II. OPERATIONS RESEARCH AND LIBRARY MANAGEMENT

II.0. INTRODUCTION

Library managers or library network planners must know in some detail how the user of a library behaves : how often he or she uses some particular reference manual, how much of the library's collection is on loan, how books should be placed with respect to accessibility, and how much of the yearly budget should be allocated to the purchase of periodicals. Generally speaking, the library manager or planner should be able to predict what will probably occur in the near future and base his/her decisions on this knowledge. For this reason it is necessary to apply statistical inference, probability theory and the techniques of *operations research* as well.

The purpose of library operations research activities can be summarised as follows :

(i) to help library managers and operators make better plans and take decisions which will enhance the maximum exploitation of the library's resources by its user community;

(ii) to provide a basis for examining and evaluating library operations and a logical basis for providing guidelines for effecting change.

General references for this part are Swanson and Bookstein (1972), Daiute and Gorman (1974)and Brophy et al. (1976).

II.1. PROGRAMMING PROBLEMS

Programming problems are generally concerned with the optimum allocation of scarce resources among a number of products or activities. These scarce resources may be materials, staff, investment capital or processing time on large, expensive machines. An optimum allocation may be one that maximises some measure of benefit or utility, such as profit, or minimises some measure of cost. The term '*linear programming*' defines a particular class of programming problems, where the criterion for selecting the 'best' values of the decision variables can be described by a linear function of these variables and where the operating rules governing the process can be expressed as a set of linear equations or linear inequalities (Dantzig (1962)).

II.1.1. Graphical solution of linear programming problems in two variables

We will begin with an elementary example. A printing-binding firm plans its production a day in advance. They basically have two kinds of products to consider : hard covers (product A) and paperbacks (product B). During the production process three basic operations are performed (which we do not specify, but call them operations I, II and III).

The processing time for products A and B are given in Table II.1.1; this table also shows the maximum capacity and the profits. How much of each product should this firm produce in order to maximise its profit? Note that to keep things simple we consider only the question of supply and not that of demand.

Table II.1.1. Process times (in minutes), capacities and profits

	A	B	maximum capacity (a day)
Operation I	1	-	440
Operation II	1	1	1000
Operation III	1.2	0.5	620
Profits ($/unit)	5	4	

To solve this problem we set :

x = the total number of hard covers (product A) produced during one day

y = the total number of paperbacks (product B) produced during one day.

Then the following inequalities must be satisfied :

$$\begin{cases} x \leq 440 , \\ x + y \leq 1000 , \\ 1.2x + 0.5y \leq 620 . \end{cases}$$

Moreover, $x \geq 0$ and $y \geq 0$. Finally, we wish to maximise $z = 5x + 4y$.

A solution such as $x = 200$, $y = 100$, satisfies all the constraints. It is called a '*feasible solution*'. The set of all feasible solutions is called the '*feasible region*'. Solving a linear problem means finding the best feasible solution in the feasible region, i.e. a point in the feasible region that maximises z.

To represent the feasible region in two dimensions, every constraint is plotted and all values of x and y that satisfy these constraints are identified (Fig.II.1.1). The non-negativity constraints imply that all feasible values of the two variables will lie in the first quadrant.

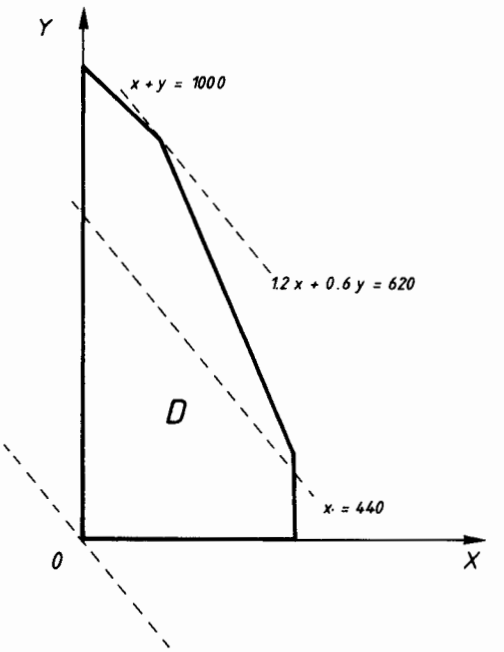

Fig.II.1.1 Constraints and feasible region for the example in Table II.1.1

The feasible region is given by the convex set D. Clearly, there is an infinite number of feasible points in the region. Our aim is to identify the

feasible point that maximises the objective function $z = f(x,y) = 5x + 4y$.

Observe that if c is a constant, the function $5x + 4y = c$ represents a straight line. Changing the value of c translates the entire line to another straight line parallel to it. In order to find an optimal solution, the objective function line is drawn for a convenient value of c such that it passes through one or more points in the feasible region. In this example we choose $c = 0$. The value of c increases when this line is moved further away from the origin. The only limitation on this increase is that the straight

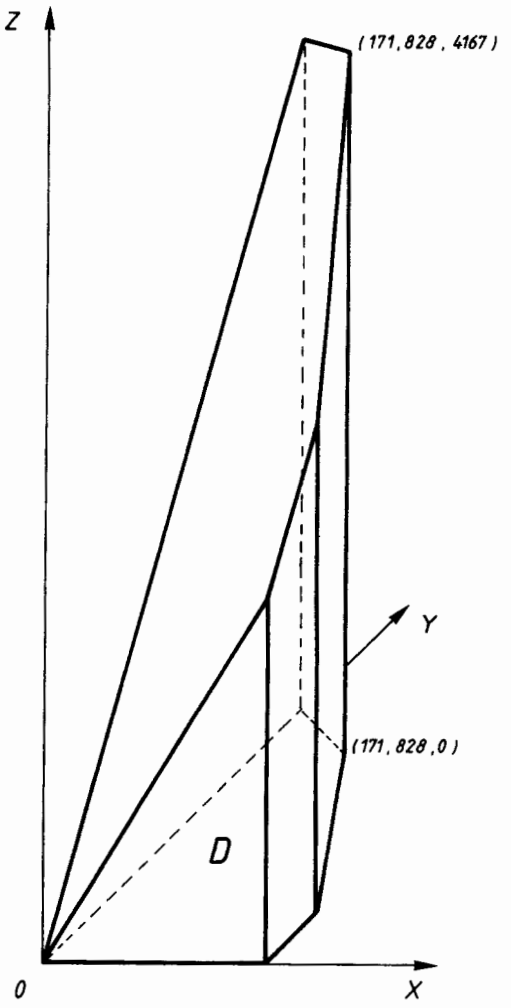

Fig.II.1.2 Feasible region D and values of the objective function (z) for the problem in Table II.1.1

line $5x + 4y = c$ has to contain at least one point of D. Proceeding in this way, we find as the best feasible point the corner point with coordinates (171.43, 828.57), resulting in the largest value for c equal to 4171.43. So, we have found that the best production schedule is to make (we need to round off of course) 171 units of product A (hard covers) and 828 units of product B (paperbacks), yielding a net profit of $ 4167. This example also illustrates the fact that the optimal value is always attained at a corner.

Fig.II.1.2 illustrates the same example in three dimensions : the objective function has been represented along the z-axis.

Similar to the example shown above, minimum problems can also be solved in two dimensions (two variables). A minimum problem can also be reduced to a maximum problem by studying -z instead of z. In the case in which there are not too many constraints, a similar method using three variables can be applied. In real life problems, however, there are many more variables and other techniques are necessary. These will be discussed in the next Section.

II.1.2. Formal statement of the linear programming problem and the simplex method

A problem in linear programming looks like the following :

1) We have a linear system of m+n equalities or inequalities, in which the last n are constraints on the sign of the n variables.

2) A linear functional, called the '*objective function*', has to be optimised (i.e. maximised or minimised).

Bear in mind the fact that a function f from a real vector space X to a real vector space Y is *linear* if for every $\lambda, \mu \in \mathbb{R}$, for every

$$x,y \in X : f(\lambda x + \mu y) = \lambda f(x) + \mu f(y) \quad . \qquad [II.1.1]$$

This linear function (function of the first degree) is said to be a linear functional if the vector space Y is $\mathbb{R}$. A function such as $f_1 : \mathbb{R}^3 \to \mathbb{R} : (x_1,x_2,x_3) \to ax_1 + bx_2 + cx_3$ $(a,b,c \in \mathbb{R})$ is a linear functional, while a function such as $f_2 : \mathbb{R}^3 \to \mathbb{R} : (x_1,x_2,x_3) \to ax_1 + bx_2 + cx_3 + d$ $(d \in \mathbb{R})$ is not. However, it is easy to see that an extremal value of f_2 is merely an extremal value of f_1 plus d.

A vector $X \in \mathbb{R}^3$ in which the components satisfy all conditions is called a 'feasible solution'. An *optimal solution* is a feasible solution that optimises the objective function. Optimal solutions do not have to exist, and if they do, they do not have to be unique. In Fig.II.1.3a we show an example of an unbounded

convex region D where no maximum value for the objective function can be found (there is of course a minimal solution); Fig.II.1.3b gives an example where there are an infinite number of optimum values, namely every point on the line segment KL.

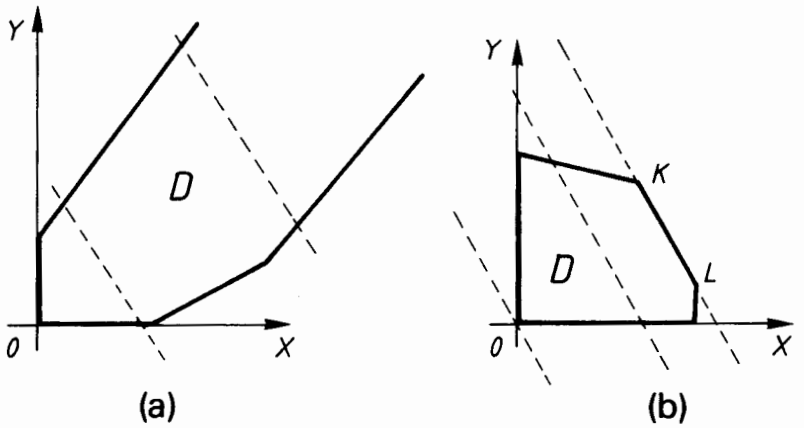

Fig.II.1.3a : Feasible region with no maximum value for the objective function (dotted lines)

Fig.II.1.3b : Feasible region with an infinity of maximum values for the objective function

Formally speaking, the standard primal maximum problem looks like the following : find the maximum of $w = c_1x_1 + c_2x_2 + \ldots + c_nx_n$, given the following constraints :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \ ; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \ ; \\ \qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \ , \end{cases} \qquad \text{[II.1.2]}$$

where

$$x_1, x_2, \ldots, x_n \geq 0 \ .$$

In matrix notation this becomes :

find the maximum of $w = C^tX$

with $AX \leq B$ and $X \geq 0$. [II.1.3]

Here :

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

$$C = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}; \quad B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}; \quad X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}; \quad 0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^n$$

and t denotes transposition. Moreover, a vector $X \in \mathbb{R}^k$ is said to be greater than or equal to a vector $Y \in \mathbb{R}^k$ if for every $i = 1,\ldots,k : x_i \geq y_i$. The vector X is often called the '*decision vector*', B the '*requirement vector*' and C the '*profit (or cost) vector*'.

A standard dual minimum problem is associated with every standard primal maximum problem

$$\text{find the minimum of } v = B^t Y$$
$$\text{with } A^t Y \geq C \text{ and } Y \geq 0 \; . \qquad \text{[II.1.4]}$$

Here $Y = (y_1, y_2, \ldots, y_m)^t$.

One can show (see e.g. Franklin (1980; p. 63)) that if the standard maximum problem has an optimal solution, then its dual minimum problem also has an optimal solution and vice versa. Moreover, the maximum value of the objective function of the maximum problem equals the minimum value of the dual problem.

There is said to be a canonical primal problem when the following problem occurs :

$$\text{maximise } w = C^t X$$
$$\text{subject to } AX = B \text{ and } X \geq 0 \; . \qquad \text{[II.1.5]}$$

Every standard problem can be converted into a canonical one by using so-called slack variables z_i. This is done as follows :

$$\sum_j a_{ij} x_j \leq b_i \text{ becomes } \left(\sum_j a_{ij} x_j\right) + z_i = b_i ; \; z_i \geq 0 \; . \qquad \text{[II.1.6]}$$

So, for practical purposes, we only have to consider canonical problems. Note again that any minimisation problem can be reduced to a maximisation problem by using -w instead of w.

An iterative procedure, known as the simplex method, is used as a practical solution technique. This algorithm's efficiency and versatility is largely responsible for the importance of linear programming. The simplex method was introduced in 1947 by George B. Dantzig. Its substantial value lies in the fact that it is fast, has a variety of applications and can answer important questions about the sensitivity of solutions to variations in the input data. With the simplex method one can also impose additional constraints and solve the problem again to examine their effect. For example, the method can quickly figure out the cost of providing an unprofitable service in order to maintain a customer's goodwill.

The simplex method is a step-by-step procedure consisting of two phases :

phase I : find a feasible solution X_0 of $AX = B$, $X \geq 0$ or prove that such a solution does not exist;

phase II : find an optimal solution.

Phase II is done iteratively, by constructing a finite sequence $X_0, X_1, X_2, \ldots, X_s$ of corner points of the feasible region D, such that X_i and X_{i+1} are adjacent and such that $w = C^t X$ increases (maximum problem). The algorithm stops at the optimal solution or when it has been proved that such a solution does not exist.

We note that the fact that an optimal solution always lies at a corner point in the feasible region reduces an originally infinite problem (any point in the feasible region could be an optimum) to a large but finite problem (a finite number of linear restrictions leads to a region in space with only a finite number of corners).

II.1.3. Integer programming

In practice, many linear programming problems do require integer solutions for some or all of the variables. For instance, it is not possible to lend a fractional number of books. Our graphical problem in II.1.1 only produced useful results for an integer number of books. The term '*integer programming*' refers to that class of linear programming problems in which some or all decision variables are restricted to being integers. However, solutions of integer programming problems are generally difficult, time consuming and expensive. Hence a practical approach is to treat all the variables as continuous and to solve the associated linear program by the

simplex method. Solutions are then rounded off to the nearest integer such that constraints are not violated. This generally provides a good approximation of the optimal integer solution, especially when the values of the integer variables are large (cf. Section II.1.1).

There are situations in which rounding off produces poor integer solutions, especially when the decision variables can only take the values 0 and 1. Special techniques are then needed to determine the optimal integer solution directly. The so-called '*branch-and-bound*' algorithm is the most widely used method for solving integer programming problems in practice. Most commercial computer codes for solving integer programs are based on this approach (Lawler and Wood (1966)). These problems cannot always be solved exactly in a reasonable amount of time. In these cases heuristic methods yielding approximate best solutions are used. We will discuss some special integer programs in subsequent sections.

II.1.4. Transportation and assignment problems

II.1.4.1. Transportation problems

Transportation problems are generally concerned with the distribution of a certain product from several sources to numerous localities at minimum cost or in a minimum amount of time.

Suppose there are m stores where books are stocked and n local libraries where they are needed. Let the supply of books available at the stores be $a_1, a_2, \ldots, a_m$ and the demands at the local libraries be $b_1, b_2, \ldots, b_n$. The unit cost of shipping books from store i to library j is c_{ij}. If a particular store cannot supply a certain library, we set the appropriate c_{ij} at $+\infty$. We wish to find an optimal shipping schedule which minimises the total cost of transportation.

This type of transportation problem can be formulated as a linear program : we define x_{ij} as the quantity shipped from store i to library j. Since i can assume values from 1 to m and j from 1 to n, the number of decision variables equals mn. This transportation problem is then described by :

$$\text{minimise} \quad w = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad \text{(total cost of transportation)},$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} \leq a_i, \; i = 1, \ldots, m \qquad \text{(supply restrictions at the stores)}$$

$$\sum_{i=1}^{m} x_{ij} \geq b_j \ , \ j = 1,\ldots,n \qquad \text{(demand requirements at the libraries)}$$

$$\forall i,j : x_{ij} \geq 0 \qquad \text{(non-negativity restrictions)} \qquad [II.1.7]$$

The library demands can clearly be met if and only if the total store supply is at least equal to the total library demand. This means that

$$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j \ . \qquad [II.1.8]$$

When the total supply equals the total demand, i.e. $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, we have a standard transportation problem. This has the form :

$$\text{minimise} \quad w = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = a_i \ , \ i = 1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \ , \ j = 1,\ldots,n$$

$$\forall i,j : x_{ij} \geq 0 \ . \qquad [II.1.9]$$

Any non-standard transportation problem, where the supplies and demands are not balanced, can be converted into a standard transportation problem by using a dummy store or a dummy library. Consider, for instance, a problem in which the total supply exceeds the total demand. To convert this into a standard problem, a dummy library is created to absorb the excess supply available at the stores. The unit cost of shipping books from any store to the dummy library is assumed to be zero since in reality the dummy library does not exist and no physical transfer of goods takes place. This leads to the following standard transportation problem :

$$\text{minimise} \quad w = \sum_{i=1}^{m} \sum_{j=1}^{n+1} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{n+1} x_{ij} = a_i \ , \ i = 1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \ , \quad j = 1,\ldots,n+1$$

$$\forall i,j : x_{ij} \geq 0 \ . \qquad \text{[II.1.10]}$$

Here $j = n+1$ is the dummy library with demand $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$, and $c_{i,n+1} = 0$ for every $i = 1,\ldots,m$.

In the situation in which the total demand exceeds the total supply, we create a dummy store with supply $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$. Here $x_{m+1,j}$ will denote the amount of the book shortage at library j.

In principle, transportation problems can be solved by using the simplex method of linear programming. But the special structure of the transportation problem has given rise to special procedures of which the solutions need less computer time; see Eiselt and von Frajer (1977, Section I.1.2).

II.1.4.2. A related problem

We mention a related transportation problem, which is not a linear programming problem. Again we have m distribution centres (the wholesale dealers in the previous problem) and n destinations (say, local libraries). Let t_{ij} be the shipping time from the i^{th} distribution centre to the j^{th} library. The problem is to minimise

$$T = \max t_{ij} \qquad \text{[II.1.11]}$$

where the maximum is taken over those (i,j) where x_{ij} (the quantity shipped from centre i to library j) is strictly positive.

Such problems can occur in situations in which time is important, e.g. in organising an interlibrary loan (IL) circuit.

II.1.4.3. Assignment problems

Finally, we consider the following *assignment problem*. The head librarian of a big university has to assign n librarians to n branch libraries. Let c_{ij} denote the efficiency factor of assigning librarian j to library i. For example, a person holding a degree in mathematics and in library and information science might be doing well in the library of the Department of Mathematics (high efficiency factor), reasonably well in the Engineering or Physics Department (intermediate efficiency factor) and probably very badly in the library for Oriental Studies (if he or she cannot read Chinese or Japanese). Each person

can be assigned to exactly one library. The problem is to assign the librarians in such a way that the total efficiency is maximised.

We define x_{ij} $\begin{cases} = 1 & \text{if librarian } j \text{ is assigned to library } i; \\ = 0 & \text{otherwise.} \end{cases}$

Since each library is assigned to exactly one librarian, we have

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for} \quad i = 1,\ldots,n \; . \qquad [II.1.12]$$

Similarly, each librarian is assigned to exactly one library :

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for} \quad j = 1,\ldots,n \; . \qquad [II.1.13]$$

The objective is to maximise

$$w = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \; . \qquad [II.1.14]$$

From this formulation we see that the assignment problem is equivalent to a standard transportation problem with n distribution centres and n local libraries, where the supply a_i is 1 for $i = 1,\ldots,n$ and the demand b_j is 1 for $j = 1,\ldots,n$.

II.1.5. Examples

II.1.5.1. A binding problem

A librarian is faced with the following problem : she has at least 500 sets of periodicals that need binding annually. The maximum budget for this is $ 7000/year. She has a choice between two methods of binding :

1) a cheap one, costing $ 10 a piece, which will last on the average 5 years;
2) a more expensive one, costing $ 20 a piece, which will last on the average 15 years.

She has, however, a contract that forces her to take at least 200 cheap ones and 100 expensive ones a year. How many cheap and how many more expensive bindings does this librarian have to order to have periodicals protected for as long as possible?

We put this problem in the following mathematical form. Let x be the number of cheap bindings and let y be the number of expensive ones. As the expensive binding method protects the periodicals three times longer than the cheap ones we want to maximise

$$w = 5x + 15y$$

with the following constraints :

$$x + y \geq 500 ,$$

$$10x + 20y \leq 7000 ,$$

$$x \geq 200 ,$$

$$y \geq 100 .$$

Note that we automatically have $x \geq 0$ and $y \geq 0$. The best solution (see Fig.II.1.4) is to order 300 cheap but less effective bindings and 200 more expensive ones. This will use up the maximum budget of $ 7000 and will provide protection for a total of 4500 years.

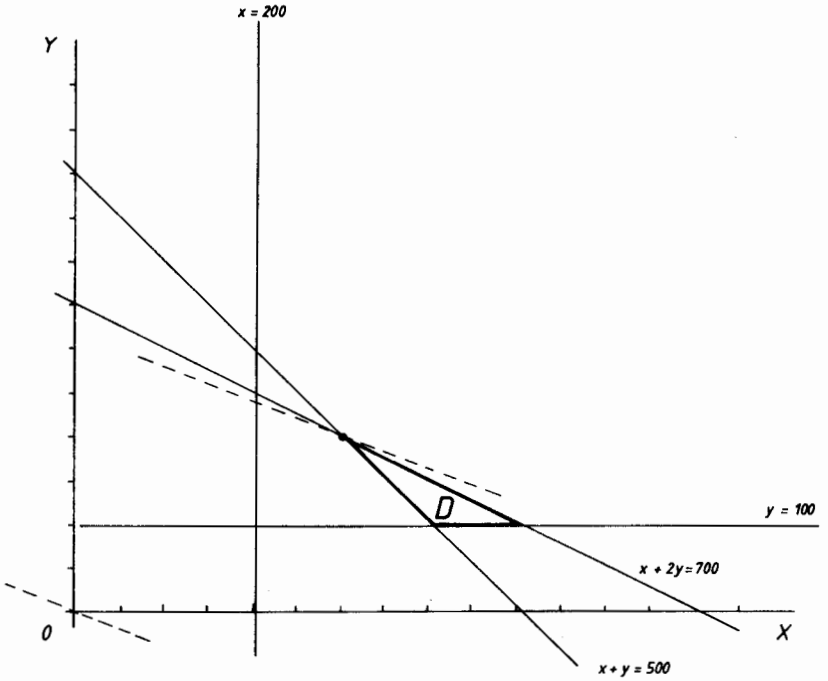

Fig.II.1.4 Graphical solution to the linear problem in Subsection II.1.5.1

II.1.5.2. Allocation of funds to different university departments for purchasing books and journals

We consider the problem of the allocation of funds as described in

Goyal (1973). The main factor in constructing the objective function is the importance weight of each department. Goyal suggests using the following formula :

$$C_i = \left(\frac{S_i + T_i}{2}\right) O_i \quad , \qquad \text{[II.1.15]}$$

where C_i is the importance of department i, S_i a factor measuring the importance which society attaches to the work of the department (e.g. the S_i-factor for the Department of Economy is larger than that for Anthropology), T_i a factor measuring the importance which the university attaches to the work of the department (e.g. a Catholic university may give a high T_i-factor to its Department of Theology; the university may also want to use a more objective factor here, based on some citation measure of the scientific output of this department). The O_i-factor measures the importance due to the size of the department. This number depends on both the number of staff members and the number of students.

Formulation of the problem.

Let the number of departments for which the funds are to be allocated be n and let the total amount of money available for the purchase of books and journals be M. The funds allocated to department i are denoted by X_i and the importance of department i is denoted by C_i, i = 1,...,n. Further, suppose that there is a positive lower limit (L_i) and an upper limit (U_i) for the funds to be allocated to department i. Of course we have $L_i \leq U_i$.

The problem as formulated up to this point can be stated as :

$$\text{maximise} \quad Z = \sum_{i=1}^{n} C_i X_i$$

$$\text{with} \quad X_i \geq L_i \geq 0 \ ,$$

$$X_i \leq U_i \ ,$$

$$\sum_{i=1}^{n} X_i \leq M \ . \qquad \text{[II.1.16]}$$

Furthermore, in actual allocation problems there may be other constraints, termed 'grouped constraints', e.g. the total funds allocated to departments i and j should not exceed U_{ij}, and the total funds allocated to departments k, ℓ and m should not be less than $L_{k\ell m}$. These additional constraints will be described as :

$$X_i + X_j \leq U_{ij}$$

and

$$X_k + X_\ell + X_m \geq L_{k\ell m} \quad . \qquad \text{[II.1.17]}$$

The solution is reached by using a slight variation of the simplex method described in Section II.1.2. Indeed, in Section II.1.2 we have only considered the case where all inequalities are of the same type. Here we have inequalities of the type $\leq$ and of the type $\geq$, but this is not a real impediment. It is sufficient to multiply both sides of the inequality by (-1) to obtain an inequality in the opposite direction.

II.1.6. Notes and comments

The development of linear programming ranks among the most important scientific advances of the mid-twentieth century. It has been estimated that 25 percent of all scientific computation on computers is devoted to the use of linear programming and closely related techniques. Not surprisingly, the development of LP software is an ongoing process. Every large computer manufacturer has provided routines. Current versions can handle more than 16000 equations in an even greater number of variables.

In the field of library and information science LP techniques have been used by Kraft and Hill (1973) and Glover and Klingman (1972) to describe a journal selection model and by Rothenberg and Ho (1977) for the location on campus of information centres, to name a few. Larry Stanfel (1979) used simple linear programming techniques to solve the problem of how to design the best three level hierarchy of libraries. Rush, Steinberg and Kraft (1974) considered the journal disposion problem, developing an integer programming strategy to solve the problem of whether to bind, microcopy or discard back issues of journals in a university library branch system. James and Margareth Bean (1985) studied the problem of scheduling the reference staff of a large academic library where available reference staffing and time flexibility are highly constrained. They modelled this as a special integer program (a so-called multiple choice integer program) and reported that their computer model constructed a schedule which was superior to the one obtained by the manual method and took only 1.2 seconds of CPU time on a large computer.

Egghe (1988a) showed how the solution of a quadratic integer 0-1 program describes the evolution of concentration places in a classification. 'Quadratic' refers here to the objective function which is no longer linear but has become a quadratic function. Zhang (1989) used multi-objective

programming to model resource sharing in a network of libraries and information centres.

Years of experience in the use of the simplex method have revealed that although the number of corner points in the feasible region grows exponentially with respect to the number of variables and the number of constraints, the time needed to reach the optimum only grows linearly with respect to the number of constraints. This is very remarkable in the light of the results of Klee and Minty (1972), who constructed examples in which the time needed by the simplex method actually does grow exponentially. However, this poor behaviour never seems to occur in practice.

Some years ago the Soviet mathematician L.G. Khachian (1979) published his ellipsoidal algorithm for linear programming. Its most important theoretical property is that the time it needs to solve the problem grows as a polynomial with respect to the variables and constraints. Although the ellipsoidal algorithm is consequently theoretically better than the simplex method, due to computational problems it did not constitute a serious challenge to it.

Meanwhile, thanks to work by Borgwardt (1982), Smale (1983) and Adler and Megiddo (1985) among others, mathematicians have shown why the average behaviour of the simplex method is so good.

One of the most recent approaches to linear programming is Karmarkar's algorithm (Karmarkar (1984)). Like Khachian's method, it runs in polynomial time. It has been claimed that numerical results proved this new method to be much faster than the simplex method (Gay et al. (1986)). Experiments by Tomlin (1987) showed that the number of iterations in Karmarkar's method grows slowly with model size. However, vast improvements in speed over the simplex method were not achieved. Another recent approach to linear programming has been developed by Renegar (1988).

II.2. SHORTEST PATH ALGORITHMS

In this section we will discuss some graph-theory methods in operations research. Indeed, many transportation problems, whether one wants to minimise time or distance, can be formulated in a graph-theory framework. We will begin with some preliminaries on graph theory.

II.2.1. Preliminaries on graph theory

A *graph* (or undirected graph) G consists of a set V of vertices (or nodes) and a set E of edges (or arcs) such that each edge $e \in E$ is associated with an unordered pair of vertices. If an edge e is associated with a unique pair of vertices i and j, we write e = (i,j) or e = (j,i). In this context (i,j) denotes an edge rather than an ordered pair.

A directed graph (or digraph) G consists of a set V of vertices and a set E of edges such that each edge $e \in E$ is associated with an ordered pair of vertices.

An edge e = (i,j) in a graph (undirected or directed) is said to be incident on i and j. The vertices i and j are said to be incident on e and to be adjacent vertices. If G is a graph with vertices V and edges E, we write G = (V,E). In this book the sets E and V are always assumed to be finite.

A path from the vertex i to the vertex j is an edge sequence from i to j. The length of this path is the number of distinct edges minus one. So if i and j are adjacent, there is a path of length 2-1 = 1 that joins them. A circuit (or cycle) is a path from i to i. We say that a graph G is connected if, given any distinct pair of vertices i and j, there is a path from i to j.

A weighted graph is a graph in which numbers are associated with the edges. The value w(i,j) associated with the edge (i,j) is referred to as the '*weight*' of (i,j). For example, if we interpret libraries as vertices and the roads between them as edges and if we assign to each road its length, we obtain a weighted graph (see Fig.II.2.1). Weights are often used to represent distances, time or costs.

An important class of graphs are trees. A *tree* is a connected graph not containing any circuits. Typically, a special vertex, called the 'root' of the tree is distinguished. There is precisely one path between any two vertices of a tree. Dendrograms (see Section I.5.4) are examples of rooted trees, where each of the terminal nodes represents an object, non-terminal nodes represent a non-singleton cluster, and the root represents the entire object set.

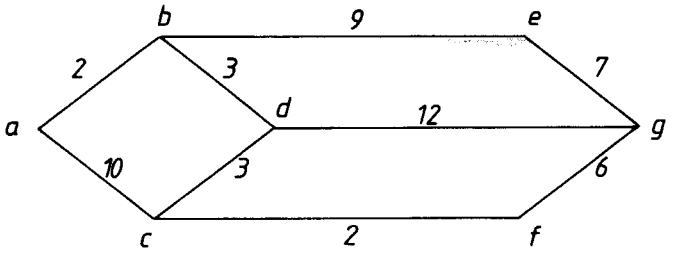

Fig.II.2.1 Weighted (undirected) graph

Graphs are a useful tool for studying spatial relationships within a library. Consider, as an example, the ground plan of a fictitious, though based on an existing, public library, given by Fig.II.2.2a and b. The - undirected - graph of spatial relationships is given by Fig.II.2.3. Nodes are locations and two locations are connected if it is possible to go directly from the first to the second.

E : Main entrance of the building
L : Entrance lobby
W : Women's toilet
M : Men's toilet
LE : Library entrance
CI : Community information
CA : Circulation desk for adults & circulation control
EC : Easy chairs and table where people can wait and sit down to rest
DC : Catalogue for the disc library
CATA : Catalogue for all adult books
DL : Disc library and circulation desk
AF : Stacks : adult fiction
EM : Emergency exit
S : Stairs
PM : Photocopying machine
CATJ : Catalogue for juvenile books
DJ : Documentation for juvenile readers
CDJ : Circulation desk for juvenile readers
J : Stacks : juvenile readers
ANFa,b : Stacks : adult non-fiction
P : Current periodicals
R : Reading room
REG : Registration desk
WO : Working area for library staff
D : Library director's office

What can we learn from this graph? First, we notice that the general design is rather linear (one node after the other) instead of starlike (one central place) or a combination of stars. The circulation desk for juvenile readers occupies a central position, as do the easy chairs in the waiting room in front of the adult circulation desk.

Further on, the library director is not within easy reach for library patrons, and people who visit the library for the first time and want to register have to traverse all sections. More importantly, catalographers and other staff occupied in the working area are at least eight nodes (a path length of seven) away from the nearest exit and eleven nodes (a path length of ten) from the toilets (for which they have to leave the library). Further inconveniences are : the great distance between the adult catalogue and the non-fiction section and the gratuitous division of this non-fiction section.

A totally new design would be required to improve this situation. A partial solution for the library staff would be to construct a second staircase from the community information section (which could easily be removed) to the reading room. This is indicated in dotted lines on Figs. II.2.2 and II.2.3. The reading room and the registration desk would also be made much more accessible this way.

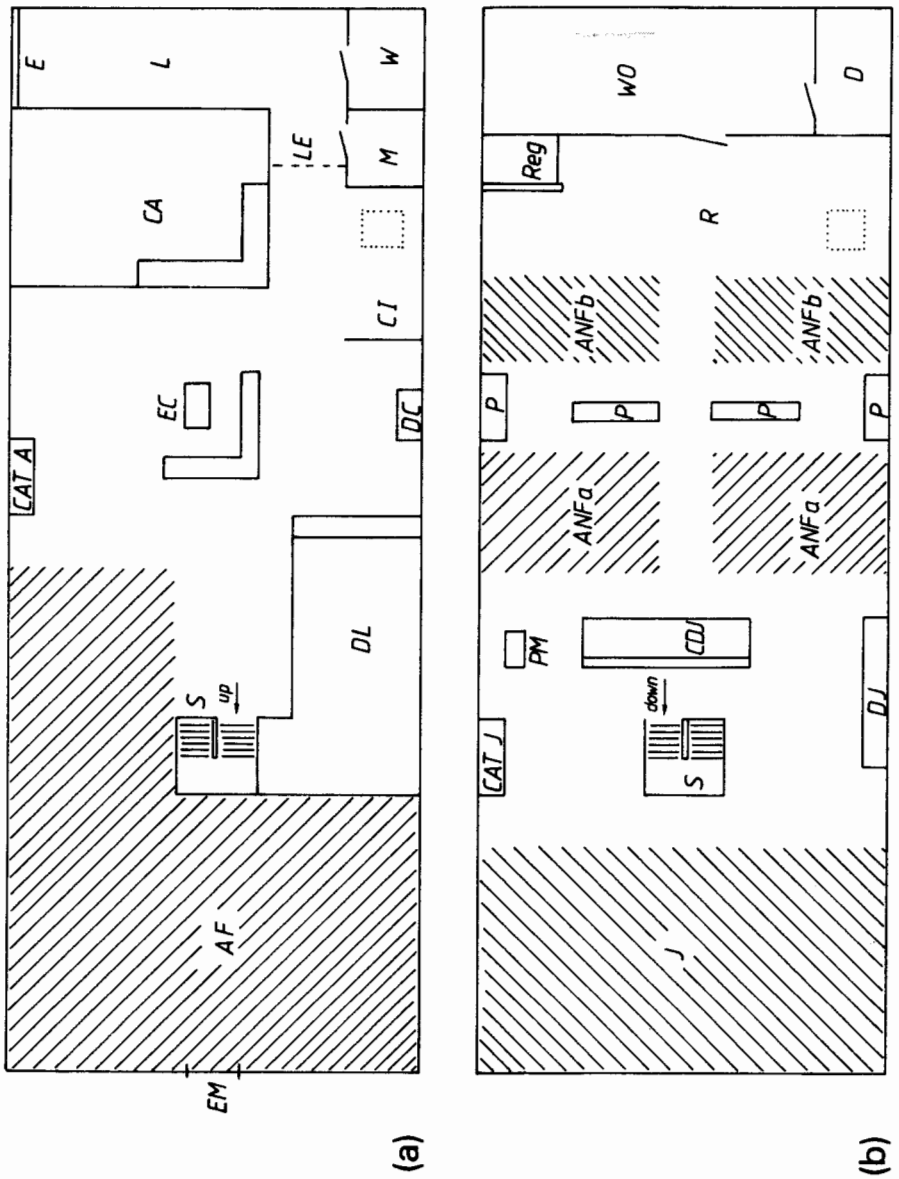

Fig.II.2.2 Ground plan of a public library

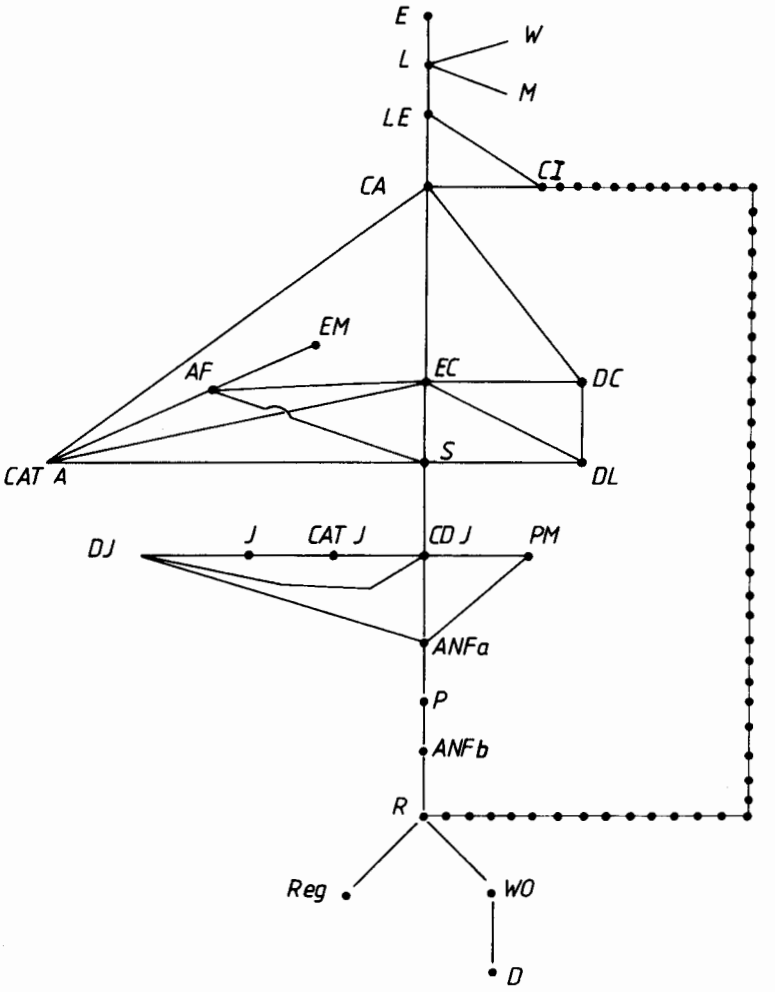

Fig.II.2.3 Graph derived from Fig.II.2.2

II.2.2. Dijkstra's shortest path algorithm

In this section we will study the problem of finding a minimum weight path between two given vertices in a weighted graph. Such a path is referred to as a *shortest path*. *Dijkstra's shortest path algorithm* solves this problem efficiently. This also means that we will be able to solve a 'least-time transportation problem' or a 'least-cost transportation problem' or many similar problems with the same algorithm. Throughout this section G denotes a connected, weighted graph. We further assume that the weights are positive

numbers and that we wish to find a shortest path from a fixed vertex a to a fixed vertex z. Later we will indicate how to find a shortest path from a fixed vertex a to any other vertex in the graph.

Dijkstra's algorithm (Dijkstra (1959)) involves assigning two labels to vertices, which will be denoted L(x) and P(x) (where x denotes a vertex). At any given time, some vertices have temporary labels and others have permanent labels. We will show later that if L(x) is a permanent label of vertex x, then L(x) is the length of a shortest path from a to x. Initially, only the vertex a has a permanent L-label. Each iteration of the algorithm changes the status of one pair of labels from temporary to permanent. The algorithm stops when z receives permanent labels. At this point L(z) gives the length of a shortest path from a to z. The P-labels are used to find the path itself.

We will now present the algorithm.

Dijkstra's shortest path algorithm (Dijkstra (1959)).

Step 1. Initialisation.
Set $L(a) := 0$; for those vertices adjacent to a set $L(x) := w(a,x)$ and $P(x) := a$. For all other vertices set $L(x) := \infty$. Let T be the set of all vertices except a.

Step 2. Check whether the algorithm has found a shortest path. If $z \notin T$: stop. L(z) is then the length of a shortest path from a to z.

Step 3. Get the next vertex.
Choose $v \in T$ with the smallest L-value. If there are ties choose any, but just one. Set $T := T \setminus \{v\}$.

Step 4. Revise labels.
For each vertex $x \in T$ adjacent to v
set $L(x) := \min \{L(x), L(v) + w(v,x)\}$
$P(x) := v$, if L(x) has been changed.

Go to step 2.

If you want to have a shortest path from a to every vertex of G, replace Step 2 by :

Step 2'. Check whether all shortest distances have been found.
If $T = \emptyset$: stop. For every $x \in G$, L(x) is the length of a shortest path from a to x.

Fig.II.2.4 presents a flow chart of this algorithm.

As an example, we will apply this algorithm to the graph of Fig.II.2.1, to find a shortest path from a to g. Every iteration is shown by a state diagram.

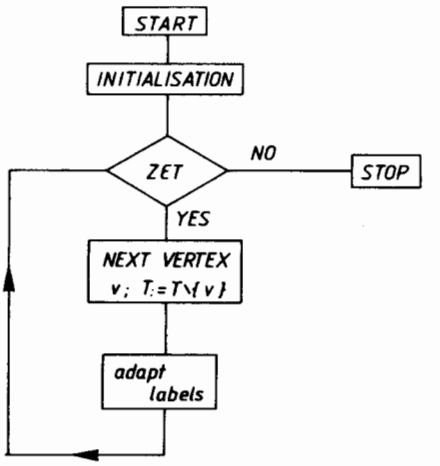

Fig.II.2.4 Flow chart of Dijkstra's algorithm

		T	L	P
Initial state	a	-	0	
	b	*	2	a
	c	*	10	a
	d	*	∞	
	e	*	∞	
	f	*	∞	
	g	*	∞	

* denotes elements that belong to T;
\- denotes elements that do not belong to T.
Elements that do not belong to T have permanent labels

Second state	a	-	0	
	b	-	2	a
	c	*	10	a
	d	*	5	b
	e	*	11	b
	f	*	∞	
	g	*	∞	
Third state	a	-	0	
	b	-	2	a
	c	*	8	d
	d	-	5	b
	e	*	11	b
	f	*	∞	
	g	*	17	d
Fourth state	a	-	0	
	b	-	2	a
	c	-	8	d
	d	-	5	b
	e	*	11	b
	f	*	10	c
	g	*	17	d

Fifth state	a	-	0	
	b	-	2	a
	c	-	8	d
	d	-	5	b
	e	*	11	b
	f	-	10	c
	g	*	16	f
Sixth state	a	-	0	
	b	-	2	a
	c	-	8	d
	d	-	5	b
	e	-	11	b
	f	-	10	c
	g	*	16	f
Seventh state	a	-	0	
	b	-	2	a
	c	-	8	d
	d	-	5	b
	e	-	11	b
	f	-	10	c
	g	-	16	f

As g has got permanent labels, this is the final state.

The shortest path from a to g has a length of 16.

This shortest path is, in reversed order : g - P(g) = f - P(f) = c - P(c) = d - P(d) = b - P(b) = a, or in the right order : a - b - d - c - f - g.

In this particular case we have found all the shortest paths beginning in a (since, in the last state T = ∅). For instance, the shortest path from a to e has length 11 and is given by the sequence a - b - e.

For a mathematical proof that Dijkstra's algorithm finds a shortest path from a vertex a to any other vertex, the reader may consult Dijkstra (1959) or Gibbons (1985).

II.2.3. Applications of Dijkstra's algorithm

As previously noted, obvious applications of Dijkstra's algorithm include all kinds of transportation problems. In this section we will give two less obvious applications.

A. An equipment replacement problem (based on Phillips et al. (1976; p.98))

Most audiovisual equipment requires more maintenance as it ages. By replacing the equipment at frequent intervals, the associated costs could be reduced. But this reduction is achieved at the expense of the increased initial costs incurred every time the equipment is replaced. One of the most important problems faced by management is to decide how often to replace the

equipment so as to minimise the total costs, including the initial cost, the costs of maintenance and running costs. As this can be formulated as a shortest path problem in a directed graph, Dijkstra's algorithm can be used to solve it.

Consider a multimedia centre planning to buy its equipment at the present time. This equipment will certainly be replaced after 4 years, but is it advisable to make replacements in between? Let K_j represent the purchase price of equipment in year j and S_k the salvage value after k years of use. The maintenance and running costs of the equipment during its k^{th} year of operation is c_k. Since costs increase with the age of the equipment, we assume that $c_{k+1} > c_k$ for all k = 1,2,3,4. To formulate the problem of determining the optimal replacement policy as a shortest path problem we construct a directed graph, as shown in Fig.II.2.5.

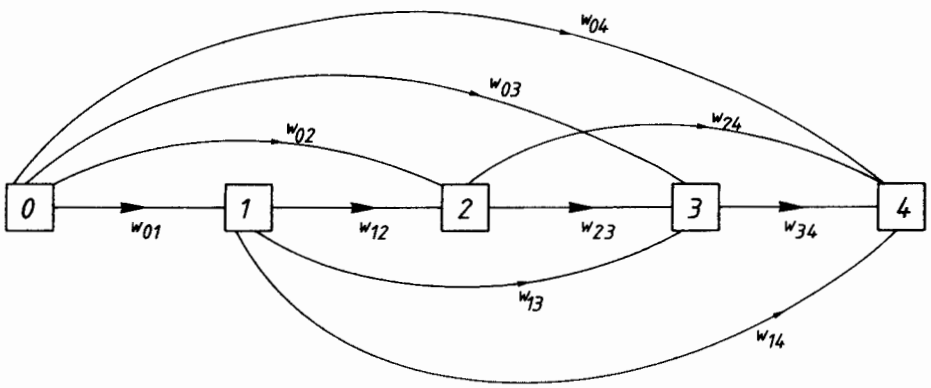

Fig.II.2.5 Directed graph representing an equipment replacement problem

Nodes 0 and 4 represent the beginning and the end of the planning period. Each intermediate node j (j = 1,2,3) represents the end of year j-1 (or the beginning of year j) where an equipment replacement is possible. From every node i there exists a connecting, directed arc to node j only if j > i. This corresponds to the situation in which, having replaced the equipment in year i, the next replacement is only possible in later years.

The distance (weight) between node i and node j is of the form :

$$w_{ij} \begin{cases} = K_i - S_{j-i} + \sum_{k=1}^{j-i} c_k & \text{for } j > i \\ = \infty & \text{for } j \leq i \ . \end{cases} \qquad [II.2.1]$$

The weight function w represents the purchase cost minus any salvage value plus the maintenance and running costs of the equipment. A value of ∞ indicates that there is no arc from i to j.

Every path from node 0 to node 4 in the graph represents a possible replacement policy. For example, the path 0-1-2-3-4 corresponds to replacing the equipment every year, so that the total cost is

$$\left(\sum_{k=0}^{3} K_k \right) - 4\, S_1 + 4\, c_1 \ . \qquad [II.2.2]$$

Another policy is to use the same equipment all four years, which corresponds to the path 0 - 4. The cost of this policy is given by

$$w_{04} = K_0 - S_4 + \sum_{k=1}^{4} c_k \ . \qquad [II.2.3]$$

Thus, determining the shortest path from 0 to 4 is equivalent to finding the minimal cost policy for the equipment replacement problem.

What does this problem look like formulated as a linear programming problem? Let us take $x_{ij} = 1$ if equipment bought in year i is replaced in year j and $x_{ij} = 0$ in all other cases (i,j = 0,...,n). The constraints are :

$$\sum_{i=0}^{n-1} x_{ij} \leq 1 \ ; \quad \sum_{j=1}^{n} x_{ij} \leq 1 \ ,$$

$$\sum_{j=1}^{n} x_{0j} = 1 \ ; \quad \sum_{i=0}^{n-1} x_{in} = 1 \ . \qquad [II.2.4]$$

Finally, the objective function to be minimised is $\sum_{i,j} w_{ij} x_{ij}$.

B. Compact book storage in libraries (Gupta and Ravindran (1974))

Consider the problem of storing books by size. Suppose the heights and the thicknesses of all books in a collection are given. Let the book heights be arranged in ascending order of their n known heights $H_1, H_2, \ldots, H_n$ ($H_1 < H_2 < \ldots < H_n$). Bear in mind that any book of height H_i can be shelved in a shelf of heigth $\geq H_i$. Since the thickness of each book is known, the required length of each height class i can be computed and is denoted by L_i.

If the books are stored upright using only one shelf height (corresponding to the tallest book) for the whole collection, then the total shelf area needed is the product of the total length and the height of the tallest book. Instead, if the collection is divided by height into two or more groups, it can easily be seen that the total shelf area needed will be less than that of the undivided collection.

The cost of constructing shelves of different heights and lengths is given as follows. For each shelf height H_i we set :

K_i = fixed cost independent of the shelf area,

C_i = variable cost per unit area.

For example, let the collection be placed in two different shelves of heights H_m and H_n ($H_m < H_n$), i.e. books of height H_m or less are placed on shelves of height H_m. Then the total cost of shelving the collection will be :

$$(K_m + C_m H_m \sum_{i=1}^{m} L_i) + (K_n + C_n H_n \sum_{i=m+1}^{n} L_i) . \qquad [II.2.5]$$

The problem is to determine the optimal set of shelf heights, and their respective lengths, which will minimise the total shelving cost.

We will show that the compact book storage problem can be formulated as a shortest path problem. Consider a directed graph of (n+1) nodes {0,1,2,...,n}, where the nodes correspond to the various book heights : $H_0,\ldots,H_n$. A weight (distance) function on the set of paths connecting node 0 to node n will be proposed in terms of the shelving cost, where each path is a possible partition of the set of all shelf heights. To make the graph model compatible with the storage problem, the following assumptions are made :

1) $0 = H_0 < H_1 < \ldots < H_n$.
2) From every node i there exists a directed arc to node j, only if $j > i$. This corresponds to the situation in the book storage problem in which, having chosen a shelf of height H_i, the height of the next shelf must be greater than H_i. So, this graph has $n(n+1)/2$ arcs.
3) The weight function between node i and node j is given by :

$$w_{ij} \begin{cases} = K_j + C_j H_j \sum_{k=i+1}^{j} L_k & \text{for } j > i , \\ = +\infty & \text{for } j \leq i . \end{cases} \qquad [II.2.6]$$

We see that, as a consequence of assumptions 1), 2) and 3), finding the shortest path between source node 0 and node n in the above graph is equivalent

to determining the number of different shelves and their respective heights, which minimises the shelving cost for the collection. For example, a minimum path solution of the form 0-3-4-5-n means : to go from node 0 to node n, the shortest route is to use the intermediate nodes 3, 4 and 5. This then says : store all books of height less than H_3 on shelves of height H_3, books of heights between H_3 and H_4 on shelves of height H_4, books of heights between H_4 and H_5 on shelves of height H_5 and the rest on the shelf of height H_n. The heights H_i are not necessarily the exact heights of all the books in the library, but may represent workable selections of them, in which the books are grouped.

A different approach to compact book storage using dynamic programming is given in Leimkuhler and Cox (1964); see also Leimkuhler (1988).

II.2.4. A matricial method of finding the length of the shortest path between each pair of vertices in a weighted graph

II.2.4.1. The method

This method contrasts with Dijkstra's algorithm which we described in Section II.2.2 and which finds the shortest path from a specified vertex to all the others (or to another specified one). Now we study the problem : find the shortest path between *each pair* of vertices in a weighted graph. It stems from Floyd (1962), based on work by Warshall (1962). As this algorithm does more than Dijkstra's, it takes more computer time. Consequently, it should only be used when one is interested in finding *all* the shortest paths in a weighted graph. On the other hand, this algorithm is more efficient than applying Dijkstra's a number of times.

Let G be a graph with n vertices and let W be the matrix for which $W(i,j)$ is the weight of the edge (v_i,v_j). If v_i and v_j are not adjacent, then $W(i,j) = \infty$ and finally $W(i,i) = 0$. Then a series of matrices $W_0, W_1, \ldots, W_n$ are constructed inductively :

$$W_0(i,j) = W(i,j) \ ,$$

$$W_k(i,j) = \min \left(W_{k-1}(i,j), W_{k-1}(i,k) + W_{k-1}(k,j)\right) . \qquad [II.2.7]$$

The matrix W_n provides the desired result.

II.2.4.2. An example

As an example we again consider Fig.II.2.1. We then find the following

series of matrices :

$$W_0 = \begin{pmatrix} 0 & 2 & 10 & \infty & \infty & \infty & \infty \\ 2 & 0 & \infty & 3 & 9 & \infty & \infty \\ 10 & \infty & 0 & 3 & \infty & 2 & \infty \\ \infty & 3 & 3 & 0 & \infty & \infty & 12 \\ \infty & 9 & \infty & \infty & 0 & \infty & 7 \\ \infty & \infty & 2 & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & 12 & 7 & 6 & 0 \end{pmatrix}$$

$$W_1 = \begin{pmatrix} 0 & 2 & 10 & \infty & \infty & \infty & \infty \\ 2 & 0 & 12 & 3 & 9 & \infty & \infty \\ 10 & 12 & 0 & 3 & \infty & 2 & \infty \\ \infty & 3 & 3 & 0 & \infty & \infty & 12 \\ \infty & 9 & \infty & \infty & 0 & \infty & 7 \\ \infty & \infty & 2 & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & 12 & 7 & 6 & 0 \end{pmatrix}$$

$$W_2 = \begin{pmatrix} 0 & 2 & 10 & 5 & 11 & \infty & \infty \\ 2 & 0 & 12 & 3 & 9 & \infty & \infty \\ 10 & 12 & 0 & 3 & 21 & 2 & \infty \\ 5 & 3 & 3 & 0 & 12 & \infty & 12 \\ 11 & 9 & 21 & 12 & 0 & \infty & 7 \\ \infty & \infty & 2 & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & 12 & 7 & 6 & 0 \end{pmatrix}$$

$$W_3 = \begin{pmatrix} 0 & 2 & 10 & 5 & 11 & 12 & \infty \\ 2 & 0 & 12 & 3 & 9 & 14 & \infty \\ 10 & 12 & 0 & 3 & 21 & 2 & \infty \\ 5 & 3 & 3 & 0 & 12 & 5 & 12 \\ 11 & 9 & 21 & 12 & 0 & 23 & 7 \\ 12 & 14 & 2 & 5 & 23 & 0 & 6 \\ \infty & \infty & \infty & 12 & 7 & 6 & 0 \end{pmatrix}$$

$$W_4 = \begin{pmatrix} 0 & 2 & 8 & 5 & 11 & 10 & 17 \\ 2 & 0 & 6 & 3 & 9 & 8 & 15 \\ 8 & 6 & 0 & 3 & 15 & 2 & 15 \\ 5 & 3 & 3 & 0 & 12 & 5 & 12 \\ 11 & 9 & 15 & 12 & 0 & 17 & 7 \\ 10 & 8 & 2 & 5 & 17 & 0 & 6 \\ 17 & 15 & 15 & 12 & 7 & 6 & 0 \end{pmatrix}$$

$$W_5 = \begin{pmatrix} 0 & 2 & 8 & 5 & 11 & 10 & 17 \\ 2 & 0 & 6 & 3 & 9 & 8 & 15 \\ 8 & 6 & 0 & 3 & 15 & 2 & 15 \\ 5 & 3 & 3 & 0 & 12 & 5 & 12 \\ 11 & 9 & 15 & 12 & 0 & 17 & 7 \\ 10 & 8 & 2 & 5 & 17 & 0 & 6 \\ 17 & 15 & 15 & 12 & 7 & 6 & 0 \end{pmatrix}$$

$$W_6 \begin{pmatrix} 0 & 2 & 8 & 5 & 11 & 10 & 16 \\ 2 & 0 & 6 & 3 & 9 & 8 & 14 \\ 8 & 6 & 0 & 3 & 15 & 2 & 8 \\ 5 & 3 & 3 & 0 & 12 & 5 & 11 \\ 11 & 9 & 15 & 12 & 0 & 17 & 7 \\ 10 & 8 & 2 & 5 & 17 & 0 & 6 \\ 16 & 14 & 8 & 11 & 7 & 6 & 0 \end{pmatrix}$$

$$W_7 \begin{pmatrix} 0 & 2 & 8 & 5 & 11 & 10 & 16 \\ 2 & 0 & 6 & 3 & 9 & 8 & 14 \\ 8 & 6 & 0 & 3 & 15 & 2 & 8 \\ 5 & 3 & 3 & 0 & 12 & 5 & 11 \\ 11 & 9 & 15 & 12 & 0 & 13 & 7 \\ 10 & 8 & 2 & 5 & 13 & 0 & 6 \\ 16 & 14 & 8 & 11 & 7 & 6 & 0 \end{pmatrix}$$

Note that all W-matrices are symmetric and have diagonal entries equal to zero. Therefore, it suffices to store the upper triangular part.

II.2.5. The travelling salesperson problem (TSP)

In the *TSP*, the problem is to find the shortest path joining all of a finite set of points of which the distances from each other are given. Note the difference between this and the problem solved by Dijkstra's algorithm : in the TSP one has to visit every point of the set, just as a salesperson visits every town on his/her route. The problem is usually formulated in such a way that one requires the salesperson to return to the point of origin, so that one actually seeks a closed path. It is important to remark here that even in case the salesperson does not return to the point of departure, the simple rule of proceeding from the origin to the nearest point, then to the point nearest to this and so on, (the so-called greedy algorithm) does not generally give the shortest path.

As an example, we consider the following table of distances (representing actual distances in km between ten Flemish university libraries, including the National Science Foundation).

Table II.2.1. Distance table between ten places (in km)

A	A↓									
B	32	←B↓								
C	53	51	←C↓							
D	49	47	4	←D↓						
E	47	44	6	2	←E↓					
F	10	33	48	44	42	←F↓				
G	87	54	82	83	88	86	←G↓			
H	8	30	58	54	52	13	84	←H↓		
I	59	83	63	69	72	58	140	71	←I↓	
J	97	122	103	106	109	95	179	107	48	←J

Using the greedy algorithm, applied 10 times using a different starting point , results in :

start in A : 493 km
B : 418 km
C : 441 km
D : 452 km
E : 453 km
F : 474 km
G : 440 km
H : 475 km
I : 463 km
J : 465 km

This illustrates the fact that this method does not find the shortest circuit : A-F-J-I-C-D-E-G-B-H-A of a length of 402 km.

The TSP can be mathematically formulated as follows : given n points and distances c_{ij} between every two points i and j, determine an ordering $i_1, i_2, \ldots, i_n$ of the n points such that $c_{i_1 i_2} + c_{i_2 i_3} + \ldots + c_{i_n i_1}$ is minimal.

The TSP is a member of the class of so-called NP-complete problems. These are problems for which there is no known algorithm which is polynomially bounded. The term 'polynomially bounded' roughly means that the running time needed to solve this problem is bounded by a polynomial in the length of the input data. Further, if such a polynomially bounded algorithm existed for the TSP, there would be one for each of these NP-complete problems. The question of finding such an algorithm is still unresolved, and most experts believe it will never be found. This belief has led to a variety of computationally affordable approximation techniques.

In 1963 Little et al. (1963) published a landmark paper which first used the term 'branch and bound'. A very simple method was proposed and tested for the TSP. Problems of 40 cities could be solved (exactly) in a few minutes of computer time. For an excellent review of the TSP, see Held et al. (1984).

II.2.6. Notes and comments

Bovet (1986) has shown that Dijkstra's algorithm can be improved if there is a lower bound on the length of the shortest path between every two nodes of a network. When the nodes are places on the map and the weights are actual distances along the road, one can use the length of a straight line between these places for this lower bound.

Based on these ideas and some work of Nicholson (1966), Mohr and Pasche (1988) found a new algorithm which expands the search from the origin and the destination simultaneously. Experiments have shown that this algorithm strictly dominates Bovet's and Dijkstra's and is usually much faster than Nicholson's. Their algorithm can also be implemented on a parallel computer. A survey on the shortest path problem can be found, for example, in Gallo and Pallottino (1986).

In their paper on the optimal storage of books Radhakrishnan and Venkatesh (1978) pointed out that due to the underlying assumptions, Gupta and Ravindran's model for compact storage (Gupta and Ravindran (1974)) is not really optimal. In their paper they present a different model which gives a better solution.

The first statement of the TSP was formulated by Karl Menger in 1930 in connection with a new definition of curve length. It is typically a problem which is easy to state but hard to solve. Indeed, it has become the prototypical hard problem in theoretical computer science.

One of the best heuristic approaches is the Lin-Kernighan algorithm (Lin and Kernighan (1973)). It is reported (Johnson (1987)) that special implementations of this algorithm find routes that are within 2 % of optimal for instances with as many as 50,000 cities.

Litke (1984) reports on a very good approximation technique in the special case of drilling holes in printed circuit boards. Indeed, to drill holes in such a way that the machine has to make the least possible movements, one has to find a path on the circuit board so that each point is visited once and so that the total path length is minimal. This particular algorithm has been used in cases with up to 17,000 points.

Finally, for relatively small problems (say less than 100 cities) there are now programs that routinely find verified optimal solutions in reasonable amounts of time (Padberg and Rinaldi (1987)).

An extension of the TSP is the multiple travelling salespersons problem. In this case a number of salespersons start from and end at the same city. The problem is to minimise the total distance covered, subject to the constraint that each city (apart from the starting place) has to be visited once by one

and only one salesperson. This is a problem confronting a publisher who has a fleet of delivery vans and wants to minimise the total distance travelled, or the total delivery time (in the latter case 'distances' between places A and B are actually times needed to drive from place A to place B). A good solution to this problem can be found in Gavish and Srikanth (1986).

II.3. QUEUEING THEORY

II.3.1. Introduction

The best known queueing situation is that at a ticket window. Here people arrive and want to be served by the person at the window. Other frequently encountered queueing situations involve production lines, the theory of scheduling and transportation and the design of automatic equipment for information handling and data processing. In traditional library science we find queues of patrons waiting at the circulation desk, queues of books waiting to be catalogued and even books on the shelves waiting to be borrowed. To clarify our terminology, we shall talk in terms of a service station where customers (or items) arrive and wait to be served by the server(s). A general queueing situation is depicted in Fig.II.3.1. We will always assume that customers arrive at random.

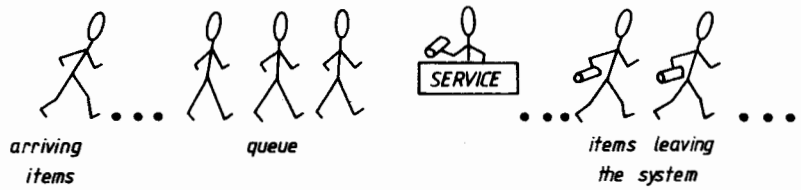

Fig.II.3.1 General queueing situation with one server

Queueing theory provides answers to the following questions :

1) What is the average number of items in the queue ($\bar{N}_q$), the average number of items in service (i.e. being served at the desk), denoted $\bar{N}_s$, and the average number of items in the whole queueing system $\bar{N}$? Of course : $\bar{N} = \bar{N}_q + \bar{N}_s$.

2) What is the expected arrival rate (λ) and the expected service rate (μ)? Here we assume a continuous service (no coffee breaks!).

3) What is the average time spent in the queue ($\bar{T}_q$) or at the desk ($\bar{T}_s$) or in the queueing system ($\bar{T}$), where again : $\bar{T} = \bar{T}_q + \bar{T}_s$?

4) How large is the *utilisation factor* (ρ) : the fraction of the total time during which servers are busy :

$$\rho = \frac{\lambda}{m\mu} \quad , \qquad \text{[II.3.1]}$$

where m is the number of servers?

5) What is the probability of having a queue, or stated otherwise, what is the probability that all servers will be busy when another customer arrives?

A queue may always occur, but when $\lambda > \mu$ we have an unstable system : the queue grows infinitely long unless the management restricts the number of customers in the queue. When $\lambda \leq \mu$ (i.e. the arrival rate is lower than the rate of service), queues may arrive here as well, due to the stochastic nature of the process. Based on the single server queue, we will show that when $\lambda/\mu \geq 0.75$, the situation has already become bad, resulting in customers who are irritated and unsatisfied because of overly long waiting times.

According to *Little's equation* (Little (1961)),

$$\bar{N} = \lambda\bar{T}, \ \bar{N}_s = \lambda\bar{T}_s \ \text{ and } \ \bar{N}_q = \lambda\bar{T}_q \quad . \qquad \text{[II.3.2]}$$

Depending on the order in which arriving customers are served, one distinguishes several queueing disciplines. Waiting customers will usually be served in order of their arrival at the service station. This situation is called '*first come, first served*' (*FCFS*). Two important variants here are '*last come, first served*' (*LCFS*, e.g. in stores) and service in random order (when we consider the entrance of a library as the beginning of a queue, customers will leave the library as if service was in random order). Priority service is also important in applications. In this case customers of a special type (e.g. elderly people) or services of a special type (e.g. short tasks) obtain service before others.

In describing the arrival process it will be assumed that customers always arrive individually. The arrival process is a stochastic process. Let t_n denote the moment of arrival of the n^{th} customer at the service station. The times $\tau_n = t_n - t_{n-1}$, $n = 1,2,\ldots$, with $t_0 = 0$, will be called the 'interarrival times'. These interarrival times are assumed to be independent, identically distributed, positive stochastic variables.

Kendall (1951) introduced a short-hand notation for describing which queueing situation is meant. The notation (A|B|m) consists of three symbols, the first referring to the type of interarrival time distribution A, where $A(t) = P(\tau \leq t)$; the second symbol denotes the service time distribution B, where $B(x) = P(\text{time for one service} \leq x)$; and the third symbol m indicates the number of servers. In this notation, which is now in common use, the symbol

M stands for the negative exponential distribution, E for the Erlang distribution (not used further on in this book; the negative exponential distribution can be considered as a special case of an Erlang distribution), G for a general distribution of a non-negative stochastic variable which is not further specified. For instance, (M|M|m) denotes the *m server queue with a negative exponential interarrival and service time distribution*; (G|G|1) is the most general single server queue. We will see that the M-situation corresponds to arrivals according to a Poisson process.

II.3.2. The (M|M|1) queue

We hypothesise that arrivals are such that the probability that a person will arrive during the interval [t,t+h], where h is small, is proportional with h, say equal to λh, $\lambda > 0$, and independent of t. This hypothesis, being in fact a mathematical formulation of the *randomness* of arrivals, together with some other technical requirements, leads to the result that the arrival process is a Poisson process (cf. Subsection I.2.4.2) (see e.g. Cohen (1969); Theorem 4.4). Then $P_n(t)$, the probability of having n arrivals in a period of length t, is equal to

$$\frac{(\lambda t)^n}{n!} e^{-\lambda t} . \qquad [II.3.3]$$

This is a Poisson distribution with a mean equal to λt. We recall (I.2.4.2) that the observation $\bar{x} \approx s^2$ is a strong indication of having a Poisson distribution.

When t = 1 (e.g. in minutes), then $P_n = \frac{\lambda^n}{n!} e^{-\lambda}$ and then the average of this distribution is equal to λ. This shows that the arrival rate (number of arrivals per unit of time) is given by λ. Hence $1/\lambda$ is the average interarrival time. For the interarrival distribution we find

$$\begin{aligned} A(t) &= P(\tau \leq t) \\ &= 1 - P(\tau > t) \\ &= 1 - P_0(t) \\ &= 1 - e^{-\lambda t} , \end{aligned} \qquad [II.3.4]$$

which is the cumulative negative exponential distribution (justifying the M-notation). Its density function a(t) is : $\frac{d}{dt}(1 - e^{-\lambda t}) = e^{-\lambda t}$.

In this situation we also assume that the probability that a service will end during the interval [t,t+h], where h is small, is proportional to h,

say equal to μh. This leads to $B(t) = 1 - e^{-\mu t}$, which indicates that service times are negatively exponentially distributed. The parameter μ is the average service rate and $1/\mu$ is the average time for one service.

For the (M|M|1) queue one sees (see, for example, Phillips et al. (1976)) the following results. The utilisation factor is the probability of having a queue and is equal to

$$\rho = \frac{\lambda}{\mu} = \frac{\text{average service time}}{\text{average interarrival time}} \quad . \qquad \text{[II.3.1]}$$

This formula shows that once the average service time is at least equal to the average interarrival time, there will certainly be a queue. We further suppose $\rho < 1$. The average number of items in the queue is :

$$\bar{N}_q = \frac{\rho^2}{1-\rho} \quad . \qquad \text{[II.3.5]}$$

The average number of items being serviced is :

$$\bar{N}_s = \rho \quad . \qquad \text{[II.3.6]}$$

Then the average number of items in the system is :

$$\bar{N} = \bar{N}_q + \bar{N}_s = \frac{\rho^2}{1-\rho} + \rho = \frac{\rho}{1-\rho} \quad . \qquad \text{[II.3.7]}$$

Using Little's equation [II.3.2] we find :

$$\bar{T}_q = \frac{\bar{N}_q}{\lambda} = \frac{\rho^2}{\lambda(1-\rho)} = \frac{\lambda}{\mu(\mu-\lambda)} \qquad \text{[II.3.8]}$$

and similarly :

$$\bar{T}_s = \frac{\bar{N}_s}{\lambda} = \frac{\rho}{\lambda} = \frac{1}{\mu} \quad \text{(average service time)} \ ,$$

$$\bar{T} = \frac{\bar{N}}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{(\mu-\lambda)} \quad . \qquad \text{[II.3.9]}$$

Finally, the probability that the time spent in the system will be greater than or equal to t is given by

$$G(t) = e^{-(\mu-\lambda)t} \quad . \qquad \text{[II.3.10]}$$

Fig.II.3.2 shows the relation between ρ and $\bar{N}$ and $\bar{N}_q$, indicating an enormous

increase in inefficiency for $\rho \to 1$.

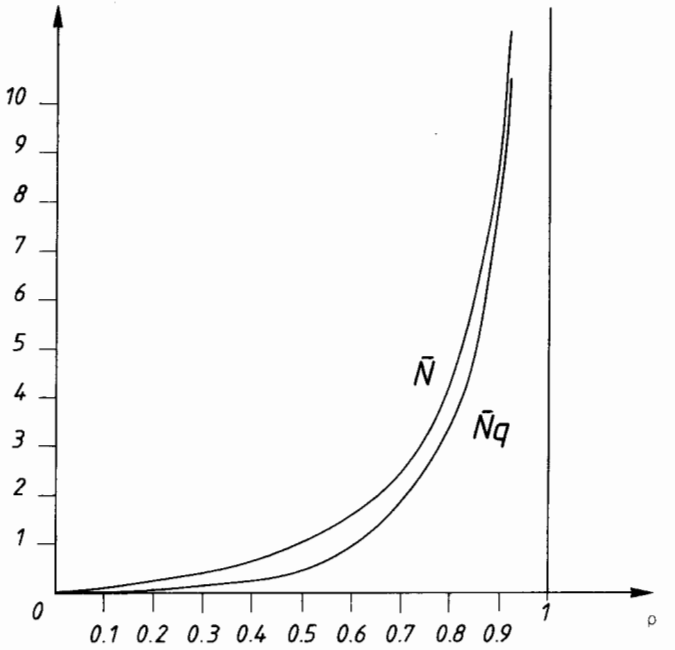

Fig.II.3.2 Graph of $\bar{N}$ and $\bar{N}_q$ as functions of ρ

Some practical calculations

Consider an (M|M|1) queue (say a library circulation desk) with $\lambda = 30$ persons per hour, $1/\mu = 1$ minute. Find the probability of having a queue, the average number of persons in the queue and the average time spent in the queue and at the desk (i.e. the time a customer needs to have books checked out).

With $\lambda = 30$ and $\mu = 60$ (using the same units to measure time!) we find an utilisation factor $\rho = 1/2$. This is the probability of having a queue. $\bar{N}_q = \frac{\rho^2}{1-\rho} = \frac{1/4}{1/2} = \frac{1}{2}$ and $\bar{T} = \frac{1}{60 - 30} = \frac{1}{30}$ hours or two minutes. So everything runs smoothly in this library.

Let us next suppose that an inexperienced person works at the circulation desk. This person has an average service time of 1 minute and 40 seconds. How does this affect the customers? Now $\rho = 0.83$, $\bar{N}_q = 4.17$ and $\bar{T} = 10$ minutes. The probability of having a queue is too large and customers have to wait too long!

As a rule of thumb one can say that for practical situations $\rho = 2/3$ is

an extremum. This means, however, that one third of the time the server is idle. So, to have an efficient queueing system, part of it (the server) must work inefficiently! In practical library work this dilemma is solved by assigning the clerk some additional tasks.

II.3.3. The (M|M|m) queue

Interarrival times and service times stay negatively exponentially distributed, but we next consider m > 1 servers. Note that there is only one queue, as illustrated in Fig.II.3.3.

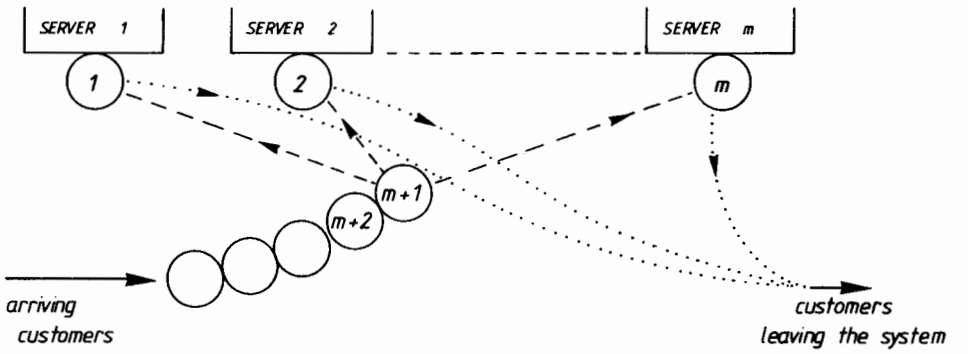

Fig.II.3.3 An (M|M|m) queue

In what follows μ will denote the average service rate per server. Then (for the results in this Section, see Bunday (1986)) the utilisation factor is $\rho = \lambda/m\mu$. The probability of having a queue will be

$$\frac{(m\rho)^m}{K\, m!\, (1-\rho)} \qquad [II.3.11]$$

with

$$K = \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} \right) + \frac{(m\rho)^m}{m!\,(1-\rho)} \quad .$$

Moreover :

$$N_s = \rho m \quad ; \qquad [II.3.12]$$

$$\bar{N}_q = \frac{(m\rho)^{m+1}}{K\, m\, m!\, (1-\rho)^2} \quad ; \qquad [II.3.13]$$

$$\bar{N} = \bar{N}_s + \bar{N}_q = \rho m + \frac{(m\rho)^{m+1}}{K\, m\, m!\, (1-\rho)^2} \quad . \qquad [II.3.14]$$

$\bar{T}_s$, $\bar{T}_q$ and $\bar{T}$ then follow again from Little's equation [II.3.2].

The relation between ρ, $\bar{N}$ and $\bar{N}_q$ is illustrated in Fig.II.3.4. Note the use of semi-logarithmic scales.

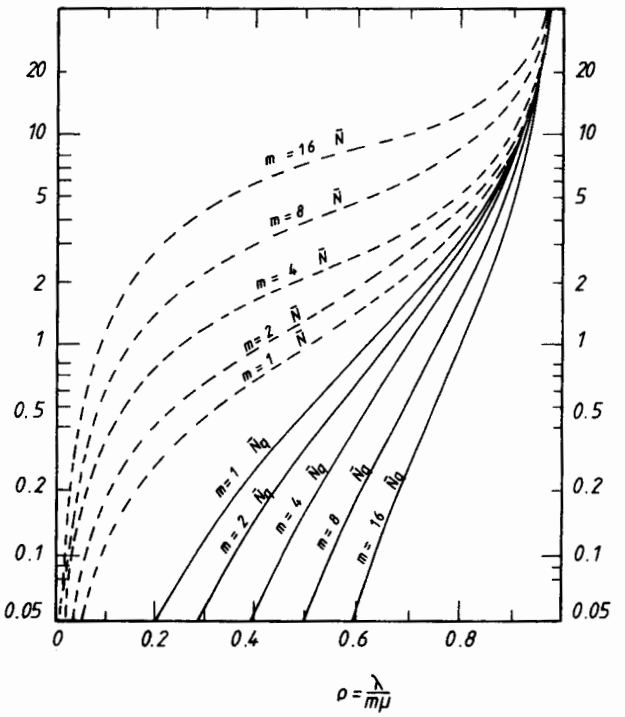

Fig.II.3.4 Graphs of $\bar{N}$ and $\bar{N}_q$ as functions of ρ, for different values of m

II.3.4. Pooled versus separate servers

Considering separate systems first, we take two (M|M|1) queues with $\lambda = 4$ and $\mu = 5$. Then $\rho = 0.8$, $\bar{T}_q = 4/5 = 0.8$ with $\bar{T} = 1$. Pooling yields an (M|M|2) queue with $\lambda = 8$, $\mu = 5$ (per server). Then $\rho = 0.8$, $\bar{T}_q = \bar{N}_q/\lambda$, equals :

$$\frac{1}{8}\left(\frac{(1.6)^3}{(1 + 1.6 + \frac{(1.6)^2}{(2)(0.2)})\ (2)(2)(0.2)^2}\right) = 0.36 \ ;$$

$$\bar{T} = \frac{1}{\mu} + \bar{T}_q = 0.56 \ .$$

This illustrates the fact that *pooling* is much more efficient than separate queues. Applied to library management this means that a central library is preferable to a decentralised one.

In the same context we can also consider one queue with a lower average service time (i.e. service is speeded up). Here we consider an (M|M|1) queue with $\lambda = 8$ and $\mu = 10$. This yields $\rho = 0.8$, $\bar{T}_n = \frac{8}{(10)(2)} = 0.4$ and $\bar{T} = 0.5$. For this situation the total time spent in the system is the smallest. Service is speeded up by using a computerised rather than a manual system.

II.3.5. Notes and comments

Queueing theory can be seen as a branch of applied probability theory. It was systematically studied at the beginning of the century. At that time the technology of automatic telephone exchanges led to a class of mathematical problems that could only be solved by applying probabilistic methods. The Danish mathematician Erlang may be considered the founder of queueing theory.

It can be shown that for the (M|G|1) queue

$$\bar{N} = \frac{2\rho - \rho^2 + \lambda^2 \, \mathrm{Var}(B)}{2\,(1-\rho)} \ , \qquad \text{[II.3.15]}$$

where $\rho = \lambda/\mu$ and Var(B) is the variance of the service time. Further : $\bar{N}_s = \rho$ and $\bar{N}_q = \bar{N} - \bar{N}_s$.

Equation [II.3.15] is known as the Pollaczek-Khintchine equation. As can be seen from this formula, the exact service distribution need not be identified; it suffices to know its mean and variance. As the Pollaczek-Khintchine equation is also independent of the queueing discipline, it is not limited to the FCFS case.

When we apply this formula to the special case where the service time is a constant (Var(B) = 0), we find

$$\bar{N}_q = \frac{\rho(2-\rho)}{2(1-\rho)} - \rho = \frac{\rho^2}{2(1-\rho)} \ .$$

Bearing in mind the analogous results for the (M|M|1) queue

$$\bar{N}_q = \frac{\rho^2}{1-\rho}$$

we see that the average number of customers in the queue would be reduced by

exactly 50 % if the variability were eliminated from the service times. In a sense, half of the queue can be attributed to service-time variability. The other half can be charged to arrival-time variability. Service time variability is reduced, for example, by imposing some 'uniform' checking-out rules on the number of books than can be checked out in one library visit, or by using a 'uniform' checking-out procedure (e.g. a bar-code system gives checking-out times independent of the type of book).

Applying queueing theory to book circulation models will be studied in subsequent chapters, based on Morse (1968). We will also have occasion to consider queues with a finite waiting capacity. A thorough introduction to the single server queue can be found in Cohen (1969). Other books on queueing theory are Morse (1958) and Lee (1966). The optimal location of a server in a network is studied in Berman et al. (1985). Queueing network models were applied by Rouse (1975, 1979) and by Smith and Rouse (1979) to optimise resource allocation within a library. Bookstein (1972) used queueing theory to compare the relative merits of dictionary catalogues (all cards arranged in a single alphabetical order) with split catalogues (subject cards segregated as a separate file).