CHAPTER I
INTRODUCTION

Mathematical programming is concerned with finding optimal solutions rather than good solutions. Though the concept of optimisation is ancient, interest in this area has accelerated enormously with the development of computers and linear programming in the 1940s. Mathematical programming problems are broadly classified into two classes namely;

1. Continuous programming problems
2. Discrete programming problems

The problem of mathematical programming is to find a maximum or a minimum of an objective function whose variables are required to satisfy a set of well defined constraints.

If the objective function is continuous or partially continuous in the variable values that also lie in a feasible region which is compact, then it is a continuous programming problem. This type of problems are solved by some procedures which are based on concepts like continuity, convexity and neighbourhood. Some of them are the Simplex procedure, (Hadely, 1994) Lagrangian multipliers method (Rao, 1984) and Steepest Ascent method (Rao, 1984).

If the decision variables take only discrete values, then it is a discrete programming problem.
If the set of solutions (solutions space) is a finite discrete set not necessarily of variables in the usual sense but may even be of other entities like permutations or combinations, then the problem is one of discrete programming.

If the solution space of a problem consists of combinatorial entities, then it is known as a combinatorial programming problem. Special techniques are available for solving different discrete programming problems. However, no general approach which is suitable for all discrete programming problems seems to be available and methods of finding optimum solutions are largely search methods (c.f. Pandit, 1963).

In the case of continuous programming problems, the existence of a solution may be doubtful, but in the case of combinatorial programming problems, the existence of an optimal solution is not in doubt. However, finding optimal solutions of these problems and recognising them as such is not trivial; one often obtains the optimal solution early in the process of solving the problem but can establish the optimality of the same only after considerable computational time and effort.

The general structure of combinatorial programming problems can be described as follows (Pandit, 1963):

"There is a numerical function defined over the domain of arrangements (permutations or selections) of a set of elements. There are also feasibility criteria. The problem is to find the arrangements which are feasible and which optimise the numerical function."
Following are some of the well-known combinatorial programming problems:

1.1 The Shortest Route Problem

There are \( n \) nodes connected with one another by ‘links’ (or arcs) of given lengths; the problem is to find the shortest route between node pairs in the above network.

1.2 The Travelling Salesman Problem

A set of \( n \) nodes (‘cities’), with distance between every ordered node pair, is given. The problem is to find a least distance route for a salesman who must visit each of these cities starting and ending his journey in the same city.

1.3 The Job Scheduling Problem

There are \( n \) jobs to be completed and each job is processed through each of the ‘\( m \)’ machines in the same order. A job cannot be processed on machine ‘\( j \)’ until it is finished on machine ‘\( j-1 \)’. Let the processing times of each job on each machine be given. The problem is to find a sequence of jobs (defined as a schedule) that minimises the total elapsed time (makespan) to complete all the jobs on all machines.

For many of the combinatorial programming problems the solution space is finite and hence it is theoretically possible to enumerate all the solutions. But often, even for problems of small size, the number of solutions can become very large and grows exponentially with the size of the problem making a complete enumerative solution not practicable.
Hence, there arises a need for implicit enumeration methods. In these methods, though theoretically all the solutions are examined, only a few are explicitly to be examined. The efficiency of the implicit enumeration algorithm basically depends on how quickly it eliminates subsets of solutions as not including any optimal solutions and ‘converge’ fast to an optimal feasible solution. There are two methods in this implicit Enumeration approach. They are 1) Lexicographic Search (Pandit, 1962) and 2) Branch and Bound (Little et al 1963).

1.4 Branch and Bound Algorithms

The Branch and Bound approach to combinatorial programming problems was first developed by Little et. al (1963) in the context of Travelling Salesman problem. Dakin (1965) gave a branch and bound algorithm to solve the general Integer Linear Programming problem. This technique has been successfully applied to solve both pure and mixed integer linear programming problems (Ravi Kumar, 1989).

The Branch and Bound method is characterised by two decision rules. One provides a procedure for the estimation of the upper bound for the value of the objective function at every stage, and the other specifies a ‘choice criterion’ for the selection of branching variable at the stage selected for further partitioning. In all the Branch and Bound algorithms the basic approach is the same, but the technique of selecting the nodes for further branching is different. A few such rules are LIFO (Last in first out), FIFO (first in first out) and LC (least cost) search.
A comprehensive surveys of Branch and Bound technique and approach, particularly in the context of travelling salesman problem and Quadratic Assignment Problem, can be found in Lawler and Wood (1966) and Agin (1966).

1.5 Lexisearch

The Lexisearch approach was first proposed by Pandit (1962) in the context of 'The Loading Problem'. From 1963 onwards this approach has been used to solve various combinatorial programming problems efficiently, e.g., The Assignment Problem (Jain, Mishra and Pandit, 1964), The Facility Location Problem (Das, 1976), The Travelling Salesman Problem (Pandit and Rajbongshi, 1976; Sundara Murthy, 1979; Subramanyam, 1980; Ramesh 1980; Chandrasekhar Reddy, 1987). The Job Scheduling Problem (Gupta , 1967; Rajabongshi, 1982; Bhanumurthy, 1986). In all these problems the lexicographic search was found to be more efficient than the branch and bound algorithms (c.f. Ravi Kumar, 1989).

It may be noted that in a general sense the Branch and Bound approach can be viewed as a particular case of lexicographic search.

1.6 Chapter-wise details

The thesis presents solution of some combinatorial programming problems using Lexisearch approach.
In Chapter II, some of the basic concepts of the Lexicographic Search approach are explained. Also this approach is explained with an illustrative example.

Chapter III discusses the Single Travelling Salesman Problem with the 'Standard Objective Function' of minimising the total tour cost. Some of the available algorithms are reported. Also the same problem, but with some structural constraints on the tour, is discussed. It is also shown that, contrary to expectation, change of scale of the costs $c_{ij}$ can have an impact on the computational time of the lexisearch algorithm.

Chapter IV considers the k-Travelling salesmen Problem with constrained tours and with different types of objective functions. After reviewing the k salesman problem with structural constraints like the number of cities to be visited by each salesman, some non-standard objective functions are discussed.

In this same chapter is presented an algorithm for single salesman problem, where the objective is to minimise the maximum arc length in the tour.

Chapter V considers the problem of finding ‘a maximum node weighted connected sub graph from a node weighted connected graph’. Situations which can be formulated as this problem can be exemplified by the following situation arising in the field of mining:

The ‘area’ to be mined out is divided into blocks, not only on the surface but underground as well. Each block has an identification number and is having a 'net value' (profit or loss as the case may be) associated
with it which is assumed to be known. However, to extract out a block it is necessary to ‘approach’ it through a neighbouring block which is already reached, except when the block of interest is a surface block. Cost of approaching a block is negligible, at least as compared to the ‘value’ of the block, and is ignored. The problem is to select the set of blocks which are ‘connected’ and have the total of their values a maximum.

The problem can be stated formally in Graph-theoretic terminology as follows:

Let $G = (A, E ; V)$ be a symmetric connected graph with $A = \{1,2,...,n\}$ as its node set, $E$ the edge set while $V = \{v_1,v_2,...,v_n\}$ is the set of values associated with the corresponding nodes. Let $S$ be a subset of the node set $A$ and let $G(S, E_S ; V_S)$ be a connected sub graph with $E_S$ and $V_S$ as the corresponding edge set and value set respectively.

The value of this sub graph is the total of the values of the nodes in $S$ and is denoted by $\text{Val}(G(S, E;V)) = \sum_{i \in S} v_i$.

The problem is to select a connected sub set $S$ of $A$ such that it will have a value not less than the value of any other connected subset $S$ of $A$.

It may be noted that unlike in the travelling salesman problem, here, the edges of the graph will not have any value attached to them but the nodes will have values $v_i$ positive or negative as the case may be.

This problem was posed in 1988 by Caccetta and Giannini. They proposed a graph theoretical approach to solve the problem. They
have suggested ‘spanning trees’ as a tool for solving the problem, but have not developed a procedure to solve the problem completely. However, in Chapter V of the present thesis, a procedure is developed to reduce where possible, the effective size of the problem by ‘merging’ appropriate blocks into connected superblocks. Further, a lexisearch algorithm to solve this problem fully is presented.

A number of randomly generated problems, of varying sizes and varying levels of connectivity have been solved by this approach and the relevant computational experience is reported.

The Quadratic Assignment problem is studied in Chapter VI. The problem can be stated as follows:

The \( n \) facilities are to be assigned to \( n \) locations. Let \( f_{ik} \) be the flow between \( i^{th} \) facility to \( k^{th} \) facility and \( d_{lj} \) be the distance between \( j^{th} \) location and \( l^{th} \). Here the objective is to assign the facilities to the locations such that the sum of pairwise interactions among the facilities weighted by the distance between their locations is minimised.

This problem can be stated formally as follows:

Minimise \( \sum f_{ik} d_{lj} x_{ij} x_{kl} \)
subject to \( \sum x_{ij} = 1 \) \( i=1,2,...,n \)
\( \sum x_{ij} = 1 \) \( j=1,2,...,n \)
\( x_{ij} = 0 \) or \( 1 \)

In this chapter, some of the 'exact' and 'heuristic' algorithms available in the literature are reported. In particular, the Lexisearch
approach to this problem as developed by Das (1976) is also summarised. A new lexicographic search approach - with a different way of defining the alphabet table and the corresponding bounding algorithm is presented along with a report on relevant computational experience.

The thesis concludes with some general observations on the problems considered in the thesis. Some interesting open problems are also presented.