Imperial College of Science and Technology

(University of London)

Department of Management Science

ALGORITHMS FOR LARGE SCALE SET COVERING PROBLEMS

by

José Paixão

A thesis submitted for the degree of

Doctor of Philosophy of the University of London

and for the

Diploma of Imperial College

December 1983

ABSTRACT

This thesis is concerned with the set covering problem, (SCP), and in particular with the development of a new algorithm capable of solving large-scale SCPs of the kind found in real life situations.

The set covering problem has a wide variety of practical applications such as crew scheduling, vehicle dispatching, facility location, information retrieval, political districting, design of switching circuits and others. A common feature of most of these applications is that they yield large and sparse SCPs normally with hundreds of rows and thousands of columns.

Despite the large amount of research that has been done on the set covering problems in the last two decades, it was only recently that reasonably large SCPs have been solved. In this thesis we present an algorithm capable of solving problem of this size more efficiently, and a test problem with 400 rows and 4000 columns is solved. This is by far the largest SCP reported solved in the literature.

The method developed in this thesis consists of a combination of decomposition and state space relaxation which is a technique recently developed for obtaining lower bounds on the dynamic program associated with a combinatorial optimization problem. The large size SCPs are decomposed into many smaller SCPs, which are then solved or appromixated by state space relaxation (SSR). Before using the decomposition and SSR, reductions both in the number of columns and the number of rows of the problem are made by applying a procedure combining preliminary tests, heuristic methods, lagrangean relaxation and linear programming.

I dedicate this thesis to

Graca, Sofia and Filipe.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor, Dr. Nicos Christofides of the Department of Management Science, Imperial College, for his help and advice throughout the course of this research.

I would like to express my gratitude to the DEIOC da Universidade de Lisboa, (Department of Applied Mathematics, University of Lisbon), for being given the opportunity of carrying out this work. The financial support from the Gulbenkian Foundation and the INIC (Instituto Nacional de Investigacao Cientifica) is very much appreciated. Also, it is with great honour that I acknowledge the award received from the 'Overseas Research Students Award Scheme'.

Many of my colleagues and friends have made my task easier; it is impossible to mention all their names, but, I would like to thank Dr Nick Bate for his help in the writing of this thesis, and my close friends Eng^{os} Manuel Heitor and José Franca who in their own way made possible the completion of my work.

Finally, a special word for my wife, Graca, without whose continual encouragement this work would not have been possible.

Page

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 DEFINITIONS

Consider a set $M=\{1,2,...,m\}$ and a family of subsets of M, $M=\{M_j \subseteq M, j \epsilon N\}$, with N a finite set. A subset $N^* \subseteq N$ is said to be a cover of M if $\cup_{j \epsilon N^*} M_j = M$. Let a cost $c_j > 0$ be associated with every $j \epsilon N$ and the total cost of the cover N^* be $\Sigma_{j \epsilon N^*} c_j$. The set covering problem, SCP, consists of finding the minimum cost cover for M and can be formulated as an integer program :

$$
\begin{aligned}
(SCP) \qquad & \min \ \Sigma_{j \epsilon N} c_j x_j \\
& \text{st.} \ \Sigma_{j \epsilon N} a_{ij} x_j \geq 1 \qquad (i \epsilon M) \\
& \quad x_j = 0 \text{ or } 1 \qquad (j \epsilon N)
\end{aligned}
$$

where $a_{ij}=1$ if $i \epsilon M_j$; $a_{ij}=0$, otherwise.

A cover is said to be prime if does not contain properly any other cover. When the costs c_j are all equal to 1 the SCP is called the unicost set covering problem.

In this thesis we will identify the set covering problem with its matrix representation and, hence, the elements of M are designated rows while the elements of N are mentioned either as columns or as variables. The subset $M_j \subseteq M$ is identified as the index set of rows covered by column index j, ie. $M_j=\{i \epsilon M : a_{ij}=1\}$. For a particular row i, the set of column indices j that cover the row i, is denoted by N_i, ie $N_i=\{j \epsilon N : a_{ij}=1\}$. The matrix $A=[a_{ij}]$, $(i=1,2,...,m; j=1,2,...,n)$, is called the constraint matrix with a_i and a^j representing, respectively,

the ith row and jth column of A. Finally, we will designate by density of the SCP the ratio $(\Sigma_{i,j}a_{ij})/(m*n)$.

A cover N^* is a feasible solution to the integer program SCP, that is $\Sigma_{j\epsilon N^*}a_{ij}x_j \geq 1$ for all $i\epsilon M$. When N^* is a prime cover no variable x_j can be removed from N^* without violating a constraint.

If a disjoint cover, or partition, N^* (ie, $M_j \cap M_k = \Phi$ for all $j,k\epsilon N^*$) is required, the problem is designated the set partitioning problem, SPP. In this case, the inequalities are replaced by equalities in the integer programming formulation given above.

The SCP and SPP are closely related to each other and both have a wide field of applications. Surveys of the SCP are given in *Garfinkel [75], Gondran [92] and Christofides and Korman [51]*; a survey for the SPP is given in *Balas and Padberg [16]*.

A list of applications of the SCP is given in the next section. There follows a discussion on the relation of the SCP to other mathematical problems, namely the SPP, the general integer program, the graph covering problem and other graph theory problems. Some theoretical results that have been obtained for the SCP are outlined in section 1.4. The final section in this Chapter is concerned with a brief survey of the algorithms developed for the SCP.

Chapter 1

1.2 APPLICATIONS OF THE SCP

1.2.1 Airline Crew Scheduling

The idea of using set partitioning or set covering for crew scheduling goes back to at least 1961 (*Spitzer [163]*). The airline crew scheduling problem consists of finding the cheapest crew schedule that covers all flight legs. Depending on whether or not crew members are allowed to be passengers on certain flights , optimal covers or partitions yield optimal schedules. The columns of the SCP or SPP represent sets of flight legs that can be done by a single crew with a cost c_j , and the rows stand for the flight legs (*Arabeyre et al. [2], Baker et al. [4], Boder [40], Marsten et al. [130], Marsten and Shepardson [131], Rubin [153],[154]*).

Other scheduling problems have been solved by using SCP's, such as mass transit crew scheduling (*Parker and Smith [140]*), bus scheduling (*Heurgon [104], Gavish et al. [80], Shepardson and Marsten [162]*) and manpower scheduling (*Tibrewala et al. [169]*). A special type of SCP appears in some personnel scheduling problems (*Baker [6],[7], Bartholdi [21], Bartholdi et al. [22]*).

1.2.2 Vehicle Dispatching

The truck delivery problem was first formulated as a SPP by *Balinski and Quandt [19]*. Given a set of locations for the customers and a single depot it is required to find the minimal cost routing among the possible routes such that all customers are visited and the number of routes does not exceed the number of available vehicles. Hence, the rows in the SPP correspond to the customers and the columns stand for feasible routes starting and ending at the depot (*Pierce [142], Christofides [49], Bodin et al. [41]*).

1.2.3 Location of Emergency Facilities

The unicost SCP has been used for solving the problem of locating the minimum number of emergency facilities (eg. fire stations, ambulances, police stations) such that they cover every point of a limited area within a fixed distance (or time). Columns of the SCP represent the feasible locations for the facilities and the rows correspond to the areas to be covered (*Berlin and Liebman [36], Bilde and Krarup [39], Church and Revelle [53], Revelle et al. [147], [148], Revelle and Swain [146], Schreuder [159], Toregas and Revelle [172],[173], Toregas et al. [171], Walker [175]*). A location problem using the SCP with additional constraints on the maximum number of facilities that can be located is given in *Christofides [48]*. A dynamic version of the SCP applied to facility location is discussed in *Gunawardane [98]*.

1.2.4 Information Retrieval

Consider the problem of retrieving information from n files where the jth file is of length c_j ($j=1.2,...n$). Suppose that m requests for information are received with $a_{ij}=1$ if the ith unit of information is in file index j. The optimal solution for the resulting SCP gives the minimum length set of libraries needed to access all the information (*Day [59]*).

1.2.5 Political Districting

Garfinkel and Nemhauser [77] used set partitioning for grouping basic population units (eg. counties) in order to form a fixed number of districts. Representing by I the set of basic population units, a subset P_j of I is a column of the SPP if the population units in P_j form a district that meets requirements on total population, contiguity, compactness, and so forth. If c_j is some measure of the unacceptability of district j then the optimal solution of the SPP with the additional constraint on the number of districts formed, gives an optimal districting plan.

A similar model for health district definition is given in *Ghiggi et al. [85]*, and another

application of partitioning in regional planning is presented by *Corley and Roberts [55]*.

1.2.6 Minimising Boolean Expressions

One of the first formalizations of the SCP appears to have been made for finding minimal representation for logical functions related to the problem of designing optimal switching circuits (*Breuer [43], Gimpel [86], Rutman [155], Steinberg [165]*).

Given a logical expression E in a disjunctive form one need only consider the set P of prime implicants of E. A simplest form for E is obtained through the minimum cardinality set $P^* \subseteq P$ such that every term of E be implied at least by one prime implicant in P^*. The optimal answer is given by a unicost set covering where the columns are elements of P and the rows stand for terms of E (*Hammer and Rudeanu [100]*).

1.2.7 Other Applications

Some other applications of the SCP or SPP reported in the literature include assembly line balancing (*Freeman and Jucker [69], Salveson [157], Steinman and Schwin [166]*), the calculation of bounds in general integer programs (*Christofides [46]*), network planning (*Crowston [56]*), network attack and defence (*Bellmore et al. [32], Bellmore and Ratliff [33]*).

Chapter 1

1.3 PROBLEMS RELATED TO THE SCP

1.3.1 SCP and SPP

The only difference between the SCP and the SPP is on the type of the constraints - inequalities for the SCP, equalities for the SPP. Moreover, the SPP can be converted into a SCP (*Lemke et al. [126]*), and, conversely, the SCP can be transformed into a SPP (*Korman [123]*). However these transformations are not symmetrical.

The SPP can be easily converted into a SCP without changing the constraint matrix. This can be done by adding slack variables y_i ($i \epsilon M$) with a suitable large cost $L > \Sigma_{j \epsilon N} c_j$ and , then, the SPP becomes equivalent to :

$$
\begin{aligned}
(1.1) \qquad & \min \ \Sigma_{j \epsilon N} c_j x_j + L * \Sigma_{i \epsilon M} y_i \\
& \text{st.} \ \Sigma_{j \epsilon N} a_{ij} x_j - y_i = 1 \qquad (i \epsilon M) \\
& \quad x_j = 0 \text{ or } 1 \qquad (j \epsilon N) \\
& \quad y_i \geq 0 \qquad (i \epsilon M)
\end{aligned}
$$

Since L is chosen to be arbitrarily large , $y_i = 0$ ($i \epsilon M$) in the optimal solution of problem (1.1) which, therefore, is the optimal solution of the SPP. Now, using $y_i = \Sigma_{j \epsilon N} a_{ij} x_j - 1$, the problem (1.1) is rewritten as :

$$
\begin{aligned}
(1.2) \qquad & \min \ \Sigma_{j \epsilon N} [(\Sigma_{i \epsilon M} a_{ij}) * L + c_j] x_j - L * m \\
& \text{st.} \ \Sigma_{j \epsilon N} a_{ij} x_j \geq 1 \qquad (i \epsilon M) \\
& \quad x_j = 0 \text{ or } 1 \qquad (j \epsilon N)
\end{aligned}
$$

which is a SCP with the same constraint matrix as the original SPP.

Transforming the SCP into a SPP needs modifications in the constraint matrix. Each column a^j is split into $\#M_j$ columns, each one of them with the same cost c_j. The first generated column, $a^{j,1}$, is equal to a^j; the second, $a^{j,2}$ is equal to a_j with the first 1 replaced by 0; the

rth column derived from column j is equal to a^j with the first (r-1) elements of a^j set equal to 0. Hence, the modified matrix has $\Sigma_{j\epsilon N}\#M_j$ columns which is the number of nonzero entries in the original constraint matrix. The constraints are made equalities and then the resulting SPP is equivalent to the SCP from which it was derived.

Although some dominance tests (*Garfinkel and Nemhauser [78]*) can be applied to reduce the size of the resultant SPP, this is still too large for practical examples. In fact, it is not clear when these transformations are worthwhile and in practice this method has not been used.

1.3.2 The SCP and the general Integer Program

Zorychta [185] shows that a 0-1 integer program can be converted to a SCP in a number of steps proportional to mn^2, where m and n are respectively the number of rows and the number of columns for the original problems.

Consider the 0-1 linear integer programming problem, BLP, defined as follows :

$$\begin{aligned} (\text{BLP}) \qquad & \min \ \Sigma_{j\epsilon N} c_j x_j \\ & \text{st.} \ \Sigma_{j\epsilon N} a_{ij} x_j = b_i \qquad (i\epsilon M) \\ & \quad x_j = 0 \text{ or } 1 \qquad (j\epsilon N) \end{aligned}$$

with $c_j \geq 0$, $a_{ij} \geq 0$ and $b_i \geq 0$ for all $i\epsilon M$ and $j\epsilon N$. Note that any integer program can be transformed into a BLP.

Firstly, an integer program is easily converted to a 0-1 problem doing the following substitution :

$$(1.3) \qquad x_j = \Sigma_{k=1}^{s(j)} 2^{k-1} x_j^k \qquad (j\epsilon N)$$

with $x_j^k = 0$ or 1 and s(j) the least integer such that $2^{s(j)} > u_j$, u_j an upper bound on the

Chapter 1

value of x_j.

Next, if there exists a negative coefficient, say $a_{ij}<0$, then replacing x_j by $1-y_j$ in the constraint, the equality can be rewritten as :

$$
\begin{aligned}
(1.4) \qquad & \Sigma_{k\neq j} a_{ik}x_k + a_{ij}(1-y_j) = b_i \quad \geq 0 \\
& \Sigma_{k\neq j} a_{ik}x_k + (-a_{ij})y_j = b_i - a_{ij} \geq 0 \\
& \Sigma_{k\neq j} a_{ik}x_k + a'_{ij}y_j = b'_i \quad \geq 0
\end{aligned}
$$

where $a'_{ij}=-a_{ij}>0$ and $b'_i=b_i-a_{ij}>0$. Also, an additional constraint $x_j+y_j=1$ has to be added to the set of constraints for the integer program.

Example 1.1

We will illustrate the above transformations for the integer program :

$$
\begin{aligned}
\min \quad & 2x_1 - 3x_2 + x_3 \\
\text{st.} \quad & 2x_1 - x_2 + x_3 = 3 \\
& -x_1 + 2x_2 + x_3 = 0 \\
& 0 \leq x_1 \leq 2 \\
& 0 \leq x_2, x_3 \leq 1 \\
& x_1, x_2, x_3 \text{ integers}
\end{aligned}
$$

First, the problem is transformed into a 0-1 integer problem replacing x_1 by $x_1^1+2x_1^2$:

$$
\begin{aligned}
\min \quad & 2x_1^1 + 4x_1^2 - 3x_2 + x_3 \\
\text{st.} \quad & 2x_1^1 + 4x_1^2 - x_2 + x_3 = 3 \\
& -x_1^1 - 2x_1^2 + 2x_2 + x_3 = 0 \\
& x_1^1, x_1^2, x_2, x_3 = 0 \text{ or } 1
\end{aligned}
$$

Now, the coefficients are all made non-negative by introducing the complementary variables y

$$
\begin{aligned}
\min \quad & 2x_1^1 + \quad + 4x_1^2 + \qquad + 3y_2 + x_3 - 3 \\
\text{st.} \quad & 2x_1 + \quad + 4x_1^2 + \qquad + y_2 + x_3 = 4 \\
& \qquad y_1^1 + \quad + 2y_1^2 + 2x_2 + \quad + x_3 = 3 \\
& x_1^1 + y_1^1 \qquad\qquad\qquad\qquad = 1
\end{aligned}
$$

$$
\begin{aligned}
& x_1^2 + y_1^2 \qquad\qquad = 1 \\
& \qquad x_2 + y_2 \qquad = 1 \\
& x_1^1, y_1^1, x_1^2, y_1^2, x_2, y_2, x_3 = 0 \text{ or } 1
\end{aligned}
$$

Chapter 1

Simplifying the notation (by setting $z_1=x_1^1$, $z_2=y_1^1$, $z_3=x_1^2$, $z_4=y_1^2$, $z_5=x_2$, $z_6=y_2$, $z_7=x_3$), the original problem is equivalent to :

$$\begin{aligned}
&\min\ 2z_1+ \quad +4z_3+ \qquad +3z_6+z_7\ -3\\
&\text{st.}\ 2z_1+ \quad +4z_3+ \qquad + z_6+z_7\ =4\\
&\qquad + z_2+ \quad +2z_4+2z_5+ \quad +z_7\ =3\\
&\qquad z_1+ z_2 \qquad\qquad\qquad\quad =1\\
&\qquad\qquad z_3+z_4 \qquad\qquad\quad =1\\
&\qquad\qquad\qquad\quad z_5+ \quad z_6 \qquad =1\\
&z_1,z_2,z_3,z_4,z_5,z_6,z_7=0 \text{ or } 1
\end{aligned}$$

In *[185]* it is shown that the BLP is equivalent to an unconstrained integer program, UIP, defined as follows :

$$\text{(UIP)} \qquad \min\ \Sigma_{j\epsilon N}c_jx_j+\Sigma_{i\epsilon M}k_i(\Sigma_{j\epsilon N}a_{ij}x_j-b_i)^2$$
$$x_j=0 \text{ or } 1$$

with the k_i ($i\epsilon M$) chosen to be greater than a positive value k dependent on the original problem.

The objective function of UIP can then be rewritten as:

$$\text{(1.5)} \qquad \min\ \Sigma_{j\epsilon N}[c_j+\Sigma_{i\epsilon M}k_i(a_{ij}^2-2b_ia_{ij})]x_j+\ 2\Sigma_{j\epsilon N}\Sigma_{k>j}(\Sigma_{i\epsilon M}k_ia_{ik}a_{ij})x_kx_j\ +\ \Sigma_{i\epsilon M}k_ib_i^2$$

Now, performing the substitution :

$$\text{(1.6)} \qquad x_{kj}=x_kx_j \text{ for } (k,j)\epsilon U=\{(p,q) : p\neq q \text{ and } \Sigma_{i\epsilon M}k_ia_{ip}a_{iq}>0\}$$

one obtains a new BLP which is proved in *[185]* to be equivalent to UIP :

$$\text{(1.7)} \qquad \min\ \Sigma_{j\epsilon N}d_jx_j\ +\ 2\Sigma_{j\epsilon N}\Sigma_{k>j}w_{kj}x_{kj}$$

$$\begin{aligned}
&\text{st.}\ x_k+x_j-1\leq x_{kj} && ((k,j)\epsilon U)\\
&\qquad x_{kj}=0 \text{ or } 1 && ((k,j)\epsilon U)\\
&\qquad x_j\ =0 \text{ or } 1 && (j\epsilon N)
\end{aligned}$$

where

(1.8) $d_j = c_j + \Sigma_{i \epsilon M} k_i (a_{ij}^2 - 2b_i a_{ij})$

$w_{kj} = \Sigma_{i \epsilon M} k_i a_{ik} a_{ij}$

Finally, the substitution

(1.9) $x_j' = 1 - x_j \qquad (j \epsilon N)$

yields the SCP :

(1.10)
$$\begin{aligned} \min \quad & \Sigma_{j \epsilon N} d_j x_j + \Sigma_{(k,j) \epsilon U} w_{kj} x_{kj} \\ \text{st.} \quad & x_k' + x_j' + x_{kj} \geq 1 \qquad ((k,j) \epsilon U) \\ & x_{kj} = 0 \text{ or } 1 \qquad ((k,j) \epsilon U) \\ & x_j' = 0 \text{ or } 1 \qquad (j \epsilon N) \end{aligned}$$

Hence, the general integer program is equivalent to a SCP with a particular constraint matrix structure. The matrix of the inequalities is of the form $M=(S,I)$ where S (corresponding to the variables x_j', $j \epsilon N$), is a matrix with exactly two entries in each row and I is a unit matrix corresponding to the variables x_{kj}, $(k,j) \epsilon U$. The number of rows in M is equal to the number of elements in the set U and, therefore, is at most equal to $n(n-1)/2$, with n the number of columns of the original problem. The number of columns of the resulting SCP is exactly equal to the number of rows plus n.

Example 1.2

For the 0-1 problem obtained from the integer linear program of example 1.1, the equivalent UIP is :

$$\begin{aligned} .\min\ & (2-12k_1-k_3)\, z_1 + (-5k_2-k_3)\, z_2 + (4-16k_1-k_4)\, z_3 + \\ & + (-8k_2 - k_4)\, z_4 + (-8k_1-k_5)\, z_5 + (3-7k_1-k_5)\, z_6 + \\ & + (1-15k_1 - 8k_2)\, z_7 + 2k_3\, z_1\, z_2 + 16\, k_1\, z_1\, z_3 + \\ & + 4k_1\, z_1\, z_6 + 4k_1\, z_7 + k_4\, k_2\, z_2\, z_5 + \\ & + 2\, k_2\, z_2\, z_7 + 2k_4\, z_3\, z_4 + 8k_1\, z_3\, z_6 + 8k_1\, z_3\, z_7 + 8k_2\, z_4\, z_5 + \\ & + 4k_2\, z_4\, z_7 + 4k_2\, z_5\, z_7 + 2k_5\, z_1\, z_6 + 2k_1\, z_6\, z_7 - \\ & - 16k_1 - 9k_2 - k_3 - k_4 - k_5 \end{aligned}$$

$$z_j = 0 \text{ or } 1 \quad (j=1,2,\ldots,7)$$

Chapter 1

Setting $k_i=1$, $i=1,2,...,5$, the SCP which is equivalent to the BLP is then :

$$\begin{aligned}
&\text{. min } 11z_1+6z_2+13z_3+9z_4+9z_5+5z_6+2z_{12}+16z_{13}+4z_{16}+4z_{17}\\
&\qquad 4z_{24}+4z_{25}+2z_{27}+2z_{34}+8z_{36}+8z_{37}+8z_{45}+4z_{47}+4z_{57}\\
&\qquad 2z_{56}+2z_{67}\\
&\text{st. } z_{12}+z_1+z_2\geq 1\\
&\qquad (...)\\
&\qquad z_{67}+z_6+z_7\geq 1\\
&\qquad z_{12},...,z_{67}=0 \text{ or } 1\\
&\qquad z_1,...,z_7=0 \text{ or } 1
\end{aligned}$$

The optimal solution for the SCP is $z_2=z_3=z_5=z_{16},z_{17}=z_{47}=z_{67}=1$ which corresponds to the solution $y_1^1=x_1^2=x_2=1$, that is $x_1=2$ and $x_2=1$ in the original problem.

1.3.3 The SCP and the Graph Covering Problem

The graph covering problem, GCP, is a special case of SCP in which each column has at most two non-zero entries (*Christofides [47]*). A graph G is a collection of vertices $v_1,v_2,...,v_n$ (denoted by the set V) and a collection of edges $e_1,e_2,...,e_m$ (denoted by the set E), each edge joining two vertices. A vertex-edge incidence matrix $A=[a_{ij}]$ $(i=1,2,...,n;j=1,2,...,m)$, associated with the graph G is defined by :

(1.11) if the edge e_j connects the vertices v_i and v_k then

$a_{ij}=a_{kj}=1$ and $a_{lj}=0$ for $l\neq i,k$

A cost $c_j>0$ is associated with each edge e_j $(j\in E)$. The GCP consists of finding the least cost set of edges such that each vertex is incident with at least one edge in the set. This is a particular case of SCP with at most two 1's per column.

The GCP is closely related to another graph theoretical problem, the maximum matching problem (*Balinski [18], Edmonds [62]*). This problem consists of finding the maximum cost subset $M\subseteq E$ such that no two edges of M meet at a commmon vertex. The maximum matching problem is a well solvable problem (*Edmonds [63],[64]*) and matching techniques

are used to solve in polynomial time the GCP .

1.3.4 The SCP and other Graph Theory Problems

A large nunmber of problems in graph theory can be formulated as SCP's although many of these problems could be solved more efficiently by specific graph-theoretic techniques . In this section, we briefly review some of those problems and outline the way they are related to the SCP :

1.3.4.1 The Minimum Dominating Vertex Set

A subset D of the vertex set V is said to be a dominant vertex set (or an externally stable set) if for every vertex v_j not in D there exists an edge from a vertex in D to v_j. The problem of finding the minimum dominating vertex set of a graph is a unicost SCP. The sets N and M of the SCP are identical and the constraint matrix is defined by :

$$(1.12) \qquad a_{ij} = \begin{cases} 1 & \text{if } i=j \text{ or there is an edge linking } v_i \text{ and } v_j \\ 0 & \text{otherwise} \end{cases}$$

1.3.4.2 The Maximum Independent Vertex Set

A subset I of the vertex set V is said to be an independent set (or an internally stable set) if there is no edge in E linking any two vertices of I. The problem of obtaining the maximum independent set of a graph G is closely related to the SCP where the columns represent the vertices of G and the rows stand for the edges. The constraint matrix is the vertex-edge incidence matrix defined by (1.11).

In fact, if we define the variable $x_j=1$ if the vertex v_j is in I, $x_j=0$ otherwise, then the

maximum independent vertex set is given by the optimal solution of the integer program :

$$(1.13) \qquad \begin{aligned} &\max \ \Sigma_{j\epsilon V} x_j \\ &\text{st.} \ \Sigma_{j\epsilon V} a_{ij} x_j \leq 1 \qquad (i\epsilon E) \\ &\quad x_j = 0 \text{ or } 1 \qquad (j\epsilon V) \end{aligned}$$

Considering the variables $y_j = 1 - x_j$ $(j\epsilon V)$ the objective function in (1.13) can be rewritten as :

$$(1.14) \qquad \Sigma_{j\epsilon V} x_j = \Sigma_{j\epsilon V}(1-y_j) = \#V - \Sigma_{j\epsilon V} y_j$$

and the constraints in (1.13) become for each edge e_i :

$$(1.15) \qquad \begin{aligned} x_k + x_l &\leq 1 \qquad && v_k \text{ and } v_l \text{ are the terminal} \\ (1-y_k)+(1-y_l) &\leq 1 \qquad && \text{vertices of } e_i \\ -y_k - y_l &\leq -1 \\ y_k + y_l &\geq 1 \\ y_k, y_l &= 0 \text{ or } 1 \end{aligned}$$

Finally, apart from the constant value #V, (1.13) is equivalent to the SCP :

$$(1.16) \qquad \begin{aligned} &\min \ \Sigma_{j\epsilon V} y_j \\ &\text{st.} \ \Sigma_{j\epsilon V} a_{ij} y_j \geq 1 \qquad (i\epsilon E) \\ &\quad y_j = 0 \text{ or } 1 \qquad (j\epsilon V) \end{aligned}$$

1.3.4.3 Graph Colouring Problem

The graph colouring problem consists of determining the minimum number of colours that can be assigned to each vertex of the graph such that no two adjacent vertices have the same colour. The graph colouring problem can be formulated as a SPP (Korman [123]).

Chapter 1

Suppose all independent sets $I_1,I_2,...,I_t$ of a graph **G** have been enumerated and define the 0-1 integer variables x_j as follows :

$$x_j = \begin{cases} 1 \text{ if the set } I_j \text{ represents all vertices with a certain colour} \\ 0 \text{ otherwise} \end{cases} \tag{1.17}$$

Consider now the vertices of **G** and define :

$$a_{ij} = \begin{cases} 1 \text{ if vertex } v_i \text{ is in } I_j \\ 0 \text{ otherwise} \end{cases} \tag{1.18}$$

Then, the graph colouring problem is equivalent to the following SPP :

$$\begin{aligned} (1.19) \qquad & \min \ \Sigma_{j=1}^{t} x_j \\ & \text{st.} \ \Sigma_{j=1}^{t} a_{ij}x_j = 1 \qquad (i\epsilon V) \\ & \quad x_j=0 \text{ or } 1 \qquad (j=1,2,...,t) \end{aligned}$$

Considering inequalities instead the equalities for the constraints $\Sigma_{j=1}^{t} a_{ij}x_j=1$, does not alter the solution of the graph colouring problem. Hence, the problem can be considered as a SCP where the inequalities correspond to possible over-colourings. That implies alternative solutions to the graph colouring problem.

Naturally, it would be surprising if the best way of solving the graph colouring problem was to convert it into a less structured problem such as the SCP, especially since the intermediate step involved the enumeration of all independent sets - itself a hard problem. This also applies to the other problems mentioned above. However, the fact that so many well known problems can be formulated as a SCP emphasizes the central role of the SCP in combinatorial optimisation. At the same time, the possibility exists that an efficient technique for solving the SCP can be adapted to one or more of these problems.

Chapter 1

1.4 THEORETICAL RESULTS FOR THE SCP

1.4.1 Complexity

Let n be a measure of the size of a problem. If the time required by an algorithm to solve the problem is a polynomial function of n, the algorithm is said to be a polynomial time algorithm. Otherwise, the algorithm requires exponential time, ie the time required by the algorithm to solve the problem can be bounded from below by an exponential function (*Horowitz and Sahni [111], Garey and Johnson [74]*).

A problem for which a polynomial time algorithm exists is called a well solved problem; in contrast, if no known polynomial time algorithm exists, the problem is said to be hard. The reason for this differentiation is that if the algorithm requires exponential time, the increase in the amount of time required is explosive compared to the increase in n. This drastically affects the size of the largest problems that can be solved.

The SCP is well known as an NP-complete problem which means that it belongs to a wide class of problems which have the property that if one problem in this class can be solved in polynomial time then all NP-complete problems can be solved in polynomial time. In view of the amount of work done on some of the problems in this class (such as the travelling salesman problem), it seems unlikely that any polynomial time algorithm for them can be found; at the very least, this will require a major breakthrough.

The above means, in practical terms, that a tree search scheme is required to obtain the optimal solution to the SCP. This increases significantly the importance of the lower bound techniques and the heuristics developed for the SCP.

One of the best known procedures for obtaining a cover that approximates the optimum of a SCP is the greedy heuristic, which consists of a sequence of steps, each of which includes in the cover a variable j which covers the maximum number of additional rows. The worst case performance of this greedy for the unicost SCP was shown by *Johnson [116] and Lovász*

[127] to be limited by the relation :

(1.20) $z_u \leq (\Sigma_{j=1}^{d} 1/j)*v(LP)$

where z_u is the value of the greedy cover, v(LP) is the optimal value of the linear program obtained from the SCP dropping the integrality constraints, and

(1.21) $d = \max_{j \in N} \#M_j$

This result is easily extended to the optimal value of the SCP, v(SCP), since $v(LP) \leq v(SCP)$:

(1.22) $z_u \leq (\Sigma_{j=1}^{d} 1/j)*v(SCP)$

This relation was shown by *Chvâtal [54]* to be valid for the general SCP with arbitrary positive costs when the greedy heuristic assigns a value 1 at step s to a variable x_j which maximizes $k_j(s)/c_j$ ($k_j(s)$ is the number of new rows covered at step s by x_j). Furthermore, *Ho [107]* has shown that the same result (1.20) applies for any greedy heuristic that includes in the cover, at step s, the variable x_j such that maximizes any arbitrary function $f(c_j,k_j(s))$.

A natural conclusion from the exposition above is that a different approach is needed in order to find a heuristic that gives a better worst case perfomance. Neverthèless, greedy heuristics have perfomed better than theoretically expected for many test problems, giving in general an upper bound within 10% of the optimum.

A further improvement has been obtained with another class of heuristics which generate a cover using the information - reduced costs - provided by a dual feasible solution for the linear program obtained from the SCP (*Balas and Ho [11]*). This class of heuristics consists of finding a dual feasible solution first and then starts introducing into the cover variables x_j whose reduced cost is positive. This is done in order of increasing values of a function $f(c_j,k_j(s))$ and then, in order to keep the cover prime, variables are removed by decreasing value of reduced cost. A discussion of bounds for this heuristic is presented in *Hochbaum [109]* but no worst case bound better than (1.20) is obtained.

A variant of this consists of using the reduced costs provided by lagrangean relaxation (*Geoffrion [81]*). Variables x_j are set equal to 1 if the reduced cost s_j is zero; otherwise, $x_j=0$. The resulting set $S=\{x_j \epsilon N : x_j=1\}$ is either a cover, or if a row (say i) is not covered, then $s_j>0$ for all $j \epsilon N_i$. Therefore, the variable x_k such that $s_k=\min_{j \epsilon N} s_j$ is introduced into the cover and the reduced costs are updated. This continues until every row is covered after which variables are removed in order to generate a prime cover. Computational experience reported in *Balas and Ho [11]* shows that this heuristic produces consistently better upper bounds than the those mentioned above. This is confirmed by the computational results presented in this thesis.

1.4.2 Relaxations of the SCP

1.4.2.1 LP Relaxation

From (1.20) and taking into account that z_u is an upper boundon v(SCP), it is straightforward that :

(1.23) $\quad v(SCP) \leq (\Sigma_{j=1}^{d}\ 1/j)*v(LP)$

Hence, the same worst case limit applies to the lower bound approximation provided by the linear programming relaxation of the SCP as to the upper bound approximation provided by the greedy heuristic. For the unicost SCP it was shown by *Ho [108]* that :

(1.24) $\quad v(SCP) \leq (1/n)(n+1/2)^{2}*v(LP)$

where n is the cardinality of N. In *[108]* it is also shown that for every $n \geq 20$ there are problems for which the bound on the ratio v(SCP)/v(LP) is about 2/5 of the bound of the ratio $z_u/v(SCP)$.

Chapter 1

An important question is to know when the solution to the LP relaxation of the SCP has an integer solution. Although much work has been done on this question no necessary and sufficient conditions yet exist for guaranteeing the integrality of the LP solution. For general integer programming problems some conditions have been derived in relation to the structure of the matrix A.

A first matricial characterization for problems such that all extreme points are integer is given by unimodularity. The constraint matrix A is said to be totally unimodular if the determinant of every square submatrix of A has value 0,+1 or -1. Considering a matrix of integers $A=[a_{ij}](i=1,2,...,m;j=1,2,...,n)$ and an m-dimensional integer vector b, we define P(A,b) as the region $P(A,b)=\{x : Ax\geq b, x\geq 0\}$. In these conditions, it is well known that for each integer b the region P(A,b) has only integer extreme points if the matrix A is totally unimodular. The transportation problem, the maximum flow problem and the shortest path problem are examples of problems with totally unimodular constraint matrices (*Garfinkel and Nemhauser [78]*).

In contrast to the examples mentioned above the SCP polyhedron may have non-integer extreme points as will be seen later in this Chapter. Therefore, total unimodularity applies only as a sufficient condition for the LP relaxation to have an integer solution. A necessary condition for A to be totally unimodular is that $a_{ij}=0,+1$ or -1 for all i and j. However, in general, is not easy to tell whether a matrix consisting only of that type of coefficient is totally unimodular (*Werra [177]*). A useful sufficient condition for assuring the total unimodularity of a matrix A is :

(i) no more than two nonzero elements appear in each column
and
(ii) the rows can be partitioned into two subsets Q_1 and Q_2 such that
  (a)if a column contains two nonzero elements with the same sign
    one element is in each of the subsets
  (b)if a column contains two nonzero elements of opposite sign,
    both elements are in the same subset

If the SCP constraint matrix satisfies the condition above then the SCP is a GCP in a bipartite graph, which is a well solved problem.

The LP solution of a SCP is also integer if the constraint matrix is balanced. A matrix of zero and ones is said to be balanced if it does not contain any odd submatrix with row sums and column sums equal to 2 (*Berge [35]*). A connected graph with a balanced vertex-edge incidence matrix is a tree and, thus, the SCP with a balanced constraint matrix is equivalent to a GCP in a tree. This is a trivial problem with little practical interest.

1.4.2.2 Lagrangean Relaxation

Other relaxations of a SCP, i.e.easier problems whose feasible region contains the region of the feasible solutions for the SCP, have been tried in order to obtain a lower bound to the optimal value v(SCP). Those relaxations include either well solvable problems such as the graph covering problem and the minimum cost network problem, or problems for which, although hard problems, there exist very efficient algorithms. One such case is the simplex method for linear programming, which has been shown to have exponential time complexity (*Klee and Minty [122], Zadeh [184]*), which has an impressive record of running quickly in practice. Likewise, some algorithms for the knapsack problem (*Balas and Zemel [17]*) have been so successful that many people consider it a "well solved" problem, even though these algorithms also have exponential time complexity (*Garey and Johnson [74]*).

Now, let us consider a general integer linear problem of the form :

$$
\begin{aligned}
(IP) \quad & \min \ \Sigma_{j\epsilon N} c_j x_j \\
& \text{st.} \ \Sigma_{j\epsilon N} a_{ij} x_j \geq b_i \qquad (i\epsilon M_1) \\
& \quad x=(x_j)\epsilon S
\end{aligned}
$$

where $S=\{ x : \Sigma_{j\epsilon N} g_{ij} x_j \geq h_i \ (i\epsilon M_2),\ x_j \geq 0$ and integer$\}$.

A lagrangean relaxation of IP relative to a set of multipliers $\lambda \geq 0$ (*Geoffrion [81]*) is the problem :

Chapter 1

$$(LIP_\lambda) \qquad \min \ \Sigma_{j\epsilon N} c_j x_j + \Sigma_{i\epsilon M_1} \lambda_i (b_i - \Sigma_{j\epsilon N} a_{ij} x_j)$$
$$\text{st.} \quad x \epsilon S$$

The objective of this relaxation is that LIP_λ be an easier problem than the IP to solve with the optimal value to LIP_λ giving a good lower bound on v(IP). If a solution $x(\lambda)$ to the relaxation LIP_λ satisfies $\Sigma_{i\epsilon M_1} \lambda_i (b_i - \Sigma_{j\epsilon N} a_{ij} x_j(\lambda)) = 0$ and is a feasible solution for IP, then $x(\lambda)$ is optimal to IP. The best lower bound obtainable from such a lagrangean relaxation is given by the optimal solution of the lagrangean dual problem :

$$(DLIP) \qquad \max_{\lambda \geq 0} \ v(LIP_\lambda)$$

One method of solving the lagrangean dual is to use subgradient optimization (*Held and Karp [102], Held et al. [103]*) which consists of an iterative method to update the lagrangean multipliers, λ_i ($i\epsilon M_1$), based on subgradient properties for the objective function of DLIP (*Geoffrion [81], Bazaraa and Shetty [23]*).

A simple lagrangean relaxation of the SCP is obtained by relaxing all the constraints $\Sigma_{j\epsilon N} a_{ij} x_j \geq 1$ ($i\epsilon M$), in order to get the following integer program :

$$(1.25) \qquad \min \ \Sigma_{j\epsilon N} (c_j - \Sigma_{i\epsilon M} a_{ij} \lambda_i) \ x_j + \Sigma_{i\epsilon M} \lambda_i$$
$$\text{st.} \quad x_j = 0 \text{ or } 1 \qquad (j\epsilon N)$$

The optimal solution to (1.25) is easily obtained by :

$$(1.26) \qquad x_j = \begin{cases} 1 & \text{if } c_j - \Sigma_{i\epsilon M} a_{ij} \lambda_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

We use the above lagrangean relaxation embedded in a 'preliminary' procedure to compute bounds on v(SCP) and at the same time performing reductions in the dimensions of the problem. This will be described in Chapter 2 with more detail.

Chapter 1

Lagrangean relaxation has been used for the SCP by *Etchberry [66], Balas and Ho [11], Hey [105] and Beasley [25]*. The first author used a tighter variation of (1.25) by finding a maximal subset $M_2 \subseteq M$ such that :

(1.27) $\quad N_i \cap N_k = \Phi$ for $i,k \in M_2$

and then relaxing all the constraints corresponding to $i \in M_1 = M\text{-}M_2$. The resulting problem is :

$$\begin{aligned} (1.28) \qquad & \min \ \Sigma_{j\in N}(c_j - \Sigma_{i\in M_1} a_{ij}\lambda_i) + \Sigma_{i\in M_1}\lambda_i \\ & \text{st.} \ \Sigma_{j\in N} a_{ij}x_j \geq 1 \qquad (i \in M_2) \\ & \quad x_j = 0 \text{ or } 1 \qquad (j \in N) \end{aligned}$$

Denoting by s_j the reduced cost $c_j - \Sigma_{i\in M} a_{ij}\lambda_i$, the optimal solution of (1.28) is given by :

$$(1.29) \qquad x_j = \begin{cases} 1 & \text{if } s_j \leq 0 \\ 1 & \text{if } s_j = \min_{k\in N_i} s_k, \ i \in M_2 \\ 0 & \text{otherwise} \end{cases}$$

Balas and Ho [11] tightened this last relaxation by adding an extra inequality which is the sum of the relaxed constraints :

$$\begin{aligned} (1.30) \qquad & \min \ \Sigma_{j\in N}(c_j - \Sigma_{i\in M} a_{ij}\lambda_i)x_j + \Sigma_{i\in M_1}\lambda_i \\ & \text{st.} \ \Sigma_{j\in N} a_{ij}x_j \geq 1 \qquad (i \in M_2) \\ & \quad \Sigma_{i\in M_1}\Sigma_{j\in N} a_{ij}x_j \geq \#M_1 \\ & \quad x_j = 0 \text{ or } 1 \qquad (j \in N) \end{aligned}$$

Solving (1.30) is not as easy as it is for (1.28) or (1.25), and an heuristic procedure was used in *[11]* to approximate the optimum of (1.30). *Beasley [25]* used the lagrangean relaxation (1.25) both as way of obtaining a lower bound to the SCP and in performing test reductions.

Hey [105] used a graph covering relaxation for the SCP. First, the SCP was formulated as a

graph covering problem with additional constraints. Then, relaxing these constraints, the resulting lagrangean relaxation is solved by an exact algorithm. A network flow relaxation produced in a similar way to the above was also used in *[105]*.

1.4.2.3 Surrogate Relaxation

Surrogate relaxation is another general relaxation technique which has been used for obtaining lower bounds to the optimal solution of an integer program (*Glover [88], Greenberg and Pierskala [95]*). The surrogate relaxation of an integer problem IP associated with any $\mu \geq 0$ is :

$$(SIP_\mu) \qquad \min \ \Sigma_{j\epsilon N} c_j x_j$$
$$\text{st.} \ \Sigma_{i\epsilon M} (\Sigma_{j\epsilon N} a_{ij} x_j)\mu_i \geq \Sigma_{i\epsilon M} \mu_i b_i$$
$$x \epsilon S$$

The optimal value $v(SIP_\mu)$ gives a lower bound on the optimal value to IP and the best value of such a bound is achieved by the surrogate dual :

$$(DSIP) \qquad \max_{\mu \geq 0} v(SIP_\mu)$$

Recent research has produced a number of procedures to solve DSIP (*Dyer [61], Karwan and Rardin [119]*). Also, some empirical results have suggested that surrogate duals may close a significant fraction of the gap between the optimum of a lagrangean dual and the optimal value to IP.

A simple surrogate constraint relaxation for the SCP gives a knapsack type problem :

$$(1.31) \qquad \min \ \Sigma_{j\epsilon N} c_j x_j$$
$$\text{st.} \ \Sigma_{i\epsilon M}(\Sigma_{j\epsilon N} a_{ij} x_j)\mu_i \geq \Sigma_{i\epsilon M} \mu_i$$
$$x_j = 0 \text{ or } 1 \qquad (j \epsilon N)$$

The extra inequality added to (1.28) in order to produce the relaxed problem (1.30) is, in fact, a surrogate type constraint. A disadvantage of this inclusion is, as is mentioned by the authors in *[11]*, that it made more difficult and therefore computationally more expensive, the calculation of reduced costs. In this thesis we present and develop a dynamic programming relaxation for the SCP which is equivalent to the surrogate relaxation and provides reduced costs for the variables in a straightforward way.

1.4.2.4 State Space Relaxation

Many combinatorial optimization problems can be formulated as a dynamic program but only a few of those problems can be solved efficiently by dynamic programming alone, since the dimension of the state space is enormous even for small size problems (*Bellman [27],[28], Dreyfus and Law [60]*). A general relaxation procedure for the state space associated with a given dynamic programming recursion was recently presented by *Christofides et al. [52]* and has been named state space relaxation.

Consider a multistage discrete system where $S=\cup_{i=1}^{m} S_i$ is the state space and let the stage be represented by i. Defining $F_{0,i}(S_0,S)$, for $S\epsilon S_i$, as the least cost of changing the state of the system from S_0 at stage 0 to a state $S\epsilon S_i$ at stage i, the general forward dynamic programming recursion for such system is :

$$(DP) \qquad F_{0,i}(S_0,S) = \min_{S'\epsilon\Delta_i^{-1}(S)} (F_{0,i}(S_0,S') + v_i(S',S))$$

$$\text{for } i=1,2,...,m \text{ and } S\epsilon S_i$$

with,

$$F_{0,0}(S_0,S_0)=0$$

$$v_1(S_0,S)=\infty \text{ if } S\notin\Delta_1(S_0)$$

where $\Delta_i(S')$ is defined as the set of all possible states $S\epsilon S_i$ that can result from $S'\epsilon S_{i-1}$, and $v_i(S',S)$ is the minimum cost of changing the system from state $S'\epsilon S_{i-1}$ to state $S\epsilon S_i$.

Chapter 1

The usual computational problem that one encounters when applying DP to a combinatorial problem is the large dimension of S_i (i=1,2,...,m). This is particularly true for problems such as the travelling salesman problem or vehicle routing problem where S_i involves all subsets of a given set.

A state space relaxation for DP is obtained by defining a mapping function g,from S to a lower dimensional new vector domain Q, and a set function Ω such that the following condition is satisfied :

(1.32) if $S' \epsilon \Delta_i^{-1}(S)$ then $g(S') \epsilon \Omega_i^{-1} g(S)$

The relaxed dynamic programming recursion is written as :

(SSR) $$f_{0,i}(t_0,t) = \min_{t' \epsilon \Omega_i^{-1}(t)} (f_{0,i-1}(t_0,t') + \tilde{v}_i(t',t))$$

for i=1,2,...,m and $t \epsilon g(S_i)$

with,

$t_0 = g(S_0)$

$f_{0,0}(t_0,t_0) = 0$

$\tilde{v}_i(t',t) = \min \{v_i(S',S) : g(S') = t' \text{ and } g(S) = t\}$

As will be shown in Chapter 3 of this thesis, the optimal value to SSR is a lower bound to DP and, more generally, that :

(1.33) $f_{0,i}(t_0,g(S)) \leq F_{0,i}(S_0,S)$ for all i and S

The choice of g plays a major role in state space relaxation since to make the relaxation useful g must be such that (*Christofides et al. [53]*) :

(i) Ω_i^{-1} can be easily computed

and

(ii) the optimization of SSR is over a small domain

or a good lower bound $\tilde{v}_i(t,g(S))$ can be obtained

These aspects will be seen in more detail in Chapter 3 where we apply SSR to a dynamic programming formulation to the SCP. Different expressions for g are presented, and subgradient optimization and state space ascent are used as methods for maximizing the bounds given by SSR.

As will be illustrated later, state space relaxation in dynamic programming is analogous to lagrangean relaxation in integer programming and the effectivness, or otherwise, of a SSR in producing bounds is relative to the specific dynamic program adopted for the problem. We also show that surrogate relaxation in integer programming is equivalent to a particular case of SSR.

1.4.3 The Set Covering Polyhedron

The characteristics of the feasible region of a SCP have been extensively studied mainly with a view to finding good rules for generating cutting planes (*Balas and Padberg [12],[13]*). However, this approach has not been very successful in practice and it was only recently that *Balas and Ho [11]* presented an effective algorithm for large SCP's based on a special type of cutting planes.

The convex hull of all feasible solutions to the SCP defines a polyhedron P. Once characterized, the facets of P can be used to generate cuts and thus improve the lower bound available so far. Let us consider an example.

Example 1.1 :

$$\begin{aligned} \min \quad & x_1+x_2+x_3 \\ \text{st.} \quad & x_1+x_2 \geq 1 \\ & x_2+x_3 \geq 1 \\ & x_1+x_3 \geq 1 \\ & x_1,x_2,x_3 = 0 \text{ or } 1 \end{aligned}$$

The feasible region for this problem is $R=\{(1,1,0),(1,0,1),(0,1,1),(1,1,1)\}$ which is graphically represented in Figure 1.1 by a small circle in each feasible point.

The convex hull of the feasible region R is composed of all the points on the faces or in the interior of the triangular pyramid shown in Fig. 1.1. The optimal solution of the LP relaxation is $x_1=x_2=x_3=1/2$ giving a lower bound $z_l=3/2$. Now, cutting the feasible solution of the LP relaxation (represented by dotted lines in Fig. 1.1) with the facet defined by points (1,1,0), (1,0,1) and (0,1,1) we would get a new value for the lower bound. In fact, this facet is analytically defined by $x_1+x_2+x_3=2$, so the cut $x_1+x_2+x_3\geq 2$ can be added to the original constraints. Solving the LP again z_l becomes equal to 2, the optimal value for the SCP of example 1.1.

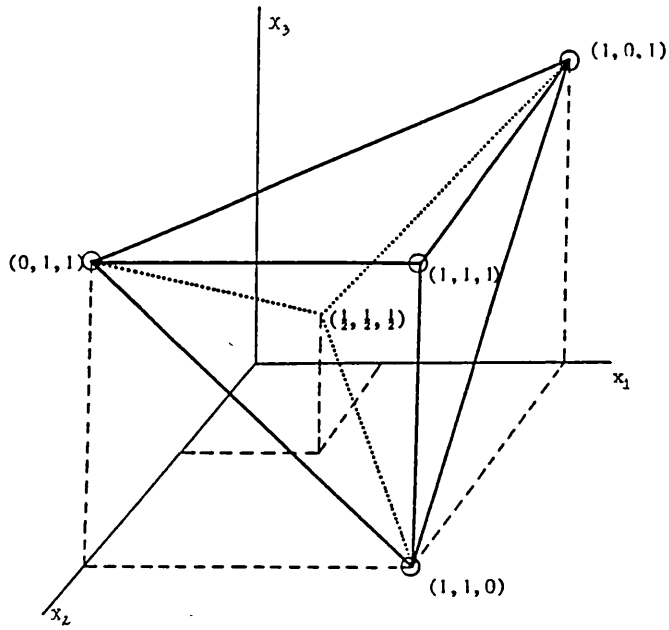


Figure 1.1

Convex hull of the feasible region for the example 1.1

Unfortunately this way of generating cuts is not a practical approach. Neither are the approaches based on adjacency in polyhedra (*Hausman and Korte [101]*), which, for the SCP, consist of considering a feasible vertex and then finding all adjacent vertices; if none of the latter vertices have a better value than the given vertex this is the optimum.

A complete characterization of all facets of an integer program is only known for a few special cases. For the SCP, *Fulkerson [71]* has shown that if the constraint matrix has no rows i and k such that $N_i\subseteq N_k$, then the inequalities $\Sigma_{j\in N}a_{ij}x_j\geq 1$ ($i\in M$) define all facets of

the form $\pi x \geq 1$ where π is a 0-1 vector. However, as is illustrated by example 1.1, not all facets of a SCP polyhedron are of this form.

1.4.4 Integer Duality

Recently a duality theory for integer programming has been developed in a way analogous to linear programming (*Burdet and Johnson [44], Jeroslow [115], Wolsey [183]*). Let us consider the integer linear program :

$$\begin{aligned} \text{(LIP)} \quad & \min \ \Sigma_{j\epsilon N} c_j x_j \\ & \text{st.} \ \Sigma_{j\epsilon N} a_{ij} x_j \geq b_i \qquad (i\epsilon M) \\ & \quad x_j = 0 \text{ or } 1 \qquad (j\epsilon N) \end{aligned}$$

where a_{ij} $(i\epsilon M; j\epsilon N)$ and b_i $(i\epsilon M)$ are integers.

The dual of the problem LIP can be defined as follows (*Jeroslow [115]*) :

$$\begin{aligned} \text{(DIP)} \quad & \max \ F(\mathbf{b}) \\ & \text{st.} \ F(a^j) \leq c_j \qquad (j\epsilon N) \\ & \quad F \text{ is a subadditive and nondecreasing function} \end{aligned}$$

where $\mathbf{b}$ is the vector $(b_i)_{i\epsilon M}$ and a^j represents the jth column of the constraint matrix $A=[a_{ij}]$, $(i\epsilon M; j\epsilon N)$. A function F is said to be subadditive if $F(a+b) \leq F(a)+F(b)$.

A review of the results relating the LIP and the DIP is presented in *Wolsey [183]* and here we only outline the two main general duality properties. The weak duality relation is stated for any primal feasible solution x and any dual feasible function F such that $F(0)=0$, as follows :

$$\Sigma_{j\epsilon N} c_j x_j \geq F(\mathbf{b}) \qquad (1.34)$$

The above is an immediate consequence of the dual feasibility of F :

Chapter 1

$$(1.35)\qquad \Sigma_{j\epsilon N}c_jx_j \geq \Sigma_{j\epsilon N}F(a^j)x_j \qquad (x_j\geq 0 \text{ and } c_j\geq F(a^j)$$
$$\geq \Sigma_{j\epsilon N}F(a^jx_j) \qquad (x_j=0 \text{ or } 1 \text{ and } F(0)=0)$$
$$\geq F(\Sigma_{j\epsilon N}a^jx_j) \qquad (F \text{ is subadditive})$$
$$\geq F(\mathbf{b}) \qquad (F \text{ is nondecreasing})$$

It is obvious that :

$$(1.36)\qquad \min_x \Sigma_{j\epsilon N}c_jx_j \geq \max_F F(\mathbf{b})$$

and the strong duality property establishes that if either LIP or DIP has a finite optimal value then the equality holds in (1.36).

Finding dual feasible functions is not an easy task without using branch-and-bound, cutting planes, dynamic programming or any other technique for integer programming. However, instead of calculating the exact function F another function f, in general not subadditive,can be used such that $f(a^j)\leq F(a^j)$ provides a lower bound to the value of DIP and, therefore, a lower bound on the optimal value for LIP. In this thesis we show how state space relaxation can be used in getting those lower bound solutions.

Finally, let us remark that the value $s_j=c_j-F(a^j)$ is a reduced cost for the variable x_j. If $F(\mathbf{b})+s_j\geq z_u$ (z_u an upper bound on the optimal value v(LIP)), then x_j is equal to 0 for any solution with a better value than z_u. In fact, fixing $x_j=1$ the remaining integer program is :

$$(LIP_j)\qquad \min \ \Sigma_{k\neq j}c_kx_k$$
$$\text{st.} \ \Sigma_{k\neq j}a_{ik}x_k\geq b_i-a_{ij} \qquad (i\epsilon M)$$
$$x_k=0 \text{ or } 1 \qquad (k\epsilon N \text{ and } k\neq j)$$

If F is dual feasible for the LIP then it is dual feasible for LIP_j and $F(\mathbf{b}-a^j)$ is a lower bound on the optimal value of LIP_j, $v(LIP_j)$. Hence,

$$(1.37)\qquad c_j+v(LIP_j) \geq c_j+F(b-a^j)$$
$$\geq c_j+F(\mathbf{b})-F(a^j) \qquad (F \text{ is subadditive})$$
$$\geq z_u$$

Chapter 1

1.5 ALGORITHMS FOR THE SCP

For the last 10-15 years several algorithms have been proposed for solving the SCP but only recently could large size problems be solved consistently. The most successful algorithms combine a tree-search procedure with one or more techniques for obtaining bounds to the problem - heuristics, linear programming, lagrangean relaxation, cutting planes, etc.

A computational survey on different methods for solving the SCP is given in *Christofides and Korman [51]* and more recent algorithms have been presented by *Etchberry [66], Balas and Ho [11] and Beasley [25]*. A tree search procedure was first presented for the SPP (*Pierce [141], Garfinkel and Nemhauser [76]*) and later adapted and improved for the SCP (*Pierce and Lasky [143], Christofides and Korman [51]*). The process was based on the idea of partitioning the constraint matrix into blocks A_j such that any column in A_j has no entry for a row $k<j$. During the course of the algorithm, blocks are searched sequentially, with blocks A_j not being searched unless all rows i such that $1 \leq i \leq j-1$ have already been covered. The tree search procedure corresponds to the sequential covering of blocks and a variety of tests were incorporated by *Christofides and Korman [51]* who report the solution of unicost problems involving up to 35 rows, 1075 columns and 20% density.

An LP-directed tree search for the SCP was presented in *Lemke et al. [126]*. A linear program is solved at any node of the search tree giving a lower bound for the optimal completion of the node. At same time, the LP solution is used in determining the next forward tree branching from the current node. For such a method a high processing time per node searched must be expected, and then the process is unlikely to be computationally efficient if the LP lower bound is not very close to the optimal solution. In this case it seems to be worthwhile to try any cutting plane technique before starting the branching.

A cutting plane approach seems to have been first used for the SCP by *House et al. [112]*. The method presented by them involves solving a sequence of problems, each one of them being obtained from the previous one by considering an additional constraint or cut, which is still a SCP type constraint. *Bellmore and Ratliff [34]* describe one method for generating cutting constraints similar to the one mentioned above but making use of the optimal solution

for the LP relaxation of the SCP. Traditional cutting plane strategies for integer programs have been used for the SCP by *Salkin and Koncal [156]* who applied *Gomory ([91])* cuts. *Martin [132]* also used a cutting plane algorithm based on Gomory cuts but with additional effort put in to enforce integrality of the simplex tableau.

Another cutting plane approach based on disjunctive cuts was proposed by *Balas [10]* and later led to an algorithm capable of solving some large size problems (*Balas and Ho [11]*). The central idea of that approach is to derive valid inequalities for the SCP from conditional lower bounds.

A conditional lower bound of an integer program is a number which would be a valid lower bound if the constraint set were amended by certain inequalities which are also called conditional. If such a conditional lower bound exceeds some known upper bound, then every solution better than the one corresponding to the upper bound violates at least one of the conditional inequalities. This yields a valid disjunction that can be used to derive valid cutting planes which, in the case of the SCP, are themselves inequalities of the set covering type. *Balas [10]* showed that this family of cuts includes the *Bellmore and Ratliff ([34])* inequalities.

Balas and Ho [11] presented an algorithm for solving the SCP based on the family of cutting planes generated from conditional bounds. The algorithm uses a set of heuristics for finding prime covers, another set of heuristics to obtain dual feasible solutions, and subgradient optimization to find lower bounds. Computational results are shown for a set of test problems with up to 200 constraints and 2000 variables. The authors in *[11]* conclude from their computational experience that the algorithm was more reliable and efficient than earlier procedures on large and sparse SCP's. However, they also point out that as problem density increases the strength of the cuts diminishes and so does the efficiency of the algorithm.

In fact, the algorithm failed to solve within 30 minutes on a DEC2050 computer, 3 out of 5 test problems with 200 rows, 1000 columns, 5% density and costs for the variables randomnly generated from the range [1,100]. Better computational results are presented for sparser problems (2% density) with 200 rows, 2000 columns and the same sort of costs. For these

problems the authors conclude that there is a high positive correlation between the number of variables left before one has to branch and the number of the nodes in the search tree. This underlines the need of obtaining reduced costs for the variables of good quality but computationally not expensive. At the same time a tight upper bound is of major importance both for removing variables and fathoming nodes in a branch bound scheme.

Before the algorithm mentioned above was presented, *Etchberry [66]* had reported computational results for an algorithm based on lagrangean relaxation and capable of solving unicost SCP's involving 50 rows, 100 columns and densities from 6% to 21%. In *[66]* a branching strategy is presented for the tree search procedure, which we use in this thesis, consisting of picking two rows i and k and splitting the current problem into two new subproblems as follows :

(i) replacing the constraints $\Sigma_{j\in N_i} x_j \geq 1$ and $\Sigma_{j\in N_k} x_j \geq 1$ by the constraint $\Sigma_{j\in N_i \cap N_k} x_j \geq 1$

or,

(ii) replacing the constraint $\Sigma_{j\in N_i} x_j \geq 1$ by $\Sigma_{j\in N_i^*} x_j \geq 1$ and the constraint $\Sigma_{j\in N_k} x_j \geq 1$ by $\Sigma_{j\in N_k^*} x_j \geq 1$ where $N_i^* = N_i - N_i \cap N_k$ and $N_k^* = N_k - N_i \cap N_k$

In other words, (i) represents the case where i and k have a common column in the optimal solution and (ii) represents the opposite case.

Recently *Beasley [25]* presented an algorithm for the SCP based on a lagrangean relaxation of the covering constraints, and making use of problem reductions tests and linear programming. Computational results are shown for large size test problems involving up to 300 rows, 3000 columns, 5% density and the costs randomnly generated from the range [1,100]. The author in *[25]* used a binary depth-first tree search procedure computing at each tree node a lower bound for the optimal completion of the node using the lagrangean relaxation (1.25). Several reduction tests are done at the initial tree node - column domination, row domination, row redundancy, row splitting, dual ascent penalties - and the LP relaxation is used to compute the optimal multipliers for the lagrangean relaxation. At the other nodes of

the tree the lagrangean lower bound is calculated and reduction tests related to lagrangean penalties and column inclusion are performed.

The great achievement of this algorithm for the test problems reported in *[25]* seems to be the enormous number of variables removed before starting the branching and even before solving the LP relaxation. For test problems with 200 rows, 2000 columns and 5% density the algorithm is reported to have reduced the number of variables to only 74 before the LP being solved. However, it seems natural that this perfomance depends very much on the quality of the upper bound produced by the greedy heuristic and is also greatly dependent on other problem data. In fact, the sort of reduction mentioned above is presented for test problems for which the initial upper bound is equal to the optimal value, while for the worst case shown for the same type of test problems (the upper bound differs to the optimum in 2.4%) the number of effective variables is 181. The LP is thus computationaly more expensive and the tree needs a much larger number of nodes to be completed.

1.6 CONCLUSIONS

In this Chapter we outlined the main aspects of the SCP and from the exposition the following conclusions can be drawn :

(i) The SCP is a combinatorial optimization problem with a large variety of practical applications. A common feature of these applications is that real life situations lead to large size and sparse SCP's with hundreds of rows and thousands of columns. In some cases, although these figures are not achieved, the structure of the SCP makes it difficult to solve (eg. unicost SCP's).

(ii) Any integer linear program can be converted to a set covering problem. The SCP has particularly strong links with other combinatorial problems, especially with several graph theoretic problems. This emphasizes the central role of the SCP in combinatorial optimization and maintains the possibility that an efficient technique for solving the SCP can be adapted to one or more of those problems.

(iii) The SCP is a NP-complete problem, that is a tree search scheme is theoretically required to obtain the optimal solution. This increases significantly the importance of the lower bound techniques and the heuristics developed for the SCP.

(iv) Greedy heuristics have been established as a way of generating feasible solutions to the SCP. Although having a poor worst case perfomance these heuristics have produced reasonable results for many test problems and it is worthwhile to use them as a first attempt to approximate the optimum.

(v) When the greedy upper bound is not very close to the optimum, further improvements have been possible using another class of heuristics which generate a cover to the SCP based on reduced costs for the variables. These reduced costs may be provided either by a dual feasible solution for the LP relaxation of the SCP, by lagrangean relaxation or, as will be seen in this thesis, by state space relaxation. However, no better worst case perfomance has been proved for this last class of heuristics either. Hence, a different approach is still needed in order to find a heuristic that gives a better worst case perfomance. In this thesis we present a different way of generating covers for the SCP based on a decomposition technique which produced in some cases an improvement on the

upper bound value obtained by the two classes of heuristics mentioned above.

(vi) Relaxation techniques have been widely used to obtain lower bounds for the SCP. In particular, LP and lagrangean relaxations have proved to be very helpful in providing good approximations from below to the optimal value.

(vii) Some examples exist where the LP produces the optimum for the SCP but this can not be predicted except in very special cases. Not only to deal with this situation but also because the optimal dual variables provide good initial information for other relaxations, it seems advisable to make use of linear programming together with any other techniques. Since large and sparse linear programs can take too long to be solved it is useful to try reductions on the size of the problems involved before utilizing the LP.

(viii) Lagrangean relaxation has proved extremely useful for many combinatorial problems. In particular, for the SCP it was successfully used for large size SCP's. Besides, lagrangean relaxation can be used to perform test reductions both on the number of the rows and the number of variables of the original problem.

(ix) For general integer programs, some experimental results have suggested that surrogate relaxation can be used to close a significant proportion of the gap between the lagrangean lower bound and the optimal value of the problem. For the SCP this idea was implicitly used yielding an improvement on the value obtained from a lagrangean relaxation of the SCP.

(x) State space relaxation (SSR) is a recently introduced lower bound technique in dynamic programming, and can be applied for many combinatorial problems that can be formulated as dynamic programs. As will be seen later the SCP is in this category and we will apply SSR to the SCP, developing a particular case which is equivalent to surrogate relaxation. One of the advantages of this technique comes from the easy way the reduced costs for the variables can be computed. It suggests also a different approach for solving the surrogate dual.

(xi) Cutting plane strategies have also been used for solving the SCP. However, cutting planes based on the LP are unlikely to be computationally efficient if the LP bound is not very close to the optimal solution. Recently, a different class of cutting planes, generated from conditional bounds, have been used in solving large size SCP's. Reduced costs play a major role in obtaining those cuts and this

maintains the possibility of making use of this cutting plane strategy when good quality reduced costs are obtainable.

(xii) Several algorithms have been proposed to solve the SCP and this has corresponded to the aim of solving larger and larger problems related to real life situations. The most successful algorithms so far combine different lower bound techniques, embedding them in a tree search procedure to solve optimally the SCP. Removing the maximum number of variables before starting the branching has proved vital for the efficiency of the algorithms in solving large test problems. Hence, test reductions both in rows and columns appear as a useful first step in any algorithm for large size SCP's. In this thesis we present and apply an algorithm for large size SCP's based on a decomposition technique. Large size SCP's are decomposed into many smaller sub-problems each one of them still being a SCP. The sub-problems are then solved or approximated by a suitable technique whose main characteristics must be speed, accuracy and utility for the general problem. This last requirement means, for instance, that information (eg. reduced costs) for a sub-problem should be used in the general problem. We show that state space has these characteristics and couples well with the decomposition technique.

Before applying the decomposition, we perform reductions in the number of rows and columns of the SCP making use of a procedure which combines preliminary reductions, heuristics, lagrangean relaxation and linear programming. At the same time, bounds, both from below and above, are obtained for the optimal value of the problem. This procedure is described in detail in Chapter 2 where computational experience with different types of SCP , is reported.

In Chapter 3, we develop the application of state space relaxation for the SCP. Some theoretical results are presented and two different relaxations are studied in detail. Computational results are given either for comparing the two relaxations between themselfes and also to compare the SSR bound with the LP relaxation bound.

The decomposition technique is presented in Chapter 4 with special focus on its relation with state space relaxation. Computational results are also shown for large scale problems and for the unicost SCP.

Finally, in Chapter 5, we describe a tree-search scheme making use of the techniques mentioned above. The complete procedure depends on the structure of the problem, with all techniques being used for solving large size SCPs. Full computational results for problems with up to 400 rows and 4000 columns are given in this Chapter.

Chapter 2

REDUCTIONS FOR LARGE SIZE SET COVERING PROBLEMS

2.1 INTRODUCTION

In this Chapter, a procedure to perform reductions on the number of columns and rows of large size SCPs is presented. The method consists of a combined utilisation of preliminary reductions, heuristics, lagrangean relaxation and linear programming. Each one of these techniques is discussed for the particular case of the SCP and reduction tests are derived. Then, the general procedure is described and an example taken from the literature is worked out to illustrate the method. Computational results are shown for three different classes of test problems and some conclusions are given at the end of the Chapter.

2.2 PRELIMINARY REDUCTIONS

Before using an algorithm for solving the SCP, a number of preliminary reductions can be made both on the columns and the rows of the problem. They can be listed (as in *Garfinkel and Nemhauser [78]*) as follows :

Reduction 2.1 : *nonpositive costs*

If a cost c_j is negative then x_j is, necessarily, equal to 1 in any optimal solution and all rows $i \in M_j$ can be removed. Clearly, if $c_j=0$ then x_j can be included in any solution without changing its value and the same reduction applies.

Reduction 2.2 : *null row*

If $N_i=\Phi$ for a particular $i\epsilon M$ then there is no feasible solution to the problem.

Reduction 2.3 : *single 1 in a row*

If any row $i\epsilon M$ has just one non-zero element a_{ij}, say $N_i=\{ j \}$, then x_j must be equal to 1 and all rows $k\epsilon M_j$ may be deleted.

Reduction 2.4 : *row dominance*

For some i and l in M if $N_i\subseteq N_l$ then l can be deleted since any cover for i is a cover for l.

Reduction 2.5 : *column dominance*

For a subset S of columns and a single column k if $\Sigma_{j\epsilon S}c_j\leq c_k$ with $c_j>0$ ($j\epsilon S$) and $M_k\subseteq \cup_{j\epsilon S} M_j$ then column x_k can be removed.

For a practical SCP, reductions 2.1 to 2.3 are unlikely to be useful, but if the problem occurs as a sub-problem of some other SCP, namely in a tree-search procedure or using any sort of relaxation, then negative costs for columns and null or single 1 rows may occur. For randomnly generated problems in which the coefficients a_{ij} are independent random variables with a fixed probability that $a_{ij}=1$, reduction 2.4 may be useful when the number of variables is large. Finally, reduction 2.5 although useful for several practical examples implies an expensive computational effort in terms of time . A computationally effective version, not covering all the possibilities, is given in *Beasley [25]* and stated here as :

Reduction 2.5': *minimum cost per row test*

For a single column $k\epsilon N$ such that $\#M_k>1$ if $c_k\geq\Sigma_{i\epsilon M_k}d_i$ with $d_i=\min_{j\epsilon Ni}c_j$ then k can be removed.

If $\#M_k=1$, say $M_k=\{ i \}$, and there exists another column $l\epsilon N_i$ and such that $c_l\leq c_k$, then k can be deleted.

This reduction is very efficient for large size problems with random coefficients c_j ranging between 1 and 99 but it is not useful when $c_j=1$ for all $j\epsilon N$ or c_j is proportional to the number of 1's in column j. As it will be seen in the following sections, reduced costs can also be used instead of real costs in reduction 2.5.

2.3 LP RELAXATION AND DUAL FEASIBLE SOLUTIONS

The linear programming relaxation of the SCP is the linear program obtained by dropping the integrality constraint $x_j=0$ or 1 for $j\epsilon N$:

$$
\begin{aligned}
(LP) \quad & \min \ \Sigma_{j\epsilon N}\, c_j x_j \\
& \text{st.} \ \Sigma_{j\epsilon N_i} x_j \geq 1 \quad (i\epsilon M) \\
& \quad 0 \leq x_j \leq 1 \qquad (j\epsilon N)
\end{aligned}
$$

Obviously, $v(LP)\leq v(SCP)$ and if the LP solution is integer then it is the optimal solution to the SCP. An algorithm for solving the SCP based on the LP relaxation is presented in Lemke et al. [126].

The dual linear problem, DLP, associated with the SCP is then :

$$
\begin{aligned}
(DLP) \quad & \max \quad \Sigma_{i\epsilon M}\, u_i \\
& \text{st.} \quad \Sigma_{i\epsilon M_j} u_i \leq c_j \ (j\epsilon N) \\
& \quad u_i \geq 0 \qquad (i\epsilon M)
\end{aligned}
$$

Two well-known results for general linear programming duality can be summarised for the particular case of the set covering problem as follows :

Chapter 2

Proposition 2.6 : Let $u=(u_i)_{i\in M}$ be a dual feasible solution for the SCP, that is, a feasible solution to (DLP).Then :

(a) $z_l(u)=\Sigma_{i\in M}u_i$ is a lower bound on v(SCP)

(b) for some column k and a known upper bound ,z_u , to the SCP if

$$z_l(u)+(c_k-\Sigma_{i\in Mk}u_i)\geq z_u$$

then x_k can be deleted.

Corollary 2.7 : Let $u=(u_i)_{i\in M}$ be a dual feasible solution to the SCP. If there exists a row, say $l\in M$, such that :

$$d_l= \min_{j\in N_l} (c_j-\Sigma_{i\in Mj}u_i)>0$$

then the lower bound $z_l(u)$ can be increased to $z_l(u)+d_l$.

Proof : this is an immediate consequence of the previous proposition since $u'=(u_i')_{i\in M}$ defined as follows :

$$u_i'= \begin{cases} u_i & ,i\neq l \\ u_i+d_l\ , & i=l \end{cases}$$

is a dual feasible solution with value $z_l(u)+d_l$.

A further result connecting Reduction 2.5 and the above proposition can be stated as follows :

Proposition 2.8 : For a single column k and a subset $S\subseteq N$ satisfying the conditions of Reduction 2.5 there exists a dual feasible solution u and an upper-bound on v(SCP), z_u, such that

$$c_k+\Sigma_{i\in M-Mk}u_i\geq z_u$$

if $v(SCP_k) - v(DLP_k) \leq c_k - \Sigma_{j\in S} c_j$

where:

$$(SCP_k)\quad \min \Sigma_{j=k} c_j x_j$$
$$\text{st.} \Sigma_{j\in N-\{k\}} x_j \geq 1 \quad (i\in M-M_k)$$
$$x_j = 0 \text{ or } 1 \quad (j\neq k)$$

and the corresponding dual linear problem:

$$(DLP_k)\quad \max \Sigma_{i\in M-M_k} u_i$$
$$\text{st.} \Sigma_{i\in M_j} u_i \leq c_j \quad (j\neq k)$$
$$u_i \geq 0 \quad (i\in M-M_k)$$

Proof : the proof is immediate since if u_i is the optimal solution for DLP_k then :

$$u'_i = \begin{cases} u_i & , i \in M - M_k \\ 0 & , i \in M_k \end{cases}$$

is feasible for the dual linear problem of the initial SCP. Hence, it follows from

$v(SCP_k) - v(DLP_k) \leq c_k - \sum_{j \in S} c_j$

that :

$$c_k + \sum_{i \in M - M_k} u'_i = c_k + v(DLP_k) \geq \sum_{j \in S} c_j + v(SCP_k) \geq v(SCP)$$

The practical importance of Proposition 2.8 is that given any dual feasible solution to the SCP it can be used to scan variables for which Reduction 2.5 applies. The procedure is computationally cheaper than reduction 2.5 and can be repeated for several different dual feasible solutions.

Chapter 2

2.4 HEURISTICS FOR OBTAINING UPPER AND LOWER BOUNDS TO THE SCP

Heuristic algorithms can be used to obtain both upper and lower bounds on the optimal value, v(SCP), for a SCP. Tight upper bounds are important in removing variables from reduced costs analysis, and also in fathoming tests when a tree-search procedure is being used to solve the problem.

A particular type of heuristics to obtain upper bounds to the SCP have been studied in detail by *Balas and Ho [11]*. They are of the 'greedy' type in the sense that a cover to the problem is generated sequentially by selecting, at each step, a variable x_j (to enter the cover) that minimizes a certain function of the coefficients of x_j. The general form of this function is $f(c_j,m_j)$ where c_j is the cost of variable x_j and m_j denotes the number of positive coefficients of x_j in those rows not yet covered.

The heuristic procedures presented in *[11]* differ in the particular form of function f, and five expressions were considered :

(2.1) (i) $f(c_j,m_j)=c_j$

(ii) $f(c_j,m_j)=c_j/m_j$

(iii) $f(c_j,m_j)=c_j/\log_2 m_j$

(iv) $f(c_j,m_j)=c_j/m_j\log_2 m_j$

(v) $f(c_j,m_j)=c_j/m_j \ln m_j$

Case (ii) is the greedy heuristic shown by *Chvâtal [54]* to have the worst case bound given by :

(2.2) $z_{heur} \leq (\Sigma_{j=1}^{d} 1/j)*v(SCP)$

where z_{heur} is the value of a solution found using the heuristic and $d=\max_{j\in N}\#M_j$. *Chvâtal [54]* proved that this was the best possible bound and *Ho [107]* has shown that there is no better bound with the other different expressions mentioned above for function f.

Although these greedy-type heuristics have a theoretically poor worst case perfomance, they have proved reasonably effective for many test problems. *Balas and Ho [11]* suggest the intermitent use of several functions f rather than a single function since in computational experiments no one function f has been found to uniformly dominate the others.

We used expressions (i) , (iii) and a slightly different version of (ii) to compute upper bounds to the SCP. The evaluation of function $f(c_j,m_j)$ is restricted to the set $N^*=\cup_{i\epsilon M^*} N_i$ where $M^*=\{i\epsilon M : \#N_i=k\}$ with k the minimum cardinality of any row. When using function (iii) we consider $f(c_j,m_j)=c_j$ for $m_j=1$. Ties in evaluating f are broken by the value of m_j (the maximum deciding which variable is chosen) or, if there still is a tie , by $\#M_j$ (the maximum cardinality variable is chosen).

The modified version for (ii) we used is

(2.3) $\quad f(c_j,m_j)=s_j/m_j$

where s_j is the reduced cost instead of the initial cost for variable x_j. This enables us to obtain , at the same time, a lower-bound to the SCP corresponding to a dual feasible solution constructed in the procedure. This is done by assigning, at each iteration, the value $f(s_j,m_j)$ to all rows which are not yet covered with an entry for the selected variable.

The general procedure is described next with f_k, $k=1,2,3$, denoting, respectively, functions (iii),(i) and the modified (ii).

Chapter 2

Procedure 2.9 . *Computes upper and lower bounds to the SCP*

Step 0 . initialisation

$k=1$

$z_u=+\infty$

go to 1

Step 1 . selection function f_k

$R=M$

$S=\Phi$

$m_j=\#M_j \quad (j\epsilon N)$

$s_j=c_j \quad (j\epsilon N)$

$u_i=0 \quad (i\epsilon M)$

go to 2

Step 2 . obtaining sets M^* and N^*

$M^*=\Phi$

for all $i\epsilon R$ if $\#N_i=\min_{k\epsilon R} \#N_k$ then $M^*=M^*\cup\{i\}$

$N^*=\cup_{i\epsilon M^*} N_{i^*}$

go to 3

Step 3 . selecting variable j^*

choose j^* such that

$$f^*=f_{j^*}(s_{j^*},m_{j^*})=\min_{j\epsilon N^*} f(s_j,m_j)$$

if $k=3$ go to 4

otherwise go to 5

Step 4 . constructing a dual feasible solution

for all $i\epsilon M^*$ do $u_i=u_i+f^*$

and for all $j\epsilon M_i$

do $s_j=s_j-f^*$

go to 5

Step 5 . updating sets and values

$S=S\cup\{j^*\}$

$R=R - M_{j^*}$

$m_j=m_j-1 \quad (j\epsilon N_i; i\epsilon M_{j^*})$

if $R=\Phi$ go to 2

otherwise go to 6

Step 6 . generating a prime cover

consider the elements $j\epsilon S$ in order and

if $S-\{j\}$ is still a cover then set $S=S-\{j\}$

go to 7

Step 7 . computing z_u and z_l

$z_u=\min(z_u,\Sigma_{j\epsilon S}c_j)$

if $k=3$ then $z_l=\Sigma_{i\epsilon M}u_i$ and stop

otherwise, set $k=k+1$ and go to 1

Chapter 2

After using procedure 2.9 the reduced costs $s_j(j\epsilon N)$ can be used to remove variables by applying the test of Proposition 2.6(b).

2.5 LAGRANGEAN RELAXATION

Lagrangean relaxation has been widely used to analyse discrete optimization problems (*Geoffrion [81],[82], Fisher et al. [68], Shapiro [161]*). The list of applications of lagrangean relaxation has grown over the last decade to include a wide variety of combinatorial problems such as the travelling salesman problem, facility location, scheduling, the generalized assignment problem and the set-covering problem.

Etchberry [66] used this approach for the SCP in which a subset of covering constraints is relaxed and subgradient optimization is used to improve the corresponding lower bound. This relaxation was also used in the cutting-plane algorithm presented by *Balas and Ho [11]*. Network flow and graph covering relaxations for the SCP have been studied by *Hey and Christofides [106]*.

In this thesis we use the simplest version of Etchberry's relaxation for the SCP in order to improve the initial heuristic lower bound obtained by procedure 2.9 and also to perform preliminary reductions on the SCP as described earlier. The method consists of relaxing all the constraints and, for a particular set of multipliers $\lambda=(\lambda_i)_{i\epsilon M} \geq 0$, solve the problem :

$$(LSCP_\lambda) \quad \min \ \Sigma_{j\epsilon N}(c_j-\Sigma_{i\epsilon Mj}\lambda_i)x_j + \Sigma_{i\epsilon M}\lambda_i$$
$$\text{st.} \quad x_j=0 \text{ or } 1 \qquad (j\epsilon N)$$

The optimal solution to $LSCP_\lambda$ is trivial :

$$(2.4) \quad x_j = \begin{cases} 1 & \text{if } c_j-\Sigma_{i\epsilon Mj}\lambda_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

and, $v(LSCP_\lambda)$ is a lower bound to the SCP. Subgradient optimization is then used to modify the multipliers $\lambda_{i\epsilon M}$ ($i\epsilon M$) in order to improve the lower bound. Applying the results of *[81]* to the particular case of $LSCP_\lambda$ we state two propositions :

Proposition 2.10 : If λ_i^* ($i\epsilon M$) is the set of Lagrange multipliers that maximizes $v(LSCP_\lambda)$ then

$$v(LSCP_{\lambda^*})=\max_{\lambda\geq 0} v(LSCP_\lambda)=v(DLP)$$

and $\lambda_i^*=u_i^*$ ($i\epsilon M$) where u_i^* are the optimal dual variables for the LP relaxation of the SCP.

In practice only a fixed number of iterations is performed in order to obtain the best lagrangean multipliers. From the above proposition , this means that the lower bound corresponding to $LSCP_\lambda$ is no better than the LP bound. But, on the other hand, it is much faster for large problems and produces immediately reduced costs to the variables which can then be used to remove variables and also generate a cover to the problem. Moreover, lagrangean relaxation and LP can be combined in order to improve the final lagrangean lower bound without consuming much more time. In the next section we will discus this combined procedure in some detail.

Proposition 2.11 : Let $v(LSCP_\lambda)$ be the optimal value to $(LSCP_\lambda)$ for some $\lambda=(\lambda_i)_{i\epsilon M}$ and z_u be an upper bound to the SCP. The following are valid :

(a) if for some variable x_k the value $s_k=c_k-\Sigma_{i\epsilon M_k}\lambda_i\geq 0$ is such that $s_k+v(LSCP_\lambda)\geq z_u$ then x_k can be removed

(b) if for some row l the value $d_l=\min_{j\epsilon N_l} s_j$ is positive then the lower bound $v(LSCP_\lambda)$ can be increased by d_l and the reduced costs updated to

$$s_j=\begin{cases} s_j-d_l & \text{if } j\epsilon N_l \\ s_j & \text{otherwise} \end{cases}$$

The above proposition gives a way of slightly improving the lagrangean lower bound and at the same time generating a new cover to the problem. In fact, this can be done by procedure 2.9 with k set equal to 3 and the following initial values in step 1 :

$$S=N(\lambda)$$

$$M(\lambda)= \cup_{j\in N(\lambda)} M_j$$

$$R=M\text{-}M(\lambda)$$

$$s_j=\max(0,c_j-\Sigma_{i\in Mj\cap M(\lambda)}\lambda_i\ )$$

$$m_j=1$$

where $N(\lambda)$ is the index set of the optimal variables for $LSCP_\lambda$. Also, in step 4 we only update the dual variable relative to the particular row $i\in M^*$ for which j^* is chosen.

2.6 COMBINING LP AND LAGRANGEAN RELAXATION

Let us suppose now that a subset of columns N_0 is used for the LP relaxation of the SCP instead of the complete set N. Then, the restricted linear program which we denote by LP_0 gives a value $v(LP_0)$ greater than or equal to v(LP) and could be even greater than the optimal solution v(SCP). Neverthless, if $v(LP_0)$ is close to v(LP) then the optimal dual variables to LP_0 can be used as a good approximation of the optimal dual variables to LP. Hence, the lagrangean multipliers in $LSCP_\lambda$ may be initialised to those values and the lagrangean lower bound improved to a value close to v(DLP). The connection between $v(LP_0)$ and v(LP) is stated in the next proposition where DLP_0 designates the dual linear problem associated with LP_0.

Proposition 2.12 : If $u^0=(u_i^0)_{i\in M}$ is an optimal solution to (DLP_0), then :

(2.5) $$v(LP_0)+\Delta_0 \leq v(LP) \leq v(LP_0)$$

where $\Delta_0 = \Sigma_{j\in N\text{-}N0}\min\ (0,c_j-\Sigma_{i\in Mj}\ u_i^0)$

Proof : first,it is obvious that $v(LP)\leq v(LP_0)$. On the other hand if the values u_i^0 are assigned to the respective lagrangean multipliers λ_i in $(LSCP_\lambda)$ it yields :

$$v(LP) \geq v(LSCP_\lambda) = \Delta_0 + \Sigma_{i \in M} u_i^0 = \Delta_0 + v(LP_0)$$

The value Δ_0 is then a good measure of the quality of the approximation $v(LP_0)$ produced by N_0. This subset must be chosen as the smallest cardinality subset of N that gives a value to Δ_0 within a desired limit. From computational experience we take N_0 as composed of :

(i) the variables x_j such that $c_j - \Sigma_{i \in Mj} \lambda_i \leq 0$ for some $\lambda = (\lambda_i)_{i \in M}$ in a fixed number of iterations for $(LSCP_\lambda)$

and

(ii) the variables of the best cover produced by procedure 2.9

2.7 COMBINED PROCEDURE

The general procedure that we used for reducing the number of rows and columns of a SCP is then summarised as follows :

Procedure 2.13 . *Computes lower and upper bounds for the SCP*
and performs reductions on the columns and rows

Step 1 . do preliminary reductions

Step 2 . do procedure 2.9 computing an upper bound z_u and a lower bound z_{heur}; obtain also a cover S and a dual feasible solution u_i, $(i \in M)$.

set : $z_l = z_{heur}$
$\lambda_i = u_i \quad (i \in M)$
$N_0 = S$
$nk = nk_1$
$k = 1$

Step 3 . solve $(LSCP_\lambda)$ and obtain z_{lagr} . Try to improve this value and generate a new cover with value $\bar{z}$
Set $N(\lambda)$ as the index set of the optimal variables to $(LSCP_\lambda)$.
Update : $N_0 = N_0 \cup N(\lambda)$
$z_l = \max(z_l, z_{lagr})$
$z_u = \min(z_u, \bar{z})$
$k = k+1$
If $k \leq nk$ go to 3
if il=2 go to 5
go to 4

Chapter 2

	i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
j	c_j															
1	1	1														
2	1		1													
3	1			1												
4	1				1											
5	1					1										
6	1						1									
7	1							1								
8	1								1	1						
9	2	1														1
10	2			1							1					
11	2					1								1		
12	2		1					1								
13	2									1						
14	2									1		1				
15	2		1									1				
16	3					1									1	
17	3							1	1							
18	3											1	1			1
19	3	1	1	1									1		1	
20	4						1		1					1		
21	4		1		1						1					
22	4	1				1							1			
23	4		1			1			1			1			1	
24	5			1			1		1			1	1			1
25	5	1			1			1			1			1		
26	5			1			1			1			1			1
27	6		1			1			1			1				
28	6	1			1									1		1
29	6		1							1			1			
30	7		1	1			1	1		1		1		1		
31	8		1			1		1		1	1		1		1	1
32	9	1	1	1	1		1		1		1	1		1		1

Tableau II.1

Input data for the SCP of example 2.14 (*Lemke et al. [126]*)

Step 4 . solve (LP_0) and obtain the optimal dual solution u
set : $\lambda_i = u_i$ ($i \epsilon M$)
$k = 1$
$nk = nk2$
$il = 2$
go to 2
Step 5 . do Reduction 2.4
stop

Example 2.14

We illustrate the application of procedure 2.13 for a well known test problem from the literature (*Lemke et al.[126]*), whose input data is shown in Tableau II.1.

STEP 1 . *Preliminary Reductions*
. Reduction 2.1 - there are no negative costs
. Reduction 2.2 - there is no null row
. Reduction 2.3 - there is no row with a single 1
. Reduction 2.4 - there is no row dominating another row
. Reduction 2.5'- for each row, the minimum cost of covering it is given below :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
d_i	1	1	1	1	1	1	1	1	1	2	2	3	2	3	2

the values $p_j = c_j - \Sigma_{i \epsilon M_j} d_i$ are then:

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p_j	0	0	0	0	0	0	0	-1	-1	-1	-1	0	1	-1	-1	-1

j	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
p_j	1	-4	-6	0	0	-1	-4	-5	-2	-3	1	0	1	-2	-6	-5

the variables such that $p_j > 0$ are removed - $x_{13}, x_{17}, x_{27}, x_{29}$ - and so are the ones such that $p_j = 0$ and $\#M_j > 1$ - $x_{12}, x_{20}, x_{21}, x_{28}$.

At the end of this step of procedure 2.13 the number of variables for the problem has been reduced to 24 columns. No rows were removed.

STEP 2 . *Procedure 2.9*
step 0 . $k = 1$
$z_u = \infty$

Chapter 2

step 1 . $S=\Phi$

$R=\{1,2,...,15\}$

$u_i=0$ $(i=1,2,...,15)$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	5	3	5	6	5	5	7	8	10

step 2 . $M^*=\{3\}$

$N^*=\{4,25,32\}$

step 3 . $\min(1, 5/\log_2 5, 9/\log_2 10) = 1$

$j^*=4$

go to 5

step 5 . $S = \{4\}$

$R = R - M_4 = \{1,2,3,5,6,7,8,9,10,11,12,13,14,15\}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	1	1	1	-	1	1	1	2	2	2	2	2	2	2	3	5	3	5	6	4	5	7	8	9

step 2 . $M^* = \{7,8,10,13,14\}$

$N^* = \{7,8,10,11,16,18,23,24,25,30,31,32\}$

step 3 . there is a tie between x_7 and x_8 for the minimum of $f(c_j,m_j)$. The variable x_8 is selected since it covers more rows than x_7.

$j^*=8$

step 5 . $S = \{4,8\}$

$R = R - M_8 = \{1,2,3,5,7,10,11,12,1,3,14,15\}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	1	1	1	-	1	1	1	-	2	2	2	1	2	2	3	5	3	4	5	4	4	6	7	8

step 2 . $M^* = \{7,10,13,14\}$

$N^* = \{7,10,11,16,18,23,25,30,31,32\}$

step 3 . $j^* = 7$

step 5 . $S = \{4,8,7\}$

$R = R - M_7 = \{1,2,3,5,6,10,11,12,13,14,15\}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	1	1	1	-	1	1	-	-	2	2	2	1	2	2	3	5	3	4	5	3	4	5	6	8

Chapter 2

step 2 . $M^* = \{ 10,13,14 \}$
$N^* = \{ 10,11,16,19,23,25,30,31,32 \}$
step 3 . $j^* = 19$
step 5 . $S = \{ 4,8,7,19 \}$
$R = R - M_{19} = \{ 5,6,10,11,13,15 \}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	-	-	-	-	1	1	-	-	1	1	2	1	1	1	2	-	1	2	4	2	2	3	3	4

step 2 . $M^* = \{ 10,13 \}$
$N^* = \{ 10,11,25,30,31,32 \}$
step 3 . $j^* = 11$
step 5 . $S = \{ 4,8,7,19,11 \}$
$R = \{ 6,10,11,15 \}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	-	-	-	-	-	1	-	-	1	1	-	1	1	-	2	-	-	1	3	1	2	2	2	4

step 2 . $M^* = \{ 10 \}$
$N^* = \{ 10,25,31,32 \}$
step 3 . $j^* = 10$
step 5 . $S = \{ 4,8,7,19,11,10 \}$
$R = \{ 6,11,15 \}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	-	-	-	-	-	1	-	-	1	-	-	1	1	-	2	-	-	1	3	-	2	2	1	3

step 2 . $M^* = \{ 6 \}$
$N^* = \{ 6,24,26,30 \}$
step 3 . $j^* = 6$
step 5 . $S = \{ 4,8,7,19,11,10,6 \}$
$R = \{ 11,15 \}$

j	1	2	3	4	5	6	7	8	9	10	11	14	15	16	18	19	22	23	24	25	26	30	31	32
s_j	1	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9
m_j	-	-	-	-	-	-	-	-	1	-	-	1	1	-	2	-	-	1	2	-	1	1	1	2

step 2 . $M^* = \{ 15 \}$
$N^* = \{ 9,18,24,26,31,32 \}$

step 3 . $j^*=9$
step 5 . $S = \{ 4,8,7,19,11,10,6,9 \}$
$R = \{ 11 \}$
step 2 . $M^*=\{ 11 \}$
$N^*=\{ 14,15,18,23,24,30,32 \}$
step 3 . $j^*=15$
step 5 . $S = \{ 4,8,7,19,11,10,6,9,15 \}$
$R = \Phi$
step 6 . S is a prime cover - $z_u=15.0$

For k=2 the procedure generates exactly the same prime cover although some variables enter into S in a different order from before.
For k=3, the procedure produces a different cover - $S=\{5,7,8,19,32\}$ - but still with the same value. The dual feasible solution obtained at the end is :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
u_i	0.9	0.9	0.9	0.9	1.0	0.9	1.0	0.9	0.1	0.9	0.9	0.15	0.9	0.15	0.9

and the corresponding lower bound is $z_l=11.4$.
The corresponding reduced costs for the variables are shown in the following tableau :

j	1	2	3	4	5	6	7	8	9	10	11	14
s_j	0.1	0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.2	0.2	0.1	1.0
j	15	16	18	19	22	23	24	25	26	30	31	32
s_j	0.2	1.85	1.05	0.0	1.95	0.15	0.45	0.4	2.05	1.4	2.9	0.0

Since $z_l+s_{31}=11.4+2.9=14.3>14$, then x_{31} is equal to 0 for any solution better than $z_u=15$ and can be removed from the problem (applying proposition 2.6 and taking into account that all the costs are integer values).
At the end of this step for procedure 2.13 the number of variables has been reduced to 23 and the bounds are $z_l=11.4$ and $z_u=15.0$. Now, setting nk1=3 and $\lambda_i=u_i$ (i=1,...,15), we enter the next step.

STEP 3 . *Lagrangean Relaxation (I)*
. iteration 1
$z_{lagr}=11.4$
$N(\lambda)=\{ 5,7,8,19,32 \}$
$N(\lambda)$ is already a prime cover with value $c[N(\lambda)]=15.0$
the multipliers are then updated by subgradient optimization :
$\lambda_i = \lambda_i + \alpha^*(z_u-z_l)(1-\Sigma_{j\in N}a_{ij}x_j) / (\Sigma_{i\in M}(1-\Sigma_{j\in N}a_{ij}x_j)^2)$
if the resulting value for λ_i is negative then $\lambda_i=0$

Chapter 2

Setting $\alpha=2.0$ the new multipliers are given in the tableau :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ_i	0.0	0.0	0.0	0.9	1.0	0.9	1.0	0.0	0.1	0.9	0.9	0.15	0.9	0.15	0.9

The corresponding reduced costs for the variables are :

j	1	2	3	4	5	6	7	8	9	10	11	14
s_j	1.0	1.0	1.0	0.1	0.0	0.1	0.0	0.9	1.1	1.1	0.1	1.0

j	15	16	18	19	22	23	24	25	26	30	32
s_j	1.1	1.85	1.05	2.7	2.85	1.95	2.15	1.3	2.95	3.2	3.6

. iteration 2

$z_{lagr}=7.8$

$N(\lambda)=\{ 5,7 \}$

Now, a cover is generated from $N(\lambda)$ using the procedure 2.9 with the following initial values :

$S=N(\lambda)$

$M(\lambda) = M_5 \cup M_7 = \{ 5,7 \}$

$R = \{ 1,2,3,4,6,8,9,10,11,12,13,14,15 \}$

s_j given by the tableau above ; $m_j=1,(j=1.....32)$: $u_i=\lambda_i,(i=1,....15)$

call procedure 2.9

step 2 . $M^*=\{ 4 \}$

$N^*=\{ 4,25,32 \}$

step 3 . min (0.1,1.3,3.6) = 0.1 then $j^*=4$

step 4 . $u_4=1.0$ and $s_4=0.0,s_{25}=1.2,s_{32}=3.5$

step 5 . $S = \{ 5,7,4 \}$

$R = \{ 1,2,3,6,8,9,10,11,12,13,14,15 \}$

step 2 . $M^*=\{ 10,14 \}$

$N^*=\{ 10,16,19,23,25,32 \}$

step 3 . min(1.1,1.85,2.7,1.95,1.2,3.5)=1.1 then $j^*=10$

step 4 . $u_{10}=2.0$ and $s_{10}=0.0,s_{25}=0.1,s_{32}=3.4$

step 5 . $S = \{ 5,7,4,10 \}$

$R = \{ 1,2,6,8,9,11,12,13,14,15 \}$

step 2 . $M^*=\{ 14 \}$

$N^*=\{ 16,19,23 \}$

step 3 . min(1.85,2.7,1.95)=1.85 then $j^*=16$

step 4 . $u_{14}=2.0$ and $s_{16}=0.0,s_{19}=0.85,s_{23}=0.1$

step 5 . $S = \{ 5,7,4,10,16 \}$

$R = \{ 1,2,6,8,9,11,12,13,15 \}$

Chapter 2

step 2 . $M^*=\{ 8,9,13 \}$
$N^*=\{ 8,11,14,23,24,25,26,30,32 \}$
step 3 . $\min(0.9,0.1,1.0,0.1,2.15,0.1,2.95,3.2,3.4)=0.1$
$j^*=23$ (tie decided by the number of new rows covered)
step 4 . $u_8=0.1$ and $s_8=0.8, s_{23}=0.0, s_{32}=3.3$
step 5 . $S = \{ 5,7,4,10,16,23 \}$
$R = \{ 6,9,12,3,15 \}$
step 2 . $M^*=\{ 9,13 \}$
$N^*=\{ 8,11,14,25,26,30,32 \}$
step 3 . $\min(0.8,0.1,1.0,0.1,2.95,3.2,3.3)=0.1$ then $j^*=25$
step 4 . $u_{13}=1.0$ and $s_{11}=s_{25}=0.0, s_{30}=3.1, s_{32}=3.2$
step 5 . $S = \{ 5,7,4,10,16,23,25 \}$
$R = \{ 6,9,12,15 \}$
step 2 . $M^*=\{ 9 \}$
$N^*=\{ 8,14,26,30 \}$
step 3 . $\min(0.8,1.0,2.95,3.1)=0.8$ then $j^*=8$
step 4 . $u_9=0.9$ and $s_8=0.0, s_{14}=0.2, s_{26}=2.15, s_{30}=2.3$
step 5 . $S = \{ 5,7,4,10,16,23,25,8 \}$
$R = \{ 6,12,15 \}$
step 2 . $M^*=\{ 6,12,15 \}$
$N^*=\{ 6,9,18,19,22,24,26,30,32 \}$
step 3 . $\min(0.1,1.1,1.05,0.85,2.85,2.15,2.05,2.3,3.2)=0.1$ then $j^*=6$
step 4 . $u_6=1.0$ and $s_6=0.0, s_{24}=2.05, s_{26}=1.95, s_{30}=2.2, s_{32}=3.1$
step 5 . $S = \{ 5,7,4,10,16,23,25,8,6 \}$
$R = \{ 12,15 \}$
step 2 . $M^*=\{ 12,15 \}$
$N^*=\{ 9,18,19,22,24,26,32 \}$
step 3 . $\min(1.1,1.05,0.85,2.85,2.05,1.95,3.1)=0.85$ then $j^*=19$
step 4 . $u_{12}=1.0$ and $s_{18}=0.2, s_{19}=0.0, s_{22}=2.0, s_{24}=1.2$
step 5 . $S = \{ 5,7,4,10,16,23,25,8,6,19 \}$
$R = \{ 15 \}$
step 2 . $M^*=\{ 15 \}$
$N^*=\{ 9,18,24,26,32 \}$
step 3 . $\min(1.1,0.2,1.2,1.95,2.25)=0.2$ then $j^*=18$
step 4 . $u_{15}=1.1$ and $s_9=0.9, s_{18}=0.0, s_{24}=1.0, s_{26}=1.75, s_{32}=2.05$
step 5 . $S = \{ 5,7,4,10,16,23,25,8,6,19,18 \}$
$R = \Phi$
step 6 . a prime cover is obtained from S by removing redundant variables considered in decreasing order of the costs :
$-x_{23}-x_{16}-x_{10}-x_7-x_4-$ and the resulting cover
is $S=\{ 5,6,8,18,19,25 \}$ with cost $c(S)=14.0$
therefore, $z_u=14.0$ and $z_l=\Sigma_{i\in M}u_i=13.0$.

Chapter 2

Variables $x_1,x_2,x_3,x_9,x_{14},x_{15},x_{22},x_{24},x_{26},x_{30}$ and x_{32} are removed by reduced costs analysis.

. iteration 3

$z_{lagr}=7.562$ and $N(\lambda)=\{ 4,5,6,7,8,10,11,18,19,23,25 \}$

since nkl=3 we go to next step of procedure 2.13.

At the end of this step for procedure 2.13 the number of variables has been reduced to n=12 and the bounds are $z_l=13.0$ and $z_u=14.0$

STEP 3 . *Linear Programming*

. $N_0=\{ 4,5,6,7,8,10,11,18,19,23,25 \}$

$N-N_0=\{ 16 \}$

The optimal value for LP_0 is $v(LP_0)=14.0$ and the optimal dual variables are u=(0.0,3.0,0.0,1.0,1.0,1.0,1.0,0.0,1.0,2.0,0.0,0.0,1.0,0.0,3.0)

Since $c_{16}-u_5-u_{14}=2.0$ the value of Δ_0 is 0 and then the lower bound obtained at this step of the procedure is $v(LP_0)+\Delta_0=14.0$ and the optimum for the problem has been found.

Remark 1 . Note that the problem has been reduced and then the value $v(LP_0)+\Delta_0$ may be greater than the linear programming relaxation of the original problem. In fact, the LP lower bound for the example above is equal to 13.4

Remark 2 . In the example only one variable was left out of N_0. It is clear that in such cases, where the number of variables left out is small, would be worthwhile to include them in N_0. However, as will be seen later, this is not the situation for large SCP's where, in general, the cardinality of N_0 is small relative to N

Chapter 2

2.8 COMPUTATIONAL RESULTS

For computational tests of procedure 2.13, we considered three different classes of problems according to the coefficients of the variables in the objective function :

(I) costs c_j randomnly generated from the interval [1,99]

(II) costs $c_j = 1.0$ ($j \epsilon N$)

(III) costs c_j proportional to the cardinality of M_j :

(i) $c_j = \#M_j$

(ii) $c_j = 3.0 + \#M_j$

In class (I), two sets of test problems were considered :

(I.1) number of rows $m = 200$
number of columns $n = 2000$
density $d = 5\%$

(I.2) number of rows $m = 300$
number of columns $n = 3000$
density $d = 2\%$

The same set of test problems was tried out for classes (II) and (III) differing only on the costs :

(II)/(III) number of rows $= 50$
number of columns $= 500$
density $= 20\%$

Table II.2 summarises the computational results for the test problems in class (I). These are identified in column (i) by a general designation **TmXk** where m is the number of rows and k is the number given to the test problem . The initial dimensions of the problem are shown in columns (ii) and (iii) .

Columns (iv) and (v) in Table II.2 give the dimensions of the test problem after doing the

preliminary reductions corresponding to step 1 of procedure 2.13 . As expected for these very large problems with random costs, the reduction on the number of variables is quite significant mainly due to reduction 2.5'. For all test problems in Table II.2 the number of variables was reduced by more than 80% . However, the resulting reduced problem is still a very hard problem with the range of the costs much tighter.

Columns (vi) to (ix) in Table II.2 refer to the execution of step 2 in procedure 2.13 . The first two columns, (vi) and (vii), show respectively the values of the upper and the lower bounds obtained by procedure 2.9 . The other two columns, (viii) and (ix), give the dimensions after using the heuristic dual variables to remove variables by reduced costs ; this did not actually occur for any of the test problems . This is a consequence of the poor quality of the heuristic lower bound which, in fact, could be improved by applying complementary slackness tests (*Balas and Ho [11], Hey [105]*). However, from computational experiments we found that not worthwhile, since the following steps in procedure 2.13 remove most of the variables that would be deleted by the heuristic tests. Besides, the heuristic dual variables proved to be good initial values for the first phase of the lagrangean relaxation in setting the set N_0.

The computational results related to step 3 of procedure 2.13 are shown in columns (x) to (xiii) in Table II.2 . The value in column (x) is the best out of the upper bound obtained by procedure 2.9 (column (vi)) and the values of the covers generated from the solutions of $LSCP_\lambda$. The upper bound was improved for six of the test problems contained in Table II.2 with the most significant decrease being achieved for test problem T300X4 . The lower bound given by the first application of lagrangean relaxation (column (xi)) is naturally much better than the heuristic lower bound . However, that did not imply any significant reduction in the dimensions of the problem as can be seen in columns (xii) and (xiii) .

Columns (xiv) to (xvii) in Table II.2 are related to the restricted linear program LP_0 . The value $v(LP_0)$ is shown in column (xv) while the corresponding lower bound to the SCP, $v(LP_0)+\Delta_0$, is given in column (xiv) . This lower bound is again , for all test problems except T200X5 and T300X3, much better than the previous value (column (xi)). The value in column (xv) is an upper bound on the value of the LP relaxation for the SCP and in all cases but for the mentioned exceptions (T200X5,T300X3) is close to the value in column

TABLE II.2

Bounds and reductions produced by procedure 2.13 for large scale SCPs

PROBLEM	INITIAL DIMENSIONS		PRELIMINARY REDUCTIONS		GREEDY HEURISTICS				LAGRANGEAN RELAXATION (I)				LINEAR PROGRAMMING (RESTRICTED)				LAGRANGEAN RELAXATION (II)			
	m	n	m	n	z_u	z_λ	m	n	z_μ	z_λ	m	n	z_λ	$\bar{z}_P$	m	n	z_μ	z_λ	m	n
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)	(x)	(xi)	(xii)	(xiii)	(xiv)	(xv)	(xvi)	(xvii)	(xviii)	(xix)	(xx)	(xxi)
T200X1	200	2000	200	349	96.0	53.0	200	349	94.0	65.67	200	344	86.84	87.60	200	154	92.0	87.12	200	146
T200X2	200	2000	200	263	77.0	31.0	200	263	77.0	53.07	200	263	64.35	67.47	200	245	74.0	66.35	199	178
T200X3	200	2000	200	381	104.0	62.0	200	381	103.0	78.53	200	380	87.84	91.66	200	354	96.0	90.72	200	176
T200X4	200	2000	200	290	74.0	40.0	200	290	74.0	54.73	200	290	69.19	69.85	200	155	74.0	69.35	199	150
T200X5	200	2000	200	281	60.0	34.0	200	281	60.0	43.19	200	279	44.53	58.25	180	278	60.0	56.16	186	125
T300X1	300	3000	300	325	227.0	141.0	300	325	227.0	188.08	299	325	214.0	215.0	299	111	215.0	215.0	-	-
T300X2	300	3000	300	307	152.0	91.0	300	307	150.0	115.05	300	307	136.79	140.3	300	306	147.0	138.4	300	287
T300X3	300	3000	300	348	228.0	138.0	300	348	223.0	200.34	300	348	184.53	217.0	300	346	223.0	211.8	300	333
T300X4	300	3000	300	482	277.0	169.0	300	482	263.0	209.50	300	482	238.8	245.5	300	477	258.0	243.0	300	444
T30005	300	3000	300	324	203.0	130.0	300	324	199.0	161.61	291	323	184.6	192.0	291	292	192.0	187.7	250	224

(xiv). Further reduction in the dimensions is achieved for most of the problems using the dual feasible solution for reduced cost analysis. As is shown in column (xvi), the number of rows is reduced for test problems T200X5, T300X1 and T300X5 while the number the columns ((xvii) in Table II.2) decreases for all the test problems.

Finally, columns (xviii) to (xxi) show the outcome of the second phase application of lagrangean relaxation. The lower bound (column (xix)) is still better for all problems and an improvement on the upper bound (column (xviii)) is obtained for all except test problems T200X4 and T200X5. As a result of this the dimension of the problems is quite significantly reduced and for test problem T300X1 the optimal value is obtained.

The reduced size test problems still have the same density and , in order to obtain the optimal solution for them , a tree-search procedure must be used as will be seen in Chapter 5. Before that a combination of a decomposition technique and state space relaxation for the SCP will be developed in order to improve the lower bound obtained at the end of procedure 2.13 . The results shown Table II.2 confirm what has been said about the heuristic procedures for the SCP. Table II.3 summarizes the information relative to upper bounds which is contained in Table II.2. From there it is evident that the greedy heuristic managed to produce a bound very close to the optimal value only in two cases (T200X4 and T200X5). The 'greedy' upper bound for test problem T200X5 is equal to the optimum while the bound for test problem T200X4 is only 1.3% above the optimal value. In this case, the heuristic based on the lagrangean reduced costs failed to improve the upper bound obtained by the greedy heuristic. The opposite occurred for all the other test problems with the gap being reduced, for all the problems, to less than 5% of the optimum. Neverthless, as can be seen in Table II.4, the greedy heuristic only takes a small part of the computational time required for performing the procedure for the test problems that we have considered. Hence, it seems acceptable to try even more expressions than the three different ones we used for the greedy selection function.

Table II.4 shows the computational times of each step of procedure 2.13 forthe test problems TmXk (m=200,300;k=1,...,5). For solving the LPwe used the code named XMP (dual simplex algorithm) developed by *Marsten [129]*, which has performed reasonably fast for many

TABLE II.3

Evolution of the upper bound in procedure 2.13

PROBLEM		GREEDY HEURISTIC		LAGRANGEAN (I)		LAGRANGEAN (II)	
(i)	z_{opt} (ii)	z_u (iii)	% (iv)	z_u (v)	% (vi)	z_u (vii)	% (viii)
T200X1	90.	96.	6.6	92.	2.2	92.	2.2
T200X2	71.	77.	8.4	77.	8.4	74.	4.2
T200X3	93.	104.	11.8	103.	10.7	96.	3.2
T200X4	73.	74.	1.3	74.	1.3	74.	1.3
T200X5	60.	60.	-	60.	-	60.	-
T300X1	215.	227.	5.6	227.	5.6	215.	-
T300X2	141.	152.	7.8	150.	6.3	147.	4.2
T300X3	218.	228.	4.5	223.	2.3	223.	2.3
T300X4	247.	277.	12.1	263.	6.4	258.	4.4
T300X5	192.	203.	5.7	199.	3.6	192.	-

TABLE II.4

Computing times for procedure 2.13
(CDC 7600 ; FTN compiler)

PROBLEM (i)	PRELIMINARY REDUCTIONS (ii)	GREEDY HEURISTIC (iii)	LAGRANGEAN RELAXATION (iv)	LINEAR PROGRAMMING (v)	LAGRANGEAN RELAXATION (vi)	TOTAL (vii)
T200X1	.49	.35	.94	4.22	.81	6.81
T200X2	.48	.33	1.04	4.30	.93	7.08
T200X3	.48	.36	1.05	5.40	.99	8.28
T200X4	.48	.33	1.91	3.17	.71	6.60
T200X5	.49	.31	.86	2.87	.75	5.28
T300X1	.45	.59	1.42	7.08	.52	10.06
T300X2	.45	.48	1.34	6.33	1.03	9.63
T300X3	.45	.50	1.52	6.90	1.50	10.87
T300X4	.45	.40	1.39	10.16	1.27	13.67
T300X5	.45	.54	1.29	4.20	1.05	7.53

practical applications. However, it is clear from Table II.4 that a large part of the computational time of procedure 2.13 is spent obtaining the LP solution. This is a natural consequence of the few reductions achieved on the number of the rows, which drastically affects the time taken by the simplex method to solve an LP problem.

This is illustrated by Table II.5 where we present some information relative to problem LP_0 for each of the mentioned test problems. As can be seen from Table II.5, for problems with the same number of rows, the fewer the number of elements in N_0 the less is the computational time required for solving LP_0. But, this can strongly affect the quality of the bound (eg. test problem T300X3 for which Δ_0 is very big). The difference between problems in terms of rows is not significant but, at least the tendency of the LP computational time to increase with m is illustrate for test problems T300X5 ($m=291;n_0=152;t=4.20$), T300X2 ($m=300;n_0=150;t=6.33$) and T300X3 ($m=300;n_0=147;t=6.90$).

Hence, a further effort to reduce the number of rows of the the problem before calling the LP could pay off both in terms of the bound and the computing time. These further reductions might be achieved by using the sort of tests considered by *Beasley [25]*, namely the row redundancy and row splitting tests. Another possiblity consists of considering the 'core' problem LP_0 with a number of rows fewer than the LP and using an heuristic for completing the dual feasible solution needed for the second phase of the lagrangean relaxation. Although, we have no consistent computational experience on this last method it seems fairly reasonale in particular for very large problems for which the time spent solving the LP becomes very critical when applying procedure 2.13.

Refering again to Table II.4, note that the computational times corresponding to the lagrangean relaxation include the calculation of the cover generated from the lagrangean solution. Then, this time depends much on the number of effective variables in the problem and, therefore, the first phase of the lagrangean relaxation is more expensive. On the other hand, the covers produced by this method were, for the larger problems, consistently better than the ones obtained from the greedy heuristic (Table II.3).

The computational results relative to the test problems of classes II and III are presented in

TABLE II.5

Dimension and computing times of the restricted LP relaxation of the SCP

PROBLEM (i)	#M (ii)	#N (iii)	$\#N_0$ (iv)	Time (v)
T200X1	200	344	120	4.22
T200X2	200	263	112	4.30
T200X3	200	380	101	5.40
T200X4	200	290	91	3.17
T200X5	200	279	79	2.87
T300X1	300	325	184	7.08
T300X2	300	307	150	6.33
T300X3	300	348	147	6.90
T300X4	300	482	185	10.11
T300X5	291	323	152	4.20

TABLE II.6

Bounds and reductions produced by procedure 2.13 for test problems of the classes II and III

PROBLEM	INITIAL DIMENSIONS		PRELIMINARY REDUCTIONS		GREEDY HEURISTICS				LAGRANGEAN RELAXATION (I)				LINEAR PROGRAMMING (RESTRICTED)				LAGRANGEAN RELAXATION (II)			
	m	n	m	n	z_μ	z_ℓ	m	n	z_μ	z_ℓ	m	m	z_ℓ	$\bar{z}_P$	m	n	z_μ	z_ℓ	m	n
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)	(x)	(xi)	(xii)	(xiii)	(xiv)	(xv)	(xvi)	(xvii)	(xviii)	(xix)	(xx)	(xxi)
T50A1	50	500	50	493	5.0	2.816	50	493	5.0	2.86	50	493	2.43	3.31	50	493	5.0	3.14	50	493
T50A2	50	500	50	492	5.0	2.93	50	492	5.0	2.99	50	492	2.34	3.45	50	492	5.0	3.27	50	472
T50A3	50	500	50	490	5.0	3.00	50	490	5.0	3.00	50	490	2.13	3.47	50	490	5.0	3.25	50	485
T50B1	50	500	50	493	64.0	50.0	50	493	64.0	50.0	50	493	-	-	-	-	-	-	-	-
T50B2	50	500	50	491	65.0	50.0	50	493	65.0	50.0	50	491	-	-	-	-	-	-	-	-
T50C1	50	500	50	491	103.0	58.39	50	491	103.0	58.39	50	491	56.37	60.4	-	491	103.0	58.66	50	491
T50C2	50	500	50	492	100.0	58.91	50	492	100.0	58.93	50	492	-	-	-	-	-	-	-	-

Table II.6 which has the same entries as Table II.2. We only show the computational results for a small set of test problems since the perfomance of the techniques described above is very similar for the different problems generated in those classes. As expected the preliminary reductions are not efficient for this type of problems and the other methods applied do not perform well either. For the proportional cost problems it so happens that the LP procedure took too long (a limit of 100 seconds in the CDC 6500 was used) and, hence, procedure 2.13 is not completed for those problems. The same occured for test problem T50C2 .

2.9 CONCLUSIONS

In this Chapter we presented a procedure to perform reductions in the dimensions of large size SCP's. The method consists of preliminary reductions techniques, heuristics, lagrangean relaxation and linear programming, in such a way that:

(i) Large test problems with $n=10*m$ (n-number of columns;m-number of rows), are reduced to problems with $n \simeq m$ but still with the same density as the original ones.

(ii) Lower and upper bounds are obtained with a gap in all cases less than or equal to 10% of the value of the best solution available. For most of the test problems we tried out, this gap was below of 6% of the upper bound value.

(iii) For SCP's with costs equal to 1.0 for all the variables, the procedure fails to produce any significant reduction in the dimensions of the problems. This is even worse for the problems with costs proportional to the number of rows covered, for which the linear programming also can take too long to produce a lower bound.

It seems possible to achieve further improvements on this perfomance and the following conclusions may be taken into account in future research :

(iv) Although the quality of the greedy upper bound may be very poor it is worthwhile to use them to generate an initial feasible solution for the SCP.

Since only a small part of the computational time of the procedure is taken to generate the greedy cover, more and different selection functions might be used.

(v) The initial lower bound obtained from the greedy heuristic has a poor quality and can be improved by simple methods. Although, from our experience, this did not affect very much the outcome of the procedure, the possibility of improving the perfomance of the subsequent lagrangean relaxation is an open question. Furthermore, if other types of reductions were considered (such as penalties or row splitting) then the quality of the greedy lower bound, and hence the quality of the first lagrangean relaxation bound, becomes important.

(vi) Increasing the number of iterations allowed for the first phase of the lagrangean relaxation did not significantly increase the corresponding lower bound. However, there is the possiblity of finding better covers generated from the lagrangean solution. Later in this thesis we report improvements on the upper bound obtained from further iterations of the lagrangean relaxation (phase I), when solving problems with 400 rows and 4000 columns.

(vii) The 'core' problem LP_0 can possibly be reduced in terms of the number of rows without affecting the corresponding lower bound. Besides, the criterion for choosing the variables to include in N_0 has not been considered in depth and further research on this particular aspect must be carried out.

(viii) For the unicost SCP, it is worthwhile to solve the LP relaxation with $N_0=N$ and then trying an improvement on this bound using lagrangean relaxation.

(x) both for the unicost SCP and the proportional costs SCP, it is still possible that other reduction techniques may be useful. However, we will report later computational results derived from the application of decomposition and state space relaxation which are consistent by good for unicost SCPs.

Chapter 3

STATE SPACE RELAXATION FOR THE SCP

3.1 INTRODUCTION

Dynamic programming can be used to solve the SCP but this requires, even for small size problems, too much storage and time to be useful in practice. In *Hey [105]*, one way to reduce the dimension of the state space of a dynamic program associated with the SCP is presented. Instead of obtaining an optimal solution to the problem , a lower bound is computed by solving dynamic programming recursions on a smaller set of states. This corresponds to an idea recently developed and called "state space relaxation" (SSR) in *Christofides et al. [52]* where it is used for the vehicle routing problem. State space relaxation is a generalisation of lagrangean relaxation and, hence, can be embedded in a tree-search procedure in order to solve optimally the original problem.

In this Chapter the application of SSR to the SCP is developed and some results, both theoretical and practical, are given. In particular, two different relaxations are presented and tested. One can be seen as an extended lagrangean relaxation while the other is equivalent to the surrogate constraint relaxation of a SCP. Different ways to perform the state space modifications are studied and results from computational experience are shown. Also, the lower bound produced by SSR is compared with the linear programming relaxation bound for test problems whose dimension ranges from 10 to 50 rows and 100 to 500 columns, as well as test problems of the classes (II) and (III) considered in Chapter 2.

Chapter 3

3.2 DYNAMIC PROGRAMMING FORMULATION

3.2.1 Definition

Let us consider the SCP with **b** representing the m-dimensional array of 1s in the right-hand-side of the constraints and with a_i and a^j representing, respectively , the ith row and jth column of the constraint matrix $A=[a_{ij}],i=1,2,,...,m,j=1,2,...,n$. Let **s** be an m-dimensional 0-1 vector representing a set of rows, $k=\beta(s)$ the index of the last 1 in **s** and F(s) be the least cost of covering the set represented by **s**. The recursive function F(s) is defined as follows :

(3.1) $F(0)=0$

(3.2) $F(s)=\min_{j\epsilon N_k}(F(s-a^j)+c_j)$
for $k=1,2,...,m$ and $s\epsilon\beta^{-1}(k)$

where N_k is the set of columns with an entry of 1 in row k and c_j is the cost of a^j ,the jth column of the constraint matrix $A=[a_{ij}]$.

$F_m(\mathbf{b})$ is the optimal value of the SCP and the corresponding optimal solution is obtained by backtracking analysis in the final dynamic programming tableau. We will illustrate for the following example :

Example 3.1 :
$$\begin{aligned} \min\ & x_1 + 3x_2 + x_3 + 2x_4 + 2x_5 \\ \text{st.}\ & x_1 + x_2 + x_3 \geq 1 \\ & x_2 + x_5 \geq 1 \\ & x_3 + x_4 \geq 1 \\ & x_1 + x_2 + x_4 \geq 1 \\ & x_1,x_2,x_3,x_4,x_5=0 \text{ or } 1 \end{aligned}$$

Chapter 3

The values for F( ) are given in Tableau III.1 where , for instance , $F(s_6)$ with $s_6=(0,1,1,0)$ is computed as follows :

$$F(s_6) = \min_{j\in N_3} (F(s_6-a^j)+c_j) = \min (F(s_6-a^3)+c_3 , F(s_6-a^4)+c_4) =$$
$$= \min (F(s_2)+1 ,F(s_2)+2) = 3$$

i	s_i	k	F
0	(0,0,0,0)	0	0
1	(1,0,0,0)	1	1
2	(0,1,0,0)	2	2
3	(1,1,0,0)	2	3
4	(0,0,1,0)	3	1
5	(1,0,1,0)	3	1
6	(0,1,1,0)	3	3
7	(1,1,1,0)	3	3
8	(0,0,0,1)	4	1
9	(1,0,0,1)	4	2
10	(0,1,0,1)	4	3
11	(0,0,1,1)	4	1
12	(1,1,0,1)	4	4
13	(1,0,1,1)	4	2
14	(0,1,1,1)	4	3
15	(1,1,1,1)	4	4

Tableau III.1
Values of the dynamic programming recursion
for the SCP of Example 3.1

Using the usual dynamic programming nomenclature , vectors s are called states and the vector domain S , on whieh the recursive function is defined, is designated the state space. The k values define the stages for the dynamic program and will be omitted when s_i is precisely identified.

The optimal value to the problem of example 1 is equal to 4 and two alternative optimal

Chapter 3

solutions exist - $(x_1=x_3=x_5=1)$ and $(x_2=x_3=1)$. The dynamic process can be graphically displayed as in Fig. 3.1, where the values for F are within square brackets and an arrow from s_p to s_l means that $F(s_l)$ is obtained from $F(s_p)$ by fixing the variable indicated next to the arrow.

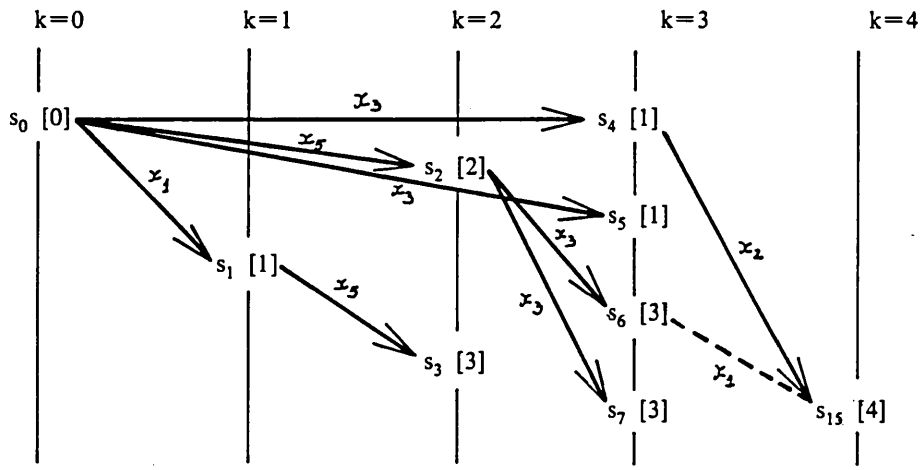


Figure 3.1

Dynamic process for solving the SCP of Example 3.1

As is suggested by Fig 3.1, the values $F(s_8)$ to $F(s_{14})$ are not necessary, and only the state $s_{15}=b$ need be considered at the last stage .

Storage requirements and computational time are the disadvantages of dynamic programming in comparison with other approaches to the SCP. In fact, for each stage k the number of states is $\Sigma_{l=1}^{k}\binom{k-1}{l-1} = 2^{k-1}$ vectors and the recursive function F has to be stored in an array with dimension $2^{m-1}+1$. In the worst case, the total number of comparisons necessary to solve the SCP using (3.1) and (3.2) is $\Sigma_{k=1}^{m} 2^*(\#N_k -1)$, *Horowitz and Sahni [111].*

In section 3.3 a state space relaxation technique for the dynamic program is used in order to reduce significantly both the state space dimension and the number of operations . However , a lower bound is obtained instead of the optimal solution to the SCP. This lower bound will

then be imbedded in a branch-and-bound to solve the problem .

Next , a different (but equivalent) dynamic programming formulation to the one above is presented, and some properties of it are discussed which are related to the state space relaxation .

3.2.2 An Alternative Dynamic Programming Formulation

Let S be a set of rows , which can be represented by an m-dimensional 0-1 vector s and let M be the index set of all rows. The recursive function $F_k(S)$ is defined as the minimum cost for covering the set S using the first k columns. That is,

$$(3.3) \qquad F_k(S) = \min_{S' \in \Delta^{-1}(S)} [F_{k-1}(S') + v_k(S',S)]$$
$$\text{for } k=1,2,...,n \text{ and } S \subseteq M$$

with initial values

$$(3.4) \qquad F_0(S) = \begin{cases} 0 & \text{if } S=\Phi \\ +\infty & \text{otherwise} \end{cases}$$

and $\Delta^{-1}(S) = \{S' : S' \subseteq S\}$. The value $v_k(S',S)$ is the minimum additional cost for covering S with S' already covered and considering only the column k :

$$(3.5) \qquad v_k(S',S) = \begin{cases} 0 & \text{if } S=S' \\ c_k & \text{if } S-S' \subseteq M_k \\ \infty & \text{if } S-S' \not\subseteq M_k \end{cases}$$

where M_k is again the index set $\{ i : a_{ik}=1\}$ and S-S' is the normal set difference operation between S and S'.

Chapter 3

Solving the dynamic program (3.3)-(3.5) the recursive function $F_k(S)$ has to be computed for all sets $S \subseteq M$, at each state k . Moreover, the computation of $F_k(S)$ implies the comparison between $2^{\#S}$ values for $k>m$. However, these values can be significantly reduced as will be seen below.

3.2.3 Properties

For each state S and at a fixed stage k the set $\Delta(S)$ is restricted as follows :

Property 3.2 : The set $\Delta(S)$ defined for a state S can , at each stage k, be reduced to :

$$\Delta_k(S) = \{S \ , \ S\text{-}M_k\} \tag{3.6}$$

and (3.3) is equivalent to

$$F_k(S) = \min\ [F_{k-1}(S) \ , \ F_{k-1}(S\text{-}M_k) + c_k] \tag{3.7}$$

for $k=1,2,\ldots,n$; $S \subseteq M$

The proof is immediate. Hence, the number of comparisons necessary to obtain , at each stage k, the value $F_k(S)$ for a fixed S has been reduced to just one. A further improvement is possible by reducing the number of states S for which F_k must be calculated when k is fixed.

Property 3.3 : Let $F_k(S)$ be defined as in (3.7) with initial values (3.4).

F_k is subadditive , that is :

$$F_k(S' \cup S'') \leq F_k(S') + F_k(S'') \tag{3.8}$$

Proof : we prove the above statement by induction on k .

If $k=0$ it is obvious, from (3.4), that (3.8) holds. Now , assuming that inequality (3.8) is valid for k let us show that the same is true for $k+1$. By definition, there are four possible combinations of values to $F_{k+1}(S')$ and $F_{k+1}(S'')$:

(1) $F_{k+1}(S')=F_k(S')$ and $F_{k+1}(S'')=F_k(S'')$

by hypothesis of induction

Chapter 3

$$F_{k+1}(S') + F_{k+1}(S'') \geq F_k(S' \cup S'')$$

and , by definition

$$F_{k+1}(S') + F_{k+1}(S'') \geq F_{k+1}(S' \cup S'')$$

(2) $F_{k+1}(S')=F_k(S')$ and $F_{k+1}(S'')=F_k(S''-M_k)+c_k$

$$F_{k+1}(S') + F_{k+1}(S'') = F_k(S') + F_k(S''-M_k) + c_k$$

(by the induction hypothesis) $\geq F_k(S' \cup (S''-M_k)) + c_k$

(by definition) $\geq F_k((S' \cup S'')-M_k) + c_k$

(by definition) $\geq F_{k+1}(S' \cup S'')$

(3) $F_{k+1}(S')=F_k(S'-M_k)+c_k$ and $F_{k+1}(S'')=F_k(S'')$

the proof is identical to the previous one

(4) $F_{k+1}(S')=F_k(S'-M_k)+c_k$ and $F_{k+1}(S'')=F_k(S''-M_k)+c_k$

again by hypothesis :

$$\begin{aligned} F_{k+1}(S')+F_{k+1}(S'') &\geq F_k((S'-M_k) \cup (S''-M_k)) + 2*c_k \\ &\geq F_k((S' \cup S'')-M_k) + 2*c_k \\ &\geq F_k((S' \cup S'')-M_k) + c_k \\ &\geq F_{k+1}(S' \cup S'') \end{aligned}$$

If the SCP is feasible then for any $S \subseteq M$ a cover $(x_{i(1)},\ldots, x_{i(s)})$, with $i(j)<i(j+1)$ $(j=1,2,\ldots,s-1)$, exists such that $S \subseteq \cup_{j=1}^{s} M_{i(j)}$ and $F_s(S)=F_s(\cup_{j=1}^{s} M_{i(j)})$. Hence, at each stage k the state space S can be reduced to

(3.9) $$S_k = \{S \epsilon S : S = \cup_{i=k1}^{k2} M_i,\ 0 \leq k1 \leq k2 \leq k\}$$

with $M_0=\Phi$

The maximum dimension of S_k is given by :

(3.10) $$\begin{aligned} \dim (S_k) &\leq 2^k \text{ for } 1 \leq k \leq m \\ &\leq 2^m \text{ for } m < k \leq n \end{aligned}$$

and the maximum number of values F( ) to be computed is still exponentially dependent on the number of rows of the SCP.

Chapter 3

In spite of being more complex the dynamic programming formulation to the SCP defined by (3.4) and (3.7) is more suitable for SSR application than the formulation by (3.1) and (3.2). The dynamic programming recursion can be improved further by using the following property which is a direct consequence of (3.8).

Property 3.4 : Let $F_k(S)$ be defined by (3.4) and (3.7).The following 'a priori' tests are valid :

(a) if $F_{k-1}(M_k) \leq c_k$ then $F_k(S) = F_{k-1}(S)$
for all $S \subseteq M$
(ie. column k is conceptually removed)

(b) denoting by $d = \min_{j \in N}(c_j / \#M_j)$ and $\bar{F}(M)$ an upper bound to $F_n(M)$,
if $F_k(S) + (m - \#S) * d \geq \bar{F}(M)$
then $F_k(S)$ can be ignored and S is not stored.

The proof is immediate.

3.2.4 The Dynamic Programming Procedure

The final dynamic programming recursion , DP ,considered for the SCP is :

$$(3.11) \qquad F_k(S \cup M_k) = \min [F_{k-1}(S \cup M_k) , F_{k-1}(S) + c_k]$$
for $k = 2,...,n$ and $S \in S_{k-1}$

where F_k is not computed if either the tests (a) or (b) of property 3.4 are effective, and $F_{k-1}(S \cup M_k)$ is only considered if $(S \cup M_k) \in S_{k-1}$.

The family of sets S_k is computed recursively as :

$$(3.12) \qquad S_k = S_{k-1} \cup \{ X \cup M_k : X \in S_{k-1} \}$$
for $k = 2,...,n$

Chapter 3

with the initial values

(3.13) $S_1 = \{ M_1, \Phi \}$

$F_1(M_1) = c_1$ and $F_1(\Phi) = 0$

A description of a procedure based on (3.11)-(3.12) and (3.13) for solving the SCP, is given below. The corresponding flowchart is shown in Figure 3.2.

Procedure 3.5 . *Dynamic programming procedure*

Step 1 . initialisation

$k=1$

$\ell=2$

$F(\Phi)=0$

$F(M_1)=c_1$

$H(S)=(0,0)$ for all $S \epsilon S$

$H(M_1)=(1,1)$

$X_{(1)}=\Phi$

$X_{(2)}=M_1$

$S_1=\{X_{(1)},X_{(2)}\}$

Step 2 . next variable

if $k=n$ go to 4

$k=k+1$

$ne=\ell$

$S_k=S_{k-1}$

Step 3 . computing F,H and S

for $j=1$ to ℓ do

$Y=X_{(j)} \cup M_k$

if $Y \epsilon S_k$ then

if $F(Y)>F(X_{(j)})+c_k$ then

$F(Y)=F(X_{(j)})+c_k$

$H(Y)=(k,j)$

endif

go to repeat

else $ne=ne+1$

$X_{(ne)}=Y$

$F(X_{(ne)})=F(X_{(j)})+c_k$

$H(X_{(ne)})=(k,j)$

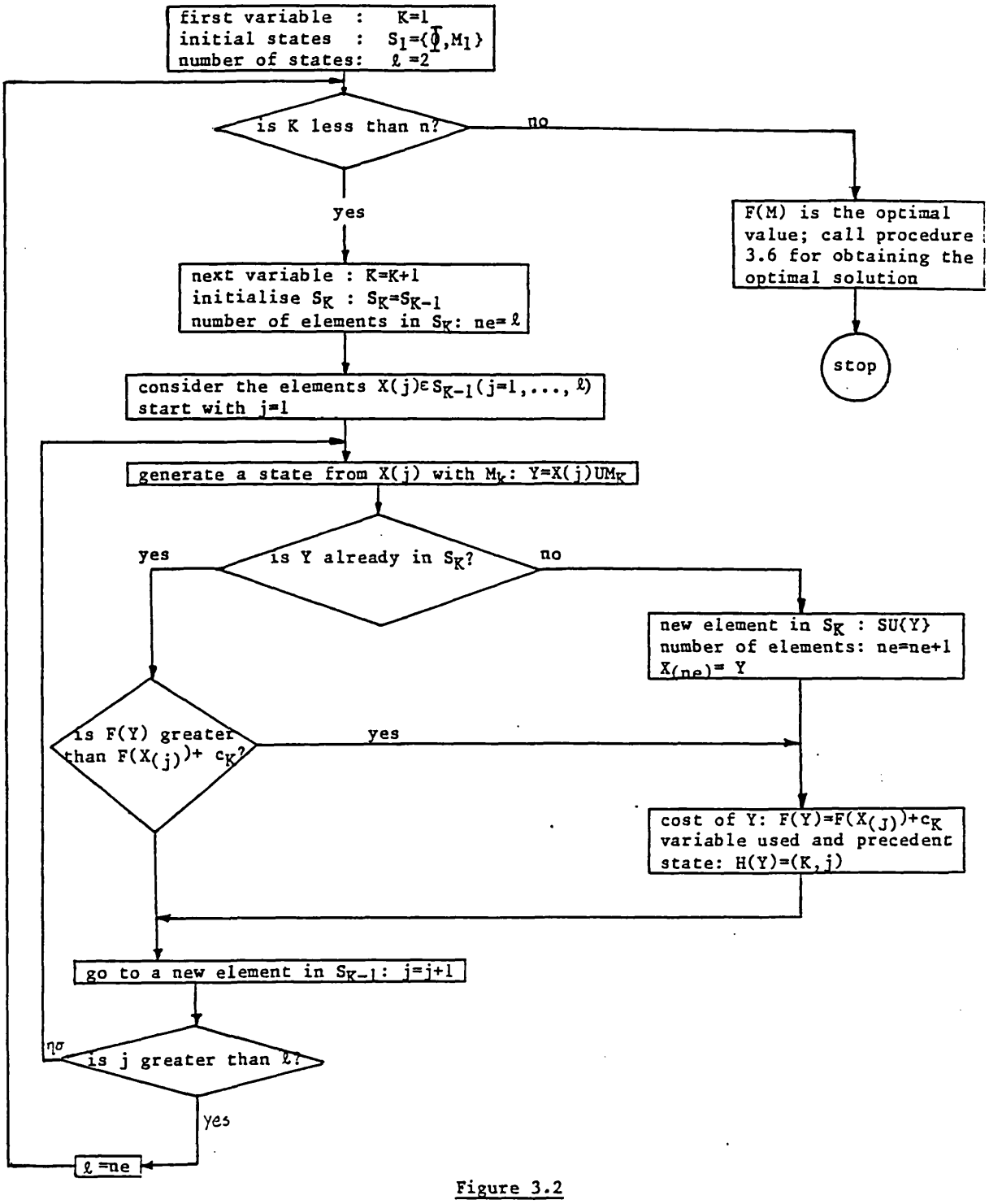


Figure 3.2

Flow diagram for procedure 3.5

Chapter 3

$S_k = S_k \cup \{X_{(ne)}\}$

endif

repeat

go to 2

endif

Step 4 . obtaining the optimal solution

F(M) is the optimal value

call procedure 3.6

stop

Procedure 3.6 . *Obtaining the optimal variables*

Step 1 . initialisation

$Z = \Phi$

$X = M$

Step 2 . constructing the optimal solution

$(\ell,j) = H(X)$

$Z = Z \cup \{\ell\}$

Step 3 . updating set X

if $j=1$ then go to step 4

otherwise do $X = X_{(j)}$ and go to step 2

Step 4 . optimal solution

Z is the index set of the optimal variables

stop

Example 3.7

Tableau III.2 contains the values of F and H for the problem of example 3.1 . For each $k=1,...,4$, the sets $X_\ell \in S_k$ are indicated with the corresponding values of F and H. For $k=5$ only M is considered and the optimal value is $F(M)=4.0$.

Using the procedure 3.5 to obtain the optimal solution :

. $Z=\Phi$

$X=\{1,2,3,4\}$

. $(\ell,j)=H(\{1,2,3,4\})=(3,3)$

$Z=Z\cup\{3\}=\{3\}$

. $X=\{1,2,4\}$

. $(\ell,j)=H(\{1,2,4\})=(2,1)$

$Z=Z\cup\{2\}=\{2,3\}$

. since $j=1$ the optimal solution is given by the set Z

i.e., $x_2=x_3=1$

	k=1				k=2	
$c_1=1$		$M_1=\{1,4\}$		$c_2=3$		$M_2=\{1,2,4\}$
1	1	2	1	1	2	3
S	ϕ	{1,4}	S	ϕ	{1,4}	{1,2,4}
F	0	1	F	0	1	3
H	(0.0)	(1.1)	H	(0.0)	(1.1)	(2.1)

	k=3					
$c_3=1$				$M_3=\{1,3\}$		
1	1	2	3	4	5	6
S	ϕ	{1,4}	{1,2,4}	{1,3}	{1,3,4}	{1,2,3,4}
F	0	1	3	1	2	4
H	(0.0)	(1.1)	(2.1)	(3.1)	(3.2)	(3.3)

	k=4						
$c_4=2$				$M_4=\{3,4\}$			
1	1	2	3	4	5	6	7
S	ϕ	{1,4}	{1,2,4}	{1,3}	{1,3,4}	{1,2,3,4}	{3.4}
F	0	1	3	1	2	4	2
H	(0.0)	(1.1)	(2.1)	(3.1)	(3.2)	(3.3)	

	k=5	
$c_5=2$		$M_5=\{2\}$
1	6	
S	{1,2,3,4}	
F	4	
H	(3.3)	

Table III.2
Values of F,H and S when using procedure 3.5
to solve the SCP of example 3.1

Chapter 3

3.3 STATE SPACE RELAXATION

3.3.1 Definition

In this section a relaxation method of the dynamic programming formulation for the SCP is presented in order to reduce the state space dimension of the dynamic program. Instead of an optimal solution for the SCP, a lower bound is obtained which can be improved either using penalties in a lagrangean fashion or using state space modifications. This bound is then to be embedded in a branch-and-bound scheme to solve the SCP.

Let us consider then the dynamic programming formulation for the SCP defined by (3.3),(3.4) and (3.5). Now, let g be a mapping function from the state space $\{S: S \subseteq M\}$ to a lower dimensional vector domain **Q** and let Ω be a set function such that :

(3.14) if $S' \epsilon \Delta^{-1}(S)$ then $g(S') \epsilon \Omega^{1}(g(S))$

A new dynamic program can be defined in **Q** as follows :

(3.15) $f_k(q) = \min \{ f_{k-1}(q') + w_k(q',q) \}$

the minimum is computed for $q' \epsilon \Omega^{1}(q)$ with $k = 1,2,...,n$.

with initial values

(3.16) $$f_1(q) = \begin{cases} 0 & \text{if } q = g(\Phi) \\ c_1 & \text{if } q = g(M_1) \\ \infty & \text{if } q > g(M_1) \end{cases}$$

and

(3.17) $w_k(q',q) = \min \{ v_k(S',S) : g(S') = q' , g(S) = q \}$

with v_k defined as in (3.5).

Property 3.8 : For any state $S \subseteq M$ and stage k

(3.18) $$f_k(g(S)) \leq F_k(S)$$

Proof : if k=1 the inequality is obvious. Now, let us assume that this is true for k and prove the same for k+1 :

$$f_{k+1}(g(S)) = \min_{q' \in \Gamma^{-1}(g(S))} [\, f_k(q') + w_{k+1}(q', g(S)) \,]$$

(from (3.14)) $$\leq \min_{S' \in \Delta^{-1}(S)} [\, f_k(g(S')) + w_{k+1}(g(S'), g(S)) \,]$$

(from hypothesis) $$\leq \min_{S' \in \Delta^{-1}(S)} [F_k(S') + w_{k+1}(g(S'), g(S)) \,]$$

(from (3.17)) $$\leq \min_{S' \in \Delta^{-1}(S)} [\, F_k(S) + v_{k+1}(S', S) \,]$$

Corollary 3.9 : (a) $f_n(g(M))$ is a lower bound to v(SCP) the optimal value to the SCP.

(b) f_n is such that :

(3.19) $$f_n(g(M_j)) \leq c_j, \text{ for all } j=1,2,...,n$$

3.3.2 Properties of the Relaxed Recursion

The computation of $w_k(q',q)$ from (3.17) still requires the knowledge of $v_k(S',S)$ calculated in the original state space. This is overcome using (3.7) in order to obtain the 'relaxed' state space recursion corresponding to (3.8). In fact, the set function Ω (as defined in (3.14)) can be restricted in a similar way to (3.7) to become :

(3.20) $$\Omega_k^1(q) = \cup_{S \in g^{-1}(q)} \{\, g(S) \,,\, g(S - M_k) \,\}$$

and $w_k(q',q)$ is then simplified to :

(3.21) $$w_k(q',q) = \begin{cases} 0 & \text{if } q' = q = g(S) \\ c_k & \text{if } q' = g(S - M_k) \end{cases}$$

Chapter 3

Therefore, the recursion in the relaxed space can be stated as :

(3.22) $f_k(q) = \min_{S \epsilon g^{-1}(q)} [f_{k-1}(q) , f_{k-1}(g(S-M_k))+c_k]$
for $k=1,2,...,n$ and $q \epsilon Q$

A last link with the original state space remains through the determination of the sets S such that $g(S)=q$. However, subadditivity of g provides the way to obtain $f_k(q)$ without resorting to explicit computation of those sets. Thus,

Property 3.10 : If g is a subadditive real function then

(3.23) $f_k(q) \geq \min [f_{k-1}(q) , f_{k-1}(q-g(M_k))+c_k]$

where $q-g(M_k)=\max(0,q-g(M_k))$

Proof : from subadditivity of g,

$g(S-M_k) \geq g(S)-g(M_k)$

$\geq q-g(M_k)$

for all $S \subseteq M$ such that $g(S)=q$ and (3.23) comes as an immediate result.

Now, using expression (3.23) with the equality and the initial conditions (3.16), the final dynamic programming recursion, DPR, on the relaxed state space is defined as :

(3.24) $f_k(q+g(M_k)) = \min [f_{k-1}(q+g(M_k)) , f_{k-1}(q)+c_k]$
for $k=2,...,n$ and $q \epsilon Q_{k-1}$

where $f_{k-1}(q+g(M_k))$ is only considered if $q+g(M_k) \epsilon Q_{k-1}$, with

(3.25) $Q_k = Q_{k-1} \cup \{ q+g(M_k) : q \epsilon Q_{k-1} \}$

and

(3.26) $Q_1 = \{ g(\Phi) , g(M_1) \}$

$f_1(g(\Phi) = 0$

$f_1(g(M_1)) = c_1$

Therefore, $f_n(g(M))$ computed in the lower dimension state space Q is a lower bound to the SCP. Figures 3.3 and 3.4 illustrate the way state space relaxation operates.

Figure 3.3 shows the 'shrinking' process of the original state space $S=\{S: S\subseteq M\}$ and the computation of $f_k(q)$, a lower bound to $f_k(g(S))$ for $g(S)=q$, is graphically displayed in Fig. 3.4. The function ρ is defined by :

(3.27) $\rho(q) = \{ S\subseteq M : g(S)\geq q \}$

3.3.3 Forms of the Mapping Function g(.)

As is suggested by Figures 3.2 and 3.3 ., the state space relaxation is defined by g and Ω. The function g can take different expressions for the SCP, such as :

(3.28)
(i) $g(S) = \#S$, the cardinality of S
(ii) $g(S) = \Sigma_{i\in S} q_i$, q_i is a non-negative integer weight associated with ith row of M
(iii) $g(S) = \beta_S$, index of the last row in S
(iv) $g(S) = \alpha_S$, index of the first row in S
(v) $g(S) = (\alpha_1,\alpha_2,...,\alpha_r)$, where $\alpha_k = \#(S\cap A_k)$ for $A_k\subseteq M$ and $1\leq k\leq r\leq m$

Several other expressions can be stated for g, in particular any combination of the previous ones. For instance, in *Hey [105]* the following two expressions were considered for g :

(vi) $g(S) = (\#S,\beta_S)$, β_S given as in (iii)
(vii) $g(S) = (\#S,\alpha_S,\beta_S)$, α_S and β_S given as before

Case (i) corresponds to the 'knapsack type' relaxation for the SCP and case (ii) is equivalent to the surrogate constraint relaxation.

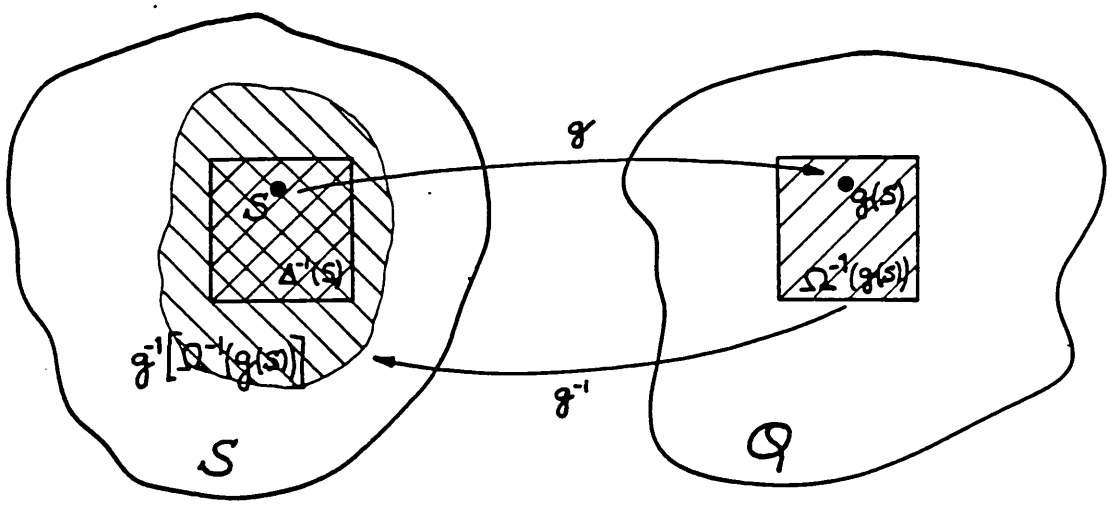


Figure 3.3

"Shrinking" effect of the mapping function g

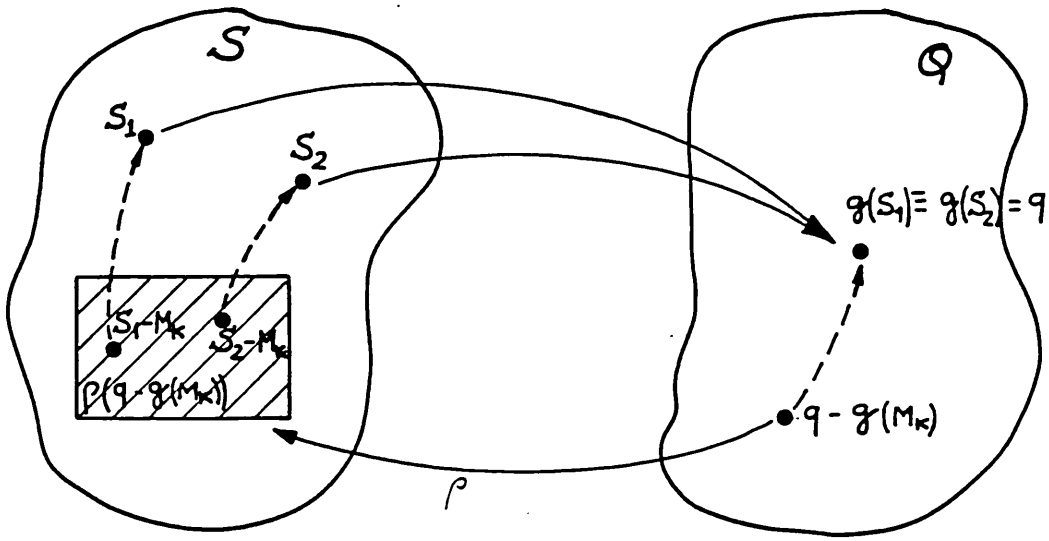


Figure 3.4

Graphical representation of state space relaxation process

Chapter 3

Example 3.11

Let us work out the problem of example 3.1 and using the weighted knapsack relaxation ((3.28)-(ii)). Consider $q_1=q_3=q_4=1$ and $q_2=2$ as the weights for the relaxation mapping function for the initial problem formulated as in (3.5)-(3.6) and (3.7). The values of $f_k(q)$ for $q=0,1,...,5$ and $k=1,...,5$ are shown in the Tableau III.3 where, for instance :

. $f_2(4) = f_1(0)+c_2 = 0.0+3.0 = 3.0$

$f_2(5) = f_1(1)+c_2 = 1.0+3.0 = 4.0$

. $f_3(4) = \min (f_2(4),f_2(2)+c_3) = \min (3.0,2.0) = 2.0$

. $f_5(5) = \min (f_4(5),f_4(3)+c_5) = \min(4.0,4.0) = 4.0$

k \ q	0	1	2	3	4	5
1	0	1	1			
2	0	1	1	3	3	4
3	0	1	1	2	2	4
4	0	1	1	2	2	4
5	0	1	1	2	2	4

Tableau III.3
Values of $f_k(q)$ for the SCP of example 3.1 using the dynamic programming formulation given by (3.5)-(3.6) and (3.7)

Since the solution corresponding to the lower bound $f_5(5)$, $(x_1=x_3=x_2=1)$ obtained by backtracking analysis, is feasible the optimal value to problem is $v(SCP)=f_5(5)=4$. Now, let us consider the same relaxation mapping function but with the dynamic programming formulation defined by (3.1) and (3.2). In this case the relaxed recursion is :

(3.29) $f_k(q)= \min_{j\in N_{k'}} (f_{k'}(q-g(M_j)+c_j)$
for $1\leq k'\leq k$

with initial values given by (3.16). Tableau III.4 contains the values of $f_k(q)$ for the example with, for instance,

. $f_1(4) = c_2 = 3.0$

. $f_2(5) = \min_{j=2,5} (f_1(5-g(M_j))+c_j) = \min (f_1(1)+3.0, f_1(3)+2.0) = 4.0$

. $f_3(5) = \min_{j=3,4} (f_2(3)+c_3, f_1(3)+c_4) = 3.0$

k	q 0	1	2	3	4	5
1	0	1	1	3	3	
2	0	1	1	2	2	4
3	0	1	1	2	2	3
4	0	1	1	2	2	3

Tableau III.4
Values of $f_k(q)$ for the SCP of example 3.1 using the dynamic programming formulation given by (3.1) and (3.2)

The lower bound $f_4(g(S_4))=3$ is worse than that obtained from the relaxation of a more complex initial dynamic programming formulation. In fact, this example shows that the effectiveness, or otherwise, of the state space relaxation in producing bounds is, for the same mapping function, dependent on the dynamic programming formulation. The same occurs with lagrangean relaxation relative to the integer programmming formulation of the problem being solved.

3.3.4 Subadditivity and Reduced Costs

The recursive function $f_n(\ )$ is not necessarily subadditive onto the relaxed state space Q. For instance, we have $f_5(2)+f_5(3)<f_5(5)$ in the example above . The subadditivity of $F_n(\ )$ defined on the original state space S can be violated because of the relaxation, but reduced costs for the variables can be obtained using $f_n(\ )$. In fact, from the definition, the cost of imposing the variable x_j in the solution is

(3.30) $\quad t_j=c_j+f_n(g(M-M_j))$

If z_u is a known upper bound and $t_j \geq z_u$ then x_j is equal to 0 for any feasible solution better

than z_u. Therefore, the reduced cost of variable x_j given by $f_n(\ )$, computed in the lower dimension state space, is :

(3.31) $\quad s_j = \max\ [0, c_j + f_n(g(M-M_j)) - f_n(g(M))]$

The next result is a general duality theory property that applies for the state space relaxation:

Property 3.12 : (a) If a row i exists such that

(3.32) $\quad d_i = \min_{j \in Ni} s_j > 0$

then the lower bound can be increased to

$z_l = f_n(g(M)) + d_i$

(b) let S be an index row subset of M and C be the family of all covers C(S) to S. Then :

(3.33) $\quad v(SCP) \geq \min_{C(S) \in C} [\Sigma_{j \in C(S)} c_j + f_n(g(M-M_{C(S)})]$

where $M_{C(S)} = \cup_{j \in C(S)} M_j$

(c) if $d_i = 0$ for all $i = 1,2,...,m$ then $X = \{x_j : s_j = 0\}$ is a feasible solution to the SCP

Proof : (a) an immediate proof comes from the inclusion in the SCP of the redundant constraint :

(A) row i is covered

The mapping function is restricted to :

$g(S) = (g(S), \delta_i)$ ($\delta_i = 1$ if $i \in S$; $\delta_i = 0$ otherwise)

Hence,

$$v(SCP) \geq f_n(g(M)) \geq \min_{j \in Ni} [c_j + f_n(g(M-M_j))]$$
$$\geq f_n(g(M)) + \min_{j \in N_i} [s_j]$$
$$\geq f_n(g(M)) + d_i$$

The proofs for (b) and (c) are immediate

In spite of requiring some computational effort because of the inspection of all possible combination of columns covering the row set S, result (b) of property 3.12 is worthwhile for the cases where $\#S=2$. This will be illustrated later in the present Chapter.

3.3.5 State Space Ascent

State space relaxation can be thought of as generalisation of lagrangean relaxation in integer programming. Constraints in integer programming appear as state variables in dynamic programming recursions and, hence, constraint relaxation in integer programs is equivalent to state space relaxation in dynamic programs. The generalisation comes from the possibility of using non-linear mapping functions, such as the cardinality of a set or the index of a particular row. Moreover, two different procedures can be used to increase the lower bound obtained by state space relaxation :

(i) using penalties in a lagrangean fashion
(ii) using state space modifications

In both cases the objective is to force the solution of the relaxed problem closer to feasiblity and, naturally, improve the lower bound. Two different mapping functions are considered and the respective state space relaxations will be studied in detail :

(SSR1) $g(S)=(\#S,\alpha,\beta)$

where $\#S$ is the cardinality of S

α is the index of the first row in S

β is the index of the last row in S

and

(SSR2) $g(S)=\Sigma_{i\epsilon S}q_i$

where q_i is a non-negative weight chosen to be associated with row i

Penalties and subgradient optimization are used for SSR1 in order to produce a better state

space ascent. On the other hand, state space modifications are adopted for SSR2.

3.4 RELAXATION SSR1

3.4.1 Recursion

Considering the dynamic programming formulation to the SCP defined by (3.11),(3.12) and (3.13), with the mapping function SSR1, the relaxed recursions corresponding to (3.24),(3.25) and (3.26) are given as :

$$(3.34)\qquad f_k((s,\alpha,\beta)\oplus(s_k,\alpha_k,\beta_k)) = \min\{f_{k-1}((s,\alpha,\beta)\oplus(s_k,\alpha_k,\beta_k)),\ f_{k-1}(s,\alpha,\beta)+c_k\}$$
$$\text{for } k=2,\ldots,n \text{ and } (s,\alpha,\beta)\in Q_{k-1}$$

where

$$(3.35)\qquad Q_k=Q_{k-1}\cup\{(s,\alpha,\beta)\oplus(s_k,\alpha_k,\beta_k) : (s,\alpha,\beta)\in Q_{k-1}\}$$
$$s_k=\#M_k$$
$$\alpha_k \text{ is the index of the first row in } M_k$$
$$\beta_k \text{ is the index of the last row in } M_k$$

with initial values :

$$(3.36)\qquad Q_1=\{g(\Phi),g(M_1)\}=\{(0,m+1,0),(\#M_1,\alpha_1,\beta_1)\}$$
$$f_1(g(\Phi))=0$$
$$f_1(g(M_1))=c_1$$

We define $g(\Phi)=(0,m+1,0)$ in order to be consistent with the operation $\oplus$ which is defined in the following way :

$$(3.37)\qquad (s,\alpha,\beta)\oplus(s_k,\alpha_k,\beta_k) = (s+s_k-\Sigma_{i,j}\delta_{ij},\ \min(\alpha,\alpha_k),\ \max(\beta,\beta_k))$$

where

$$(3.38)\qquad \delta_{ij} = \begin{cases} 1 \text{ if } i=j \\ \qquad\qquad \text{for } i=\alpha,\beta\ ;\ j=\alpha_k,\beta_k \\ 0 \text{ otherwise} \end{cases}$$

Chapter 3

k	l	s	α	β	f
1	1	0	5	0	0
1	2	2	1	4	1
2	3	0	5	0	0
2	4	2	1	4	1
2	5	3	1	4	3
3	6	0	5	0	0
3	7	2	1	4	1
3	8	3	1	4	2
3	9	2	1	3	1
3	10	4	1	4	4
4	11	0	5	0	0
4	12	2	1	4	1
4	13	3	1	4	2
4	14	2	1	3	1
4	15	4	1	4	4
5	16	0	5	0	0
5	17	2	1	4	1
5	18	3	1	4	2
5	19	2	1	3	1
5	20	4	1	4	4
5	21	1	2	2	2
5	22	3	1	3	3

Tableau III.5

Values $f_k(s,\alpha,\beta)$ for the state space relaxation SSR1 of example 3.1

Example 3.13 :

Let us illustrate the use of (3.37) and (3.38) for the problem of example 3.1 for the cases :

. $f_3(3,1,4)$

Chapter 3

$(2,1,4)\oplus(2,1,3)=(3,1,4)$

$f_3(3,1,4) = \min\{f_2(3,1,4) , f_2(2,1,4)+c_3\}=\min\{3 , 2\}=2$

. $f_3(4,1,4)$

$(3,1,4)\oplus(2,1,3)=(4,1,4)$

$f_3(4,1,4) = f_2(3,1,4)+c_3=4$

Tableau III.5 gives the complete list of values $f_k(s,\alpha,\beta)$ obtained from (3.37) and (3.38) for the SCP of the example 3.1.

3.4.2 Procedure SSR1

Next, we describe a procedure based on (3.34)-(3.35) and (3.36) for obtaining a lower bound for the SCP. The flowchart for this procedure is shown in Figure 3.5.

Procedure 3.14 . *Computing a lower bound to the SCP using SSR1*

input . m - number of rows
n - number of columns
c_k - cost of variable index k (k=1,2,...,n)
α_k - index of the first row in M_k (k=1,2,...,n)
β_k - index of the last row in M_k (k=1,2,...,n)
s_k - cardinality of M_k (k=1,2,...,n)

step 1 . initialisation
k=1
ℓ=2
s(1)=(0,m+1,0)
f(1)=0
h(1)=(0,0)
$s(2)=(\#M_1,\alpha_1,\beta_1)\oplus s(1)$
$f(2)=c_1+f(1)$
h(2)=(1,1)

step 2 . next variable
if k=n then go to 6
k=k+1
j=1
ne=ℓ

step 3 . merging

Chapter 3

$(s,\alpha,\beta)=s(j)+(s_k,\alpha_k,\beta_k)$
for i equal to 1 to ℓ do
if $(s,\alpha,\beta)\neq s(i)$ go to repeat
if $f(i)>f(j)+c_k$ then
$f(i)=f(j)+c_k$
$h(i)=(k,j)$
endif
go to 4
repeat
$ne=ne+1$
$s(ne)=(s,\alpha,\beta)$
$f(ne)=f(j)+c_k$
$h(ne)=(k,j)$

step 4 . new element in Q
$j=j+1$
if $j\leq\ell$ then go to 3
else go to 5

step 5. updating the state space dimension
$\ell=ne$
go to 2

step 6 . lower bound
for i equal 1 to ne do
if $s(i)\neq(m,1,m)$ go to repeat
f(i) is a lower bound to the SCP
$z_l=f(i)$
go to 7
repeat

step 7 . lower bound solution
$j=i$
$S=\Phi$
while j not equal 1 do
$(k,j)=h(j)$
$S=S\cup\{k\}$
repeat
S is the index set of the lower bound variables
stop

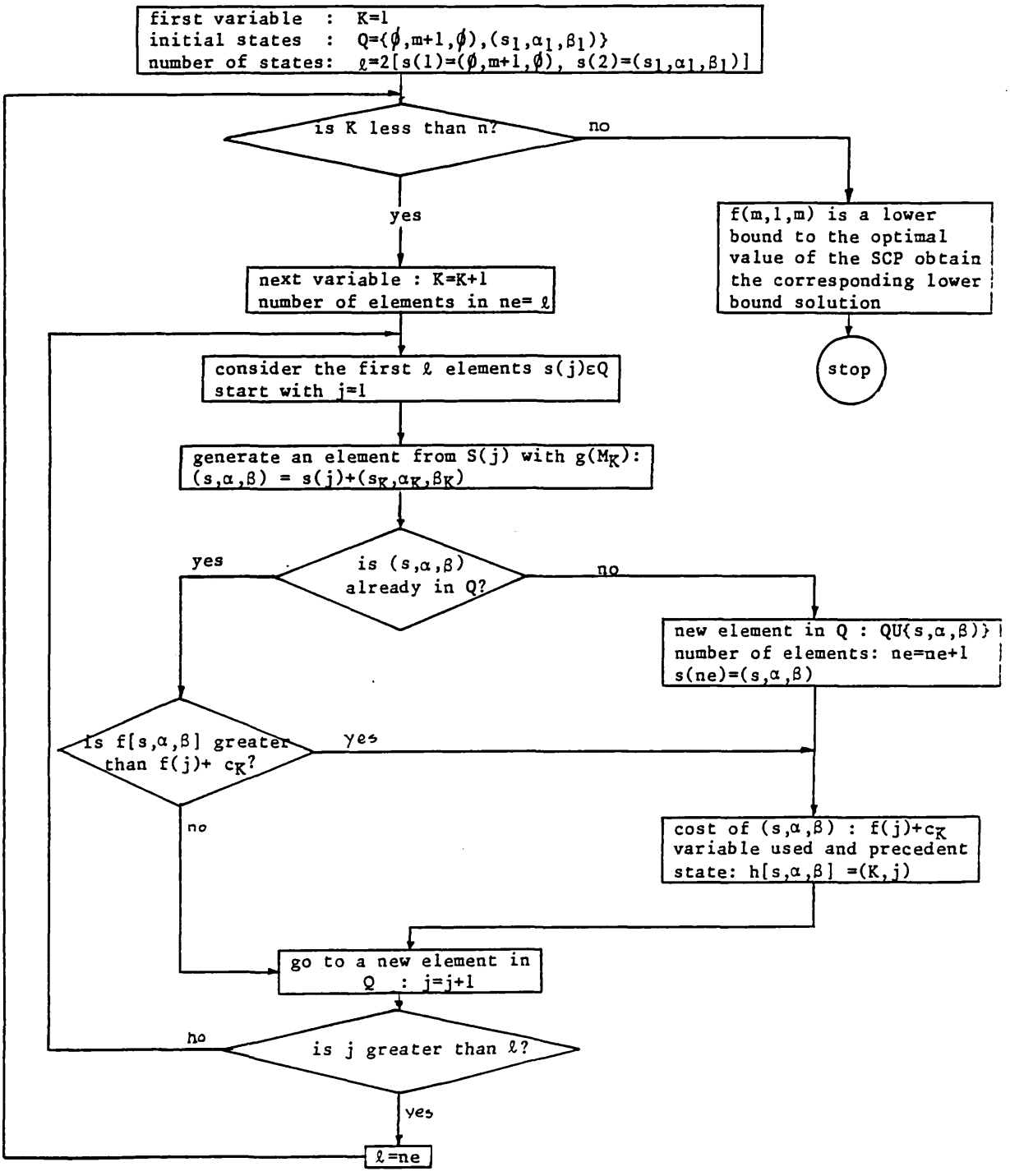


Figure 3.5

Flow diagram for procedure 3.14

Chapter 3

3.4.3 Subgradient Optimization

The lower bound solution given by the set S in procedure 3.14 can, naturally, be infeasible for the original SCP. Computing penalties λ_i (i=1,2,...,m) in a normal lagrangean fashion and updating subsequently the costs c_k, a new lower bound is obtained using procedure 3.14 again.

In fact, consider the SCP with the redundant constraints :

(3.39) $\quad H = \{ x : \Sigma_{k=1}^{n} s_k x_k \geq m \ , \ \alpha = \min_k \alpha_k = 1 \ , \ \beta = \max_k \beta_k = m \}$

where s_k, α_k and β_k are defined as before. Now, proceeding with the lagrangean relaxation of the row covering constraints the following relaxed problem is obtained :

$(SSR1_\lambda)$ $\quad \min \ \Sigma_{k=1}^{n}(c_k - \Sigma_{i \in M_j}\lambda_i)x_k + \Sigma_{i=1}^{m}\lambda_i$

subject to H

For each value of λ a lower bound $f_\lambda(m,1,m)$ is computed by procedure 3.14 and, hence, normal subgradient methods are used to solve the problem :

(DSSR1) $\quad \max_{\lambda \geq 0} f_\lambda(m,1,m)$

Subgradient optimisation was used with the penalties being computed at iteration number ℓ as follows :

(3.40) $\quad \lambda_i^{\ell} = \max \ (\ 0 \ , \ \lambda_i^{\ell-1} + \gamma^*[(z_u - z_l)^*(1 - \Sigma_{k \in N_i} x_k)] \ / \ [\Sigma_i(1 - \Sigma_{k \in N_i} x_k)^2] \ )$

The parameter γ is initially set equal to 2.0, being reduced to half of its value whenever the procedure fails to improve the bound after a fixed number of consecutive iterations. We set the initial values for the multipliers equal to u_i (i=1,2,...,m), the dual feasible solution obtained by the greedy heuristic described in Chapter 2.

Chapter 3

3.4.4 Improving the Bound

Consider the following index sets computed at iteration ℓ :

$$N_\ell^- = \{ k\epsilon N : c_k^\ell = c_k - \Sigma_{i\epsilon M_k}\lambda_i^\ell \leq 0 \} \quad (3.41)$$
$$N_\ell^+ = \{ k\epsilon N : c_k^\ell > 0 \}$$

Obviously, $x_k=1$ for $k\epsilon N_\ell^-$ and $\Sigma_{k\epsilon N_\ell^-} c_k^\ell + \Sigma_{i=1}^m \lambda_i^\ell$ is the lagrangean relaxation optimal value at iteration ℓ. Therefore, the lower bound given by SSR1 is at least as good as the lower bound produced by the lagrangean relaxation $LSCP_\lambda$ described in Chapter 2. On the other hand, procedure 3.14 takes longer than the lagrangean procedure to perform the same number of iterations. A better procedure in terms of both value of the lower bound and computational time required, can be achieved by combining those two aspects of each method.

With the modified costs c_k^ℓ produced during the lagrangean iterations, and which can be also negative, the initial state s(1) in step 1 of procedure 3.14 is not (0,m+1,0) but :

$$s(1) = \bigoplus_{j\epsilon N_\ell^-} (s_k,\alpha_k,\beta_k) = (s_0,\alpha_0,\beta_0) \quad (3.42)$$

Hence

$$f(1) = \Sigma_{k\epsilon N_\ell^-} c_k^\ell + \Sigma_{i=1}^m \lambda_i^\ell \quad (3.43)$$

is the corresponding lagrangean relaxation value. Then, a decision is taken whether to carry on with procedure 3.11 or to consider f(1) as the lower bound obtained at iteration ℓ and update the multipliers for iteration $\ell+1$.

A decision rule that performed reasonably well for the test problems was - if $s_0 \geq (2/3)*m$ then the state space relaxation procedure is completed. If not, the procedure is stopped and the lagrangean bound is considered.

Chapter 3

3.4.5 Removing Variables

Applying (3.27) to SSR1 the following value, computed at each iteration l, is a lower bound to the objective function when the variable x_k is fixed equal to 1 :

$$(3.44) \qquad t_k^l = c_k^l + \min_{Q \in Tk} f(Q)$$

where $T_k \subseteq Q$ is the subspace $\{Q \in Q: Q \oplus g(M_k) = (m,1,m)\}$. The variable x_k is removed if $t_k^l \geq z_u$, with z_u a nown upper bound to the SCP.

Performing the computation of (3.44) for all the variables consumes an amount of time which is too large. Then, instead of the exact value t_k^l we use a lower bound given by :

$$(3.45) \qquad \bar{t}_k^l = \max [\, f_l(m,1,m) \,,\, c_k^l + f_l(m-\#M_k,1,m)]$$

3.4.6 Example

Let us consider the same example we used to illustrate procedure 2.13 in Chapter 2. Applying procedure 2.9, an upper bound $z_u = 15.0$, a lower bound $z_l = 11.4$ and a dual feasible solution are available. The variables $x_{12}, x_{13}, x_{17}, x_{20}, x_{21}, x_{27}, x_{28}, x_{29}$ and x_{31} have been removed from the problem. We then apply procedure SSR1 (with the lagrangean improvement) to the following reduced problem :

k	1	2	3	4	5	6	7	8	9	10	11	14
c_k	1	1	1	1	1	1	1	1	2	2	2	2
α_k	1	2	3	4	5	6	7	8	1	3	5	9
β_k	1	2	3	4	5	6	7	9	15	10	13	11
s_k	1	1	1	1	1	1	1	2	2	2	2	2

k	15	16	18	19	22	23	24	25	26	30	32
c_k	2	3	3	3	4	4	5	5	5	7	9
α_k	2	5	11	1	1	2	3	1	3	2	1
β_k	11	14	15	14	12	14	15	13	15	13	15
s_k	2	2	3	5	3	5	6	5	5	7	10

The lagrangean multipliers are set equal to the dual variables obtained from the greedy

Chapter 3

heuristic:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ_i	0.9	0.9	0.9	0.9	1.0	0.9	1.0	0.9	0.1	0.9	0.9	0.15	0.9	0.15	0.9

and the corresponding reduced cost for the variables are :

k	1	2	3	4	5	6	7	8	9	10	11	14	15
s_k	0.1	0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.2	0.2	0.1	1.0	0.2

k	16	18	19	22	23	24	25	26	30	32
s_k	1.85	1.05	0.0	1.95	0.15	0.45	0.4	2.05	1.4	0.0

Now, as the first step for SSR1 we compute :

.$s(1)=g(M_5)\oplus g(M_7)\oplus g(M_8)\oplus g(M_{19})\oplus g(M_{32})=$

$=(1,5,5)\oplus(1,7,7)\oplus(2,8,9)\oplus(5,1,14)\oplus(10,1,15)=$

$=(2,5,7)\oplus(2,8,9)\oplus(5,1,14)\oplus(10,1,15)=$

$=(4,5,9)\oplus(5,1,14)\oplus(10,1,15)=(9,1,9)\oplus(10,1,15)=(15,1,15)$

$f(1)=\Sigma_{i=1}^{15}\lambda_i = 11.4$

Since $s(1)=g(M)=(15,1,15)$, this iteration of the state space relaxation is completed Since $s(1)=g(M)=(15,1,15)$, this iteration for the state space relaxation is completed and subgradient optimization is used for computing the new multipliers. The values are naturally the same as the ones obtained at iteration 2 of Step 2 in example 2.14, i.e. :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ_i	0.0	0.0	0.0	0.9	1.0	0.9	1.0	0.0	0.1	0.9	0.9	0.15	0.9	0.15	0.9

The new costs are then :

k	1	2	3	4	5	6	7	8	9	10	11	14	15
s_k	1.0	1.0	1.0	0.1	0.0	0.1	0.0	0.9	1.1	1.1	0.1	1.0	1.1

k	16	18	19	22	23	24	25	26	30	32
s_k	1.85	1.05	2.7	2.85	1.95	2.15	1.3	2.95	3.2	3.6

The variables are considered by increasing value of the reduced costs and the first step consists of computing :

. $s(1)=g(M_5)\oplus g(M_7)=(1,5,5)\oplus(1,7,7)=(2,5,7)$

Chapter 3

$f(1)=\Sigma_{i=1}^{15}\lambda_i+c_5'+c_7'=7.8$

Proceeding with the state space relaxation procedure, the variable x_4 is considered :

. $s(2)=g(M_4)\oplus(2,5,7)=(3,4,7)$

$f(2)=0.1+7.8=7.9$

The following variable to be considered (by increasing order of cost c'), is x_6 :

. $s(3)=g(M_6)\oplus s(1)=(1,6,6)\oplus(2,5,7)=(3,5,7)$

$f(3)=0.1+7.8 = 7.9$

. $s(4)=g(M_6)\oplus s(2)=(1,6,6)\oplus(3,4,7)=(4,4,7)$

$f(4)=0.1+7.9 = 8.0$

Continuing the process for all the variables, a final number of 204 states is obtained with the final lower bound - $f(15,1,15)=11.25$. In the next tableau we show the values for the states S such that $\alpha_S=1$ and $\beta_S=15$, which are used in the computation of the values t_k (see (3.45)) :

1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
β_1	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
$\hat{f}$	7.8	7.8	7.9	8.9	9.0	9.1	9.1	9.2	10.0	10.1	10.25	10.35	10.35	10.45	11.25

Applying (3.45), we obtain the following values t_k for the variables :

k	1	2	3	4	5	6	7	8	9	10	11	14
t_k	11.45	11.45	11.45	11.25	11.25	11.25	11.25	11.25	11.45	11.45	11.25	11.35
k	15	16	18	19	22	23	24	25	26	30	32	
t_k	11.45	12.2	11.4	12.8	13.2	12.05	12.15	11.4	13.05	12.4	12.6	

Note that for row 14 ($N_{14}=\{16,19,23\}$) :

. $\min_{j\in N14} = \min\ (12.2,12.8,12.05) = 12.05$

and this means that the lower bound is increased to the value $z_1=12.05$. Also, new costs t_k can be computed for the variables. For example :

. $t_1=\min_{j=16,19,23}[c_1'+c_j'+f_\lambda(m-(M_1\cup M_j),1,m)$

Hence,

j=16 . $g(M_1\cup M_{16})=g(\{1,5,14\})=(3,1,14)$

$c_1'+c_{16}'+f(12,1,15)=13.20$

j=19 . $g(M_1\cup M_{19})=g(\{1,2,3,12,14\})=(5,1,14)$

$c_1'+c_{19}'+f(10,1,15)=13.80$

j=23 . $g(M_1\cup M_{23})=g(\{1,2,5,8,11,14\})=(6,1,14)$

$c_1'+c_{23}'+f(9,1,15)=12.95$

Doing the same for all the other variables, we obtain :

k	1	2	3	4	5	6	7	8	9	10	11	14
t_k	12.95	13.05	12.95	12.05	12.05	12.05	12.05	12.95	12.25	12.25	12.05	12.85

k	15	16	18	19	22	23	24	25	26	30	32
t_k	13.20	12.20	12.20	12.80	14.00	12.05	13.10	12.25	13.90	14.15	13.45

Since $d_9=\min(t_8,t_{14},t_{26},t_{30})=12.85$, the lower bound is further improved to $z_l=12.85$ and the variable x_{30} is removed from the problem ($z_u=15.0$).

If we generate a cover from the lagrangean solution as we did for this problem in Chapter 2, then a new upper bound is obtained ($z_u=14.0$) and the variables $x_2,x_{15},x_{22},x_{24},x_{26},x_{32}$ are also removed from the problem. Then, a new iteration of the state space relaxation would be initiated and the process continued until either (i) a fixed number of iterations have been performed, or (ii) the lower bound has not increased for a fixed number of iterations, or (iii) the optimal solution is achieved ($z_l>z_u-1.0$).

3.4.7 Computational Results for SSR1

Table III.6 presents the computational results obtained using SSR1 for a set of test problems randomnly generated with the cost of the variables in the interval [1,99] and the number of variables n=10*m (m=10,20,30,40 and 50). For each value of m, five different problems were generated which are identified in Table III.6 by Pm.k, where m is the number of rows and k a number assigned to the test problem. All of the test problems have an average of 3.5 entries per column; which means that the average density ranges from 35% for the ten row problems (P10.1-P10.5) to 7% for the fifty row problems (P50.1-P50.5).

The first five columns in Table III.6 give details about the test problems. These details include : designation, number of rows (m), number of columns (n), density (d) and the optimal value (z_{opt}). Lower bound information is given in columns (vi) to (ix). Columns (vi)

TABLE III.6

Comparison between the state space relaxation SSR1 and a combination of this with lagrangean relaxation

PROBLEM					SSR1		LSCP1/SSR1	
	m	n	d	z_{opt}	$z_{\mathcal{L}}$	t	$z_{\mathcal{L}}$	t
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)
P10.1	10	99	34.4	2	2.0*	.056	2.0*	.054
P10.2	10	98	40.0	12	12.0*	.092	12.0*	.092
P10.3	10	98	36.2	18	15.96	1.343	8.0	1.209
P10.4	10	100	33.7	17	17.0*	.110	17.0*	.114
P10.5	10	100	37.3	4	4.0*	.080	4.0*	.081
P20.1	20	198	17.3	40	35.24	8.777	33.14	3.630
P20.2	20	197	17.7	31	31.0*	.373	31.0*	.367
P20.3	20	191	17.6	22	22.0*	.222	22.0*	.228
P20.4	20	195	18.2	45	39.94	10.440	39.72	2.725
P20.5	20	198	17.6	40	38.55	9.454	38.69	2.336
P30.1	30	291	11.9	23	21.00	2.961	22.06	2.993
P30.2	30	289	11.9	52	48.62	49.908	46.02	2.872
P30.3	30	291	11.7	31	31.0*	12.206	31.0*	5.185
P30.4	30	293	11.9	52	46.02	+	51.68	7.041
P30.5	30	297	11.5	66	50.54	17.353	55.67	10.012
P40.1	40	391	8.8	84	66.63	+	82.92	9.396
P40.2	40	389	8.9	54	46.45	+	54.0*	2.030
P40.3	40	392	8.8	56	48.84	13.040	55.21	7.662
P40.4	40	385	8.9	64	57.85	34,530	60.59	8.988
P40.5	40	390	9.0	49	43.22	49.813	47.45	7.506
P50.1	50	487	7.1	61	52.90	+	55.73	13.997
P50.2	50	491	7.1	68	55.43	+	68.0*	4.811
P50.3	50	492	6.9	76	65.48	20.717	74.31	8.308
P50.4	50	486	7.1	71	62.92	9.956	69.35	11.440
P50.5	50	489	7.2	80	68.14	17.415	74.41	10.130

MNF5 Compiler
CDC 6500 seconds

and (vii) refer to procedure 3.14 with initial values for λ_i equal to the dual feasible solution produced by the greedy heuristic described in Chapter 2. Columns (viii) and (ix) correspond to the combined lagrangean and state space relaxation with the same initial values for the multipliers.

A maximum number of 25 iterations and a time limit of approximately 10 seconds (CDC6500;MNF5 compiler) were imposed. The only exception to those limits is for the first two iterations which could take as long as 50 seconds in order to assure that a minimum of two iterations would be performed. Even so, iteration number two couldn't be completed for 5 problems which are labelled with (+) in column (vii) giving the time required to compute the lower bound shown in column (vi). Column (ix) presents the time corresponding to the lower bound values given in column (viii). A label (*) in column (vi) and/or (viii) means that the respective lower bound is optimal.

Apart from problems P10.3,P20.1 and P30.2 the lower bound in column (viii) is always greater than or equal to the value shown in (vi). The larger the size of the problem the greater is the difference between the values in those columns. Furthermore, the time taken to obtain the lower bound z_l using the lagrangean-state space combination is for all cases (except P50.4), at least as good as the corresponding value in column (vii).

Therefore, the combination of lagrangean and state space relaxation proves to be more efficient in both the lower bound value and the time consumed. A more sophisticated "decision rule" than that mentioned in 3.4.4 could lead to even better results.

Chapter 3

3.5 RELAXATION SSR2

3.5.1 Definition

Applying SSR2 to the dynamic programming formulation for the SCP given by (3.11),(3.12) and (3.13) the recursions on the relaxed state space are of the form (see (3.24)-(3.25)-(3.26)) :

$$(3.46) \qquad f_k(q+q^k) = \min \left(f_{k-1}(q+q^k) , f_{k-1}(q)+c_k \right)$$
$$\text{for } k=2,...,n \text{ and } q \epsilon Q_{k-1}$$

where $q^k=\Sigma_{i \epsilon M_k} q_i$ and $f_{k-1}(q+q^k)$ is only considered if $q+q^k \epsilon Q_{k-1}$. The sets Q_k (K=1,2,...,n) are defined recursively by :

$$(3.47) \qquad Q_k = Q_{k-1} \cup \{ q+q^k : q \epsilon Q_{k-1} \} \qquad (k=2,...,n)$$

and the initial values are set as follows :

$$(3.48) \qquad Q_1 = \{ g(\Phi)=0 , g(q^1) \}$$
$$f_1(g(\Phi))=f_1(0)=0$$
$$f_1(g(M_1))=c_1$$

SSR2 corresponds to the "knapsack-type" relaxation of the SCP in integer programming and the procedure described in the next section is a straightforward adaptation of a dynamic programming procedure for the 0-1 knapsack problem presented in *Horowitz and Sahni [111] (Chpt. 5).*

3.5.2 Procedure SSR2

A procedure for obtaining a lower bound to the SCP, based on (3.46)-(3.47) and (3.48), is described next. The flowchart for this procedure is presented in Figure 3.6.

Chapter 3

Procedure 3.15 . *Computing a lower bound to the SCP using SSR2*

input . m - number of rows
n - number of columns
c_k - cost of variable index k (k=1,2,...,n)
q^k - weight of variable index k (k=1,2,...,n)
Q(M) - total weight of the rows ($\Sigma_{i\epsilon M} q_i$)

step 1 . initialisation
k=1
i1=1
i2=2
q(1)=0
f(1)=0
$q(2)=q^1$
$f(2)=c_1$
h(1)=2

step 2 . iteration k
k=k+1
if k greater than n go to 6
otherwise do
i=i1
j=i2
h(k)=j
for j equal i1 to i2 do
mk=j
$qq=q(j)+q^k$
if qq less than Q(M) go to repeat
else go to 3
repeat

step 3 . obtaining Q_k
for j equal i to mk do
$qq=q(j)+q^k$
$ff=f(j)+c_k$
if i≤i2 and f(i)<ff do
q(j)=q(i)
f(j)=f(i)
i=i+1
j=j+1
go to repeat
elseif i≤i2 and f(i)=ff then
qq= max (q(i),qq)
go to repeat

Chapter 3

```
                    endif
                    if qq>q(l-1) then
                            q(l)=qq
                            f(l)=ff
                            l=l+1
                    endif
                    while i≤i2 and q(i)≤q(l-1) do i=i+1
            repeat
step 4 . merge in remaining terms from Q_{k-1}
            while i≤i2 do
                    q(l)=q(i)
                    f(l)=f(i)
                    i=i+1
                    l=l+1
            repeat
step 5 . initialise for Q_{k+1}
            i1=i2+1
            i2=l
            go to 2
step 6 . obtain the solution
            qq=q(l)
            ff=f(l)
            ns=0
            i1=h(n)
            for k equal n-1 to 1 do
                    i2=i1-1
                    i1=h(k)
                    for j equal i1 to i2 do
                            if q(j)≠qq then go to repeat
                            else ns=ns+1
                            S(ns)=k
                            qq=qq-q^k
                            ff=ff-c_k
                            go to repeat
                    repeat
            repeat
```

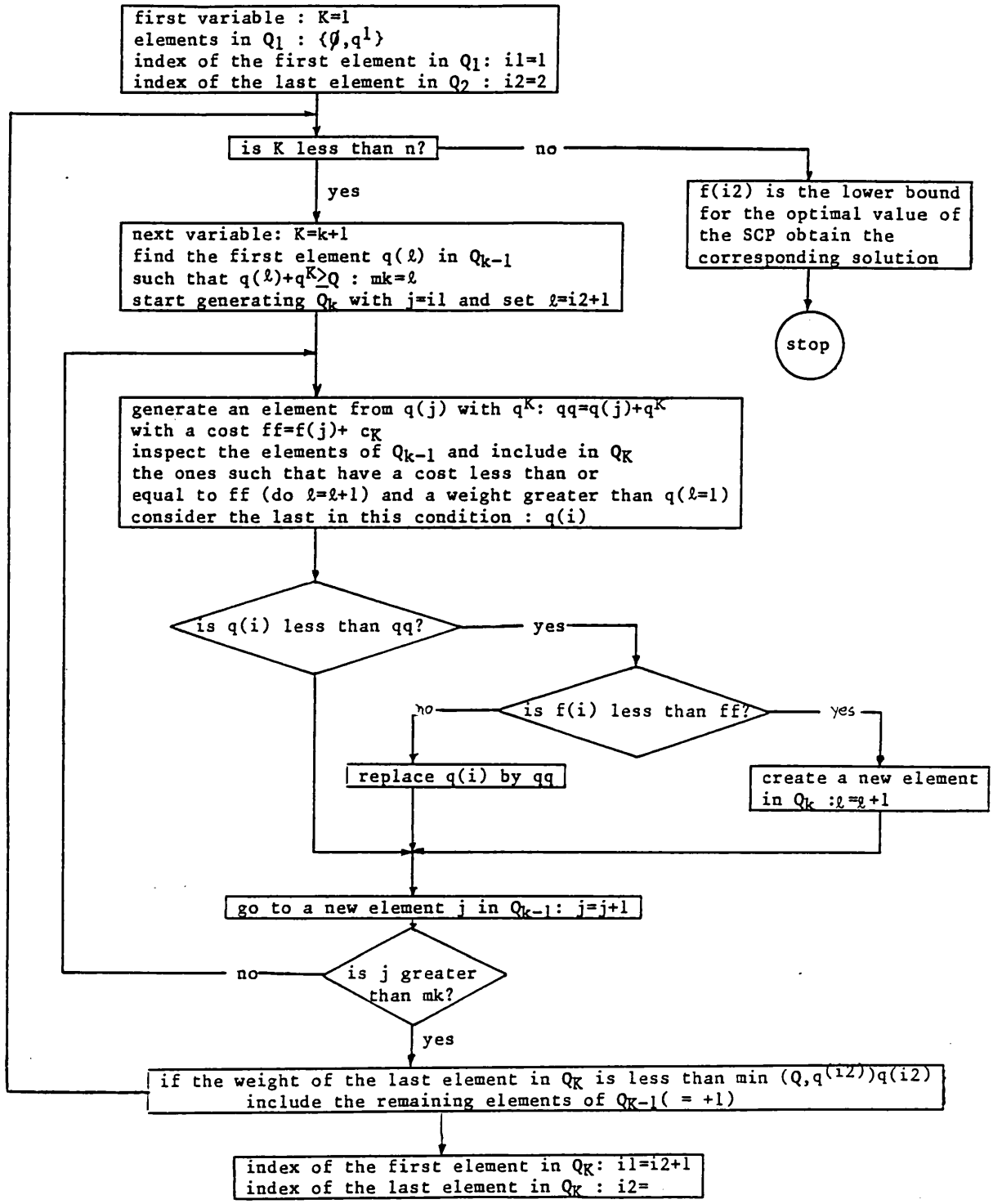


Figure 3.6

Flow diagram for procedure 3.15

Chapter 3

3.5.3 Improving the Lower Bound

Applying (3.27) to SSR2, a lower bound for the value of the objective function when the variable x_k is fixed equal to 1 is given by :

(3.49) $\quad t_k = c_k + f(Q-\Sigma_{i\epsilon Mk}q_i)$

The variable x_k is then removed if $t_k \geq z_u$, with z_u a known upper bound. From property 3.12-(a), if there exists a row i such that :

(3.50) $\quad d_i = \min_{k\epsilon N} (t_k-f(Q)) > 0$

then the lower bound can be improved to $f(Q)+d_i$. A further attempt to improve this value can be made using the result (b) of property 3.12. That is, if there exist two rows, say i and l, such that :

(3.51) $\quad d_{i,l} = \min_{(j,k)\epsilon N_{i,l}} (c_j+\delta_{jk}c_k+f(Q-g(M_j \cup M_k))-f(Q)) > 0$

then the lower bound can be improved to the value $f(Q)+d_{i,l}$. The set $N_{i,l}$ is the product set of N_i and N_l, i.e. $N_{i,l}=\{(j,k): j\epsilon N_i, k\epsilon N_l\}$, and the coefficient δ_{jk} is equal to 1 if $k\neq j$, otherwise is 0.

This last result is illustrated in Figure 3.7 for the example used in Chapter 2, (*Lemke et al. [126]*). Iterations are indicated on the horizontal and correspond to different values of q_i ($i=1,2,...,m$) chosen in a manner that will be discussed in the next section. The value of f(Q) are represented on the vertical. The points joined by a dotted line are the lower bounds computed by procedure 3.15 while the points connected by a plain line correspond to the lower bound after test (3.51). As can be seen this test is very useful for the first iteration and gives an improvement on the bound until iteration number 5 (inclusive). Finally, at iteration 18 the value of z_l becomes, after (3.51), equal to the optimal value for the SCP.

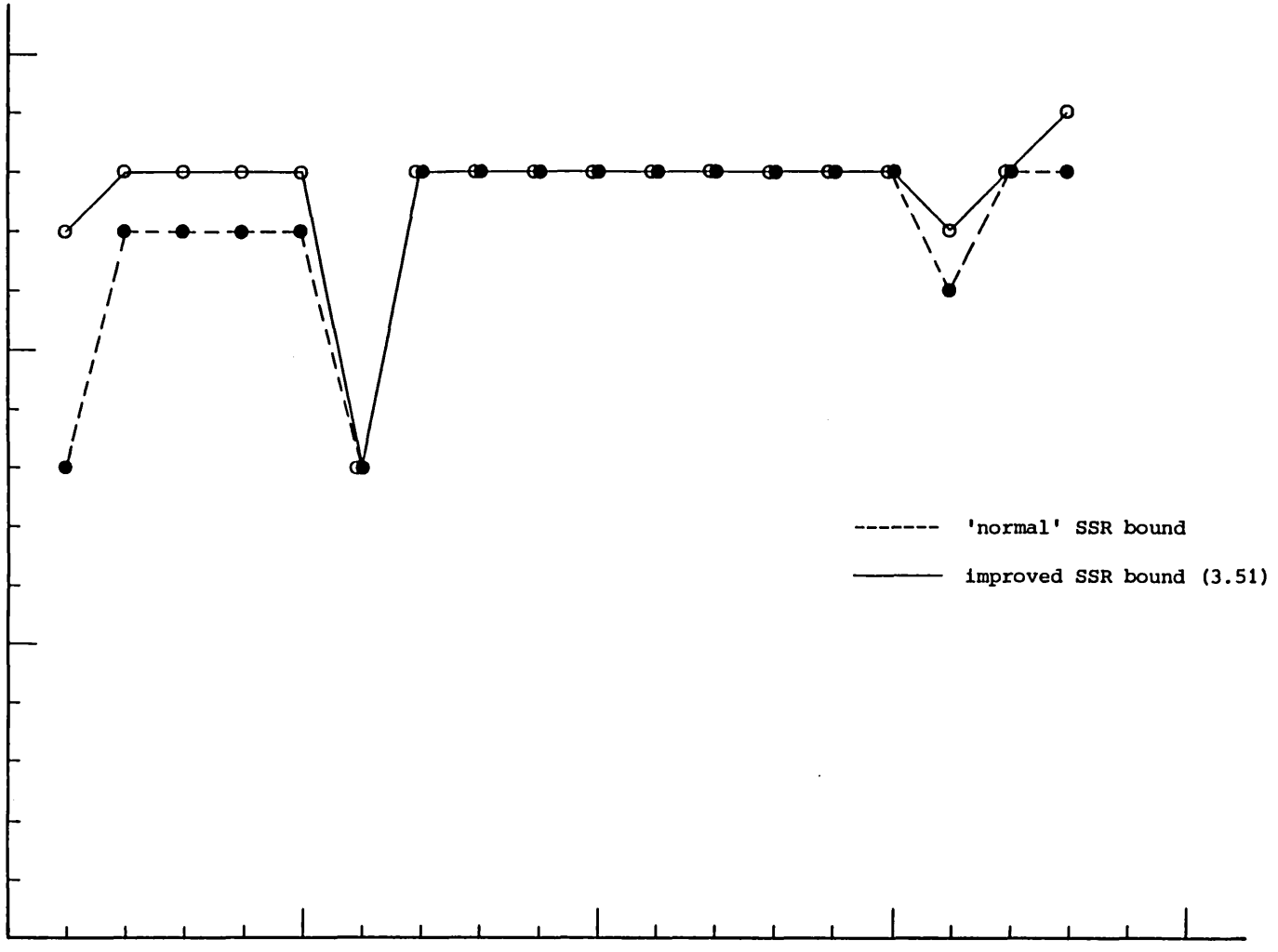


Figure 3.7

Improvement on the lower bound from (3.51) for the SCP of example 2.14

Chapter 3

3.5.4 State Space Modifications

The example considered in the previous section and represented in Figure 3.7 also shows how modifications on the values q_i ($i \epsilon M$) may yield improvements to the lower bound. In fact, the best choice for the q_i would be such that the original state space is mapped onto the relaxed space as uniformly as possible but not exceeding the maximum dimensionality, let us say $\tilde{Q}$ Denoting by $f_q(Q)$ the lower bound produced by SSR2 for a particular vector $q=(q_i)$, the new problem to be dealt with is :

$$\text{(DSSR2)} \qquad \max_{q \geq 0} f_q(Q)$$
$$\text{st.} \qquad Q = \Sigma_{i \epsilon M} q_i \leq \tilde{Q}$$

Since SSR2 is equivalent to the surrogate constraint relaxation in integer programming, the related techniques can be used for solving DSSR2. A subgradient optimization type method presented in *Dyer [61]* to solve surrogate dual problems was adapted for the DSSR2. This method is compared with two other procedures which are based on the simple idea of increasing the weight for non-covered rows and reducing it for the overcovered ones.

From the general surrogate duality theory a good choice for the initial values for the weights q_i would be the optimal dual variables for the LP relaxation of the SCP.However, solving the linear program may increase the computing time and storage requirements with no special benefits, and an alternative which is adopted consists of using a dual feasible solution given by the greedy heuristic described in Chapter 2. Since only integer values are considered for the weights, the initial values are given by :

$$(3.52) \qquad q_i = \lfloor \gamma^* u_i \rfloor \qquad (i \epsilon M)$$

where $\lfloor a \rfloor$ means the maximum integer value less than or equal to a; γ is a positive factor dependent on the values u_i ($i \epsilon M$) and used to approximately maintain the proportionality betwween the dual values. A final remark about DSSR2 concerns the constraint $\Sigma_{i \epsilon M} q_i \leq \tilde{Q}$ When computing new values of q_i there is the possiblity of violating that constraint. A straightforward way to overcome this would be to divide all the q_i by the same factor, say d.

But, then, some of the q_i/d values will be non-integer and $\lfloor q_i/d \rfloor$ must be used. Another way, which proved to be better over several tests, is to share the difference $\Sigma_{i\in M}q_i-\tilde{Q}*(1-\alpha)$, $0\leq\alpha\leq0.5$, among the values q_i as follows :

$$(3.53) \qquad q_i = q_i - [(\Sigma_{i\in M}q_i-(1-\alpha)\tilde{Q})*(q_i/\tilde{Q})]$$

Different values of α were tested and the value $\alpha=0.25$ was chosen.

3.5.5 Modification of the weights q_i

A. Procedure A

Let $x=(x_j)$ be an optimal solution to the relaxed dynamic program. If x is feasible for the original SCP then the problem has been solved; if not a straightforward way to perform state space modifications is to increase the value of the q_i corresponding to uncovered rows and, at the same time, decrease the weight for the overcovered ones, ie. :

$$(3.54) \qquad q_i = \max (0 , q_i+(1-\Sigma_{j\in N_i}x_j)*h)$$

where h is an integer step length. The choice $h=1$ produced good results for medium size problems.

B. Dual type Procedure (Procedure B)

Let us consider again the optimal solution $x=(x_j)$ to the relaxed dynamic program and let i be the index of a row not covered by x. Our aim is to increase the value q_i by an amount $\Delta_i\geq0$ such that :

$$(3.55) \qquad f(Q+\Delta_i) \geq \min_{j\in N_i} t_j$$

where t_j is given by expression (3.49). Hence, a reasonable choice for Δ_i is the minimum integer value satisfying :

(3.56) $\quad f(\Delta_i) \geq \min_{j \in Ni}(t_j - f(Q)) = d_i$

with d_i obtained as in (3.50).

If i is the index of a oversatisfied row then the respective weight can be decreased by an amount $\Delta_i \geq 0$ such that :

(3.57) $\quad \min_{j \in Ni} t_j \geq f(Q-\Delta_i)$

and, again, (3.55) truncated is a possible choice for Δ_i as the minimum integer value that satisfies it. Hence, the weights can be updated as follows :

(3.58) $\quad q_i = q_i + f^{-1}(d_i)*(1-\Sigma_{j \in Ni} x_j)$

where

(3.59) $\quad f^{-1}(d_i) = \min \{ q : 0 \leq q \leq Q \text{ and } f(q) \geq d_i \}$

For $d_i = 0$, we consider $f^{-1}(d_i) = 1$.

C. Subgradient Type Procedure (Procedure C)

As mentioned above, SSR2 is equivalent to the surrogate constraint relaxation for the SCP. Hence, the techniques developed for solving the dual surrogate problem may be used as a procedure to perform state space modifications. In particular, a subgradient type procedure presented in *Dyer [61]* was tested imposing additional restrictions relative to the integrality of the weights q_i and the maximum available dimension $\tilde{Q}$.

Chapter 3

Before describing the procedure let us point out that as has been shown (*Karwan and Rardin [120]*) the subgradient methods used in many lagrangean approaches are inappropriate in the surrogate case. In fact, the objective function for the dual surrogate problem is, in general, not suitable for those techniques because of the possiblity of having no subgradients at some points. The concept of "quasi-subgradients" presented in *Greenberg and Pierskalla [96]* was used by *Dyer [61]* to construct a "quasi-subgradient" method which we adapted in the following way :

(3.60)

(i) normalize the weights : $q_i = q_i / \Sigma_{i \in M} q_i^2$

(ii) compute for the lower bound solution
$$r_i = 1 - \Sigma_{j \in N_i} x_j$$
$$d_i' = r_i - (\Sigma_{i \in M} q_i r_i) * q_i$$
$$d_i = d_i' / \Sigma_{i \in M} (d_i')^2$$

(iii) update the weigths
$$q_i' = q_i + \tau * d_i$$
$$q_i = [q_i']$$

(iv) impose the additional restrictions
$$q_i = q_i - (Q - (1-\alpha)\tilde{Q}) / Q)$$
$$q_i = [q_i]$$

The parameter τ is the step size at each iteration and is initially set equal to 2.0, being halved whenever the procedure fails to improve the lower bound after a number of consecutive iterations.

3.5.6 Example

Let us consider again the example from *Lemke et al. [126]*, in the situation immediately after using the greedy heuristic procedure. Then, the problem has been reduced to 23 effective variables and the bounds are : $z_l = 11.4$ and $z_u = 15.0$. The dual feasible variables obtained from the greedy heuristic are used for setting the initial weights for SSR2 by using (3.52) with $\gamma = 10$:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
q_i	9	9	9	9	10	9	10	9	1	9	9	1	9	1	9

Chapter 3

$Q=\Sigma_{i=1}^{15} q_i=113$ and the values c_j and q^j for the variables are :

j	1	2	3	4	5	6	7	8	9	10	11	14	15
c_j	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0	2.0
q^j	9	9	9	9	10	9	10	10	18	18	19	10	18

j	16	18	19	22	23	24	25	26	30	32
c_j	3.0	3.0	3.0	4.0	4.0	4.0	5.0	5.0	7.0	9.0
q^j	11	19	29	20	38	46	46	29	56	90

Applying the procedure SSR2, the final dynamic programming tableau is :

q	10	20	30	39	49	58	67	76	85	94	104	113
f	1	2	3	4	5	6	7	8	9	10	11	12

The lower bound is then $f(113)=12.0$ corresponding to the solution $x_1=x_2=x_3=x_5=x_7=x_8=x_{11}=x_{23}=1$ and the values t_j for the variables are :

j	1	2	3	4	5	6	7	8	9	10	11	14	15
t_j	12	12	12	12	12	12	12	12	12	12	12	13	13

j	16	18	19	22	23	24	25	26	30	32
t_j	14	13	12	13	12	12	12	14	13	12

All d_i ($i=1,\ldots,15$), computed by using (3.50), are equal to 0 and then no improvement on the bound is possible. The weights of the rows are updated using procedure B :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
q_i	9	8	9	10	8	10	10	8	1	10	9	2	9	1	10

For this iteration of the dynamic programming procedure, the bound obtained is $f(\Sigma_{i=1}^{15} q_i)=f(114)=12.0$. Again, no improvement on the lower bound is achieved. Then, a new iteration is initiated with the weights modified to :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
q_i	9	8	9	9	9	9	10	8	2	10	9	3	9	2	10

j	1	2	3	4	5	6	7	8	9	10	11	14	15
c_j	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0	2.0
q^j	9	8	9	9	9	9	9	10	19	19	18	11	17

j	16	18	19	22	23	24	25	26	30	32
c_j	3.0	3.0	3.0	4.0	4.0	4.0	5.0	5.0	7.0	9.0
q^j	11	22	31	21	36	48	47	33	56	90

Chapter 3

The final tableau, obtained using the recursion (3.46)-(3.47)-(3.48), is given below :

q	0	10	19	31	41	50	60	69	79	90	100	109	116
f	0	1	2	3	4	5	6	7	8	9	10	11	12

The corresponding t_j for the variables (using (3.49)) are :

j	1	2	3	4	5	6	7	8	9	10	11	14	15
t_j	12	12	12	12	12	12	12	12	12	12	12	13	12

j	16	18	19	22	23	24	25	26	30	32
t_j	14	13	12	14	13	12	12	14	13	12

Let us apply (3.51) for rows 14 and 11. The minimum cost of covering these two rows is :

. $d_{14,11} = \min \{c_k + c_j + f(116 - \Sigma_{i \in Mj \cup Mk} q_i) : j = 16,19,23 ; k = 14,15,18,23,24,32\}$

Since $t_{16}, t_{23}, t_{14}, t_{18} \geq 13.0$, we only need to consider the cases where $j = 19$ and $k = 15,24,32$. Computing the expression above with one of the variables $x_{14}, x_{16}, x_{18}, x_{23}$ variables shows that $d_{14,11} \geq 13.0$. Let us then see the three remaining possible cases :

. $c_{19} + c_{15} + f(116 - (q_1 + q_2 + q_3 + q_{11} + q_{12} + q_{14})) = 5 + f(76) = 13.0$

$c_{19} + c_{24} + f(116 - (q_1 + q_2 + q_3 + q_6 + q_8 + q_{11} + q_{12} + q_{14} + q_{15})) = 8 + f(49) = 13.0$

$c_{19} + c_{32} + f(27) = 15.0$

Hence, the lower bound is increased to 13.0 and also the values t_j can be updated for all the variables by computing :

. $t_j = \min_{k = 16,19,23} [c_j + c_k + f(116 - \Sigma_{i \in Mj \cup Mk} q_i)]$, $j \neq 16,19,23$

The new values are :

j	1	2	3	4	5	6	7	8	9	10	11	14	15
t_j	13	14	13	12	12	12	12	12	13	13	12	13	13

j	16	18	19	22	23	24	25	26	30	32
t_j	14	13	12	16	13	13	14	14	14	15

The variables x_{22} and x_{32} are removed from the problem. Also, a cover can be generated by selecting first the variables with minimum t_j :

. $S = \{4,5,6,7,8,11,19\}$

$R = \{10,11,15\}$

and then completing the cover in the same way as we did for obtaining covers from the lagrangean solution. That is,

. $M^* = \{10\}$

min $(t_{10},t_{25})=13.0$ and since $N_{10}\cap R=N_{25}\cap R=\{10\}$
the variable x_{10} is chosen because it is the cheapest.

. $S=\{4,5,6,7,8,11,19,10\}$
$R=\{11,15\}$

. $M^*=\{\ 15\ \}$
min $(t_9,t_{18},t_{24},t_{26})=13.0$
x_{18} is the selected variable

. $S=\{4,5,6,7,8,11,19,10,18\}$
$R=\Phi$

. a prime cover is obtained from S by removing x_5
$S=\{4,6,7,8,10,11,18,19\}$ with cost $c(S)=14.0$

Thus, the variables x_2,x_{16},x_{25},x_{26} and x_{30} are removed from the problem. The SCP has been reduced in such way that, now, the rows indexed 7,10 and 13 have only one effective variable covering each one of them. Hence, the corresponding variables - x_7,x_{10} and x_{13} - are fixed equal to 1; after this, x_3 and x_5 become redundant and are removed.
The state space procedure is resumed considering only weights for the rows of the reduced problem :

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
q_i	9	8	-	9	-	9	-	8	2	-	9	3	-	2	10

and the values for the effective columns are :

j	1	4	6	8	9	14	15	18	19	23	24
c_j	1.0	1.0	1.0	1.0	2.0	2.0	2.0	3.0	3.0	4.0	5.0
q^j	9	9	9	10	19	11	17	22	22	21	41

The bound obtained from SSR2 is then $f(69)=9.0$ which added to the value of the fixed variables gives $z_l=14.0$, the optimal solution.

3.5.7 Computational Results for SSR2

Table III.7 gives computational results using procedure SSR2 with the three types of state space modifications described above. The same set of test problems described in 3.4.6 was considered with a maximum number of iterations fixed at 25 and a time limit of approximately 10 seconds (CDC6500). The number of rows, number of columns and density

Table III.7

Comparisons between the procedures for modifying the weights of the value in SSR2

PROBLEM		Procedure A				Procedure B				Procedure C			
(i)	z_{opt} (ii)	z_g (iii)	I (iv)	t (v)	t_f (vi)	z_g (vii)	I (viii)	t (ix)	t_f (x)	z_g (xi)	I (xii)	t (xiii)	t_f (xiv)
P10.1	2	2.0*	1	.056	.056	2.0*	1	.056	.056	2.0*	1	.060	.060
P10.2	12	12.0*	1	.099	.099	12.0*	1	.090	.090	12.0*	1	.099	.099
P10.3	18	18.0*	2	.245	.245	18.0*	2	.246	.246	17.0	3	.348	1.761
P10.4	17	17.0*	1	.125	.125	17.0*	1	.120	.120	17.0*	1	.117	.117
P10.5	4	4.0*	1	.081	.081	4.0*	1	.080	.080	4.0*	1	.080	.080
P20.1	40	39.0	18	2.636	3.431	40.0+	23	3.449	3.587	36.0	17	2.802	3.795
P20.2	31	31.0*	1	.355	.355	31.0*	1	.365	.365	31.0*	1	.336	.336
P20.3	22	22.0*	1	.246	.246	22.0*	1	.245	.245	22.0*	1	.255	.255
P20.4	45	45.0+	19	3.908	4.928	45.0+	19	3.959	4.947	43.0	12	3.042	5.921
P20.5	40	39.0	4	.644	2.662	39.0	4	.645	2.697	38.0	3	.546	3.050
P30.1	23	22.0	7	1.533	1.889	22.0	21	3.460	3.843	21.0	21	3.667	4.146
P30.2	52	52.0+	15	3.277	4.916	52.0+	18	3.768	4.655	51.0	7	2.073	5.706
P30.3	31	31.0*	2	.729	.729	31.0*	2	.708	.708	22.0*	4	.859	.859
P30.4	52	49.0	7	3.345	10.395	49.0	7	3.211	10.083	46.0	7	3.500	10.144
P30.5	58	56.0	21	8.252	9.368	56.0	21	8.214	9.331	48.0	1	.905	10.022
P40.1	84	80.0	12	6.753	10.403	82.0	16	8.378	10.220	66.0	2	2.190	10.450
P40.2	54	54.0*	7	2.007	2.007	54.0*	7	2.061	2.068	54.0*	11	3.700	3.700
P40.3	56	56.0+	18	3.247	3.893	55.0	5	1.800	3.878	49.0	11	4.364	7.837
P40.4	64	64.0*	16	6.565	6.565	64.0+	14	6.112	9.888	58.0	5	3.080	10.200
P40.5	49	49.0+	22	9.720	10.037	49.0+	8	5.540	10.030	41.0	1	1.176	10.059
P50.1	61	58.0	11	7.754	10.257	60.0	15	10.268	10.272	52.0	2	2.313	10.630
P50.2	68	67.0	15	5.109	5.975	66.0	8	4.299	6.526	55.0	2	2.116	10.202
P50.3	76	73.0	19	9.925	10.431	74.0	18	9.320	10.160	65.0	4	3.178	10.382
P50.4	71	69.0	16	8.117	10.358	69.0	16	8.163	10.398	61.0	2	1.858	10.093
P50.5	80	78.0	10	5.868	10.137	78.0	10	5.899	10.086	65.0	1	1.389	10.039

MNF5 compiler
CDC 6500

are omitted from Table III.7. There are four entry columns for each state space modification technique. The first column gives the best lower bound $z_l=f(Q)$; the second, the number of iterations needed to compute z_l, with the corresponding computational time (t) in the next column. Finally, the fourth entry column for each method shows the total time (t_f) spent at the end of all iterations. Whenever the state space relaxation procedure managed to identify the optimal solution a star (*) is added to the value of z_l. When the lower bound is equal to the optimal solution but the procedure fails to identify the optimality of the bound we put a plus mark (+) rather than (*).

From Table III.7 it is clear that all the techniques perform very well for the 10 row problems. The quality of the bound decreases with the dimension of the problems. However, for the 50 row problems the best out of the three values is always within 3% of the optimal solution.

The subgradient type method fails to achieve a reasonable lower bound value for most of the problems with more than 20 rows. For procedure (C), it is significant that for the problems with 50 rows the best value is obtained at the first iteration and the method fails to improve it after either 25 iterations or 10 seconds.

The procedure A and the dual type procedure B perform in a very similar way. The latter gives better lower bound values for problems P20.1,P30.1,P50.1 and P50.3, while the other only produces a better bound for P50.2. A combination of these two procedures is presented in the next section and the corresponding lower bound is compared with the ($LSCP_\lambda$/SSR1) and the LP relaxation bounds.

Chapter 3

3.6 SSR AND LP RELAXATION

3.6.1 Improving SSR2

After updating the weights q_i for all the rows, an upper bound to the value $z_l=f(Q)$ can be computed. If that upper bound, say $\bar{z}_p$ is less than or equal to z_l^*, the best lower bound obtained so far, then new weights must be calculated and the iteration need not to be performed. An easy way of computing $\bar{z}_p$ is given by the next procedure.

Procedure 3.16 . *Computes an upper bound on the value f(Q)*

step 1 . initialisation
- $S=\Phi$
- $L=N$
- $Q=\Sigma_{i\in M}q_i$
- $\bar{z}_p=0$

step 2 . choosing a variable
- find j* such that
- $c_{j^*}/q^{j} = \min_{j\in L}\{c_j/q^j\}$
- go to 3

step 3 . updating
- $\bar{z}_p=\bar{z}_p+c_{j^*}$
- $S=S \cup \{ j^* \}$
- $L=L-\{ j^* \}$
- $Q=Q-q^{j^*}$
- if $Q>0$ go to 2
- if $Q=0$ go to 5
- otherwise go to 4

step 4 . removing redundant columns from S
- $Q=-Q$
- for $j\in S$ do
  - if $q^j\leq Q$ then
    - $S=S-\{ j \}$
    - $Q=Q-q^j$
    - $\bar{z}_p=\bar{z}_p-c_j$
  - endif
- repeat
- go to 5

step 5 . upper bound

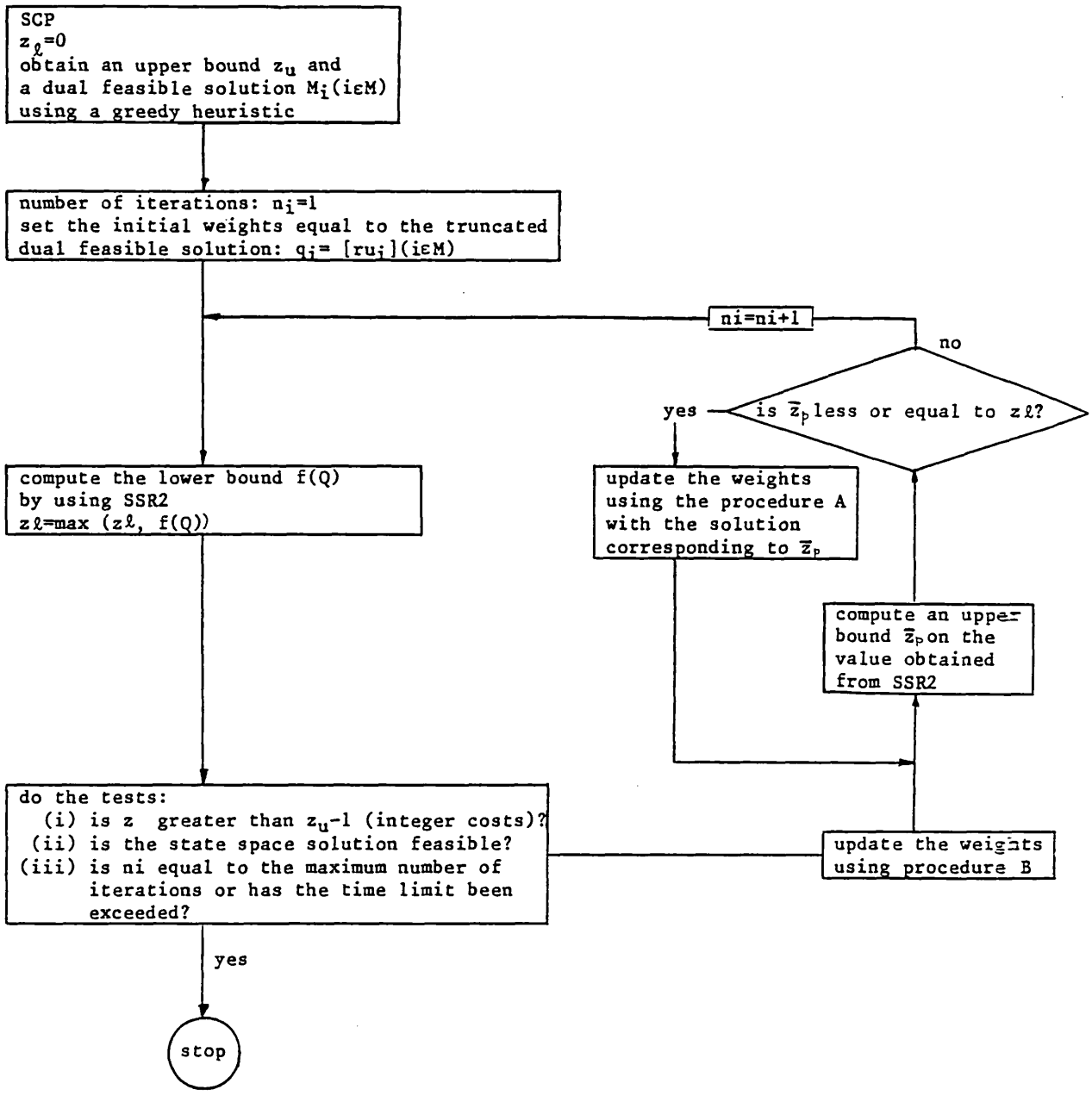


Figure 3.8

Flow diagram for the final version of SSR2

Chapter 3

$\tilde{z}_P$ is an upper bound on f(Q)
S is the index set of the variables corresponding to zz
stop

Using this upper bound a new method to perform the state space modifications to SSR2 was produced combining the procedures A and B. The scheme of this method is shown in Figure 3.8.

3.6.2 Computational Results

Table III.8 compares the perfomances of :

(i) SSR1 (with the lagrangean improvement)

and

(ii) SSR2 (with the upper bound test)

for the same set of test problems which was previously considered.

A maximum number of 25 iterations and a time limit of 10 seconds were imposed again for both state space relaxations. Columns (i),(ii) and (iii) in Table III.8 refer to the problem showing, respectively, the designation of the test problem, the optimal value (z_{opt}) and the greedy upper bound (z_u). The lower bound obtained using SSR1 is shown in column (iv) with the respective computational time given in column (v). The corresponding information for SSR2 is presented in columns (vi) and (vii). Again, a plus mark (+) next to a lower bound value means that this is equal to optimal value but the procedure failed to recognize it. A star (*) means that the optimality of the lower bound was identified.

From Table III.8 it is clear that SSR2 performs, in general, better and faster than SSR1; the only exception is problem P30.4. The gap between the lower bounds is quite large for some test problems such as P20.4, P30.2 and P50.1 for which is about 10% of the optimal value.

For comparing SSR2 and the LP relaxation of the SCP we considered a different set of test

TABLE III.8

Comparison between the final versions of SSR1 and SSR2

PROBLEM			SSR1		SSR2	
(i)	z_{opt} (ii)	z_{μ} (iii)	z_{ℓ} (iv)	time (v)	z_{ℓ} (vi)	time (vii)
P10.1	2	2	2.*	.054	2.*	.060
P10.2	12	12	12.*	.092	12.*	.092
P10.3	18	19	18.*	1.209	18.*	.242
P10.4	17	17	17.*	.114	17.*	.118
P10.5	4	4	4.*	.081	4.*	.081
P20.1	40	45	33.14	2.037	39.	1.493
P20.2	31	31	31.	.367	31.*	.337
P20.3	22	22	22.	.228	22.*	.234
P20.4	45	55	39.72	2.476	45.+	3.574
P20.5	40	44	31.69	2.119	40.*	2.195
P30.1	23	25	22.82	2.993	23.+	3.374
P30.2	52	58	46.07	2.868	52.+	2.893
P30.3	31	31	31.*	5.180	31.*	.774
P30.4	52	61	51.08	7.048	52.*	8.995
P30.5	58	66	55.67	9.788	57.	6.867
P40.1	84	86	82.92	9.391	83.	9.067
P40.2	54	54	54.*	2.030	54.*	1.920
P40.3	56	58	55.21	7.662	56.*	2.021
P40.4	64	66	60.59	8.988	64.*	3.338
P40.5	49	57	47.45	6.818	49.+	5.540
P50.1	61	66	55.27	2.629	61.+	8.772
P50.2	68	61	68.*	4.839	68.*	3.544
P50.3	76	79	74.31	7.618	75	6.478
P50.4	71	77	69.35	11.436	70	9.153
P50.5	80	82	74.41	10.170	79	7.112

CDC 6500
MNF5 compiler

problems. In fact, since our main objective is to use state space relaxation in a general decomposition procedure for large size SCPs, the interval from which the cost of the variables were randomnly generated was reduced to [1,20]. This is consistent with the results shown in the previous Chapter where most of the variables with large costs were removed and the resulting SCP is a harder problem in the sense that much more effort has to be done to produce further reductions. Also, we tried out a set of random diagonal-band problems which are difficult for the LP to solve. Finally, the problems of the classes (III) considered in 2.9 were tried out for SSR2.

Table III.9 reports the computational results obtained with test problems for comparing SSR2 and LP. The first four columns in the table refer to the test problem - (i) identification, (ii) number of rows, (iii) number of columns and density. Columns (v) and (viii) are relative to the bound provided by SSR2 and the respective computing time. The final lower bound and computational time for SSR2 are given in columns (vii) and (viii), respectively. Whenever the state space procedure took longer than the LP, we present in column (v) the best bound produced during the execution of SSR2 until approximately the time taken by the LP. In column (vi) is given the exact time when the first value was produced by SSR2.

Table III.9 is also divided into two parts. The top one refers to problems randomnly generated with no special structure and with integer costs from the interval [1,20]. For these test problems, both SSR2 and LP produce good bounds but the linear programme takes longer for the majority of the cases (the XMP code was used to solve the LP and the time spent in processing the data for this code is not included). For all the cases except P50.8 the bound in column (vii) is better than the one in column (ix), and it is significant that the LP produced an integer solution only in 3 cases. Although the bound can be rounded up (since the costs are integer values) to the nearest integer, still some additional effort is required to identify the optimal value. Moreover, as will be seen in the next Chapter the sub-SCPs produced by the decomposition technique that we used, have no integer costs and for them the rounding up does not apply.

The bottom part of Table III.9 refers to computational results for test problems randomnly generated with a partial band-diagonal structure. That is, the constraint matrix has a

TABLE III.9

Comparison between state space relaxation and linear programming relaxation for the SCP

PROBLEM				SSR2				LP	
	m	n	d	z'_ℓ	t'	z_ℓ	t	z_ℓ	t
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)	(x)
P10.6	10	100	50%			3.0	.139	3.0-	.656
P10.7	10	100	42%			2.0*	.112	2.0*	.553
P10.8	10	100	36%			5.0*	.160	5.0*	.529
P20.6	20	200	44%			4.0*	2.269	4.0*	2.266
P20.7	20	200	40%			5.0*	2.409	4.6	2.163
P20.8	20	200	33%			5.0	1.452	4.5	2.356
P30.6	30	300	39%			4.0	2.747	3.8	6.183
P30.7	30	300	33%			6.0*	5.252	5.3+	5.186
P30.8	30	300	30%			7.0	4.669	6.9	5.601
P40.6	40	400	33%			5.0*	4.260	5.0+	8.137
P40.7	40	400	25%			7.0*	4.779	6.1	8.401
P40.8	40	400	21%			10.0*	13.817	9.4+	16.171
P50.6	50	500	22%			8.0	10.511	7.9	23.181
P50.7	50	500	21%			9.0	15.865	8.6	15.691
P50.8	50	500	21%			9.0	13.173	9.6	15.108
P10.9	10	100	50%			13.0*	.289	12.3+	.852
P10.10	10	100	50%			10.0*	.132	10.0*	.705
P10.11	10	100	27%			22.0*	.112	22.0*	.543
P10.12	10	100	22%			41.0*	.078	34.3	.531
P10.13	10	100	15%			61.0*	.118	47.6	.461
P20.9	20	200	50%			12.0*	1.777	11.0	2.858
P20.10	20	200	23%	55.0	1.100	58.0	6.448	46.3	1.974
P20.11	20	200	20%			16.0*	.603	14.3	1.309
P20.12	20	200	19%			56.0*	.228	56.0	1.696
P20.13	20	200	15%			75.0*	.703	68.5	1.419
P30.9	30	300	50%			21.0	2.483	20.1	9.135
P30.10	30	300	50%	21.0	2.421	22.0*	27.772	19.6	12.696
P30.11	30	300	30%			56.0*	.705	51.3	2.148
P30.12	30	300	20%			72.0	1.382	61.0	2.386
P30.13	30	300	10%			101.0*	.269	100.3+	2.182
P40.9	40	400	25%			14.0	2.044	13.6	11.594
P40.10	40	400	21%	31.0	1.190	32.0	11.693	25.3	5.540
P40.11	40	400	18%	41.0	1.146	43.0	4.370	38.6	3.680
P40.12	40	400	18%			35.0*	1.457	31.6	3.539
P40.13	40	400	17%			70.0*	1.130	63.3	3.530
P50.9	50	500	18%			31.0	2.062	28.3	12.766
P50.10	50	500	17%			35.0	4.594	32.5	4.887
P50.11	50	500	17%			51.0	12.719	46.7	16.400
P50.12	50	500	16%	33.0	3.604	35.0	5.381	33.0	4.851
P50.13	50	500	16%			65.0	1.935	64.4	11.757

CDC 6500 seconds
MNF5 compiler

submatrix with the ones concentrated near the main diagonal. This makes the problems particularly difficult for the LP and, in fact, the results shown in Table III.9 confirm it. For all of the test problems, the bound obtained from SSR2 is better than the LP bound and in many cases the gap is quite significant - P10.12, P10.13, P20.10, P20.13, P30.12, P40.10, P40.13. Also, the computing time for SSR2 is, in general, much less than the corresponding LP time. When the SSR2 procedure took longer than the LP, the bound shown in column (v) is always greater than the LP bound, except for P50.2 where they are equal.

Finally, test problems in classes (II) and (III) considered in Chapter 2 (section 2.9) were also tried using SSR2. Table III.10 shows the corresponding computational results, with columns (i) to (iii) giving information about the test problem. Columns (iv) and (v) in the table show, respectively, the lower and upper bounds obtained from procedure 2.13 (described in Chapter 2) for the test problems. Column (vi) give the state space relaxation bound and the corresponding computational time is presented in column (vii). Columns (viii) and (ix) show the same information relative to the LP.

For the unicost problems, the bound produced by SSR2 is consistently better than the LP bound but the reverse occurs for the test problem T50C2. In this case the state space relaxation bound is even less than the lower bound produced by procedure 2.13. No improvement in this bound is obtained from SSR2 for the test problem T50B1 either. However, the test cases presented the computing time of SSR2 is much less than the corresponding computational time taken by the LP.

Table III.10

Comparison betwen SSR2 and LP relaxation
for the test problems of classes II and III

PROBLEM					SSR2		LP	
(i)	n (ii)	c_j (iii)	z_ℓ (iv)	z_μ (v)	z_ℓ (vi)	t (vii)	z_ℓ (viii)	t (ix)
T50A1	500	1.0	3.14	5.0	4.0	16.198	3.30	50.823
T50A2	500	1.0	3.27	5.0	4.0	18.823	3.33	49.577
T50A3	500	1.0	3.25	5.0	4.0	18.991	3.38	51.847
T50B1	500	$\#M_j$	50.0	64.0	50.0	1.334	-	>150.0
T50C2	500	$3+\#M_j$	58.66	103.0	57.0	8.683	60.2	86.926

CDC 6500 seconds
MNF5 compiler

Chapter 3

3.7 CONCLUSIONS

In this Chapter we studied the application of the state space relaxation technique to the SCP. Some results, both theoretical and practical, were given showing that :

(i) SSR performs reasonably well for medium and small size SCPs even in the cases where the problems have a particularly hard structure or the costs are equal to 1.0 for all the variables. The exception seems to be the problems with costs proportional to the number of rows covered by the column, which are not easily solved by any other known algorithm.

(ii) apart from the lower bound value other useful information for the problem is obtainable from SSR2. In particular, reduced costs (in the form of a minimum cost for the objective function when a variable is forced in the solution), are produced and can be used either to improve the lower bound, or to remove variables from the problem, or also or generate a new cover for the SCP. The reduced costs are also used in performing state space modifications.

(iii) SSR is a lower bound technique in dynamic programming which is equivalent to lagrangean relaxation in integer programming. Hence, SSR may be applied for other combinatorial optimization problems in particular for special versions of the SCP such as the cardinality constrained SCP (used in facility location), the cyclic constraint SCP (appearing in some personnel scheduling problems), and the dynamic SCP (facility location).

(iv) the quality of the bound obtained from SSR for large unicost SCPs opens the possiblity of developing an algorithm for this type of problems and making use of the additional information provided by the method.

Chapter 4

DECOMPOSITION AND STATE SPACE RELAXATION FOR THE SCP

4.1 INTODUCTION

A method for finding lower bounds to the SCP combining decomposition and state space relaxation is described in this Chapter. Large size problems are decomposed into many smaller sub-problems which are solved or approximated using state space relaxation SSR2. Also, the heuristic procedure described in Chapter 2 is used for finding the initial values of the weights in SSR2 and, at the same time, obtaining a dual feasible solution which can lead to an improvement of the lower bound. Good reduced costs are obtained from the combination of state space relaxation and decomposition making possible further reductions on the number of variables. Subgradient optimization is used to update the decomposed costs in a lagrangean fashion.

The example which we have been using (*Lemke et al. [126]*), is worked out through the Chapter illustrating the procedure. Computational results are presented for the large scale test problems and for the unicost SCPs considered in Chapter 2.

Chapter 4

4.2 DECOMPOSITION OF THE SCP

4.2.1 Definition

The set M of constraints of the SCP is partitioned into r disjoint subsets, $R_1,R_2,...,R_r$. Thus the constraint matrix is given by :

(4.1) $$A = \begin{bmatrix} A_1 \\ A_2 \\ (...) \\ A_r \end{bmatrix}$$

Whenever a^j_l, the jth column of A_l,(l=1,2,...,r), is non-zero a variable y^j_l is defined with a cost d^j_l such that :

(4.2) $$\Sigma_{l \epsilon T_j} d^j_l = c_j$$

where T_j is the index set of the submatrices A_l for which variable x_j generates a decomposed variable y^j_t

Example 4.1
Let us apply the decomposition process to the example taken from *Lemke et al. [126]*, which data we rewrite in Tableau IV.1 considering the reductions produced by the greedy heuristic (see Chapter 2). Consider r=3 with A_1,A_2 and A_3 the matrices obtained from the constraint matrix corresponding to, respectively, the first, second and third sets of 5 rows. The sets T_j (j=1,2,...,23) and the variables y^j_t ($t \epsilon T_j$; j=1,2,...,23), are given in Tableau IV.2 where a 1 in the column relative to y^j_t,(t=1,2,3), means that the corresponding variable is defined.

Chapter 4

i		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
j	c_j															
1	1	1														
2	1		1													
3	1			1												
4	1				1											
5	1					1										
6	1						1									
7	1							1								
8	1								1	1						
9	2	1														1
10	2			1							1					
11	2					1								1		
12	2									1		1				
13	2		1									1				
14	3					1									1	
15	3											1	1			1
16	3	1	1	1									1		1	
17	4					1							1			
18	4		1			1			1			1			1	
19	5			1			1		1			1	1			1
20	5	1			1			1			1			1		
21	5			1			1			1			1			1
22	8		1	1			1	1		1		1		1		
23	9	1	1	1	1		1		1		1	1		1		1

Tableau IV.1

Input data for the SCP of example 4.1 (*Lemke et al. [126]*)

A possible choice for d_t^j is :

(4.3) $$d_t^j = (\#[M_j \cap R_t] * c_j)/\#M_j$$

which, applied to the example, yields the decomposed costs shown also in Tableau IV.2 .

Chapter 4

j	T_j	y_1^j	y_2^j	y_3^j	d_1^j	d_2^j	d_3^j
1	{1}	1			1		
2	{1}	1			1		
3	{1}	1			1		
4	{1}	1			1		
5	{1}	1			1		
6	{2}		1			1	
7	{2}		1			1	
8	{2}		1			1	
9	{1,3}	1		1	1		1
10	{1,2}	1	1		1	1	
11	{1,3}	1		1	1		1
12	{2,3}		1	1		1	1
13	{1,3}	1		1	1		1
14	{2,3}		1	1		3/2	3/2
15	{3}			1			3
16	{1,3}	1		1	9/5		6/5
17	{1,3}	1		1	8/3		4/3
18	{1,2,3}	1	1	1	8/5	4/5	8/5
19	{1,2,3}	1	1	1	5/6	5/3	5/2
20	{1,2,3}	1	1	1	2	2	1
21	{1,2,3}	1	1	1	1	2	2
22	{1,2,3}	1	1	1	2	3	3
23	{1,2,3}	1	1	1	18/5	27/10	27/10

Tableau IV.2

'Decomposed' variables and costs for the SCP in example 4.1

Finally, we denote by A_i' the set of indices $j \epsilon A_i$ for which the decomposed variable y_i^j is defined. For the example being considered :

$$A_1' = \{1,2,3,4,5,9,10,11,13,14,16,17,18,19,20,21,22,23\}$$
$$A_2' = \{6,7,8,10,12,18,19,20,21,22,23\}$$
$$A_3' = \{9,11,12,13,14,15,16,17,18,19,20,21,22,23\}$$

The general SCP can then be reformulated as the problem :

Chapter 4

$$(SCP_{(d)}) \qquad \min \ \Sigma_{j\epsilon N}\Sigma_{l\epsilon T_j} d_l^j y_l^j$$

$$\text{st.} \ \Sigma_{j\epsilon A_1'} y_1^j \geq 1 \qquad (i\epsilon R_1)$$

$$\Sigma_{j\epsilon A_2'} y_2^j \geq 1 \qquad (i\epsilon R_2)$$

(4.4) $\qquad (...)$

$$\Sigma_{j\epsilon A_r'} y_r^j \geq 1 \qquad (i\epsilon R_r)$$

(4.5) $\qquad y_l^j = (\Sigma_{t\epsilon Tj} y_t^j)/\#T_j \qquad (l\epsilon T_j; j\epsilon N)$

(4.6) $\qquad y_l^j = 0 \ \text{ or } \ 1 \qquad (l\epsilon T_j; j\epsilon N)$

The constraints (4.5) assure that all the decomposed variables y_l^j relative to the same original x_j have the same value in the optimal solution, ie. if one of them is equal to 1 then all the other ones must be equal to 1 too. Relaxing these constraints, (4.5), a new SCP is obtained :

$$(RSCP_{(d,\nu)}) \qquad \min \ \Sigma_{j\epsilon N}\Sigma_{l\epsilon Tj} d_l^j y_l^j + \Sigma_{j\epsilon N}\Sigma_{l\epsilon Tj} \nu_l^j [y_l^j - (\Sigma_{t\epsilon Tj} y_t^j)/\#T_j] =$$

$$= \Sigma_{j\epsilon N}\Sigma_{l\epsilon Tj} [d_l^j + \nu_l^j - \Sigma_{t\epsilon Tj} \nu_t^j/\#T_j] \ y_l^j$$

$$\text{st.} \ \Sigma_{j\epsilon A_i'} y_l^j \geq 1 \qquad (i\epsilon R_i; l=1,2,...,r)$$

$$y_l^j = 0 \text{ or } 1 \qquad (l\epsilon T_j; j\epsilon N)$$

Now, problem $RSCP_{(d,\nu)}$ is separable in r subproblems $RSCP_{(d,\nu,l)}, (l=1,2,...,r)$, which are also set covering problems :

$$(RSCP_{(d,\nu,l)}) \qquad \min \ \Sigma_{j\epsilon A_l'} [d_l^j + \nu_l^j + (\Sigma_{t\epsilon Tj} \nu_t^j)/\#T_j] y_l^j = \Sigma_{j\epsilon A_l'} \bar{d}_l^j y_l^j$$

$$\text{st.} \ \Sigma_{j\epsilon A_l'} y_l^j \geq 1 \qquad (i\epsilon R_l)$$

$$y_l^j = 0 \text{ or } 1 \qquad (j\epsilon A_l')$$

where

Chapter 4

(4.7) $\bar{d}^j_{\lambda} = d^j_{\lambda} + \nu^j_{\lambda} - (\Sigma_{t\epsilon T_j} \nu^j_t)/\#T_j \qquad (j\epsilon A^{\lambda}_{\beta}; \beta = 1,2,\ldots,r)$.

Taking into consideration the results on lagrangean relaxation mentioned in the first two Chapters, it comes from above that the following relation is valid :

(4.8) $v(SCP) \geq v(RSCP_{(d,\nu)}) = \Sigma^r_{\beta=1} v(RSCP_{(d,\nu,\beta)})$

and naturally :

(4.9) $v(SCP) \geq \max_{\nu} \Sigma^r_{\beta=1} v(RSCP_{(d,\nu,\beta)})$

Hence, a lower bound to the SCP is obtained solving the r subproblems $(RSCP_{(d,\nu,\beta)})$ and subgradient optimization can be used to update the decomposed costs in order to improve that value.

4.2.2 Initial Values

If λ_i, $(i\epsilon M)$, are the optimal lagrangean multipliers for the lagrangean relaxation of the SCP defined in Chapter 2 as $LSCP_{\lambda}$, then the initial values for d^j_{λ} $(j\epsilon N; \lambda\epsilon T_j)$, can be set as follows:

(4.10) $d^j_{\lambda} = c_j/\#T_j$

and

(4.11) $\nu^j_{\lambda} = \Sigma_{i\epsilon R_{\lambda} \cap M_j} \lambda_i$

These initial values guarantee that the bound obtained from the relaxation $RSCP_{(d,\nu)}$ is at least as good as the one produced by $LSCP_{\lambda}$ and, therefore, the LP bound. This is stated by the following property :

Chapter 4

Property 4.1 : $v(RSCP_{(d,\nu)}) \geq \max_{\lambda \geq 0} v(LSCP_\lambda) = v(LP)$

Proof : let us consider d_l^j and ν_l^j ($l \in T_j$; $j \in N$), given by (4.10) and (4.11) with λ_i, ($i \in M$), the optimal lagrangean multipliers to $LSCP_\lambda$. Then, the coefficient $\bar{d}_l^j$ for the variable y_l^j in the objective function of $RSCP_{(d,\nu)}$ is :

(4.12) $$\bar{d}_l^j = d_l^j + \nu_l^j - (\Sigma_{t \in Tj} \nu_t^j)/\#T_j =$$
$$= c_j/\#T_j + \Sigma_{i \in Rl \cap Mj} \lambda_i - (\Sigma_{t \in Tj} \Sigma_{i \in Rt \cap Mj} \lambda_i)/\#T_j$$

The condition (4.2) is satisfied :

$$\Sigma_{l \in Tj} \bar{d}_l^j = \Sigma_{l \in Tj} c_j/\#T_j +$$
$$+ \Sigma_{l \in Tj} \Sigma_{i \in Rl \cap Mj} \lambda_i - \#T_j * [\Sigma_{t \in Tj} \Sigma_{i \in Rt \cap Mj} \lambda_i/\#T_j] = c_j$$

From definition :

(4.13) $$\cup_{l \in Tj} (R_l \cap M_j) = M_j$$

and

$$(R_l \cap M_j) \cap (R_t \cap M_j) = \Phi \text{ for } l \neq t$$

yielding :

(4.14) $$\Sigma_{t \in Tj} \Sigma_{i \in Rt \cap Mj} \lambda_i = \Sigma_{i \in Mj} \lambda_i$$

Hence, (4.12) takes the following expression :

(4.15) $$\bar{d}_l^j = (c_j - \Sigma_{i \in Mj} \lambda_i)/\#T_j + \Sigma_{i \in Rl \cap Mj} \lambda_i$$

and the objective function of $RSCP_{(d,\nu)}$ becomes :

(4.16) $$\Sigma_{j \in N} \Sigma_{l \in Tj} [(c_j - \Sigma_{i \in Mj} \lambda_i)/\#T_j + \Sigma_{i \in Rl \cap Mj} \lambda_i] y_l^j$$

Now, if we relax the constraints of $RSCP_{(d,\nu)}$ and associate with each row i the same multiplier λ_i, the following problem is obtained :

(P) min $\Sigma_{j \in N} \Sigma_{l \in Tj} [(c_j - \Sigma_{i \in Mj} \lambda_i)/\#T_j + \Sigma_{i \in Rl \cap Mj} \lambda_i] y_l^j +$
$+ \Sigma_{l=1}^{r} \Sigma_{i \in Rl} \lambda_i (1 - \Sigma_{j \in Ni} y_l^j)$

st. $y_l^j = 0$ or 1 $\quad (j \in N; l \in T_j)$

Rewritting the objective function of P we obtain :

$$\min \quad \Sigma_{j\in N}\Sigma_{k\in Tj}[(c_j-\Sigma_{i\in Mj}\lambda_i)/\#T_j]y_k^j + \Sigma_{i\in M}\lambda_i$$

$$\text{st.} \quad y_k^j=0 \text{ or } 1 \qquad (j\in N; k\in T_j)$$

which optimal solution is given by :

$$y_k^j = \begin{cases} 1 & \text{if } c_j-\Sigma_{i\in Mj}\lambda_i \\ 0 & \text{otherwise} \end{cases} \qquad (k\in T_j)$$

with the same value as the optimal value of $LSCP_\lambda$.

Therefore, $v(RSCP_{(d,\nu)}) \geq \max_{\lambda\geq 0} v(LSCP_\lambda) = v(LP)$

4.2.3 Updating the Costs

Subgradient optimization can be used to update the decomposed costs for $RSCP_{(d,\nu)}$. A solution y_k^j is obtained for each subproblem $RSCP_{(d,\nu,l)}$ and feasibility to $RSCP_{(d,\nu)}$ is tested by checking the constraints (4.5). Then, the multipliers $\nu=(\nu_i)$ $(i\in M)$, are updated as follows :

$$(4.17) \qquad \nu_k^j = \nu_k^j + \alpha(z_u-z_l)*(y_k^j-\Sigma_{t\in Tj}y_t^j/\#T_j) \Big/ [\Sigma_{h\in Tj}(y_h^j - \Sigma_{t\in Tj}y_t^j/\#T_j)^2]$$

which can be simplified to :

$$(4.18) \qquad \nu_k^j = \nu_k^j + \alpha*(z_u-z_l)*(\#T_j*y_k^j-p_j) \;/\; p_j(\#T_j-p_j)$$

where $p_j=\Sigma_{h\in Tj}y_h^j$

The iterative process stops when either a feasible solution to $SCP_{(d)}$ is obtained, the bound $v(RSCP_{(d,\nu)})>z_u-1$ (if the costs are integer), or the bound has not increased for several iterations.

4.2.4 Reduced Costs

Let x_j be a variable in the original SCP and y_k^j, $k\in T_j$, the corresponding variables in subproblems $(RSCP_{(d,\nu,l)})$, $l=1,2,...,r$. A subadditive function $f_l(.)$ can be defined in each

subproblem for all $j\epsilon A'_\ell$ by :

$$(4.19)\qquad f_\ell(a^j_\ell) = \min \Sigma_{k\epsilon A'_\ell} \bar{d}^k_\ell y^k_\ell$$
$$\text{st. } \Sigma_{k\epsilon A'_\ell} a_{ik} y^k_\ell \geq a_{ij} \qquad (i\epsilon R_\ell)$$
$$y^k_\ell = 0 \text{ or } 1 \qquad (k\epsilon A'_\ell)$$

It is clear that $d^j_\ell - f_\ell(a^j_\ell)$ is the reduced cost of variable y^j_ℓ in the subproblem $RSCP_{(d,\nu,\ell)}$ and, from the subadditivity of f_ℓ, an upper bound for covering the column a^j of the original SCP is given by $\Sigma^r_{\ell=1} f_\ell(a^j_\ell)$. Therefore, the reduced cost of the jth column of the total SCP is given by :

$$(4.20)\qquad s_j = \max\ [0, c_j - \Sigma^r_{\ell=1} f_\ell(a^j_\ell)]$$

and x_j can be fixed equal to 0 if :

$$(4.21)\qquad s_j > z_u - \Sigma^r_{\ell=1} v(RSCP_{(d,\nu,\ell)})$$

Obtaining $f_\ell(.)$ from (4.19) implies solving a set covering problem for all y^j_ℓ, $j\epsilon A'_\ell$, which is impractical even for small size problems. Hence, an upper bound $f'_\ell(.)$ is used instead of $f_\ell(.)$, leading to a lower bound on the reduced cost s_j given by :

$$(4.22)\qquad s'_j = \max\ [0, c_j - \Sigma^r_{\ell=1} f'_\ell(a^j_\ell)]$$

Now, if $u_{i,\ell}$ is a dual feasible solution to the subproblem $RSCP_{(d,\nu,\ell)}$, obtained by using one of the heuristics described in Chapter 2, then :

$$(4.23)\quad s_j = \sum_{\ell=1}^{r} (d^j_\ell - \sum_{i\epsilon M_j \cap R_\ell} u_{i,\ell})$$

is a reduced cost associated with the lower bound which corresponds to a dual feasible solution for the total problem $RSCP_{(d,\nu)}$

In fact, setting $u_i = u_{i,\ell}$ $(i\epsilon R_\ell)$, for the total problem $RSCP_{(d,\nu)}$, the dual feasibility of $u=(u_1,\ldots,u_m)$ is an immediate consequence of (4.2). Thus,

(4.24) $\Sigma_{i\epsilon Mj} u_i = \Sigma_{l=1}^{r} \Sigma_{i\epsilon R_l^j \cap Mj} u_{i,l} \leq \Sigma_{l=1}^{r} \bar{d}_l^j$

and, therefore :

(4.25) $\Sigma_{i\epsilon Mj} u_i \leq \Sigma_{l=1}^{r} [d_l^j + \nu_l^j - (\Sigma_{t\epsilon Tj} \nu_t^j)/\#T_j] = \Sigma_{l=1}^{r} d_l^j = c_j$

Another way of obtaining a lower bound on the reduced costs for the variables in the total SCP is provided by state space relaxation as it will be seen in the next section.

4.2.5 Generating a new Cover

If $y(l)$, $l=1,...,r$, are the optimal solutions for respectively the subprolblems $RSCP_{(d,\nu,l)}$, $l=1,...,r$, then a cover S can be generated for the total SCP by setting :

(4.26) $S=\{j\epsilon N : \max_{l\epsilon Tj} y_l^j\}$

The set S is then reduced to a prime cover by sequentially removing the redundant variables, i.e. the $k\epsilon S$ such that $\Sigma_{j\epsilon S} a_{ij} x_j \geq 2$ for $i\epsilon M_k$. These variables are considered by decreasing order of the costs.

4.3 STATE SPACE RELAXATION AND DECOMPOSITION

4.3.1 Introduction

Instead of obtaining the optimal value for each subproblem $RSCP_{(d,\nu,l)}$, a lower bound on that value, $\nabla(RSCP_{(d,\nu,l)})$, can be used with a corresponding lower bound to the original SCP given by $\Sigma_{l=1}^{r} \nabla(RSCP_{(d,\nu,l)})$. Any lower bound technique for the SCP may then be applied but, naturally, the quality of this bound, both in value and computing time, plays an

important role.

As it was seen in Chapter 3, state space relaxation (SSR) performs reasonably well for small size SCPs. Furthermore, the reduced costs provided by SSR for each sbproblem can be used as a reduction test on the number of variables of the total SCP.

Since an upper bound for each problem is needed, a heuristic of the greedy type described in Chapter 2 is used also. Hence, we use a two step procedure for each subproblem $RSCP_{(d,\nu,\mathcal{D})}$:

Procedure 4.2 . *obtaining a lower bound for subproblem* ℓ

Step 1 . an upper bound z_u^{ℓ}, a lower bound z_{lb}^{ℓ} and reduced costs $u_{i,\ell}$ are obtained using the greedy heuristic.
If $z_{lb}^{1}=z_u^{\ell}$ then z_{lb}^{ℓ} is optimal and the subproblem $RSCP_{(d,\nu,\mathcal{D})}$ has been solved.
If not go to 2.

Step 2 . using the values $u_{i,\ell}$ for setting the initial weights of the rows, a fixed number of iterations is performed to the state space relaxation of the dynamic programming formulation for subproblem $RSCP_{(d,\nu,\mathcal{D})}$. A lower bound is given by $f_{\ell}(Q)$ $(Q=\Sigma_{i\epsilon R_{\ell}}q_i)$, computed at each iteration and the value z_{lb}^{ℓ} is updated to $z_{lb}^{\ell}=\max(z_{lb}^{\ell},f_{\ell}(Q))$.

4.3.2 Reduced Costs from SSR

Let us consider a subproblem $RSCP_{(d,\nu,\mathcal{D})}$ for which the state space ralaxation SSR2 is applied. Denoting by $q_{i,\ell}$ $(i\epsilon R_{\ell})$ the corresponding weights, it follows from (3.27) that

$$t_{\ell}^{j} = d_{\ell}^{j} + f_{\ell}(Q_{\ell}-q_{\ell}^{j}) \qquad (4.27)$$

is a lower bound on the value of the ℓth subproblem when the variable x_j is fixed equal to 1. The values Q_{ℓ} and q_{ℓ}^{j} are defined as follows :

Chapter 4

(4.28) $Q_\ell = \Sigma_{i\in R_\ell} q_{i,\ell}$

$q_\ell^j = \Sigma_{i\in M_{j,\ell}} q_{i,\ell} \qquad (M_{j,\ell}=M_j\cap R_\ell)$

Now, consider the whole problem $RSCP_{(d,r)}$ and the following state space mapping function :

(4.29) $g(S) = (\Sigma_{i\in S_1} q_{i,1},\ldots,\Sigma_{i\in S_r} q_{i,r}) \quad , \quad S_\ell = S\cap R_\ell \quad (\ell=1,2,\ldots,r)$

Applying again (3.27), the value

(4.30) $t_j = c_j + f(Q_1-q_1^j,\ldots,Q_r-q_r^j)$

is a lower bound on the value of $RSCP_{(d,r)}$ when the variable x_j is forced to be equal to 1 (f is the recursive function relative to the relaxation (4.29)). Thus,

(4.31) $f(Q_1-q_1^j,\ldots,Q_r-q_r^j)\geq\Sigma_{\ell=1}^r f_\ell(Q_\ell-q_\ell^j)$

and, hence :

(4.32) $c_j + f(Q_1-q_1^j,\ldots,Q_r-q_r^j)\geq c_j + \Sigma_{\ell=1}^r f_\ell(Q_\ell-q_\ell^j)$

$\geq\Sigma_{\ell=1}^r(d_\ell^j+f_\ell(Q_\ell-q_\ell^j))$

where we consider $d_\ell^j=0$ and $q_\ell^j=0$ for $\ell\notin T_j$. Therefore, a lower bound on the value t_j is given by :

(4.33) $\bar{t}_j = \Sigma_{\ell=1}^r(d_\ell^j+f_\ell(Q_\ell-q_\ell^j))$

Note that if several iterations, say nk, are performed for the decomposition, then we can keep the maximum $\bar{t}_j$ out of the nk values obtained to the variable x_j. That is, we take :

(4.34) $t_j^* = \max\ [t_j^*,\Sigma_{\ell=1}^r[d_\ell^j+f_\ell(Q_\ell-q_\ell^j)]]$

with t_j^* initially set equal to 0.

Chapter 4

4.3.3 Generating Covers

Since for some l, the partial solution $y(l)$ considered in 4.2.5 could be not feasible for the subproblem $RSCP_{(d,r,l)}$, the set S obtained by (4.26) may be not a cover for the total SCP. In this case new variables are added to S being chosed by a proccess similar to the greedy heuristic.

A different cover can be also generated making use of the information relative to the reduced cost obtained from SSR and the decomposition. This is done in a way completely similar to the proccess of generating covers from the lagrangean relaxation reduced costs. That is, the variables for which the reduced costs

$$(4.35) \qquad \xi_j = \max\ (0, \bar{t}_j - \Sigma_{l=1}^{r} f_l(Q_l))$$

is equal to 0 enter the set S. If there is a row, say indexed i, such that is not covered, then the variable $j \epsilon N_i$ with minimum $\bar{t}_j$ enters the set S which is then reduced to a prime cover. This is done by removing redundant variables considerd in decreasing order of the original costs.

4.4 DECOMPOSITION PROCEDURE

A general procedure based on decomposition and state space relaxation for the SCP can be described as follows :

Procedure 4.3 . *obtaining a lower bound for the SCP*

Input . z_{lb} (lower bound), z_u (upper bound) and λ_i ($i \epsilon M$)
(lagrangean multipliers corresponding to z_{lb}), obtained
from procedure 2.13.
dsub - dimension for the subproblems
nsub - number of subproblems
nk - number of iterations for the decomposition
ns - number of iterations for the state space relaxation

Step 1 . initialisation

k=1

compute the initial values for the decomposed costs, by using (4.8)-(4.9)

Step 2 . iteration k

for ℓ=1 to nsub

do procedure 4.2

obtaining z_{lb}^{ℓ} (lower bound for lth subproblem)

$X_\ell=\{j\epsilon N: y_j^\ell\}$ (partial solution)

t_j^ℓ ($\ell\epsilon T_j; j\epsilon A^\ell$)

repeat

$z_{lb}=\max(z_{lb},\Sigma_{\ell=1}^{nsub} z_{lb}^{\ell})$

if $z_{lb}>z_u$-1 (integer costs) go to 7

otherwise go to 3

Step 3 . generating a cover

$S = \cup_{\ell=1}^{nsub} X_\ell$

$R = M - \cup_{j\epsilon S} M_j$

if $R\neq\Phi$ complete the cover using a greedy function

reduce S to a prime cover and set $z_u=\min(z_u,\Sigma_{j\epsilon S} c_j)$

Step 4 . computing the costs $\bar{t}_j$

for j=1 to n do

$\bar{t}_j=\Sigma_{\ell\epsilon Tj} t_j^\ell+\Sigma_{\ell\notin Tj} z_{lb}^{\ell}$

if $\bar{t}_j>z_u$-1 remove j from the problem

repeat

try to improve the lower bound by doing :

smax=0.0

for i=1 to m do

$d_i=\min_{j\epsilon N} \bar{t}_j$

if $d_i>$smax then smax=d_i

repeat

$z_{lb}=\max(z_{lb},$smax$)$

if $z_{lb}>z_u$-1 go to 7

Step 5 . obtaining a prime cover from the costs $\bar{t}_j$

$S = \Phi$

for j=1 to n do

if $\bar{t}_j-\Sigma_{\ell=1}^{nsub} z_{lb}^{\ell}$ then $S = S \cup \{j\}$

repeat

for $j\epsilon S$ if $S-\{j\}$ is still a cover then remove j from S

$z_u = \min(z_u,\Sigma_{j\epsilon S} c_j)$

Step 6 . updating the decomposed costs

k=k+1

if k>nk go to 8

otherwise update the lagrangean multipliers using (4.17) and go to 2

if the solution is feasible for (4.4) and the all the partial

solutions are optimal, then $z_u = z_{lb}$ and go to 7

Step 7 . optimal solution

the current upper bound is the optimal value

stop

Step 8 . lower bound value

z_{lb} is the lower bound value obtained from the decomposition

stop

4.5 SORTING THE MATRIX

Nothing has been said so far about the structure of the constraint matrix $A=[a_{ij}]$ of the original SCP. However, if the matrix A is almost in block diagonal form when the decomposition is used, then the number of generated variables y_t^j ($t \epsilon T_j$) is smaller and so it is less likely that constraints (4.5) will be violated. Also, the computing time is likely to be reduced with a more structured data for the problem.

We used a heuristic procedure developed by *Nakornchai [135]* which sorts the matrix into blocks by tries to cluster the non-zero elements near the diagonal. This procedure sorts alternately the columns and rows of the matrix until no further improvement is made. For large-scale SCPs, this procedure proved quite useful by reducing the number of decomposed variables in more then 10% and so enabling to a much faster computation of the lower bound.

Although the example which we have been using (*Lemke et al. [126]*), is not the best one to illustrate the sorting we show next how an improvement on an available lower bound is possible by using the decomposition for the mentioned example.

In Tableau IV.3 we show the matrix after being sorted by the heuristic procedure. As it can be seen, the number of variables y_t^j ($t \epsilon T_j; j \epsilon N$), was reduced from 43 to 38, that is, more than 10% which is significant for a small problem with no structure.

Example 4.4

Let us then consider the example we have been using after being reduced to a 15 row and 23 column SCP. We also consider the sorted matrix as is shown in Tableau IV.3. In order to

Chapter 4

	i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		(3)	(11)	(15)	(2)	(6)	(1)	(7)	(13)	(4)	(10)	(12)	(9)	(8)	(14)	(5)
j	c_j															
23	9	1	1	1	1	1	1		1	1	1			1		
19	5	1	1	1		1						1		1		
22	7	1	1		1	1		1	1				1			
20	5						1	1	1	1	1					
21	5	1		1		1						1	1			
16	3	1			1		1					1			1	
18	4		1		1									1	1	1
15	3		1	1								1				
17	4						1					1				1
8	1												1	1		
9	2			1			1									
10	2	1									1					
11	2								1							1
12	2		1										1			
13	2		1		1											
14	2														1	1
1	1						1									
2	1				1											
3	1	1														
4	1									1						
5	1															1
6	1					1										
7	1							1								

Tableau IV.3

Sorted matrix for the SCP of example 4.1

((i) - previous index of row i)

facilitate the computations we use a dual feasible solution different from the fractional one provided by the greedy heuristic. The dual feasible solution we use to derive the decomposed costs was taken from *Balas and Ho ([11])*, where it is used as a first lower lower bound approximation for the same example. Now, we apply the decomposition with :

. dsub=5
nsub=3
nk=1
ns=1
z_u=15.0
z_{lb}=12.0
u = (0,0,2,0,1,0,1,2,1,1,0,1,0,3,0)

STEP 1 . the initial decomposed costs are computed using (4.12) with (4.8) and (4.9):

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
d_1		1	1			1			2	1/2		1/2	2		5/2	0		1/2	4		7/2	5/3	11/3
d_2	1			1			1		0	3/2	2					0	2			5		11/3	14/3
d_3					1			1			0	3/2		3	1/2	3	2	7/2	1		3/2	5/3	2/3

STEP 2 . iteration 1 for the decomposition

. subproblem 1

Following the procedure strictly, we would use the greedy heuristic to obtain a dual feasible solution for the subproblem and then initialise the weights from that. To avoid too many computations in the example, the weights are set from the dual feasible solution for the total problem :

i	1	2	3	4	5	Q_1
q_{i1}	0	0	2	0	1	3

Then, the data for the SSR of the subproblem is :

j	2	3	6	9	10	12	13	15	16	18	19	21	22	23
d_1	1	1	1	2	1/2	1/2	2	5/2	0	1/2	4	7/2	5/3	11/3
q_1	0	0	1	2	0	0	0	2	0	0	3	3	1	3

Naturally the variable x_{16} is included into the partial solution set X_1 and the rows index 1 and 4 are covered with a cost 0.0 in the subproblem. Therefore, x_2,x_3,x_{10} and x_{16} have a reduced cost equal to the corresponding decomposed

cost in the subproblem and need not to be considered in the SSR procedure. The final tableau for this procedure is :

q	0	1	2	3
f_1	0	1	2	3

which corresponds to the solution $X_1 = \{16,6,9\}$ with a value $z^1_{lb}=3$ and yielding the following values for t^j_1 ($j\epsilon A'_1$) :

j	2	3	6	9	10	12	13	15	16	18	19	21	22	23
t_1	4	4	3	3	7/2	7/2	5	7/2	3	7/2	4	7/2	11/3	11/3

X_1 is not feasible and $d_2=\min(t^{12}_1,t^{13}_1,t^{15}_1,t^{18}_1,t^{19}_1,t^{22}_1,t^{23}_1)=7/2$. Therefore, the lower bound is improved to $z^1_{lb}=7/2$ and the t^j_1 are updated to :

j	2	3	6	9	10	12	13	15	16	18	19	21	22	23
t_1	9/2	9/2	7/2	7/2	4	7/2	5	7/2	7/2	7/2	4	4	11/3	11/3

Note that, for instance, t^{10}_1 is obtained as follows :
$t^{10}_1=\min_{j=12,13,15,18,19,22,23}\ [d^{10}_1+d^j_1+f(Q_1-\Sigma_{i\cap M10\cup Mj}q_i)]$
The solution {6,9,12,16} is optimal with value $z^1_{lb}=7/2$.

. subproblem 2
Again, we set the weights from the available dual feasible solution :

i	6	7	8	9	10	Q_2
q_i	0	1	2	1	1	5

and the input data relative to the SSR for the subproblem is :

j	1	4	7	9	10	11	16	17	20	22	23
d_1	1	1	1	0	3/2	2	0	2	5	11/3	14/3
q_1	0	1	1	0	1	2	0	0	5	3	4

variables x_9 and x_{16} are included in the solution set X_2 and the variables x_1,x_9,x_{16} and x_{17} have reduced costs equal to the respective decomposed costs. Now, solving the dynamic program we get the following final tableau :

Chapter 4

q	0	1	2	3	4	5
f_2	0	1	2	3	4	5

corresponding to the solution $X_2=\{9,16,20\}$ which is feasible, therefore optimal for the subproblem. The values t_2^j ($j\epsilon A_2'$) are :

j	1	4	7	9	10	11	16	17	20	22	23
t_2	6	5	5	5	11/2	5	5	7	5	17/3	17/3

. subproblem 3

The weights for the rows are :

i	11	12	13	14	15	Q_3
q_1	0	1	0	3	0	4

and the subsequent input data for the SSR variables is :

j	5	8	11	12	14	15	16	17	18	19	21	22	23
d_3	1	1	0	3/2	3	1/2	3	2	7/2	1	3/2	5/3	2/3
q_3	0	1	0	1	3	0	3	0	3	0	1	0	0

Initially, $X_3=\{11\}$ and the final dynamic tableau is :

q	0	1	2	3	4
f_3	0	1	5/2	3	4

The partial solution is $X_3=\{11,8,14\}$ which is not feasible. However, changing x_{14} with x_{16} the solution becomes feasible. The t_3^j ($j\epsilon A_3'$) are:

j	5	8	11	12	14	15	16	17	18	19	21	22	23
t_3	5	4	4	9/2	4	9/2	4	6	9/2	5	9/2	17/3	14/3

The lower bound obtained at the end of this step is then $z_{lb}=3.5+5.0+4.0=12.5>12.0$, the previous value.

Chapter 4

STEP 3 . generating a cover from the partial solution

$S = X_1 \cup X_2 \cup X_3 = \{6,9,12,16\} \cup \{9,16,20\} \cup \{8,11,16\} = \{6,8,9,11,12,16,20\}$

which is a cover with no redundant variables and cost $c(S)=16.0$; hence, no better upper bound value has been found.

STEP 4 . computing the values t_j

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
t_1	3.5	4.5	4.5	3.5	3.5	3.5	3.5	3.5	3.5	4.0	3.5	3.5	5.0	3.5
t_2	6.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.5	5.0	5.0	5.0	5.0
t_3	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.5	4.0	4.0
t_j	13.5	13.5	13.5	12.5	13.5	12.5	12.5	12.5	12.5	12.5	13.5	12.5	13.0	14.0	12.5

j	15	16	17	18	19	20	21	22	23
t_1	3.5	3.5	3.5	3.5	4.0	3.5	4.0	11/3	11/3
t_2	5.0	5.0	7.0	5.0	5.0	5.0	5.0	17/3	17/3
t_3	4.5	4.0	6.0	4.5	5.0	4.0	4.5	17/3	14/3
t_j	13.0	12.5	16.5	13.0	14.0	12.5	13.5	15.0	14.0

The variables x_{17} and x_{22} are removed from the problem. Also, since $smax=d_2 = \min(t_{12},t_{13},t_{15},t_{18},t_{19},t_{23}) = 13.0$, the lower bound is further improved to $z_{lb}=13.0$

STEP 5 . obtaining a cover from the values t_j

$S = \{j \in N : t_j \leq z_{lb}\} = \{4,6,7,8,9,11,12,14,15,16,18,20\}$

which generates the prime cover $S = \{6,9,12,14,16,20\}$ with cost $c(S)=16.0$.

STEP 6 . updating the decomposed costs

The constraints (4.5) are not satisfied for x_{11} and x_{12} since $y_2^{11}=0$ and $y_3^{11}=1$, $y_1^{12}=1$ and $y_3^{12}=0$. Then, the respective decomposed costs are updated using (4.18) with $\alpha=1.0$:

$d_2^{11} = 2.0 + (1/2)*(15.0-13.0)*(-1) = 1.0$

$d_3^{11} = 0.0 + (1/2)*(15.0-13.0)*(2-1) = 1.0$

$d_1^{12} = (1/2) + (1/2)*(2.0)*(2-1) = 3/2$

$d_3^{12} = (3/2) + (1/2)*(2.0)*(-1) = 1/2$

The costs for the new iteration of the decomposition procedure are :

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
d_1		1	1		1			2	1/2			3/2	2		5/2	0		1/2	4		7/2	5/3	11/3
d_2	1		1			1		0	3/2	1						0	2			5	11/3	14/3	
d_3				1		1				1	1/2		3	1/2	3	2	7/2	1		3/2	5/3	2/3	

Chapter 4

STEP 2 . solving the subproblems

. subproblem 1

For this subproblem, the SSR provides the lower bound $z_{lb}^1=3.5$ corresponding to the solution set $X_1=\{6,15,16\}$.

. subproblem 2

The lower bound obtained from the SSR is $z_{lb}^2=4.5$ with the solution set $X_2=\{4,7,9,10,11,16\}$.

. subproblem 3

The solution for this subproblem is $X_3 = \{8,14\}$ with the lower bound value $z_{lb}^3=4.0$ (obtained after reduced cost analysis) .

Therefore, the lower bound obtained for the total problem is then $z_{lb}=12.0$.

STEP 3 . generating a cover from the partial solution

$S=\{6,15,16\}\cup\{4,7,9,10,11,16\}\cup\{8,14\} = \{4,6,7,8,9,10,11,14,15,16\}$

which is reduced to a prime cover by removing x_{19} and x_{14} (in this order), resulting $S=\{4,6,7,8,10,11,15,16\}$ with cost $c(S)=14.0$.

Hence, the upper bound has been improved to $z_u=14.0$.

STEP 4 . the values t_j corresponding to this iteration of the decomposition are :

j	1	2	3	4	5	6	7	8	9	10	11	12	13
t_1	3.5	4.5	4.5	3.5	3.5	3.5	3.5	3.5	3.5	4.0	3.5	4.5	5.0
t_2	5.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5
t_3	4.0	4.0	4.0	4.0	5.0	4.0	4.0	4.0	4.0	4.0	5.0	4.0	4.0
t_j	13.0	13.0	13.0	12.0	13.0	12.0	12.0	12.0	12.0	12.5	13.0	13.0	13.5

j	14	15	16	18	19	20	21	23
t_1	3.5	3.5	3.5	3.5	4.0	3.5	4.0	11/3
t_2	4.5	4.5	4.5	4.5	4.5	5.0	4.5	17/3
t_3	4.0	4.5	4.0	4.0	5.0	4.0	5.0	14/2
t_j	12.0	12.5	12.0	12.0	13.5	12.5	13.5	14.0

Applying (4.34) we remove $x_1,x_2,x_3,x_5,x_{10},x_{13},x_{19},x_{21}$ and x_{23} from the problem. The resulting SCP is easily shown to have a lower bound $z_l\geq 15.0$ and therefore the optimal value is $z_u=14.0$ (note that the lower bound exceeds z_u because x_{10} has been removed and then x_{20} is fixed equal to 1, since it is the only effective column for row 10).

Chapter 4

4.6 COMPUTATIONAL RESULTS

The method described in this Chapter was tested for the large size test problems of Chapter 2, considered after the reductions produced by procedure 2.13. Also, unicost SCPs with 50 rows, 500 columns and 20% density were used in testing the decomposition technique.

Table IV.4 shows the value of the lower bound obtained after 10 iterations of the the decomposition procedure for test problem T200X2 (m=199;n=178;d=5.4%) with four different dimensions for the subproblems - 7,10,13 and 17 rows. Limits on the number of iterations and maximum computing time for the subproblems were imposed to assure a total time similar for all the different 'decompositions'. Tables IV.5 and IV.6 give the same information relatively to test problems T200X3 (m=200;n=176;d=5.4%) and T200X5 (m=186;n=124;d=5.3%).

Column (i) of Tables IV.4 to IV.6 refers the iteration number for the decomposition procedure with the corresponding lower bound and accumulated computing time given in columns (ii) to (ix) corresponding to each one of the four different dimensions considered for the subproblems. The computational times are given for the CDC 7600 computer using the FTN compiler.

From the results shown in Tables IV.4 to IV.6, it seems that there is little to choose from the four different subproblem sizes tested. In fact, either the lower bound value or the corresponding computing time do not vary much with the size of the subproblems. Thus, we are going to make an arbitrary decision and choose to decompose the SCP into 10-row subproblems for the large SCPs.

The lower bound increases slowly with the value obtained at the first iteration being one of the best out of the ten values. This suggests the possibility of subgradient optimization not being as efficient in this application as it is for many other relaxations and, at least, other techniques may be tried to compute the multipliers ν. However, the decomposition lower bound was always better than the one obtained after using procedure 2.13. Also, it enabled further reductions in the number of variables of the problem by performing reduced cost analysis with the values t_j (4.32)-(4.34).

This is shown in Table IV.7 where we present the bounds obtained for the test problems TmXk (m=200 and 300;k=1,...,5), when using the decomposition technique after procedure

TABLE IV.4

Comparison of four different dimensions for the subproblems in the decomposition (T200X2)

	7		10		13		17	
(i)	z_ℓ (ii)	t (iii)	z_ℓ (iv)	t (v)	z_ℓ (vi)	t (vii)	z_ℓ (viii)	t (ix)
1	66.38	.155	66.45	.140	66.40	.139	66.40	.224
2	66.37	1.152	66.34	1.087	66.39	1.016	66.37	1.147
3	66.35	1.447	66.33	1.433	66.32	1.364	66.32	1.592
4	66.31	1.817	66.37	1.820	66.34	1.771	66.39	1.990
5	66.31	2.168	66.40	2.184	66.36	2.134	66.36	2.476
6	66.32	2.513	66.43	2.539	66.37	2.448	66.32	2.979
7	66.35	2.900	66.38	2.915	66.41	2.945	66.36	3.518
8	66.37	3.268	66.39	3.313	66.41	3.211	66.31	4.086
9	66.38	3.638	66.39	3.707	66.45	3.583	66.31	4.666
10	66.38	4.003	66.39	4.098	66.58	3.865	66.30	5.306

CDC 7600
FTN compiler

TABLE IV.5

Comparison of four different dimensions for the subproblems in the decomposition (T200X3)

	7		10		13		17	
(i)	z_ℓ (ii)	t (iii)	z_ℓ (iv)	t (v)	z_ℓ (vi)	t (vii)	z_ℓ (viii)	t (ix)
1	90.85	.136	91.03	.132	90.92	.125	90.94	.211
2	90.88	.894	91.07	.872	90.95	.835	90.88	1.056
3	90.87	1.131	91.05	1.103	90.85	1.129	90.92	1.468
4	90.89	1.385	91.07	1.392	90.85	1.371	90.89	1.690
5	90.92	1.644	91.11	1.611	90.85	1.719	90.82	2.024
6	90.94	1.809	91.06	1.876	90.85	2.042	90.91	2.847
7	90.91	2.166	91.07	2.195	90.76	2.414	90.90	2.975
8	90.91	2.424	91.00	2.490	90.81	2.917	90.93	3.387
9	90.83	2.719	91.86	2.841	90.81	3.339	90.97	3.803
10	90.92	2.973	91.84	3.112	90.88	3.710	90.93	4.202

CDC 7600
FTN compiler

TABLE IV.6

Comparison of four different dimensions for the subproblems in the decomposition (T200X5)

	7		10		13		17	
	z_ℓ	t	z_ℓ	t	z_ℓ	t	z_ℓ	t
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)
1	56.60	.112	56.77	.098	56.66	.179	56.22	.283
2	56.60	.543	56.74	.472	56.67	.630	56.13	.934
3	56.57	.755	56.80	.669	56.54	1.055	56.11	1.421
4	56.55	1.971	56.74	.931	56.48	1.547	56.21	1.909
5	56.59	1.201	56.76	1.178	56.57	1.954	56.25	2.407
6	56.61	1.410	56.67	1.566	56.54	2.392	56.29	2.918
7	56.61	1.677	56.71	1.920	56.64	2.264	56.32	3.413
8	56.64	1.903	56.69	2.277	56.57	3.211	56.18	4.041
9	56.66	2.167	56.79	2.584	56.65	3.660	56.21	4.758
10	56.68	2.335	56.78	2.931	56.60	4.089	56.18	5.472

CDC 7600
FTN compiler

2.13. Columns (ii) to (v) in Table IV.7 remind the outcome of this procedure given in detail in Chapter 2 while the columns (vi) to (ix) give information about the bounds and reductions produced after 5 iterations of the decomposition procedure.

The final lower and upper bounds obtained from procedure 2.13 are shown in columns (ii) and (iii) of the table, with the number of effective rows and columns for the problem given in entries (iv) and (v), respectively. The corresponding values for the decomposition are presented in columns (vi) - lower bound, (vii) - upper bound, (viii) -number of rows, and (ix) - number of columns. From the Table IV.7 it is clear that, although the improvement on the lower bound was not large (but effective for the test problems), the number of variables was reduced in all cases. This reduction is significant for test problems T200X2, T200X4 and T300X1 for which a new upper bound was obtained from the decomposition procedure. Also for problems T200X3 and T300X5 more than 10% of the variables were removed by reduced cost analysis even with the upper bound value not being tightened.

The perfomance of the decomposition procedure is also illustrated in Table IV.8. There, we present the computational results relative to the unicost test problems T50A1 (m=50;n=500;d=20%) when using the decomposition with two different sizes for the subproblems. Column (i) in Table IV.8 refers to the iteration number and columns (ii) and (v) give the corresponding lower bounds for the two decompositions. The cumulative number of variables removed is shown in columns (iv) and (vii) respectively for each decomposition. The significance of these figures comes from the fact that the LP relaxation for this problem gives a lower bound $z_l=3.31$ with no variables being removed. Hence, in either of the decompositions the lower bound is improved over 4% and some variables are removed.

The computing times given in columns (iii) and (vi), which are expressed in Cyber 170/855 seconds, show that the better quality of the bound produced from the 7-subproblem decomposition is paid in terms of time. Furthermore, as will be seen in next Chapter, the 10-subproblem decomposition (subproblems with 5 rows each one) proves better when the decomposition procedure is imbedded in a tree-search scheme for solving the problem.

TABLE IV.7

Computational results from the decomposition for the large scale test problems

PROBLEM	PROCEDURE 2.13				DECOMPOSITION			
	z_ℓ	z_μ	m	n	z_ℓ	z_μ	m	n
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)
T200X1	87.11	92.0	200	148	87.36	92.0	200	138
T200X2	66.35	74.0	199	178	66.45	72.0	199	146
T200X3	90.72	96.0	200	176	91.11	96.0	200	152
T200X4	69.35	74.0	199	150	69.61	73.0	199	125
T200X5	56.16	60.0	186	125	56.80	60.0	185	116
T300X1	215.00	215.0	-	-	-	-	-	-
T300X2	138.41	147.0	300	287	138.89	142.0	201	175
T300X3	211.83	223.0	300	333	213.87	223.0	300	323
T300X4	243.03	258.0	300	444	243.87	258.0	300	437
T300X5	187.70	192.0	250	224	188.85	192.0	212	192

TABLE IV.8

Lower bounds and computing times from the decomposition of a unicost SCP with two different dimensions for the subproblems

	5			8		
	z	t	n^R	z	t	n^R
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
1	3.35	0.584	-	3.43	0.624	1
2	3.36	1.803	-	3.45	1.682	1
3	3.37	2.507	-	3.49	2.527	2
4	3.39	3.231	-	3.52	3.453	4
5	3.40	3.973	-	3.45	4.609	5
6	3.42	4.750	-	3.47	6.971	8
7	3.42	5.521	-	3.48	7.026	8
8	3.43	6.324	1	3.50	8.296	9
9	3.44	7.149	3	3.51	9.525	10
10	3.45	7.995	4	3.44	11.048	10

Cyber 170/855 seconds
FTN compiler

Chapter 4

4.7 CONCLUSIONS

In this Chapter we presented and developed a procedure based on decomposition and state space relaxation for large size SCPs. From the exposition, we outline the following :

(i) The decomposition method, in which the SCP is divided into many smaller SCPs, is shown to produce improvements on the LP relaxation bound (if the dual optimal variables are available), or on the bound obtained from the lagrangean relaxation used for the SCP in this thesis (using the best lagrangean multipliers). This theoretical result was confirmed by the practical experience reported in this thesis.

(ii) State space relaxation couples well with the decomposition, both giving good approximations to the optimal values for the subproblems and providing reduced costs (in the form of a minimum value for the objective function when a particular variable is forced in the solution), which can be used in the total problem.

(iii) large size SCPs may then be further reduced using the decomposition method after procedure 2.13 has been completed. For some test problems, the decomposition also produced an improvement on the upper bound value.

(iv) The method also applies for the unicost SCP and seems possible that hard structure problems may be sucessfully tackled by combined use of decomposition and state space relaxation.

(v) A further step towards this consists of choosing carefully the partitions having different sizes for the subproblems or sorting the matrix. We sucessfully used a heuristic for sorting the constraint matrix into an almost block diagonal matrix.

(vi) Other techniques rather than subgradient optimization may be more efficient in computing the lagrange multipliers associated with the decomposition constraints (4.5) and further research on this topic could be worthwhile.

Chapter 5

TREE-SEARCH PROCEDURE FOR SOLVING THE SCP

5.1 INTRODUCTION

In this Chapter, a branch-and-bound algorithm for obtaining the optimal solution for a SCP is presented. The procedure combines the different techniques described in the previous Chapters - preliminary reductions, heuristic methods, lagrangean relaxation, linear programming, state space relaxation and decomposition - imbedding them in a tree-search scheme.

The utilisation of all these methods or only part of them, depends on the characteristics of the SCP being solved. For instance, all the techniques mentioned above are used in solving randomnly generated large size problems. For unicost SCPs, only heuristics, linear programming, state space relaxation and decomposition are used.

The tree search procedure is of the depth-first type with the branching rule chosen to take advantage of the decomposition by generating subproblems in the tree either with fewer number of rows or with sparser constraint matrix. The criterion for choosing the rows to branch, depends again on the type of the problem, as will be seen later.

At each node of the tree, a lower bound for the completion of the node is computed and a cover is generated from the corresponding solution. Also, reduced costs are obtained and used to remove temporarily variables from the problem.

The example from *Lemke et al. [126]* will be used again for illustrating the tree search procedure. Finally, computational results are shown for large scale SCPs with up to 400 rows

and 4000 columns, and for unicost SCPs with 50 rows, 500 columns and 20% density.

5.2 DESCRIPTION OF THE TREE

5.2.1 Root Node

At the root node of the tree we make use of the techniques described in Chapter 2 in order to reduce both the number of columns and the number of rows of the SCP. Also, a lower and an upper bound are obtained. All steps in procedure 2.13 are performed for randomnly generated large size problems. For the unicost SCPs, the greedy heuristic to compute an upper bound to the problem is used and then the LP lower bound is obtained using all the effective variables for the problem.

The next step consists of applying the procedure 4.3 relative to the decomposition method making use of the best lagrangean multipliers (case of large SCPs), or the optimal dual variables (unicost SCPs). The original SCP is, hence, split into many smaller and each one of these is solved or approximated by using procedure 4.2. Reduced costs (in the form of minimum value for the SCP when a variable is fixed equal to 1), are obtained and used to remove variables from the problem. If any number of variables is removed then reductions 2.2 and 2.3 (Chapter 2) are performed.

5.2.2 Branching

In case of further analysis on the current subproblem in the tree is required, then a branching occurs in the tree. This is done by making use of a branching strategy presented by *Etchberry [66]* which can be stated as :

Chapter 5

Branching strategy 5.1 :

Let i and l be the indices of two distinct rows of a SCP. Either

(i) i and l have a common variable, say $j \epsilon N_i \cap N_l$, in the optimal solution

or

(ii) i and l have no common covering variable in the optimal solution

This implies that :

(5.1) in case (i) the set N_i is restricted to $N_i \cap N_l$ and the row l is temporarily removed from the problem

or,

(5.2) in case (ii) the sets N_i and N_l are restricted to $N_i = N_i - (N_i \cap N_l)$ and $N_l = N_l - (N_i \cap N_l)$ respectively

This branching strategy is very suitable for the decomposition technique. In fact, in case (i) one row is deleted (row indexed l) and the other one (row i) has a reduced number of non-zero entries. Therefore, the number of rows and the density are both lower than in the previous node in the tree. On the other hand, for case (ii) all variables common to rows i and l, are removed and then the subproblem in the tree is sparser.

For choosing the rows i and l we adopted the following :

Rule 5.2 . *Choosing the rows for branching*

Step 1 . obtain i as the index of the row with maximum marginal cost for the problem under consideration, i.e. $s_i = smax = \max_{m \epsilon M} s_m$ with $s_m = \min_{j \epsilon N_m} t_j^*$ (obtained from (4.34))

Step 2 . select the variable $j^* \epsilon N_i$ such that $t_{j^*}^* = s_i = \min_{j \epsilon N_i} t_j^*$

Step 3 . if $M_{j^*} = \{ i \}$ then go to step 4.

otherwise, choose l as the index of the row with maximum cardinality cardinality and different from i :

$$\#N_l = \max_{m \in Mj^*} \#N_m \quad (m \neq i)$$

Step 4 . since $M_{j^*} = \{ i \}$, pick the column in N_i with
the next least cost t_j^* and go to step 3.
If there is no variable left to be inspected in N_i that means
that the one with minimum original cost must be fixed equal to 1
and so row i is covered. Then, go to step 1.

For the unicost SCPs, we choose the row index i as the minimum cardinality row instead of the one with the maximum value s_i. Since the difference between the values s_m ($m \epsilon M$) is rather small for this type of problem, it is reasonable to adopt a criterion based on the cardinality of the sets M_j in order to reduce the number of nodes in the tree. As will be seen later this is confirmed by computational experience.

5.2.3 Intermediate Node

At each intermediate tree node we perform 20 iterations for the lagrangean relaxation ($LSCP_\lambda$) of the problem under consideration. The initial values for the multipliers are the ones associated with the best lower bound obtained at the previous node. Hence, a first lower bound is computed for the optimal completion of the node and the corresponding multipliers are used to compute the decomposed costs by using (4.10), (4.11) and (4.12).

Since from the computational experience reported in the previous Chapter, the lower bound provided by the decomposition method at the first iteration is always better than the lagrangean bound and then increases very slowly from there, only one iteration is performed at each intermediate node.

Prime covers are generated from the partial solution sets and also from the values t_j as was seen in Chapter 4. Variables are temporarily removed from the problem if the corresponding value t_j for the current problem exceeds the value of the best upper bound, z_u. If the original costs of the variables are all integer then we can use z_u-1.0 instead.

Chapter 5

5.2.4 Backtracking on the Tree

Backtracking on the search tree occurs when :

(i) the lower bound obtained at the current node (z_{lb}) is greater than or equal to upper bound value (z_u), i.e. $z_{lb} > z_u - 1$ (integer costs).

or,

(ii) there is no feasible solution for the completion of the tree node.

When the bactracking occurs, the columns and rows removed during the analysis of the current node, are retrieved to the problem.

5.3 EXAMPLE

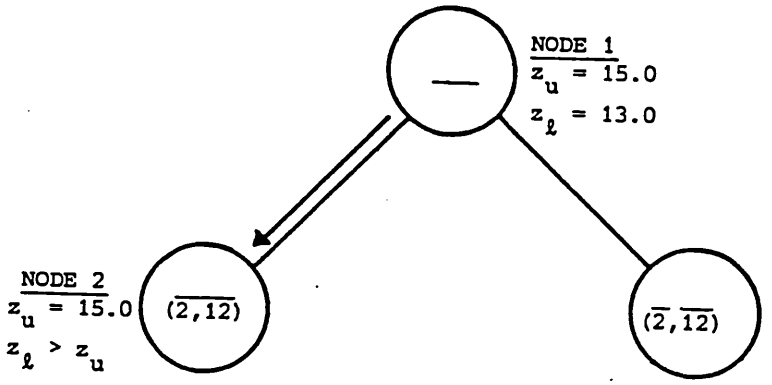
Let us consider the problem which we have been using as an example (*Lemke et al. [126]*), and for better illustrating the branch-and-bound we start branching at the end of the first iteration for the decomposition procedure (see.example 4.4). Applying the rule 5.2 :

step 1 . $s_2 = smax = max_{m=1,\ldots,15} s_m = 13.0$

step 2 . $t_{12} = t_{15} = t_{18} = 13.0$ and we arbitrarily choose $j^* = 12$

step 3 . $M_{12} = \{2,12\}$ and row 12 is obviously chosen

Then, two new nodes are generated in the search tree :



NODE 2 : node 2 is the first one to analyse. First, row 12 is temporarily removed

from the problem and $N_2=\{12\}$ which means that x_{12} must be in the optimal solution of the problem corresponding to node. Hence, also row 2 is temporarily removed and a fixed cost equal to 2.0 must be added to any bound obtained.

Now, a fixed number of iterations for the lagrangean relaxation would be performed. However, to avoid too many computations for the example, we consider the same dual feasible solution we used in example 4.4. That is, $u=(0,0,2,0,1,0,1,2,1,1,0,1,0,3.0)$, and the decomposed costs are computed as before :

j	1	2	3	4	5	6	7	8	9	10	11	14
d_1^j		1	1			1			2	1/2		-
d_2^j	1			1			1		0	3/2	2	-
d_3^j					1			1			0	-

j	15	16	18	19	20	21	23
d_1^j	5/2	0	1/2	4		4	11/3
d_2^j		0			5		14/3
d_3^j	1/2	3	7/2	1		1	2/3

. START SOLVING THE SUBPROBLEMS

. subproblem 1

i	1	2	3	4	5	Q
q_1	0	-	2	0	1	3

j	2	3	6	9	10	13	15	16	18	19	21	23
d_1^j	1	1	1	2	1/2	2	5/2	0	1/2	4	4	11/3
q_1^j	0	0	1	2	0	0	2	0	0	3	3	3
t_1^j	4	4	3	3	7/2	5	7/2	3	7/2	4	4	11/3

The optimal solution is provided by SSR with value $z_{lb}^1=3.0$. The partial solution set is $X_1=\{16,6,9\}$ and the values t_1^j $(j\epsilon A'_1)$ are given in the bottom line of the tableau above.

. subproblem 2

i	6	7	8	9	10	Q
q_1	0	1	2	1	1	5

Chapter 5

j	1	4	7	9	10	11	16	20	23
d_2^j	1	1	1	0	3/2	2	0	5	14/3
q_2^j	0	1	1	0	1	2	0	5	4
t_2^j	6	5	5	5	11/2	5	5	5	17/3

The optimal solution is again obtained from the SSR - $X_2=\{16,20\}$ - with value $z^2_{lb}=5.0$ and the corresponding t_2^j ($j\epsilon A'_2$) are given in the tableau above.

. subproblem 3

i	11	12	13	14	15	Q
q_1	0	-	0	3	0	3

j	5	8	11	14	15	16	18	19	21	23
d_3^j	1	1	0	3	1/2	3	7/2	1	1	2/3
q_3^j	0	0	0	3	0	3	3	0	1	0
t_3^j	4	4	3	3	7/2	3	7/2	4	4	11/3

The solution provided by the SSR is $X_3=\{11,14\}$, which is not feasible. The lower bound is $z^3_{lb}=3.0$ with the subsequent t_3^j ($j\epsilon A'_3$) given in the tableau above.
Since, $d_{13}=\min(t_3^8,t_3^{18},t_3^{23})=3.5$ the lower bound is further improved to $z^3_{lb}=3.5$. The new values for the t_3^j are :

5	8	11	14	15	16	18	19	21	23
4.5	4.5	3.5	3.5	3.5	3.5	3.5	4	4	11/3

Hence, the lower bound obtained so far for the completion of node 2 is $z_{lb}=11.5+2.0=13.5$ and the costs t_j are :

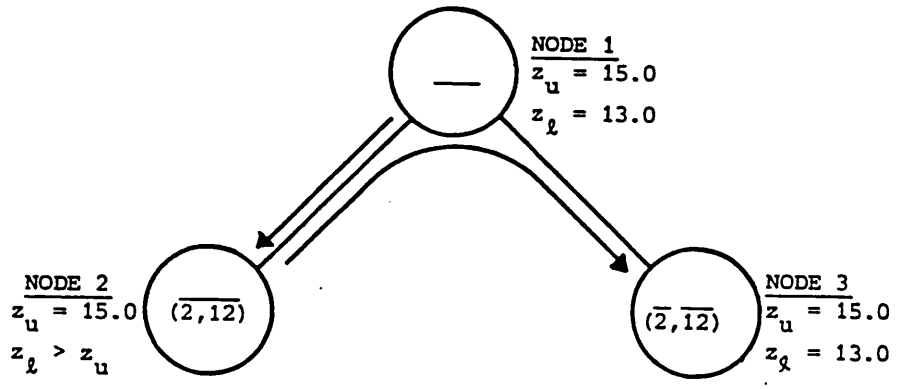
j	1	2	3	4	5	6	7	8	9	10	11	12
t_1^j	3	4	4	3	3	3	3	3	3	7/2		-
t_2^j	6	5	5	5	5	5	5	5	5	11/2	5	-
t_3^j	7/2	7/2	7/2	7/2	9/2	7/2	7/2	9/2	7/2	7/2	7/2	-
t_j	14.5	14.5	14.5	13.5	14.5	13.5	13.5	14.5	13.5	14.5	13.5	-

j	13	14	15	16	18	19	20	21	23
t_1^j	5	3	7/2	3	7/2	4	3	7/2	11/3
t_2^j	5	5	5	5	5	5	5	5	17/3
t_3^j	7/2	7/2	7/2	7/2	7/2	4	7/2	4	11/3
t_j	15.5	13.5	14.5	13.5	14.0	15.0	13.5	15.5	15.0

Then $x_1,x_2,x_3,x_5,x_8,x_{10},x_{13},x_{19},x_{21}$ and x_{23} are temporarily removed from the problem, since $t_j>z_u-1$ for these variables. The resulting problem is further reduced by applying reduction 2.3 (single 1 in a row) :

i	1	3	4	5	6	7	8	9	10	11	13	14	15
j													
4								1					
6				1									
7						1							
9		1		1									
11						1							1
14												1	1
15	1									1			
16	1		1		1								
18			1								1	1	1
20					1	1	1	1	1				

Then, the variables x_{15},x_{16},x_{18} and x_{20} are fixed equal to 1, which means that the fixed valye for the lower bound goes up to 17.0. Therefore, the analysis of the node is completed and, since $z_{lb}>z_u-1$, a backtrack on the tree occurs :



NODE 3 : all variables removed except x_{17} and x_{22} are retrieved and the same occurs for the rows.

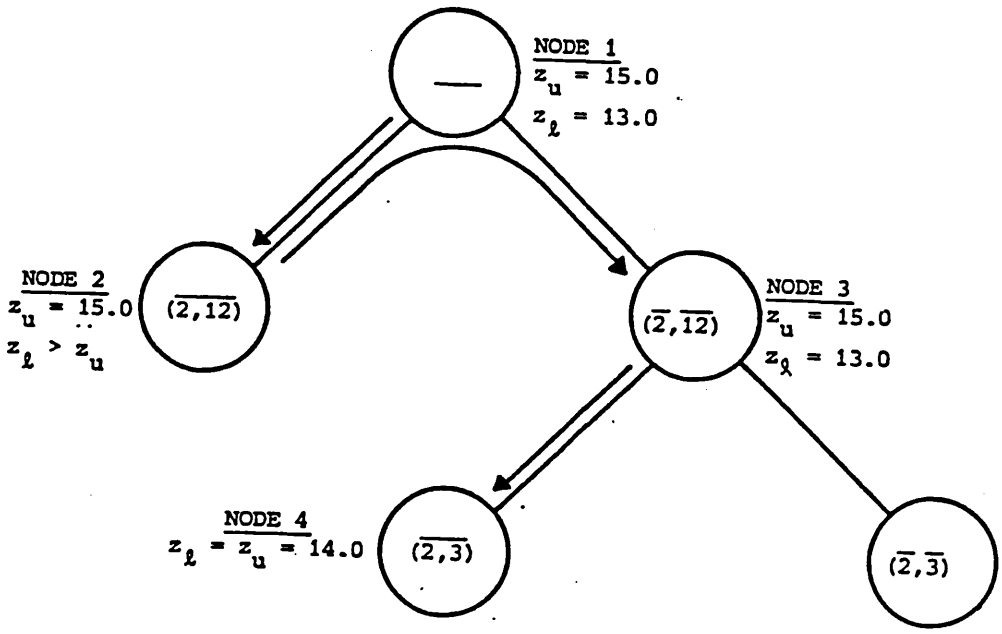
At this node of the tree, the sets N_2 and N_{12} are restricted to :

$N_2=\{13,15,18,19,23\}$

$N_{12}=\{8,21\}$

and this means that $x_{12}=0$. Using again the same dual feasible solution for generating the decomposed costs, we start the decomposition procedure with the same values as at the root node. Doing the computations, we get the same lower

.. bounds for the subproblems as at the root node (first iteration of the decomposition in example 4.4). Also the values t_j are the same and this means that row 2 is again chosen for the branching. .nl The selected variable is now $j^*=15$ and from $M_{j^*}=\{2,3,11\}$ is row indexed 3 is the second row for the branching :



NODE 4 : at node 4, we replace N_2 by $N_2 \cap N_3=\{15,19,23\}$ and row indexed 3 is temporarily removed from the problem. Recomputing the decomposed costs :

j	1	2	3	4	5	6	7	8	9	10	11	12
		1	1			1				1/2		-
	1			1			1		2	3/2	2	-
					1			1			0	-

j	13	14	15	16	18	19	20	21	23
	2		3/2	0	1/2	3		3	7/3
				3			5		16/3
		3	3/2	3	7/2	2		2	4/3

. subproblem 1

i	1	2	3	4	5	0
q_i	0	0	-	0	1	0

Chapter 5

j	2	3	6	9	10	13	15	16	18	19	21	23
d	1	1	1	-	1/2	2	3/2	0	1/2	3	3	7/3
q	0	0	0	-	0	0	0	0	0	1	1	1
t	10/3	10/3	10/3	-	17/6	13/3	23/6	7/3	17/6	3	3	7/3

$z_{lb}^1=7/3$; $X_1=\{16,23\}$

. subproblem 2

i	6	7	8	9	10	Q
q_1	0	1	2	1	1	5

j	1	4	7	9	10	11	16	20	23
d	1	1	1	2	3/2	2	0	5	16/3
q	0	1	1	0	1	2	0	5	4
t	6	5	5	7	11/2	5	5	5	19/3

$z_{lb}^2=5$, $X_2=\{16,20\}$

. subproblem 3

i	11	12	13	14	15	Q	q_1	0	1	0	3	0	4

j	5	8	11	14	15	16	18	19	21	23
d	1	1	0	3	3/2	3	7/2	2	2	4/3
q	0	1		3	0	3	3	0	1	0
t	5	4	4	4	9/2	4	9/2	6	5	16/3

$z_{lb}^3=4.0$; $X_3=\{11,8,16\}$

The lower bound obtained at the end of this iteration for the decomposition is $z_{lb}=7/3+5+4=34/3$. However, from the analysis of the values t_j the lower bound is increased to $z_{lb}=14.0$ $[=s_2=\min(t_{15},t_{19},t_{23})]$.

This time, a cover with better cost is obtained from the set :

$S=\{ x_j : t_j \leq z_{lb} \}$

Removing the redundant variables by decreasing order of the costs, one obtains :

$S=\{4,7,8,9,10,11,13,16\}$

with cost $c(S)=14.0$. Therefore, a backtrack on the tree occurs, since $z_{lb}=z_u=14.0$ at node 4.

Chapter 5

NODE 5 : the variable x_{15} is retrieved to the problem and so is the row indexed 3.
Now, the sets N_2 and N_3 are restricted to $N_2=\{13,18\}$ and $N_3=\{9,21\}$. The matrix of the SCP corresponding to node 5 is then :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1						1									
2				1											
3	1														
4									1						
5															1
6					1										
7							1								
8											1	1			
9			1			1									
10	1									1					
11								1							1
13		1		1											
14														1	1
15											1				
16	1			1		1					1			1	
18		1		1								1	1	1	
19	1				1						1		1		
20						1	1	1	1	1					
21	1		1		1						1	1			
23	1			1	1	1		1	1	1			1		

and the decomposed costs are :

1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	18	19	20	21	23
	1	1			1			2	1/2		2			0	1/2	5/2		7/2	7/3
			1			1		0	3/2	2				0					16/3
				1			1			0		3	3	3	7/2	3/2		3/2	4/3

Solving again the three subproblems we would get the initial lower bound $z_{lb}=12.5$ which is increased to 13.0 by reduced cost analysis. The variables $x_1,x_2,x_3,x_5,x_{13},x_{15},x_{19}$ and x_{23} are temporarily removed from the problem. The variable x_{18} is fixed equal to 1 (since $N_2=\{18\}$ by the removing of x_{13}).

New branching now occurs in the tree. There is a tie for choosing the row i, since d_2=smax and row indexed 2 has been covered by x_{18}. Then a decision is taken picking the least cardinality row with minimum index - row 3, with j*=9 and l=6.

NODE 6 : solving the SCP corresponding to node 6 one would get a lower bound z_{lb}=16.0 and, therefore, a backtrack occurs on the tree.

NODE 7 : finally, for node 7, another feasible solution with value equal to 14.0 is obtained - X={18,20,21}.
The tree search is now over (see Figure 5.1), and the optimal value for the example is 14.0 .

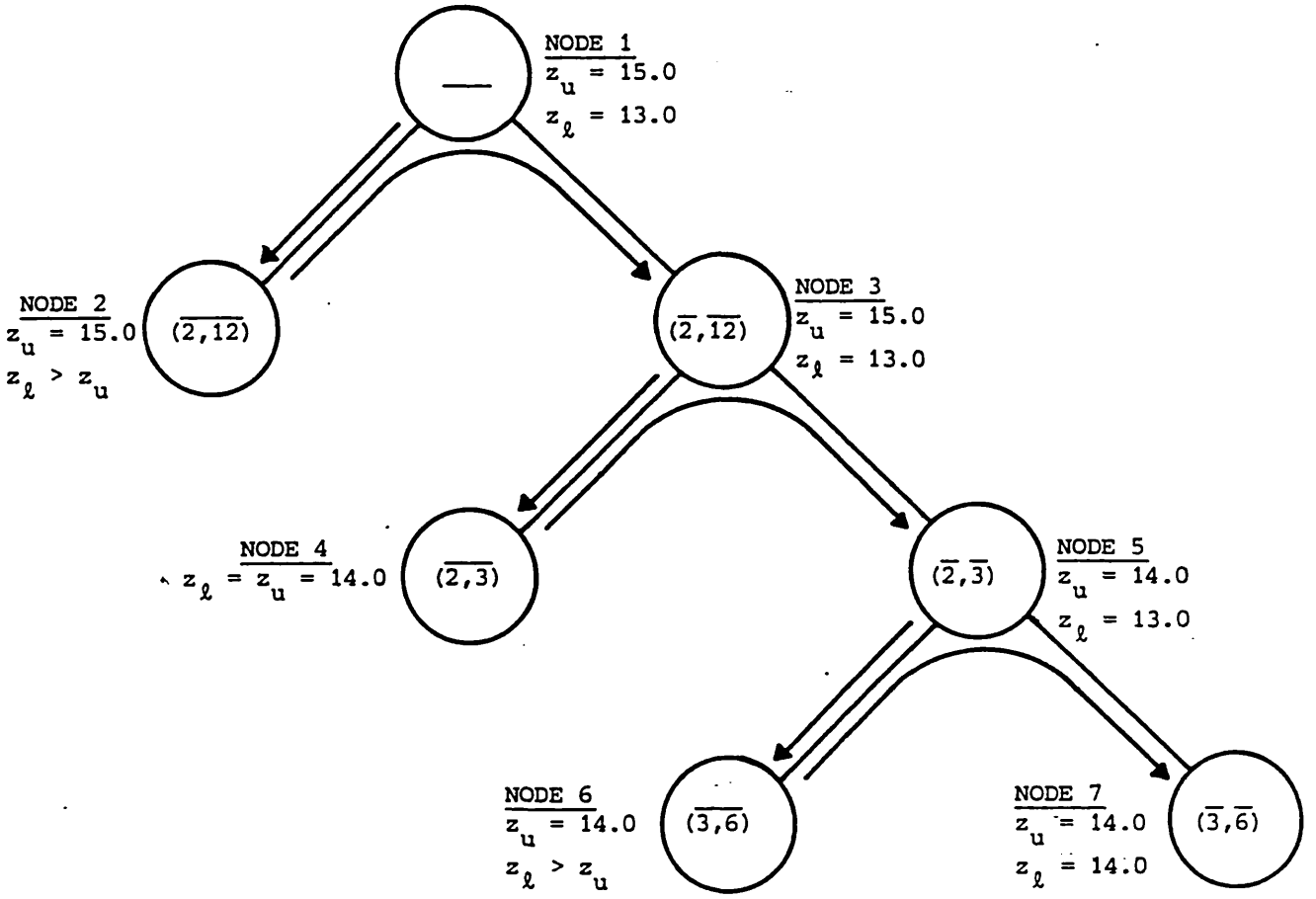


Figure 5.1

Complete search tree for the SCP in example (5.3)

Chapter 5

5.4 COMPUTATIONAL RESULTS

Table V.1 shows the complete computational results for the test problems of class (I) considered in Chapter 2. Note that the first set of problems, T200Xk (k=1,..,5), are 5% density problems with 200 rows and 2000 columns. The other set, T300Xk (k=1,...,5), is composed by problems with 300 rows, 3000 columns and 2% density. For both sets of problems, the costs of the variables are randomnly generated from the range [1,99].

As can be seen, all problems are solved using a CDC 7600 computer. This occurs in spite of the upper bound at the beginning of the tree being more than 3% far from the optimal for problems T200X2 (4.2%), T200X3 (3.2%), T300X2 (3.1%) and T300X4 (4.4%). In all cases the number of nodes generated in the tree was less than 100.

The capability of the method to solve large size SCPs is illustrated in Table V.2 where we show the computational results relative to a test problem with 400 rows, 4000 columns, 2% density and costs generated from the range [1,99]. This problem was solved on a CYBER 170/855 which is slightly faster than the CDC 6600 (and therefore slower than the CDC 7600 used to solve the other test problems mentioned above). The computing times given in Table V.2 were obtained by using the command -INFORM,JOBCOST- right before and after running each one of the parts into which we split the computer code.

Unicost test problems with 50 rows, 500 columns and 20% density, were also solved using the algorithm with the variations adopted for this type of SCP. That is, the greedy heuristic was first used to obtain an upper bound and then the complete LP relaxation was solved. The optimal dual variables were used to set the initial decomposed costs and the tree search procedure was performed.

Table V.3 shows the computational results for five test problems solved on a CYBER 170/855 computer. The first column in Table V.3, (i), refers to the designation of the problem and the following ones give the information relative to each one of the three phases described above - column (ii) shows the upper bound value obtained from the greedy heuristic; column (iv) gives the LP lower bound; (vi), the optimal value for the problem and (viii) the number of nodes

TABLE V.1

Final computational results for large scale SCPs

PROBLEM	PROCEDURE 2.13					TREE SEARCH			
(i)	z_ℓ (ii)	z_u (iii)	m (iv)	n (v)	time (vi)	z_{opt} (vii)	n (viii)	time (ix)	total time (x)
T200X1	87.12	92.0	200	146	6.81	90.0	15	7.96	14.77
T200X2	66.35	74.0	199	178	7.08	71.0	73	28.69	35.77
T200X3	90.72	96.0	200	176	8.28	93.0	41	17.07	25.35
T200X4	69.35	94.0	199	150	6.60	73.0	77	25.22	31.82
T200X5	66.16	60.0	186	125	5.28	60.0	37	9.75	15.03
T300X1	215.0	215.0	-	-	10.06	215.	-	-	10.06
T300X2	138.4	147.0	300	287	9.63	141.	20	15.02	24.65
T300X3	211.8	223.0	300	333	10.87	218.	56	21.50	32.37
T300X4	243.0	258.0	300	444	13.87	247.	17	28.96	42.63
T300X5	187.2	192.0	250	224	7.53	192.	30	18.98	26.51

CDC 7600
FTN compiler

TABLE V.2

Computational results for a SCP with 400 rows, 4000 columns and 2% density

PRELIMINARY REDUCTIONS				TIME
m	n			
400	619			
GREEDY HEURISTIC				
z_L	z_u	m	n	
147.	251.	400.	619.	
LAGRANGEAN RELAXATION (I)				
z_L	z_u	m	n	
207.4	230.	400	618	10.367
LINEAR PROGRAMMING				
z_L	z_{LP}	m	n	
206.86	227.05	400	617	35.799
LAGRANGEAN RELAXATION (II)				
z_L	z_U	m	n	
218.60	230.0	400	524	7.151
TREE SEARCH				
z_{opt}	nodes			
230.	131			560.584
				613.901

Cyber 170/855
FTN compiler

generated. Columns (iii), (v) and (viii) give the partial computing times in Cyber 855 seconds whose total is shown in column (ix). The figures giving computational times include input and output time since they were obtained by using the command -INFORM,JOBCOST- right before and after running the code. Therefore, they are not accurate but they are overestimates of the real computing time in the mentioned machine.

The unicost SCPs were decomposed into 10 subproblems and the choice of the branching rows was made with respect to cardinality, instead of the cost d_i ($i \epsilon M$). It is significant to mention that for problem T50A3, Rule 5.2 was tried out and generated a tree with as much as twice the number of the nodes generated by the version adopted.

TABLE V.3

Computational results for the unicost SCPs

	GREEDY HEURISTIC		LINEAR PROGRAMMING		TREE SEARCH			TIME
(i)	z_u (ii)	t (iii)	z_{ℓ} (iv)	t (v)	z_{opt} (vi)	n (vii)	t (viii)	(ix)
T50A1	5.0	1.21	3.30	5.83	5.0	167	278.23	285.28
T50A2	5.0	1.42	3.40	3.66	5.0	140	252.98	258.06
T50A3	5.0	1.51	3.47	5.20	5.0	191	323.87	330.58
T50A4	5.0	1.28	3.35	5.99	5.0	190	313.68	320.95
T50A5	5.0	1.47	3.39	6.09	5.0	166	301.50	309.06

Cyber 170/855
FTN compiler

Chapter 5

5.5 CONCLUSIONS

In this Chapter we concluded the description of an algorithm for large scale SCPs based on decomposition and state space relaxation. From the final computational results, it is possible to say that :

(i) The algorithm is capable of solving large size SCPs and, in particular, a test problem with 400 rows, 4000 columns and 2% density was solved. This is by far the largest SCP reported solved in the literature.

(ii) The algorithm with a few variations solves unicost SCPs with 50 rows, 500 columns and 20% density which are the largest dimensions for this type of problem that have been reported solved.

Taking into account the conclusions and suggestions for future research formulated in the previous Chapters, one may say that further improvements are likely to be produced by :

(iii) Improving the procedure used to obtain bounds on the optimal value and perform reductions in both the number of the rows and the number of the columns. This can possibly be done by trying more heuristics for generating covers, performing different types of reduction tests (based on lagrangean penalties) or producing an even more reduced LP restricted relaxation but with good accuracy.

(iv) Producing a better decomposition depending on the structure of the problem and allowing variable sizes for the subproblems. Also, other techniques rather than subgradient optimisation may be more suitable for computing the lagrangean multipliers associated with the decomposition constraints.

REFERENCES

1. Aho, A.V., Hopcroft, J.E. and Ullman, J.D. "The Design and Analysis of Computer Algorithms", Addison-Wesley, (1974).

2. Arabeyre, J.P., Fearnley, J., Steiger, F.C. and Teather, W. "The Airline Crew Scheduling Problem: a survey", Transportation Science, 3, (1969), pp140-163.

3. Avis, D., "A Note on Some Computationally Difficult Set Covering Problems", Mathematical Programming, 18, (1980), pp138-145.

4. Baker, E., Bodin, L.D., Finnegan, W.F., and Ponder, R.J. "Efficient Heuristic Solutions to an Airline Crew Scheduling Problem", AIEE Transactions, 11, (1979), pp79-84.

5. Baker, E. and Fisher, M. "Computational Results for very large Air Crew Scheduling Problems", Omega, The International Journal of Management Science, 9, (1981), pp613-618.

6. Baker, K.R., "Scheduling a Full-time Workforce to meet Cyclic Staffing Requirements", Management Science, 20, (1978), pp1561-1568.

7. Baker, K.R., "Workforce Allocation in Cyclic Scheduling Problems : a survey", Operational Research Quarterly, 27, (1976), pp155-167.

8. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-one Variables", Operations Research, 13, (1965), pp517-546

9. Balas, E., "Set Covering with Cutting Planes from Conditional Bounds", MSRR 399, Carnegie-Mellon University, 1977.

10. Balas, E., "Cutting Planes from Conditional Bounds : A new approach for set covering", Mathematical Programming Study 12, Combinatorial Optimization, (1980), pp19-36.

11. Balas, E. and Ho, A., "Set Covering Algorithms using Cutting Planes, Heuristics and Subgradient Optimization: A computational study", Mathematical Programming, Study 12, Combinatorial Optimization, (1980), pp37-60.

12. Balas, E. and Padberg, M.W. "On the Set Covering Problem II: an algorithm", Management Sciences Research Report No 295, Carnegie-Mellon University, (1972).

13. Balas, E., and Padberg, M.W. "On the Set Covering Problem", Operations Research, 20, (1972), pp1152-1161.

14. Balas, E. and Padberg, M.W. "Set Partitioning: a survey", Combinatorial Programming: methods and applications (Ed. B.Roy). D. Reidel Publishing Company, (1974).

15. Balas, E. and Padberg, M.W. "On the Set Covering Problem, II. An Algorithm for Set Partitioning", Operations Research, 23, (1975), pp74-90.

16. Balas, E. and Padberg, M.W. "Set Partitioning : a survey", Combinatorial Optimization, Ed.N. Christofides et al, John Wiley & Sons, (1979).

17. Balas, E. and Zemel, E., "Solving large zero-one Knapsack Problems", Management Science Research Report, No 408, Carnegie-Mellon University, (1976).

18. Balinski, M.L., "On Maximum Matching, Minimum Covering and their connections", Proceedings of the Princeton Symposium on Mathematical Programming, Ed.H.W. Kuhn, Princeton University Press, Princeton, (1970).

19. Balinski, M.L. and Quandt, R.E. "On an Integer Program for a Delivery Problem", Operations Research, 12, (1964), pp300-304.

20. Ball, M., Bodin, L.D. and Dial, R., "A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles", Working paper 81-007, (1981), College of Business and Management, University of Maryland.

21. Bartholdi, III, J.J., Orlin, J.B. and Ratliff, H.D., "Cyclic scheduling via Integer Programs with Circular Ones", Operations Research, 28, (1980), pp1034-1035.

22. Bartholdi, III, J.J., "A Guaranteed-Accuracy Round-off Algorithm for Cyclic Scheduling and Set Covering", Operations Research, 29, (1981), pp501-510.

23. Bazaraa, M., and Shetty, C.M., "Nonlinear Programming: Theory and Algorithms", John Wiley & Sons, New York, (1979).

24. Beale, E.M.L., "Survey of Integer Programming", Operational Research Quarterly, 16, (1965), pp219-228.

25. Beasley, J.E., "An Algorithm for Set Covering Problems", Internal Report, Department of Management Science, Imperial College, London, (1983).

26. Bell, D.E. and Shapiro, J.F., "A convergent Duality Theory for Integer Programming", Operations Research, 25, (1977).

27. Bellman, R., "Dynamic Programming", Princeton University Press, Princeton, NJ, (1957).

28. Bellman, R. "On the Reduction of Dimensionality for Classes of Dynamic Processes", Journal of Math Anal Appl, 3, (1969), p358.

29. Bellman, R. and Dreyfus, S.E., "Function Approximation and Dynamic Programming", Mathematical Tables and Aids to Computation, 13, (1959), p247.

30. Bellman, R. and Dreyfus, S.E., "Applied Dynamic Programming", Princeton University Press, Princeton, NJ, (1957).

31. Bellman, R. and Kalaba, R., "On the k^{th}-best policies", Journal of SIAM, 8, (1960), pp582-588.

32. Bellmore, M., Greenberg, H.J. and Jarvis, J.J., "Multi-commodity Disconnecting Sets", Management Science, 16, (1970), B427-B433.

33. Bellmore, M. and Ratliff, H.D., "Optimal Defence of Multi-commodity Networks", Management Science, 18, (1971), pp174-185.

34. Bellmore, M. and Ratliff, H.D., "Set Covering and Involuntory Bases", Management Science, 18, (1971), pp194-206.

35. Berge, C., "Balanced Matrices", Mathematical Programming, 2, (1972), pp19-31.

36. Berlin, G.N. and Liebman, J.C.,"Mathematical Analysis of Emergency Ambulance Location", Socio-Economic Planning Sciences, 8, (1974), pp323-328.

37. Berlin, G.N., ReVelle,C.S. and Elzinga, D.J., "Determining Ambulance-hospital Locations for on-scene and Hospital Services". Environment and Planning A, 8, (1976), pp553-561.

38. Bigman, D. and ReVelle, C., "An Operational Approach to Welfare Considerations in Applied Public Facility Location Models", The John Hopkins University, Baltimore, Maryland 21218, (1978).

39. Bilde, O., and Krarup, J., "Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem", Annals of Discrete Mathematics, 1, (1977), pp79-97.

40. Boder, J.M. "A Method for solving Crew Scheduling Problems" Operational Research Quarterly, 12, (1975), pp55-62.

41. Bodin, L., Golden, B., Assad, A., and Ball, M., "The State of the Art in the Routing and Scheduling of Vehicles and Crews", Working paper, 81-035, (1981), College of Business and Management, University of Maryland.

42. Breu, R. and Burdet, C.A. "Branch and Bound Experiments in 0-1 Programming", Mathematical Programming Study, 2, (1974), ppl-50.

43. Breuer, M.A. "Simplication of the Covering Problem with Application to Boolean Expressions", Journal for the Association of Computing Machinery, 17, (1970), ppl60-181.

44. Burdet, C.A. and Johnson, E.L., "A Subadditive Approach to solve Linear Integer Programs", Annals of Discrete Mathematics, 1, (1977), ppll7-144.

45. Camerini, P.M. Fratta, L. and Maffioli, F., "On Improving Relaxation Methods by Modern Gradient Techniques" Mathematical Programming Study, 3, (1975), pp26-34.

46. Christofides, N., "Zero-One Programming using non-binary Tree Search", The Computer Journal, 14, (1971), pp418-421.

47. Christofides, N., "Graph Theory : an Algorithmic Approach", John Wiley & Sons, (1975).

48. Christofides, N., "A Minimax Facility Location Problem and the Cardinality Constrained Set Covering Problem", MSRR 375, Carnegie-Mellon University, Pittsburgh, (1975).

49. Christofides, N., "The Vehicle Routing Problem", Revue francaise d'Automatique, Informatique et Recherche Opérationelle, 10, (1976), pp55-70.

50. Christofides, N. and Hey, A.M., "Graph Covering Relaxations of the Set-covering Problem - Part I", Report IC-OR-79-04, Department of Management Science, Imperial College, London, (1979).

51. Christofides, N. and Korman, S., "A computational Survey of methods for the set covering problem", Management Science, 5, (1975), pp591-599.

52. Christofides, N., Mingozzi, A., and Toth, P., "State Space Relaxation Procedures for the Computation of Bounds to Routing Problems", Networks, 11, (1981), pp145-164.

53. Church, R., and Revelle, C. "The maximal covering location problem", The Regional Science Association, XXXII, (1974), pp101-118.

54. Chvatal, V., "A Greedy Heuristic for the Set-Covering Problem", Mathematics of Operations Research, 4, (1979), pp233-235.

55. Corley, H.W. and Roberts, S.D., "A Partitioning Problem with Applications in Regional Design", Operations Research, 20, (1972), pp1010-1019.

56. Crowston, W.B. "Decision Network Planning Models", Ph.D. Thesis, Carnegie-Mellon University, (1968).

57. Dantzig, G.B. and Ramser, J.H. "The truck dispatching problem". Management Science, 6, (1959), pp80-96.

58. Daskin, M.S. and Stern, E.H., "A Hierarchical Objective Set-Covering Model for Emergency Medical Service Vehicle Deployment", Transportation Science, 15, (1981), pp137-152.

59. Day, R.H. "On Optimal extracting from a multiple file data storage system: an application of Integer Programming", Operations Research, 13, (1965), pp482-494.

60. Dreyfus, S.E., and Law, A.M., "The Art and Theory of Dynamic Programming", Academic Press, New York, (1977).

61. Dyer, M.E., "Calculating Surrogate Constraints". Mathematical Programming, 19, (1980), pp255-278.

62. Edmonds,J., "Covering and Packing in a Family of Sets", Bulletin of American Mathematical Society, 68, (1962), pp494-499.

63. Edmonds, J., "Maximum Matchings and a polyhedron with 0-1 vertices". Journal of Research National Bureau of Standards, 69B, (1965), ppl25-130.

64. Edmonds, J. and Johnson, E.L., "Matching: a well solved class of integer linear programs", in 'Proc. Calgary International Conference on Combinatorial Structures and their applications", Gordon and Breach, New York, (1970), pp84-96.

65. Erlenkotter, D.,"A Comparative Study of Approaches to Dynamic Location Problems", European Journal of Operations Research, 6, (1981), ppl33-173.

66. Etcheberry, J., "The Set-Covering Problem. A new implicity enumeration algorithm", Operations Research, 25, (1977), pp760-772.

67. Everett, H., "Generalised Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources", Operations Research 11, (1963), pp399-417.

68. Fisher, M.L., Northrup, W.D. and Shapiro, J.F., "Using Duality to Solve Discrete Optimization Problems: theory and computational experience". Mathematical Programming Study, 3, (1975), pp56-94.

69. Freeman, D.R. and Jucker J.V., "The Line Balancing Problem" Journal of Industrial Engineering, 18, (1967), pp361-364.

70. Frehel, J., "Le Probleme de Partition Sous Constrainte", Combinatorial Programming: Methods of Applications (Ed. B.Roy) NATO Advanced Study Institute Series (1974), Series C: Mathematical and Physical Sciences.

71. Fulkerson, D.R., "Blocking and Anti-blocking Pairs of Polyhedra", Mathematical Programming, 1, (1971), ppl68-194.

72. Fulkerson, D.R., Nemhauser, G.L. and Trotter, L.E.(jnr), "Two Computationally Difficult Set Covering Problems that arise in computing the 1-width of incidence matrices of Steiner Triple Systems", Mathematical Programming Study 2, (1974), pp72-81.

73. Gabow, H., "An Efficient Implementation of Edmond's Algorithm for Maximum Matching on Graphs", Journal for the Association of Computing Machinery, 23, (1975), pp221-234.

74. Garey, M. and Johnson, D.S., "Computers and Intractability: A Guide to the Theory of NP completeness", Freeman & Co. San Francisco, (1978).

75. Garfinkel, R.S., "Optimal Set Covering: a survey", Perspectives in Optimization: a collection of expository articles" (ed. Geoffrion), Reading, Man., Addison-Wesley (1972).

76. Garfinkel, R.S. and Nemhauser, G., "The Set-Partitioning: set covering with equality constraints", Operations Research, 17, (1969), pp848-856.

77. Garfinkel, R.S. and Nemhauser, G.L., "Optimal Political Districting by Implicity Enumeration Techniques", Management Science, 16, (1970), ppB-495/B-508.

78. Garfinkel, R.S. and Nemhauser, G.L., "Integer Programming", John Wiley and Sons, (1972).

79. Gavish, B., "On Obtaining the 'best' multipliers for a Lagrangean Relaxation for Integer Programming", Computers and Operations Research, 5, (1978), pp55-71.

80. Gavish, B., Schweiter, P., and Shlifer,E., "Assigning Buses to Schedules in a Metropolitan Area", Computer and Operation Research, 5, (1978), pp129-138.

81. Geoffrion, A.M. "Lagrangean Relaxation for Integer Programming", Mathematical Programming Study, 2, (1974), pp82-114.

82. Geoffrion, A.M. "How can Specialized Discrete and Convex Optimization Methods be married?", Annals of Discrete Mathematics, 1, (1977), pp205-220.

83. Geoffrion, A.M. and Marsten, R.E. "Integer Programming Algorithms: a framework and state-of-the-art survey". Management Science, 18, (1972), pp465-491.

84. Geoffrion, A.M. and Macbride, R.,"Lagrangean Relaxation Applied to Capacitated Facility Location Problems", AIIE Transactions, 10, (1978), pp40-47.

85. Ghiggi, C., Puliafito, P.P. and Zoppoli, R., "A Combinatorial Method for Health-Care Districting", in Optimization Techniques: Modelling and Optimization in the service of man, Part 1, Proceedings, 7th IFIP Conference, Ed.Springer Verlag (1976).

86. Gimpel, J.P. "A Reduction Technique for Prime Implicant Tables" IEEE Transactions Electronic Computers, EC-14, (1965), pp535-541.

87. Glover, F.,"A Multiphase-dual Algorithm for the Zero-one Integer Programming Problem", Operations Research, 13, (1965) pp879-919.

88. Glover, F., "Surrogate Constraints", Operations Research, 16, (1968), pp741-749.

89. Glover, F., "Surrogate Constraint Duality in Mathematical Programming", Operations Research, 23, (1975), pp434-451.

90. Glover, F., and Woolsey, E., "Converting the 0-1 Polynomial Programming Problem to an 0-1 linear program", Operations Research, 22, (1974), pp180-182.

91. Gomory, R.E., "Some Polyhedron Connected with Combinatorial Problems", Linear Algebra, 2, (1969), pp451-558.

92. Gondran, M., "Set Partitioning and Covering Algorithms Applications and Algorithms", Bulletin Dir.Etud.et Recherche, 2, (1976) (French), pp59-68.

93. Granot, F., and Hammer, P., "On the role of generalised covering problems", Center for Cybernetic Study Research Report CS96, University of Texas at Austin, (1972).

94. Greenberg, H.J. "The Generalised Penalty Function Surrogate Model", Operations Research, 21, (1973), pp162-178.

95. Greenberg, H.J. and Pierskalla, W.P. "Surrogate Mathematical Programming", Operations Research, 18, (1970), pp924-938.

96. Greenberg, H.J. and Pierskalla, W.P. "Quasi-functions and Surrogate duality", Cahier Centre d'etude de Recherche Operationalle, 15, (1973), pp437-448.

97. Guha, D., "The Set-Covering Problem with Equality Constraints", Operations Research, 21, (1973), pp348-351.

98. Gunawardane, G., "Dynamic Versions of Set Covering type Public facility location problems", European Journal of Operation Research, 10, (1982), pp190-195.

99. Hakimi, S.L. and Frank,H., "Maximum Internally Stable Sets of a Graph", Journal of Mathematical Analysis and Applications, 25, (1969), pp296-308.

100. Hammer, P.L. and Rudeanu, S., "Boolean Methods in Operations Research and Related Areas", Springer-Verlag, (1968).

101. Hausman, D. and Korte, B., "Adjacency on 0-1 Polyhedra", Mathematical Programming Study, 8, (1978), pp106-127.

102. Held, M. and Karp, E., "The Travelling Salesman Problem and Minimum Spanning Trees II", Mathematical Programming, 1 (1971), p6-25.

103. Held, M., Wolfe, P. and Crowder, H., "Validation of Subgradient Optimization", Mathematical Programming, 6, (1974), pp62-88.

104. Heurgon, E., "Un Probleme de Recouvrement: l'habillage des honaires d'une tigne d'autobus". RAIRO, V-1, (1972), pp13-29.

105. Hey, A., "Algorithms for the Set Covering Problem", Ph.D.Thesis, Department of Management Science, Imperial College, London, (1980).

106. Hey, A.M. and Christofides, N., "Lower Bounds to the Set Covering Problem from Network Flow Relaxations", Bell Laboratories Report, 1981, Piscataway, New Jersey, U.S.A

107. Ho, A.C., "Worst Case Analysis of a Class of Set Covering Heuristics", Management Science Research Report no 473, Carnegie-Mellon University, (1979).

108. Ho, A.C., "Optimal Integer and Fractional Covers: a sharp bound on their ratio", GSIA, Carnegie-Mellon University, (1981).

109. Hochbaum, D.,"Approximation Algorithms for the Weighted Set Covering and Node Cover Problems", GSIA, Carnegie-Mellon University, (1980).

110. Hoffman, A.J., and Kruskal, J.B., "Integral Boundary Points of Convex Polyhedra", Linear Inequalities and Related Systems, Eds.H.N.Kuhn and A.W.Tucker, Princeton University Press, Princeton, NJ, (1956), pp223-246.

111. Horowitz, E.J. and Sahni, S., "Fundamentals of Computer Algorithms", Pitmans Publishing, (1979), London.

112. House, R., Nelson, L. and Rado, T., "Computer Studies of a certain class of linear integer programs", Recent advances in Optimization Techniques, Eds. Lavi and Vogel John Wiley and Sons, New York (1965).

113. Hung, M.S. and Brown, J.R., "An Algorithm for a Class of Loading Problems", Naval Research Logistics Quarterly, 25, (1978), pp289-297.

114. Itai, A. and Rodeh, M., "Covering a Graph by Circuits", Automata Languages of Programming, ed. G.Ausiello and C. Bohm, 62, de série lecture notes on computing sciences.

115. Jeroslow, R.G., "Cutting Plane Theory: Algebraic Methods, Discrete Mathematics, 23 (1978), pp121-150.

116. Johnson, D.S. "Approximation Algorithms for Combinatorial Problems", Journal Computer System Sci., 9, (1974), pp256-278.

117. Karp, R.M., "On the Computational Complexity of Combinatorial Problems", Networks, 5, (1975), pp45-68.

118. Karp, R.M., "The Probabilistic Analysis of some Combinatorial Search Algorithms", Algorithms and Complexity, Ed. J.F.Tranb, Academic Press (1976), pp1-19.

119. Karwan, M.H. and Rardin, R.L. "Some Relationships between Lagrangean and Surrogate duality in Integer Programming", Mathematical Programming, 17, (1979), pp320-334.

120. Karwan, M.H. and Rardin, R.L. "Surrogate Duality in a Branch-and-bound Procedure". Naval Research Logistics Quarterly, 28, (1981), pp93-100.

121. Kerningan, B.W. and Lin, S., "An Efficient Heuristic Procedure for Partioning Graphs", Bell Systems Technical Journal, 49, (1970), pp291-307.

122. Klee, V. and Minty, G.J. "How Good is the Simplex Algorithm?", Inequalities III, ed. O.Shisha, Academic Press, (1972), ppl59-175.

123. Korman, S.M. "Graph Colouring and Related Problems in OR", Ph.D.Thesis, Department of Mangement Science, Imperial College, London, (1975).

124. Krarup, J. and Pruzan, P.M. "Selected Families of Location Problems", Annals of Discrete Mathematics, 5, (1979), pp327-387.

125. Lawler, E.L. "Covering Problems: duality relations and a new method of solution", Journal of SIAM, 14, (1966), ppl115-1132.

126. Lemke, C.E., Salkin, H.M. and Spielberg, K., "Set Covering by Single-Branch Enumeration with Linear-programming Subproblems". Operations Research, 19, (1971), pp998-1022.

127. Lovász, L., "On the Ratio of Optimal Integral and fractional Covers", Discrete Mathematics, 13, (1975), pp383-390.

128. Marsten, R.E., "An Algorithm for Large Set Partitioning Problems", Management Science, 20, (1974), pp774-787.

129. Marsten, R.E., "The Design of the XMP Linear Programming Library", ACM Transactions on Mathematical Software, 7, (1981), pp481-497.

130. Marsten, R.E., Muller, M.R. and Killion, C.L., "Crew Planning at Flying Tiger: a successful application of Integer Programming", Technical Report No 553, Management Information Systems Department, University of Arizona, (1978).

131. Marsten, R. and Shepardson, F. "Exact Solutions of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications", Networks 11, (1981), pp167-177.

132. Martin, G.T., "An Accelerated Euclidean Algorithm for Integer Linear Programming", Recent Advances in Mathematical Programming, Eds. G. Graves and P. Wolfe, McGraw-Hill, (1963), pp311-317.

133. Mevert, P. and Rohde, M.,"Set Partitioning and Side Constraints", Working Paper 24/78, Freie Universitat, Berlin, (1978).

134. Michaud, P., "Exact Implicity Enumeration Methods for Solving the Set Partitioning Problem", IBM Journal of Research and Development, 16, (1972), pp573-578.

135. Nakornchai, V., "Interactive Computer Methods for Plant layout Scheduling and Group Technology", Ph.D Thesis, Department of Management Science, Imperial College, London, (1982).

136. Nemhauser, G.L., "Introduction to Dynamic Programming", John Wiley & Sons, New York, (1966).

137. Nemhauser, G.L., Trotter, L.E. and Nauss, R.M. "Set Partitioning and Chain Decomposition", Management Science 20, (1974), pp1413-1423.

138. Nemhauser, G.L. and Weber, G.M. "Optimal Set Partitioning Matchings and Lagrangean Duality", Naval Logistics Research Quarterly, (1979), pp553-563.

139. Padberg, M.W., "On the facial Structure of Set Packing Polyhedra", Mathematical Programming, 5, (1973), pp199-215.

140. Parker, M. and Smith, B.,"Two Approaches to Computer Crew Scheduling", Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling, Ed.A. Wren, North-Holland Publishing Co., (1981), pp193-222.

141. Pierce, J.B., "Application of Combinatorial Programming to a Class of all-zero-one Integer Programming Problems", Management Science, 15, (1968), pp191-209.

142. Pierce, J.F., "A Two Stage Approach to the solution of the Vehicle Dispatching Problem", presented at the 17th TIMS International conference, London, (1970).

143. Pierce, J.F. and Lasky, J.S. "Improved Combinatorial Programming Algorithms for a class of All-zero-one Programming Problems", Management Science, 19, (1973), pp528-543.

144. Quine, W.V. "The Problem of Simplifying Truth Functions", American Mathematical Monthly, 59, (1952), pp521-531.

145. Reiter, C. and Sherman, G., "Discrete Optimizing" SIAM Review, 13, (1965), pp864-889.

146. ReVelle, C.S. and Swain, R.W., "Central Facilities Location". Geographical Analysis, 2, (1970), pp30-42.

147. ReVelle, C.S., Toregas, C. and Falkson, L., "Applications of the Location Set-Covering Problem", Geographical Analysis, 8, (1976), pp65-76.

148. ReVelle, C.S., Marks, D., and Liebman, J.C., "An Analysis of Private and Public Sector Location Models", Management Science, 16, (1976), pp692-707.

149. Richardson, R., "An Optimization Approach to Routing Aircraft", Transportation Science, 10, (1976), pp52-71.

150. Roth, R., "Computer Solutions to Minimum-Cover Problems", Operations Research, 17, (1969), pp455-465.

151. Rothstein, M., "Operations Research in the airlines", Bureau of business research and services, University of Connecticut Storrs, Connecticut (paper presented at the joint ORSA/TIMS/AIEE Atlantic City meeting) (1972).

152. Roy, B. "An Algorithm for a General Constrained Set Covering Problem". Computing and Graph Theory, Read, ed. Academic Press, (New York) (1972).

153. Rubin, J. "A Technique for the Solution of Massive Set Covering Problems, with Application to Airline Crew scheduling", IBM Philadelphia Scientific Center Technical Report No 320-3004, (1971).

154. Rubin, J., "A Technique for the Solution of Massive Set Covering Problems with Application to Airline Crew Scheduling", Transportation Science, 7, (1973), pp34-48.

155. Rutman, R.A. "An Algorithm for Placement of Interconnected Elements Based on Minimum Wire Length", Proceedings of AFIPS Congress, 20, (1964), 477.

156. Salkin, H. and Koncal, R.D., "Set Covering by an All-integer Algorithm: computational experience". Journal for the Association of Computing Machinery, 20, (1973), pp189-193.

157. Salveson, M.E. "The Assembly Line Balancing Problem", Transactions ASME, 77, (1955), pp939-947.

158. Samuelsson, H.M., "Solving Large Set Covering Problems", WP196, State University of New York, Buffalo, (1974).

159. Schreuder, J.A.M. "Application of a location Model for Fire Stations in Rotterdam". European Journal of Operational Research, 6, (1981), pp212-219.

160. Sergeev, N.D. "Minimization of the number of arithmetic operations in problems with sparse symmetric matrices". USSR Comp.Math and Math.Phys, 12, (1977), pp19-29.

161. Shapiro, J.R. "Generalised Lagrange Multipliers in Integer Programming ", Operations Research, 19, (1971) pp68-76.

162. Shepardson, F. and Marsten, R.E. "A Lagrangean Relaxation Algorithm for the Two Duty Period Scheduling Problem". Management Science 26, (1980), pp274-281.

163. Spitzer, M. "Solution to the Crew Scheduling Problem". AGIFORS Symposium, (1961).

164. Steiger, F. and Neiderer, M. "Scheduling Air Crews by Integer Programming", presented at IFIP Congress, Edinburgh, (1980).

165. Steinberg, L., "The Blackboard Wiring Problem: a placement algorithm", SIAM Review, 3, (1961), p37.

166. Steinmann, H. and Schwinn, R. "Computational Experience with Zero-one Programming Problem", Operations Research, 17, (1969), pp917-920.

167. Thiriez, H.M., "The Set Covering Problem : a group theoretic approach", Revue francaise d'Automatique, Informatique et Recherche Opérationelle, 3, (1971), pp83-104.

168. Thuve, H.,"Frequency Planning as a Set Partitioning Problem" European Journal of Operational Research, 1 (1981), pp29-32.

169. Tibrewala, R., Phillipe, D. and Browne, J., "Optimal Scheduling of Two consecutive Idle Periods", Management Science, 19, (1982), pp71-75.

170. Topalian, K.R. "Tree Search Methods and the Set Covering Problem". M.Sc.Thesis, Department of Management Science, Imperial College, London, (1971).

171. Toregas, C., Swain, R., ReVelle, C.S. and Bergman, L., "The Location of Emergency Service Facilities". Operations Research 19, (1971), ppl363-1373.

172. Toregas, C. and Revelle, C., "Optimal Location under Time or Distance Constraints". Papers of The Regional Science Association, 28, (1972).

173. Toregas, C. and Revelle, C. "Binary Logic Solutions to a Class of Location Problem", Geographical Analysis, 5 (1973), ppl45-155

174. Trubin, V., "On a Method of Solution of Integer Linear Programming Problems of special kind", Soviet Math.Dokl. 10, (1969), ppl544-1546.

175. Walker, W. "Using the Set Covering Problem to Assign Fire Companies to Fire Houses". Operations Research, 22, (1974), pp275-277.

176. Walukiewicz, S., "Some Aspects of Integer Programming Duality", European Journal of Operational Research, 7, (1981), ppl96-202.

177. Werra, D.de., "On some Characterisations of Totally Unimodular Matrices", Mathematical Programming, 20, (1981), ppl4-21.

178. White, J.A. and Case, K.E., "On Covering Problems and The Central Facilities Location Problem", Geographical Analysis.

179. White, L.J. and Gillenson, M. "An Efficient Algorithm for Minimum k-covers in Weighted Graphs". Mathematical Programming 8, (1975), pp20-42.

180. Wollmer, R.D. "Multicommodity Supply and Transportation Networks with Resource Constraints", The Rand Corporation Report R.M-6143-PR.

181. Wolsey, L.A. "Generalised Dynamic Programming Methods in Integer Programming". Mathematical Programming, (1973), pp222-232.

182. Wolsey, L.A. "Valid Inequalities and super Additivity for 0-1 IP's", Mathematics of Operations Research, 2, (1977), pp66-72.

183. Wolsey, L.A. "Integer Programming Duality: price functions and sensitivity analysis", Mathematical Programming, 20, (1981), pp173-195.

184. Zadeh, N. "A Bad Network Problem for the Simplex Method and other Minimum Cost Flow Algorithms". Mathematical Programming, 5, (1973), pp255-266.

185. Zorychta, K. "On Converting the 0-1 Linear Programming Problems to an SCP", Bulletin de Académie Polonaise des Sciences, Serie des Sciences Mathematiques, Astronomiques et Physiques, 25, (1977), pp919-923.