring themselves out in the best light), but then have no grounds for disputing the relative importance of the weights. These appear to be non-linear problems, since they are the ratio of linear expressions. But actually they can each be transformed into an LP. Let $r_i = \frac{1}{c_{i1}t_1 + c_{i2}t_2}$, then we can write the formulation in terms of r_i

$$\begin{aligned} \text{minimise } & a_{i1}r_i s_1 + a_{i2}r_i s_2 \\ \text{subject to } & a_{j1}s_1 + a_{j2}s_2 \geq c_{j1}t_1 + c_{j2}t_2, \text{ for every } j \in \{1, \ldots, m\} \setminus \{i\}, \\ & c_{i1}r_i t_1 + c_{2i}r_i t_2 = 1 \\ & s_1, s_2, t_1, t_2 \geq 0. \end{aligned}$$

This still is not linear, but we can multiply the first constraint through by r_i and denote $r_i s_1$ by y_{i1}, $r_i s_2$ by y_{i2}, $r_i t_1$ by v_{i1}, and $r_i t_2$ by v_{i2}. The model then becomes

$$\begin{aligned} \text{minimise } & a_{i1}y_{i1} + a_{i2}y_{i2} \\ \text{subject to } & a_{j1}y_{i1} + a_{j2}y_{i1} \geq c_{j1}v_{i1} + c_{j2}v_{i2}, \text{ for every } j \in \{1,\ldots,m\} \setminus \{i\}, \\ & c_{i1}v_{i1} + c_{2i}v_{i2} = 1 \\ & y_{i1}, y_{i2}, v_{i1}, v_{i2} \geq 0. \end{aligned}$$

It turns out that in the solution of this version of the problem it is optimal for DMUi to choose weights equal to the dual values of the constraints in the formulation demonstrated in the previous section. For example, for DMU5, the above LP looks like:

$$\begin{aligned} \text{minimise } & 20y_{i1} + 32y_{i2} \\ \text{subject to } & 3y_{i1} + 15y_{i2} \geq 36v_{i1} + 40v_{i2} \\ & 10y_{i1} + 14y_{i2} \geq 72v_{i1} + 65v_{i2} \\ & 21y_{i1} + 15y_{i2} \geq 108v_{i1} + 100v_{i2} \\ & 15y_{i1} + 21y_{i2} \geq 96v_{i1} + 90v_{i2} \\ & 17y_{i1} + 7y_{i2} \geq 48v_{i1} + 40v_{i2} \\ & 14y_{i1} + 2y_{i2} \geq 24v_{i1} + 30v_{i2} \\ & 90v_{i1} + 95v_{i2} = 1 \\ & y_{i1}, y_{i2}, v_{i1}, v_{i2} \geq 0. \end{aligned}$$

Solving this, we obtain the solution

$$\begin{aligned} y_{i1} &= 0.0404, \\ y_{i2} &= 0.02, \\ v_{i1} &= 0, \\ v_{i2} &= 0.0105. \end{aligned}$$

This is the same as the dual values of the LP solved in the previous section. We could then convert the variables $y_{i1}, y_{i2}, v_{i1}, v_{i2}$ back to the original weights s_1, s_2, t_1, t_2.

21.6 Exercises

Exercise 21.1. A local council wishes to evaluate the efficiency of four local primary schools. The three outputs that are measured are:

- Output 1 = average reading age
- Output 2 = average mathematics level
- Output 3 = average science level.

The three inputs are

- Input 1 = average educational level of mothers (defined by age they left education)
- Input 2 = number of parent visits to school (per child)
- Input 3 = teacher-to-student ratio.

The relevant data is given in the table below

	Input 1	Input 2	Input 3	Output 1	Output 2	Output 3
School 1	18.2	4	0.05	9	7	6
School 2	19.6	5	0.05	10	8	7
School 3	15.4	6	0.06	11	7	8
School 4	21	8	0.08	9	9	9

Determine the efficiency numbers of the schools and give the weights to the inputs and outputs that the schools should choose to bring themselves out in the best light.

Chapter 22

Minimum Cost Flows

In previous chapters, we discussed a very important class of Linear Programs (LPs), which are known as *network problems*. These problems are important for various reasons. One reason is that they can be represented not only as a LP, but also as a specific mathematical object called a *graph*. Looking at problems in terms of graphs can be a helpful way of analysing them and often provides a fresh perspective on the problem. Moreover, because of their mathematical structure, network problems can be solved much faster than general LPs. Therefore, from a practical perspective, the modeler is in a very convenient situation if it is possible to represent a problem as a network problem. Finally, network problems are guaranteed to have solutions that are integer if the right-hand sides of the constraints are integer (flow integrality property). For practical applications this can be very important, i.e. when fractional solutions do not make sense. In this chapter, we continue the discussions on LP formulations for network problem.

Suppose we are modeling the distribution of a single homogeneous product from factories, or *supply* nodes, to customers, or *demand* nodes. The amounts available at each factory and the amounts required by the customers are known. The product is sent to customers via intermediary points (warehouses or distribution centres). There are capacity limitations on some of the transportation links. The objective is to minimise the total cost of meeting customer demands. This situation corresponds to the most general network flow problem, referred as *minimum cost flow problem.*

LP formulation. Let $G = (N, A)$ be a directed graph where N is the set of vertices, or nodes, and A the set of arcs. Let $S, D \subseteq N$ be the subsets of N containing the supply and demand vertices respectively. Let a_i denote the available supply at vertex $i \in S$ and b_i the demand at vertex $i \in D$, c_{ij} denote the cost per unit of flow travelling on arc (i, j), l_{ij} and u_{ij} respectively denote the lower and upper capacity bounds for arc (i, j). The decision variable is defined as x_{ij} to capture the amount of flow on arc (i, j). Then, we can find a minimum

cost flow by solving the following program.

$$
\begin{aligned}
\text{minimise} \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{subject to} \quad & \sum_{j:(i,j)\in A} x_{ji} - \sum_{j:(j,i)\in A} x_{ij} \le a_i, & & \text{for every } i \in S, \\
& \sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0, & & \text{for every } i \in N\backslash(S \cup D), \\
& \sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} \ge b_i, & & \text{for every } i \in D, \\
& l_{ij} \le x_{ij} \le u_{ij}, & & \text{for every } (i,j) \in A.
\end{aligned}
$$

The objective is to minimise the total cost of the flows, which is the sum of the flows multiplied by the costs on the arcs. There are three types of flow-balance constraints: for the supply vertices, the net out-flow must be at most the supplies; for the demand vertices, the net in-flow must be at least the demands; and for the other vertices, the net out-flow must be equal to the net in-flow. The last set of constraints are the upper and lower bounds on the flows. The number of variables is the number of arcs; the number of constraints is the number of vertices. Each variable has 2 non-zero elements in its column, one $+1$ and one -1. The constraint matrix is actually the *incidence matrix* of the network: the non-zero elements in the column of arc (i, j) are -1 in the row vertex i and $+1$ in the row vertex j. The rows of the matrix correspond to the vertices and the columns to the arcs. Cell $\{i, e\}$ of the incidence matrix is -1 if vertex i is the origin of arc e; $+1$ if vertex i is the destination of arc e; 0 if vertex i is not incident to arc e. The graph in Figure 22.1 is represented by the incidence matrix given in the table below.

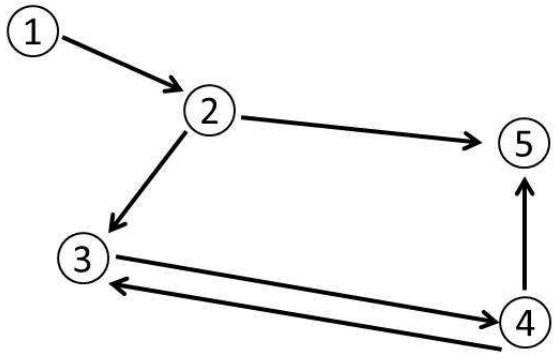


Figure 22.1: An example of directed graph

vertex/arc	(1,2)	(2,3)	(2,5)	(3,4)	(4,3)	(4,5)
1	-1	0	0	0	0	0
2	+1	-1	-1	0	0	0
3	0	+1	0	-1	+1	0
4	0	0	0	+1	-1	-1
5	0	0	+1	0	0	+1

22.1 Maximum Flow Problem

In this problem we wish to send as much flow as possible from a specified supply vertex s to a specified destination vertex t. There are no costs on the arcs of the network but there are capacities. We can think of this as a problem of trying to maximise the amount of some commodity shipped from a factory to a store, for example. We can model it as a min-cost flow problem by adding an artificial edge from the sink to the source, and maximising the flow value on that edge. The general LP formulation is as follows,

$$\begin{aligned} &\text{maximise} && x_{ts} \\ &\text{subject to} && \sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0, \quad \text{for every } i \in N, \\ & && l_{ij} \le x_{ij} \le u_{ij}, \ \text{for every } (i,j) \in A. \end{aligned}$$

In the example in Figure 22.2, we wish to find the maximum flow from vertex 1 to vertex 6. The values associated with the solid arcs are the capacities. The dashed arc is introduced to the graph to model this as a maximal flow problem. Again there are more efficient ways of

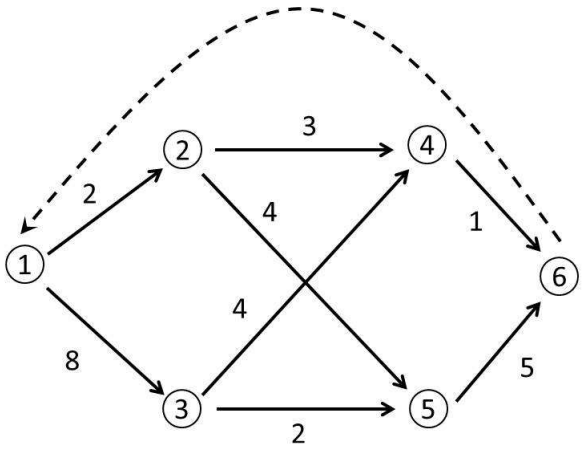


Figure 22.2:

solving maximal flow problems, such as the algorithms described by Ford and Fulkerson [2].

22.2 The Assignment Problem

Consider the problem of assigning n workers to n jobs so as to minimise the total time required to complete the jobs. We assume worker i takes time t_{ij} to complete job j, and each worker is assigned to precisely one job. Define variables x_{ij} for every $i, j \in \{1, \ldots, n\}$, where

$$x_{ij} = \begin{cases} 1 & \text{if worker } i \text{ is assigned to job } j, \\ 0 & \text{otherwise.} \end{cases}$$

Then we can formulate the problem as the following 0/1 program,

$$\begin{aligned} \text{minimise} \quad & \sum_{i,j=1}^{n} t_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1, \text{ for every } i \in \{1, \ldots, n\}, \\ & \sum_{i=1}^{n} x_{ij} = 1, \text{ for every } j \in \{1, \ldots, n\}. \\ & x_{ij} \in \{0, 1\}, \text{ for every } i, j \in \{1, \ldots, n\}. \end{aligned}$$

Alternatively, we can consider this as a transportation problem. We create n sources and n sinks corresponding to n workers and n jobs respectively. Then we create an arc going from each worker to each job whose cost is the time it takes for that worker to complete that job (Figure 22.3). The supply of each worker and the demand of each job is set as 1. Due to the integrality property of the transportation problem, we can relax the integrality and solve instead the relaxation for this problem. The flow integrality theorem guarantees the existence of an optimal integral solution.

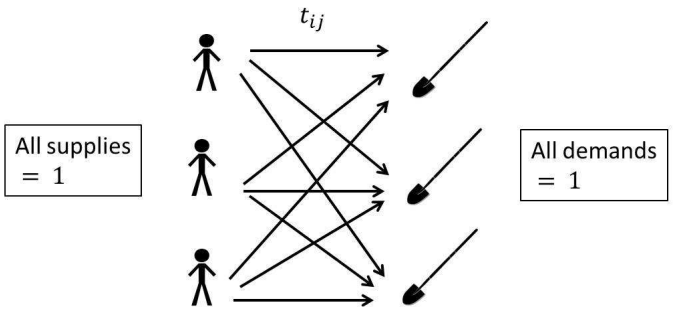


Figure 22.3: Assignment problem

22.3 Critical path analysis

In Chapter 3 we studied the critical path analysis (CPA) and described the CPA algorithm. We now formulate this problem as a special case of the network flow problem. We illustrate

the concept of critical path analysis with an example (see Williams [3]). Consider a project of building a house. There are several tasks that must be completed, and these tasks are represented by the arcs in Figure 22.4. The numbers on the arcs correspond to the time taken to complete the task and the vertices represent the beginnings and the ends of the tasks. The structure of the network represents the dependencies of the tasks, so for example the walls cannot be built until the foundations have been laid and the bricks have been obtained. There is a dummy arc from vertex 4 to vertex 2 to represent the fact that the roofing cannot be undertaken until the walls have been built. Our aim is to minimise the total amount of time

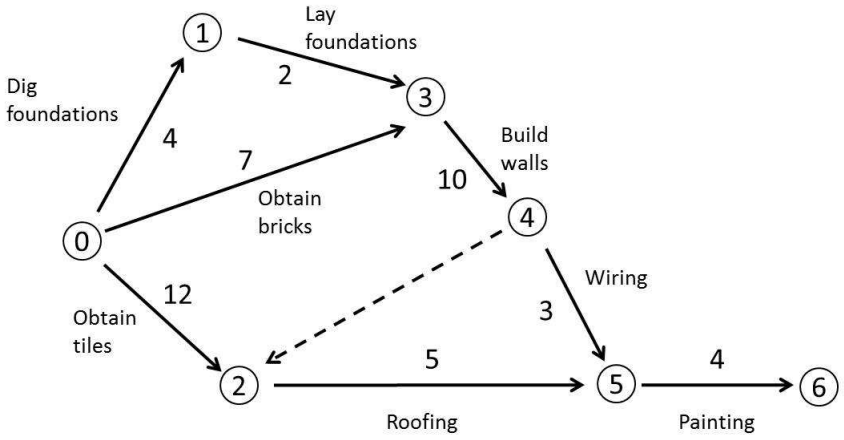


Figure 22.4: Critical path analysis of house building

it takes to complete the project, and with this in mind we define variable t_i as the start time for the activities starting at vertex $i \in \{0, 1, \ldots, 5\}$. We also define z as the finish time of the whole project. The LP formulation is then formulated as follows

$$\begin{aligned} & \text{minimise } z \\ \text{subject to } & -t_0 + t_1 \geq 4 \\ & -t_0 + t_2 \geq 12 \\ & -t_0 + t_3 \geq 7 \\ & -t_1 + t_3 \geq 2 \\ & -t_3 + t_4 \geq 10 \\ & -t_4 + t_2 \geq 0 \\ & -t_2 + t_5 \geq 5 \\ & -t_4 + t_5 \geq 3 \\ & -t_5 + z \geq 4. \end{aligned}$$

The constraints arise from the dependencies of the project. For example, since the walls cannot be built until the foundations have been laid, we must have $t_3 \geq t_1 + 2$. The whole project cannot be completed until the painting has been finished, so $z \geq 4 + t_5$. We assume

that the house building starts at day 0, then $t_0 = 0$. By solving the LP, we obtain the solution

$$\begin{aligned} \text{Project completion time } z &= 26 \text{ days}, \\ t_1 &= 4, \\ t_2 &= 17, \\ t_3 &= 7, \\ t_4 &= 17, \\ t_5 &= 22. \end{aligned}$$

Which activities can and cannot be delayed without delaying the whole project, and by how long? The answer to this question can be found by examining the solution to see which constraints are active. We see that of the 9 constraints, the ones that are active are numbers 1, 3, 5, 6, 7 and 9. Therefore the activities that cannot be delayed are all of them except digging the foundations and laying the foundations, which can be delayed by a total of 1 day. In addition, we may also delay obtaining tiles and wiring. The activities that cannot be delayed lie along a *critical path*, which is 0-3-4-2-5-6. It can be shown that the activities that cannot be delayed always lie along some critical path in problems like this, and that this critical path is the longest path through the network.

22.4 Exercises

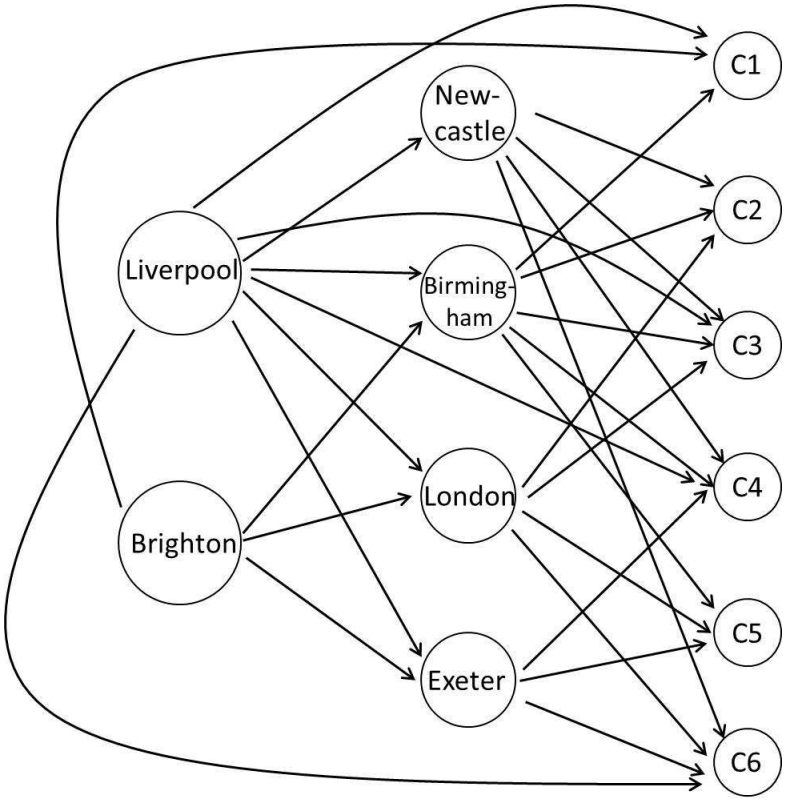
Exercise 22.1. A firm producing fashion goods at two plants close to Shanghai and Guangzhou, respectively, each with a maximum capacity of 1100 cases (collections) per week, has a contract to make regular weekly deliveries of 500 cases to New York, 400 to Berlin, 1000 to London, and 300 to Paris. Cost of transport by sea in £s per case is:

	Guangzhou	London	Paris	New York	Berlin	Rotterdam
Shanghai	2.00	6.00	5.40			
Guangzhou		5.00		10.00		4.00
London				8.00		
Rotterdam				5.60	2.00	

(a) Model the problem, implement in Excel Solver, and find the minimum cost pattern of transport, using these facilities only.

(b) China Southern has entered the Skyteam alliance, and as a result air-freight rates have been reduced. A sales manager draws the headquarter's attention to the fact that the four weeks average delivery period from China to New York and the two and a half weeks from China to Europe is costing the firm money in the capital tied up in the 'pipeline'. The goods are worth £2000 per case, the rate of interest on capital is 0.2% per week, and air-freight rates from Guangzhou to New York and Rotterdam (including carbon off-setting) are £30.00 and £20.00 per case, respectively. Should the firm use air-freight, and, if so, how should it re-arrange its deliveries? (Assume that air-freight and all routes within Europe take zero time, while the sea transport from Europe to New York takes one and a half weeks.)

Exercise 22.2. A company has two factories, one at Liverpool and one at Brighton. In addition it has four depots with storage facilities at Newcastle, Birmingham, London and Exeter. The company sells its products to six customers C1, C2,. . . ,C6 (see the diagram below). Customers can be supplied from either a depot or from the factory directly.



The distribution costs, in £s per tonne delivered, are known; they are in given in the tables below where a dash indicates that a supplier cannot supply a particular customer or depot.

Factories\ Depots	Newcastle	Birmingham	London	Exeter
Liverpool	1	1	2	0.4
Brighton	-	0.6	1	0.4

Factories and depots\ Customers	C1	C2	C3	C4	C5	C6
Liverpool	1	-	3	4	-	2
Brighton	4	-	-	-	-	-
Newcastle	-	3	1	3	-	2
Birmingham	2	1	1	2	1	-
London	-	3	3	-	1	3
Exeter	-	-	-	3	1	3

Each factory has a monthly capacity given below which cannot be exceeded:

Factory	Liverpool	Brighton
Capacity (1000s of tonnes)	150	200

Each depot has a maximum monthly throughput given below which cannot be exceeded:

Depot	Newcastle	Birmingham	London	Exeter
Max. throughput (1000s of tonnes)	70	50	100	40

Each customer has a monthly requirement below which must be met:

Customer	C1	C2	C3	C4	C5	C6
Monthly requirement (1000s of tonnes)	50	10	40	35	60	20

The company would like to determine:

(a) What distribution pattern would minimise total cost?

(b) What would be the effect of increasing factory and depot capacities on distribution costs?

(c) Certain customers have expressed preferences for being supplied from factories or depots to which they are accustomed. The preferred suppliers are:

Customer	Preferred factory/depot
C1	Liverpool (factory)
C2	Newcastle (depot)
C5	Birmingham (depot)
C6	Exeter or London (depots)

Is it possible to meet all customers' preferences regarding suppliers and if so what is the extra cost of doing so?

Bibliography

[1] Dijkstra, E. W. (1959) A note on two problems in connection with graphs. *Numerische Mathematik* 1 269-271.

[2] Ford, L. W and Fulkerson, D. R. (1962) *Flows in networks*, Princeton University Press, Princeton, NJ.

[3] William, H. P. (2013) *Model building in mathematical programming* 5th edition, Wiley.

Chapter 23

Integer programming

23.1 Introduction

A strong limitation in the applicability of linear programming is that the variables can take arbitrarily fractional values in an optimal solution. In most practical situations however, many quantities are only allowed to take discrete values. For example, it usually does not make sense to return a fractional answer, if a company has to decide about the number of storage facilities to open, or about the number of vehicles to purchase. Such problems can be formulated as *Integer Programs (IP)*. The problem has several variants. By pure *Integer Program*, we mean a program with all variables restricted to be integer

$$\max\left\{c^\top x : Ax \leq b,\ x \in \mathbb{Z}^n\right\}.$$

In practice, we usually have a mixture of continuous and discrete variables. This is called *Mixed Integer Program* (MIP). In this case, we can partition the variables in two parts and write the formulation as

$$\max\left\{c_1^\top x_1 + c_2^\top x_2 : Ax \leq b,\ x_1 \in \mathbb{R}^{n_1},\ x_2 \in \mathbb{Z}^{n_2}\right\}.$$

There is an important trade-off between LP and IP. LP is less powerful for modelling as we cannot enforce discrete values in the solution, that can be done in IP. On the other hand, there are highly efficient algorithms for LP. There are several algorithms for LP that are not only efficient in practice, but also have theoretical guarantees that they terminate rapidly in terms of the input problem size. In contrast, there are no such guarantees at all for IP (it is a hard problem in a rigorous mathematical sense of hardness). Whereas several algorithms perform reasonably well in practice, they are slower than LP algorithms often by orders of magnitudes, and moreover, we do not have theoretical guarantees that they terminate in a reasonable amount of computing time. We can compare LP to a *Swiss army knife* due to its simplicity and efficiency. Rather than a Swiss army knife, an IP solver could be compared to a contractor with a large toolbox and years of practical experience - yet we might still not be completely sure that he is able to repair our household problem. Moreover, his work is much more expensive than to fix it ourselves. While formulating models is simple for IP, the solution methods are far from simple. They are built on decades of extensive research

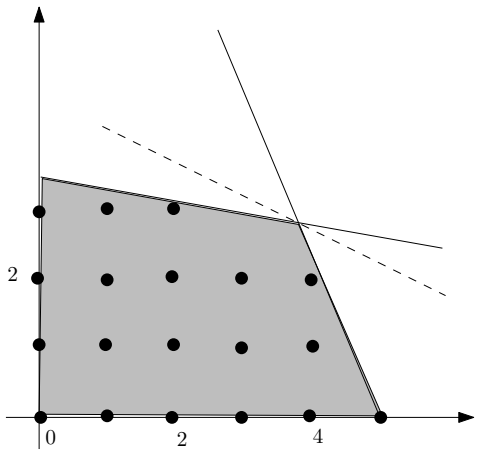


Figure 23.1: An IP problem and its LP relaxation

and expertise. For this reason, we should always save with the integer variables: Use them only if really needed and try to keep their number and usage limited. The most fundamental techniques for solving IPs include *Branch & Bound* method and *Cutting Planes* method. We give a brief overview of Branch & Bound method in this chapter.

23.2 Solving IP

In this section, we briefly describe the fundamental Branch & Bound method for solving IPs. Let us start with a simple example.

$$\begin{aligned} \text{maximise} \quad & 2x_1 + 5x_2 \\ \text{subject to} \quad & 12x_1 + 5x_2 \leq 60, \\ & 2x_1 + 10x_2 \leq 35, \\ & x_1, x_2 \geq 0 \quad \text{and integer.} \end{aligned}$$

The feasible region is illustrated in Figure 23.1, where the dots are the feasible integer points. The LP optimal solution is at point (3.864, 2.727) with the objective equal to 21.364. Visually we can see that the possible candidate optimum points are $(2, 3)$ with objective value 19, and $(4, 2)$ with objective value 18 - hence the optimal solution is $x_1 = 2$, $x_2 = 3$.

For LPs, the solution method is based on the fact that every local optimal solution is also a global one. This is not true for IPs. For example, point $(4, 2)$ does not have any integer neighbours that would be feasible and also give a better objective value - yet it is not optimal. For this reason, we cannot use dual solutions to verify optimality any longer. There is no simple certificate to convince your boss that you have found an optimal solution.

23.2.1 Linear programming relaxation

The difference between a LP problem and an IP problem consists in the fact that the IP has additional constraints, namely the integrality constraints. Problem Y that is derived

from problem X by not taking into account some of the latter constraints is called a *relaxation* of problem X. When we disregard the integrality constraints of an IP, we arrive at a normal LP. Therefore, such a LP problem is called the linear programming relaxation of the original IP. As the linear programming relaxation has a larger feasible region than the original IP, the objective function value of an IP can never be better than that of its linear programming relaxation. In other words: For a maximisation (minimisation) IP problem, its linear programming relaxation provides an upper (lower) bound for the optimal solution. In Figure 23.1, the grey region is feasible region of the linear programming relaxation, and 21.364 is the optimum of the relaxed problem, as compared to the integer optimum 19.

Given the fractional optimal solution, we might try rounding it to the closest integer solution. However, this might be infeasible, or feasible but not optimal. In the above example, the fractional optimal solution was $(3.864, 2.727)$. Rounding it would give $(4, 3)$, that is infeasible as in the second constraint we get $38 \leq 35$. But even if the rounded solution is feasible, it might not be optimal. Yet sometimes we might get an acceptable solution by rounding. This is possible if

- The rounded solution is feasible, *and*
- The difference between the optimal objective value of the LP relaxation and the objective function value of the rounded feasible solution is acceptable for the practical purpose that the model serves.

Rounding is typically useful if we have integer variables that can take large values. However, rounding binary variables we usually get infeasible or very bad solutions.

23.2.2 The Branch & Bound algorithm

A lot of research has been and is being applied to methods to solve IPs. The most successful approach is *Branch & Bound*, developed in 1960 by Ailsa Land and Alison Doig, at the LSE Operational Research Department (the predecessor of the OR Group). All the commercial mathematical programming systems offer an integer programming solution procedure based on some variation of this method.

Branch & Bound can be thought of as a systematic exploration of the feasible region and an elimination of those parts which can be shown either not to contain an integer solution or not to contain the optimal solution. Branch & Bound algorithm is a divide-and-conquer approach: the feasible region is partitioned into a collection of smaller regions. Each region is itself represented by linear constraints and the objective function can be optimised over these smaller regions. The partitions of the feasible region can themselves be further partitioned into even smaller regions. A bound on the value of the objective function, for a partition and all its sub-partitions, is the optimal value of the objective function over the feasible region defined by the partition. In practice the partitioning of the feasible region is performed sequentially - that is the branching operation. The bounding operation determines the choice of partition on which to perform the branching operation. The algorithm is best represented by a tree graph.

Let us return to the example we considered earlier. The algorithm starts by solving the linear programming relaxation. This corresponds to the *root node* of the *Branch & Bound*

tree. If the solution satisfies the integer condition we stop, otherwise it is necessary to branch. Select a variable whose value is non-integer and create two new sub-problems. The constraints of the new sub-problems are chosen so that the current non-integer solution is feasible in neither sub-problem. In the example, the solution at the root node is: $x_1 = 3.864$, $x_2 = 2.727$. We arbitrarily choose to branch on x_1. This is done by creating two sub-problems: one with the extra constraint $x_1 \leq 3$ and the other with $x_1 \geq 4$ (Figure 23.2(a)). The two subproblems are shown in Figure 23.2(b).

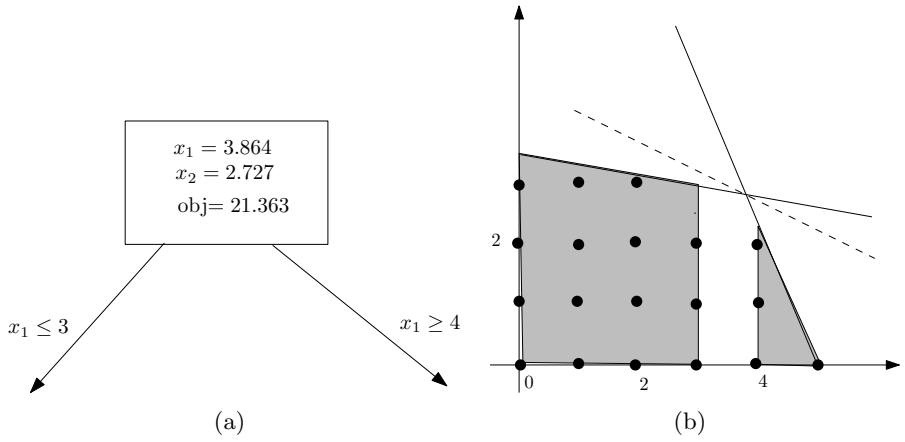


Figure 23.2: Branch & Bound: branch on the root node

We choose (arbitrarily) to solve the left-hand problem first. This gives a solution $x_1 = 3$ and $x_2 = 2.9$ with objective value 20.5. Notice that because an additional constraint, $x_1 \leq 3$, has been added, the value of the objective function either stays the same or decreases. As x_2 is non-integer two further sub-problems with the constraints $x_2 \leq 2$ and $x_2 \geq 3$ are created (Figure 23.3). There are now three unsolved sub-problems.

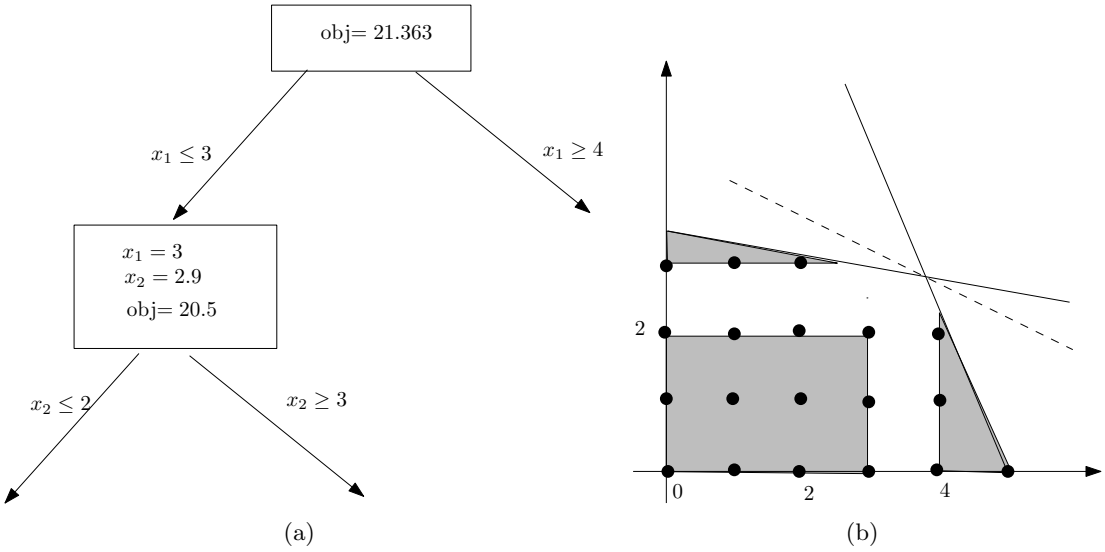


Figure 23.3: Branch & Bound: second branch

If we now solve the leftmost problem again this gives an integer solution $x_1 = 3$, $x_2 = 2$ with objective value 16. This solution becomes *incumbent* solution, i.e. the best known so far. The branch is said to be fathomed by integrality and now the algorithm backtracks and selects one of the unsolved sub-problems for solution. Arbitrarily choosing the most recently formed unsolved sub-problem and solving it gives the tree in Figure 23.4.

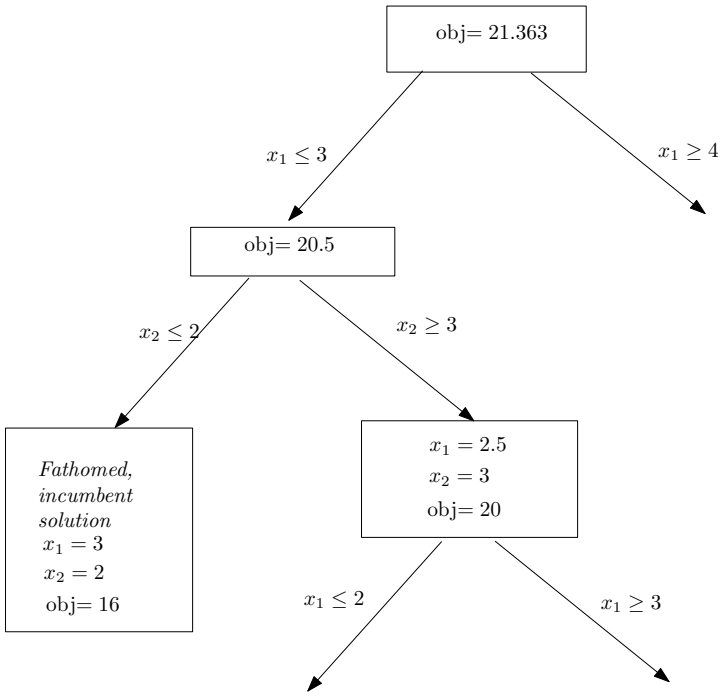


Figure 23.4: Branch & Bound: fathom by integrality

Continuing branching and solving gives the tree on Figure 23.5. A better integer solution $x_1 = 2$, $x_2 = 3$ is found with objective value 19. Replacing $(3, 2)$, this becomes the new incumbent solution. The branch $x_2 \geq 4$ is *fathomed by infeasibility*: adding $x_1 \leq 3$ and $x_2 \geq 4$ to the system, there is not any feasible solution to the LP.

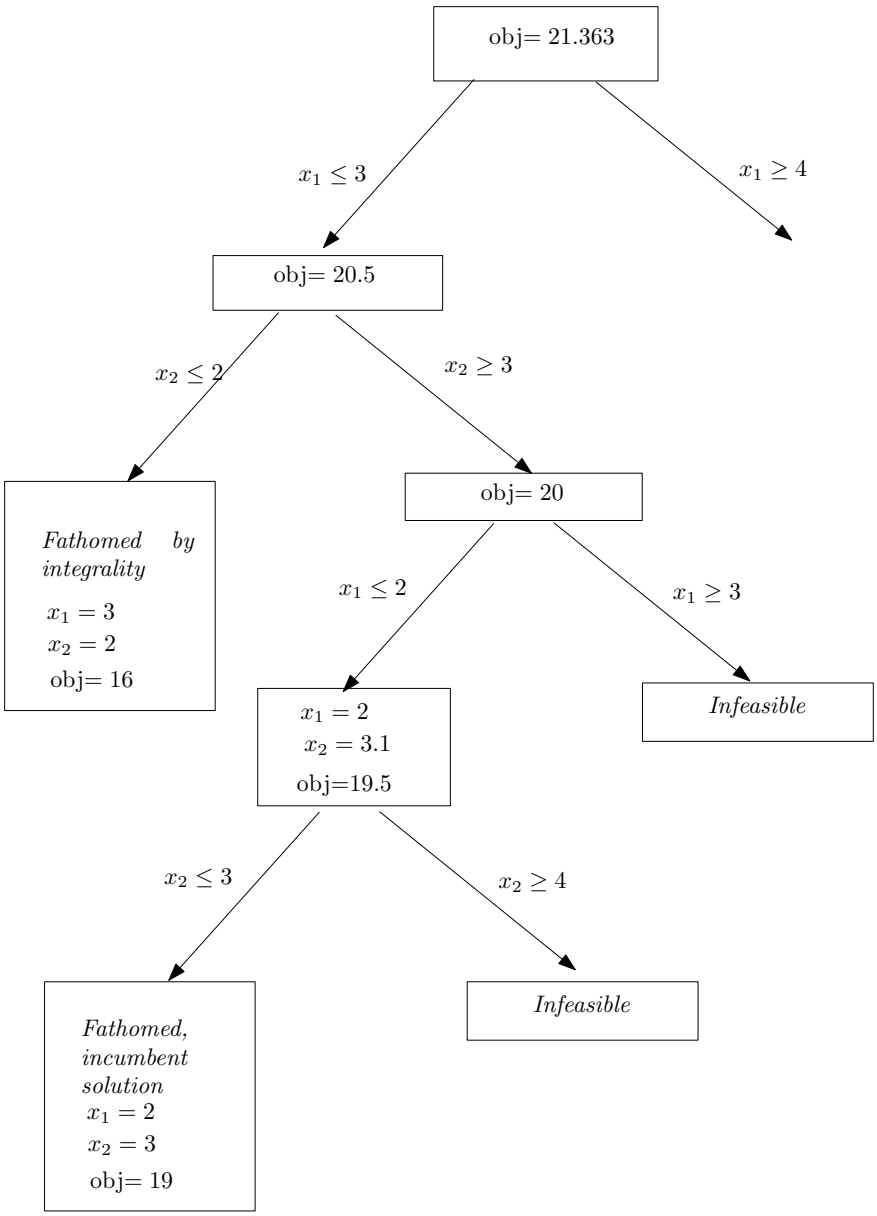


Figure 23.5: Branch & Bound: fathom by infeasibility

At this point, the entire branch of the tree where $x_1 \leq 3$ is completely fathomed. So we backtrack to the very beginning and solving the sub-problem with the single extra constraint $x_1 \geq 4$. Continuing in the same fashion as before gives the tree below and subsequently branching gives the tree on Figure 23.6. At the sub-problem with $x_1 \geq 4$, $x_2 \leq 2$, the LP

optimum value is 18.3. As extra constraints are added to form the sub-problems, the value of the objective function cannot increase, it either stays the same of decreases. Consequently, in all later sub-problems on this branch we get solutions with objective value less than or equal to 18.3. However we have a better incumbent solution of value 19. Consequently, there is no point in continuing to branch. We say that the sub-problem $x_1 \geq 4$, $x_2 \leq 2$ is *fathomed by bound*. The incumbent, and thus optimal solution to the problem at the end of the entire Branch & Bound search is $x_1 = 2$, $x_2 = 3$ with an objective function value of 19.

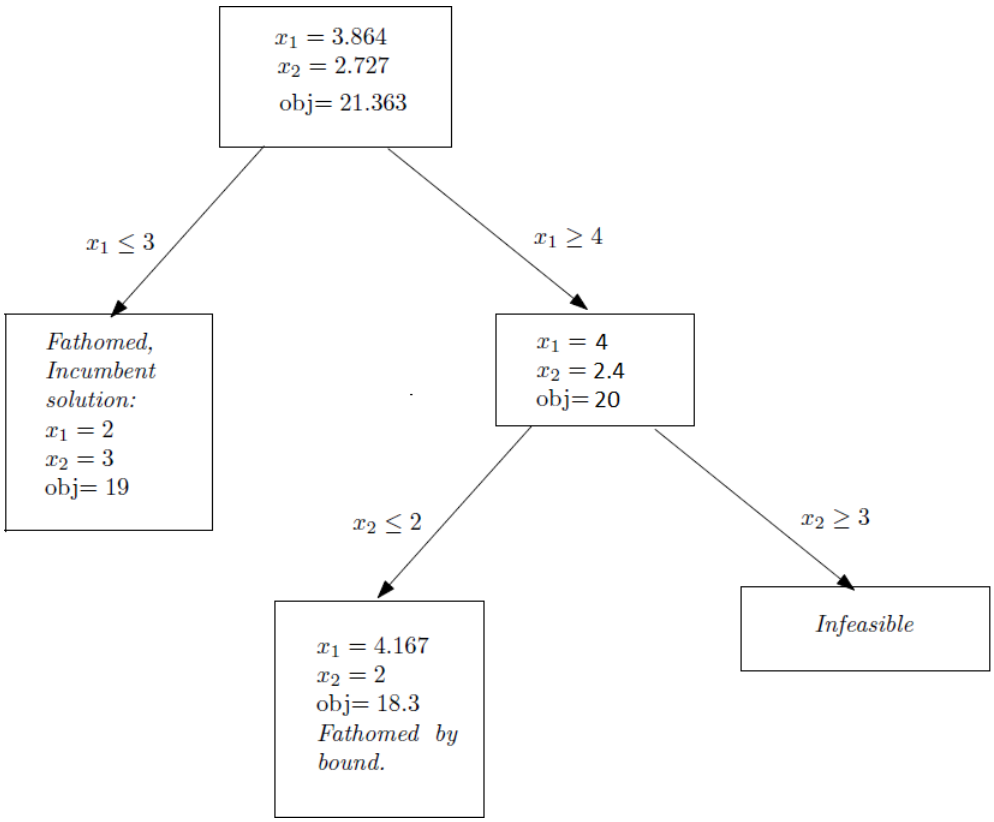


Figure 23.6: Branch & Bound: completed

Notice that in order to solve this problem in 2 variables it was necessary to solve 11 linear programs. In general, the number of steps may blow up exponentially: even if we have only k binary variables, we might end up with 2^k different cases. It requires substantially more work to solve an integer program than a linear program - in some cases, it takes prohibitively long. That is why great care should be taken in setting up an integer program - is there possibly a way to formulate the problem in a more economic way, with fewer integer variables?

23.3 Binary variables

As an important special case, we can have *binary variables* with only two possible values, 0 and 1. These are also called *indicator variables* as they can model *yes-no* decisions, indicating

the choice of certain resources. E.g. $x_s = 1$ expresses the fact that we open a certain storage facility s, and $x_s = 0$ means that we do not open it. In this section, we illustrate typical usage of binary variables in modelling by giving a number of examples.

23.3.1 The big-M method

Let z represent the quantity of product to be manufactured. The cost per unit of production is c. However, this is just the variable production cost. If any of the product is manufactured a fixed cost is incurred which is f. That is

$$\text{production cost} = \begin{cases} 0 & \text{if } z = 0 \\ f + cx & \text{if } z > 0. \end{cases}$$

Whereas this cost function might seem linear, it is *not*. (Setting the cost to be $f + cz$ everywhere would be linear, however, it would give f for $z = 0$ instead of 0.) We will need a binary variable to express such a cost function. We wish to minimise the total cost. To model this situation we introduce a new binary $(0 - 1)$ variable δ which we will use to represent the logical states: *(a)* no product is produced; *(b)* some amount $z > 0$ is produced. In other words we need to model the following situation

$$\text{if } z > 0 \text{ then } \delta = 1. \tag{23.1}$$

To model this logical condition it is necessary to introduce the maximum amount of the product that can be produced, that is the maximum value of x. Let this be M (we do not need the exact maximum value, only an upper bound). We now model this situation by introducing the following constraint

$$z \leq M\delta. \tag{23.2}$$

Clearly, if $z > 0$ (i.e. some production is undertaken), then δ is forced to 1 ($z > 0$ and $\delta = 0$ would give the contradiction $0 < z \leq 0$). Thus we can now write the total cost (to be minimised) as

$$\text{production cost} = f\delta + cz.$$

Note that because we are minimising cost, the objective will be trying to set $\delta = 0$ if this is possible, so that we don't have to worry about fixed cost being incurred when no production is undertaken, i.e. when $z = 0$ then the minimisation will set $\delta = 0$.

This is a widely applicable modelling strategy, often referred to as the "big-M method". We can introduce a binary variable δ to model the relationship (23.1), provided an upper bound M on the possible values of z. Note that the inequality (23.2) does not directly imply that $\delta = 0$ whenever $z = 0$. This is achieved only because we want to minimise $f\delta$ in the objective. Sometimes we may need to formulate this converse implication directly, as illustrated by the next example.

23.3.2 A blending problem - the small-m method

We have to blend three ingredients, A, B and C. The extra condition is: if A is included in the blend then B must also be included. Notice that the blend could be made entirely of ingredient

B. Let x_A, x_B and x_C denote the proportions of the ingredients. The maximum value of all three variables is 1 (that is 100%). The minimum proportion at which B is considered to be present in the blend is $m = 0.01$. We may model this situation by introducing a binary variable δ indicating whether A is in the blend or not. The situation is now modelled as follows.

$$\begin{aligned} x_A &\leq \delta, \\ x_B &\geq m\delta, \\ x_A + x_B + x_C &= 1, \\ x_A, x_B, x_C &\geq 0, \\ \delta &\in \{0, 1\}. \end{aligned}$$

If $\delta = 0$ this forces $x_A = 0$, i.e. the blend will be made entirely of products B and C. On the other hand if $\delta = 1$, this forces $x_B \geq m$, i.e. the proportion of B will be at least $m = 0.01$. Such a formulation is called the small-m method. For a variable z ($z = x_B$ in the example), we aim to express the converse of (23.1), that is

$$\text{if } \delta = 1 \text{ then } z > 0.$$

However, this cannot be done with full precision. Analogously to the upper bound M in the previous method, we need a lower bound m which expresses the intended minimal positive value of z. That is, every value $0 \leq z \leq m$ is neglected and considered as 0 in effect. Hence we use the inequality

$$z \geq m\delta.$$

23.4 Exercises

Exercise 23.1. Consider the following IP

$$\begin{aligned} \text{maximise} \quad & x_1 + 0.64x_2 \\ \text{subject to} \quad & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x_1, x_2 \geq 0 \quad \text{and integer.} \end{aligned}$$

(a) Solve the linear programming relaxation using Excel Solver. If you round the solution, is it feasible?

(b) Solve the IP by performing a Branch & Bound search *manually*, i.e. add appropriate constraints to the linear programming relaxation and solve the resulting LPs with Excel Solver.

Exercise 23.2. Bruce is going on holiday to Hawaii, and to save money he is only taking hand luggage. He can take a maximum of M cubic centimetres of hand luggage onto his flight. He has n different items he might need on his trip, indexed $i = 1, \ldots, n$. He attaches a value of u_i to a unit of item i, and the space taken by this item would be s_i cubic centimetres. He owns m_i units of item i - it is possible to take more than one unit.

Help Bruce to decide which items to take in order to maximise the total value of the luggage contents by formulating an IP. Extend this formulation by incorporating the following additional constraints.

(a) Some items are fragile, let $F \subset \{1, \ldots, n\}$ denote their set. The total amount of space where fragile items can be stored securely is at most f; however, this can also be used to store robust (non-fragile) items as well.

(b) C is the set of clothing items. The number of clothes Bruce needs is at least c.

(c) Bruce owns two flippers, a snorkel, a pair of goggles and a pair of swimming trunks. These are items 1, 2, 3 and 4 respectively (so $m_1 = 2$, $m_2 = m_3 = m_4 = 1$). Bruce would need all these items to go snorkelling but if he only brings his swimming trunks he can still go swimming.

Chapter 24

Piecewise linear approximation and separable programming

In many applications, the model exhibits nonlinear features: we may have to optimise a nonlinear objective, and also the feasible region may not be defined by linear constraints. We could have, for example, a maximisation problem with increasing returns to scale. This can happen, for example, when a higher number of goods produced leads to lower variable cost per unit of the product. In this chapter, we consider one particular approach to solving non-linear problems, by approximating the objective function by a piecewise linear function. We will also consider a special case of non-linear functions called *convex* functions and we will see that approximating these by piecewise linear functions is particularly computationally easy.

24.1 Piecewise linear objective functions

In this section, we will approximate a continuous objective function of one variable by a piecewise linear function. The idea used here can be generalised to objective functions that are non-linear in more than one variable and to objective functions that are piecewise linear.

Suppose we wish to minimise a non-linear function $f(x)$ defined for $0 \leq x \leq 100$, as pictured in Figure 24.1. We approximate $f(x)$ by a piecewise linear function by splitting the range of x into four intervals of equal length, as shown in the figure. If we take one of these intervals, for example $[25, 50]$, we can express any x in this interval by using non-negative variables θ_2 and θ_3, as a convex combination of 25 and 50

$$x = \theta_2(25) + \theta_3(50) \text{ and } \theta_2 + \theta_3 = 1$$

Using these values of θ_2 and θ_3 we can approximate the true value of $f(x)$ by combining $f(25)$ and $f(50)$ as follows

$$f(x) \approx \theta_2 f(25) + \theta_3 f(50).$$

So to approximate the value of $f(x)$ for any $x \in [0, 100]$, we could use non-negative continuous variables $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ to write

$$f(x) \approx \theta_1 f(0) + \theta_2 f(25) + \theta_3 f(50) + \theta_4 f(75) + \theta_5 f(100)$$

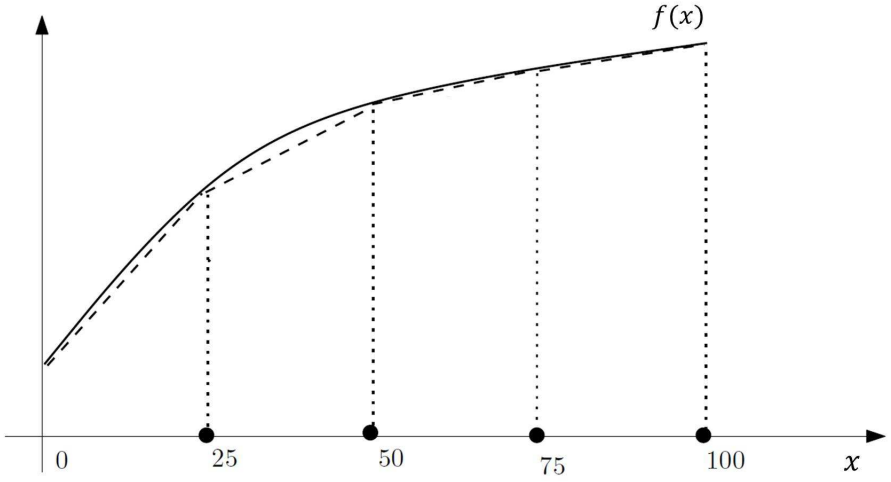


Figure 24.1: Piecewise linear approximation of a non-linear function with one variable

with the constraints

$$x = \theta_1(0) + \theta_2(25) + \theta_3(50) + \theta_4(75) + \theta_5(100)$$
$$\sum_{i=1}^{5} \theta_i = 1,$$

provided we can also enforce the *additional* restriction that only two consecutive variables θ_i can be positive, and all other θ_i equal to zero.

We do this by introducing binary variables $\delta_1, \delta_2, \delta_3, \delta_4 \in \{0, 1\}$ which we will use to represent if a point x is in the interval $[0, 25]$, $[25, 50]$, $[50, 75]$ or $[75, 100]$, respectively. That is, a binary variable has the value 1 if and only if the variable x is within the corresponding interval (if x is on the boundary of two intervals, one interval is chosen arbitrarily). We achieve this by the following set of constraints

$$\begin{aligned}
\theta_1 &\leq \delta_1 \\
\theta_2 &\leq \delta_1 + \delta_2 \\
\theta_3 &\leq \delta_2 + \delta_3 \\
\theta_4 &\leq \delta_3 + \delta_4 \\
\theta_5 &\leq \delta_4 \\
\sum_{i=1}^{4} \delta_i &= 1.
\end{aligned}$$

These constraints ensure that exactly one of the δ_i is 1 and the θ_i are zero for all but two indices.

For the general problem, we can split the range of x into intervals $[a_1, a_2]$, $[a_2, a_3]$,...,$[a_{k-1}, a_k]$, and approximate $f(x)$ by linear functions on each interval. For our minimisation problem,

the formulation is as follows

$$\begin{aligned}
\text{minimise } & \sum_{i=1}^{k} \theta_i f(a_i) \\
\text{subject to } & \theta_1 \leq \delta_1 \\
& \theta_i \leq \delta_{i-1} + \delta_i \text{ for } i = 2, ..., k-1 \\
& \theta_k \leq \delta_{k-1} \\
& \sum_{i=1}^{k} \theta_i = 1 \\
& \sum_{i=1}^{k-1} \delta_i = 1 \\
& \theta_i \geq 0, \forall i = 1, ...k \\
& x = \sum_{i=1}^{k} \theta_i a_i \\
& \delta_i \in \{0, 1\}, \forall i = 1, ..., k-1. \qquad (24.1)
\end{aligned}$$

The approximation we have used here might affect the solution quality. It is hence very important to approximate on a sufficiently fine partition of the domain. Nevertheless, using too many parts can yield a prohibitive increase in the problem size. Using linear approximations, we always have to be aware of these questions and understand the quality of the solution delivered.

24.2 Convex program

Let us now examine a simple type of nonlinear problem that can often be solved as LP. Consider a problem in the following form

$$\begin{aligned}
\text{minimise } & f(x) \\
\text{subject to } & Ax \leq b.
\end{aligned}$$

Here the feasible region is defined by linear inequalities (a polyhedron), but the objective function is nonlinear. The function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if for any values $x, y \in \mathbb{R}^n$, the line segment connecting the points $(x, f(x))$ and $(y, f(y))$ is above the function surface (see Fig. 24.2). Formally, for every $0 < \lambda < 1$, and every x, y in the domain of the function, we require

$$f(\lambda x + (1 - \lambda)y)) \leq \lambda f(x) + (1 - \lambda)f(y).$$

Whereas solving nonlinear programmes is very difficult in general, minimising a convex objective over a convex set can be done efficiently, and is not much harder in practice than LPs. This is due to the fact that in a convex program, every local optimum is also a global one. Due to this property, we can solve convex program almost as efficiently as LP. The above example is a special case when we are minimising a convex objective over linear constraints. Let

us note that most solvers cope with convex objectives via different techniques, not discussed here.

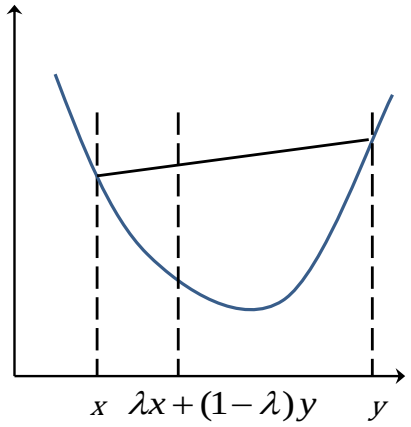


Figure 24.2: Convex function

Clearly, every linear function is convex. A common nonlinear convex function is the quadratic objective

$$f(x_1, \ldots, x_n) = \sum x_i^2.$$

A quadratic function may or may not be convex. For example $f(x) = -x^2$ is *not convex*.

The cubic objective

$$f(x_1, \ldots, x_n) = \sum x_i^3$$

is not convex on the entire $\mathbb{R}^n$ (*Why?*) However, if our variables are restricted to be non-negative, the function is convex on this restricted domain. For convex programming, this is sufficient: we require the objective to be convex only on the set of feasible solutions.

24.2.1 Piecewise linear approximation of convex objectives

An even more special class of convex objectives is *piecewise linear convex objectives*, (see Figure 24.3(a)). If the objective is convex but not piecewise linear, we can approximate it by a piecewise linear convex function (see Figure 24.3(b)), as in the previous section. We partition the domain of the function into sufficiently small regions, and on each region, we replace the function by a linear one.

The difference between the the piecewise linear approximation (24.1) in the previous section and the piecewise linear approximation here is that the fact $f(x)$ is convex eliminates the need for binary variables. Recall that we needed the variables δ_i to ensure that at most two adjacent θ_i are non-zero, while all other θ_i had to be zero. However, when we have a convex piecewise linear function (or if we are approximating a convex function by a piecewise linear function), every point the co-ordinates of which are expressed by a sum of more than two non-zero θ_i, has an objective function value that is higher than the objective function

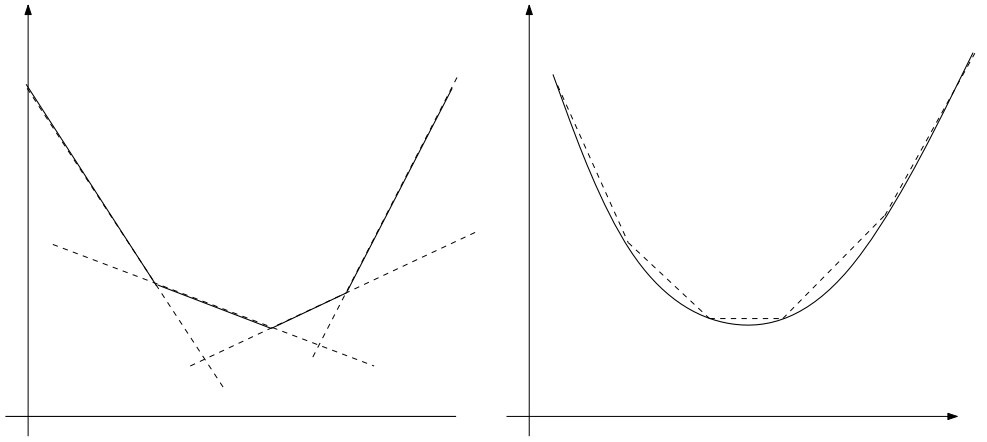


Figure 24.3: Piecewise linear function (a) and piecewise linear approximation of convex function (b)

value that we would get by expressing the same point as a sum of at most two non-zero θ_i. Therefore, if we have a minimisation problem, the model itself ensures that we will never have more than two non-zero θ_i, simply because the minimisation problem will always choose the alternative with the lower objective function value.

As an example, consider the convex function depicted in Figure 24.4. The point $x = 20$ could be represented as a convex combination of the points $0, 25, 50, 75, 100$ in many different ways, for example $20 = \frac{3}{5}(0) + \frac{2}{5}(50)$ or $20 = \frac{1}{5}(0) + \frac{4}{5}(25)$. But if we represent it in the former way, then the corresponding approximation for $f(x) \approx \frac{3}{5}f(0) + \frac{2}{5}f(50)$ will lie on the upper dotted line shown in Figure 24.4, whereas representing x in the latter way, the corresponding approximation for $f(x) \approx \frac{1}{5}f(0) + \frac{4}{5}f(25)$ will lie on the lower dotted line. For a minimisation problem, in the optimal solution x will always be represented in the latter form.

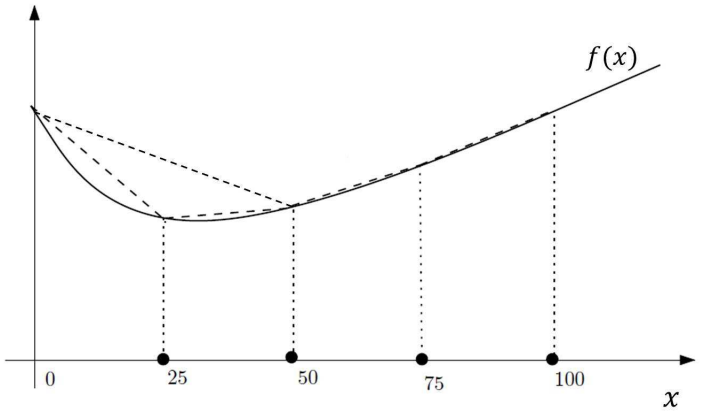


Figure 24.4: An example of piece wise linear approximation of convex function

To summarise, we can formulate our approximation of the problem as the following LP

$$\begin{aligned} \text{minimise } & \sum_{i=1}^{k} \theta_i f(a_i) \\ \text{subject to } & \sum_{i=1}^{k} \theta_i = 1 \\ & \theta_i \geq 0, \forall i = 1, ...k. \end{aligned}$$

As a consequence, optimising a convex objective function is considerably faster than a non-convex one because we do not lose time by executing the Branch-and-Bound algorithm. This also implies that we can afford using many more breakpoints (and hence will have a much more precise approximation) when the objective function is convex.

Note that this argument worked because we were *minimising* a *convex* function. It would also work if we were *maximising* a *concave* function. The function $f : \mathbb{R}^n \to \mathbb{R}$ is *concave*, if for any values $x, y \in \mathbb{R}^n$, the line segment connecting the points $(x, f(x))$ and $(y, f(y))$ is *below* the function surface. If we rewrite a problem of minimising a convex function as a maximisation problem (by multiplying the function by -1) then it turns into a problem of maximising a concave function.

24.2.2 An alternative approach to piecewise linear approximation of convex functions

Let us consider another (actually equivalent) approach to formulating the approximation described in the previous section. This approach is based on the observation that the piecewise linear function, as depicted in Figure 24.3(b) is the same as the maximum of some linear functions. Hence we can reduce the problem to a min-max problem. Note that this approach works only if the function we are approximating is convex (or concave if we are maximising).

Let us consider an example where the objective contains only one nonlinear term. Let f be the function depicted in Figure 24.4, and consider the following LP

$$\begin{aligned} \text{minimise } & 10x_1 + f(x_2) + 12x_3 \\ \text{subject to } & 2x_1 + 4x_2 + 7x_3 \geq 237 \\ & 9x_1 + 6x_2 + 4x_3 \geq 270 \\ & 0 \leq x_1, x_2, x_3 \leq 100. \end{aligned}$$

Since $0 \leq x_2 \leq 100$, we need to approximate $f(x_2)$ on the $[0, 100]$ interval. Let us partition this into 4 intervals of length 25 with breakpoints $t_1 = 0, t_2 = 25, t_3 = 50, t_4 = 75, t_5 = 100$. Let s_1, s_2, s_3, s_4 denote the slope of the segments, and let

$$f_i(x_2) = f(t_i) + s_i \cdot (x_2 - t_i), \forall i = 1, 2, 3, 4.$$

With this notation, we consider the approximate objective function

$$\tilde{f}(x_2) = \text{maximise}\{f_1(x_2), f_2(x_2), f_3(x_2), f_4(x_2)\}.$$

Notice that $\tilde{f}(x_2)$ is linear, and on the interval $[t_i, t_{i+1}]$, $\tilde{f}(x_2) = f_i(x_2)$. Thus we can approximate the original system by the following one, introducing the new variable w

$$\begin{aligned}
\text{minimise} \quad & 10x_1 + w + 12x_3 \\
\text{subject to} \quad & 2x_1 + 4x_2 + 7x_3 \geq 237 \\
& 9x_1 + 6x_2 + 4x_3 \geq 270 \\
& 0 \leq x_1, x_2, x_3 \leq 100 \\
& w \geq f(0) + s_1 \cdot x_2 \\
& w \geq f(25) + s_2 \cdot (x_2 - 25) \\
& w \geq f(50) + s_3 \cdot (x_2 - 50) \\
& w \geq f(75) + s_4 \cdot (x_2 - 75).
\end{aligned}$$

24.3 Separable programming

Approximating an objective function by a piecewise linear function requires one property of the underlying function that we have not mentioned yet: the function must be separable. A function in several variables is called separable if it can be expressed as a sum of functions in single variables, i.e. a function $f(x_1, x_2, ..., x_n)$ is said to be separable if

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} f_i(x_i).$$

For example, the function

$$f(x_1, x_2) = 2x_1 + x_2^3$$

is separable because it is the sum of functions $f_1(x_1) = 2x_1$ and $f_2(x_2) = x_2^3$.

Solving an optimisation problem by means of a separable function is called *separable programming*.

If this condition is given, we can model each component of the sum independently as a piecewise linear function (or approximate it by a piecewise linear function). If this condition is not given, we can, perhaps surprisingly, often transform a function into a separable function. Consider, for example, the function

$$f(x_1, x_2) = x_1 x_2,$$

which is clearly not a separable function. However, by introducing the variables u_1 and u_2 with

$$u_1 = \frac{1}{2}(x_1 + x_2) \text{ and } u_2 = \frac{1}{2}(x_1 - x_2), \tag{24.2}$$

we can write the product $f(x_1, x_2) = x_1 x_2$ as $u_1^2 - u_2^2$ because

$$\begin{aligned}
u_1^2 - u_2^2 &= \frac{1}{4}(x_1^2 + 2x_1x_2 + x_2^2) - \frac{1}{4}(x_1^2 - 2x_1x_2 + x_2^2) \\
&= x_1 x_2.
\end{aligned}$$

Hence we can add constraints (24.2) to the model, write $u_1^2 - u_2^2$ wherever the product $x_1 x_2$ occurs in the model, and finally approximate u_1^2 and u_2^2 by piecewise linear functions.

Another approach for arriving at a separable function that can then be approximated by a piecewise linear function consists in replacing the product $x_1 x_2$ by a variable y, adding the constraint

$$\log y = \log x_1 + \log x_2,$$

and approximating the logarithm by a piecewise linear function. The feasibility of this method, however, depends very much on the numerical stability of the solver the modeller uses.

Remarks

1. The two methods presented above can be used to successively transform any polynomial function into a separable function. As any (sufficiently smooth) function can be approximated by a polynomial function, Separable Programming is a helpful approach to many problems.

2. The general concept of transforming a function into a separable one can also be applied to a constraint with a non-linear left-hand side, which allows for modelling more complex feasible regions than the feasible regions in LPs, which are bounded by hyperplanes. The approximation of such a separable function, of course, will be (piecewise) linear again.

3. Sometimes it is for computational reasons not possible to use binary variables for approximating a non-convex separable objective function by a piecewise linear function. In this case, specific algorithms for Separable Programming can be used. They are a modification of the Simplex Algorithm, which makes sure that at any extreme point under consideration no more than two θ_i are basic variables. While this procedure ensures a precise evaluation of the piecewise linear function used for approximation, it cannot guarantee any more that a true global optimum is reached. In these cases, the modeller can only expect to find a local optimum.

4. Instead of transforming a function into a separable one, alternative options are possible. In particular, any function can be approximated on a multi-dimensional 'grid' for each point of which the value of the function is calculated. For all other points that are not on the grid, the value of the function is linearly approximated by the points on the grid that are in the neighbourhood of these points. (For details, see H.P. Williams, *Model Building in Mathematical Programming* at the end of chapter 9.3.) This method, however, comes with many binary variables and is therefore only rarely the method of choice.

24.4 Exercises

Exercise 24.1. Model and solve the following mathematical programme with Excel Solver by approximating it by a piecewise linear function with 5 equidistant breakpoints

$$\begin{aligned} \text{minimise} \quad & 2x^2 + 12.5y \\ \text{subject to} \quad & x \geq 2 \\ & x + y \geq 12 \\ & x \leq 4 \\ & x, y \geq 0. \end{aligned}$$

Chapter 25

More on non-linear functions and Integer Programs

In this chapter we begin by examining the problem of modelling an important non-linear function: the absolute value function. This is important because absolute values are used to represent distances and we are often interested in maximising or minimising distances. We will see that in such problems we can sometimes find a LP formulation and in other cases we must use binary variables. We will also address what distinguishes good from bad IP formulations.

25.1 Modelling absolute values

In the previous chapter, we saw how we could model the minimisation of a piecewise linear convex function using a LP. An important example of a piecewise linear convex set is the absolute value function $x \mapsto |x|$. If we wish to minimise an objective that includes the term $|x|$ we can model this by defining a new variable $u \geq 0$ and adding constraints $u \geq x$ and $u \geq -x$. Then we can replace $|x|$ in the objective by u and since we are minimising, this ensures that u is chosen to be $\max\{x, -x\} = |x|$.

25.1.1 The facility location problem

So see how this works in practice, let us consider the problem of the optimal placement of a public facility such as a swimming pool or library. Suppose there are n small towns that are being taken into consideration, whose locations are given by coordinates $(a_1, b_1), \ldots, (a_n, b_n)$. We must choose a point (x, y) to locate the facility. Each town would like to minimise their distance to the facility. We measure the distance between the towns and the facility using the *Manhattan metric*, so that the distance between town j and the facility is $|a_j - x| + |b_j - y|$.

There are various approaches to this problem (multiple objectives), and we consider two. The first is to minimise the average distance between the facility and the towns, perhaps weighted by population or some other measure of importance. Let the population of town j be p_j. Then we try to minimise the function $\sum_{j=1}^{n} p_j(|a_j - x| + |b_j - y|)$. This is clearly not a linear function, but it is a sum of convex functions, and therefore convex itself.

We introduce new variables $u_1, ..., u_n \geq 0$ to model the terms $|a_j - x|$ and $v_1, ..., v_n \geq 0$ to model the terms $|b_j - y|$ so that the problem becomes

$$\begin{aligned} \text{minimise } & \sum_{j=1}^{n} p_j(u_j + v_j) \\ \text{subject to } & u_j \geq (a_j - x) \\ & u_j \geq -(a_j - x) \\ & v_j \geq (b_j - y) \\ & v_j \geq -(b_j - y), \forall j = 1 \dots n. \end{aligned}$$

The model will ensure that u_j is chosen to be the maximum of $(a_j - x)$ and $-(a_j - x)$, which is equal to $|a_j - x|$. Similarly for v_j.

The second approach to this problem is to minimise the maximum distance of the facility from any of the towns. In this case, we can define a new variable z which will represent this maximum distance, and add constraints to say that z should be at least the value of $u_j + v_j$ for each $j = 1, \cdots, n$. Then the new LP formulation is:

$$\begin{aligned} \text{minimise } & z \\ \text{subject to } & z \geq u_j + v_j \\ & u_j \geq (a_j - x) \\ & u_j \geq -(a_j - x) \\ & v_j \geq (b_j - y) \\ & v_j \geq -(b_j - y), \forall j = 1 \dots n. \end{aligned}$$

We could in fact make the above formulation more concise by eliminating the variables u_j and v_j and writing four lots of constraints in z, x and y for every j. (*Exercise: do this!*)

25.1.2 The obnoxious facility location problem

Now consider the problem of locating a facility such as a waste disposal plant. In this case we wish to *maximise* the distance of the facility from the towns, subject to the restriction that the facility must be located within some given area. Unfortunately, maximising an objective function that includes the term $|x|$ (or indeed maximising any convex function) is not as easy as minimising, and we must resort to using binary variables. We write x as $x = x^+ - x^-$, where $x^+ \geq 0$ is the 'positive part' of x and $x^- \geq 0$ is the 'negative part'. The interpretation of x^+ and x^- is that if x is positive it is equal to x^+ and if x is negative it is equal to $-x^-$. In this case the absolute value of x can be expressed as $|x| = x^+ + x^-$.

In order to ensure that the model chooses x^+ and x^- in the way that we intend, we introduce a binary variable θ the represent whether x is positive ($\theta = 0$) or x is negative ($\theta = 1$), and the following big-M constraints

$$\begin{aligned} & x^+ \leq (1 - \theta)M \text{ and} \\ & x^- \leq \theta M, \end{aligned}$$

where M is an upper bound for $|x|$. These constraints ensure that at most one of x^+ and x^- can be non-zero.

We can apply this method to the Obnoxious Facility Location Problem, either maximising a weighted average of the distances of the towns from the facility or maximising the minimum distance.

25.2 Good IP formulations and the convex hull of the feasible region

Solving an IP requires much more computational effort than solving a LP. Therefore, when setting up an IP model, the modeller should reflect on the question of whether the formulations chosen is computationally favourable. Obviously, the formulation of a model is computationally favourable when only a few LP problems have to be solved to find an optimal solution of the IP problem.

In order to see when only a few LP have to be solved, recall the circumstances under which Branch-and-Bound fathoms a branch of the search tree:

(a) when the sub-problem is infeasible

(b) when the sub-problem has an integer solution

(c) when the solution of the sub-problem has a lower objective function value (in the case maximisation) than some integer solution that has previously been found.

While improving the search of the optimal solution with respect to the last one of these criteria is primarily a matter of a smart search strategy (i.e. of the algorithmic knowledge that we provide), we can improve the computational time that Branch and Bound needs with respect to the first two criteria by taking care of the declarative knowledge incorporated in our model. That is we can try to use smart inequalities for our model.

The first one of the three criteria above suggests that we build our model such that the feasible region is as small as possible. The second criterion suggests that as many extreme points of the feasible region as possible should be integer. This idea gives rise to the following definition:

Definition 25.1. *For a given IP, the* convex hull *is the smallest possible convex set that contains all feasible solutions.*

Consider the following IP

$$\begin{aligned} \text{maximise} \quad & -x_1 + x_2 \\ \text{subject to} \quad & x_1 \leq 1.5 \\ & x_2 \leq 1.5 \\ & x_1 + x_2 \leq 2 \\ & x_1, x_2 \geq 0 \text{ and integer.} \end{aligned}$$

Figure 25.1 that shows the convex hull of this IP (dark grey) and the feasible region of its linear programming relaxation (union of light grey and dark grey sets). Obviously, this IP is

maximised at point A, while the optimum of its linear programming relaxation is point B. This means that Branch-and-Bound will need to solve at least two LPs. If, however, we had modelled the problem by using the constraints $x_1 \leq 1$, $x_2 \leq 1$ and $x_1, x_2 \geq 0$, i.e. if we had modelled the convex hull of the problem, solving one LP problem would be enough because the optimum of the linear programming relaxation (point A) would also be the optimum of the IP. Such a formulation of a model is called *sharp*.

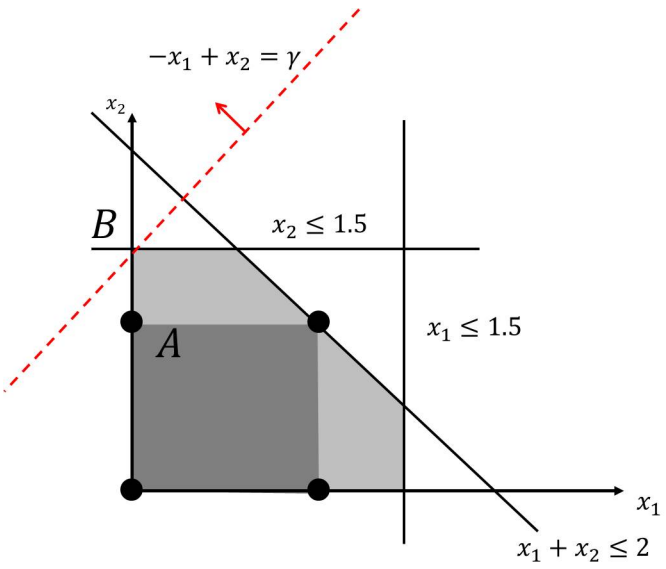


Figure 25.1: An IP example: convex hall and linear programming relaxation

This holds in general: whenever we can model the convex hull of an IP problem (i.e. find a sharp formulation), we can be sure that solving the LP relaxation of the IP problem will provide us with the optimal solution of the IP problem.

Unfortunately, the convex hull is known for only very few subclasses of integer programming problems. In the general case, we have to resort to Branch and Bound or other computationally expensive approaches. However, it is always helpful to try to find a formulation that leads to an LP relaxation with a rather small feasible region. The closer we can approach the perfect formulation, i.e. the convex hull, by the way in which we set up our inequalities, the better our formulation will be. And this is likely to reduce considerably the computational time we will need to solve the problem. This is illustrated by the following examples.

Firstly consider a situation where both binary variables δ_B and δ_C take the value 1 if binary variable δ_A takes the value 1. F1 and F2 below are two valid formulations for this situation:

$$\begin{aligned} &\text{F1: } \delta_A \leq \delta_B \text{ and F2: } 2\delta_A \leq \delta_B + \delta_C \\ &\qquad \delta_A \leq \delta_C \end{aligned}$$

If we restrict our attention to 0-1 values for the three variables, both formulations are equivalent: both exclude solutions where $\delta_A = 1$ and at least one of δ_B and δ_C is equal to zero. This however is not true for the LP relaxations of these two formulations. To see this consider

the non-integral solution $\delta_A = 0.5$, $\delta_B = 0$, $\delta_C = 1$. Since this solution is nonintegral, it is beneficial that it is a-priori excluded from consideration during the Branch and Bound search. Clearly this solution violates the system in F1, but is a valid solution for F2. In general, note that the single inequality in F2 can be obtained by adding the two inequalities in F1, so it is implied by the inequalities in F1. Therefore, any solution that satisfies F1 will also satisfy F2. The converse, however is not true as we have just seen. As a result, F1 is a stronger formulation for this situation.

In general, if θ is a binary variable that is only allowed to assume the value 1 if n binary variables $\delta_1, \ldots, \delta_n$ all take the value 1, the formulation

$$\theta \leq \delta_i, \forall i = 1, \ldots, n$$

is preferred to the formulation

$$n\theta \leq \sum_{i=1}^{n} \delta_i.$$

Similarly to the previous example, in the situation n binary variables $\delta_1, \ldots \delta_n$ are only allowed to assume the value 1 if a binary variable θ takes the value 1, the formulation

$$\theta \geq \delta_i, \forall i = 1, \ldots, n$$

is prefered to the the formulation

$$n\theta \geq \sum_{i=1}^{n} \delta_i.$$

25.3 Dual values in Integer Programming

In Integer Programming the dual values produced by the solver do in general not have a meaning. The optimal solution of the IP will have been found as the optimal solution of a particular LP problem. However, this LP will consist of the original constraints and some other constraints that have been added during the branching process to partition the feasible region. The dual values of this LP do not give any sensitivity information. The optimal solution and the next best solution may be far apart on the Branch and Bound tree on quite different branches. At one of the solutions there is no information about the existence or the properties of the others.

Note, however, that some theoretical effort has been put into defining and analysing dual problems of IP problems (as opposed to the duals of LP problems that we have discussed in previous lectures and that the paragraph above refers to). Typically, these IP duals are used for deriving good upper bounds for maximisation problems (or lower bound for minimisation problems) by means of a weak duality property. Hence they provide us with an alternative approach to solving IP problems, which can also be used in combination with Branch and Bound. These IP duals (such as the so-called Lagrangean dual) must not be confused with the duals of LP problems.

25.4 Exercises

Exercise 25.1. A nuclear reactor is to be located in the region $R = \{(x, y) : 0 \leq x, y \leq 10\}$. Also in this region are 6 towns with coordinates

$$\begin{aligned}(a_1, b_1) &= (1, 6)\\ (a_2, b_2) &= (3, 1)\\ (a_3, b_3) &= (4, 6)\\ (a_4, b_4) &= (8, 3)\\ (a_5, b_5) &= (9, 1)\\ (a_6, b_6) &= (10, 6).\end{aligned}$$

The government wishes to choose a location for the reactor that is far away from the towns. Model and solve the problem of choosing the optimal position $(x, y) \in R$ for the reactor in the two cases that government wishes to maximise

(a) the average distance, $\frac{1}{6}\sum_{i=1}^{6} |x - a_i| + |y - b_i|$ between the towns and the reactor,

(b) the minimum distance, $\min\{|x - a_i| + |y - b_i| : i = 1, \cdots, 6\}$ between the towns and the reactor.

Exercise 25.2. A number of power stations are committed to meeting the following electricity load demands over a day:

12 p.m. to 6 a.m.	15000 megawatts
6 a.m. to 9 a.m.	30000 megawatts
9 a.m. to 3 p.m.	25000 megawatts
3 p.m. to 6 p.m.	40000 megawatts
6 p.m. to 12 p.m.	27000 megawatts

There are three types of generating unit available: 12 of type 1, 10 of type 2, and 5 of type 3. Each generator has to work between a minimum and a maximum level. There is an hourly cost of running each generator at minimum level (column A). In addition there is an extra hourly cost for each megawatt at which a unit is operated above minimum level (column B). To start up a generator also involves a cost.

You should prepare a cyclic plan repeated every day. If a generator is running around the clock, then no startup cost has to be paid. If a generator runs from 6am to 9am and from 3pm to 6pm, then the startup cost has to be paid twice.

The cost information is given in the table below (with cost in £s).

	Minimum Level	Maximum Level	A	B	Startup cost
Type 1	850 MW	2000 MW	10000	20	20000
Type 2	1250 MW	1750 MW	26000	13	10000
Type 3	1500 MW	4000 MW	30000	30	5000

In addition to meeting the estimated time load demands there must be sufficient generators working at any time to make it possible to meet an increase in load of up to 15%. This increase would have to be accomplished by adjusting the output of generators already operating within their permitted limits.

(a) Which generators should be working in which periods of the day to minimise total cost?

(b) What would be the saving of lowering the 15% reserve output guarantee; i.e. what does this security of supply guarantee cost?

Chapter 26

Simulation

26.1 Introduction

Deterministic models are used when the system can be studied analytically, that is when we can write down closed form solutions for particular outputs. Once you know the closed form formulae, the results can be computed in seconds. However, such results might only be available for simple models, which are very rough approximations of the real world. On the other hand, stochastic models have been developed to deal with uncertainty with the help of probability and statistics. One way of analysing stochastic system is using simulation. Simulation is one of the most frequently employed techniques to model processes (especially random processes) that are too complex to be solved by analytical methods.

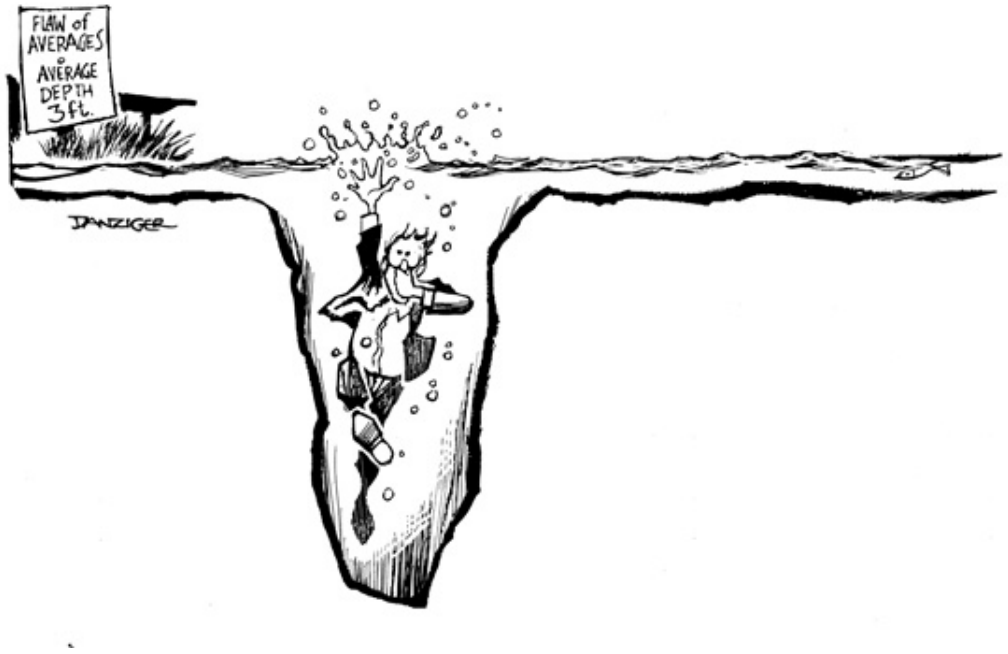
Of particular interest to us are systems where one or more of the variables show variability around a central value (i.e. average for example). Many real-life systems show this behavior. For example, daily sales of most items in a supermarket vary according to the weather, the season, the day of the week, etc. Looking only at the average value of daily sales might be misleading or at least provide limited information. Sam Savage refers this as "the Flaw of Averages" and states that "*plans based on the assumption that average conditions will occur are usually wrong.*" An extreme example involves the statistician who drowned while fording a river that was ***on average only three feet deep***![1]

In this chapter, we will investigate a few cases in which acting on the average can be extremely flawed. We will demonstrate how to appropriately analyse using Monte Carlo Simulation and Discrete Event Simulation.

26.2 Simulation examples

- **A manufacturing company.** The company is having problems with their production line. And at the same time demand is increasing, but only during certain periods of time. Before expanding the production capacity by introducing a new production line, the manager carries out a simulation which can incorporate the multiple factors and uncertainties (i.e. varying demand throughout the year, potential delay with the supply

[1] See book "The Flaw of Averages: Why We Underestimate Risk in the Face of Uncertainty" by Sam L. Savage for further reference.



of raw materials, unexpected fires or strikes). It is possible to identify the bottlenecks, how delays in the production might affect sales.

- **Oil industry.** In the oil industry, previously state-run oil companies are opening opportunities to multinationals for joint ventures in the exploitation of new oil fields. Simulation helps the interested companies to carry out an economic evaluation that takes into account the risk factor. This analysis starts by calculating the Net Present Value (NPV) based initially upon an average value. A simulation analysis is followed to allow the company to compute the downside risk.

- **A&E Department.** Long queues are forming at the hospital's A&E unit. The manager has read about other hospitals where more staff have been added, only to find that the queues remain. The manager has a friend who works in OR and tells him to carry out a simulation of the A&E unit. A simulation model is developed to replicates the arrivals of patients at different times of the day, delays on seeing the nurses and/or doctors, and other factors. This will allow the manager to identify possible bottlenecks, investigate different alternatives and analyse what the results and costs are of increasing staff at particular times of the day.

26.3 Monte Carlo simulation

Simulating random systems using *coin-tossing*! For comparison, simulating a chemical manufacturing process, or the behavior of an airplane wing, might not be Monte Carlo. Monte Carlo simulation is an analytical technique in which a large number of simulations are run using random quantities for uncertain variables and looking at the distribution of results to

infer which values are most likely. The techniques was first used by scientists working on the atom bomb.

An important application of Monte Carlo simulation is to allow people to account for risk in quantitative analysis and decision making. The technique is used by professionals in such widely disparate fields as finance, project management, energy, manufacturing, engineering, research and development, insurance and etc.

We will use two examples (Toy.xlsx and NPV.xlsx, check Moodle) to demonstrate Monte Carlo Simulation using @Risk.

26.4 Discrete event simulation

Discrete event simulation (DES) is the process of codifying the behavior of a complex system as an ordered sequence of well-defined events. In this context, an event comprises a specific change in the system's state at a specific point in time. Common applications of DES include stress testing, evaluating potential financial investments, and modeling procedures and processes in various industries, such as manufacturing, healthcare and service.

We will use an ATM queuing example (ATM.xlsx, check Moodle) to demonstrate to demontrate discrete event simulation using Excel modelling.

26.5 Advantages and disadvantages of simulation

Simulation gives the ability to gain insights into the model solution which may be impossible to attain through other techniques. Simulation provides a convenient experimental laboratory to perform "what if" and sensitivity analysis. An accurate and detailed simulation model may be time consuming to develop. Simulation is, in effect, a trial and error method of comparing different policy inputs. You should not expect that the solution obtained will be optimal: some input which was not considered could have been better. Please note that simulation is not exact. To get a good answer, you may have to run many trials. Like any technique, simulation is not magic. You must understand the reality to make a good model. In particular, correlations between variables may be vitally important (e.g., for predicting a collapse of the financial system).

26.6 TransAtlantic Airline case study

Please download the case study on Moodle.

26.7 Exercises

Exercise 26.1. Estimate the value of π using simulation.
Hints: you can generate random points within a 2 by 2 square and check how many of them fall inside a cirle of radius 1.

Exercise 26.2. A newsvendor has to purchase newspapers from a publisher at the start of the day at a given price. The number of newspapers he can sell is unknown at this time, and

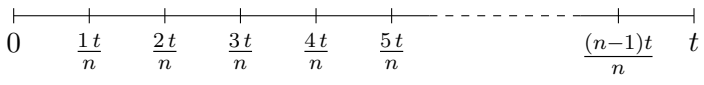
so he has to balance the risk of purchasing newspapers he cannot sell with the risk of turning away customers empty-handed. He is able to sell spare newspapers back to the publisher, for a much reduced salvage value. Specifically, the newsvendor buys newspapers for 0.20 and sells them for 0.45. Unsold newspapers can be salvaged for 0.05 per unit. Suppose the number of customers who arrive can be approximately modelled by a gamma distribution with alpha parameter 100 and beta parameter 5. As the median of this distribution is about 500, he decides to purchase 500 newspapers.

Use @Risk to perform some risk analysis and examine if the newsvendor is doing the right purchasing strategy.

Appendix A

A.1 The Poisson process, Poisson distribution, and exponential distribution

Subdivide the time interval [0,t) into a large number n of subintervals of width t/n, as illustrated below.



If n is large enough, then by assumption (b) in Section 19.2.1, the probability of exactly one arrival occurring in any given time interval is "essentially" $\lambda t/n$ (to within terms of order smaller than $1/n$), and the probability of no arrivals is essentially $1 - \lambda t/n$. The *Poisson process* is the set of all arrival times, in the large-n limit.

The *Poisson distribution* governs the number of arrivals $X(t)$ occurring in time t. To compute it, note that "k arrivals in $[0, t)$" means that some k of the n subintervals each have exactly one arrival, and the other $n - k$ have no arrivals. By assumption (a), arrivals in each of the n intervals are independent, thus the probability of exactly k arrivals is given by the binomial distribution

$$f_n(k) = \binom{n}{k} \left(\frac{\lambda t}{n}\right)^k \left(1 - \frac{\lambda t}{n}\right)^{n-k}, \quad k = 0, 1, 2, 3, \ldots$$

Recalling that $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ and rearranging factors in the product above, it follows that:

$$\begin{aligned}
\mathbb{P}(X(t) = k) &= \lim_{n\to\infty} f_n(k) \\
&= \frac{(\lambda t)^k}{k!} \cdot \lim_{n\to\infty} \frac{n!}{(n-k)!}\frac{1}{n^k} \cdot \lim_{n\to\infty} \left(1 - \frac{\lambda t}{n}\right)^n \cdot \lim_{n\to\infty} \left(1 - \frac{\lambda t}{n}\right)^{-k} \\
&= \frac{(\lambda t)^k}{k!} e^{-\lambda t},
\end{aligned}$$

where the last equality comes from the fact that the first and third limits are 1, while the second is $e^{-\lambda t}$. This is the Poisson distribution function with rate λt. Setting $t = 1$ gives the familiar rate-λ Poisson distribution function $e^{-\lambda}\lambda^k/k!$.

We now show that the *exponential distribution* governs the interarrival time T. Condition on the event that there is an arrival at time τ, and, for any given t, consider the event that the next interarrival time T has $T > t$, i.e., the event that there is no arrival in the interval $(\tau, \tau + t)$. By the memoryless property, this event is independent of there having been an arrival at time τ, thus its probability is simply the (unconditioned) probability of there being 0 events in the interval $(\tau, \tau + t)$. But the calculation above for the number of events in the interval $[0, t)$ applies equally to this interval (indeed to any length-t interval). Thus,

$$\mathbb{P}(T > t) = \mathbb{P}(\text{no event in } (\tau, \tau + t)) = \mathbb{P}(X(t) = 0) = e^{-\lambda t}.$$

Let $A(t)$ be the CDF of the interarrival time distribution. By definition, $A(t) = \mathbb{P}(T \leq t)$, so

$$A(t) = 1 - \mathbb{P}(T > t) = 1 - e^{-\lambda t}.$$

In essence we are now done: $A(t)$ is familiar as the CDF of an exponential distribution of rate λ. All properties of this distribution are well known, including the CDF and the mean, but for thoroughness, let us derive both.

By definition, the CDF $A(t)$ is the integral of the PDF $a(t)$, i.e., $A(t) = \int_0^t a(\tau)d\tau$, so the PDF is the derivative of the CDF: $a(t) = \frac{d}{dt}A(t)$. Here,

$$a(t) = \frac{d}{dt}(1 - e^{-\lambda t}) = \lambda e^{-\lambda t}.$$

The mean interarrival time is

$$\begin{aligned}\mathbb{E}[T] &= \int_0^\infty t\, a(t)\, dt = \int_0^\infty t\, \lambda e^{-\lambda t}\, dt = \frac{1}{\lambda}\int_0^\infty x e^{-x}\, dx \\ &= \frac{1}{\lambda}\left(-(1+x)e^{-x}\right)\Big|_0^\infty = \frac{1}{\lambda}.\end{aligned}$$

A.2 Geometric progression

A *geometric progression* is a sequence of the form $1, a, a^2, a^3, \ldots\ a^n, \ldots$, where $a > 0$. The sum of the first n terms of the sequence, called a geometric *series*, is

$$S_n = 1 + a + a^2 + a^3 + \cdots + a^n.$$

Multiplying by $1 - a$ on both sides of the above equality, we obtain

$$(1-a)S_n = (1 + a + a^2 + a^3 \cdots + a^n) - (a + a^2 + a^3 + \cdots a^n + a^{n+1}) = 1 - a^{n+1},$$

implying

$$S_n = \frac{1 - a^{n+1}}{1 - a}.$$

The sum of an infinite geometric series is $+\infty$ if $a \geq 1$, while if $a < 1$ it is:

$$1 + a + a^2 + a^3 + \cdots = \lim_{n\to+\infty} S_n = \lim_{n\to+\infty} \frac{1 - a^{n+1}}{1 - a} = \frac{1}{1 - a}. \tag{A.1}$$

A.3 Arithmetic-geometric progression

An *arithmetic geometric progression* is a sequence of the form $a, 2a^2, 3a^3, \dots\ na^n, \dots$, where $a > 0$. The sum of the first n terms of the sequence is

$$S_n = a + 2a^2 + 3a^3 + \cdots + na^n.$$

Multiplying by $1 - a$ on both sides of the above equality, we obtain

$$\begin{aligned}(1-a)S_n &= (a + 2a^2 + 3a^3 \cdots + na^n) - (a^2 + 2a^3 + 3a^4 + \cdots (n-1)a^n + na^{n+1}) \\ &= a + a^2 + a^3 + \cdots + a^n - na^{n+1},\end{aligned}$$

implying

$$S_n = \frac{a}{1-a}(1 + a + \cdots + a^{n-1}) - \frac{na^{n+1}}{1-a}.$$

Note that the term in parenthesis is a geometric series, therefore it equals $(1 - a^n)/(1 - a)$, and we obtain

$$S_n = \frac{a - a^{n+1}}{(1-a)^2} - \frac{na^{n+1}}{1-a}.$$

The sum of an infinite arithmetic-geometric series is $+\infty$ if $a \geq 1$, while if $a < 1$ it is:

$$a + 2a^2 + 3a^3 + \cdots = \lim_{n \to +\infty} S_n = \frac{a}{(1-a)^2}. \tag{A.2}$$