ALGORITHMS FOR LOCATION AND ROUTING PROBLEMS IN DISTRIBUTION SYSTEMS

Roberto Baldacci

The Management School
Imperial College of Science, Technology and Medicine
University of London

ALGORITHMS FOR LOCATION AND ROUTING PROBLEMS IN DISTRIBUTION SYSTEMS

Roberto Baldacci

A thesis submitted for the degree of

DOCTOR OF PHILOSOPHY OF THE UNIVERSITY OF LONDON

and the

DIPLOMA OF IMPERIAL COLLEGE

January 1999

The Management School
Imperial College of Science, Technology and Medicine
University of London

To my Family

ABSTRACT

This thesis is concerned with the study of location and routing problems in distribution systems.

In the first part of the thesis we consider the Capacitated p-median Problem (CPMP) in which a set of n customers must be partitioned into p disjoint clusters so that the total dissimilarity within each cluster is minimized and constraints on maximum cluster capacities are met. The total dissimilarity of a cluster is computed as the sum of the dissimilarities existing between each customer of the cluster and the median associated with the cluster.

We develop both an exact and a heuristic algorithm for solving the CPMP. The exact method is based on a set partitioning formulation of the problem and the heuristic method on a recently proposed metaheuristic, namely, the Bionomic algorithm. The computational results for test problems obtained from the literature show the effectiveness of both algorithms.

In the second part of the thesis we consider a class of routing problems.

We describe a two-commodity network flow approach to derive new integer programming formulations for the Vehicle Routing Problem (VRP), the Traveling Salesman Problem (TSP) with mixed deliveries and collections and the TSP with Backhauls. These formulations are used to derive new lower bounds based on linear relaxation strengthened by new valid inequalities.

An exact algorithm is developed for the VRP with Backhauls based on a set partitioning formulation. Problems of large size can be solved to optimality.

Finally, a real-life application of the Multi-Depot Period VRP to resource planning in the utilities sector is presented. The computational implementation of the planning model is described and results are obtained with reference to a specific case-study.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Dr. Eleni Hadjiconstantinou, for her guidance, support and for providing me with many ideas and suggestions which have led to new results presented in this thesis.

I would like also to express my thanks to Professor Nicos Christofides and Professor Aristide Mingozzi for their suggestions and useful discussions.

I am very grateful to the Fondazione Cassa di Risparmio di Cesena and to the Consiglio Nazionale delle Ricerche for giving me the opportunity to carry out this work. Their financial support is gratefully acknowledged.

I also thank my other colleagues, Mr. Marco Boschetti and Dr. Vittorio Maniezzo for sharing with me their research experience and friendship.

Last, but not least, I am very much indebted to my parents and my sisters for their support and encouragement throughout my studies.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

1.1 DISTRIBUTION SYSTEMS

Logistics is described by Eilon et al. (1971) as "the provision of goods and services from a supply point to a demand point". A complete logistics system covers the entire process of moving raw materials and input requirements from suppliers to plants, the conversion of the inputs into products (or outputs) by a manufacturing process, and the delivery of these products to the final customers through intermediate stores or depots. The *Distribution System* covers the supply of goods from the plants to the customers. It is concerned with the movement and storage of goods via various distribution channels to the supply chain endpoint. The distribution process represents the most costly part of this chain (see Christofides (1981)).

The management of a distribution system involves a variety of decision-making problems at both the strategic and tactical operational levels. Decisions relating to the number and location of facilities (plants, warehouses or depots) may be viewed as strategic, while the problem of routing the vehicles to deliver goods from depots to customers could be classified as tactical. The distinction between strategic/operational *Location Problems* and *Routing Problems* does not only depend on the nature of the decisions involved but is also associated with the time-span and frequency of the decisions. The problem of depot location is considered by management once every few years and once such a decision is taken and implemented it cannot be easily changed without incurring major capital investment. The vehicle routing problem, on the other hand, arises on a regular basis (e.g. daily), needs to be resolved repeatedly and routes may have to be altered from one-time period to another.

Location and Routing Problems cannot always be treated in isolation; they are interlinked and a solution to the one may affect the other. Efficient solutions to tactical and operational problems such as route planning can be successfully incorporated in providing satisfactory solutions to strategic problems.

This chapter provides general concepts relevant to Combinatorial Optimization Problems and briefly surveys some known methodologies used by the algorithms developed in the thesis. The motives and goals for the research in location and routing problems are clearly stated. An overview of the thesis is presented at the end of this chapter.

1.1.1 LOCATION PROBLEMS

The depot location problem is not a single problem, but a combination of inter-related sub-problems. The number of depots in the system, their respective sizes, their locations, the allocation of customers to depots - all these are inter-related problems which need to be closely examined. Furthermore, in determining the location of depots it may be necessary to take account of the availability of suitable sites, the proximity of trunk roads (or access to other means of transport) and the availability of labour, as well as numerous other factors. Consequently, the general approach has been to treat the problem as a multi-stage decision process in which the parameters are optimized in sequence, each time assuming the other parameters to remain constant.

A basic location decision problem with many practical applications involves determining the location of facilities, such as industrial plants or depots, to minimize the cost (or maximize the profit) of satisfying the demand for some commodity. In general there are fixed costs for locating the facilities and transportation costs for distributing the commodities between the facilities and the customers. This problem has been extensively studied in the literature and is commonly referred to as the *plant location problem*, or *facility location problem*. Variations of this problem have been considered in the literature. When each potential facility has a constraint on the maximum demand that it can supply (known as capacity), the problem becomes the *capacitated facility* *location problem*. When capacity constraints are not considered, the problem is referred to the *simple* or *uncapacitated facility location problem*, or, for short, the UFL problem. If the number of facilities to locate in the UFL problem is specified the problem is known as the p-facility location problem. When there are no fixed costs associated with the facilities the problem is known as the *p-median problem*.

There is a vast literature on the UFL problem (see Krarup and Pruzan (1983)). An interesting review of key aspects of Location Theory and its applications in real-world discrete location problems is given in the book edited by Mirchandani and Fransis (1990).

The UFL problem has several applications: locating a bank account (see Cornuejols et al. (1977)), clustering analysis (see Mulvey and Crowder (1979)), lock-box location (see Kraus et al. (1970)) and portfolio management (Beck and Mulvey (1982)). This problem also arises as a subproblem in several contexts, such as in network design, vehicle routing and, of course, location theory when additional constraints, such as capacity constraints, are present. This latter problem is known as the *Capacitated p-median problem* (CPMP). The CPMP is the subject of chapters 2 and 3 of this thesis, where both heuristic and exact methods are presented.

1.1.2 ROUTING PROBLEMS

The Vehicle Routing Problem (VRP) was originally posed by Dantzig and Ramser (1959) as the Truck Dispatching Problem and has grown to be an important area of Operations Research. It has been estimated that transportation costs account for nearly half, 47.5%, of the total logistics cost (Institute of Logistics and Distribution Management (1985)), and approximately 70% of the value-added costs in the soft drink industry (see Golden and Wasil (1987)). This percentage is on average 20% of any product value, but it varies widely with the type of product. There is, therefore, a clear incentive for organizations to use transport as efficiently as possible and a variety of computerized routing software has been implemented for this purpose.

In its basic version the VRP can be stated as follows. A set of customers, each with a known location and a known requirement for some commodity, is to be supplied from a single depot by delivery vehicles of known capacity. The problem is to design routes for the vehicles, subject to the following constraints:

(a) the requirements of all customers must be met;

(b) vehicle capacity must not be violated, i.e. the total load allocated to each vehicle must not exceed its capacity.

The objective of this problem may be stated as that of minimizing the cost of completing the delivery routes.

There is a vast literature on the VRP. For recent surveys of VRP solution methods and their applications see Magnanti (1981), Bodin et al. (1983), Christofides (1985), Golden and Assad (1986,1988), Bodin (1990), Laporte (1992b) and Fisher (1995). See also the recent bibliographies by Laporte and Osman (1995) and by Laporte (1997).

If the fleet consists of a single vehicle having a sufficiently large capacity, so that constraint (b) can be ignored, the problem consists of finding the shortest tour to visit all customers and this is the same as the classical Traveling Salesman Problem (TSP). The first major algorithmic study of the TSP is that of Dantzig et al. (1954). The TSP is NP-hard (see Garey and Johnson (1979)) and its study has given rise to several theoretical and algorithmic results, some having far reaching effects in other areas of combinatorial optimization. Polyhedral theory (see Grötschel and Padberg (1985)) is probably the most significant of these and has led to the development of powerful exact algorithms.

The book by Lawler et al. (1985) contains an account of the main results on the TSP until 1985. For a more recent survey, see Laporte (1992a).

Real VRPs usually include complications beyond the basic model. Typical complications include the following.

(a) The objective function in real problems can be quite complex, including terms dependent on the distance travelled, the number of vehicles used, the time duration of routes (as with overtime pay for drivers) and penalties for not delivering to all customers.

(b) The time of delivery to a customer may be constrained to fall within defined *time windows*.

(c) In many situations the same vehicles used for goods delivery are also used to collect pallets, empty bottles, etc. for return to the depot. When these activities are particularly significant they are often referred to as *backhauls*.

(d) The characteristics of the vehicles can introduce a variety of constraints beyond the simple vehicle capacity constraints. The vehicle fleet can be heterogeneous, with each vehicle having a distinct capacity. There can be multiple capacity constraints because of both weight and volume restrictions. Sometimes vehicles are divided into compartments for storage of different products (such examples include the delivery of petrol to gas-stations, delivery of refrigerated and non-refrigerated items to supermarkets, etc.). Customer/vehicle compatibility constraints may restrict the set of customers that a vehicle can feasibly service.

(e) Routes may extend over more than one day and/or a vehicle may perform more than one trip in a day returning to the depot several times for reloading.

(f) There may be more than one distribution depot and these depots may be interacting in a way that makes it impossible to consider any one in isolation. For example, vehicles may leave from one depot, supply some customers, return to a second depot to reload (perhaps with a product not available at the first depot), visit another set of customers and finally arrive back at the first depot (or at a third depot).

(g) Periodic routing problems arise in the distribution of products such as soft drinks, snacks foods, beer and bread. In these applications the distributing firm is interested in developing a set of daily routes for some *T* day period so that each customer receives delivery at a designed frequency.

(h) Inventory routing problem arise in the distribution of liquid products such as industrial gases or gasoline. In these problems, each customer has an inventory of the product, and the distributor must determine the timing and amount of deliveries so that the customer does not run out of product.

1.2 COMBINATORIAL OPTIMIZATION

This section provides general concepts relevant to Combinatorial Optimization (CO) Problems.

CO is a term that describes those areas of mathematical programming that are concerned with discrete structure. A large number of practical problems can be formulated and solved as CO problems.

Very generally, a *mathematical programme* is an optimization problem subject to constraints in $\mathbb{R}^n$ of the form:

$$\begin{aligned} &Min\ f(\mathbf{x}), \\ &subject\ to \quad \mathbf{x} \in S \subseteq \mathbb{R}^n, \end{aligned} \tag{1.1}$$

where $\mathbb{R}^n$ is the set of all n-dimensional vectors of real numbers and f is a real-valued function defined on S. The set S is called the *constraint* set and f is called the *objective function.* The vector $\mathbf{x} \in \mathbb{R}^n$ has components $x_1, x_2, \ldots, x_n$ which are the *variables* of the problem. Every $\mathbf{x} \in S$ is called a *feasible solution* to (1.1). If there is an $\mathbf{x}^\circ \in S$ satisfying:

$$-\infty < f(\mathbf{x}^\circ) \leq f(\mathbf{x}), \text{ for all } \mathbf{x} \in S,$$

then $\mathbf{x}^\circ$ is called an *optimal solution* (or also a *global optimal solution* to (1.1)). Notice that considering only the case of minimization of (1.1) is not restrictive, since the search for a maximum of function f reduces immediately to the problem of minimization of $g = -f$.

The objective in a mathematical programming problem is to establish whether an optimal solution exists and then to find one, or perhaps all, optimal solutions. In an applied context, it is convenient to think of (1.1) as a model of decision making in which S represents the set of all permissible decisions and f assigns a utility or profit to each $\mathbf{x} \in S$. Applications of this model abound in the real world and are relevant to various branches of engineering, business, and the physical and social sciences.

Different classes of problems can be obtained by placing restrictions on the type of function under consideration and on the values that the variables can take. Problems in which the decision variables are discrete, i.e. where the solution is a set, or a sequence, of integers or other discrete objects, are called *combinatorial problems.* The problem of finding optimal solution to such problems is therefore known as *Combinatorial Optimization.*

Some examples of this kind of problems are as follows.

Example 1.2.1 *The 0-1 Knapsack Problem.*

Given a set of *n items* and a *knapsack,* with:

$$p_j = \text{profit of item } j,$$

$$w_j = \text{weight of item } j,$$

$$c = \text{capacity of the knapsack},$$

determine the subset J of items which should be packed in order to maximize

$$\sum_{j \in J} p_j$$

such that

$$\sum_{j \in J} w_j \leq c .$$

Here the solution is represented by the subset $J \subseteq \{1,2,\ldots,n\}$.

Example 1.2.2 *The Vehicle Routing Problem.*

A set of M identical vehicles of capacity Q is located at a central depot and must be used to supply a set of n customers. Each customer i requires a supply of q_i units from the depot. Every route performed by a vehicle which starts and ends at the depot and the load carried must be smaller or equal than Q. Each customer must be visited exactly once by a vehicle route. The distance between customer i and j is d_{ij}.

The problem is to assign customers to vehicles and find the order in which each vehicle visits its customers so as to minimize

$$\sum_{k=1}^{M} \sum_{i=0}^{n_k} d_{\pi_{ik}, \pi_{i+1k}}$$

such that

$$\sum_{i=1}^{n_k} q_{\pi_{ik}} \leq Q, \; k = 1,2,\ldots,M$$

$$\text{and } \sum_{k=1}^{M} n_k = n .$$

Here, vehicle k visits n_k customers and π_{ik} is the i-th customer visited along the k-th route. The solution is represented by the permutation $\{\pi_{11},\ldots,\pi_{n_1 1},\ldots,\pi_{1M},\ldots,\pi_{n_M M}\}$. Notice that the depot is represented by the customer π_{0k} and by the customer π_{n_k+1k}, for each k.

1.2.1 INTEGER LINEAR PROGRAMMING PROBLEMS

Combinatorial problems, such as those described above, can be solved by (i) formulating the problem as an integer linear programming problem (e.g., by introducing integer variables) and (ii) by using the solution techniques developed for solving integer linear programming problems.

An *integer linear programming problem* (ILP) is a mathematical programming problem in which:

$$f(\mathbf{x}) = \mathbf{cx} \tag{1.2}$$

and

$$S = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \text{ and integer}\}, \tag{1.3}$$

where **A** is an $m \times n$ matrix, **b** is an m-vector, **c** is an n-vector and **0** is an n-vector of zeros. In more standard form the ILP is written as

$$\left.\begin{aligned} \mathit{Min}\quad & \mathbf{cx} \\ \mathit{subject\ to}\quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \text{ integer} \end{aligned}\right\} \tag{1.4}$$

In summation notation (1.4) is

$$\left.\begin{aligned} \mathit{Min}\quad & \sum_{j=1}^{n} c_j x_j \\ \mathit{subject\ to}\quad & \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1,\dots,m \\ & x_j \geq 0 \text{ integer}, \quad j = 1,\dots,n \end{aligned}\right\} \tag{1.5}$$

A *linear programming problem* (LP) is a mathematical programming problem in which $f(\mathbf{x})$ is given by (1.2) and

$$S = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

The LP obtained by dropping the integrality constraints from the ILP (1.3) is referred to as the *corresponding* LP, or the *linear relaxation* of the ILP.

In general, the problem

$$\text{P1: } \mathit{Min}\ f(\mathbf{x}),\ \mathbf{x} \in S_1$$

is said to be a *relaxation* of the problem

$$\text{P2: } Min\ f(\mathbf{x}),\ \mathbf{x} \in S_2$$

if $S_1 \supseteq S_2$. Similarly, P2 is said to be a *restriction* of P1.

Notice that if $\mathbf{x}^\circ$ is an optimal solution to P1 and $\mathbf{x}^*$ is an optimal solution to P2, then $f(\mathbf{x}^\circ) \leq f(\mathbf{x}^*)$. Furthermore, if $\mathbf{x}^\circ \in S_2$, then $\mathbf{x}^\circ$ is an optimal solution to P2.

An important special case of the ILP (1.5) is the *binary* ILP, where $x_j \geq 0$ and integer is replaced by $x_j = 0 \text{ or } 1$. A generalization of the ILP is the *mixed integer linear program* (MILP), where only some of the variables are constrained to be integer.

The following terms are used in the description of the solution space of a discrete optimization problem.

The set of all possible solutions of a MILP or ILP may be described by a set of linear constraints. Finding these constraints and their properties is the subject of *polyhedral theory.* For a detailed treatment of this subject see Rockafellar (1970), Padberg and Grötschel (1985) and Nemhauser and Wolsey (1988).

Given a set $S \subseteq \mathbb{R}^n$, a point $\mathbf{x} \in \mathbb{R}^n$ is a *convex combination* of points of *S* if there exists a finite set of points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$ in *S* and a vector $\lambda \in \mathbb{R}^t$ of non-negative values with $\sum_{i=1}^{t} \lambda_i = 1$ and $\mathbf{x} = \sum_{i=1}^{t} \lambda_i \mathbf{x}_i$. The *convex hull* of *S*, denoted by *conv*(*S*), is the set of all points that are convex combinations of *S*. An important result is that *conv*(*S*) can be described by a finite set of linear inequalities. Further $min\{\mathbf{cx} : \mathbf{x} \in S\} = min\{\mathbf{cx} : \mathbf{x} \in conv(S)\}$. Thus any MILP or ILP can be represented as a LP, provided we know a set of linear inequalities that represent the solution space.

A *polyhedron P* can be represented in the form $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} \leq \mathbf{b}\}$. A polyhedron of the form $P = \{\mathbf{x} \in \ ^n : \|\mathbf{x}\| \leq b\}$ for some $b > 0$, where $\|\mathbf{x}\|$ is a norm (e.g. euclidean) of **x**, is bounded. A bounded polyhedron is a *polytope.*

A nonempty set $S \subseteq \mathbb{R}^n$ is called *affinely independent,* if for every finite set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} \subseteq S$, the equations $\sum_{i=1}^{k} \alpha_i \mathbf{x}_i = 0$ and $\sum_{i=1}^{k} \alpha_i = 0$ imply $\alpha_i = 0$, $i = 1, \ldots, k$; otherwise *S* is called *affinely dependent.*

A polyhedron P is of dimension k, denoted by $dim(P)=k$, if the maximum number of affinely independent points of P is $k+1$. $P \subseteq \mathbb{R}^n$ is full-dimensional if the dimension of P is n.

The inequality $\pi\mathbf{x} \leq \pi_0$ is called a *valid inequality* for P if it is satisfied for all points in P. Given an ILP, a linear inequality that cut out part of the feasible region of the corresponding LP while leaving the feasible region of the ILP intact is called a *cutting plane.* If $\pi\mathbf{x} \leq \pi_0$ is a valid inequality for P, and $F = \{\mathbf{x} \in P : \pi\mathbf{x} = \pi_0\}$, then F is called a *face* of P. A face of P is a *facet* of P if $dim(F) = dim(P) - 1$. This leads to the results that for each facet F of P, one of the inequalities representing F is necessary in the description of P. Thus the use of facets in the description of the solution space yields a system of inequalities of smallest number. Also, if P defines the convex hull of integer solutions of a discrete optimization problem, then the use of facet defining inequalities is most likely to give the tightest lower bound in a branch and cut scheme.

In the case of the 0-1 Knapsack Problem, we can formulate the problem as a binary ILP by defining:

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is packed} \\ 0 & \text{otherwise} \end{cases}$$

The problem then reduces to the following integer program:

$$\begin{aligned} Max \quad & \sum_{j=1}^{n} p_j x_j \\ subject\ to \quad & \sum_{j=1}^{n} w_j x_j \leq c \\ & x_j = 0 \text{ or } 1,\ j \in \{1,2,\ldots,n\} \end{aligned}$$

There is an important class of binary ILP's in which $a_{ij} = 0 \text{ or } 1$ for all i and j, and $b_i = 1$ for all i. Special methods have been developed for these so-called *covering* and *partitioning* problems.

The *set covering problem* is the zero-one integer program

$$\begin{aligned} Min \quad & \mathbf{cx} \\ subject\ to \quad & \mathbf{Ex} \geq \mathbf{e} \end{aligned}$$

$$x_j = 0 \text{ or } 1, j \in \{1,2,\ldots,n\}$$

where $\mathbf{E} = [e_{ij}]$ is an $m \times n$ matrix whose entries e_{ij} are 0 or 1, $\mathbf{c} = [c_j]$, $j = 1,\ldots,n$, is a cost row with positive components, $\mathbf{x} = [x_j]$, $j = 1,\ldots,n$, is a vector of zero-one variables and $\mathbf{e}$ is an m vector of 1's. If the $\mathbf{Ex} \geq \mathbf{e}$ constraints are replaced by the equalities

$$\mathbf{Ex} = \mathbf{e},$$

the integer program is referred to as a *set partitioning problem.* If we think of the columns of **E** and **e** as sets, the set covering problem is equivalent to finding a cheapest union of sets from **E** that covers every component of **e**, where component i of **e** is covered if at least one of the selected sets (columns) from **E** has a 1 in row i. In the set partitioning, we seek a cheapest union of disjoint sets from **E** which covers **e**. The set covering and the set partitioning problem are representative of numerous real world situations. These include applications in the areas of airline crew scheduling, vehicle routing, stock cutting, map coloring, and other instances.

Detailed formulation and analysis of various other CO problems can be found in Garfinkel and Nemhauser (1972) and Christofides at al. (1979a).

1.2.2 COMPUTATIONAL COMPLEXITY OF CO PROBLEMS

The theory of Computational Complexity has been developed for evaluating and for classifying problems as "hard" or "easy".

Generally, a problem can be solved by a step by step procedure, called *algorithm.* In order to evaluate the running time of an algorithm we count "steps" instead of "machines cycles", and we use an *O-notation* to express the running time function.

The *size* of an instance of a problem is measured by the length of the shortest coding necessary to specify the data completely. Given an instance of size n and a real function $g(n)$ of n, we say that the complexity of an algorithm is $O(g(n))$, if $f(n)$, the maximum time required to execute the algorithm is such that:

$$|f(n)| < c|g(n)|$$

where c is a constant. The precise value of c depends on the computer used. Algorithms with complexity $O(n)$ are called *linear*; those with complexity $O(n^k)$ are called *polynomial* (e.g. algorithms of complexity $O(n^2)$, $O(n\,log(n))$, etc). On the other hand, an algorithm is called *exponential* if its complexity is not of polynomial order (e.g. algorithms of complexity $O(2^n)$, $O(n!)$, etc).

Problems are classified into four classes based on their computationally complexity: *P*, *NP* (*Non-deterministic Polynomial*), *NP-complete* and *NP-hard*.

Problems for which polynomial time algorithms are known belong to the class *P*. By this definition and intuitively, we can think of *P* as a class of "easy" problems since efficient (polynomial time) algorithms exist for solving this class of problems. The class *NP* encompass all problems in *P* as well as other problems which can be solved by a non-deterministic algorithm in polynomial time.

The concept on a non-deterministic polynomial algorithm can be viewed intuitively as follows. First observe that problems in *NP* are decision or recognition problems: that is, for example, rather than ask for the optimal length of the TSP tour, one may ask "is there a tour of length less than L?". The recognition and the optimization version of a problem are closely related since if we can solve the recognition version of a problem, we can also solve its related optimization version. Now imagine a computer which has the property that each time it faces a choice it divides into several copies of itself, with each copy being explored in parallel. The recognition problem is solved if and only if one of the copies answers the recognition problem in the affirmative. If the maximum time taken by a copy is polynomially bounded, then the problem is in *NP*. Another way of looking at this intuitively is to suppose that we could "guess" a solution to the problem, and require that checking the answer could be carried out in polynomial time.

The class *NP-complete* is a subset of *NP* having the property that all problems in *NP* can be reduced in polynomial time to one of them. A problem P1 can be reduced to problem P2 in polynomial time if any instance of P1 can be transformed in polynomial time into an instance of P2, such that the solution of P1 can be obtained in polynomial time from the solution of the instance of P2. A problem is *NP-hard* if every problem in *NP* is polynomially reduceable to it. Whether polynomial time algorithms exist for all problems in *NP* is currently an open question. Figure 1.1 shows the commonly believed relationship among P, NP, NP-complete, and NP-hard problems.

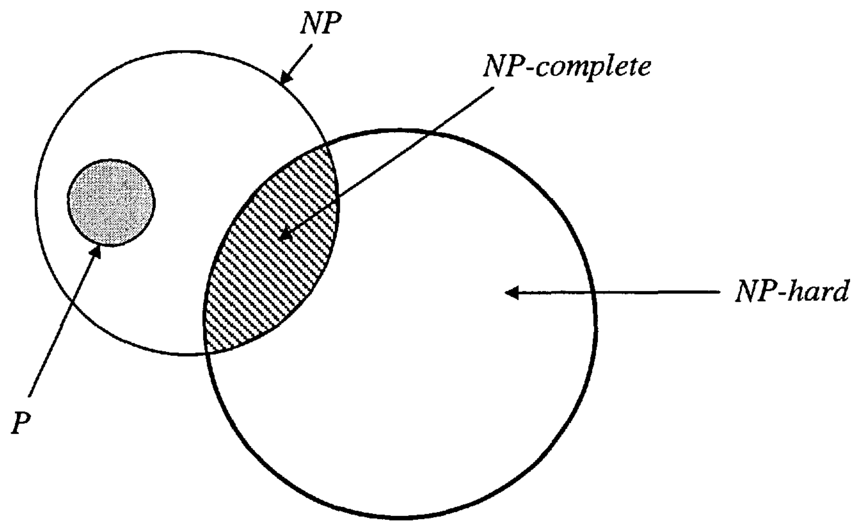


Figure 1.1. Relationship among *P*, *NP*, *NP-complete* and *NP-hard*

For a more complete discussion on computational complexity, the interested reader is referred to Garey and Johnson (1979), Lewis and Papadimitriou (1981) and Papadimitriou and Steiglitz (1982).

The fact that many CO problems are *NP-hard*, suggests that there is no guarantee that an optimal solution will be found in a reasonable amount of computing time. Thus, algorithms can be classified into two broad categories:

(i) *exact solution algorithms* that guarantee an optimal solution at the possible expense of high computational time and memory requirements, thus possibly allowing only small-size instances to be solved;

(ii) *heuristic algorithms* that produce a feasible solution in a reasonable amount of computing time with the risk that it may be sub-optimal.

For the detailed discussion about heuristic techniques for CO problems one can refer to Reeves (1993).

1.3 Methodological preliminaries

In this section we provide a brief description of some known methodologies used in the development of the algorithms presented in the thesis.

1.3.1 Lagrangean Relaxation and Subgradient Optimization

Finding good solutions to hard problems in CO by using an enumerative procedure such as branch and bound method, involves the calculation of upper bounds and lower bounds on the objective function, in order to accelerate the fathoming process and thereby curtail the enumeration. General techniques for generating good upper bounds, in the case of minimization problems, are essentially based on heuristic methods. One of the most efficient techniques for obtaining good lower bounding functions consists of solving the problem obtained by relaxing some of the constraints of the initial problem in the Lagrangean fashion. The use of Lagrangean relaxation in CO originates in the work of Held and Karp (1970, 1971) concerning the TSP.

Consider the following general zero-one problem which we shall refer to as problem P:

$$\text{(P)} \qquad \begin{aligned} \textit{Min} \quad & \mathbf{cx} \\ \textit{subject to} \quad & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{Bx} \geq \mathbf{d} \\ & x_j \in \{0,1\},\ j = 1,\ldots,n \end{aligned}$$

The Lagrangean relaxation of problem P with respect to the constraint set $\mathbf{Ax} \geq \mathbf{b}$ is defined by introducing a Lagrangean multiplier vector $\boldsymbol{\lambda} \geq \mathbf{0}$ which is attached to this constraint set and brought into the objective function to give the following problem called *Lagrangean lower bound program* (LLBP):

$$\text{(LLBP)} \qquad \begin{aligned} \textit{Min} \quad & \mathbf{cx} + \boldsymbol{\lambda}(\mathbf{b} - \mathbf{Ax}) \\ \textit{subject to} \quad & \mathbf{Bx} \geq \mathbf{d} \\ & x_j \in \{0,1\},\ j = 1,\ldots,n \end{aligned}$$

It can be easily shown that problem LLBP provides a lower bound on the optimal solution to the original problem P for any $\boldsymbol{\lambda} \geq \mathbf{0}$. Notice that we are interested in finding the value for the multipliers that give the maximum lower bound, i.e. the lower bound that is as close as possible to the value of the optimal integer solution. This involves finding multipliers which correspond to the following maximization problem called the *Lagrangean dual program*:

$$\underset{\lambda \geq 0}{Max} \left\{ \begin{array}{rl} Min & \mathbf{cx} + \lambda(\mathbf{b} - \mathbf{Ax}) \\ subject\ to & \mathbf{Bx} \geq \mathbf{d} \\ & x_j \in \{0,1\}, j = 1,\ldots,n \end{array} \right\}$$

Applications of Lagrangean relaxation to CO problems can be found in Geoffrion (1974), Fisher (1981) and Beasley (1993).

One approach to deciding values for the Lagrangean multipliers $\{\lambda_i\}$ is to use *subgradient optimization*. Subgradient optimization is an iterative procedure which, starting from an initial set of Lagrangean multipliers, modifies them in a systematic fashion. It can be viewed as a procedure which attempts to maximize the lower bound value derived from LLBP (i.e. to solve the Lagrangean dual program) by a suitable choice of multipliers.

Consider the relaxed constraints in summation notation, that is:

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

The basic subgradient optimization procedure is as follows.

Algorithm 1.1: Subgradient optimization

Step 1. Let π be a user-defined parameter satisfying $0 < \pi \leq 2$.

Let z_{UB} be an upper bound to the optimal solution cost of problem P.

Decide upon an initial set $\{\lambda_i\}$ of multipliers.

Let $z_{max}(= -\infty)$ be the maximum lower bound found.

Step 2. Solve LLBP with the current set of multipliers $\{\lambda_i\}$ to get a solution $\{x_j\}$ of value z_{LB}. Set $z_{max} = max(z_{max}, z_{LB})$.

Step 3. Define *subgradients* G_i for the relaxed constraints, evaluated at the current solution, by:

$$G_i = b_i - \sum_{j=1}^{n} a_{ij} x_j, i = 1,\ldots,m$$

Step 4. Define a (scalar) step size T by

$$T = \frac{\pi(z_{UB} - z_{LB})}{\sum_{i=1}^{m} G_i^2}$$

Step 5. Update λ_i using

$$\lambda_i = max(0, \lambda_i + TG_i), \quad i = 1, \ldots, m$$

and go to Step 2 to resolve LLBP with this new set of multipliers.

Generally, the termination rule is based either upon limiting the number of iterations that can be done, or upon the value of π where π is reduced during the course of the procedure. At the end of the procedure, z_{max} represents the best lower bound found.

For more information on this subject, the interested reader is referred to Held et al. (1974), Sandi (1979) and Beasley (1993).

1.3.2 BRANCH AND BOUND METHODS

Enumerative (*branch and bound*, implicit enumeration) methods solve a CO problem by breaking up its feasible set into successively smaller subsets. The origins of the branch and bound idea go back to the work of Land and Doig (1960).

The basic principle of a branch and bound method is the partition of an initial problem P_0 into a number of subproblems $P_1, P_2, \ldots, P_k$, whose totality represent problem P_0. Each one of these subproblems is resolved separately by:

either (i) finding its optimal solution,

or (ii) showing that the value of the optimal solution to the subproblem is worse than the best solution for the original problem P_0 obtained so far,

or (iii) showing that the subproblem is infeasible.

Partitioning a problem P_0 into a number of subproblems allows easier problems to be resolved, either because of their smaller size, or because of their structure which may not be shared by the initial problem P_0. However, in general, a subproblem which is difficult to resolve, can be further partitioned into yet smaller subproblems $P_{i_1}, P_{i_2}, \ldots, P_{i_r}$. This partitioning, (also called *branching*), can be repeated for subproblems at different levels. An example is shown in Figure 1.2.

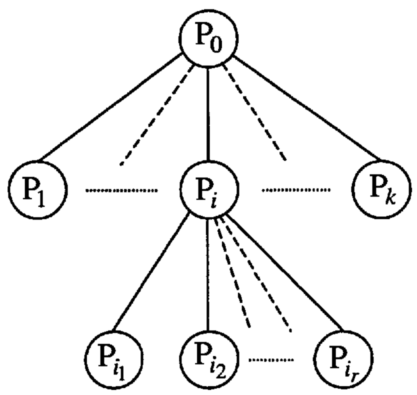


Figure 1.2. A branch and bound tree

Branch and bound methods make use of bounds on the objective function in order to discard certain subproblems from further consideration and thereby curtail the enumeration. The bounds are obtained by replacing the problem over a given subset with an easier (relaxed) problem. The branch and bound procedure ends when each subproblem has been resolved. The best solution found during the procedure is a global optimum.

For any problem P, let $v(\mathrm{P})$ be the value of an optimal solution to P. The essential ingredients of any branch and bound procedure applied to a CO problem P of the form $Min\{f(\mathbf{x}) \mid \mathbf{x} \in S\}$ are:

(i) a relaxation of P, i.e. a problem R of the form $Min\{g(\mathbf{x}) \mid \mathbf{x} \in T\}$, such that $S \subseteq T$ and for every $\mathbf{x}, \mathbf{y} \in S$, $f(\mathbf{x}) < f(\mathbf{y})$ implies $g(\mathbf{x}) < g(\mathbf{y})$;

(ii) a branching or separation rule, i.e. a rule for breaking up the feasible set $\{\mathrm{P}_i\}$ of the current subproblem P_i into subsets $\{\mathrm{P}_{i_1}\}, \{\mathrm{P}_{i_2}\}, \ldots, \{\mathrm{P}_{i_r}\}$ such that $\bigcup_{j=1}^{r} \{\mathrm{P}_{i_j}\} = \{\mathrm{P}_i\}$, where $\{\mathrm{P}\}$ is used to represent the set of all feasible solutions to problem P;

(iii) a lower bounding procedure, i.e. a procedure for finding (or approximating from below) $v(\mathrm{R}_i)$ for the relaxation R_i of each subproblem P_i; and

(iv) a subproblem selection rule, i.e. for choosing the next subproblem to be processed.

Additional ingredients, not always present but always useful when present, are:

(v) an upper bounding procedure, i.e. a heuristic for finding feasible solution to P; and

(vi) a testing procedure, whereby it is possible to fix the values of some variables (reduction, variable fixing) or to discard an entire subproblem (dominance tests) using the logical implications of the constraints and bounds.

The general branch and bound procedure for solving a given problem P, can be described as follows:

Algorithm 1.2: Branch and bound

Step 1. (*Initialization*) Put P on the list of active subproblems. Initialize the upper bound at $z_{UB} = \infty$.

Step 2. (*Subproblem selection*) If the list is empty, stop: the solution associated with z_{UB} is optimal (or, if $z_{UB} = \infty$, P has no solution). Otherwise choose a subproblem P_i according to the subproblem selection rule and remove P_i from the list.

Step 3. (*Lower bounding*) Solve the relaxation R_i of P_i or bound $v(R_i)$ from below, and let z_{LB_i} be the value obtained.

If $z_{LB_i} \geq z_{UB}$, go to Step 2.

If $z_{LB_i} < z_{UB}$ and the solution is a feasible a solution for P, store it in place of the previous best solution, set $z_{UB} = z_{LB_i}$ and go to Step 5.

Step 4. (*Upper bounding: optional*) Use a heuristic to find a solution for P. If a better solution is found than the current best, store it in place of the latter and update z_{UB}.

Step 5. (*Reduction: optional*) Apply variable fixing and dominance tests.

Step 6. (*Branching*) Apply the branching rule to P_i, i.e. generate new subproblems $P_{i_1}, P_{i_2}, \ldots, P_{i_r}$, place them on the list, and go to Step 2.

For general surveys on branch and bound methods see Garfinkel and Nemhauser (1972, Ch. 4), Balas (1975), Garfinkel (1979), Spielberg (1979) and Balas and Toth (1985).

1.3.3 BRANCH AND CUT METHODS

Branch and cut methods combine cutting-planes and search-tree methods to solve integer programs to optimality. The problem to be solved is first formulated as an integer program on a subset of some linear space $\mathbb{R}^n$. Then this problem may be solved using a linear programming relaxation embedded in a branch and bound technique to get integer solutions. The cutting-plane phase and the enumeration phase are integrated in a branch and cut algorithm and new information about the known partial linear description of the polytope associated to the problem can be exploited during the enumeration phase. This approach gives dramatic savings, both in terms of time and memory allocation, compared to a standard branch and bound incorporating linear programming relaxation procedures.

The branch and cut method was introduced by Padberg and Rinaldi (1991) to solve large instances of the TSP. In the procedure of Padberg and Rinaldi the only cutting planes that are used correspond to inequalities that are valid for the polytope associated to the problem, preferably facets. In contrast with many previous algorithms which solve the identification problem only when the solution of the current relaxation is integral, they use identification procedures even if the current solution is fractional and the identification of the cutting planes is tried at all the nodes of the search tree.

The first phase of a branch and cut algorithm is simply a cutting plane approach. If one reaches the point when the optimal solution for the current linear programming relaxation is not feasible for the integer program and no more cutting planes can be identified, then one must start branching and creating a search-tree. A branching procedure is executed, but at each node of the search-tree one must keep trying to identify more cutting planes before trying to branch again. If the cutting planes that are used correspond to valid inequalities for the polytope associated with the problem, they are *globally* valid, i.e. across the entire search-tree, and can be kept in the linear programming relaxation of the problem.

The core of each branch and cut method is the cutting planes procedure used for generating lower bounds. Consider the general ILP formulation as being of the following form:

$$
\text{(P)} \qquad \begin{aligned} \mathit{Min}\quad & \mathbf{cx} \\ \mathit{subject\ to}\quad & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{Bx} \geq \mathbf{d} \\ & \mathbf{x} \geq \mathbf{0} \text{ integer} \end{aligned}
$$

A possible relaxation of this problem is given by:

$$
\text{(LP)} \qquad \begin{aligned} \mathit{Min}\quad & \mathbf{cx} \\ \mathit{subject\ to}\quad & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}
$$

A valid lower bound to P can be generated using the following procedure.

Algorithm 1.3: Cutting planes

Step 1. Initialize the LP-relaxation constraint set with $\mathbf{Ax} \geq \mathbf{b}$.

Step 2. Solve problem LP and let $\overline{\mathbf{x}}$ be its solution.

Step 3. If $\overline{\mathbf{x}}$ satisfies constraints $\mathbf{Bx} \geq \mathbf{d}$ of problem P and is integral, then solution is also an optimal solution of P, Stop. Otherwise proceed to Step 4.

Step 4. Find one or more valid inequalities that are violated by $\overline{\mathbf{x}}$.

Step 5. If none is found, Stop. Otherwise add the violated inequalities to problem LP and go to Step 2.

The above procedure terminates when no further valid inequalities can be found or an optimal solution has been found. The problem solved in Step 4 is called the *separation problem*. The separation problem can be solved by an *exact* procedure or a *heuristic* procedure that may find violated inequalities, but that in case it cannot find any, is unable to guarantee that no violated inequalities exist.

When embedded in a branch and bound procedure, a check that the lower bound generated at Step 2 is less then the best upper bound must be included. If the lower bound is at least equal to the best known upper bound then the subproblem is fathomed. Step 3 is modified so that the integral solution becomes the best known solution and the upper bound is set accordingly.

1.4 RESEARCH MOTIVES AND GOALS

This thesis is based on the development of new exact methods for the CPMP and the VRP. To this end, we wish to focus our attention on mathematical models which can be easily extended to deal with a wide range of constraints without changing the original nature of the model. We have chosen the Set Partitioning (SP) approach to model these problems since adding a few additional constraint to the original SP formulation does not change the original structure. In the CPMP each column of SP represents a feasible cluster for a given median and the additional constraint limits the number of clusters in any feasible solution. In the VRP, each column is a feasible route and it is sufficient to add an extra constraint to limit the number of routes of the solution. In both cases, practical cluster constraints in the CPMP (route constraints in the VRP) can be easily incorporated by removing from the SP model the infeasible clusters (routes). The resulting SP problem cannot be solved directly since the number of variables can be too large but it can be used to compute a lower bound without generating the entire SP matrix as it has been proposed by Mingozzi et al. (1994) for the basic VRP. This method combines in an additive manner dual ascent procedures that explore different relaxations of the problem in order to compute a dual solution of the LP-relaxation of the SP model. One of the procedures proposed allows to deal with any practical route constraint. Our goal is then to extend this technique for solving to optimality complex location and routing problems and to derive new exact methods for the CPMP and the VRP with Backhauls that are competitive with the exact methods already proposed in the literature.

The detailed study of the VRP literature inspired us to further investigate the two commodity network flow formulation of the TSP proposed by Finke et al. (1984). This formulation is interesting in different ways. It can be shown that its LP-relaxation satisfies a weak form of the subtour elimination constraints. As part of this research, we will examine several ideas to modify the original TSP formulation of Finke et al. in order to incorporate additional constraints for the TSP. This will lead to the design of new exact methods for TSP and VRP with additional constraints. We hope to demonstrate that the new exact methods are competitive with, if not better than, other exact methods proposed in the literature for the same problems.

As far as new heuristic methods is concerned, we will concentrate our attention on techniques which can be applied to a wide range of problems. The Bionomic algorithm proposed by Christofides (1994) is a new entry to operations research and, in our opinion, provides an important generalisation and improvement of genetic based techniques. It allows a better exploration and exploitation of the search space to be performed, is robust and less problem specific and it is capable of generating good solutions within reasonable computational times. Our main goal in this research is to design a Bionomic algorithm for the CPMP that can be used also for solving problems with additional constraints. We aim to demonstrate that the new heuristic technique is computationally competitive with other sophisticated heuristic methods.

Finally, a case-study will examine the use of heuristic methods, specifically designed for routing/location problems, in providing high quality solutions to real-life problems. For this purpose, we will consider the resource planning problem of an utility company which provides preventive maintenance services to a set of customers using a fleet of mobile gangs based at some depots. Our goal is to design a heuristic algorithm for this problem and to test the usefulness of the method by applying it to a real case.

1.5 THESIS OVERVIEW

This section provides a detailed overview of the thesis.

In Chapter 2, a revised version of Maniezzo et al. (1998), we consider the Capacitated p-Median Problem (CPMP) in which a set of n customers must be partitioned into p disjoint clusters so that the total dissimilarity within each cluster is minimized and constraints on maximum cluster capacities are met. The total dissimilarity of a cluster is computed as the sum of the dissimilarities existing between each entity of the cluster and the median associated to the cluster. We describe a heuristic algorithm based on the Bionomic Algorithm as an effective method to solve the CPMP. The chapter also presents an effective local search technique for the CPMP. Computational results show the effectiveness of the proposed approach, when compared to the best performing heuristics so far presented in the literature.

In Chapter 3 we present an exact algorithm for solving the CPMP based on a Set Partitioning formulation of the problem. A valid lower bound to the optimal solution cost is obtained by combining two different heuristic methods for solving the dual of the LP-relaxation of the exact formulation. The computational performance of the new exact algorithm has been evaluated on two classes of test problems proposed in the literature and on two new classes of difficult CPMP instances with additional constraints. The results show that the exact algorithm is able to solve exactly CPMP's of size up to 100 customers.

In Chapter 4 we describe a two-commodity network flow approach to derive new integer programming formulations for different routing problems. The basic Vehicle Routing Problem (VRP) is examined in which a fleet of M vehicles stationed at a central depot is to be optimally routed to supply customers with known demands subject to vehicle capacity constraints. We present a new integer programming formulation for the VRP based on a two-commodity network flow approach. A lower bound based on a linear relaxation of the new formulation strengthened by a set of valid inequalities is derived. The bound is embedded in a branch and cut procedure to solve the problem optimally. The computational results on a set of problem instances derived from the literature show that the lower bound obtained is tight and that the branch and cut algorithm has been able to solve to optimality problems up to 100 customers. We extend the two-commodity network flow approach to derive new integer programming formulations for other routing problem like the TSP with mixed deliveries and collections and the TSP with Backhauls. These formulations are used to derive new lower bounds based on linear relaxation strengthened by new valid inequalities. The resulting cutting plane procedure has been applied to a set of instances taken from the literature and involving problems up to 150 customers. The results show that the branch and cut algorithm has been able to solve to optimality problems up to 150 customers.

In Chapter 5, a revised version of Mingozzi et al. (1999), we consider the Vehicle Routing Problem with Backhauls (VRPB) in which a fleet of vehicles located at a central depot is to be optimally used to serve a set of customers (called *Linehaul* customers) requiring deliveries from the depot and to collect products from a set of customers (called *Backhaul* customers) to be unloaded at the depot. Each route starts and ends at the depot and the Backhaul customers must be visited after the Linehaul customers. A new (0-1) integer programming formulation of this problem is presented. We describe a procedure that computes a valid lower bound to the optimal solution cost by combining different heuristic methods for solving the dual of the LP-relaxation of the exact formulation. An algorithm for the exact solution of the problem is presented. Computational tests on problems proposed in the literature show the effectiveness of the proposed algorithms in solving problems up to 100 customers.

In Chapter 6, a revised version of Hadjiconstantinou and Baldacci (1998), we consider the resource planning problem of a utility company, which provides preventive maintenance services to a set of customers using a fleet of mobile gangs based at some depots. The problem is to determine the boundaries of the geographic areas served by each depot, the list of customers visited each day and the routes followed by the gangs. The objective is to provide improved customer service at minimum operating cost subject to constraints on frequency of visits, service time requirements, customer preferences for visiting on particular days and other routing constraints. The problem has been approached as a Multi-Depot Period Vehicle Routing Problem (MDPVRP) and a heuristic algorithm has been developed to solve it. The computational implementation of the complete planning model is described with reference to a pilot study and results are presented.

Finally, in Chapter 7, we provide a summary of the entire thesis highlighting the main contributions of the completed work. Current limitations and suggestions for further research are also discussed.

CHAPTER 2

A BIONOMIC APPROACH TO THE CAPACITATED *P*-MEDIAN PROBLEM

2.1 INTRODUCTION

The Capacitated p-Median Problem (CPMP) is a particular location problem in which a set of n customers must be partitioned into p disjoint clusters so that the total dissimilarity within each cluster is minimized and constraints on maximum cluster capacities are met. The total dissimilarity of a cluster is computed as the sum of the dissimilarities existing between each customer of the cluster and the median associated to the cluster. This problem, which appears also under the names of the Capacitated Warehouse Location Problem, Sum-of-Stars Clustering Problem and others, is NP-hard (Garey and Johnson (1979)) and has already been extensively studied in clustering and location theory. A number of exact algorithms have been proposed in the literature for the CPMP. Pirkul (1987) describes a branch and bound method which uses the Lagrangean relaxation of the partitioning constraints. An exact technique based on a set partitioning formulation of the CPMP with side constraints has been investigated by Hansen et al. (1994). In Chapter 3 we present a new exact method for solving the CPMP based on the set partitioning approach and we compare its computationally performance with Pirkul's algorithm (which we implemented).

Heuristic algorithms have been proposed by Mulvey and Beck (1984) and Pirkul (1987). Metaheuristic approaches are described in Golden and Skiscim (1986) and in Osman and Christofides (1994).

A problem closely related to the CPMP accepts multiple partial assignments of customers to clusters and give rise to a mixed integer formulation of the problem. For this latter problem, exact algorithms have been proposed by Christofides and Beasley (1983), Leung and Magnanti (1989) and Aardal (1994), while heuristic methods have been investigated by Van Roy (1986) and Beasley (1988).

In this chapter we propose the use of a metaheuristic technique recently presented by Christofides (1994), called Bionomic Algorithm, as a viable method for solving the CPMP. The resulting algorithm integrates the main steps of the Bionomic approach with a Lagrangean-based lower bound to the CPMP.

The chapter is structured as follows. In Section 2.2 we present a classical mathematical formulation of the CPMP and in Section 2.3 we summarize the heuristic and exact methods proposed in the literature. In Section 2.4 we describe a new heuristic algorithm based on the Mulvey and Beck approach. In Section 2.5 we describe the main steps of the Bionomic algorithm, while in Section 2.6 our new heuristic method for the CPMP is presented. Computational results are shown in Section 2.7.

2.2 A MATHEMATICAL FORMULATION OF THE CPMP

Let $N = \{1,\dots,n\}$ be a set of n customers and $\left[d_{ij}\right]$ be a $n \times n$ matrix indicating the dissimilarities between pairs of customers of set N. We assume that $d_{ij} \geq 0$ and $d_{ii} = 0$ for all $i, j \in N$. A positive integer weight q_i is associated with each customer i, $i \in N$.

Any subset $B \subseteq N$ is called a *cluster*. Given a cluster B, the customer $j^* \in B$ such that

$$\sum_{i \in B} d_{ij^*} \leq \sum_{i \in B} d_{ij} \quad \forall j \in B \setminus \{j^*\}$$

is called the *median* of B and will be denoted with $\pi(B)$.

A positive integer weight Q_j is associated with each customer j, $j \in N$, which denotes the capacity of j when it is used as the median of a cluster.

A cluster B is *feasible* if

$$\sum_{i \in B} q_i \leq Q_{\pi(B)},$$

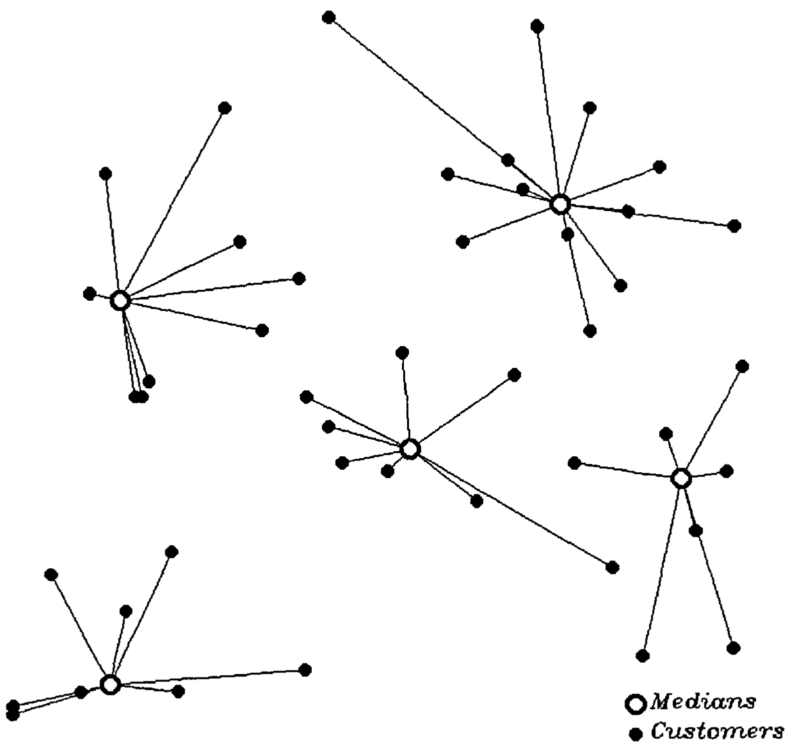


Figure 2.1. Example of a CPMP solution

where $Q_{\pi(B)}$ denotes the capacity of median $\pi(B)$.

For a given integer p, $2 \le p \le n$, a feasible CPMP solution is represented by a partition $S = \{B_1, B_2, \ldots, B_p\}$ of N into p feasible clusters and its cost is given by the sum of the cluster dissimilarities, that is :

$$z(S) = \sum_{\ell=1}^{p}\left(\sum_{i \in B_\ell} d_{i\pi(B_\ell)} \right),$$

where $\pi(B_\ell)$ denotes the median of cluster B_ℓ. An optimal CPMP solution corresponds to a partitioning of the customer set N into p feasible clusters of minimum cost. Figure 2.1 shows an example of a CPMP solution.

Let ξ_{ij} be a (0-1) variable that is one if and only if a customer i is assigned to a cluster whose median is j. We assume that $\xi_{jj} = 1$ means that customer j is chosen to be a median of a cluster. A mathematical formulation of the CPMP is as follows.

(F) $$z(\mathrm{F}) = Min \sum_{i \in N} \sum_{j \in N} d_{ij}\xi_{ij} \qquad (2.1)$$

$$\textit{subject to} \quad \sum_{j\in N} \xi_{ij} = 1, \qquad \forall i \in N \tag{2.2}$$

$$\sum_{i\in N} q_i \xi_{ij} \le Q_j \xi_{jj}, \qquad \forall j \in N \tag{2.3}$$

$$\sum_{j\in N} \xi_{jj} = p \tag{2.4}$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall i, j \in N \tag{2.5}$$

Constraints (2.2) force each customer to be assigned to a cluster, constraints (2.3) impose that the total capacity of a median must not be exceeded, constraint (2.4) specifies that the total number of clusters must be equal to p and constraints (2.5) are the integrality constraints.

Different relaxations of formulation F have been proposed in the literature to derive lower bounds to CPMP. Mulvey and Beck (1984) proposed a Lagrangean relaxation of constraints (2.2), while Beasley (1988) used a Lagrangean relaxation of constraints (2.2), (2.3) and (2.4).

We briefly describe below the lower bound proposed by Mulvey and Beck (1984) and used by Pirkul (1987) to obtain an exact branch and bound method. We will use also this lower bound in the Bionomic heuristic method proposed in Section 2.6.

The Lagrangean relaxation of the assignment constraints (2.2) using multipliers λ_i, $i \in N$, leads to the lower bound LB, which is based on the following formulation LR:

$$\text{(LR)} \qquad \text{LB} = \textit{Min} \quad \sum_{i\in N}\sum_{j\in N} (d_{ij} - \lambda_i)\xi_{ij} + \sum_{i\in N} \lambda_i \tag{2.6}$$

subject to (2.3), (2.4) and (2.5).

The value of LB can be computed as follows.

Let $h_j = \textit{Min}\left\{ \sum_{i\in N} (d_{ij} - \lambda_i) y_i : \text{s.t.} \sum_{i\in N} q_i y_i \le Q_j \text{ and } y_i \in \{0,1\}, i \in N \right\}$ and let $\{h_{j_1}, h_{j_2}, \ldots, h_{j_p}\}$ be the p-least cost values of $\{h_1, h_2, \ldots, h_n\}$, then $\text{LB} = \sum_{k=1}^{p} h_{j_k} + \sum_{i\in N} \lambda_i$. A classical subgradient optimization technique can be used to maximize the value of the lower bound. We use procedure MT2 of Martello and Toth (1990) for solving n knapsack problems at each subgradient iteration. The order of complexity, C(LR), of each lower bound iteration is equal to the sum of the complexity of procedure MT2 plus $O(n \log n)$ which is the time required for ordering the n knapsack values. Procedure MT2 is a branch and bound method which is very fast, requiring on average a few hundredth of seconds, to solve any of our knapsack instances.

2.3 A LITERATURE REVIEW FOR THE CPMP

In this section we briefly describe the heuristic and exact algorithms proposed in the literature to solve the CPMP.

2.3.1 HEURISTIC ALGORITHMS FOR THE CPMP

In this section we outline the heuristic algorithms for the CPMP proposed by Mulvey and Beck (1984) and by Osman and Christofides (1994), against which the new Bionomic approach is compared.

2.3.1.1 MULVEY AND BECK'S HEURISTICS

Mulvey and Beck (1984) proposed two related heuristic algorithms for the CPMP. The first one (hereafter called MB1) aims at minimizing the total customer assignment *regret*, where the regret of the assignment of a customer is defined to be the absolute value of the difference in dissimilarity between the customer's first and second nearest medians. MB1 starts by randomly generating p medians and assigning customers to them in an order specified by decreasing regrets. When (and if) all customers are assigned, that is, customers are clustered around the respective medians, an intra-cluster phase re-assigns each cluster to the median that minimizes the sum of dissimilarities between the specific median and all other cluster members. Possibly, a new set of medians is identified, in this case the assignment/re-assignment process is repeated. When the medians remain stable across iterations, pairwise interchanges of customers between clusters are used to optimize the solution locally. MB1 simply repeats the above process for a predetermined number of iterations.

The second heuristic (hereafter called MB2) is a modification of MB1. It is based on the subgradient optimization of problem LR described in Section 2.2. The starting set of medians, randomly selected at each iteration in MB1, is substituted in MB2 by the median set identified by the corresponding subgradient iteration. The following phase of assignment of customers to medians is the same as in MB1. MB2 terminates either after a predefined number of iterations or when the difference between the lower and upper bound is less than a given threshold.

2.3.1.2 Osman and Christofides' heuristic

Osman and Christofides (1994) presented a heuristic algorithm for the CPMP, hereafter called OC, which is based on a hybrid Simulated Annealing (SA) / Tabu Search (TS) metaheuristic technique. The essential features of this technique are drawn from the probabilistic acceptance of solutions of SA and the neighborhood exploration of TS. Specifically, the probabilistic SA acceptance is combined with three TS-derived features. The first is a non monotonic cooling schedule that occasionally increases the temperature, in order to escape from local optima but without starting the search from scratch. The second is a systematic neighborhood search, as opposed to the random exploration that is typical of SA. The third is the terminating condition, which is not based on the number of iterations, as it is usually the case in TS, but on the number of temperature resets performed without improving the best solution.

This algorithm has been applied to a variety of combinatorial optimization problems, consistently yielding improved performance over standard SA (Osman and Laporte (1996)). In particular, its application to the CPMP, containing a specific local optimization technique results in a very effective heuristic.

2.3.2 Exact algorithms for the CPMP

Exact methods for solving the CPMP have been proposed by Pirkul (1987) and by Hansen et al. (1994).

Pirkul (1987) describes two heuristic methods and an exact branch and bound method for the Capacitated Concentrator Location Problem. The problem arises in the topological design of computer communication networks and deals with the design process of dividing network nodes into groups, and selecting a concentrator location for each group so that all the nodes in a group can be assigned to the same concentrator without violating its capacity constraint. Pirkul makes use of the Lagrangian relaxation approach to develop optimal and heuristic solution procedures for the problem. The exact method is based on a branch and bound procedure which uses the Lagrangean relaxation of the partitioning constraints (2.2). In Chapter 3 we computationally compare Pirkul's algorithm (which we implemented) with our new exact algorithm. A detailed description of Pirkul's algorithm can be found in Section 3.5.2.

Hansen et al. (1994) proposed an exact technique based on a set partitioning with side constraints formulation of the CPMP. The algorithm combines the column generation technique of linear programming with branch and bound. Column generation was originally proposed by Gilmore and Gomory (1961). This technique extends the revised simplex algorithm of linear programming and allows the solution of linear programs with an extremely large number of columns by determining the entering column using the solution of an auxiliary combinatorial problem. This last problem depends on the type of problem considered. For the CPMP, the entering column of the linear program corresponding to each node in the branch and bound tree, is determined by solving a *knapsack problem with incompatibilities* for which a specific algorithm is proposed. Hansen et al. made a theoretical comparison between the lower bounds obtained by column generation and by Lagrangean relaxation (see Section 3.3.1). The computational results show that the overall algorithm allows solution of medium-sized problems (with number of customers n=75 or 81).

2.4 THE HEURISTIC PROCEDURE HEUMED

In this section we describe a new heuristic, called HEUMED, which is based on the Mulvey and Beck approach. HEUMED is an iterative multistart procedure where, at each iteration, a new set of p medians is randomly generated. Given this set of medians, the algorithm loops over two phases.

In the first phase the customers are assigned to the medians selected by solving a Generalized Assignment Problem (GAP). The second phase uses the clustering resulting from the first phase and tries to find a better median for each cluster, by solving an assignment problem. If a better solution is found, then both phases are iterated.
A step-by-step description of HEUMED is given below.

Algorithm 2.1: HEUMED

```
Step 0.  Initialization
         Set $\tilde{z}_P = \infty$ and $t = 1$.
Step 1.  Iteration t
         Randomly generate an initial set of $p$ medians $J'$.
Step 2.  Phase 1 (GAP)
         Assign the $n$ customers $N$ to the medians in $J'$ by solving a GAP that is
         obtained from problem F by removing constraint (2.4) and by setting $\xi_{jj} = 1$ if
         $j \in J'$, 0 otherwise, $\forall j \in N$.
         Let $\xi'$ be the GAP solution of cost $z'_P$.
         Update $\tilde{z}_P = Min[\tilde{z}_P, z'_P]$.
         Let $C_k = \{i \mid i \in N \text{ and } \xi'_{ik} = 1\}$ be the subset of customers assigned to median
         $k \in J'$ in the GAP solution.
Step3.   Phase 2 (Local improvements)
         Let $c_{kj} = \sum_{i \in C_k} d_{ij}$ be the cost of assigning cluster $C_k$ to median $j \in N$.
         Solve the Assignment Problem (AP) on matrix $[c_{kj}]$ and let $\mathbf{x}^*$ be an optimal
         AP solution of cost $z^*_{\text{AP}}$ (we assume $x^*_{kj} = 1$ if cluster $C_k$ is assigned to
         median $j \in N$ and $x^*_{kj} = 0$ otherwise).
         Update $\tilde{z}_P = Min[\tilde{z}_P, z^*_{\text{AP}}]$.
         If $\tilde{z}_P < z'_P$ then set $J' = \left\{ j \mid j \in N \text{ and } \sum_{k=1}^{p} x^*_{kj} = 1 \right\}$, return to Step 2.
Step 4.  Termination condition
```

Set t=t+1; if t does not exceed an a-priori fixed number of iterations go to Step 1, otherwise stop.

We use the heuristic algorithm MTHG of Martello and Toth (1990) for solving the GAP in Step 2. Each HEUMED iteration requires $O(np \log p + n^2)$ time (Step 2) for solving the GAP and $O(n^2)$ time (Step 3) for solving the assignment problem. Hence, the overall time complexity is $O(np \log p + n^2)$.

2.5 OUTLINE OF BIONOMIC ALGORITHMS

Bionomic Algorithms (BAs), introduced by Christofides (1994), are a class of metaheuristic techniques that provide the main steps for a global optimization method, which must be completed and specified in a way tailored to the particular optimization problem one has to solve.

Bionomic algorithms are closely related to other optimization techniques already presented in the literature. In particular, they share the core of their approach with Genetic Algorithms (GAs) (see Holland (1975) and Goldberg (1989)) and Evolution Strategies (ES) (see Rechenberg (1973) and Bäck et al. (1991)). BAs, GAs and ESs are in fact evolutionary metaheuristic algorithms that update a whole population of solutions (the solution set) at each iteration. Moreover, the updating process in all of them consists of defining a child solution from a set of parent solutions of the previous generation, where the exact definition of the child often goes through some randomization step. Within this general framework, the BA shares with the evolutionary scatter search approach of Glover (1977) (see also Glover (1997)), the possibility of having variable-sized solution sets and the use of multiple parents, whereas GA limit the number of parents to two (a recent version of ES allows a random sampling of the population to select more than two parents). On the other side, the BA formally requires a local optimization of the solutions (called *maturation*), an activity first introduced in the scatter search approach that was excluded from GAs until the late-1980s, though it has now become standard practice in GAs applied to combinatorial optimization problems. We assume, for convenience, that the local optima produced by the maturation step are all distinct, although it is possible that different source solutions will be improved to yield the same local optimum.

The steps specific to BAs are those for defining a parent set. Both GAs and ESs in fact, essentially let the user free to decide how to define the parent sets and the standard practice is to choose them randomly from the population (certain forms of GAs bias the randomization to favor higher quality solution in selecting one or both of the parents). The BA instead, defines a procedure based on the identification of maximal independent sets of a graph defined on the solution set. Such an approach constitutes a refinement of the scatter search proposal of generating parents with reference to clustering strategies, and explicitly introduces a special diversification criterion into the selection of parents. This aspect, together with the generality of the method used for generating child solutions, make BAs well-suited to combinatorial optimization. On the other hand, GAs can hardly exploit the structure and the properties of the solutions and ESs are ill-adapted since they are directed towards continuous spaces.

The structure of BA is as follows.

Let $g = 1,\dots, g_{max}$ be the index of generations, $s = 1,\dots,s_g$ be the index of solutions in generation g, $\chi^{gs} = (\chi^{gs}(1),\dots,\chi^{gs}(n))$ be an n-dimensional 0-1 vector representing solution s of generation g and, finally, let $z(\chi^{gs})$ be the evaluation function of a solution χ^{gs}. Given two solutions χ^{gk} and χ^{gh}, the Hamming distance between χ^{gk} and χ^{gh} is defined as the number $\sum_{i=1}^{n} |\chi^{gk}(i) - \chi^{gh}(i)|$.

The BA algorithm goes through the following five steps.

Algorithm 2.2: Bionomic Algorithm

Step 1. Initialization

1.1 Set $g = 1$.

1.2 Choose s_1 (number of solutions in generation 1).

1.3 Create s_1 distinct initial feasible solutions (randomly or using a heuristic).

1.4 Let $\hat{\mathbf{X}}^1 = \left\{\hat{\chi}^{1s} \mid s = 1,\ldots,s_1\right\}$ be the solution set.

Step 2. *Maturation*

2.1 Improve each solution in $\hat{\mathbf{X}}^g$ individually by a process based on a local optimization.

2.2 Let χ^{gs} be the local optimum associated to $\hat{\chi}^{gs}$ and let $\mathbf{X}^g$ be the set of local optima derived from $\hat{\mathbf{X}}^g$.

2.3 Let $z(\chi^{gs})$ be the cost of solution $\chi^{gs} \in \mathbf{X}^g$.

Step 3. *Propagation, definition of parents sets*

3.1 Allocate *frequency of inclusion* θ_{gs} to each solution χ^{gs} by mapping $z(\chi^{gs})$ onto a suitable positive integer value.

3.2 Choose a positive Hamming distance Δ. Generate the solution adjacency graph $G(\mathbf{X}^g)$ by considering adjacent any two solutions whose Hamming distance is not greater than Δ.

3.3 Generate the r^{th} parent set P_{gr} as a maximal independent set of $G(\mathbf{X}^g)$ (there may be many such sets, but we seek only one).

3.4 Update $\theta_{gs} = \theta_{gs} - 1$, $\forall \chi^{gs} \in P_{gr}$. If $\theta_{gs} = 0$ for some χ^{gs}, remove the corresponding vertex from $G(\mathbf{X}^g)$ (in general, to assure a vertex will be removed, θ_{gs} may be reduced by the minimum positive value over the set rather than by 1).

3.5 Repeat steps 3.3 and 3.4 to generate the next parent set until $G(\mathbf{X}^g)$ is null or is a complete graph (which implies P_{gr} is null). Let $\mathcal{P}_g = \{P_{gr} \mid r = 1,\ldots,r_g\}$ be the family of the generated parent sets.

Step 4. *Propagation, definition of child solutions*

4.1 Let $\pi(S,\varepsilon)$ be a many-to one mapping of $S \subseteq \mathbf{X}^g$ to a solution $\chi \in \hat{\mathbf{X}}^{g+1}$, where ε is a random vector that affects the mapping. The new generation is then $\hat{\mathbf{X}}^{g+1} = \left\{\chi \mid \pi\left(P_{gr}, \varepsilon_j\right), r = 1,\ldots,r_g, j = 1,\ldots,\eta_r\right\}$, where η_r is the number of offspring of parent set r.

Step 5. Termination

5.1 Repeat steps 2 to 4 until the generation limit is reached.

5.2 Choose the best solution found as the answer.

The parameters of the algorithm are: $g_{\max}$, the number of generations, s_1, the number of elements of the first population, Δ, the Hamming distance to be used as a threshold for the definition of the adjacency graph G used in Step 3, and η_r, the number of offspring of each parent set. Moreover, to complete the algorithm, it is necessary to specify the frequency of inclusion function, that maps the values $z(\chi^{gs})$ onto the corresponding θ_{gs}, and $\pi(S,\varepsilon)$, i.e., how to obtain child solutions from parent sets. Moreover, the maturation phase includes a local optimization-based procedure, such as steepest descent, tabu search, simulated annealing or any other method which has to be detailed for the specific problem.

2.6 A BIONOMIC ALGORITHM FOR THE CPMP

This section describes the BA we developed and implemented for the CPMP. A CPMP solution s of generation g is denoted by $\chi^{gs} = \left(\xi^{gs}\right)$, where ξ^{gs} is an $n \times n$ dimensional (0-1) vector representing a feasible solution of problem F.

The evaluation function $z(\chi^{gs})$ is computed as $z(\chi^{gs}) = \sum_{i \in N} \sum_{j \in N} d_{ij} \xi_{ij}^{gs}$.

Following the description of the BA presented in Section 2.5, we have the following algorithm.

Algorithm 2.3: Bionomic Algorithm for the CPMP

Step 1. Initialization

Set $g = 1$ and $s_1 = 100$.

Create s_1 feasible solutions as follows.

i) Randomly select a subset J' of p medians from set N and set $\xi_{jj}^{1s} = 1, j \in J'$ and $\xi_{jj}^{1s} = 0, j \in N \backslash J'$.

ii) Assign the customers N to medians J' by solving the corresponding GAP and set $\xi_{ij}^{1s} = 1$, if customer $i \in N$ has been assigned to $j \in J'$, $\xi_{ij}^{1s} = 0$ otherwise.

Let $\hat{\mathfrak{X}}^{g} = \left\{\xi^{gs} \mid s = 1, \ldots, s_g\right\}$ be the solution set of the first generation.

Step 2. *Maturation*

Improve each solution in $\hat{\mathfrak{X}}^{g}$ individually by applying steps 2 and 3 of algorithm HEUMED as described in Section 2.4.

Denote with $\mathfrak{X}^{g}$ the set of local optima derived from $\hat{\mathfrak{X}}^{g}$ and by $z(\chi^{gs})$ the cost of solution $\chi^{gs} \in \mathfrak{X}^{g}$.

Step 3. *Propagation, definition of parents sets*

3.1 Define the frequency of inclusion values for each θ_{gs} for each solution χ^{gs} by ranking the population solutions of set $\mathfrak{X}^{g}$ in decreasing values of $z(\chi^{gs})$ and setting $\theta_{gs} = \left\lceil \frac{rank\left(x^{gs}\right)}{5} \right\rceil$.

Steps 3.2 to 3.5 are the same as in the BA algorithm of Section 2.5, except that we only consider $r_g \leq r_{\max}, \forall g$, where $r_{\max}$ is a system parameter.

Step 4. *Propagation, definition of child solutions*

For each parent set $P_{gr} \in \mathcal{P}_g$, denote by J' the subset of medians used in the solution belonging to P_{gr}, that is: $J' = \left\{j \mid \xi_{jj}^{gs} = 1, \forall \xi^{gs} \in P_{gr}\right\}$.

Compute the lower bound LB by setting $\xi_{jj} = 0$, $\forall j \in N \setminus J'$ in problem LR and select the η_r best different solutions produced by the subgradient optimization method used for computing LB (see equation (2.6)).

Each solution proposed by the lower bound consists of a set $\tilde{J}$ of p medians and of an assignment of the customers to such medians. The assignment may be infeasible due to the relaxation of constraints (2.2). To construct a feasible CPMP solution χ^{g+1s} we proceed as follows:

- A customer i is assigned to a median $j \in \tilde{J}$ if in the bound solution it was assigned only to j, the median capacity is accordingly decreased. Let $\tilde{N} \subseteq N$ be the subset of customers that in the bound solution were either not assigned to any median or assigned to more than one median. Assign $\tilde{N}$ to $\tilde{J}$ by solving the corresponding GAP.

Step 5. Termination

5.1 Repeat steps 2 to 4 until the generation limit is reached.

5.2 Choose the best solution found as the answer.

In the computational results shown in Section 2.7, we used the following setting of the parameters: $s_1 = 100$, $g_{\max} = 10$, $r_{\max} = 200$, $\eta_r = 1, \forall r$. To compute Δ we calculated the average d_{AVG} and the standard deviations d_{STD} of the Hamming distances between each pair of solutions in $\mathbb{X}^g$, and we set $\Delta = d_{AVG} - 0.7 d_{STD}$. In Step 4 we performed 20 subgradient iterations to compute bound LB on each parent set P_{gr}.

For each generation g of algorithm BA, Step 2 requires $O\left(\left|\hat{\mathbb{X}}^g\right|\left(np\log p + n^2\right)\right)$ time, Step 3 requires $O\left(r_g\left|\hat{\mathbb{X}}^g\right|\log\left|\hat{\mathbb{X}}^g\right|\right)$ time and Step 4 requires $O\left(r_g\left(C(\mathrm{LR}) + \eta_r\left(np\,log\,p + n^2\right)\right)\right)$ time. Hence the overall time complexity of the BA for a generic generation g is

$$O\left(\left|\hat{\mathbb{X}}^g\right|\left(\left(np\log p + n^2\right) + r_g\log\left|\hat{\mathbb{X}}^g\right|\right) + r_g\left(C(\mathrm{LR}) + \eta_r\left(np\log p + n^2\right)\right)\right).$$

Since $r_g \le r_{\max}$ and, $\left|\hat{\mathbb{X}}^g\right| \le r_{\max}\eta_r, \forall g$, the order of complexity is $O\left(np\log p + n^2 + C(\mathrm{LR})\right)$. Moreover, since $p << n$, the overall complexity becomes $O\left(n^2 + C(\mathrm{LR})\right)$.

2.7 COMPUTATIONAL RESULTS

In order to provide computational results and to validate the algorithm proposed we used five classes of problems, called class A, B, C, D and E, respectively. The first and second classes, A and B, consist of the 20 problem instances used by Osman and Christofides (1994); class A contains 10 problems of size n=50 and p=5 while class B contains 10 problems of size n=100 and p=10. In these two classes of problems the dissimilarity matrices correspond to Euclidean distance matrices.

As it is known (Aardal (1994)) that instances with cost randomly generated in the unit square are in general more difficult than instances with costs representing the Euclidean distances between points randomly generated in the unit square (the latter is the case of the Osman and Christofides' problems), we generated three other classes of problems with random costs.

Problem classes C and D contain 10 symmetric instances each, of size n=50 and p=5 for class C and n=100 and p=10 for class D.

The last class of problems, E, contains 10 asymmetric instances of size n =50 and p=5.

For problem classes C, D, and E the values of the dissimilarity matrix $[d_{ij}]$ are integers randomly generated in the interval [1,200], the customer weights are integers randomly generated in the interval [1,50] while the median capacities were computed as follows:

$$Q_j = \frac{\sum_{i \in N} q_i}{p(0.82 + rand(1) \cdot 0.14)}, \; j = 1,\ldots,n$$

where $rand(1)$ indicates a random number generated with uniform probability density on the interval [0,1].

The algorithms were coded in Fortran 77 and run on a IBM PC equipped with a Pentium 166 MHz CPU. Most instances of the five classes of problems have been solved to optimality by means of the exact branch and bound procedure described in the next chapter (see Section 3.5.2).

In tables 2.1 to 2.5, we compare the results obtained by algorithms BA and HEUMED to the optimal or best known solution cost, to the best solution obtained by the Osman and Christofides' algorithm and to the best solution obtained by our implementation of the two versions of the Mulvey and Beck heuristics, z_{MB1} and z_{MB2}, respectively. Since the Osman and Christofides' algorithm has one of its essential features in the terminating condition (see Section 2.3.1.2), and since the parameter setting for this condition allowed the Osman and Christofides' code to take less CPU time than that used for the other algorithms, we modified the stopping criterion in order to make a fair comparison. Let z_{OC} denote the results obtained by the modified Osman and Christofides' algorithm and let z_{OC2} denote the results obtained by the original code. Moreover, to provide information on the speed of converge of the BA, BA2 represents the results obtained by BA after two generations.

We report for BA, BA2, OC,OC2, HEUMED and MB1 the average of the best solutions obtained over five runs, the average time needed to obtain those best solutions and, every second line, the best of the five values produced by the corresponding algorithm for the problem considered. The results of MB2 are obviously relative to a single run.

All the heuristics, except OC2, were run for 600 seconds.

Tables 2.1, 2.2, 2.3 and 2.5 show the following columns, where all times are in seconds:

Probl.: a problem instance identifier;

z^* : optimal solution of the corresponding instance obtained by using the exact branch and bound method described in Section 3.5.2;

z_{BA} : average of the best solutions obtained by the bionomic algorithm;

t_{BA} : average time taken by the bionomic algorithm to get its best solutions;

z_{BA2}: average of the best solutions obtained by the bionomic algorithm after 2 generations;

t_{BA2} : average time taken by the bionomic algorithm to get its best solutions in the first 2 generations;

z_{OC} : best solution obtained by the modified OC algorithm;

t_{OC} : time taken by the modified OC algorithm to get its best solution;

z_{OC2}: best solution obtained by the original OC algorithm;

t_{OC2}: time taken by the original OC algorithm to get its best solution;

t_{OCtot2}: total time taken by the original OC algorithm;

z_{HEU}: average of the best solutions obtained by the HEUMED algorithm;

t_{HEU}: average time taken by the HEUMED algorithm to get its best solutions;

z_{MB1}: average of the best solution obtained by the MB1 heuristic;

t_{MB1}: average time taken by the MB1 heuristic to get its best solution;

z_{MB2}: best solution obtained by the MB2 heuristic;

t_{MB2}: time taken by the MB2 heuristic to get its best solution;

LB: value of lower bound LR.

Table 2.4 shows the same columns except that we substituted column z^* by column z_{BEST}. This is because not all problems of class D could be solved to optimality, in fact we imposed a limit of 100000 to the number of tree search nodes explored during the branch and bound. Therefore in the table we report the best known solutions for each problem and we indicate by an asterisk which solutions are optimal.

In the last four lines of all tables we report:

- the average percentage distance from optimality, computed on the ten problems of each problem class, of the average of the best solutions obtained over the five runs (*average percentage avg. error*);
- the average of the best of the five values produced by the corresponding algorithm for the problem considered (*average percentage min. error*);
- the average CPU time, computed on the ten problems, of the average time taken by each heuristic to solve the corresponding problem (*average CPU time*);
- the number of problems for which the corresponding heuristic was capable of finding the optimal solution cost (*number of optimal solutions*).

Table 2.1 shows that BA, OC, OC2 and HEUMED are able to find the optimum for all 10 problems, while both versions of Mulvey and Beck failed to solve to optimality some problems. The BA was also very efficient on all instances, in fact, it always finds the optimal solution within its first two iterations, as it is shown in the BA2 columns.

Table 2.2 presents the results obtained by the same algorithms on problem class B.

The results obtained by the BA are comparable to those obtained by the OC algorithm in terms of quality of the solution provided. On this problem class OC was the better performing code and HEUMED has also been very effective.

Problems reported in tables 2.1 and 2.2 do not really demonstrate the superiority of BA over OC, OC2 or HEUMED.

Table 2.3 presents the results obtained by the same algorithms on problem class C. On this more difficult problem class, BA was the only one capable of finding all the optimal solutions. The average percentage errors on both the best solutions found and the average solutions on the five tests are better for BA than for any other heuristic.

The inherent difficulty of these instances is testified by the average distance of bound LB from optimality. The fact that, despite the degraded quality of the bound indication, our propagation procedure obtains good results testifies to the robustness of the approach proposed.

Table 2.4 presents the results obtained by the same algorithms on problem class D.

Also for this problem class, whose complexity is testified by the fact that only three out of ten problems could be solved to optimality, BA shows a superior performance. No other heuristic has been able to provide such good solutions. The time bound of 600 seconds proved to be too tight for OC, HEUMED, MB1 and MB2.

Table 2.5 presents the results obtained on problem class E. We could not provide a comparison with the OC algorithm, since it requires symmetric instances.

On this problem class BA shows a very good performance, considering the best solutions found, with respect to the other tested heuristics, even though its average results do not dominate those of MB1.

As a general comment on all five tables, note that MB1 is by far the most stable of all heuristics: only in one case in fact (see problem CCPX19) the average is different from the best solution found.

Table 2.1. Computational results of problem class A

Probl	LB	z*	z_{BA}	t_{BA}	z_{BA2}	t_{BA2}	z_{OC}	t_{OC}	z_{OC2}	t_{OC2}	t_{OC2tot}	z_{HEU}	t_{HEU}	z_{MB1}	t_{MB1}	z_{MB2}	t_{MB2}
CCPX1	705	713	713.00	0.14	713.00	0.14	713.00	1.58	713.00	1.58	4.34	713.00	0.06	713.00	0.05	740	0.33
			713		713		713		713			713		713			
CCPX2	740	740	740.00	0.02	740.00	0.02	740.00	0.58	740.00	0.58	3.94	740.00	0.22	740.00	0.02	740	0.28
			740		740		740		740			740		740			
CCPX3	749	751	751.00	0.12	751.00	0.12	751.00	1.25	751.00	1.25	4.10	751.00	0.17	751.00	0.02	751	7.09
			751		751		751		751			751		751			
CCPX4	651	651	651.00	0.05	651.00	0.05	651.00	0.58	651.00	0.58	3.39	651.00	0.05	651.00	0.05	651	0.33
			651		651		651		651			651		651			
CCPX5	664	664	664.00	0.13	664.00	0.13	664.00	1.70	664.00	1.70	3.77	664.00	24.00	664.00	0.33	666	1.27
			664		664		664		664			664		664			
CCPX6	778	778	778.00	0.00	778.00	0.00	778.00	1.00	778.00	1.00	5.52	778.00	0.30	778.00	0.02	778	74.34
			778		778		778		778			778		778			
CCPX7	779	787	787.00	0.10	787.00	0.10	787.00	2.42	787.00	2.42	5.15	787.00	0.26	787.00	0.53	789	40.22
			787		787		787		787			787		787			
CCPX8	771	820	820.00	0.20	820.00	0.20	820.00	5.80	820.00	5.80	10.18	820.00	0.89	820.00	3.93	820	25.05
			820		820		820		820			820		820			
CCPX9	713	715	715.00	0.14	715.00	0.14	715.00	3.98	715.00	3.98	11.43	715.00	0.65	715.00	0.05	715	4.18
			715		715		715		715			715		715			
CCPX10	816	829	829.00	0.01	829.00	0.01	829.00	5.21	833.40	1.56	3.40	829.00	0.38	832.00	87.58	837	33.13
			829		829		829		829			829		832			
Average percentage avg.error			0.00		0.00		0.00		0.06			0.00		0.04		0.52	
Average percentage min.error			0.00		0.00		0.00		0.00			0.00		0.04			
Average CPU time			0.09		0.09		2.41		2.05			2.70		9.26		18.62	
Number of optimal solutions			10		10		10		10			10		9		6	

Table 2.2. Computational results of problem class B

Probl	LB	z*	z_{BA}	t_{BA}	Z_{BA2}	t_{BA2}	z_{OC}	t_{OC}	z_{OC2}	t_{OC2}	t_{OC2tot}	z_{HEU}	t_{HEU}	z_{MB1}	t_{MB1}	z_{MB2}	t_{MB2}
CCPX11	1000	1006	1006.00	25.22	1006.00	25.22	1006.00	24.58	1006.00	24.58	100.57	1006.00	7.33	1007.00	222.64	1071	2.09
			1006		1006		1006		1006			1006		1007			
CCPX12	958	966	966.00	105.18	966.20	41.32	966.00	44.28	966.00	44.28	128.67	966.00	30.42	966.00	6.37	972	1.10
			966		966		966		966			966		966			
CCPX13	1022	1026	1026.00	167.60	1026.80	94.68	1026.00	23.21	1026.00	23.21	106.13	1026.00	37.68	1026.00	7.03	1105	1.59
			1026		1026		1026		1026			1026		1026			
CCPX14	972	982	982.00	101.02	982.00	101.02	983.80	43.00	985.60	13.23	74.36	982.00	20.54	985.00	64.73	998	1.26
			982		982		982		985			982		985			
CCPX15	1079	1091	1091.80	127.42	1092.40	57.30	1092.60	44.81	1092.60	44.81	129.85	1092.00	33.20	1092.00	348.11	1102	1.21
			1091		1092		1091		1091			1092		1092			
CCPX16	947	954	954.20	219.99	954.80	40.68	954.60	57.65	954.60	57.65	136.88	954.00	49.46	954.00	14.43	1072	2.09
			954		954		954		954			954		954			
CCPX17	1024	1034	1034.00	256.51	1035.40	137.42	1034.80	186.83	1039.00	60.87	122.47	1034.00	221.02	1034.00	93.37	1122	1.92
			1034		1034		1034		1039			1034		1034			
CCPX18	1032	1043	1043.00	88.70	1043.00	88.70	1043.80	105.54	1045.20	31.75	94.73	1043.00	72.64	1043.00	38.42	1266	1.76
			1043		1043		1043		1045			1043		1043			
CCPX19	1024	1031	1031.40	231.25	1032.80	125.24	1032.60	36.40	1032.60	36.40	82.87	1031.20	412.73	1033.40	331.32	1067	1.59
			1031		1031		1031		1031			1031		1032			
CCPX20	972	1005	1013.00	291.98	1014.60	70.51	1007.60	140.80	1009.00	26.41	70.60	1010.00	159.01	1014.00	65.39	1014	391.43
			1013		1013		1005		1005			1008		1014			
Avg. percentage avg.error			0.09		0.16		0.10		0.18			0.06		0.16		6.42	
Avg. percentage min.error			0.08		0.09		0.00		0.10			0.04		0.15			
Average CPU time			161.49		78.21		70.71		36.32			104.40		119.18		40.60	
Number of optimal solutions			9		8		10		7			8		5		0	

Table 2.3. Computational results of problem class C

Probl	LB	z*	z_{BA}	t_{BA}	z_{BA2}	t_{BA2}	z_{OC}	t_{OC}	z_{OC2}	t_{OC2}	t_{OC2tot}	z_{HEU}	t_{HEU}	z_{MB1}	t_{MB1}	z_{MB2}	t_{MB2}
RSYM1	812	886	886.00	79.07	898.80	19.30	924.60	123.88	925.20	14.79	30.91	904.00	406.37	940.00	48.89	910	147.04
			886		886		886		886			886		940			
RSYM2	638	770	770.00	103.98	804.40	24.06	806.00	7.2971	820.80	5.33	12.42	786.40	418.86	776.00	43.69	804	28.85
			770		778		776		794			770		776			
RSYM3	722	844	844.00	54.77	874.40	11.99	874.80	15.57	874.80	15.57	33.34	860.80	316.25	876.00	48.56	844	344.12
			844		844		844		844			844		876			
RSYM4	686	716	716.00	8.05	716.00	8.05	766.50	51.583	768.40	4.16	10.92	729.60	266.62	726.00	66.24	726	21.43
			716		716		716		716			722		726			
RSYM5	680	778	781.20	260.81	854.40	22.21	867.20	91.023	897.60	5.34	11.65	828.80	550.55	778.00	74.21	868	56.16
			778		814		778		814			794		778			
RSYM6	676	784	785.20	108.74	798.00	10.69	807.60	155.18	859.60	5.09	12.48	790.40	470.04	790.00	44.67	796	114.78
			784		784		784		802			784		790			
RSYM7	754	844	844.00	35.39	860.80	7.76	858.40	9.0874	872.80	7.55	18.27	853.60	369.46	882.00	35.80	886	110.55
			844		844		844		844			844		882			
RSYM8	632	690	690.00	15.80	696.00	13.26	713.20	26.132	716.00	9.56	22.46	690.00	347.33	704.00	45.88	692	15.88
			690		690		690		692			690		704			
RSYM9	648	774	774.00	141.29	818.40	11.03	809.60	119.61	812.00	5.41	11.80	804.80	296.34	802.00	34.67	808	24.18
			774		802		780		780			782		802			
RSYM10	646	742	742.00	48.83	788.00	17.26	762.80	97.813	802.40	7.87	15.14	764.00	276.17	812.00	12.43	760	29.67
			742		786		742		742			742		812			
Avg. percentage avg.error			0.06		3.59		4.63		6.66			2.36		3.30		3.40	
Avg. percentage min.error			0.00		1.48		0.15		1.10			0.38		3.30			
Average CPU time			85.67		14.56		69.72		8.07			371.80		45.50		89.27	
Number of optimal solutions			10		6		8		5			7		1		1	

Table 2.4. Computational results of problem class D

Probl	LB	z_{BEST}	z_{BA}	t_{BA}	z_{BA2}	t_{BA2}	z_{OC}	t_{OC}	z_{OC2}	t_{OC2}	t_{OC2tot}	z_{HEU}	t_{HEU}	z_{MB1}	t_{MB1}	z_{MB2}	t_{MB2}
RSYM11	760	936	959.60	432.73	1063.60	92.32	1069.60	217.96	1098.40	131.25	318.35	1352.00	239.22	1144.00	104.09	1748	1.16
			936		998		1010		1074			1302		1144			
RSYM12	770	948	975.60	433.35	1099.60	92.95	1042.40	192.60	1043.60	170.30	410.21	1328.80	276.30	1210.00	177.72	1650	3.35
			948		1058		1002		1006			1196		1210			
RSYM13	754	822 *	832.40	367.75	914.40	92.72	949.20	256.14	949.20	256.14	467.51	1341.60	475.46	1198.00	80.12	1874	1.26
			822		874		906		906			1318		1198			
RSYM14	778	910	960.40	484.92	1095.60	70.67	1058.80	219.07	1078.40	244.27	517.66	1363.20	333.48	1220.00	73.50	1820	1.10
			910		1054		1006		1032			1306		1220			
RSYM15	768	870 *	950.40	517.66	1090.40	57.35	1054.80	176.49	1068.40	134.20	379.99	1354.40	476.00	1158.00	28.15	1734	1.93
			924		1032		998		998			1296		1158			
RSYM16	688	762 *	773.20	487.90	891.60	61.49	923.60	231.93	970.40	194.06	426.21	1230.80	233.27	1068.00	96.72	1476	1.21
			764		818		880		888			1152		1068			
RSYM17	706	816	834.80	371.39	964.40	97.54	959.20	197.71	970.40	162.07	359.41	1278.40	296.45	1098.00	24.68	1478	1.98
			816		928		920		920			1274		1098			
RSYM18	676	850	876.80	440.19	1008.00	66.10	953.20	189.83	953.20	189.83	337.49	1210.00	242.42	1078.00	23.49	1724	1.15
			850		1000		916		916			1130		1078			
RSYM19	736	870	906.40	536.22	1018.80	72.29	1036.00	242.22	1036.00	242.22	404.91	1331.20	189.42	1108.00	148.25	1616	1.21
			870		946		992		992			1324		1108			
RSYM20	748	888	938.00	444.93	1039.20	72.60	1043.60	266.06	1055.60	217.40	442.25	1303.20	156.92	1222.00	153.83	1542	1.27
			888		994		990		1020			1232		1222			
Avg. Percentage avg.error			3.87		17.45		16.36		17.89			50.99		32.66		92.14	
Avg. Percentage min.error			0.65		11.88		10.93		12.45			44.49		32.66			
Average CPU time			451.70		77.60		219.00		194.17			291.89		91.05		1.56	
Num. best known solutions			8		0		0		0			0		0		0	

Table 2.5. Computational results of problem class E

Probl	LB	z*	z_{BA}	t_{BA}	Z_{BA2}	t_{BA2}	z_{HEU}	t_{HEU}	z_{MB1}	t_{MB1}	z_{MB2}	t_{MB2}
BMM1	955	1058	1109.20	90.15	1126.60	3.80	1092.40	249.08	1086.00	66.30	1097	433.41
			1058		1058		1086		1086			
BMM2	862	1025	1065.40	302.44	1109.60	19.31	1064.40	259.90	1036.00	19.93	1125	185.82
			1025		1077		1036		1036			
BMM3	927	1081	1081.00	158.98	1115.00	28.66	1117.20	287.91	1130.00	60.07	1171	77.86
			1081		1110		1111		1130			
BMM4	841	933	941.00	41.09	1044.80	4.45	969.00	165.76	933.00	76.56	974	336.37
			933		973		933		933			
BMM5	878	1045	1045.00	222.41	1124.40	15.05	1110.60	444.30	1060.00	50.84	1094	299.56
			1045		1060		1094		1060			
BMM6	886	1006	1051.20	355.46	1084.60	11.29	1054.20	260.34	1012.00	61.90	1074	63.90
			1008		1025		1009		1012			
BMM7	870	983	1054.80	321.43	1085.00	11.56	1030.00	206.10	983.00	47.71	1101	69.06
			984		1038		983		983			
BMM8	839	939	939.00	18.70	943.80	11.47	984.20	343.19	967.00	52.38	965	370.72
			939		939		963		967			
BMM9	909	1021	1039.00	289.60	1089.20	13.98	1046.20	292.96	1029.00	26.98	1054	111.10
			1021		1021		1031		1029			
BMM10	838	1014	1017.00	104.95	1102.00	9.86	1031.00	396.36	1055.00	62.50	1043	152.75
			1014		1065		1014		1055			
Average percentage avg.error			2.35		7.13		3.90		1.84		5.87	
Average percentage min.error			0.03		2.58		1.53		1.84			
Average CPU time			190.52		12.94		290.59		52.52		210.06	
Number of optimal solutions			8		3		3		2		0	

2.8 Summary

The chapter describes the results obtained by applying a new metaheuristic technique, called the Bionomic Algorithm, to the capacitated p-median problem (CPMP). Bionomic Algorithms are evolutionary metaheuristic algorithms that update a whole set of solutions (a population of solutions) at each main cycle. They differ from similar previously presented algorithms, namely Genetic Algorithms and Evolution Strategies, because they explicitly direct the choice of the solutions to combine in order to define an offspring, that is, a solution in the population of the next iteration. This feature introduces a diversification mechanism for clustering by reference to maximal independent sets, carried out over progressively smaller domains, to provide a specific refinement of the scatter search proposal for generating parents from clustering strategies. The parent selection process, together with the use of problem-specific ways to produce an offspring from the parents, makes Bionomic Algorithms well-suited to combinatorial optimization applications.

The implementation of the Bionomic Algorithm presented for the CPMP in this chapter has the following characteristics. Maturation is based on a state-of-the-art heuristic for the Generalized Assignment Problem (GAP), a problem to which CPMP reduces once the p medians are chosen. Propagation, specifically the definition of a child solution once a parent set is assembled, is based on the computation of a Lagrangean lower bound for the CPMP.

The computational results, presented both on standard data sets from the literature and on more difficult symmetric and asymmetric cost instances, attest the effectiveness of the approach. Our findings can motivate future research that could examine additional types of clustering strategies (e.g., incorporating intensification criteria as well as diversification criteria) for choosing and combining multiple parents.

CHAPTER 3

AN EXACT ALGORITHM FOR SOLVING THE CAPACITATED *P*-MEDIAN PROBLEM BASED ON A SET PARTITIONING APPROACH

3.1 INTRODUCTION

The purpose of this chapter is to present a new exact method for solving the CPMP. In this chapter the CPMP is formulated as a Set Partitioning Problem with a side constraint (SP). Each column of the SP corresponds to a feasible cluster and the additional constraint forces any feasible solution to contain exactly p clusters. Also, in this chapter, a valid lower bound to the CPMP is developed. This lower bound is computed as the cost of a feasible solution to the dual of the LP-relaxation of SP (called DSP). The procedure for computing the lower bound, called HDSP, combines two different heuristic algorithms. Each of these heuristic procedures finds a feasible solution to DSP without requiring the entire set of the dual constraints. The dual solution obtained and a valid upper bound to the CPMP are then used to eliminate a large number of clusters that cannot belong to any optimal CPMP solution. However, the size of the reduced SP problem might still be too large for a branch and bound method. In this case, we develop a procedure, called EHP, for solving problem SP where the set of clusters is replaced with a subset of limited size. The optimal solution of the resulting problem might not be an optimal CPMP solution, however, the solution obtained from EHP allows us to estimate how far this solution is from the optimal solution to the CPMP. Furthermore, the EHP is readily extendible to deal with additional constraints, such as the maximum cluster cardinality, customer incompatibility, etc.

The chapter is structured as follows. In Section 3.2 the set partitioning mathematical formulation of the CPMP is presented. Section 3.3 describes the heuristic procedure HDSP for solving problem SP. Section 3.4 presents the method used for generating the clusters, i.e. the columns of the set partitioning formulation of CPMP. The EHP method for solving the CPMP is described in Section 3.5. In Section 3.6, computational results are presented for a number of problems drawn from the literature and for a new set of problems with additional constraints. Finally, conclusions are presented in Section 3.7.

3.2 A SET PARTITIONING FORMULATION OF THE CPMP

The CPMP can be formulated as a set partitioning problem with an additional constraint as follows.

- Let $\mathcal{P}_j$ be the index set of all feasible clusters whose median is customer j, $j \in N$, and let $\mathcal{B} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \cup \mathcal{P}_n$.
- Let $\mathcal{B}_i$ be the index set of all clusters containing customer $i \in N$.
- Let c_ℓ, B_ℓ and π_ℓ indicate the cost, the subset of customers and the median of cluster $\ell \in \mathcal{B}$, respectively.
- Let x_ℓ be a (0-1) variable that is equal to 1 if and only if cluster $\ell \in \mathcal{B}$ belongs to the optimal solution.

The resulting mathematical formulation, called SP, of problem CPMP is as follows :

(SP) $$z(\text{SP}) = Min \sum_{\ell \in \mathcal{B}} c_\ell x_\ell \tag{3.1}$$

$$subject\ to \sum_{\ell \in \mathcal{B}_i} x_\ell = 1, \qquad \forall i \in N \tag{3.2}$$

$$\sum_{\ell \in \mathcal{B}} x_\ell = p \tag{3.3}$$

$$x_\ell \in \{0,1\}, \qquad \forall \ell \in \mathcal{B}. \tag{3.4}$$

Equations (3.2) impose the requirement that each customer $i \in N$ is assigned to one cluster. Equation (3.3) forces the solution to contain p clusters and constraints (3.4) are the integrality constraints. The formulation SP is easily extendible to deal with additional cluster constraints simply by removing from $\mathcal{B}$ any infeasible cluster.

Hansen et. al (1994) have shown that solving the LP-relaxation of formulation F described in Section 2.2 cannot yield a larger lower bound than the one obtained by solving the LP-relaxation of formulation SP.

Problem SP cannot be solved directly since the number of columns may be enormous even for CPMP instances of moderate size. In the following, the dual problem (called DSP) of the linear relaxation of SP, is used in order to generate a valid lower bound to the CPMP. The method used to solve DSP is heuristic (which of course does not affect the optimality of the final CPMP solution) and it does not require the explicit generation of the cluster-index set $\mathcal{B}$. Moreover, the dual solution obtained is used to reduce the set $\mathcal{B}$ by removing those clusters that cannot belong to any optimal CPMP solution.

Let u_i, $i \in N$, be the dual variables associated with constraints (3.2) and w be the dual variable of constraint (3.3). The dual of the LP-relaxation of SP, called DSP, is as follows:

(DSP)
$$z(\text{DSP}) = Max \quad \sum_{i \in N} u_i + pw \tag{3.5}$$

$$subject\ to \quad \sum_{i \in B_\ell} u_i + w \le c_\ell, \qquad \forall \ell \in \mathcal{B} \tag{3.6}$$

$$\left.\begin{array}{l} u_i \text{ unrestricted, } i \in N \\ w \text{ unrestricted} \end{array}\right\} \tag{3.7}$$

Like problem SP, problem DSP is impractical to solve, since the number of constraints is equal to the number of variables in SP.

3.2.1 Variable reduction of problem SP

Let $(\mathbf{u}', w')$ be a feasible solution of DSP of cost $z'(\text{DSP})$ and let $\mathbf{x}'$ be a feasible solution of SP of cost $z'(\text{SP})$. It is well known, from linear programming duality theory, that $z'(\text{DSP}) \leq z'(\text{SP})$ and, consequently, any feasible solution of DSP provides a valid lower bound to SP.

Let c'_ℓ be the *reduced cost* of cluster $\ell \in \mathcal{B}$ corresponding to the dual solution $(\mathbf{u}', w')$, that is:

$$c'_\ell = c_\ell - \sum_{i \in B_\ell} u'_i - w' \tag{3.8}$$

This dual solution $(\mathbf{u}', w')$ can be used to reduce the number of variables of SP as it is established by the following theorem.

Theorem 3.1. Let $S' = \{\ell : \ell \in \mathcal{B},\ \text{s.t. } x'_\ell = 1\}$. The following relationship holds:

$$z'(\text{SP}) = z'(\text{DSP}) + \sum_{\ell \in S'} c'_\ell \tag{3.9}$$

Proof. From equation (3.8), we have:

$$\sum_{\ell \in S'} c'_\ell = \sum_{\ell \in S'} c_\ell - \sum_{\ell \in S'} \sum_{i \in B_\ell} u'_i - \sum_{\ell \in S'} w'$$

Since $\mathbf{x}'$ is a feasible solution of CPMP, we have:

$$\sum_{\ell \in S'} \sum_{i \in B_\ell} u'_i + \sum_{\ell \in S'} w' = \sum_{i \in N} u'_i + pw'$$

and hence,

$$\sum_{\ell \in S'} c'_\ell = \sum_{\ell \in S'} c_\ell - \sum_{i \in N} u'_i - pw' \tag{3.10}$$

Noticing that $z'(\text{SP}) = \sum_{\ell \in S'} c_\ell$ and that $z'(\text{DSP}) = \sum_{i \in N} u'_i + pw'$, from equation (3.10) we obtain equation (3.9)■

Corollary 3.1. Let $z(\text{UB})$ be the cost of a feasible CPMP solution and $(\mathbf{u}', w')$ be a feasible solution of DSP of cost $z'(\text{DSP})$. Any optimal solution of SP of cost less than $z(\text{UB})$ cannot contain any cluster $\ell \in \mathcal{B}$ whose reduced cost is greater or equal to $z(\text{UB}) - z'(\text{DSP})$.

Proof. It follows directly from Theorem 3.1■

Corollary 3.1 states that an optimal CPMP solution can be obtained by replacing in problem SP the set $\mathcal{B}$ with the subset $\mathcal{B}'$ defined as follows:

$$\mathcal{B}' = \{\ell : \ell \in \mathcal{B}, \text{ s.t. } c'_\ell < z(\text{UB}) - z'(\text{DSP})\} \tag{3.11}$$

therefore, an optimal solution of problem SP can be obtained by replacing set $\mathcal{B}$ with $\mathcal{B}'$. However, the size of $\mathcal{B}'$ might still be too large for solving problem SP, even if the gap $z(\text{UB}) - z'(\text{DSP})$ is small. In this case we propose to solve problem SP by using a subset $\mathcal{F} \subseteq \mathcal{B}'$ containing a limited number of clusters so that the resulting problem is solvable by an integer programming solver (e.g. CPLEX (1996)).

The optimal solution obtained for the resulting problem SP is not guaranteed to be an optimal CPMP solution. However, the method used to choose the subset $\mathcal{F}$ allows us to estimate the distance of the solution obtained from optimality.

3.3 A HEURISTIC PROCEDURE FOR SOLVING PROBLEM DSP

In this section we describe a heuristic procedure (called HDSP) for finding a feasible solution to DSP that is based on the following observation.

Consider the integer program P:

$$\text{(P)} \qquad \begin{aligned} \mathit{Min}\; z(\text{P}) = \;& \mathbf{cx} \\ \mathit{subject\ to}\;\; & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \text{ integer} \end{aligned}$$

The dual D of the LP-relaxation of P is:

$$\text{(D)} \qquad \begin{aligned} \mathit{Max}\; z(\text{D}) = \;& \mathbf{wb} \\ \mathit{subject\ to}\;\; & \mathbf{wA} \leq \mathbf{c} \\ & \mathbf{w} \text{ unrestricted} \end{aligned}$$

A feasible solution $\overline{\mathbf{w}}$ of D of cost $\overline{z}(\text{D})$ can be obtained by means of the following simple observation. Assume that $\mathbf{w}$ is a feasible solution of D of cost $z(\text{D})$ and that $\mathbf{w}'$ is a feasible solution of cost $z'(\text{D}')$ of the following problem D′:

$$\text{(D}'\text{)} \qquad \begin{aligned} \mathit{Max}\; z'(\text{D}') = \;& \mathbf{w}'\mathbf{b} \\ \mathit{subject\ to}\;\; & \mathbf{w}'\mathbf{A} \leq \mathbf{c} - \mathbf{wA} \\ & \mathbf{w}' \text{ unrestricted} \end{aligned}$$

Since D′ imposes that $(\mathbf{w}+\mathbf{w}')\mathbf{A} \le \mathbf{c}$, it is easy to see that $\overline{\mathbf{w}} = \mathbf{w}+\mathbf{w}'$ is a feasible solution of D of cost $\overline{z}(\mathrm{D}) = z(\mathrm{D}) + z'(\mathrm{D}')$. This observation justifies the validity of the following algorithm HDSP for solving problem D.

***Algorithm 3.1*:** Algorithm HDSP for finding a feasible solution of D

Let $\mathrm{H}^1, \mathrm{H}^2, \ldots, \mathrm{H}^k$ be k different heuristic procedures for solving D.

Step 1. Set $\overline{\mathbf{w}} = \mathbf{0}$ and $\overline{z}(\mathrm{D}) = 0$.

Step 2. Repeat Step 3 for $r = 1,2,\ldots,k$.

Step3. Use the heuristic procedure H^r for finding a solution $\mathbf{w}'$ of cost $z'(\mathrm{D}')$ to the following problem D′:

$$(\mathrm{D}') \qquad \begin{aligned} \mathit{Max}\, z'(\mathrm{D}') = \;& \mathbf{w}'\mathbf{b} \\ \mathit{subject\ to}\;& \mathbf{w}'\mathbf{A} \le \mathbf{c}' \\ & \mathbf{w}' \text{ unrestricted} \end{aligned}$$

where $\mathbf{c}' = \mathbf{c} - \overline{\mathbf{w}}\mathbf{A}$. Update $\overline{\mathbf{w}} = \overline{\mathbf{w}} + \mathbf{w}'$ and $\overline{z}(D) = \overline{z}(D) + z'(D')$.

The application of procedure HDSP to CPMP involves, at each iteration r, the use of a heuristic procedure H^r to solve the following problem DSP′.

$$(\mathrm{DSP}') \qquad \begin{aligned} z'(\mathrm{DSP}') = \mathit{Max}\;& \sum_{i\in N} u_i' + pw' \\ \mathit{subject\ to}\;& \sum_{i\in B_\ell} u_i' + w' \le c_\ell', \qquad \ell \in \mathcal{B} \\ & u_i' \text{ unrestricted, } i \in N \\ & w' \text{ unrestricted} \end{aligned}$$

where $c_\ell' = c_\ell - \sum_{i\in B_\ell} \overline{u}_i - \overline{w}$ and $(\overline{\mathbf{u}}, \overline{w})$ is a feasible solution to DSP of cost $\overline{z}(\mathrm{DSP})$ provided by the first $(r-1)$ iterations of HDSP.

Notice that DSP′ is the dual of the linear relaxation of problem SP′ that is obtained from SP replacing in the objective function (3.1) each cost c_ℓ with the reduced cost c_ℓ', $\forall \ell \in \mathcal{B}$.

The cost $\bar{z}(\mathrm{D})$ of the solution $\overline{\mathbf{w}}$ of D obtained by HDSP is limited from above by the optimal solution cost of the LP-relaxation of SP, hence, the lower bound $\bar{z}(\mathrm{DSP})$ to the CPMP obtained by HDSP is not better than the optimal cost of the LP-relaxation of SP. This method has been applied by Mingozzi et al. (1994) for solving the Vehicle Routing Problem, by Bianco et al. (1994) for the Multiple Depot Vehicle Scheduling, by Mingozzi et al. (1995) for the Crew Scheduling Problem and is used in Chapter 5 for solving for the Vehicle Routing Problem with Backhauls.

The procedure HDSP involves two heuristics. The first, called H^1, solves problem $\mathrm{DSP}^1(\equiv \mathrm{DSP})$ and does not require the generation of set $\mathcal{B}$ while the second procedure, called H^2, solves DSP^2 and requires the generation of a limited subset of set $\mathcal{B}$.

3.3.1 PROCEDURE H^1

Procedure H^1 is based on the lower bound obtained from formulation F of Section 2.2 by relaxing the set partitioning constraints (2.2) in a Lagrangean fashion. (Hansen et. al (1994) have shown that the optimal solution of this relaxation and of the LP-relaxation of the SP formulation have the same value). Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be the lagrangean multipliers associated to constraints (2.2). The relaxed problem, called LR(λ), is as follows :

$$(\mathrm{LR}(\lambda)) \qquad z(\mathrm{LR}(\lambda)) = Min \sum_{i\in N}\sum_{j\in N}(d_{ij} - \lambda_i)\xi_{ij} + \sum_{i\in N}\lambda_i$$

$$\textit{subject to} \quad \sum_{i\in N} q_i \xi_{ij} \le Q_j \xi_{jj}, \qquad \forall j \in N$$

$$\sum_{j\in N} \xi_{jj} = p$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall\, i, j \in N$$

For a given set of multiplier vector λ, problem LR(λ) can be decomposed into n independent knapsack problems, called $\mathrm{KP}_j(\lambda)$, $j = 1,\ldots,n$, of the form :

$$(\mathrm{KP}_j(\lambda)) \qquad z\big(\mathrm{KP}_j(\lambda)\big) = Min \sum_{i\in N}(d_{ij} - \lambda_i)\xi_{ij}$$

$$\textit{subject to} \quad \sum_{i\in N} q_i \xi_{ij} \le Q_j$$

$$\xi_{ij} \in \{0,1\},\ i \in N$$

An optimal $\mathrm{KP}_j(\lambda)$ solution corresponds to a cluster $B_{\ell^*_j}$ such that $\ell^*_j = arg\ min_{\ell \in \mathcal{P}_j} \left\{ c_\ell - \sum_{i\in B_\ell} \lambda_i \right\}$. An optimal solution of LR(λ) is obtained as follows.

Let $z(\mathrm{KP}_{j_1}(\lambda)), z(\mathrm{KP}_{j_2}(\lambda)), \ldots, z(\mathrm{KP}_{j_p}(\lambda))$ be the p-least cost solutions of the n Knapsack problems $\mathrm{KP}_j(\lambda), j = 1,\ldots,n$. It is easy to see that the optimal LR(λ) solution is given by:

$$\xi^*_{j_k j_k} = \begin{cases} 1, & k = 1,\ldots,p \\ 0, & k = p+1,\ldots,n \end{cases}$$

and

$$\xi^*_{ij_k} = 1 \text{ if } i \in B_{\ell^*_{j_k}},\ \xi^*_{ij_k} = 0 \text{ otherwise, } \forall i \in N,\ k = 1,\ldots,p$$

and

$$\xi^*_{ij_k} = 0,\ \forall i \in N,\ k = p+1,\ldots,n.$$

The value of the objective function for this solution is $z(\mathrm{LR}(\lambda)) = \sum_{k=1}^{p} z(\mathrm{KP}_{j_k}(\lambda)) + \sum_{i\in N} \lambda_i$.

Theorem 3.2 shows that the solution of LR(λ), for any vector λ, can be transformed into a feasible DSP solution.

Theorem 3.2. Let ξ^* be the optimal solution of LR(λ) of cost $z(\mathrm{LR}(\lambda))$ for a given vector λ. A feasible solution $(\mathbf{u}^1, w^1)$ of DSP of cost $z(\mathrm{DSP}^1) = z(\mathrm{LR}(\lambda))$ is obtained by setting, for every customer $i \in N$:

$$u_i^1 = \begin{cases} \lambda_i, & \text{if } \xi^*_{ii} = 0 \\ \lambda_i + z(\mathrm{KP}_i(\lambda)) - \sigma, & \text{if } \xi^*_{ii} = 1 \end{cases} \tag{3.12.a}$$

$$w^1 = \sigma \tag{3.12.b}$$

where $\sigma = \max\{z(\mathrm{KP}_j(\lambda))\xi^*_{jj} : j \in N\}$.

Proof. It is sufficient to show that, for any $j \in N$:

$$\sum_{i \in B_\ell} u^1_i + w^1 \leq c_\ell,\ \forall \ell \in \mathcal{P}_j \tag{3.13}$$

We must consider two cases:

(a) $\xi^*_{jj} = 0$.

Using equations (3.12.a) and (3.12.b), inequalities (3.13) can be written as:

$$\sigma \leq c_\ell - \sum_{i \in B_\ell} \lambda_i,\ \forall \ell \in \mathcal{P}_j. \tag{3.14}$$

As $\xi^*_{jj} = 0$, from the definition of σ, we have

$$z(\mathrm{KP}_j(\lambda)) \geq \sigma. \tag{3.15}$$

Moreover, since $z(\mathrm{KP}_j(\lambda))$ is the optimal solution cost of problem $\mathrm{KP}_j(\lambda)$ we have

$$z(\mathrm{KP}_j(\lambda)) \leq c_\ell - \sum_{i \in B_\ell} \lambda_i,\ \forall \ell \in \mathcal{P}_j. \tag{3.16}$$

Hence, from (3.15) and (3.16) we obtain inequalities (3.14).

(b) $\xi^*_{jj} = 1$.

Using equations (3.12.a) and (3.12.b), inequalities (3.13) can be written as

$$\sum_{i \in B_\ell} \lambda_i + z(\mathrm{KP}_j(\lambda)) - \sigma + \sigma \leq c_\ell,\ \forall \ell \in \mathcal{P}_j$$

that is equivalent to

$$z(\mathrm{KP}_j(\lambda)) \leq c_\ell - \sum_{i \in B_\ell} \lambda_i,\ \forall \ell \in \mathcal{P}_j \tag{3.17}$$

and this latter inequality is verified since $z(\mathrm{KP}_j(\lambda))$ is an optimal solution of problem $\mathrm{KP}_j(\lambda)$ ■

Algorithm H^1 for solving DSP^1 is an iterative procedure that finds a feasible solution of the problem $\underset{\lambda}{Max}[z(\mathrm{LR}(\lambda))]$. An iteration of H^1 consists of computing a new vector λ and of finding a new solution of the resulting problem LR(λ). The method used for updating λ, at each iteration, is as follows.

Let ξ^* be the optimal solution of LR(λ). Let h_i be the number of medians that customer $i \in N$ is assigned to, in the solution ξ^*, i.e. $h_i = \sum_{j \in N} \xi^*_{ij}$. In any feasible CPMP solution we have $h_i = 1$, $i \in N$, hence, a subgradient optimization method can be used to change λ as follows:

$$\lambda_i = \lambda_i - \varepsilon \frac{z(\text{UB}) - z(\text{LR}(\lambda))}{\sum_{k \in N} (h_k - 1)^2} (h_i - 1), \ \forall i \in N$$

where $z(\text{UB})$ is a valid upper bound to the optimal CPMP solution cost and ε is the step size (and is a parameter). A feasible DSP^1 solution $(\mathbf{u}^1, w^1)$ of cost $z(\text{DSP}^1)$ is given by equations (3.12) using the vector λ^* that has produced, within a priori fixed number of iterations, the best approximate solution of the problem $\underset{\lambda}{Max}[z(\text{LR}(\lambda))]$.

3.3.2 Procedure H^2

Procedure H^2 is a heuristic procedure based on linear programming that finds a feasible solution of the following problem :

$$(\text{DSP}^2) \qquad z(\text{DSP}^2) = Max \ \sum_{i \in N} u_i + pw$$

$$\textit{subject to} \ \sum_{i \in B_\ell} u_i + w \le c^2_\ell, \qquad \forall \ell \in \mathcal{B}$$

$$\left.\begin{array}{l} u_i \text{ unrestricted}, \forall i \in N \\ w \text{ unrestricted} \end{array}\right\}$$

where c^2_ℓ is the reduced cost of cluster $\ell \in \mathcal{B}$ computed according to the dual solution $(\mathbf{u}^1, w^1)$ produced by procedure H^1, that is :

$$c^2_\ell = c_\ell - \sum_{i \in B_\ell} u^1_i - w^1.$$

Problem DSP^2 could not be solved directly as it might involve a huge number of constraints. In this section we describe a procedure called H^2, for reducing the number of constraints of DSP^2 so that the resulting problem, called RD, can be solved directly and any solution of RD is a feasible DSP^2 solution. Problem RD is obtained from DSP^2 as follows:

(i) The number of constraints (3.6) are reduced by replacing $\mathcal{B}$ with a subset $\mathcal{F}$ of limited size;

(ii) Constraints are added to force any RD solution to satisfy constraints (3.6) for any $\ell \in \mathcal{B} \backslash \mathcal{F}$.

Reduced problem RD

Let $\mathcal{F}_j$ be a subset of $\mathcal{P}_j$, $j \in N$, that satisfies the following two conditions :

$$\left.\begin{array}{ll} c_\ell^2 < z(\mathrm{UB}) - z\left(\mathrm{DSP}^1\right), \forall \ell \in \mathcal{F}_j & \text{(a)} \\ \underset{\ell \in \mathcal{F}_j}{Max}\left[c_\ell^2\right] \le \underset{\ell \in \mathcal{P}_j \backslash \mathcal{F}_j}{Min}\left[c_\ell^2\right] & \text{(b)} \end{array}\right\} \qquad (3.18)$$

A procedure for computing the sets $\mathcal{F}_j$, $j \in N$, is presented in Section 3.4. The reduced dual problem RD is obtained from DSP^2 by replacing the cluster set $\mathcal{B}$ with the subset $\mathcal{F} = \bigcup_{j \in N} \mathcal{F}_j$.

Problem RD is as follows:

$$(\mathrm{RD}) \qquad z(\mathrm{RD}) = Max \sum_{i \in N} u_i + pw \qquad (3.19)$$

$$subject\ to \quad \sum_{i \in B_\ell} u_i + w \le c_\ell^2, \qquad \forall \ell \in \mathcal{F} \qquad (3.20)$$

$$u_i \le U_i, \qquad \forall i \in N \qquad (3.21)$$

$$w \le 0 \qquad (3.22)$$

where the upper bound U_i, $i \in N$, are chosen such that :

$$\sum_{i \in B_\ell} U_i \le c_\ell^2, \forall \ell \in \mathcal{B} \backslash \mathcal{F} \qquad (3.23)$$

Theorem 3.3. Any feasible solution of RD is also a feasible solution of DSP^2.

Proof. Constraints (3.21) and (3.23) imply that every feasible solution to RD satisfies the following inequalities :

$$\sum_{i\in B_\ell} u_i + w \le \sum_{i\in B_\ell} U_i + w \le c_\ell^2 + w, \ \forall \ell \in \mathcal{B}\setminus\mathcal{F} \tag{3.24}$$

From inequalities (3.22) and (3.24), the theorem is proved.

Problem RD can be considered to be the dual of the following problem RP:

(RP) $\quad z(\text{RP}) = Min \ \sum_{\ell\in\mathcal{F}} c_\ell^2 x_\ell + \sum_{i\in N} U_i y_i$

subject to $\sum_{\ell\in\mathcal{F}_i} x_\ell + y_i = 1, \quad \forall i \in N$

$\sum_{\ell\in\mathcal{F}} x_\ell + y_0 = p$

$x_\ell \ge 0, \quad \forall \ell \in \mathcal{F}$

$y_i \ge 0, \quad \forall i \in N \cup \{0\}$

Procedure H^2 finds an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ of RP of cost $z^*(\text{RP})$ and the corresponding optimal dual solution $(\mathbf{u}^*, w^*)$ of RD. Hence, we have $z(\text{DSP}^2) = z^*(\text{RP})$ and $\mathbf{u}^2 = \mathbf{u}^*$, $w^2 = w^*$.

An optimal CPMP solution

Procedure HDSP finds a solution $(\mathbf{u}', w')$ of DSP of cost $z'(\text{DSP}) = z(\text{DSP}^1) + z(\text{DSP}^2)$ by setting $\mathbf{u}' = \mathbf{u}^1 + \mathbf{u}^2$ and $w' = w^1 + w^2$.

The cases where the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ of RP corresponds to an optimal solution of problem SP are as follows:

(a) $\mathbf{x}^*$ integer, $\mathbf{y}^* = 0$.

This solution is also an optimal CPMP solution of cost $z^*(\text{SP}) = z'(\text{DSP})$.

(b) $\mathbf{x}^*$ not integer, $U_i = \infty, \forall i \in N$, $\mathbf{y}^* = 0$.

In this case the clusters of any optimal CPMP solution are in set $\mathcal{F}$, hence, the solution $(\mathbf{u}', w')$ is an optimal solution of problem DSP and an optimal CPMP solution can be obtained by solving problem SP where set $\mathcal{B}$ is replaced with $\mathcal{F}$.

(c) $y^* \neq 0$.

The RP solution achieved is not feasible for CPMP even if variables x^*_ℓ, $\ell \in \mathcal{F}$, have integer values. In fact, $y^*_i > 0$, for some $i \in N$, implies that vertex i is not fully covered by the clusters that are in the optimal RP solution, i.e. set $\mathcal{F}$ might not contain an optimal or even a feasible CPMP solution.

3.3.2.1 THE COMPUTATION OF $U_i, i \in N$

A valid method for computing U_i, $i \in N$, in order to satisfy constraints (3.23) is as follows. Let $c^{max}_j = \underset{\ell \in \mathcal{F}_j}{Max}\left\{c^2_\ell\right\}, \forall j \in N$ and let j^* be such that:

$$c^{max}_{j^*} / Q_{j^*} = \underset{j \in N}{Min}\left[c^{max}_j / Q_j\right].$$

U_i, $i \in N$, is assigned the following value:

$$U_i = q_i \, c^{max}_{j^*} / Q_{j^*} \tag{3.25}$$

It is easy to show that the values assigned to U_i, $i \in N$ according to expression (3.25) satisfy inequalities (3.23). In fact, for any $\ell \in \mathcal{B} \setminus \mathcal{F}$,

$$\sum_{i \in B_\ell} U_i = \sum_{i \in B_\ell} q_i \, c^{max}_{j^*} / Q_{j^*} \tag{3.26}$$

since

$$c^{max}_{j^*} / Q_{j^*} \leq c^{max}_{\pi_\ell} / Q_{\pi_\ell} \leq c^2_\ell / Q_{\pi_\ell}$$

From (3.26), we obtain:

$$\sum_{i \in B_\ell} U_i \leq \sum_{i \in B_\ell} q_i \, c^2_\ell / Q_{\pi_\ell} \tag{3.27}$$

Moreover, as $\sum_{i \in B_\ell} q_i / Q_{\pi_\ell} \leq 1$, from inequalities (3.27) we have :

$$\sum_{i \in B_\ell} U_i \leq c^2_\ell, \forall \, \ell \in \mathcal{B} \setminus \mathcal{F} \blacksquare$$

3.4 Generation of set $\mathcal{F}$

In this section, a procedure, called GEN($\mathcal{F}_j$), that generates, for a given median j, a cluster set $\mathcal{F}_j$ satisfying conditions (3.18a) and (3.18b) is described. GEN($\mathcal{F}_j$) is derived from an exact dynamic programming procedure for generating set $\mathcal{P}_j$ by limiting the size of the state space graph in such a way that the states generated at the last stage of the recursion correspond to a set $\mathcal{F}_j$.

Let $c^2(B) = \sum_{i \in B} \left(d_{i\pi(B)} - u_i^1\right) - w^1$ be the reduced cost of cluster B for median $\pi(B)$ with respect to the dual solution $\left(\mathbf{u}^1, w^1\right)$ produced by procedure H^1. Let $\overline{N} = (i_1, i_2, \ldots, i_{n-1})$ be the ordered set of the $(n-1)$ customers obtained from N by removing the median customer j. From a computational point of view, the vector $\overline{N}$ is ordered so that $\left(d_{i_k j} - u_{i_k}^1\right) \le \left(d_{i_{k+1} j} - u_{i_{k+1}}^1\right)$, $k = 1, \ldots, n-2$.

Consider a feasible cluster $B \subseteq \{j, i_1, i_2, \ldots, i_k\}$ and let $f_k(B)$ be the reduced cost of the feasible cluster of minimum reduced cost that can be obtained by expanding B with the customer subset $\{i_{k+1}, i_{k+2}, \ldots, i_{n-1}\}$. The value of $f_k(B)$ can be computed as follows:

$$f_k(B) = c^2(B) + g_{k+1}(B) \tag{3.28}$$

where,

$$\begin{aligned} g_{k+1}(B) = \quad & Min \sum_{r=k+1}^{n-1} \left(d_{i_r j} - u_{i_r}^1\right) y_r \\ \textit{subject to} \quad & \sum_{r=k+1}^{n-1} q_{i_r} y_r \le \left(Q_j - \sum_{i \in B} q_i\right) \\ & y_r \in \{0,1\},\ r = k+1, \ldots, n-1 \end{aligned}$$

We assume $g_n(B) = 0$ so that $f_{n-1}(B) = c^2(B)$. It is easy to verify that function $f_k(B)$ has the following properties

(P1) $$f_k(B) \le f_{k+1}(B),\ k = 1, \ldots, n-2 \tag{3.29}$$

(P2) $$f_k(B) \le f_{k+1}(B \cup \{i_{k+1}\}),\ k = 1,\dots,n-2 \qquad (3.30)$$

Let $\mathcal{R}_k$ be the set of all feasible clusters that can be obtained by using the first k customers $\{i_1, i_2, \dots, i_k\}$ of $\overline{N}$ and such that:

$$f_k(B) < z(\text{UB}) - z(\text{DSP}^1),\ \forall B \in \mathcal{R}_k$$

We assume that $j \in B$ and $\sum_{i \in B} q_i \le Q_j$, $\forall B \in \mathcal{R}_k$. Every cluster B such that $\pi(B) \ne j$ is removed from $\mathcal{R}_{n-1}$. It is evident that the resulting set $\mathcal{R}_{n-1}$ is a subset of $\mathcal{P}_j$, that is: $\mathcal{R}_{n-1} = \{B \in \mathcal{P}_j : c^2(B) < z(\text{UB}) - z(\text{DSP}^1)\}$, and therefore, $\mathcal{R}_{n-1}$ can be used instead of $\mathcal{P}_j$ for generating $\mathcal{F}_j$. However, the size of each $\mathcal{R}_k$, $k = 1,\dots,n-1$, might be too large even if the gap $z(\text{UB}) - z(\text{DSP}^1)$ is small. To overcome this problem, we propose a procedure called GEN($\mathcal{F}_j$) that generates, for a given median j, a sequence of subsets $\overline{\mathcal{R}}_k$, $k = 1,\dots,n-1$, satisfying the following two conditions:

$$\left.\begin{array}{ll} |\overline{\mathcal{R}}_k| \le \Delta & \text{a} \\ \underset{B \in \overline{\mathcal{R}}_k}{Max}[f_k(B)] \le \underset{B \in \mathcal{R}_k \setminus \overline{\mathcal{R}}_k}{Min}[f_k(B)] & \text{b} \end{array}\right\} \qquad (3.31)$$

The cluster set $\mathcal{F}_j$ corresponds to $\overline{\mathcal{R}}_{n-1}$ after having removed any cluster B such that $\pi(B) \ne j$.

Procedure GEN($\mathcal{F}_j$) makes use of a temporary set T_k representing, at each stage k, a subset of $\mathcal{R}_k$ such that $\underset{B \in T_k}{Max}[f_k(B)] \le \underset{B \in \mathcal{R}_k \setminus T_k}{Min}[f_k(B)]$. The set $\overline{\mathcal{R}}_k$ is then extracted from T_k. At each stage k of GEN($\mathcal{F}_j$), $\overline{\mathcal{R}}_k \subseteq T_k \subseteq \mathcal{R}_k$.

***Algorithm* 3.2:** GEN($\mathcal{F}_j$)

Step 0. Set $T_0 = \{\{j\}\}$ and k=0.

Step 1. *(Define the subset $\overline{\mathcal{R}}_k \subseteq T_k$)*

if $|T_k| \le \Delta$, then set $\overline{\mathcal{R}}_k = T_k$ and define $h_k = z(\text{UB}) - z(\text{DSP}^1)$.

if $|T_k| > \Delta$, then let $\overline{\mathcal{R}}_k$ be the largest subset of T_k such that $|\overline{\mathcal{R}}_k| \le \Delta$ and $\underset{B \in \overline{\mathcal{R}}_k}{Max}[f_k(B)] \le \underset{B \in T_k \setminus \overline{\mathcal{R}}_k}{Min}[f_k(B)]$. Set $h_k = \underset{B \in \overline{\mathcal{R}}_k}{Max}[f_k(B)]$ and $T_{k+1} = \varnothing$.

Step 2. (*Generate* T_{k+1})

For any $B \in \overline{\mathcal{R}}_k$, consider the two clusters B and $B' = B \cup \{i_{k+1}\}$ as possible elements of T_{k+1}.

If $f_{k+1}(B) \le h_k$, then add B to T_{k+1}.

If $\sum_{i \in B'} q_i \le Q_j$ and $f_{k+1}(B') \le h_k$ then add B' to T_{k+1}.

Step 3. Set $k = k+1$. If $k < n-1$, then go to Step 1.

Step 4. (*Define* $\mathcal{F}_j$)

Remove from $\overline{\mathcal{R}}_{n-1}$ every cluster B such that $\pi(B) \ne j$.
Set $\mathcal{F}_j$ be the largest subset of $\overline{\mathcal{R}}_{n-1}$ such that $|\mathcal{F}_j| \le \Delta$ and $\underset{B \in \mathcal{F}_j}{Max}\left[c^2(B)\right] \le \underset{B \in \overline{\mathcal{R}}_{n-1} \setminus \mathcal{F}_j}{Min}\left[c^2(B)\right]$. Set $h_{n-1} = \underset{B \in \mathcal{F}_j}{Max}\left[c^2(B)\right]$.

It is easy to note that $h_1 \ge h_2 \ge \ldots \ge h_{n-1}$ since $f_{k+1}(B) \le h_k$, $\forall B \in T_{k+1}$ while $\overline{\mathcal{R}}_{k+1} \subseteq T_{k+1}$. In order to prove that every $\overline{\mathcal{R}}_k$, $k = 1, \ldots, n-1$, satisfies conditions (3.31), it is sufficient to show that

$$f_k(B) \ge h_k, \ \forall B \in \mathcal{R}_k \setminus T_k. \tag{3.32}$$

In fact, due to Step 1 if condition (3.32) holds, $f_k(B) \ge h_k$, $\forall B \in \mathcal{R}_k \setminus \overline{\mathcal{R}}_k$ and $f_k(B) \ge h_k$, $\forall B \in T_k \setminus \overline{\mathcal{R}}_k$.

Assume that $\mathcal{R}_{k-1}$ satisfies condition (3.31). This is true for $k = 2$ if $\Delta \ge 2$ since $|\mathcal{R}_1| = 2$. By contradiction, assume that there exists a cluster $B^* \in \mathcal{R}_k \setminus T_k$ such that $f_k(B^*) < h_k$. There are two cases:

1. $i_k \in B^*$. For property P2, $f_{k-1}(B^* \setminus \{i_k\}) \le f_k(B^*)$. Since $f_k(B^*) < h_k$ and $h_{k-1} \ge h_k$, $f_{k-1}(B^* \setminus \{i_k\}) < h_{k-1}$ and $(B^* \setminus \{i_k\}) \in \overline{\mathcal{R}}_{k-1}$. Consider Step 2 at stage

$k-1$. The expansion of cluster $(B^* \setminus \{i_k\}) \in \overline{\mathcal{R}}_{k-1}$ produces B^* that is added to T_k since $f_k(B^*) < h_{k-1}$ and this show the contradiction.

2. $i_k \notin B^*$. Due to property P1 and $h_{k-1} \geq h_k$, $f_{k-1}(B^*) < h_{k-1}$ and, hence, $B^* \in \overline{\mathcal{R}}_{k-1}$. Therefore, at Step 2, the cluster B^* is added to T_k and this shows the contradiction.

3.5 METHODS FOR SOLVING THE CPMP

In this section, two methods for solving the CPMP are described. The first, called EHP, consists of reducing the number of variables of the integer program SP so that the resulting problem can be solved by an integer programming solver (e.g. CPLEX). The second is the branch and bound method proposed by Pirkul (1987) which has been implemented in order to compare the computational performance of algorithm EHP.

3.5.1 THE EHP PROCEDURE

Whenever procedure H^2 ends without having found the optimal solution, it is necessary, as described in Section 3.2, to solve the following problem SP$'$:

$$\text{(SP}'\text{)} \qquad z(\text{SP}') = Min \sum_{\ell \in \mathcal{B}'} c_\ell x_\ell$$

$$subject\ to \quad \sum_{\ell \in \mathcal{B}_i \cap \mathcal{B}'} x_\ell = 1, \qquad \forall i \in N$$

$$\sum_{\ell \in \mathcal{B}'} x_\ell = p$$

$$x_\ell \in \{0,1\}, \forall \ell \in \mathcal{B}'$$

where set $\mathcal{B}'$ is computed according to expression (3.11). However, the size of $\mathcal{B}'$ may be still too large. In this case, set $\mathcal{B}'$ is replaced with a subset $\mathcal{F} \subseteq \mathcal{B}'$ in such a way that the resulting problem becomes solvable by an integer programming code. The solution achieved might not be an optimal CPMP solution, but it is possible to evaluate its distance from optimality.

Let $(\mathbf{u}',w')$ be the heuristic solution of DSP of cost $z'(\mathrm{DSP})$ produced by the bounding procedure HDSP. Also let $c'_\ell = c_\ell - \sum_{i\in B_\ell} u'_i - w'$ be the reduced cost of cluster $\ell \in \mathcal{B}$ with respect to $(\mathbf{u}',w')$. Let $\mathcal{F}_j$ be the subset of $\mathcal{P}_j$, $j \in N$, that satisfies conditions (3.18), where the reduced costs $\{c^2_\ell\}$ are replaced with $\{c'_\ell\}$ and $z(\mathrm{DSP}^1)$ is substituted with $z'(\mathrm{DSP})$. The subsets $\mathcal{F}_j$, $j \in N$, are computed by means of procedure GEN($\mathcal{F}_j$) using the dual solution $(\mathbf{u}',w')$ instead of $(\mathbf{u}^1,w^1)$.

Let $\mathbf{x}^*$ be an optimal integer solution, of cost $z^*(\mathrm{SP}')$, of the set partitioning problem SP$'$ where set $\mathcal{B}'$ is replaced by set $\mathcal{F}$ (we assume $z^*(\mathrm{SP}') = \infty$ if set $\mathcal{F}$ does not contain any feasible CPMP solution). Notice that, if $z^*(\mathrm{SP}') < \infty$, the solution $\mathbf{x}^*$ is a feasible CPMP solution. The following recognizes if $\mathbf{x}^*$ is an optimal CPMP solution. Let L_j be a lower bound to the reduced cost of the least reduced cost cluster in the set $\mathcal{P}_j \setminus \mathcal{F}_j$, $j \in N$ and assume that $L_j = \infty$ if $|\mathcal{F}_j| < \Delta$. Let $GAPMIN = \mathop{Min}_{j\in N}[L_j]$. The following two cases exist:

(a) $z^*(\mathrm{SP}') \le z'(\mathrm{DSP}) + GAPMIN$: $\mathbf{x}^*$ is optimal for the CPMP, since any feasible CPMP solution involving some cluster of set $\mathcal{B}\setminus\mathcal{F}$ would have a cost greater or equal than $z'(\mathrm{DSP}) + GAPMIN$.

(b) $z^*(\mathrm{SP}') > z'(\mathrm{DSP}) + GAPMIN$: $\mathbf{x}^*$ might not be an optimal CPMP solution, and $z'(\mathrm{DSP}) + GAPMIN$ is a valid lower bound to the optimal CPMP solution cost.

In constrained clustering, as described in Hansen and Jaumard (1997), additional requirements are imposed on the clusters such as bounds on clusters cardinality, incompatibility between customers, etc. Additional constraints can be easily incorporated both in the bounding procedure H^2 and in the solution method EHP by changing Step 2 of algorithm GEN($\mathcal{F}_j$) to reject any infeasible cluster B.

3.5.2 THE BRANCH AND BOUND METHOD BB

In order to evaluate the computational performance of algorithm EHP we implemented the Branch and Bound procedure (BB) proposed by Pirkul (1987). BB makes use of the lower bound LR(λ) obtained from formulation F (see Section 2.2) by relaxing the partitioning constraints (2.2) in a Lagrangean fashion (see Section 3.3.1). The tree-search is a two level binary tree-search in which the first level of the tree is formed by fixing variables ξ_{jj}, $j = 1,\dots,n$, which define the medians of the solution. Whenever a leaf node of this top level tree is reached, it can be treated as the root of a new sub-tree which is explored by fixing variables ξ_{ij} $i, j = 1,\dots,n, i \neq j$, which correspond to assigning the customers to the medians defined at the top level of the tree. At each tree node, a lower bound is computed by means of the subgradient procedure applied to the Lagrangean relaxation LR(λ). The values of the multipliers λ are initialized with the penalties associated with the lower bound found at the predecessor node. 50 subgradient iterations are carried out at each tree node, with the exception of the root node where 300 subgradient iterations are performed.

3.5.3 A NUMERICAL EXAMPLE

In this section we describe a numerical example to illustrate procedure EHP. The test problem chosen for the example was CCPX16 (see Section 3.6). The number of customers is n=100, among which p=10 must be chosen as medians. The capacity Q of each median is equal to 120 and the total demand of customers (i.e. $\sum_{i \in N} q_i$) is equal to 1060. The data corresponding to this problem test can be found in Appendix A.1. The upper bound z(UB) was set equal to 955, that is, the cost of the heuristic solution found by the Bionomic algorithm (see Section 2.7), plus 1.

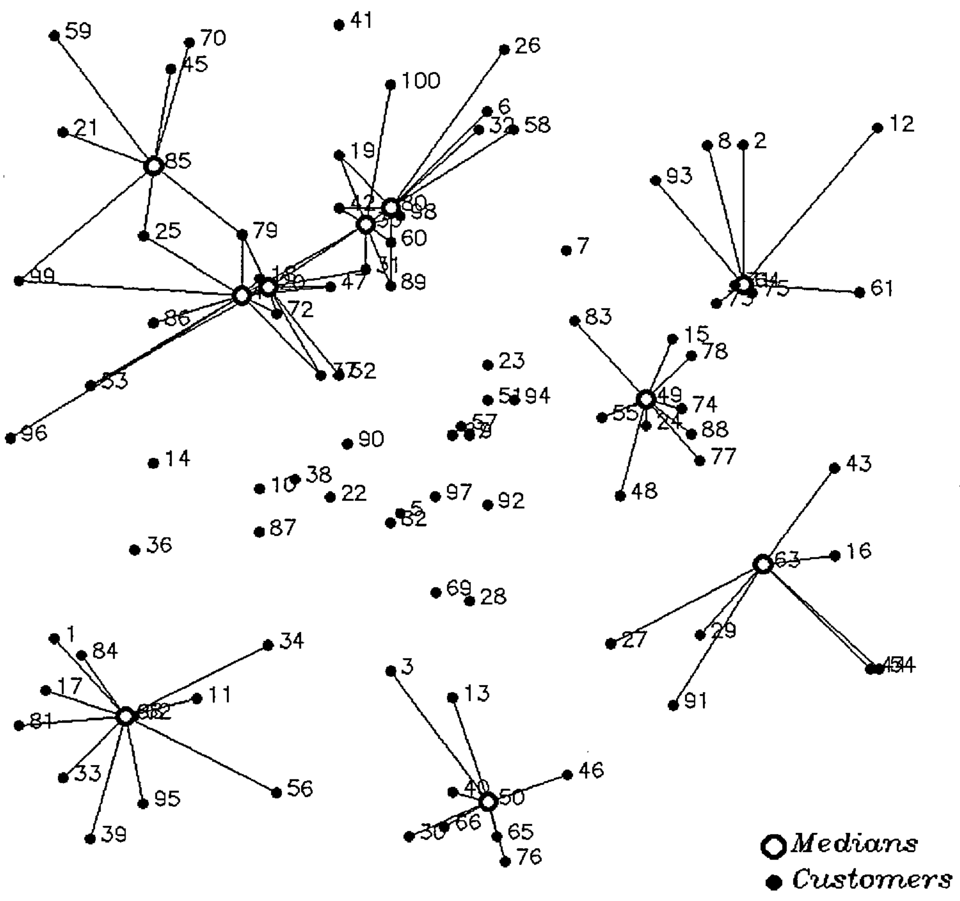


Figure 3.1. Example: solution to procedure H^1 of cost 950.2

Table 3.1. Example: details of the lower bound obtained by procedure H^1

	Median	Load	Cluster
1	68	109	{1,11,17,33,34,39,56,62,68,81,84,95}
2	49	87	{15,24,48,49,55,74,77,78,83,88}
3	64	107	{2,8,12,61,64,71,73,75,93}
4	63	100	{16,27,29,43,44,54,63,91}
5	20	110	{4,18,20,31,37,47,52,53,72,79,86}
6	50	85	{3,13,30,40,46,50,65,66,76}
7	35	117	{4,19,20,31,35,42,60,80,89,98,100}
8	4	118	{4,18,20,25,37,47,53,72,79,86,96,99}
9	80	117	{6,19,26,32,35,42,58,60,80,89,98}
10	85	94	{21,25,45,59,70,79,85,99}

The lower bound $z(\text{DSP}^1)$ was computed by performing 300 iterations of procedure H^1 using an initial value of the step size ε equal to 3.0. A value equal to 950.2 was obtained within 3.2 seconds on a Silicon Graphics Indy machine (MIPS R4400/200 MHz processor). Table 3.1 reports the details of the lower bound computation and Figure 3.1 shows the corresponding solution.

The value $z(\text{DSP}^2)$ of the lower bound obtained by procedure H^2 was equal to 1.1, hence, the value of the final lower bound obtained by procedure HDSP was 951.3. The number of clusters (i.e. $|\mathcal{F}|$) generated by procedure GEN($\mathcal{F}_j$) (see Section 3.4) was 2311. The total computing time of procedure HDSP was 4.9 seconds. Table 3.2 reports the details of the lower bound solution produced by H^2 and Figure 3.2 graphically displays the corresponding solution.

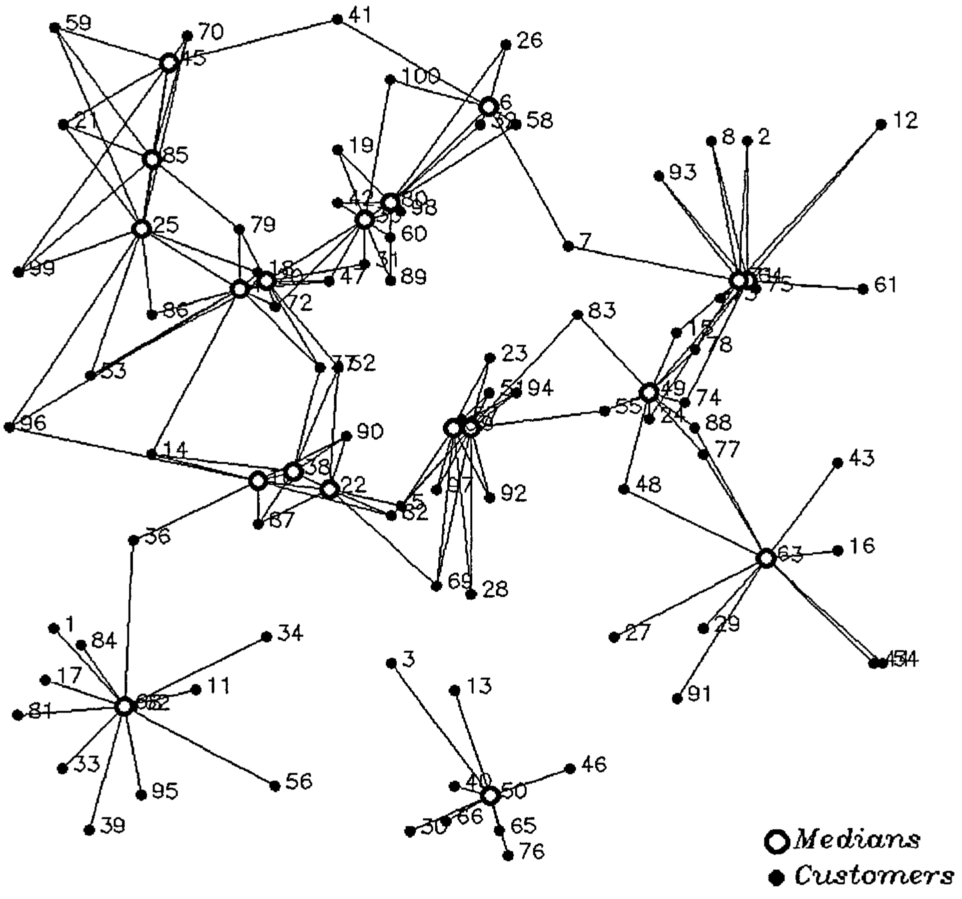


Figure 3.2. Example: solution to procedure H^2 of cost 1.1

Table 3.2. Example: details of the lower bound obtained by procedure H^2

$\{x_\ell\}$ variables				
Median	Load	Value	Coefficient	Cluster
4	118	0.33	0.10	{96,25,47,14,53,79,37,86,20,72,18,4}
6	71	0.67	0.00	{7,41,100,32,58,26,6}
9	120	0.33	0.00	{55,69,28,97,82,92,23,94,67,51,57,9}
9	111	0.33	0.10	{83,28,5,97,92,23,94,67,51,57,9}
10	120	0.33	0.00	{82,96,36,90,22,38,14,87,10}
20	110	0.33	0.10	{53,52,31,47,86,79,37,72,4,18,20}
22	117	0.33	0.00	{52,69,5,82,90,38,87,10,22}
25	113	0.33	0.10	{4,18,70,96,53,59,45,21,86,85,99,25}
35	113	0.33	0.00	{20,47,89,19,42,31,98,80,60,35}
35	119	0.33	0.00	{72,100,89,19,42,31,98,80,60,35}
38	113	0.33	0.00	{52,37,14,90,87,22,10,38}
45	91	0.33	0.80	{99,41,85,21,59,70,45}
49	87	0.67	0.00	{78,15,77,83,48,55,88,74,24,49}
50	85	1.00	0.00	{3,13,46,30,76,66,65,40,50}
63	111	0.33	0.00	{48,27,91,43,29,16,54,44,63}
63	115	0.33	0.00	{88,27,91,43,29,16,54,44,63}
63	117	0.33	0.00	{77,27,91,43,29,16,54,44,63}
64	118	0.33	0.00	{74,93,8,73,2,61,71,75,12,64}
64	120	0.33	1.10	{49,78,93,8,73,2,61,71,75,12,64}
67	115	0.33	0.00	{69,28,92,97,5,94,23,51,9,57,67}
68	109	0.33	0.00	{34,56,81,1,17,84,33,39,11,95,62,68}
68	115	0.67	0.00	{36,34,56,81,1,17,84,33,39,11,95,62,68}
71	118	0.33	0.80	{7,24,15,93,8,73,61,2,75,12,64,71}
80	117	0.33	0.20	{26,32,58,89,42,6,19,35,60,98,80}
85	94	0.33	0.00	{79,25,99,70,21,59,45,85}

The cost of the integer solution found by CPLEX 4.0 was 954, obtained in 1.44 seconds. The cardinality of the set of clusters generated (i.e. $|\mathcal{F}|$) was 1160. The value of *GAPMIN* was equal to 4.8. Therefore, the solution found was also the optimal solution of the problem since $954 \leq 951.3 + 4.8$ (=956.1). The total computing time of procedure EHP was 6.4 seconds. Table 3.3 reports the details of the optimal solution found which is presented graphically in Figure 3.3.

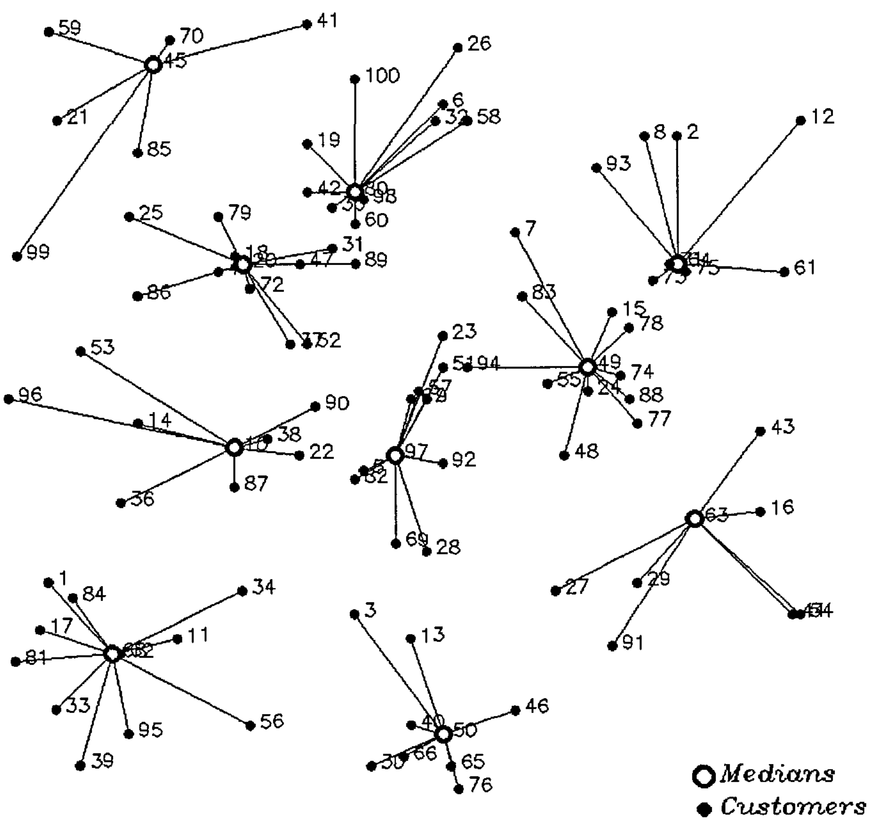


Figure 3.3. Example: optimal solution found by procedure EHP

Table 3.3. Example: details of the optimal solution of cost 954

	Median	Load	Cluster
1	10	114	{10,14,22,36,38,53,87,90,96}
2	20	118	{4,18,20,25,31,37,47,52,72,79,86,89}
3	49	108	{7,15,24,48,49,55,74,77,78,83,88,94}
4	80	117	{6,19,26,32,35,42,58,60,80,98,100}
5	68	109	{1,11,17,33,34,39,56,62,68,81,84,95}
6	63	100	{16,27,29,43,44,54,63,91}
7	64	107	{2,8,12,61,64,71,73,75,93}
8	45	91	{21,41,45,59,70,85,99}
9	50	85	{3,13,30,40,46,50,65,66,76}
10	97	111	{5,9,23,28,51,57,67,69,82,92,97}

This test problem has also been solved to optimality using the branch and bound method BB described in the previous section. The total computing time of the branch and bound method BB was 1912 seconds and the total number of nodes in the tree search was 2690.

3.6 COMPUTATIONAL RESULTS

Algorithms EHP and BB described in Section 3.5 have been coded in Fortran 77 and run on a Silicon Graphics Indy machine (MIPS R4400/200 MHz processor) on four classes of test problems, called A, B, C, D, respectively. Classes A and B are drawn from the literature and classes C and D are two new sets of test problems generated in order to evaluate the performance of EHP on CPMP instances with additional constraints. CPLEX 4.0 is used as the LP-solver in procedure H^2 and as the integer programming solver in EHP.

The problems of classes A and B correspond to the CPMP instances used by Osman and Christofides (1994); set A contains 10 problems of size n=50 and p=5 while set B contains 10 problems of size n=100 and p=10. In these two sets of problems the dissimilarity matrices correspond to Euclidean distance matrices. Both test problem classes A and B were used to evaluate the computational performance of the Bionomic Algorithm for the CPMP in Chapter 2.

The problems of classes C and D are derived from problems of class A by imposing additional constraints on the clusters. These constraints are bounds on the cluster *cardinality* and *incompatibilities* between customers. The incompatibilities are defined by an incompatibility matrix $[\iota_{kj}]$ where $\iota_{kj} = 1$ if customer k cannot be in the same cluster of customer j, 0 otherwise. We impose a maximum cluster cardinality of 6 for problems of class C and 11 for problems of class D. The incompatibility matrices are generated by randomly defining 5 incompatibilities in such a way that the optimal CPMP solution of the corresponding class A problem becomes infeasible (see Table 3.4). For solving problems of classes C and D, algorithm BB has been modified by changing the branching strategy of the second level of the tree so that branches that lead to infeasible solutions are rejected.

Figure 3.4 shows examples of CPMP solutions with additional constraints. Figure 3.4(a) presents the optimal solution of problem CCPX5 of class A. Figures 3.4(b) and 3.4(c) display CPMP solutions obtained by considering 5 incompatibilities between customers and by imposing a maximum cardinality of 6 and of 11 for problem C5 of class C (Figure 3.4(b)) and for problem D5 of class D (Figure 3.4(c)), respectively. In the figures, the customers with the same letter (A, B, C, D or E) are incompatible, i.e. cannot be in the same cluster.

Table 3.4. Problem classes C and D: incompatibilities between customers

Problems	Incompatibilities
C1,D1	{3,7} {8,43} {11,41} {19,47} {33,48}
C2,D2	{1,7} {3,19} {10,37} {11,32} {20,50}
C3,D3	{2,25} {5,49} {8,39} {14,45} {27,50}
C4,D4	{2,42} {9,46} {10,17} {11,16} {24,48}
C5,D5	{3,12} {6,48} {7,9} {18,31} {21,44}
C6,D6	{2,8} {4,14} {10,46} {12,47} {18,45}
C7,D7	{2,6} {4,18} {10,36} {15,49} {27,50}
C8,D8	{3,8} {4,6} {9,49} {22,44} {23,46}
C9,D9	{2,15} {8,42} {9,41} {13,50} {17,40}
C10,D10	{1,6} {5,41} {10,46} {15,50} {28,45}

The results obtained are presented in tables 3.5 to 3.8. The columns in these tables are defined as follows:

Probl.:	a problem instance identifier.
z(UB):	cost of the best CPMP solution found by the heuristic algorithms of Chapter 3 (see Section 2.7), plus 1. The upper bounds for problems of classes C and D were computed using the heuristic algorithm MB1 (see Section 2.3.1.1) which has been modified in order to incorporate the additional constraints.
$z^*(\mathrm{SP}')$:	cost of the optimal CPMP solution found by algorithm EHP (or cost of the best solution found).
$z(\mathrm{DSP}^1)$:	lower bound produced by procedure H^1 after 300 subgradient iterations.
t_{H^1}:	computing time of bounding procedure H^1.
$\%\mathrm{E}_{\mathrm{DSP}^1}$:	percentage error of the lower bound $z(\mathrm{DSP}^1)$ (i.e. $\%\mathrm{E}_{\mathrm{DSP}^1}=100z(\mathrm{DSP}^1)/z^*(\mathrm{SP}')$).

z'(DSP):	final lower bound produced by procedure HDSP.
t_{HDSP}:	total computing time of procedure HDSP.
$\%\mathrm{E}_{\mathrm{HDSP}}$:	percentage error of the lower bound z'(DSP) (i.e. $\%\mathrm{E}_{\mathrm{HDSP}}=100z'(\mathrm{DSP})/z^*(\mathrm{SP}')$).
LS:	$= z'(\mathrm{DSP}) + GAPMIN$, where *GAPMIN* is the value defined in Section 3.5.1. (The solution of cost $z^*(\mathrm{SP}')$ produced by EHP is optimal if $z^*(\mathrm{SP}') \leq LS$).
$\lvert\mathcal{F}\rvert$:	number of clusters generated in EHP.
t_{EHP}:	total computing time of procedure EHP including t_{HDSP}.
$z^*(\mathrm{BB})$:	cost of the optimal CPMP solution found by algorithm BB (or cost of the best solution found).
t_{BB}:	computing time of algorithm BB.

A time limit of 3600 CPU seconds has been imposed in both the EHP and BB algorithms for all test problems. The parameter Δ, used in GEN($\mathcal{F}_j$), has been set to 2000 in both procedures H^2 and EHP for all test problems.

Table 3.5 shows that most of the problems of class A are relatively easy, with the only exception of problem CCPX8. In fact, bound $z(\mathrm{DSP}^1)$ is already close to the optimum and bound z'(DSP) is capable of finding four optimal solutions. The good quality of the bound enables both EHP and BB to be very effective on all problems, except CCPX8. Note that on this set, BB has been able, within the time limit of 3600 sec., to prove the optimality of all instances, while EHP was unable to find an optimal solution for CCPX8.

Table 3.6 shows the results of problems of class B. For these problems, z'(DSP) does not improve much over $z(\mathrm{DSP}^1)$, $z(\mathrm{DSP}^1)$ being already very close to the optimal solution cost on all instances. Algorithm EHP proves to be superior to BB in fact it is able to prove the optimality of eight solutions, while BB is able to solve to optimality only four problems. Furthermore EHP finds an improved solution of problem CCPX18 without proving its optimality, since its cost is higher than the corresponding *LS* value.

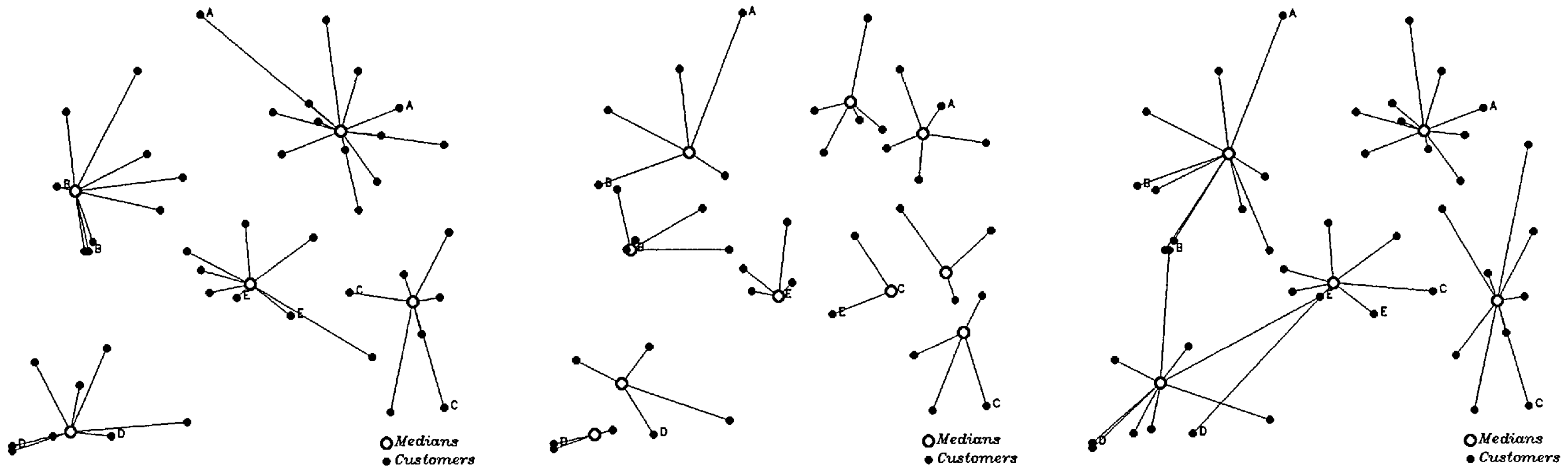
Algorithm BB cannot improve the upper bound of six out of the ten instances. Moreover, EHP finds the optimal solution in a much smaller computing time.

Table 3.7 presents the results for the instances of class C that include additional constraints. Algorithm EHP is able to prove the optimality of nine solutions out of ten, while BB can solve only one problem. Notice that the quality of lower bound $z(DSP^1)$ is much worse than on the class A problems (from which the class C problems are drawn) and that $z'(DSP)$ makes an important contribution in filling the gap $z^*(SP')$-$z'(DSP)$.

Table 3.8 shows the results for problems of class D. The problems of class D are harder than the corresponding ones of class C for both algorithms EHP and BB. This reflects a worse quality of lower bound H^1 even though $z'(DSP)$ continues to contribute significantly over $z(DSP^1)$. EHP finds, for all problems, a better solution than the initial upper bound, but cannot prove the optimality of any of the solutions obtained. Algorithm BB, on the other hand, is unable to improve over the initial upper bound for any of the ten instances.

3.7 SUMMARY

In this chapter, a new method for the Capacitated p-Median Problem (CPMP) based on a Set Partitioning formulation of the problem has been presented. A valid lower bound to the optimal solution cost is obtained by combining two different heuristic methods for solving the dual of the LP-relaxation of the exact formulation. The dual solution obtained is used for generating a reduced set partitioning problem that can be solved by an integer programming solver. The solution achieved might not be an optimal CPMP solution, however the new method allows us to estimate its maximum distance from optimality. The computational performance of the new exact algorithm has been evaluated on two classes of test problems proposed in the literature and on two new sets of difficult CPMP instances with additional constraints. The results show that the exact algorithm is able to solve exactly CPMP's of size up to 100 customers.



(a) Solution of Problem CCPX5 (b) Solution of Problem C5 (c) Solution of Problem D5

Figure 3.4: Example of CPMP solutions with additional constraints

Table 3.5. Computational results of problem class A

		HDSP						EHP				BB	
Probl.	z(UB)	z(DSP^1)	t_{H1}	%E$_{DSP1}$	z'(DSP)	t_{HDSP}	%E$_{HDSP}$	z^*(SP$'$)	LS	$\|\mathcal{F}\|$	t_{EHP}	z^*(BB)	t_{BB}
CCPX1	714	704.2	0.9	98.8	705.0	1.4	98.9	713	715	570	1.7	713	5.4
CCPX2	741	739.5	0.3	99.9	740.0	0.4	100.0	740 a	742	10	0.4	740	0.8
CCPX3	752	748.2	0.7	99.6	749.0	0.9	99.7	751	753	64	1.0	751	2.4
CCPX4	652	650.2	0.4	99.9	651.0	0.6	100.0	651 a	653	23	0.6	651	0.7
CCPX5	666	663.1	0.5	99.9	664.0	0.7	100.0	664 a	667	19	0.7	664	4.1
CCPX6	779	777.6	0.4	100.0	778.0	0.5	100.0	778 a	780	41	0.6	778	2.0
CCPX7	788	778.1	1.7	98.9	778.3	2.6	98.9	787	788	1915	4.7	787	29.3
CCPX8	821	770.8	1.9	94.0	772.1	26.7	94.2	821 c	789	30491	3600.0	820	1990.3
CCPX9	716	712.1	1.2	99.6	712.6	1.6	99.7	715	717	355	1.8	715	14.3
CCPX10	830	815.1	2.3	98.3	818.4	4.5	98.7	829	834	2511	12.2	829	288.5
Averages			1.0	98.9		4.0	99.0				362.4		233.8

(a) optimal solution obtained by procedure HDSP.

(c) no solution found of cost smaller than z(UB).

Table 3.6. Computational results of problem class B

		HDSP						EHP				BB	
Probl.	z(UB)	z(DSP[1])	t_{H1}	%E_{DSP1}	z'(DSP)	t_{HDSP}	%E_{HDSP}	z^*(SP′)	LS	$\|\mathcal{F}\|$	t_{EHP}	z^*(BB)	t_{BB}
CCPX11	1007	1000.1	3.6	99.4	1001.1	6.4	99.5	1006	1008	2551	11.7	1006	1099.3
CCPX12	967	958.2	4.1	99.2	958.5	11.6	99.2	966	968	17458	832.8	967 c	3600.0
CCPX13	1027	1021.1	2.8	99.5	1021.6	3.7	99.6	1026	1028	995	5.1	1026	88.3
CCPX14	983	971.1	4.2	98.9	971.8	12.0	99.0	982	984	17362	304.3	983 c	3600.0
CCPX15	1092	1079.1	4.8	98.9	1079.8	16.6	99.0	1091	1092	19394	558.4	1092 c	3600.0
CCPX16	955	950.2	3.2	99.6	951.3	4.9	99.7	954	956	1160	6.4	954	1911.6
CCPX17	1035	1024.1	4.7	99.0	1025.3	11.8	99.2	1034	1036	10252	280.5	1035 c	3601.0
CCPX18	1044	1031.2	4.3	98.9	1031.7	15.9	98.9	1043 b	1040	23211	1137.8	1044 c	3600.0
CCPX19	1032	1025.1	4.3	99.4	1026.3	7.5	99.5	1031	1033	3105	22.2	1031	3115.8
CCPX20	1006	972.2	6.7	96.6	972.8	77.3	96.7	1006 c	984	62872	3600.0	1013 c	3600.0
Averages			4.3	99.0		16.8	99.0				675.9		2781.8

(b) this solution was not proved to be optimal.

(c) no solution found of cost smaller than z(UB).

Table 3.7. Computational results of problem class C

		HDSP						EHP				BB	
Probl.	z(UB)	z(DSP^1)	t_{H1}	%E$_{DSP1}$	z'(DSP)	t_{HDSP}	%E$_{HDSP}$	z^*(SP$'$)	LS	$\|\mathcal{F}\|$	t_{EHP}	z^*(BB)	t_{BB}
C1	477	423.2	1.5	88.9	435.0	72.9	91.4	476	476	46392	2644.6	477 c	3600.0
C2	519	472.2	1.8	97.6	480.3	2.6	99.2	484	486	240	2.7	484	909.0
C3	504	440.1	1.7	87.5	468.6	35.2	93.2	503 b	499	26695	74.8	504 c	3600.0
C4	464	422.1	1.9	94.0	441.8	8.7	98.4	449	468	809	9.9	464 c	3600.0
C5	521	474.1	2.0	96.8	490.0	3.3	100.0	490	492	91	3.3	491 b	3600.0
C6	564	513.2	1.8	92.5	544.9	20.9	98.2	555	572	1312	23.5	564 c	3600.0
C7	536	487.3	2.6	92.1	501.0	26.0	94.7	529	531	16714	369.2	536 c	3600.0
C8	479	436.1	1.9	93.4	461.9	5.6	98.9	467	469	298	5.7	479 c	3600.0
C9	497	452.3	2.8	91.9	489.7	21.1	99.5	492	513	269	21.4	497 c	3600.0
C10	544	495.1	2.5	94.5	512.5	5.2	97.8	524	526	709	5.7	544 c	3600.0
Averages			2.0	92.9		20.1	97.1				316.1		3330.9

(b) this solution was not proved to be optimal.

(c) no solution found of cost smaller than z(UB).

Table 3.8. Computational results of problem class D

		HDSP						**EHP**				**BB**	
Probl.	z(UB)	z(DSP^1)	t_{H1}	%E$_{DSP1}$	z'(DSP)	t_{HDSP}	%E$_{HDSP}$	z^*(SP′)	LS	$\lvert\mathcal{F}\rvert$	t_{EHP}	z^*(BB)	t_{BB}
D1	819	704.3	1.5	86.1	738.7	126.1	90.3	818 b	763	39487	478.2	819 c	3600.0
D2	835	739.3	1.6	93.8	758.7	68.4	96.3	788 b	771	39837	85.9	835 c	3600.0
D3	857	747.4	1.7	87.3	768.7	138.6	89.8	856 b	787	46250	249.0	857 c	3600.0
D4	734	650.1	1.6	90.5	682.3	81.5	95.0	718 b	697	37240	94.7	734 c	3600.0
D5	776	663.2	2.2	85.6	723.9	108.3	93.4	775 b	742	39930	126.1	776 c	3600.0
D6	878	777.3	1.6	91.6	810.9	87.2	95.5	849 b	831	41040	172.9	878 c	3600.0
D7	878	777.2	2.2	91.2	818.4	113.6	96.1	852 b	832	46663	131.4	878 c	3600.0
D8	870	770.4	1.9	90.3	801.6	108.0	94.0	853 b	818	44253	224.3	870 c	3600.0
D9	804	712.1	2.0	92.7	736.4	85.1	95.9	768 b	746	40311	170.8	804 c	3600.0
D10	922	816.2	4.2	91.2	845.9	111.1	94.5	895 b	869	46395	568.5	922 c	3600.0
Averages			4.2	90.0		102.8	94.1				230.2		3600.0

(b) this solution was not proved to be optimal.

(c) no solution found of cost smaller than z(U)

CHAPTER 4

NEW EXACT ALGORITHMS FOR ROUTING PROBLEMS BASED ON A TWO-COMMODITY NETWORK FLOW FORMULATION

4.1 INTRODUCTION

Routing problems require the determination of optimal sequences subject to a given set of constraints. The best known problem of this type is the classical Traveling Salesman Problem (TSP), calling for a minimum cost Hamiltonian cycle on a given graph. Another well known routing problem is the Vehicle Routing Problem (VRP) that involves the optimization of the distribution of goods from a single depot to a given set of customers with known demands using a given number of vehicles of fixed capacity. Both the TSP and the VRP play a central role in distribution planning and have been studied extensively over the past four decades. For the TSP, see the book edited by Lawler et al. (1985), Laporte (1992a) and Jünger et al. (1995). For the VRP, see Magnanti (1981), Bodin et al. (1983), Christofides (1985), Golden and Assad (1986,1988), Bodin (1990), Laporte (1992b) and Fisher (1995). See also the recent bibliographies by Laporte and Osman (1995) and by Laporte (1997).

The TSP has been proven to be NP-hard by Karp (1972). The VRP is a generalization of the TSP and is also a NP-hard problem.

In this chapter we investigate new integer programming formulations for routing problems which are based on the two-commodity network flow formulation of the TSP described by Finke et al. (1984). This formulation is interesting in many different ways. It can be shown that its LP-relaxation satisfies a weak form of the subtour elimination constraints. The formulation can also be modified to accommodate different constraints and, therefore, is capable of being extended to different routing problems. The two-commodity formulation has been used by Lucena (1986) to derive new lower bounds for the VRP and by Langevin et al. (1993) for solving the TSP and the Makespan Problem with time windows. In this chapter, we use the two-commodity approach to derive new integer programming formulations for the VRP, the TSP with mixed deliveries and collections (TSPDC) and the TSP with Backhauls (TSPB). New lower bounds are obtained from the linear relaxation of these formulations which are further strengthened by new valid inequalities.

The chapter is organized as follows. In Section 4.2 the original two-commodity network flow formulation of Finke et al. (1984) is described and a new formulation for the symmetric TSP is presented. In Section 4.2, a lower bound for the TSP based on the LP-relaxation of the new formulation strengthened by valid inequalities is described. In Sections 4.3 the new TSP formulation is extended to derive a new integer programming formulation and an exact branch and cut algorithm for the VRP. In Section 4.4 new formulations and exact branch and cut algorithms for both TSPDC and TSPB are described. In Sections 4.3 and 4.4, computational results of the new algorithms are also presented. Finally, conclusions are presented in Section 4.5.

4.2 A TWO-COMMODITY FORMULATION OF THE TSP

Let $G=(V,A)$ be a directed graph where $V=\{1,2,\ldots,n\}$ denotes the set of vertices and A the set of arcs. A non-negative cost c_{ij} is associated with each arc $(i,j)\in A$. The TSP is the problem of finding a minimum cost Hamiltonian circuit on graph G.

Finke et al. (1984) introduced the following two-commodity network flow formulation for the TSP.

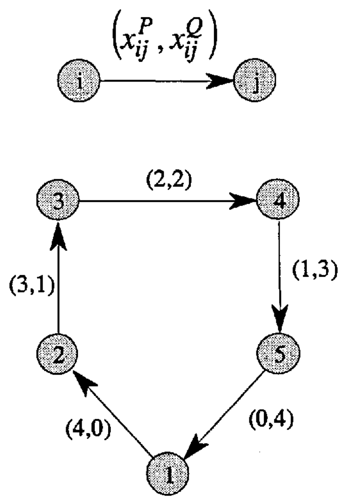


Figure 4.1. Flows in the Hamiltonian circuit

For each $i \in V$, the amount p_i of a commodity P and the amount q_i of a commodity Q are defined as follows:

$$p_i = \begin{cases} (n-1) & \text{for } i = 1 \\ -1 & \text{otherwise} \end{cases} \tag{4.1}$$

and

$$q_i = \begin{cases} -(n-1) & \text{for } i = 1 \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

The idea behind the formulation is that the salesman, when traversing a Hamiltonian circuit, should always carry with him, through any arc, the same total combined amount, $n-1$, of the two-commodities. Let us suppose, for instance, that his tour starts from vertex 1, with $n-1$ units of P and 0 units of Q. This makes a total combined amount of $n-1$ units of flow. At the following vertex in the tour, the salesman leaves one unit of P and picks-up one unit of Q, as implied by (4.1) and (4.2). Once again, the total combined amount of flow he will be carrying will be equal to $n-1$ units. Proceeding in this way, the salesman finally arrives back to vertex 1 carrying 0 units of P and $n-1$ units of Q. One interpretation that could be made is that, at any arc of the Hamiltonian circuit, the amount of commodity P represents the number of vertices left to be visited by the salesman. Conversely, the amount of commodity Q represents the number of vertices that have already been visited. This process is illustrated, for a 5-vertex Hamiltonian circuit, in Figure 4.1.

Let x_{ij}^P and x_{ij}^Q be the units of commodity P and commodity Q carried by the salesman in traversing arc $(i,j) \in A$, respectively. Let ξ_{ij} be a binary variable that is equal to 1 if arc $(i,j) \in A$ is in the optimal TSP solution, 0 otherwise.

The formulation of the TSP proposed by Finke et al. (1984) is as follows:

$$\text{(TSP)} \qquad z(\text{TSP}) = Min \sum_{(i,j)\in A} c_{ij}\xi_{ij} \tag{4.3}$$

$$\left.\begin{aligned} &\textit{subject to} \quad \sum_{j\in V} x_{ij}^P - \sum_{j\in V} x_{ji}^P = p_i, && (4.4)\\ &\sum_{j\in V} x_{ij}^Q - \sum_{j\in V} x_{ji}^Q = q_i, && (4.5)\\ &\sum_{j\in V}\left(x_{ij}^P + x_{ij}^Q\right) = n-1, && (4.6)\end{aligned}\right\} \quad \forall i \in V$$

$$\left.\begin{aligned} &x_{ij}^P + x_{ij}^Q = (n-1)\xi_{ij}, && (4.7)\\ &x_{ij}^P \geq 0, x_{ij}^Q \geq 0, && (4.8)\\ &\xi_{ij} \in \{0,1\}, && (4.9)\end{aligned}\right\} \quad \forall (i,j) \in A$$

Constraints (4.4), (4.5) and (4.8) define a feasible flow for commodity P and a feasible flow for commodity Q. Constraints (4.6) and (4.7) ensure that there is exactly one arc leaving each vertex $i \in V$ which carries a combined total flow of $n-1$ units. There is also exactly one leaving arc with $n-1$ units at each vertex since equations (4.4) and (4.5) add up to 0. In addition, there must be a path from vertex 1 to every other vertex j and a path from j back to 1 because of the supply-demand pattern. Hence (4.3)-(4.9) characterize exactly a Hamiltonian circuit of cost $z(\text{TSP})$.

Using the flow variables $\{x_{ij}^P\}$ and $\{x_{ij}^Q\}$ only and eliminating constraints (4.7), the above formulation can be rewritten as follows:

$$\text{(TSP)} \qquad z(\text{TSP}) = Min \frac{1}{(n-1)} \sum_{(i,j)\in A} c_{ij}\left(x_{ij}^P + x_{ij}^Q\right) \tag{4.10}$$

subject to (4.4), (4.5), (4.6), (4.8) and

$$x_{ij}^P + x_{ij}^Q \in \{0, n-1\}, \qquad \forall (i,j) \in A \tag{4.11}$$

Then, a valid lower bound for the TSP can be obtained by replacing the integer conditions (4.11) with the inequalities $x_{ij}^P + x_{ij}^Q \le (n-1)$. However, these constraints are redundant because of (4.6). Consequently, we have the following LP-relaxation:

(LRTSP) $$z(\text{LRTSP}) = Min \ \frac{1}{(n-1)} \sum_{(i,j)\in A} c_{ij}\left(x_{ij}^P + x_{ij}^Q\right)$$

$$\textit{subject to} \ \ (4.4), (4.5), (4.6) \text{ and } (4.8)$$

A comparison of the lower bound derived from the two-commodity formulation LRTSP and from the Assignment Problem (AP) yields Theorem 4.1.

The AP problem associated with the TSP is defined as follows.

(AP) $$z(\text{AP}) = Min \ \sum_{(i,j)\in A} c_{ij}\xi_{ij}$$

$$\textit{subject to} \ \left.\begin{aligned} &\sum_{j\in V}\xi_{ij} = 1, \\ &\sum_{j\in V}\xi_{ji} = 1, \end{aligned}\right\} \ \forall i \in V$$

$$\xi_{ij} \ge 0, \qquad \forall (i,j) \in A$$

Theorem 4.1 (*Finke et al. (1984)*). The two lower bounds for the TSP satisfy the inequality $z(\text{LRTSP}) \ge z(\text{AP})$. Both bounds coincide for indistinguishable commodities *P=Q*.

Proof. Consider the combined flow ξ_{ij} per unit of flow

$$\xi_{ij} = \left(x_{ij}^P + x_{ij}^Q\right)/(n-1). \tag{4.12}$$

Constraints (4.6) imply that $\sum_{j\in V}\xi_{ij} = 1$ and (4.4), (4.5) imply $\sum_{j\in V}\xi_{ji} = 1$. Hence $\xi = [\xi_{ij}]$ is a feasible solution of the assignment problem AP. Thus, $z(\text{LRTSP}) \ge z(\text{AP})$.

Now suppose we have identical commodities *P* and *Q*. Let ξ be an integer valued assignment solution. The arcs with $\xi_{ij} = 1$ form a collection of circuits. Since *P* and *Q* are interchangeable, we have a total requirement of zero units at each vertex. It is therefore easy to find numbers x_{ij}^P and x_{ij}^Q, satisfying $x_{ij}^P + x_{ij}^Q = (n-1)$ whenever

$\xi_{ij} = 1$, which satisfy constraints (4.4), (4.5) and (4.8). Hence $z(\text{LRTSP}) = z(\text{AP})$ for identical commodities *P* and *Q*■

A circuit of graph *G* is called *prehamiltonian* if it passes at least once through all vertices of *G*. A *prehamiltonian graph* is a graph that possesses such a circuit. Consider a feasible solution $\mathbf{x}^P$ and $\mathbf{x}^Q$ of the two-commodity LP-relaxation LRTSP and define its flow supporting graph $G^x = (V, A^x)$ where $A^x = \{(i,j) \in A \mid x_{ij}^P + x_{ij}^Q > 0\}$.

For simplicity, we will use the following notations. For given subsets of nodes $S, S' \subseteq V$, the set $(S : S')$ (cut between *S* and S') is the set of arcs with one end-node in *S* and the other in S' $((S : S') = \{(i,j) \in A : i \in S, j \in S'\})$, $\gamma(S)$ is the set of arcs with both end-nodes in *S* ($\gamma(S) = \{(i,j) \in A : i, j \in S\}$). For any subset *F* of arcs, *x*(*F*) denotes the sum of the ξ_{ij} values over all arcs $(i,j) \in F$.

Theorem 4.2 (*Finke et al. (1984)*). The graph $G^x = (V, A^x)$ is prehamiltonian.

Proof. G^x is prehamiltonian if, and only if, G^x is strongly connected. Suppose $i, j \in V$. A vertex $i \neq 1$ has a supply of one unit of *Q*. Since the vertex 1 is the only sink for *Q*, there must be a path in G^x from *i* to 1. Similarly, a vertex $j \neq 1$ has a requirement of one unit of *P*. Vertex 1 is the only source for *P*, i.e., there is also a path in G^x from 1 to *j*. Hence one obtains a path from *i* to *j* for all pairs of vertices■

Theorem 4.3 (*Finke et al. (1984)*). Let $\mathbf{x}^P$ and $\mathbf{x}^Q$ be a feasible two-commodity flow. Then the associated assignment solution $\xi = (\mathbf{x}^P + \mathbf{x}^Q)/(n-1)$ satisfies the following weak version of subtour elimination constraints:

$$x((W : V - W)) \geq \min(|W|, |V - W|)/(n-1) \tag{4.13}$$

$$x(\gamma(W)) \leq |W| - \min(|W|, |V - W|)/(n-1) \tag{4.14}$$

for all $W \subseteq V$ with $2 \leq |W| \leq n-2$.

Proof. Let $(W, V-W)$ be a partitioning. Suppose that vertex $1 \in W$. The set $V - W$ has a requirement of $|V - W|$ units of commodity *P*. Hence at least $|V - W|$ units are sent from W to $V-W$, i.e., $x((W:V-W)) \geq |V-W|/(n-1)$. Similarly, if $1 \in V-W$, at least the total supply of $|W|$ units of Q has to be sent out of W. Hence $x((W:V-W)) \geq |W|/(n-1)$. Thus (4.13) is valid. Inequality (4.14) is an immediate consequence of (4.13)∎

An optimal integer solution to the assignment problem corresponds to a collection of circuits or subtours. One characteristic property of this configuration, including the case of a single (Hamiltonian) circuit is the symmetry

$$x((W:V-W)) = x((V-W:W)) \tag{4.15}$$

for all subsets $W \subseteq V$.

Theorem 4.4 (*Finke et al. (1984)*). The associated assignment solution $\xi = \left(\mathbf{x}^P + \mathbf{x}^Q\right)/(n-1)$ possesses the symmetry property.

Proof. The theorem is true for the trivial cases $|W| \leq 1$ and $|W| \geq n-1$. For the remaining subsets W one may assume that $1 \in V-W$. Then there are exactly $|W|$ more units of P entering W than there are units of P leaving W. Similarly, there is exactly an excess of $|W|$ units of Q leaving W. Hence we have the same total two-commodity flow in both directions∎

4.2.1 A NEW FORMULATION FOR THE SYMMETRIC TSP

In this section we describe a new integer programming formulation for the symmetric TSP (STSP) based on the two-commodity flow formulation of Finke et al. (1984) described in the previous section.

The formulation of Finke et al. for the STSP would require to define a directed graph containing two arcs ((i,j) and (j,i)) for each edge $\{i,j\} \in E$. In this case the formulation of Finke et al. requires $2n(n\text{-}1)$ variables $\{x_{ij}^P\}$ and $\{x_{ij}^Q\}$, and $n(n\text{-}1)$ variables $\{\xi_{ij}\}$.

The new formulation we are going to describe requires half the number of variables of the Finke et al. formulation.

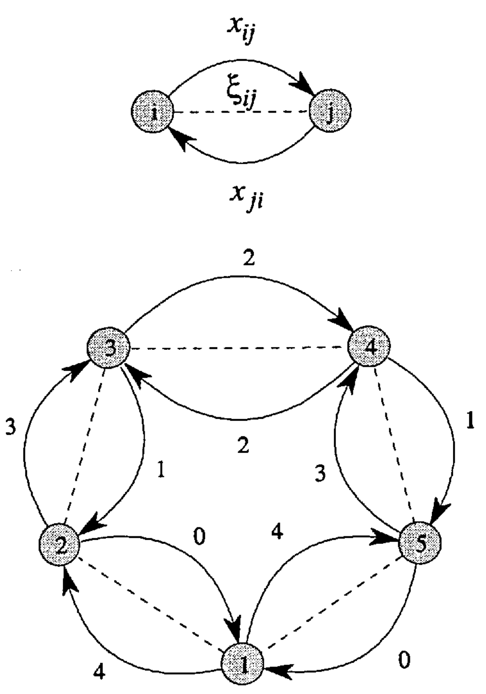


Figure 4.2. The symmetric TSP

Consider the STSP on the complete undirected graph $G=(V,E)$, with vertex set $V=\{1,2,\ldots,n\}$ and edge set $E=\{\{i,j\}:i,j\in V,i<j\}$.

Let ξ_{ij} be a 0-1 binary variable equal to 1 if edge $\{i,j\}\in E$ is in solution, 0 otherwise. Let x_{ij} be the flow value of arc (i,j), $i,j\in V,i\neq j$. The new formulation is obtained from the observation that the flows of both commodities P and Q carried by the salesman in traversing edge $\{i,j\}\in E$ can be represented by the variables x_{ij} and x_{ji}, respectively, as shown in the example of Figure 4.2.

For simplicity, we will use the following notation. For given subset of nodes $S\subseteq V$, the set $\delta(S)$ (*coboundary* of S) is the set of edges with one end-node in S and the other in $V\setminus S$ ($\delta(S)=\{\{i,j\}\in E:i\in S,j\in V-S\}$), $\gamma(S)$ is the set of edges with both end-nodes in S ($\gamma(S)=\{\{i,j\}\in E:i,j\in S\}$). For any subset F of edges, $x(F)$ denotes the sum of the ξ_{ij} values over all edges $\{i,j\}\in F$.

The new mathematical formulation for the STSP is as follows:

(STSP) $z(\text{STSP}) = Min \sum_{\{i,j\}\in E} c_{ij}\xi_{ij}$ (4.16)

$$\text{subject to} \quad \left.\begin{aligned} &\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = -2, && i \neq 1 \quad (4.17) \\ &\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 2(n-1), && i = 1 \quad (4.18) \end{aligned}\right\} \forall i \in V$$

$$x_{ij} + x_{ji} = (n-1)\xi_{ij}, \qquad \forall\{i,j\}\in E \tag{4.19}$$

$$x(\delta(i)) = 2, \qquad \forall i \in V \tag{4.20}$$

$$x_{ij} \geq 0, \qquad \forall i,j \in V, i \neq j \tag{4.21}$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall\{i,j\}\in E \tag{4.22}$$

Constraints (4.17), (4.18) and (4.21) define a feasible flow for variables $\{x_{ij}\}$ and the supply-demand pattern impose that there is a path from vertex 1 to every other vertex *j* and a path from *j* back to 1. Constraints (4.19) and (4.20) force the degree of each vertex to be 2. Constraints (4.22) are the integrality constraints. Hence (4.16)-(4.22) characterize exactly a Hamiltonian cycle of cost $z(\text{STSP})$.

Note that formulation STSP can be written in terms of variables $\{x_{ij}\}$ only since each ξ_{ij} can be replaced by $(x_{ij} + x_{ji})/(n-1)$. Hence, the number of variables of the STSP formulation is $n(n-1)$ which is half the number of variables of the original formulation for the STSP. Similarly, the number of constraints has been reduced from $3n$ (original formulation) to $2n$ (STSP formulation).

A valid lower bound for the TSP can be obtained by replacing the integer conditions (4.22) with the inequalities $x_{ij} + x_{ij} \leq (n-1)$. Consequently, we have the following LP-relaxation:

(LRSTSP) $z(\text{LRSTSP}) = Min \dfrac{1}{(n-1)} \sum_{\{i,j\}\in E} c_{ij}(x_{ij} + x_{ij})$

subject to (4.17), (4.18), (4.20), (4.21) and

$$x_{ij} + x_{ji} \leq (n-1), \ \forall\{i,j\}\in E$$

4.2.2 VALID INEQUALITIES

In this section we describe some valid inequalities that are satisfied by any feasible integer solution of the STSP formulation, but that are not necessarily satisfied by a feasible LRSTSP solution. Hence, the lower bound derived from the LP-relaxation can be further improved by introducing such violated inequalities.

Trivial inequalities

The inequalities $0 \le \xi_{ij} \le 1$, for $\{i,j\} \in E$, are referred to as the *trivial inequalities.*

Flow inequalities

Consider $x_{ij} + x_{ji} = (n-1)\xi_{ij}$, $\forall \{i,j\} \in E, i \ne 1, j \ne 1$. In any feasible integer solution, if $\xi_{ij} = 1$, then $x_{ij} \ge 1$ and $x_{ji} \ge 1$, while in a feasible LRSTSP solution, we might have $\xi_{ij} > 0$ and either $x_{ij} = 0$ or $x_{ji} = 0$. Therefore in LRSTSP we can impose the following constraints.

$$\left.\begin{aligned} x_{ij} &\ge \xi_{ij} \\ x_{ji} &\ge \xi_{ij} \end{aligned}\right\} \quad \forall \{i,j\} \in E, i \ne 1, j \ne 1$$

or, using equation $x_{ij} + x_{ji} = (n-1)\xi_{ij}$:

$$\left.\begin{aligned} x_{ij}(n-2) - x_{ji} &\ge 0 \\ x_{ji}(n-2) - x_{ij} &\ge 0 \end{aligned}\right\} \quad \forall i,j \in V, i < j, i \ne 1 \qquad (4.23)$$

We define inequalities (4.23) to be the *flow inequalities.*

Subtour elimination inequalities

The classical subtour elimination inequalities for the two-commodity flow formulation of the TSP can be expressed in terms of the $\{x_{ij}\}$ variables used in formulation STSP. The original subtour elimination inequalities introduced by Dantzig et al. (1954) are given by:

$$x(\gamma(W)) \leq |W|-1, \ \forall W \subset V, 2 \leq |W| \leq n-2$$

or

$$x(\delta(W)) \geq 2, \ \forall W \subset V, 2 \leq |W| \leq n-2.$$

The variables ξ_{ij} can be expressed as $\xi_{ij} = (x_{ij} + x_{ji})/(n-1)$ in formulation STSP and, therefore, the subtour elimination constraints become:

$$\left.\begin{array}{ll} \sum\limits_{\substack{i,j \in W \\ i<j}} (x_{ij} + x_{ji}) \leq (|W|-1)(n-1), \ \forall W \subset V, 2 \leq |W| \leq n-2 & \text{(a)} \\ \text{or} & \\ \sum\limits_{i \in W} \sum\limits_{j \in V-W} (x_{ij} + x_{ji}) \geq 2(n-1), \ \forall W \subset V, 2 \leq |W| \leq n-2 & \text{(b)} \end{array}\right\} \quad (4.24)$$

The LP-relaxation violates a subtour elimination inequality if and only if the minimum weight cut in G^x has weight less than 2. Since the minimum weight cut in a graph with nonnegative edge weights can be found in polynomial time with the algorithm proposed by Gomory and Hu (1961), the *separation problem* for the subtour elimination inequalities can be solved in polynomial time. The Gomory-Hu algorithm is based on the computation of $n-1$ maximum flow problems on some weighted graphs derived from G^x. The complexity of a maximum flow algorithm is $O(|V||E|log(|V|^2/|E|))$ (see Goldberg and Tarjan (1988)), and so the complexity of the algorithm is $O(|V|^2|E|log(|V|^2/|E|))$. For large instances of the TSP such a complexity is expensive in terms of actual computation time, since the separation problem has to be solved several times in a branch and cut algorithm. For this reason many heuristic procedures have been proposed to find violated subtour elimination inequalities within short time (see, e.g., Crowder and Padberg (1980) and Grötschel and Holland (1991)). The procedure we implemented for the identification of violated subtour elimination inequalities is based on the method proposed by Padberg and Rinaldi (1990). Padberg and Rinaldi describe an exact algorithm that finds the minimum weight cut in a graph with a drastic reduction in the number of maximum flow computations. Even though the algorithm has the same worst case time bound as the Gomory-Hu algorithm, it runs much faster in practice and it allows the execution of an exact separation algorithm at every iteration of a branch and cut procedure. The idea of this algorithm is to exploit some simple sufficient conditions on G^x that guarantee that two nodes belong to the same shore of a minimum cut. If two nodes satisfy one of these conditions then they are *contracted.* The *contraction* of two nodes in G^x produces a new weighted graph where the two nodes are identified into a single node; loops are removed and any two parallel edges are replaced by a single edge with weight equal to the sum of their weights. The resulting graph has one node less and the shores of a minimum cut in it can be turned into the shores of a minimum cut in G^x, by replacing the node that results from the identification with the two original nodes. The contraction of a pair of nodes can be applied recursively until no more reductions apply. At this point the Gomory-Hu algorithm can be applied to the resulting reduced graph.

A different algorithm also based on the contraction of pairs of nodes is proposed by Nagamochi and Ibaraki (1992a, 1992b). The algorithm does not require the computation of a maximum flow and runs in $O(|V||E| + |V|^2 \, log(|V|))$ time. Another algorithm for the minimum cut is proposed in Hao and Orlin (1992). This is a modified version of a maximum flow algorithm and it is able to compute the minimum cut within the same running time required by the computation of a single maximum flow ($O(|V||E| \, log(|V|^2 / |E|))$). Karger (1993) proposed a randomized algorithm for computing a minimum cut with high probability. The algorithm runs in $O(|E||V|^2 \, log^3(|V|))$. An improved version ($O(|V|^2 \, log^3(|V|))$) of the same algorithm has been proposed by Karger and Stein (1993).

4.3 THE VEHICLE ROUTING PROBLEM

In this section the VRP is formulated by using a two-commodity network flow model and an exact branch and cut method for solving the problem to optimality is described.

The VRP is the problem of designing, for a vehicle fleet located at a central depot, a number of feasible routes such that each one starts and ends at the depot and vehicle capacity is not exceeded. The objective is to supply a set of customers requiring deliveries at minimum total distribution cost. In real-world VRPs the distribution cost includes many elements, such as the cost of fuel, tyres, maintenance costs, driver wages, the cost of distance travelled and time spent to visit all customers. In addition to vehicle capacity restrictions, real-world VRP's (see Christofides and Mingozzi (1990)) involve complicated constraints like time-windows to visit customers, customer-vehicle incompatibilities, mixed deliveries or collections on the same route, multiple interacting depots, etc. The practical importance of the problem provides the motivation for the effort involved in the development of heuristic algorithms (see Bodin et al (1983), Christofides et al. (1979b) and Fisher and Jaikumar (1981)). The reader can refer to Christofides (1985), Magnanti (1981) and Osman (1993) for a survey of vehicle routing applications, model extensions and solution methods.

The VRP has been shown to be NP-hard. The fact that few algorithms have been produced to date, which can solve the VRP optimally reflects the difficulty of this problem. During the past fifteen years, exact algorithms have also been developed to solve capacitated routing problems of reasonable size to optimality. For example, Agarwal et al. (1989) and Mingozzi et al. (1994) use a set partitioning and column generation approach and Fisher (1994) uses a lagrangian approach based on the minimum k-tree relaxation. Hadjiconstantinou et al. (1995) proposed an exact algorithm that uses lower bounds obtained from a combination of two relaxations of the original problem which are based on the computation of q-paths and k-shortest paths. Other exact approaches are presented in the surveys of Christofides (1985) and Laporte (1992b). Fisher (1994) reports on the solution of some test problems with up to 100 customers to optimality using a lagrangian relaxation approach embedded in branch and bound. On the other hand, one standard test problem with 76 customers (see Christofides and Eilon (1969)) have never been solved to optimality.

Another approach to optimally solve VRPs is the polyhedral approach which has proved to be efficient for large TSP instances (see for example, Padberg and Rinaldi (1991)). This approach extends to the VRP the successful results of polyhedral combinatorics developed for the TSP by Chvatal (1973) and by Grötschel and Padberg (1979,1985). Initial investigations in the polyhedral aspects of the identical customer VRP and in the similarities between the TSP and VRP polyhedra were performed by Laporte and Nobert (1984), Laporte et al. (1985), Araque (1990), Araque et al. (1990) and by Campos et al. (1991). A more complete description of the VRP polyhedron can be found in Cornuejols and Harche (1993) and in Augerat and Pochet (1995).

Computational results using the polyhedral approach are reported in Araque et al. (1994) (for the identical customer case), Cornuejols and Harche (1993) and Laporte et al. (1985). They solved moderate size problems involving up to 60 customers. Augerat et al. (1995) present a branch and cut algorithm to solve the VRP which is based on the partial polyhedral description of the corresponding polytope. The valid inequalities used in their method can be found in Cornuejols and Harche (1993) and in Augerat and Pochet (1995). Augerat et al. concentrated mainly on the design of separation procedures for different classes of valid inequalities. Several separation heuristics have been implemented and compared for the capacity constraints (generalized subtour elimination inequalities). The computational results show that the capacity constraints play a crucial role in the development of a cutting plane algorithm for the VRP. Augerat et al. also implemented heuristic separation algorithms for other classes of valid inequalities that also led to significant improvements: comb and extended comb inequalities, generalized capacity inequalities and hypotour inequalities. The resulting cutting plane algorithm has been applied to a set of instances taken from the literature. Some branching strategies have been implemented to develop a branch and cut algorithm that has been able to solve large VRP instances (up to 135 customers).

4.3.1 A TWO-COMMODITY FORMULATION OF THE VRP

The VRP can be formulated as follows. A complete undirected graph $G=(V,E)$ is given where $V=\{0,1,\dots,n\}$ is the set of vertices and E is the set of edges.

To every edge $\{i,j\}\in E$ is associated a non-negative cost c_{ij}. $V'=V\setminus\{0\}$ is a set of n vertices, each vertex corresponding to a customer and 0 is the vertex corresponding to the depot. Henceforth, $i\in V'$ will be used interchangeably to refer both to a customer and its vertex location. Each customer i requires a supply of q_i units from depot 0. A set of M identical vehicles of capacity Q is located at the depot and must be used to supply the customers. It is required that every route performed by a vehicle starts and ends at the depot and that the load carried is less than or equal to Q. The cost of a route corresponds to the distance travelled (computed as the sum of the costs of the arcs forming the route). The problem we consider is to design M routes, one for each vehicle, so that all customers are visited and the sum of the route costs is minimized.

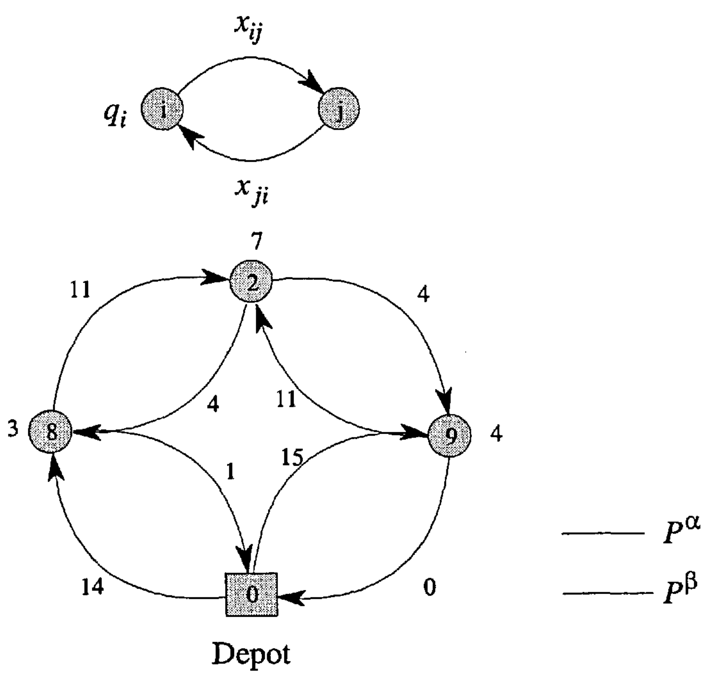


Figure 4.3. Flow circuits for a three customers route

The idea behind this formulation is to use two flow variables, x_{ij} and x_{ji}, to represent an edge $\{i, j\}$ of a feasible VRP solution. If a vehicle travels from i to j then x_{ij} represents the load of the vehicle and x_{ji} represents the empty space on the vehicle (i.e., $x_{ji} = Q - x_{ij}$), whereas, if the vehicle travels from j to i then x_{ij} and x_{ji} represent the empty space on the vehicle and the load, respectively.

The flow variables x_{ij} define two flow circuits for any route of a feasible solution: one circuit is defined by the flow variables representing the vehicle load, while the second one is defined by the flow variables representing the empty space on the vehicle.

In Figure 4.3 is shown a three customer route for a vehicle of capacity Q=15 and the two circuits P^{α} and P^{β} represented by the flow variables x_{ij} defining the route. Circuit P^{α} is formed by the variables representing the vehicle load: the flow $x_{08} = 14$ indicates the total demand of the three customers, $x_{82} = 11$ represents the load of the vehicle in traveling from 8 to 2 after having unloaded 3 load units at customer 8, $x_{29} = 4$ represents the load of the vehicle in traveling from 2 to 9 after having unloaded

7 load units at customer 2, finally $x_{90} = 0$ represents the load of the vehicle in returning to the depot after having unloaded the remaining 4 load units at customer 9. Note that for every edge $\{i, j\}$ of the route we have $x_{ij} + x_{ji} = Q$.

Let ξ_{ij} be a 0-1 binary variable equal to 1 if edge $\{i, j\} \in E$ is in solution, 0 otherwise. Let x_{ij} be the flow value of arc (i, j), $i, j \in V, i \neq j$.

The mathematical formulation for the VRP is as follows:

$$\text{(VRP)} \quad z(\text{VRP}) = Min \sum_{\{i,j\} \in E} c_{ij}\xi_{ij} \tag{4.25}$$

$$\textit{subject to} \quad \sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = -2q_i, \qquad i \neq 0 \quad \Big\} \; \forall i \in V \tag{4.26}$$

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 2\sum_{j \in V'} q_j, \qquad i = 0 \quad \Big\} \; \forall i \in V \tag{4.27}$$

$$x_{ij} + x_{ji} = Q\xi_{ij}, \qquad \forall \{i, j\} \in E \tag{4.28}$$

$$x(\delta(i)) = 2, \qquad i \neq 0 \quad \Big\} \; \forall i \in V \tag{4.29}$$

$$x(\delta(i)) = 2M, \qquad i = 0 \quad \Big\} \; \forall i \in V \tag{4.30}$$

$$x_{ij} \geq 0, \qquad \forall i, j \in V \tag{4.31}$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall \{i, j\} \in E \tag{4.32}$$

Constraints (4.26), (4.27) and (4.31) define a feasible flow for variables $\{x_{ij}\}$. Constraints (4.28) together with (4.29) and (4.30) force the degree of each customer to be 2 and the degree of the depot to be *2M*, respectively. Constraints (4.32) are the integrality constraints. The supply-demand pattern involved ensures that there are paths from vertex 0 to any vertex in V', and back from these vertices to vertex 0. Since, from (4.28) and (4.31), $x_{ij} + x_{ji} = Q\xi_{ij}$, $\forall \{i, j\} \in E$, the capacity of the vehicle will never be exceeded in the route allocated to it.

Note that the STSP formulation described in Section 4.2.1 can be derived from the VRP formulation by setting $q_i = 1$, $\forall i \in V'$, $M = 1$ and $Q = n - 1$.

4.3.2 A LOWER BOUND FROM THE LP-RELAXATION

A valid lower bound for the VRP can be obtained by considering the LP-relaxation of formulation VRP. Using equations (4.28), we can eliminate $\{\xi_{ij}\}$ variables and the LP-relaxation can be written as follows:

$$\text{(LRVRP)} \qquad z(\text{LRVRP}) = Min \ \frac{1}{Q} \sum_{\{i,j\}\in E} c_{ij}\left(x_{ij} + x_{ji}\right) \tag{4.33}$$

$$subject\ to \ \ (4.26), (4.27), (4.29), (4.30), (4.31) \text{ and}$$

$$x_{ij} + x_{ji} \leq Q, \ \forall\{i,j\} \in E \tag{4.34}$$

As described for the TSP case, some valid inequalities can be added to the LP relaxation of formulation VRP in order to improve the lower bound. These inequalities are satisfied by any feasible integer solution of the VRP formulation, but they are not necessarily satisfied by a LRVRP solution. Hence, the lower bound can be strengthened by identifying violated inequalities and adding them to the LP relaxation. For the LRVRP formulation we consider the *trivial inequalities* used for the STSP formulation and we modify the *flow inequalities* and the *subtour inequalities* as described below.

Flow inequalities

Consider $x_{ij} + x_{ji} = Q\xi_{ij}, \ \forall\{i,j\} \in E, i \neq 0, j \neq 0$. In any feasible integer solution, if $\xi_{ij} = 1$, then $x_{ij} \geq q_j$ and $x_{ji} \geq q_i$. Hence, any feasible LRVRP solution where $\xi_{ij} > 0$ and either $x_{ij} = 0$ or $x_{ji} = 0$, for some edge $\{i,j\} \in E$, cannot be a feasible VRP solution. Therefore, we can force any LRVRP solution to satisfy the following constraints.

$$\left.\begin{aligned} x_{ij} &\geq q_j \xi_{ij} \\ x_{ji} &\geq q_i \xi_{ij} \end{aligned}\right\} \quad \forall\{i,j\} \in E, i \neq 0, j \neq 0$$

or, using equation $x_{ij} + x_{ji} = Q\xi_{ij}$:

$$\left.\begin{aligned} x_{ij}\left(Q - q_j\right) - q_j x_{ji} &\geq 0 \\ x_{ji}\left(Q - q_i\right) - q_i x_{ij} &\geq 0 \end{aligned}\right\} \quad \forall i,j \in V, i < j, i \neq 0$$

Capacity constraints

Denote by $r(S)$, $S \subset V'$, a lower bound on the minimum number of vehicles needed to satisfy the customers demand in a set S in any feasible solution, that is:

$$r(S) = \left\lceil \left(\sum_{i \in S} q_i \right) \Big/ Q \right\rceil$$

where $\lceil x \rceil$ is the smallest integer greater that or equal to x.

We obtain the following valid inequality:

$$x(\delta(S)) \geq 2r(S)$$

also called *generalized subtour elimination constraint.*

Harche and Rinaldi (1991) showed that the separation problem for the capacity constraints is NP-Complete and designed several heuristics. Augerat et al. (1995) used different heuristics to identify violated capacity constraints. They compare the heuristics of Harche and Rinaldi (1991) with a *greedy shrinking algorithm* and a *tabu search* based heuristic. The computational results for the lower bounds obtained using the different identification heuristics on a set of instances taken from the literature show that the best performance is achieved by combining the greedy shrinking algorithm and the tabu search algorithm.

We identified violated capacity constraints by means of the greedy shrinking algorithm proposed by Augerat et al. Given an initial subset of customers S, at each iteration, a customer k is added to S in such a way that $x((S:k))$ is maximized. Customer subsets are randomly generated and the number of initial subsets is fixed to ten times the number of customers.

4.3.3 A BRANCH AND CUT METHOD FOR THE VRP

We implemented a branch and cut algorithm based on the one proposed by Augerat et al. (1995). The cutting plane algorithm, for the identification of the valid inequalities described in the previous section, is applied to every subproblem until no violated inequality is found or the solution does not increase during an apriori fixed number of iterations. A subproblem is fathomed if an integer feasible solution is found or the lower bound obtained is not less than the current upper bound. If a subproblem is not fathomed, it is divided in two subproblems by branching on a given inequality as explained below. The subproblem to be explored is selected as the one having minimum lower bound.

The branching strategy used is the following. Let S be a subset of customers for which $x(\delta(S))-2r(S)=p(S)$, $0<p(S)<2$, then we can create two subproblems: one adding constraint $x(\delta(S))=2r(S)$ and the other adding constraint $x(\delta(S))\geq 2r(S)+2$. The selection of subset S is carried out in two steps: firstly, a candidate list of subsets is build heuristically and, secondly, one of them is selected from this list according to some strategy. The list of candidate subsets is build by the same heuristic algorithm used for the identification of the capacity constraints. An initial subset of customer subsets is randomly generated and then the greedy shrinking algorithm is used to expand this subset in order to generate a new list where each subset S satisfies $0<p(S)<2$. In order to have a balanced tree search, we force each candidate subset to satisfy $0.75\leq p(S)\leq 1$. We select four subsets according to the following criteria that are exactly the same used by Augerat et al. (1995).

- Select set S_1 with maximum demand.
- Select set S_2 which is farthest from the depot.
- Select set S_3 such that $x(\delta(S_3))$ is as close as possible to 3.
- Select set S_4 such that $x(\delta(S_4))$ is as close as possible to 2.75.

Then, one out of four subsets (S_1, S_2, S_3, and S_4) is chosen for branching using the method described by Applegate et al. (1994) for the TSP. For each subset $S\in\{S_1,S_2,S_3,S_4\}$ we solve the two corresponding subproblems and compute the minimum of the increases in the lower bounds with respect to the lower bound of the current node. Then, we choose the subset which leads to the maximum of these minimum increases.

4.3.4 Computational results

The branch and cut algorithm described in the previous section has been coded in Fortran 77 and experimentally evaluated on a set of 16 difficult VRP instances taken from the literature. All computations were performed on a Silicon Graphics Indy (MIPS

R 4400/200 MHz processor), using CPLEX 4.0 (1996) as the LP-solver. For each problem the vertices of graph G are located in the plane and the cost c_{ij} is computed as integer value equal to $\left\lfloor e_{ij} + \frac{1}{2} \right\rfloor$, where e_{ij} is the Euclidean distance between points i and j. This is the same cost function used by Mingozzi et al. (1994), Augerat et al. (1995) and it is the one proposed in the TSPLIB (Reinelt (1991)).

The tables show the following columns:

Prob:	problem name identifier;
n:	number of customers;
Q:	vehicle capacity;
M:	number of vehicles;
RQ%:	percentage of tightness of the capacity constraints computed as $100 \sum_{i \in V} q_i \Big/ MQ$.
Reference:	reference from which each instance has been taken and where the complete data can be found;
$z(UB)$:	cost of the best known heuristic VRP solution found in the literature;
LB0:	lower bound $z(LRVRP)$;
$\%E_{LB0}$:	percentage error of lower bound *LB0*;
LB1:	final lower bound at the root node of the branch and cut algorithm obtained by the cutting plane algorithm for the identification of the valid inequalities;
$\%E_{LB1}$:	percentage error of lower bound *LB1*;
t_{LB1}:	total computing time of lower bound *LB1*;
$\%E_A^c$:	percentage error of the lower bound produced by Augerat et al. (1995) by considering only the identification of capacity constraints;
$t_{E_A^c}$:	total computing time of the lower bound produced by Augerat et al. by considering only the identification of capacity constraints (seconds of a Sun Sparc 10 machine);
$\%E_A$:	final lower bound produced by Augerat et al.;

t_{E_A} :	total computing time of the final lower bound produced by Augerat et al. (seconds of a Sun Sparc 10 machine);
$\%E_M$:	percentage error of the lower bound produced by Mingozzi et al. (1994);
t_{E_M} :	total computing time of the lower bound produced by Mingozzi et al. (seconds of a Silicon Graphics Indy, MIPS R4400/200 MHz processor);
$\%E_F$:	percentage error of the lower bound produced by Fisher (1994);
t_{E_F} :	total computing time of the lower bound produced by Fisher (seconds of an Apollo Domain 3000 machine);
z^* :	cost of the optimal VRP solution;
nodes:	number of nodes generated by the branch and cut algorithm;
t_{BC} :	total computing time of the branch and cut algorithm; we impose a time limit of 3600 CPU seconds.

The percentage errors are computed as the ratio of the lower bound divided by $z(UB)$ and multiplied by 100.

Table 4.1 shows the data of the test problems. Table 4.2 shows the comparison of different lower bounds, while Table 4.3 shows the number of cuts generated for computing the lower bound *LB1*. Table 4.4 shows the problems solved to optimality within the imposed time limit of 3600 CPU seconds and reports the number of nodes of the branch and cut algorithm and the total computing time required for finding the optimal solution. In the last 4 lines of Table 4.2 we report the average percentage errors of lower bounds *LB0* and *LB1* on all the problem instances and on the problem instances solved by Augerat et al. (1995), by Mingozzi et al. (1994) and by Fisher (1994), respectively. Note that the Apollo Domain 3000 is about 15 times slower than the Silicon Graphics Indy we used. The costs used by Fisher are real ones and therefore the lower bounds should not be compared directly because the optimal solution costs may not be the same; nevertheless, the comparison between the respective percentage errors may override this difficulty.

The results show that the average percentage error computed on all the problem instances of lower bound *LB1* is equal to 98%. By observing Table 4.2 we note that the addition of the valid inequalities substantially improves the value of the lower bound *LB0*. A comparison between lower bound *LB1* and the one computed by Augerat et al. shows that the average percentage errors $\%E_{LB1}$ and $\%E_A^c$ are the same. Moreover, the results show that the improvements on the quality of the lower bounds obtained by Augerat et al. after considering other classes of valid inequalities (comb and extended comb inequalities, generalized capacity inequalities and hypotour inequalities) is on average equal to 0.3. Three instances were solved using the cutting plane algorithm at the root node of the branch and cut tree. Table 4.4 shows that the branch and cut algorithm has been able to solve problems up to 100 customers.

Table 4.1. Test Problems

Prob	n	Q	M	$RQ\%$	Reference
1	15	55	5	93.8	CMT81
2	15	90	3	95.6	CMT81
3	20	58	6	94.5	CMT81
4	20	85	4	96.7	CMT81
5	21	60	4	93.7	CMT81
6	21	40	6	93.7	CMT81
7	25	48	8	95.6	CMT81
8	50	160	5	97.1	CE69
9	75	140	10	97.4	CE69
10	75	220	7	88.6	CE69
11	75	180	8	94.7	CE69
12	100	200	8	91.1	CE69
13	100	200	10	90.5	CMT79
14	44	2010	4	89.8	FIS94
15	71	30000	4	95.7	FIS94
16	134	2210	7	94.5	FIS94

CMT81:	Christofides et al. (1981a).
CE69:	Christofides and Eilon (1969).
CMT79:	Christofides et al. (1979b);
FIS94:	Fisher (1994).

Table 4.2: Comparison of lower bounds

		Two-commodity					Augerat et al.				Mingozzi et al.		Fisher	
Prob	$z(UB)$	*LB0*	*%LB0*	*LB1*	*%LB1*	t_{LB1}	$\%E_A^c$	$t_{E_A^c}$	$\%E_A$	t_{E_A}	$\%E_M$	t_M	$\%E_F$	t_F
1	333	290.5	87.2	321.4	96.5	2	-	-	-	-	97.9	2	-	-
2	277	243.3	87.8	265.5	95.8	1	-	-	-	-	97.4	2	-	-
3	430	371.8	86.5	426.8	99.2	3	-	-	-	-	100.0	2	-	-
4	358	309.7	86.5	346.1	96.7	2	-	-	-	-	100.0	3	-	-
5	375	311.0	82.9	375.0	100.0	2	100.0	1	100.0	5	100.0	2	-	-
6	495	395.6	79.9	484.0	97.8	4	-	-	-	-	97.4	3	-	-
7	606	511.4	84.4	606.0	100.0	5	-	-	-	-	100.0	2	-	-
8	521	500.6	96.1	514.5	98.8	51	98.8	17	99.3	129	99.3	84	96.7	5745
9	832	773.0	92.9	791.3	95.1	117	94.9	347	95.4	1919	97.8	206	90.5	11038
10	683	650.5	95.2	660.7	96.7	182	96.8	147	97.3	1052	97.4	320	-	-
11	735	693.5	94.4	711.6	96.8	109	96.8	258	97.1	1282	97.8	251	-	-
12	817	775.6	94.9	795.0	97.3	232	97.5	294	97.9	1708	97.4	404	-	-
13	820	770.1	93.9	819.8	100.0	120	99.9	180	100.0	167	99.5	382	99.8	15578
14	724	614.3	84.8	724.0	100.0	20	100.0	8	100.0	12	-	-	99.6	2984
15	238	205.1	86.2	232.5	97.7	43	97.7	11	98.7	59	-	-	98.3	6301
16	1165	1005.1	86.3	1155.8	99.2	2271	99.4	1513	99.5	2024	-	-	97.4	15230
Averages:														
All instances			88.7		98.0									
Augerat et al.			90.8		98.2		98.2		98.5					
Mingozzi et al.			89.4		97.8						98.6			
Fisher			90.0		98.5								97.0	

Table 4.3. Details of the number of cuts of lower bound *LB1*

Prob	Trivial Inequalities	Flow Inequalities	Capacity Constraints	Total
1	17	48	77	142
2	13	19	35	67
3	19	45	151	215
4	17	44	81	142
5	16	33	137	186
6	24	45	258	327
7	24	28	410	462
8	33	100	399	532
9	56	206	710	972
10	47	138	772	957
11	48	165	559	772
12	62	196	774	1032
13	72	256	1061	1389
14	27	103	457	587
15	42	126	440	608
16	53	564	2711	3328

Table 4.4. Problems solved to optimality

Prob	z^*	*nodes*	t_{BC}
1	333	14	10
2	277	58	26
3	430	4	7
4	358	68	67
5	375	0	2
6	495	90	144
7	609	0	5
8	521	66	660
13	820	4	149
14	724	0	20

4.4 THE TSP WITH DELIVERY AND COLLECTION CONSTRAINTS

In this section we extend the two-commodity STSP formulation described in Section 4.2.1 to deal with the TSPDC and the TSPB.

The TSPDC is defined as follows. We are given a complete undirected graph $G=(V,E)$ where $V=\{0,n+1\}\cup D\cup C$ is the set of vertices and E is the set of edges. Vertices 0 and n+1 represent the depot location, the set D corresponds to the delivery customers (requiring delivery from depot 0) and the set C corresponds to the collection customers (sending goods to depot n+1). To every edge $\{i,j\}\in E$ is associated a non-negative cost c_{ij} and a quantity q_i is associated with each customer $i\in D\cup C$ (we assume that the q_i values are non-negative integers and that $q_0=q_{n+1}=0$). A vehicle of capacity Q is located at depot 0 and must be used to supply the delivery customers and to collect goods from the collection customers. The TSPDC consists of determining a path starting at depot 0 and ending at depot n+1, serving each customer exactly once, and having minimum length, defined as the sum of the costs of the arcs forming the path. The total load of the vehicle along the tour must never exceed the vehicle capacity, Q. To ensure the feasibility of the problem we assume that $Q\geq Max\left[\sum_{i\in D}q_i,\sum_{i\in C}q_i\right]$.

The TSPDC is NP-Hard in the strong sense since it generalizes the TSP. The problem has many practical applications in the design and management of distribution systems, like the transportation of under-privileged children from home to vacation locations described in Mosheiov (1994). Mosheiov (1994) proposed a mathematical model for TSPDC and heuristic algorithms based on the extension of methods for the standard TSP. For one of the proposed algorithms a worst-case performance ratio equal to 1+α was also proved, where α is the worst-case performance ratio of the TSP heuristic used. Anily and Mosheiov (1994) described a new heuristic with worst-case performance ratio equal to 2 based on the solution of Shortest Spanning Trees. Gendreau et al. (1997) proposed two heuristic algorithms for the TSPDC. The first is based on the optimal solution of a special case of TSPDC arising when graph G is a cycle. In particular, they derive a linear time algorithm for the optimal solution of this special case and use it as a base for developing a heuristic for the general TSPDC. The worst-case performance of the proposed algorithm is studied, and a tight ratio of 3 is derived. The computational results show that this heuristic generally produces better solutions with respect to those previously proposed in the literature. A further improvement is obtained by means of a second heuristic based on the tabu search approach (see, Glover (1989,1990) and Glover and Laguna (1997)) that uses a neighborhood based on exchanges of two arcs. Computational results show that the tabu search of Gendreau et al. (1997) outperforms the previous heuristics.

The special case of TSPDC, known as TSPB, where in any feasible solution all delivery customers must precede the collection customers has been studied by Gendreau at al. (1996) who presented the extension to TSPB of the GENIUS heuristic for the TSP. The generalization of TSPDC related to the VRP, where several vehicles are available, has been considered by Halse (1992) who proposed a mathematical formulation and a heuristic algorithm based on a Lagrangian relaxation of the problem. To our knowledge, no exact methods have been proposed for the optimal solution of the TSPDC and the TSPB.

4.4.1 A TWO-COMMODITY FORMULATION OF THE TSPDC

Following the idea used in formulating the VRP as a two-commodity network flow model, each feasible TSPDC path from depot 0 to depot n+1, can be represented by two flow circuits: the first one representing the vehicle load and the second one representing the vehicle empty space. As an example, consider the tour of Figure 4.4. Let the vehicle capacity Q be equal to 10. The flow x_{ij} on arc (i, j) of the flow circuit (0,1,2,3,4,5) represents the *vehicle load*, while the flow x_{ji} on arc (j,i) of the flow circuit (5,4,3,2,1,0) represents the *vehicle empty space*. Note that the flow on arc x_{01} of the example is the total demand of the delivery customers, that is, $Q_D = \sum_{i \in D} q_i = 9$, while, the flow on arc x_{45} is the total demand of the collection customers, that is, $Q_C = \sum_{i \in C} q_i = 10$. Moreover, for each edge $\{i, j\} \in \{\{0,1\},\{1,2\},\{2,3\},\{3,4\},\{4,5\}\}$ we have $x_{ij} + x_{ji} = Q$.

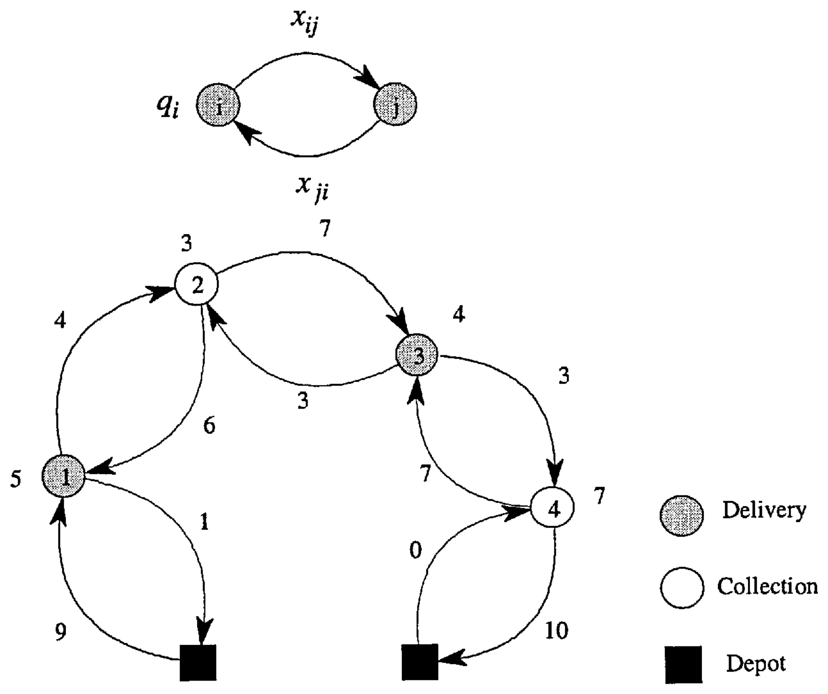


Figure 4.4. Flows in the TSPDC

Let $V' = D \cup C$ and let ξ_{ij} be a 0-1 binary variable equal to 1 if edge $\{i, j\} \in E$ is in solution, 0 otherwise. Let x_{ij} be the flow value of arc (i, j), $i, j \in V, i \neq j$.

The mathematical formulation for the TSPDC is as follows:

$$\text{(TSPDC)} \quad z(\text{TSPDC}) = Min \sum_{\{i,j\} \in E} c_{ij}\xi_{ij} \tag{4.35}$$

$$\textit{subject to} \quad \sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = -2q_i, \qquad \forall i \in D \tag{4.36}$$

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 2q_i, \qquad \forall i \in C \tag{4.37}$$

$$\sum_{j \in V'} x_{0j} = Q_D \tag{4.38}$$

$$\sum_{j \in V'} x_{j0} = Q - Q_D \tag{4.39}$$

$$\sum_{j \in V'} x_{n+1j} = Q - Q_C \tag{4.40}$$

$$\sum_{j \in V'} x_{jn+1} = Q_C \tag{4.41}$$

$$x_{ij} + x_{ji} = Q\xi_{ij}, \qquad \forall \{i,j\} \in E \tag{4.42}$$

$$x(\delta(i)) = 2, \qquad \forall i \in V' \tag{4.43}$$

$$x(\delta(i)) = 1, \qquad i = 0, n+1 \tag{4.44}$$

$$x(\delta(S)) \geq 2, \qquad \forall S \subset V' \ s.t. \ \sum_{i \in S \cap D} q_i = \sum_{i \in S \cap C} q_i \tag{4.45}$$

$$x_{ij} \geq 0, \qquad \forall i, j \in V, i \neq j \tag{4.46}$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall \{i,j\} \in E \tag{4.47}$$

Constraints (4.36) to (4.41) and (4.46) define a feasible flow for variables $\{x_{ij}\}$. Constraints (4.42) together with (4.43) and (4.44) force the degree of each customer to be 2 and the degree of both depots 0 and n+1 to be 1. Constraints (4.45) are the subtour elimination constraints. Constraints (4.47) are the integrality constraints.

Note that in the TSPDC formulation, due to the supply-demand patterns of the delivery and collection customers, it is necessary to introduce the subtour elimination constraints as shown in the example of Figure 4.5. In fact, the solutions shown in Figure 4.5(a) and 4.5(b) are feasible for the TSPDC formulation without constraints (4.45).

4.4.2 A Two-commodity formulation of the TSPB

The mathematical formulation of the TSPB can be easily derived from the one described for the TSPDC in the previous section. We can assume, without loss of generality, that $q_i = 1$, $\forall i \in D \cup C$ (since all deliveries must be made by the single vehicle before any collections), $Q = Max[|D|, |C|]$ and that $|D| \geq |C|$. Figure 4.6 shows an example of a TSPB feasible tour in terms of the two-commodity flows.

Let $V' = D \cup C$ and let ξ_{ij} be a 0-1 binary variable equal to 1 if edge $\{i,j\} \in E$ is in solution, 0 otherwise. Let x_{ij} be the flow value of arc (i,j), $i,j \in V, i \neq j$.

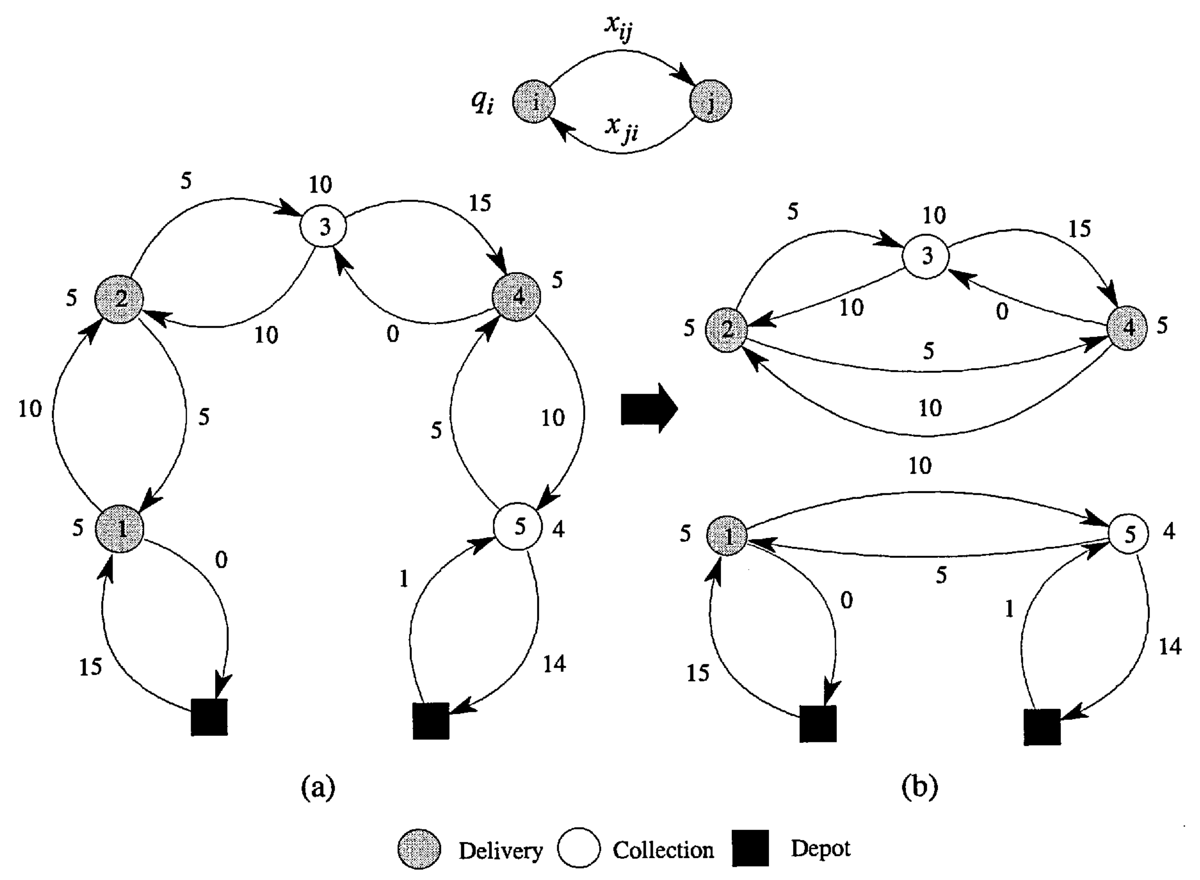


Figure 4.5: Subtours in the TSPDC

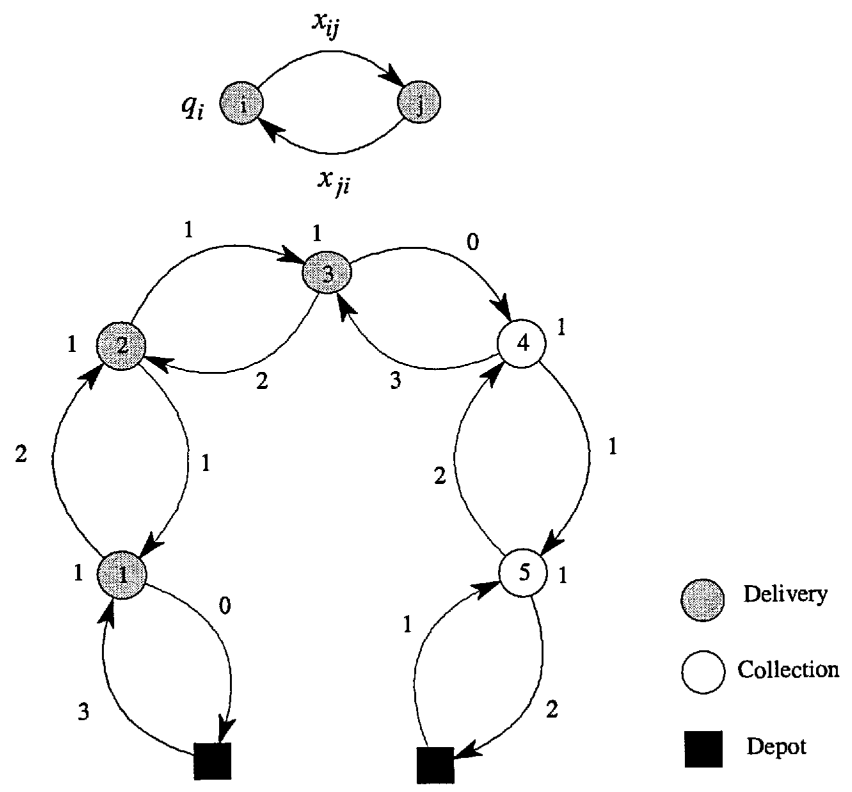


Figure 4.6: Flows in the TSPB

The mathematical formulation for the TSPB is as follows:

$$\text{(TSPB)} \qquad z(\text{TSPB}) = Min \sum_{\{i,j\}\in E} c_{ij}\xi_{ij} \tag{4.48}$$

$$subject\ to \quad \sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = -2, \qquad \forall i \in D \tag{4.49}$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 2, \qquad \forall i \in C \tag{4.50}$$

$$\sum_{j\in V'} x_{0j} = |D| \tag{4.51}$$

$$\sum_{j\in V'} x_{j0} = 0 \tag{4.52}$$

$$\sum_{j\in V'} x_{n+1j} = |D| - |C| \tag{4.53}$$

$$\sum_{j\in V'} x_{jn+1} = |C| \tag{4.54}$$

$$x_{ij} + x_{ji} = |D|\xi_{ij}, \qquad \forall \{i,j\} \in E \tag{4.55}$$

$$x(\delta(i)) = 2, \qquad \forall i \in V' \tag{4.56}$$

$$x(\delta(i)) = 1, \qquad i = 0, n+1 \tag{4.57}$$

$$\sum_{i\in D}\sum_{j\in C} x_{ij} = 0 \tag{4.58}$$

$$x_{ij} \geq 0, \qquad \forall i,j \in V, i \neq j \tag{4.59}$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall \{i,j\} \in E \tag{4.60}$$

Constraints (4.49) to (4.54) and (4.59) define a feasible flow for variables $\{x_{ij}\}$. Constraints (4.55) together with (4.56) and (4.57) force the degree of each customer to be 2 and the degree of both depots 0 and $n+1$ to be 1. Constraints (4.58) avoid subtours and force deliveries to precede collections. Constraints (4.60) are the integrality constraints.

Note that the TSPB can also be solved as an Asymmetric TSP. This can be done by setting the asymmetric cost matrix $[c_{ij}]$ as follows:

$$c_{ij} = \infty \qquad \forall i \in C,\ \forall j \in D$$

$$c_{i0} = c_{0i} = \infty,\ c_{n+1i} = \infty \qquad \forall i \in C$$

$$c_{i\,n+1} = c_{n+1\,i} = \infty,\ c_{i0} = \infty \quad \forall i \in D$$

4.4.3 Lower bounds from the LP-relaxation

Valid lower bounds for the TSPDC and for the TSPB can be obtained by means of the LP-relaxation of the corresponding formulations. Similarly for the STSP case, the value of the lower bound can be strengthened by the trivial inequalities, the flow inequalities and the subtour elimination inequalities. The flow inequalities, described for the STSP in Section 4.2.2, can be generalized for the TSPDC and for the TSPB as described below.

Flow inequalities

Consider $x_{ij} + x_{ji} = Q\xi_{ij}$, $\forall \{i, j\} \in E, i \neq 0, j \neq 0$. In any feasible integer solution, if $\xi_{ij} = 1$, then we must consider the following three cases:

a) $i, j \in D$.

Then $x_{ij} \geq q_j \xi_{ij}$ and $x_{ji} \geq q_i \xi_{ij}$, or, using equation $x_{ij} + x_{ji} = Q\xi_{ij}$,

$$x_{ij}(Q - q_j) - q_j x_{ji} \geq 0 \text{ and } x_{ji}(Q - q_i) - q_i x_{ij} \geq 0. \tag{4.61}$$

b) $i \in D$ and $j \in C$.

Then $x_{ji} \geq q_i \xi_{ij}$ or, using equation $x_{ij} + x_{ji} = Q\xi_{ij}$,

$$x_{ji}(Q - q_i) - q_i x_{ij} \geq 0. \tag{4.62}$$

c) $i, j \in C$.

Then $x_{ij} \geq q_i \xi_{ij}$ and $x_{ji} \geq q_j \xi_{ij}$, or, using equation $x_{ij} + x_{ji} = Q\xi_{ij}$,

$$x_{ij}(Q - q_i) - q_i x_{ji} \geq 0 \text{ and } x_{ji}(Q - q_j) - q_j x_{ij} \geq 0. \tag{4.63}$$

Therefore, inequalities (4.61), (4.62) and (4.63) can be added to the LP-relaxation of formulations TSPDC and TSPB to eliminate infeasible fractional solutions.

4.4.4 A BRANCH AND CUT METHOD FOR THE TSPDC AND THE TSPB

The branch and cut method described for the VRP in Section 4.3.3 can be easily adapted for solving the TSPDC and the TSPB. The tree search is a binary tree search in which at each node the procedure for the identification of the trivial inequalities, flow inequalities and subtour elimination inequalities is applied until no violated inequality is found or the solution does not increase for a certain number of iterations. A forward branching at a certain node involves the selection of a subset of customers S for which $0 < x(\delta(S)) - 2 < 2$, and the generation of two new subproblems: one of them adding constraints $x(\delta(S)) = 2$ and the other adding constraint $x(\delta(S)) \geq 4$. The branching subset selection is carried out similarly for the VRP as described in Section 4.3.3. Firstly, a candidate list of subsets is build heuristically and, secondly, one of them is selected from this list according to some strategy. The list of candidate subsets is build by the same heuristic algorithm used for the identification of the capacity constraints. An initial subset of customer subsets is randomly generated and then the greedy shrinking algorithm is used to expand this subset in order to generate a new list where each subset S satisfies $0 < p(S) < 2$. In order to have a balanced tree search, we impose that each candidate subset satisfy $0.75 \leq p(S) \leq 1$. We select four subsets according to the following criteria.

- Select set S_1 with maximum cardinality.
- Select set S_2 which is farthest from the depot.
- Select set S_3 such that $x(\delta(S_3))$ is as close as possible to 3.
- Select set S_4 such that $x(\delta(S_4))$ is as close as possible to 2.75.

Then, one out of four subsets (S_1, S_2, S_3, and S_4) is chosen for branching using the method described by Applegate et al. (1994) for the TSP.

4.4.5 A NUMERICAL EXAMPLE

In this section we present a numerical example to illustrate the lower bound computation for an instance of the TSPDC. We have used one of the 20 test problems generated by Mosheiov (1994) for simulating the problem arising in the transportation

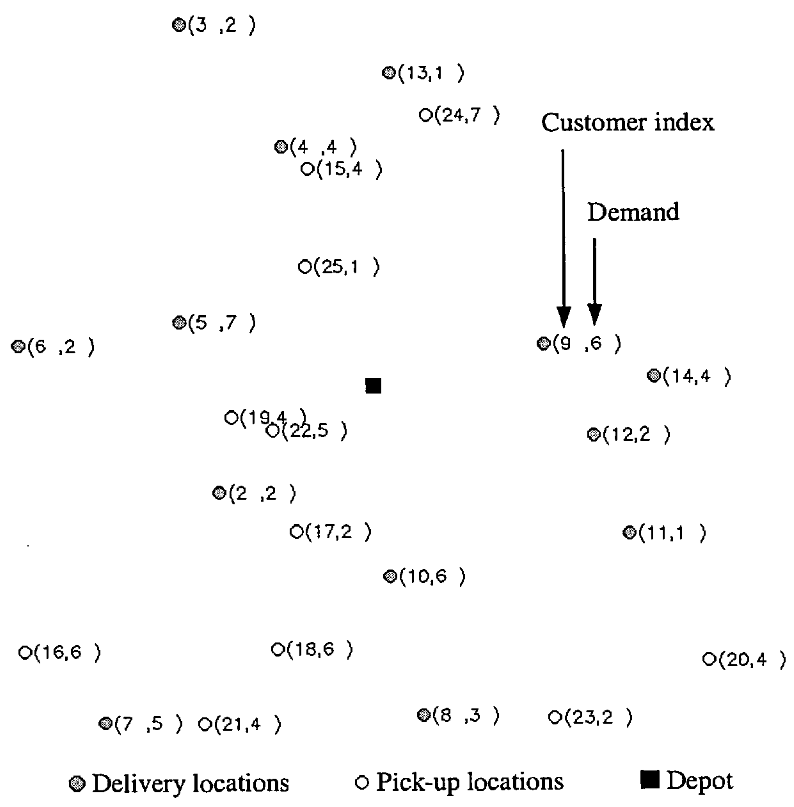


Figure 4.7. Example: customer locations

of under-privileged children from home to vacation locations. All locations were randomly generated from a uniform distribution in the square [(-500,500)×(-500,500)] and the central station was located at the origin. The number of children at pick-up and delivery locations were generated from a uniform distribution in [1,8] and the bus capacity was set equal to 45 seats (i.e. Q=45). The example we consider has 24 customer locations, 13 of which are delivery locations (i.e. $|D| = 13$) and 11 are pick-up locations (i.e. $|C| = 11$). The distances between customer locations were computed using the Euclidean distance as proposed in the TSPLIB (Reinelt (1991)). The total delivery demand is equal to 45 and is equal to the total pick-up demand (i.e. $Q_D = Q_C = Q = 45$). Table 4.5 presents the data of the problem and Figure 4.7 shows the customer locations. The cost of the assignment solution is 3873 and the cost of the optimal TSP solution, that is a valid lower bound on the cost of the optimal solution of the problem, is 4430. The optimal TSP solution is shown in Figure 4.8.

Table 4.5. Example: problem data

Delivery locations				Pick-up locations			
	X-coordinate	Y-coordinate	demand		X-coordinate	Y-coordinate	demand
1	-211.651	-147.830	2	14	-90.618	299.859	4
2	-265.178	499.830	2	15	-480.287	-366.363	6
3	-126.376	330.654	4	16	-106.647	-202.037	2
4	-267.075	88.573	7	17	-134.373	-364.665	6
5	-486.620	56.890	2	18	-196.116	-43.097	4
6	-368.762	-466.753	5	19	455.724	-383.012	4
7	66.033	-456.440	3	20	-232.731	-467.685	4
8	234.876	57.177	6	21	-138.385	-61.276	5
9	22.466	-263.939	6	22	247.014	-461.857	2
10	349.490	-205.735	1	23	74.257	373.238	7
11	302.043	-70.028	2	24	-95.265	164.968	1
12	23.194	432.094	1				
13	384.468	11.208	4				

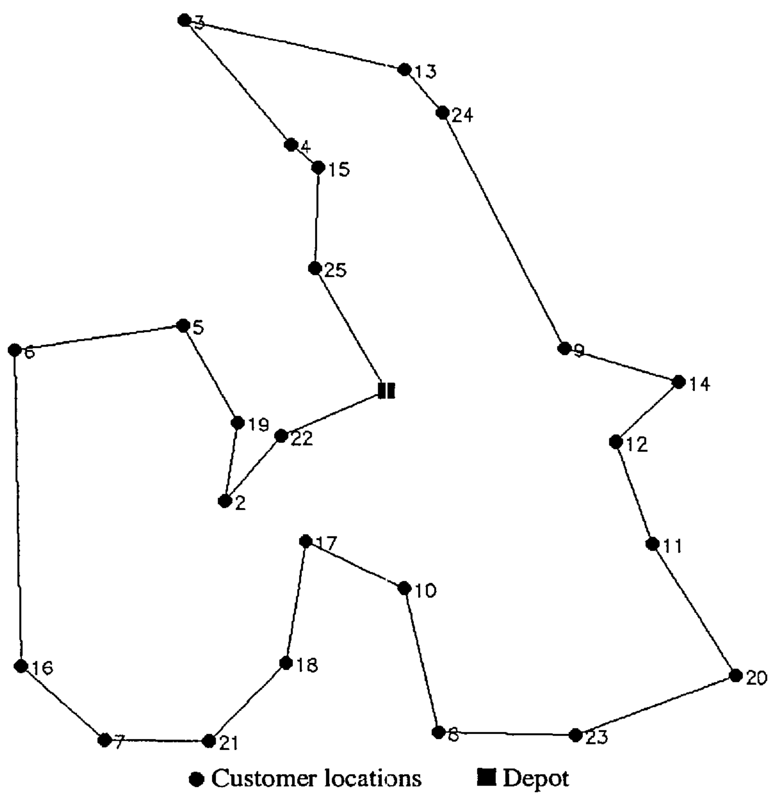
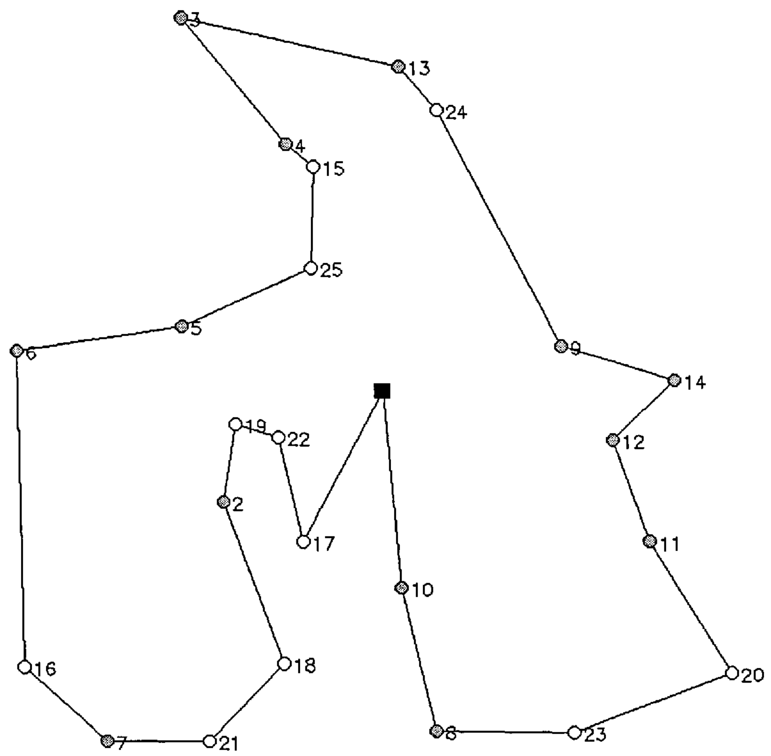
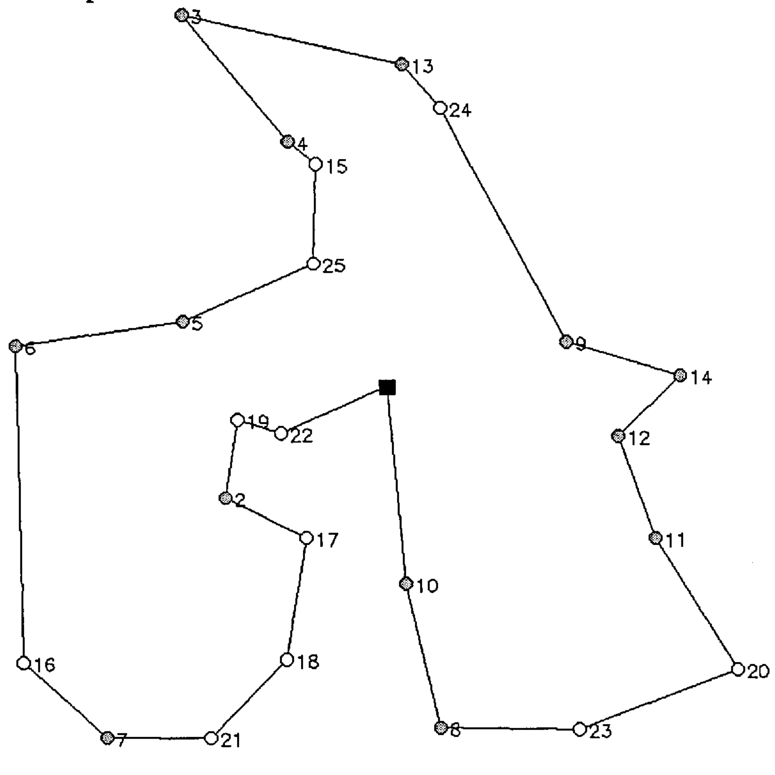


Figure 4.8. Example: optimal TSP solution of cost 4430



● Delivery locations ○ Pick-up locations ■ Depot

Figure 4.9. Example: heuristic solution of cost 4631 found by Mosheiov (1994)



● Delivery locations ○ Pick-up locations ■ Depot

Figure 4.10. Example: optimal TSPDC solution of cost 4464

Table 4.6. Example: flows of the optimal TSPDC solution

i	j	x_{ij}	x_{ji}	i	j	x_{ij}	x_{ji}
1	10	45	0	8	23	36	9
1	22	0	45	9	14	10	35
2	17	7	38	9	24	29	16
2	19	36	9	11	12	41	4
3	4	33	12	11	20	3	42
3	13	10	35	12	14	39	6
4	15	29	16	13	24	9	36
5	6	27	18	15	25	33	12
5	25	11	34	17	18	9	36
6	16	25	20	18	21	15	30
7	16	14	31	19	22	40	5
7	21	26	19	20	23	7	38
8	10	6	39				

The cost of the heuristic solution found by the algorithm proposed by Mosheiov is 4631 (see Figure 4.9).

The value of the LP-relaxation of formulation TSPDC was 3975.6 and the value of the final lower bound obtained after the application of the separation procedures for the identification of valid inequalities was 4464.0. The solution found was integer, therefore, 4464 was also the cost of the optimal solution of the problem. The separation procedures found 13 trivial inequalities, 10 flow inequalities and 7 subtour elimination inequalities. The total computing time was 1.46 seconds on a Silicon Graphics Indy (MIPS R4400/200 MHz processor). We have used CPLEX 4.0 (1996) as the LP-solver. Figure 4.10 and Table 4.6 show the optimal solution found and the values of the $\{x_{ij}\}$ variables, respectively.

4.4.6 Computational results

In this section we present the results of the branch and cut algorithm described in Section 4.4.4 on both TSPDC and TSPB instances. The algorithm has been coded in Fortran 77 and run on a Silicon Graphics Indy (MIPS R4400/200 MHz processor). We have used CPLEX 4.0 (1996) as the LP-solver.

For the TSPDC we consider two classes of test problems, called A, B, respectively. The classes correspond to a subset of the TSPDC instances proposed by Gendreau et. al (1997). The problems of class A is made up of instances derived from symmetric VRP instances from the literature; set A contains 27 instances with *n* ranging between 16 and 151.

The problems of class B consist of random Euclidean instances. The original sets of instances of class B generated by Gendreau et al. is composed of 300 problems. We select a total of 40 instances, with *n* ranging between 26 and 151. Problem input data of class A and B have been kindly provided by Gendreau et al.

For the TSPB we consider two classes of test problems, called A and B, respectively. Both classes of test problems we examine is made up of TSPB instances derived from VRPB instances from the literature. For each VRPB instance we obtain a TSPB instance where the customer set is composed of delivery and collection customers, the depot and the cost matrices are the same as in the VRPB instance. The problem class A is made up of 21 instances derived from the VRPB instances generated by Toth and Vigo (1996) from 11 VRP test problems proposed in the literature. The problem of class B is made up of instances derived from randomly generated Euclidean VRPB instances proposed by Goetschalckx and Jacobs-Blecha (1989). Set A contains 21 instances with *n* between 22 and 100 while set B contains 14 instances with *n* varying between 26 and 151.

Tables 4.7 to 4.10 show the following columns:

Prob:	problem name identifier;
n:	number of customers;
$\|D\|$:	number of delivery customers;
$\|C\|$:	number of collection customers;

Tables 4.7 and 4.8 show the following columns:

$z(UB)$:	cost of the TSPDC solution found by the heuristic algorithm of Gendreau et al. (1997);
z^*:	cost of the optimal TSPDC solution (or cost of the best solution found by the branch and cut algorithm);
LB0:	lower bound obtained by the LP-relaxation of formulation TSPDC;

$\%E_{LB0}$:	percentage error of lower bound *LB0*;
LB1:	final lower bound at the root node of the branch and cut algorithm obtained by the cutting plane algorithm for the identification of the valid inequalities;
$\%E_{LB1}$:	percentage error of lower bound *LB1*;
t_{LB1}:	total computing time of lower bound *LB1*;
t_{BC}:	total computing time of the branch and cut algorithm. We impose a time limit of 3600 seconds. If the time limit is reached, the instance is marked with an asterisk;

Tables 4.9 and 4.10 show the following columns:

z^*:	cost of the optimal TSPB solution (or cost of the best solution found by the branch and cut algorithm);
LB0:	lower bound obtained by the LP-relaxation of formulation TSPB;
$\%E_{LB0}$:	percentage error of lower bound *LB0*;
LB1:	final lower bound at the root node of the branch and cut algorithm obtained by the cutting plane algorithm for the identification of the valid inequalities;
$\%E_{LB1}$:	percentage error of lower bound *LB1*;
t_{LB1}:	total computing time of lower bound *LB1*;
t_{BC}:	total computing time of the branch and cut algorithm. We impose a time limit of 3600 seconds. If the time limit is reached, the instance is marked with an asterisk;

The percentage errors are computed as the ratio of the lower bound divided by z^* and multiplied by 100.

Table 4.7 shows the computational results for the TSPDC on problem instances of class A. The results show that the lower bound is tight, as shown by the average percentage error equal to 99.4. By observing Table 4.7 we note that the branch and cut algorithm has been able to improve significantly on the quality of the heuristic solutions found by Gendreau et al. In fact, the average over the 27 instances of the percentage ratios of the heuristic solution with respect to the TSPDC solution found by the branch and cut algorithm is 103.9. Only one instance (e-151-d) has not been solved to optimality within the imposed time limit, but, the cost of the solution has been reduced from 801 to 703.

Table 4.8 shows the results obtained on problem instances of class B. On this more difficult problem set, only 16 instances out of 40 have been solved to optimality. The inherent difficulty of these instances is testified by the percentage error $\%E_{LB1}$.

Nevertheless, the branch and cut algorithm has been able to substantially improve the quality of the heuristic solutions as testified by the average, equal to 108.0, of the percentage ratios of the heuristic solution with respect to the TSPDC solution found by the branch and cut algorithm.

Tables 4.9 and 4.10 show the computational results for the TSPB on problem instances of class A and B, respectively. For the computation of the optimal solutions using the branch and cut algorithm, we have used an upper bound equal to 1.015 the value of lower bound *LB1* obtained at the root node of the branch and cut tree. From the results we can draw the following conclusions. The lower bounds computed are tight, as shown by the average percentage errors equal to 99.7 and 99.6 for problems of class A and B, respectively. The addition of the valid inequalities substantially improves the value of the lower bound *LB0*. Because of the good quality of the lower bound *LB1*, the branch and cut algorithm has been able to solve to optimality all the 35 instances in reduced computing time. Problems up to 150 customers have been solved to optimality.

Table 4.7. TSPDC: computational results of problem class A

Problem Data						Two-commodity					
Prob	n	$\|D\|$	$\|C\|$	$z(UB)$	z^*	*LB0*	$\%E_{LB0}$	*LB1*	$\%E_{LB1}$	t_{LB1}	t_{BC}
e-016-b	16	7	8	221	218	202.3	92.8	218.0	100.0	1	1
e-016-c	16	7	8	223	221	202.3	91.5	218.0	98.6	1	2
e-016-d	16	7	8	228	221	202.4	91.6	218.0	98.7	1	2
e-026-b	26	12	13	320	306	238.5	77.9	305.0	99.7	1	1
e-026-c	26	12	13	320	308	238.5	77.4	305.0	99.0	1	3
e-026-d	26	12	13	340	313	238.7	76.3	305.0	97.4	1	31
e-036-b	36	17	18	364	351	320.2	91.2	349.5	99.6	2	3
e-036-c	36	17	18	364	351	320.3	91.2	349.5	99.6	2	3
e-036-d	36	17	18	369	351	320.3	91.3	349.5	99.6	2	3
e-045-b	45	22	22	619	619	398.7	64.4	619.0	100.0	4	11
e-045-c	45	22	22	620	619	399.3	64.5	619.0	100.0	4	7
e-045-d	45	22	22	620	619	400.6	64.7	619.0	100.0	4	9
e-051-b	51	25	25	442	426	376.4	88.3	422.5	99.2	2	44
e-051-c	51	25	25	439	426	376.4	88.3	422.5	99.2	2	35
e-051-d	51	25	25	445	426	376.5	88.4	422.5	99.2	3	43
e-076-b	76	37	38	565	538	484.2	90.0	538.0	100.0	5	15
e-076-c	76	37	38	554	539	484.3	89.8	538.0	99.8	5	38
e-076-d	76	37	38	580	539	484.4	89.9	538.0	99.8	5	26
e-101-b	101	50	50	505	501	326.6	65.2	496.5	99.1	58	1021
e-101-c	101	50	50	507	501	326.6	65.2	496.5	99.1	43	1830
e-101-d	101	50	50	513	502	326.7	65.1	498.5	99.3	49	1348
e-121-b	121	60	60	554	535	349.3	65.3	532.0	99.4	43	1322
e-121-c	121	60	60	549	535	349.2	65.3	532.0	99.4	44	3540
e-121-d	121	60	60	556	535	348.2	65.1	532.0	99.4	43	2872
e-151-b	151	75	75	748	698	572.3	82.0	696.0	99.7	22	280
e-151-c	151	75	75	763	699	572.4	81.9	696.0	99.6	24	1990
e-151-d	151	75	75	801	703 *	572.4	81.4	696.0	99.0	51	3600
Averages							79.5		99.4		

Table 4.8. TSPDC: computational results of problem class B

Problem Data						Two-commodity					
Prob	n	$\|D\|$	$\|C\|$	$z(UB)$	z^*	$LB0$	$\%E_{LB0}$	$LB1$	$\%E_{LB1}$	t_{LB1}	t_{BC}
e1_25_b	26	12	13	434	407	309.3	76.0	392.5	96.4	2	540
e1_25_c	26	12	13	459	412 *	309.5	75.1	392.6	95.3	3	3600
e1_25_d	26	12	13	433	433 *	310.0	71.6	392.8	90.7	4	3600
e1_25_e	26	16	9	401	401	311.3	77.6	392.5	97.9	2	4
e2_25_b	26	12	13	477	462	343.3	74.3	457.0	98.9	2	9
e2_25_c	26	12	13	477	462	354.1	76.7	457.0	98.9	2	3
e2_25_d	26	12	13	477	462	354.3	76.7	457.0	98.9	3	4
e2_25_e	26	13	12	462	460	346.5	75.3	457.0	99.3	2	2
e1_50_b	51	25	25	646	592	459.3	77.6	587.5	99.2	4	71
e1_50_c	51	25	25	710	601 *	454.4	75.6	587.5	97.8	4	3600
e1_50_d	51	25	25	645	597 *	454.8	76.2	587.5	98.4	4	3600
e1_50_e	51	25	25	662	634 *	458.9	72.4	587.5	92.7	3	3600
e2_50_b	51	25	25	635	599	490.2	81.8	599.0	100.0	4	5
e2_50_c	51	25	25	635	599	490.2	81.8	599.0	100.0	4	4
e2_50_d	51	25	25	635	599	495.3	82.7	599.0	100.0	3	3
e2_50_e	51	22	28	684	612	498.1	81.4	600.3	98.1	3	1594
e1_75_b	76	37	38	807	710	538.3	75.8	705.7	99.4	15	46
e1_75_c	76	37	38	806	710	538.4	75.8	705.7	99.4	16	95
e1_75_d	76	37	38	807	710	538.8	75.9	705.7	99.4	19	62
e1_75_e	76	42	33	769	728 *	550.9	75.7	711.7	97.8	11	3600
e2_75_b	76	37	38	716	658	555.3	84.4	655.0	99.5	8	36
e2_75_c	76	37	38	716	660	555.4	84.2	655.0	99.2	10	73
e2_75_d	76	37	38	721	677 *	555.7	82.1	655.0	96.8	17	3600
e2_75_e	76	36	39	766	706 *	557.9	79.0	661.0	93.6	11	3600
e1_100_b	101	50	50	832	797	625.4	78.5	790.0	99.1	28	270
e1_100_c	101	50	50	851	805 *	625.6	77.7	790.0	98.1	34	3600
e1_100_d	101	50	50	847	805 *	626.0	77.8	790.0	98.1	41	3600
e1_100_e	101	52	48	903	812 *	628.0	77.3	790.0	97.3	17	3600
e2_100_b	101	50	50	900	810 *	676.1	83.5	781.5	96.5	23	3600
e2_100_c	101	50	50	941	835 *	676.2	81.0	781.5	93.6	24	3600
e2_100_d	101	50	50	980	856 *	674.3	78.8	781.5	91.3	66	3600
e2_100_e	101	43	57	886	809 *	678.1	83.8	789.5	97.6	23	3600
e1_150_b	151	75	75	1059	980 *	768.2	78.4	952.5	97.2	108	3600
e1_150_c	151	75	75	1147	982 *	768.3	78.2	952.5	97.0	108	3600
e1_150_d	151	75	75	1155	1010 *	783.5	77.6	956.5	94.7	127	3600
e1_150_e	151	61	89	1060	984 *	773.7	78.6	952.5	96.8	58	3600
e2_150_b	151	75	75	974	918 *	687.3	74.9	894.6	97.5	218	3600
e2_150_c	151	75	75	984	938 *	687.4	73.3	894.6	95.4	204	3600
e2_150_d	151	75	75	993	947 *	687.6	72.6	877.7	92.7	182	3600
e2_150_e	151	77	73	986	920 *	697.6	75.8	894.6	97.2	89	3600
Averages							77.8		97.2		

Table 4.9. TSPB: computational results of problem class A

PROBLEM DATA				TWO-COMMODITY						
Prob	n	$\|D\|$	$\|C\|$	z^*	*LB0*	$\%E_{LB0}$	*LB1*	$\%E_{LB1}$	t_{LB1}	t_{BC}
eil2250	22	11	10	360	310.4	86.2	360.0	100.0	1	1
eil2266	22	14	7	350	317.9	90.8	350.0	100.0	1	1
eil2280	22	17	4	350	277.7	79.3	347.5	99.3	1	2
eil2350	23	11	11	647	609.5	94.2	647.0	100.0	1	1
eil2366	23	15	7	626	531.1	84.8	618.0	98.7	1	2
eil2380	23	18	4	639	587.6	91.9	639.0	100.0	1	1
eil3050	30	15	14	529	381.1	72.0	529.0	100.0	2	2
eil3066	30	20	9	504	363.1	72.0	504.0	100.0	2	2
eil3080	30	24	5	510	360.8	70.7	510.0	100.0	2	2
eil3350	33	16	16	575	518.2	90.1	572.3	99.5	2	3
eil3366	33	22	10	579	517.5	89.4	574.0	99.1	2	2
eil3380	33	26	6	543	458.9	84.5	543.0	100.0	2	2
eil5150	51	25	25	561	500.8	89.3	560.0	99.8	3	4
eil5166	51	34	16	522	459.0	87.9	522.0	100.0	2	2
eil5180	51	40	10	537	476.5	88.7	533.5	99.3	3	8
eila7650	76	37	38	683	622.0	91.1	681.0	99.7	6	12
eila7666	76	50	25	711	651.0	91.6	709.5	99.8	5	12
eila7680	76	60	15	661	595.1	90.0	659.0	99.7	5	48
eila10150	101	50	50	809	685.1	84.7	804.9	99.5	13	227
eila10166	101	67	33	814	715.8	87.9	813.3	99.9	12	14
eila10180	101	80	20	793	690.4	87.1	790.0	99.6	14	71
Averages						85.9		99.7		

Table 4.10. TSPB: computational results of problem class B

PROBLEM DATA				TWO-COMMODITY							
Prob	n	$\lvert D\rvert$	$\lvert C\rvert$	z^*	*LB0*	$\%E_{LB0}$	*LB1*	$\%E_{LB1}$	t_{LB1}	t_{BC}
A1	26	20	5	147639	122783.1	83.2	147005.0	99.6	1	2
B1	31	20	10	176414	151422.7	85.8	176414.0	100.0	2	2
C1	41	20	20	188006	160470.3	85.4	188006.0	100.0	2	2
D1	39	30	8	165101	139427.7	84.4	165101.0	100.0	2	2
E1	46	30	15	204970	169430.3	82.7	204198.0	99.6	3	3
F1	61	30	30	226396	172594.9	76.2	226396.0	100.0	4	5
G1	58	45	12	213732	165548.0	77.5	212251.8	99.3	4	6
H1	69	45	23	248858	200245.3	80.5	247744.5	99.6	5	7
I1	91	45	45	287499	228182.8	79.4	285643.5	99.4	9	31
J1	95	75	19	266839	211802.4	79.4	264295.0	99.0	11	496
K1	114	75	38	304183	257096.9	84.5	303840.5	99.9	16	18
L1	151	75	75	366616	296481.0	80.9	364714.5	99.5	75	222
M1	126	100	25	312814	265271.6	84.8	310524.0	99.3	37	795
N1	151	100	50	349013	290647.6	83.3	347695.5	99.6	36	55
Averages						82.0		99.6		

4.5 SUMMARY

We have investigated new integer programming formulations for routing problems which are based on the two-commodity network flow formulation of the TSP described by Finke et al. (1984). This formulation is interesting in many different ways. It can be shown that its LP-relaxation satisfies a weak form of the subtour elimination constraints. The formulation can also be modified to accommodate different constraints and, therefore, is capable of being extended to different routing problems. In this chapter, a new two-commodity network flow formulation for the symmetric TSP (STSP) has been derived and extended to derive new integer programming formulations for a class of different routing problems. The VRP has been examined in which a fleet of M vehicles stationed at a central depot is to be optimally routed to supply customers with known demands subject to vehicle capacity constraints. We investigated a new integer programming formulation for the VRP and a new lower bound based on the linear relaxation of the two-commodity formulation. The lower bound, strengthened by a set of valid inequalities, has been embedded in a branch and cut procedure to solve the problem optimally. The computational results on a set of problem instances derived from the literature show that the lower bound obtained is tight and that the branch and cut algorithm has been able to solve to optimality problems up to 100 customers.

The STSP formulation has also been extended to deal with other routing constraints such as delivery and collection constraints. We considered the TSP with mixed deliveries and collections (TSPDC) in which a vehicle located at a central depot must be optimally used to serve a set of customers partitioned into two subsets of delivery and collection customers. The vehicle capacity must not be exceeded along the tour and the total length of the tour must be minimized. A new mathematical formulation has been derived for the TSPDC and another one for the special case, known as TSP with Backhauls, where in any feasible solution all delivery customers must precede the collection customers. New lower bounds have been obtained from the linear relaxation of these formulations which have been further strengthened by valid inequalities and embedded in a branch and cut procedure to solve the problems optimally. The resulting cutting plane procedure has been applied to a set of instances taken from the literature and involving problems up to 150 customers. The results show that the branch and cut algorithm has been able to solve to optimality problems up to 150 customers.

Future research can focus on the investigation of other valid inequalities and on the extension of the two-commodity network flow model to other routing problems such as the VRP with Backhauls and the Multi-Depot Vehicle Routing Problem.

CHAPTER 5

AN EXACT METHOD FOR THE VEHICLE ROUTING PROBLEM WITH BACKHAULS

5.1 INTRODUCTION

The Vehicle Routing Problem with Backhauls (VRPB) considered in this chapter is an extension of the Vehicle Routing Problem (VRP) and is defined as follows. A set of capacitated vehicles stationed at a central depot are to be used to design a number of routes, each one starting and ending at the depot, in order to supply a set of customers (called *Linehaul* customers) requiring deliveries from the depot and to collect products from a set of customers (called *Backhaul* customers) to be unloaded at the depot. Each customer is visited exactly once and in each feasible route the Backhaul customers are visited after all Linehaul customers are supplied. The total load supplied to the Linehaul customers as well as the total load collected from the Backhaul customers must not exceed, separately, the vehicle capacity. The objective is to minimize the total distance travelled. The VRPB is known to be NP-hard in the strong sense.

Several heuristic algorithms for the solution of VRPB have been presented in the literature. Deif and Bodin (1984) proposed an extension of the well-known Clarke-Wright VRP heuristic, where the saving of the arcs connecting Linehaul to Backhaul customers are modified so as to delay the formation of mixed routes. Different extensions of the Clarke and Wright algorithm have been presented by Casco et al. (1988), and Golden et al. (1985) (in this latter paper the precedence constraint between Linehaul and Backhaul customers is not present). Goetschalckx and Jacobs-Blecha (1989) proposed an algorithm that builds initial routes with customers of the same type by using a spacefilling curves heuristic; these routes are then merged to form the final set of vehicle routes. More recently, Goetschalckx and Jacobs-Blecha (1993) described an extension to VRPB of the Fisher-Jaikumar VRP heuristic. Toth and Vigo (1996) proposed a different cluster-first-route-second heuristic based on a Lagrangean relaxation of a new formulation of the VRPB, which starts from a (possibly infeasible) solution, and tries to improve it through a local search procedure based on inter-route and intra-route arc exchanges. Computational results show that the algorithm of Toth and Vigo outperforms both the heuristics of Goetschalckx and Jacobs-Blecha (1989) and (1993). Anily (1996) describes a heuristic method for the VRPB that converges to the optimal solution, under mild probabilistic conditions and when there are no restrictions on the order in which the Linehaul and Backhaul customers are visited. A 1.5 approximation algorithm for the single vehicle version of the VRPB is given by Gendreau et al. (1997).

Yano et al. (1987) proposed an exact, set-covering based, algorithm for the special case of VRPB in which the number of customers of each type in a circuit is not greater than four. The variant of VRPB where time window constraints are present has also been considered in the literature (see, e.g., Kontovradis and Bard (1995), Duhamel et al. (1994) and Gélinas et al. (1995)). Toth and Vigo (1997) present an exact approach for the solution of the VRPB with both symmetric and asymmetric cost matrices. They first give a new integer linear programming model where the VRPB is reformulated as an asymmetric problem. The model is used to derive a Lagrangian lower bound, based on projection of the solution space, which requires the determination of the shortest spanning arborescences with fixed degree at the depot vertex, and the optimal solution of min-cost flow problems. The Lagrangian lower bound is strengthened in a cutting-plane fashion, by separating valid inequalities, and is combined, according to the additive approach proposed by Fischetti and Toth (1989), with a lower bound obtained by dropping the capacity constraints. A branch-and-bound algorithm which makes use of reduction procedures, dominance criteria, feasibility checks and a heuristic algorithm are also presented.

In this chapter we describe a new (0-1) integer programming formulation of the VRPB based upon a set-partitioning approach.

We use a heuristic procedure to solve the dual problem, called D, of the LP-relaxation of the formulation in order to obtain a valid lower bound to the VRPB. This procedure, called HDS, combines two different heuristic algorithms each one finding a feasible solution to D without requiring the entire set of the dual constraints. The dual solution thus obtained and a valid upper bound to the VRPB are then used to reduce the number of routes (e.g., the variables of the integer formulation) which may form an optimal solution. However, the size of the reduced integer problem might still be too large for solving it by a branch and bound method. In this case we propose a procedure, called EHP, that consists of reducing the number of variables of the integer program so that the resulting problem can be solved by an integer programming solver. This method may terminate without having found an optimal solution. However, procedure EHP provides a means to estimate the maximum deviation from optimality of the VRPB solution obtained.

The chapter is organized as follows. Section 5.2 describes the basic VRPB. Section 5.3 gives the new integer programming model for VRPB and presents different pricing procedures for reducing the variables of the ILP model. Section 5.4 presents the bounding procedure HDS, while algorithm EHP is described in Section 5.5. Computational results are given in Section 5.6, followed by a summary in Section 5.7.

5.2 THE BASIC VRPB

Let $G=(V,A)$ be a directed graph such that $V=\{0\}\cup L\cup B$, where $L=\{1,\dots,n\}$ corresponds to n Linehaul customers, $B=\{n+1,\dots,n+m\}$ corresponds to m Backhaul customers and the vertex 0 represents the depot. A non-negative cost d_{ij} is associated with each arc $(i,j)\in A$ and a non-negative integer quantity q_i is associated with each customer $i\in L\cup B$. A fleet of M identical vehicles of capacity Q is located at the depot and must be used to supply the Linehaul customers and make collections from the Backhaul customers.

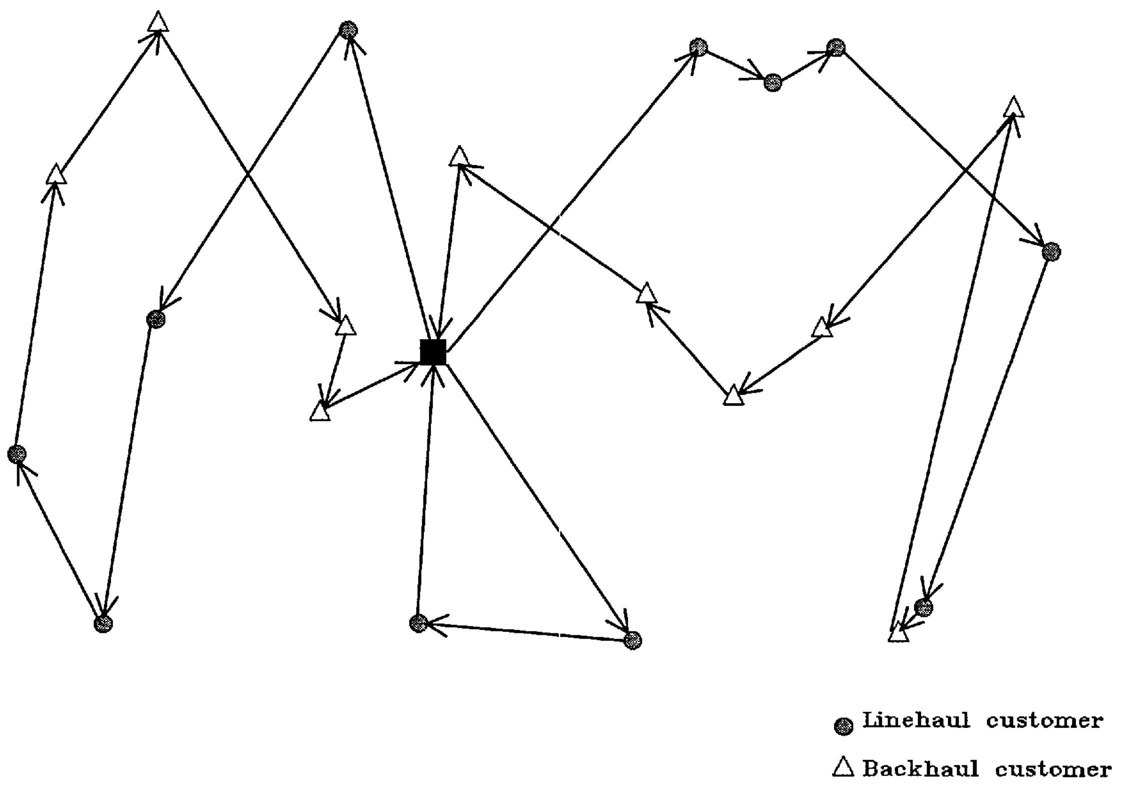


Figure 5.1. Example of a VRPB solution

It is required that every route performed by a vehicle starts and ends at the depot and that the load of all Linehauls and Backhauls does not exceed, separately, the vehicle capacity. Furthermore, in any feasible route, all Linehaul customers must precede all Backhaul customers. The cost of a route corresponds to the sum of the cost of the arcs forming such route. The problem we consider is of designing M routes, one for each vehicle, so that each customer is visited exactly once and the sum of the route costs is minimized. Figure 5.1 shows an example of a VRPB solution.

It is easy to see that if $L = \varnothing$ or $B = \varnothing$ then the VRPB reduces to the VRP, proving that the problem is NP-hard (see Garey and Johnson (1979)).

In order to ensure feasibility, we assume that $M \geq Max[M_L, M_B]$, where M_L (resp. M_B) is the minimum number of vehicles needed to visit all the Linehaul (resp.

Backhaul) customers. The values M_L and M_B can be computed by solving the Bin Packing Problem (see Martello and Toth (1990)) associated with the Linehaul and Backhaul customers. Following Toth and Vigo (1997), we assume that routes containing only Backhaul customers are not allowed (that is $M_L \geq M_B$). We must point out that the method we are going to describe for solving the VRPB can be easily extended to deal with the case where $M_L < M_B$.

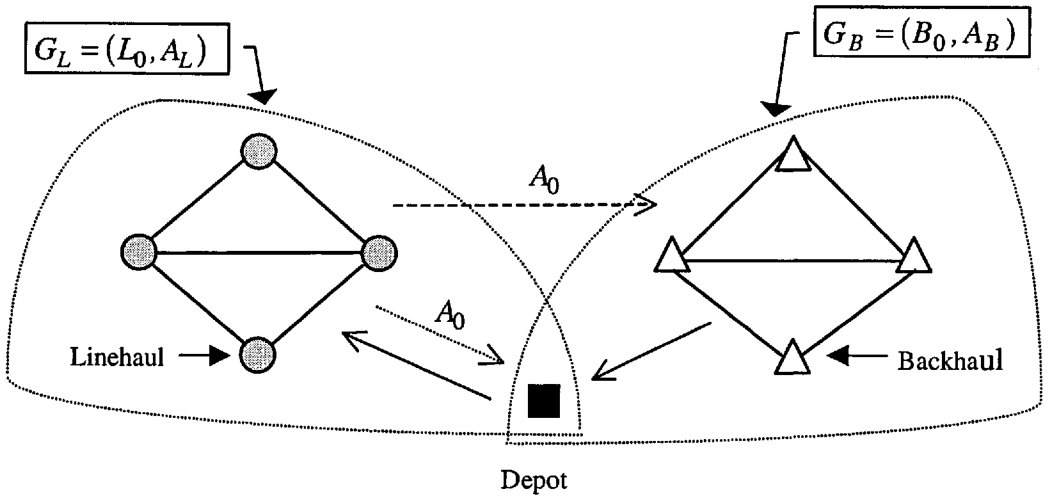


Figure 5.2. The arc set of graph G

Let us denote by $G_L = (L_0, A_L)$ and $G_B = (B_0, A_B)$ the two subgraphs of G induced by the Linehaul and Backhaul customers, respectively, where :

$$\left.\begin{array}{l} L_0 = L \cup \{0\} \text{ and } A_L = \{(i,j) : (i,j) \in A \text{ s.t. } i,j \in L_0\} \\ B_0 = B \cup \{0\} \text{ and } A_B = \{(i,j) : (i,j) \in A \text{ s.t. } i,j \in B_0\} \end{array}\right\} \tag{5.1}$$

Let us define $A_0 = \{(i,j) : (i,j) \in A \text{ s.t. } i \in L, j \in B_0\}$. Figure 5.2 shows the arc set of graph G.

An elementary path P in G_L starting at vertex 0 (resp. in G_B ending at vertex 0) is called a *feasible path* if its load satisfies the following inequalities :

$$Q^L_{\min} \leq \sum_{i \in P} q_i \leq Q \quad \text{(resp. } Q^B_{\min} \leq \sum_{i \in P} q_i \leq Q\text{)} \tag{5.2}$$

where $Q^L_{\min}$ (resp. $Q^B_{\min}$) represents the minimum load of Linehaul customers (resp. Backhaul customers) of any feasible path in G_L (resp. G_B).

The values $Q^L_{\min}$ and $Q^B_{\min}$ are computed as follows :

$$Q^L_{min} = Max\left[0, \left(\sum_{i \in L} q_i\right) - (M-1)Q\right] \text{ and } Q^B_{min} = Max\left[0, \left(\sum_{i \in B} q_i\right) - (M-1)Q\right]$$

We will use $t(P)$ to indicate both the terminal vertex of a feasible path P in G_L and the starting vertex of a feasible path P in G_B. Note that any pair of feasible paths P in G_L and P' in G_B and the arc $(t(P), t(P')) \in A_0$ form a feasible route that is obtained by appending to the end of P the arc $(t(P), t(P'))$ and the path P'. Furthermore, any feasible path P in G_L leads to a feasible route involving Linehaul customers only by appending to P the arc $(t(P), 0) \in A_0$. Since we assume $M_L \geq M_B$, no feasible route exists involving Backhaul customers only. Finally, we observe that the M routes of any feasible VRPB solution consist of M feasible paths in G_L, at least M_B feasible paths in G_B and M arcs of the subset A_0.

5.3 A MATHEMATICAL FORMULATION OF THE VRPB

Let $\mathcal{L}$ be the index set of all feasible paths in G_L. We denote with $\mathcal{L}_i \subseteq \mathcal{L}$ (resp. $\mathcal{L}^E_i \subseteq \mathcal{L}$) the index set of all paths passing through (resp. ending at) customer $i \in L$.

Let $\mathcal{B}$ be the index set of all feasible paths in G_B. We denote by $\mathcal{B}_i \subseteq \mathcal{B}$ (resp. $\mathcal{B}^S_i \subseteq \mathcal{B}$) the index set of all paths passing through (resp. starting at) customer $i \in B$. We indicate with c_ℓ the cost of path $\ell \in \mathcal{L} \cup \mathcal{B}$. In the following we will use $t(P_\ell)$ and/or t_ℓ, to denote the terminal vertex of path P_ℓ, if $\ell \in \mathcal{L}$, or the starting vertex of path P_ℓ, if $\ell \in \mathcal{B}$.

Let us define the following binary variables: x_ℓ, $\ell \in \mathcal{L}$, y_ℓ, $\ell \in \mathcal{B}$ and ξ_{ij}, $(i,j) \in A_0$. We have $x_\ell = 1$, $y_{\ell'} = 1$ and $\xi_{ij} = 1$ if and only if paths $\ell \in \mathcal{L}$, $\ell' \in \mathcal{B}$ and arc $(i,j) \in A_0$ are in the optimal VRPB solution.

An integer programming formulation of the VRPB is as follows :

$$\text{(IP)} \qquad z(\text{IP}) = Min \ \sum_{\ell \in \mathcal{L}} c_\ell x_\ell + \sum_{\ell \in \mathcal{B}} c_\ell y_\ell + \sum_{(i,j) \in A_0} d_{ij} \xi_{ij} \tag{5.3}$$

$$subject\ to \ \sum_{\ell \in \mathcal{L}_i} x_\ell = 1 , \qquad \forall i \in L \tag{5.4}$$

$$\sum_{\ell \in \mathcal{B}_j} y_\ell = 1 , \qquad \forall j \in B \tag{5.5}$$

$$\sum_{\ell \in \mathcal{L}_i^E} x_\ell - \sum_{j \in B_0} \xi_{ij} = 0 , \qquad \forall i \in L \tag{5.6}$$

$$\sum_{\ell \in \mathcal{B}_j^S} y_\ell - \sum_{i \in L} \xi_{ij} = 0 , \qquad \forall j \in B \tag{5.7}$$

$$\sum_{(i,j) \in A_0} \xi_{ij} = M \tag{5.8}$$

$$x_\ell \in \{0,1\}, \quad \forall \ell \in \mathcal{L}, \ y_\ell \in \{0,1\}, \forall \ell \in \mathcal{B}, \xi_{ij} \in \{0,1\} , \quad \forall (i,j) \in A_0 \tag{5.9}$$

Equations (5.4) and (5.5) require that each vertex $i \in L$ and $j \in B$ be visited by one route. Equation (5.6) forces the solution to contain an arc of A_0 starting at vertex $i \in L$ whenever such solution contains a feasible path in G_L ending at vertex $i \in L$. Equation (5.7) requires in the solution an arc (i,j) with $i \in L$ if such a solution contains a feasible path in G_B starting at vertex $j \in B$. Equation (5.8) forces the solution to contain M routes by requiring that M arcs of the set A_0 are in solution. Since the set A_0 contains all arcs $(i,0), \forall i \in L$, routes containing only Linehaul customers are allowed.

Problem IP cannot be solved directly, even for problems of moderate size as the number of variables may be too large. In this chapter we describe a heuristic procedure that finds a feasible solution of the dual problem D of the linear relaxation of IP, thus providing a valid lower bound to the VRPB. This procedure does not require the explicit generation of the path set $\mathcal{L}$ and $\mathcal{B}$. Moreover, this dual solution is used by the exact procedure that solves the VRPB to reduce drastically sets $\mathcal{L}$ and $\mathcal{B}$ removing those paths that cannot belong to any optimal VRPB solution.

Let u_i, $i \in L$ and $v_j, j \in B$, be the dual variables associated with constraints (5.4) and (5.5), respectively. Indicate by α_i, $i \in L$, and $\beta_j, j \in B$, the dual variables associated with constraints (5.6) and (5.7), respectively. Finally, associate with constraint (5.8) the dual variable w.

The dual of the LP-relaxation of IP is the following :

$$\text{(D)} \qquad z(\text{D}) = Max \sum_{i \in L} u_i + \sum_{j \in B} v_j + Mw \tag{5.10}$$

$$subject\ to \quad \sum_{k \in P_\ell} u_k + \alpha_i \le c_\ell , \qquad \forall \ell \in \mathcal{L}_i^E , i \in L \tag{5.11}$$

$$\sum_{k \in P_\ell} v_k + \beta_j \le c_\ell , \qquad \forall \ell \in \mathcal{B}_j^S , j \in B \tag{5.12}$$

$$-\alpha_i - \beta_j + w \le d_{ij} , \qquad \forall (i,j) \in A_0 \tag{5.13}$$

$$\left.\begin{array}{lll} u_i , \alpha_i & \text{unrestricted ,} & \forall i \in L \\ v_j , \beta_j & \text{unrestricted ,} & \forall j \in B \\ w & \text{unrestricted} & \end{array}\right\} \tag{5.14}$$

Note that we assume $\beta_0 = 0$ in the dual constraints (5.13), $\forall (i,0) \in A_0$.

5.3.1 Variable reduction of problem IP

Let $(\mathbf{u}', \mathbf{v}', \alpha', \beta', w')$ be a feasible solution of D of cost $z'(\text{D})$ and let $(\mathbf{x}', \mathbf{y}', \xi')$ be a feasible solution of IP of cost $z'(\text{IP})$. We denote by c'_ℓ and d'_{ij} the reduced costs according to $(\mathbf{u}', \mathbf{v}', \alpha', \beta', w')$ of each path $\ell \in \mathcal{L} \cup \mathcal{B}$ and each arc $(i,j) \in A_0$, respectively, that is :

$$\left.\begin{array}{ll} c'_\ell = c_\ell - \sum_{k \in P_\ell} u'_k - \alpha'_i , & \ell \in \mathcal{L}_i^E , i \in L \\ c'_\ell = c_\ell - \sum_{k \in P_\ell} v'_k - \beta'_j , & \ell \in \mathcal{B}_j^S , j \in B \\ d'_{ij} = d_{ij} + \alpha'_i + \beta'_j - w' , & (i,j) \in A_0 \end{array}\right\} \tag{5.15}$$

From linear programming duality we have $z'(\text{D}) \le z'(\text{IP})$. Furthermore, we can use these two solutions to reduce the variables of IP as is established by the following theorem.

Theorem 5.1. Let $X=\{\ell : \ell \in \mathcal{L}, \text{ s.t. } x'_\ell = 1\}$, $Y=\{\ell : \ell \in \mathcal{B}, \text{ s.t. } y'_\ell = 1\}$ and $H=\{(i,j) : (i,j) \in A_0, \text{ s.t. } \xi'_{ij} = 1\}$. The following relationship holds :

$$z'(\mathrm{IP}) = z'(\mathrm{D}) + \sum_{\ell\in X} c'_\ell + \sum_{\ell\in Y} c'_\ell + \sum_{(i,j)\in H} d'_{ij} \tag{5.16}$$

Proof. From equations (5.15) we have :

$$\begin{aligned}\sum_{\ell\in X} c'_\ell + \sum_{\ell\in Y} c'_\ell + \sum_{(i,j)\in H} d'_{ij} &= \sum_{\ell\in X} c_\ell - \sum_{\ell\in X}\left(\sum_{k\in P_\ell} u'_k + \alpha'_{t_\ell}\right) \\ &+ \sum_{\ell\in Y} c_\ell - \sum_{\ell\in Y}\left(\sum_{k\in P_\ell} v'_k + \beta'_{t_\ell}\right) \\ &+ \sum_{(i,j)\in H} d_{ij} + \sum_{(i,j)\in H}\left(\alpha'_i + \beta'_j - w'\right)\end{aligned} \tag{5.17}$$

Since $(\mathbf{x}', \mathbf{y}', \xi')$ is a feasible solution of IP, we have :

$$\sum_{\ell\in X}\left(\sum_{k\in P_\ell} u'_k + \alpha'_{t_\ell}\right) = \sum_{i\in L} u'_i + \sum_{\ell\in X} \alpha'_{t_\ell} \tag{5.18}$$

$$\sum_{\ell\in Y}\left(\sum_{k\in P_\ell} v'_k + \beta'_{t_\ell}\right) = \sum_{i\in B} v'_i + \sum_{\ell\in Y} \beta'_{t_\ell} \tag{5.19}$$

$$\sum_{(i,j)\in H}\left(\alpha'_i + \beta'_j - w'\right) = \sum_{\ell\in X} \alpha'_{t_\ell} + \sum_{\ell\in Y} \beta'_{t_\ell} - Mw' \tag{5.20}$$

and, hence, from expression (5.17) we obtain :

$$\begin{aligned}\sum_{\ell\in X} c'_\ell + \sum_{\ell\in Y} c'_\ell + \sum_{(i,j)\in H} d'_{ij} &= \sum_{\ell\in X} c_\ell + \sum_{\ell\in Y} c_\ell + \sum_{(i,j)\in H} d_{ij} \\ &- \sum_{i\in L} u'_i - \sum_{j\in B} v'_j - Mw'\end{aligned} \tag{5.21}$$

Note that $z'(\mathrm{IP}) = \sum_{\ell\in X} c_\ell + \sum_{\ell\in Y} c_\ell + \sum_{(i,j)\in H} d_{ij}$ and $z'(\mathrm{D}) = \sum_{i\in L} u'_i + \sum_{j\in B} v'_j + Mw'$, therefore, from equation (5.21) we obtain equation (5.16)■

Corollary 5.1. Let $z(\text{UB})$ be the cost of a feasible VRPB solution and $(\mathbf{u}',\mathbf{v}',\alpha',\beta',w')$ be a feasible solution of D of cost $z'(\text{D})$. Any optimal solution of IP of cost less than $z(\text{UB})$ cannot contain any path $\ell \in \mathcal{L} \cup \mathcal{B}$ or any arc $(i,j) \in A_0$ whose reduced cost is greater or equal to $z(\text{UB}) - z'(\text{D})$.

Proof. Follows directly from Theorem 5.1.

Corollary 5.1 states that an optimal VRPB solution can be obtained by replacing in problem IP sets $\mathcal{L}$, $\mathcal{B}$ and A_0 with subsets $\mathcal{L}' \subseteq \mathcal{L}$, $\mathcal{B}' \subseteq \mathcal{B}$ and $A_0' \subseteq A_0$ defined as follows.

$$\left.\begin{aligned} \mathcal{L}' &= \{\ell : \ell \in \mathcal{L}, \text{s.t.}\, c_\ell' < z(\text{UB}) - z'(\text{D})\} \\ \mathcal{B}' &= \{\ell : \ell \in \mathcal{B}, \text{s.t.}\, c_\ell' < z(\text{UB}) - z'(\text{D})\} \\ A_0' &= \{(i,j) : (i,j) \in A_0\, , \text{s.t.}\, d_{ij}' < z(\text{UB}) - z'(\text{D})\} \end{aligned}\right\} \tag{5.22}$$

Note that expressions (5.22) require the computation of the reduced costs of the paths of $\mathcal{L}$ and $\mathcal{B}$ and of the arcs of A_0. The effectiveness of expressions (5.22) increases if the gap between upper bound $z(\text{UB})$ and lower bound $z'(\text{D})$ is small.

5.3.2 Further Variable Reduction

A further reduction of sets $\mathcal{L}'$, $\mathcal{B}'$ and A_0' can be achieved by means of the following observation.

Reduction of $\mathcal{L}'$

Consider a VRPB solution containing a given path $\ell \in \mathcal{L}'$. If this solution is feasible, it must also contain a path from $t_\ell \in L$ to the depot to complete the route emerging from ℓ. We have two cases :

(i) the path completing ℓ consists of arc $(t_\ell, 0)$. In this case, from Theorem 5.1, the resulting VRPB solution cannot be smaller than

$$z'(\mathrm{D}) + c'_\ell + d'_{t_\ell 0}; \tag{5.23}$$

(ii) the path completing ℓ starts at t_ℓ, goes directly to some Backhaul customer (say j) and returns to the depot passing through Backhaul customers. This VRPB solution has a cost not smaller than

$$z'(\mathrm{D}) + c'_\ell + \underset{j\in B}{Min}\left[d'_{t_\ell j} + \underset{r\in \mathcal{B}'^S_j}{Min}\left[c'_r\right]\right] \tag{5.24}$$

where $\mathcal{B}'^S_j = \mathcal{B}^S_j \cap \mathcal{B}'$, $j \in B$.

Corollary 5.2. An optimal VRPB solution cannot contain any path $\ell \in \mathcal{L}'$ such that

$$c'_\ell + \underset{j\in B_0}{Min}\left[d'_{t_\ell j} + \underset{r\in \mathcal{B}'^S_j}{Min}\left[c'_r\right]\right] \geq z(\mathrm{UB}) - z'(\mathrm{D}), \tag{5.25}$$

where we assume $\underset{r\in \mathcal{B}'^S_0}{Min}\left[c'_r\right] = 0$.

Reduction of $\mathcal{B}'$

Observations similar to those made to establish Corollary 5.2 lead to the following corollary.

Corollary 5.3. An optimal VRPB solution cannot contain any path $\ell \in \mathcal{B}'$ such that :

$$c'_\ell + \underset{i\in L}{Min}\left[\underset{r\in \mathcal{L}'^E_i}{Min}\left[c'_r\right] + d'_{i t_\ell} \right] \geq z(\mathrm{UB}) - z'(\mathrm{D}), \tag{5.26}$$

where $\mathcal{L}'^E_i = \mathcal{L}^E_i \cap \mathcal{L}'$, $i \in L$.

Reduction of A'_0

Any VRPB solution containing arc $(i,j) \in A'_0$ must contain either a path of set $\mathcal{L}'^E_i$ and a path of $\mathcal{B}'^S_j$, if $j \neq 0$, or arc $(i,0)$, if $j = 0$. Hence, from Theorem 5.1 we have the following corollary.

Corollary 5.4. An optimal VRPB solution cannot contain arc $(i,j) \in A_0$ if

$$\underset{\ell \in \mathcal{L}_i'^E}{Min}[c'_\ell] + d'_{ij} + \underset{r \in \mathcal{B}_j'^S}{Min}[c'_r] \geq z(\mathrm{UB}) - z'(\mathrm{D}). \qquad (5.27)$$

Note that the tests described might be insufficient to reduce the size of sets $\mathcal{L}'$ and $\mathcal{B}'$ so that problem IP can become solvable. This might happen even if the gap $z(\mathrm{UB}) - z'(\mathrm{D})$ is small. In section 5.5 we describe a procedure to reduce $\mathcal{L}'$ and $\mathcal{B}'$ so that the resulting problem IP can be solved but without any guarantee that the solution obtained is an optimal VRPB solution. However, the reduction of sets $\mathcal{L}'$ and $\mathcal{B}'$ is such that it is possible to estimate the maximum distance from optimality of the solution obtained.

5.4 A HEURISTIC PROCEDURE FOR SOLVING PROBLEM D

It is well known, from linear programming duality, that the cost of any feasible solution to D is a lower bound to the optimal solution cost of IP. In this section we describe a heuristic procedure (called HDS) for finding a feasible solution to D that is based on the following general idea. A feasible solution $\mathbf{w} = \mathbf{w}^1 + \mathbf{w}^2 + \cdots + \mathbf{w}^k$ of the linear program :

$$\text{(LP)} \qquad \begin{aligned} &Max && \mathbf{wb} \\ &subject\ to && \mathbf{wA} \leq \mathbf{c} \\ & && \mathbf{w}\ \text{unrestricted} \end{aligned}$$

can be obtained by successively solving a sequence of k linear programs $\mathrm{LP}^1, \mathrm{LP}^2, \ldots, \mathrm{LP}^k$ by means of k different heuristic procedures $\mathrm{H}^1, \mathrm{H}^2, \ldots, \mathrm{H}^k$. Procedure H^r finds a feasible solution $\mathbf{w}^r$ of the linear program LP^r defined as follows:

$$(\mathrm{LP}^r) \qquad \begin{aligned} &Max && \mathbf{w}^r\mathbf{b} \\ &subject\ to && \mathbf{w}^r\mathbf{A} \leq \mathbf{c}^r \\ & && \mathbf{w}^r\ \text{unrestricted} \end{aligned}$$

where $\mathbf{c}^r = \mathbf{c} - \left(\mathbf{w}^0 + \mathbf{w}^1 + \cdots + \mathbf{w}^{r-1}\right)\mathbf{A}$ and $\mathbf{w}^0 = \mathbf{0}$. Note that linear program LP^1 coincides with LP.

This general method has been applied by Mingozzi et al. (1994) for solving the Vehicle Routing Problem, by Bianco et al. (1994) for the Multiple Depot Vehicle Scheduling, by Mingozzi et al. (1995) for the Crew Scheduling Problem and has been used in Chapter 3 for the CPMP.

The procedure HDS that we propose for solving problem D involves two heuristic procedures H^1 and H^2 used in sequence. Procedure H^1 finds a feasible solution $(\mathbf{u}^1, \mathbf{v}^1, \alpha^1, \beta^1, w^1)$ of problem $D^1(\equiv D)$ without requiring the generation of sets $\mathcal{L}$ and $\mathcal{B}$. The second procedure, H^2, solves problem D^2 that is obtained from D by replacing the path costs c_ℓ, $\ell \in \mathcal{L} \cup \mathcal{B}$ and the arc costs d_{ij}, $(i,j) \in A_0$ with the reduced costs c'_ℓ and d'_{ij} computed according to the solution $(\mathbf{u}^1, \mathbf{v}^1, \alpha^1, \beta^1, w^1)$ obtained by procedure H^1. Procedure H^2 requires the generation of limited subsets of sets $\mathcal{L}$ and $\mathcal{B}$.

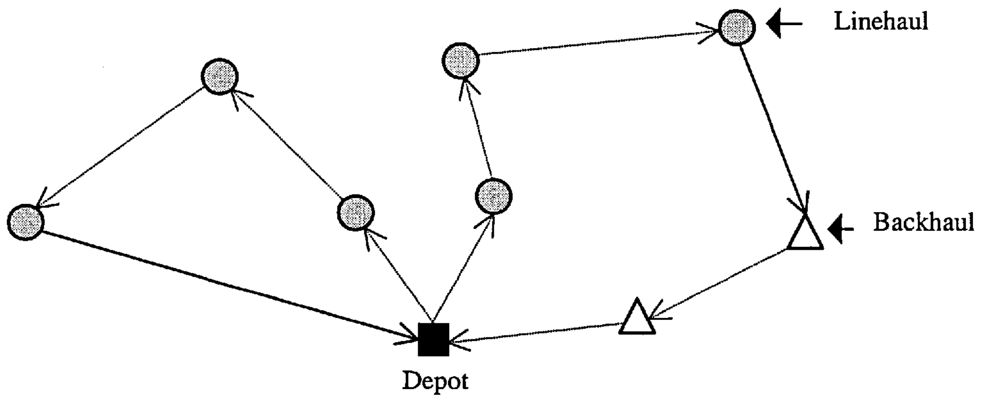


Figure 5.3. Structure of a feasible VRPB solution

5.4.1 PROCEDURE H^1

This procedure is based on the observation that any route of a feasible VRPB solution contains an arc of set A_0 (see Figure 5.3). A lower bound to the VRPB can be obtained as follows. Associate to each arc $(i,j) \in A_0$ a cost representing a lower bound to the cost of the least cost route passing through it. Therefore the sum on the costs of the M vertex-disjoint arcs of minimum cost of A_0 is a valid lower bound to the VRPB.

This problem, called $PR(\lambda, \mu)$, is defined as follows.

Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $\mu = (\mu_1, \mu_2, \ldots, \mu_m)$ be two vectors of unrestricted real numbers associated with the Linehaul and Backhaul customers, respectively.

Let us associate with each arc $(i,j) \in A$ a cost $\bar{d}_{ij}$ as follows :

$$\bar{d}_{ij} = \begin{cases} d_{ij} - \lambda_j \text{ if } j \in L \\ d_{ij} - \mu_j \text{ if } j \in B \end{cases} \tag{5.28}$$

Denote by φ_i^L (resp. φ_j^B) a lower bound to the cost of the least cost feasible path ending at vertex i of G_L (resp. starting at vertex j of G_B) using arc costs $\{\bar{d}_{ij}\}$ defined by expressions (5.28). Therefore, the values φ_i^L, $i \in L$, and φ_j^B, $j \in B$, satisfy the following inequalities :

$$\left.\begin{array}{l} \varphi_i^L \le c_\ell - \sum\limits_{k \in P_\ell} \lambda_k \text{ , } \ell \in \mathcal{L}_i^E \text{ , } i \in L \\ \varphi_j^B \le c_\ell - \sum\limits_{k \in P_\ell} \mu_k \text{ , } \ell \in \mathcal{B}_j^S \text{ , } j \in B \end{array}\right\} \tag{5.29}$$

A valid lower bound b_{ij} to the cost of the least cost route passing through arc $(i,j) \in A_0$ can be computed as follows.

$$b_{ij} = \varphi_i^L + d_{ij} + \varphi_j^B \text{ , } \quad \forall (i,j) \in A_0 \tag{5.30}$$

where we assume $\varphi_0^B = 0$. In subsection 5.4.1.2 we describe a method for computing φ_i^L and φ_j^B that does not require the enumeration of path sets $\mathcal{L}$ and $\mathcal{B}$.

The mathematical formulation of $\text{PR}(\lambda, \mu)$ is as follows.

$$\text{PR}(\lambda,\mu) \quad z(\text{PR}(\lambda,\mu)) = Min \sum_{(i,j) \in A_0} b_{ij}\xi_{ij} + \sum_{i \in L} \lambda_i + \sum_{j \in B} \mu_j \tag{5.31}$$

$$subject\ to \quad \sum_{j \in B_0} \xi_{ij} \le 1 \text{ , } \qquad \forall i \in L \tag{5.32}$$

$$\sum_{i \in L} \xi_{ij} \le 1 \text{ , } \qquad \forall j \in B \tag{5.33}$$

$$\sum_{(i,j) \in A_0} \xi_{ij} = M \tag{5.34}$$

$$\xi_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A_0 \qquad (5.35)$$

Constraints (5.32) and (5.33) force the arcs of a feasible $\text{PR}(\lambda,\mu)$ solution to be vertex-disjoint and constraint (5.34) requires that exactly M arcs are in the solution.

Let $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$ and $\nu = (\upsilon_1, \upsilon_2, \ldots, \upsilon_m)$ be two vectors of dual variables associated with constraints (5.32) and (5.33), respectively, and let δ be the dual variable associated with constraint (5.34).

The dual of the LP-relaxation of $\text{PR}(\lambda,\mu)$, called $\text{DPR}(\lambda,\mu)$, is then :

$$\text{DPR}(\lambda,\mu) \quad z(\text{DPR}(\lambda,\mu)) = Max \sum_{i \in L} \eta_i + \sum_{j \in B} \upsilon_j + M\delta + \sum_{i \in L} \lambda_i + \sum_{j \in B} \mu_j \qquad (5.36)$$

$$subject\ to \quad \eta_i + \upsilon_j + \delta \le b_{ij}, \qquad \forall (i,j) \in A_0 \qquad (5.37)$$

$$\left.\begin{array}{ll} \eta_i \le 0, & \forall i \in L \\ \upsilon_j \le 0, & \forall j \in B \\ \delta \text{ unrestricted} & \end{array}\right\} \qquad (5.38)$$

Note that we assume $\upsilon_0 = 0$ in the inequality (5.37).

In the following theorem we prove that a feasible solution of D can be obtained from any feasible solution of $\text{DPR}(\lambda,\mu)$ and the predetermined vectors λ and μ, showing that $z(\text{PR}(\lambda,\mu))$, for any λ and μ, is a valid lower bound to the VRPB.

Theorem 5.2. Let (η, υ, δ) be a feasible solution of $\text{DPR}(\lambda,\mu)$ of cost $z(\text{DPR}(\lambda,\mu))$ for a given pair of vectors λ and μ. A feasible solution of D of cost $z(\text{D}) = z(\text{DPR}(\lambda,\mu))$ is given by :

$$\left.\begin{array}{ll} u_i = \lambda_i + \eta_i, & \alpha_i = \varphi_i^L - \eta_i,\ i \in L \\ v_j = \mu_j + \upsilon_j, & \beta_j = \varphi_j^B - \upsilon_j,\ j \in B \\ w = \delta & \end{array}\right\} \qquad (5.39)$$

hence, $z(\text{DPR}(\lambda,\mu))$ is a valid lower bound for IP for any λ and μ.

Proof. Firstly it is easy to see that the values of the dual variables of D computed according to equation (5.39) satisfy constraints (5.14).

Consider now a path $\ell \in \mathcal{L}_i^E$. Substituting into the left-hand-side of inequalities (5.11) the values of $\mathbf{u}$ and α_i given by equations (5.39) yields :

$$\sum_{k\in P_\ell} u_k + \alpha_i = \sum_{k\in P_\ell} \lambda_k + \sum_{k\in P_\ell} \eta_k + \varphi_i^L - \eta_i . \tag{5.40}$$

Note that $\sum_{k\in P_\ell} \eta_k - \eta_i = \sum_{k\in P_\ell \setminus \{i\}} \eta_k$ and, since $\eta_k \le 0$, $k \in L$, we have $\sum_{k\in P_\ell \setminus \{i\}} \eta_k \le 0$.

Therefore, from equation (5.40), we obtain :

$$\sum_{k\in P_\ell} u_k + \alpha_i \le \sum_{k\in P_\ell} \lambda_k + \varphi_i^L . \tag{5.41}$$

From inequalities (5.29) and (5.41) we have

$$\sum_{k\in P_\ell} u_k + \alpha_i \le \sum_{k\in P_\ell} \lambda_k + c_\ell - \sum_{k\in P_\ell} \lambda_k (= c_\ell) . \tag{5.42}$$

Inequality (5.42) shows that the values of $\mathbf{u}$ and α given by equation (5.39) satisfy constraints (5.11). In a similar way it is easy to show that the values of $\mathbf{v}$ and β given by equations (5.39) satisfy inequalities (5.12). Finally, let us prove that the values of α , β and w given by equation (5.39) satisfy inequalities (5.13). In fact, for each arc $(i,j) \in A_0$ we have :

$$-\alpha_i - \beta_j + w = -\varphi_i^L + \eta_i - \varphi_j^B + \upsilon_j + \delta . \tag{5.43}$$

From inequalities (5.37) and the definition of b_{ij} we obtain :

$$\eta_i + \upsilon_j + \delta \le \varphi_i^L + \varphi_j^B + d_{ij} \tag{5.44}$$

or

$$-\varphi_i^L + \eta_i - \varphi_j^B + \upsilon_j + \delta \le d_{ij} . \tag{5.45}$$

From equations (5.43) and inequalities (5.45) we obtain inequalities (5.13)■

5.4.1.1 Improving value $z(\mathrm{PR}(\lambda,\mu))$

Algorithm H^1 is an iterative procedure that finds a feasible solution of problem D^1 by finding a feasible solution of the following problem:

$$\underset{\lambda,\mu}{Max}[z(\mathrm{PR}(\lambda,\mu))] . \tag{5.46}$$

An iteration of H^1 consists of computing new vectors λ and μ and of finding a new solution of the resulting problem $\mathrm{PR}(\lambda,\mu)$. The method used for changing λ and μ, at each iteration, is as follows.

Let ξ^* be an optimal $\mathrm{PR}(\lambda,\mu)$ solution for given λ and μ and $H^* = \{(i,j) : (i,j) \in A_0 , \xi^*_{ij} = 1\}$. Denote with L^* (resp. B^*) the set of Linehaul (resp. Backhaul) terminal vertices of the arcs of H^* (i.e. $L^* = \{i : (i,j) \in H^*\}$ and $B^* = \{j : (i,j) \in H^*\}$). We indicate with P_i^L, $i \in L$, and P_j^B, $j \in B$, the paths of cost φ_i^L and φ_j^B, respectively. Let h_i be the number of times that vertex $i \in L \cup B$ appears in the paths P_k^L , $k \in L^*$ and P_k^B , $k \in B^*$. It is obvious that in any feasible VRPB solution we have $h_i = 1$, $i \in L \cup B$, hence, a subgradient optimization method can be used to change λ and μ as follows :

$$\lambda_i = \lambda_i - \varepsilon \frac{z(\mathrm{UB}) - z(\mathrm{PR}(\lambda,\mu))}{\sum_{j \in L}(h_j - 1)^2 + \sum_{j \in B}(h_j - 1)^2}(h_i - 1), \quad i \in L \cup B. \tag{5.47}$$

The solution of D^1 is given by equation (5.39) using the values of λ and μ that produces the best approximate solution of problem (5.46) and the values of η, υ and δ of an optimal solution of the corresponding problem $\mathrm{DPR}(\lambda,\mu)$.

5.4.1.2 THE COMPUTATION OF φ_i^L , $i \in L$ AND φ_j^B , $j \in B$

We describe a method for computing φ_i^L , $i \in L$, an equivalent method can be used for computing φ_j^B , $j \in B$. A path (not necessarily elementary) $\Phi = (0, i_1, i_2, \ldots, i_k)$ in graph G_L and such that $\sum_{i \in \Phi} q_i = q$ is called a **q-path** (see Christofides et al. (1981a)). Let $f_i(q)$ be the cost of the least cost q-path in G_L from depot 0 to vertex $i \in L$, using arc costs $\{\bar{d}_{ij}\}$ defined by expression (5.28). Christofides et al. (1981a) describe a dynamic programming algorithm of complexity $O(Qn^2)$ for computing value $f_i(q)$, for each $i \in L$, and $q = q_i, q_i + 1, \ldots, Q$ with the restriction that the q-path corresponding to $f_i(q)$ should not contain **loops** formed by three consecutive vertices. Using function $f_i(q)$, the lower bound φ_i^L, $i \in L$, can be computed as

$$\varphi_i^L = \underset{q_i \le q \le Q}{Min}\left[f_i(q)\right]. \tag{5.48}$$

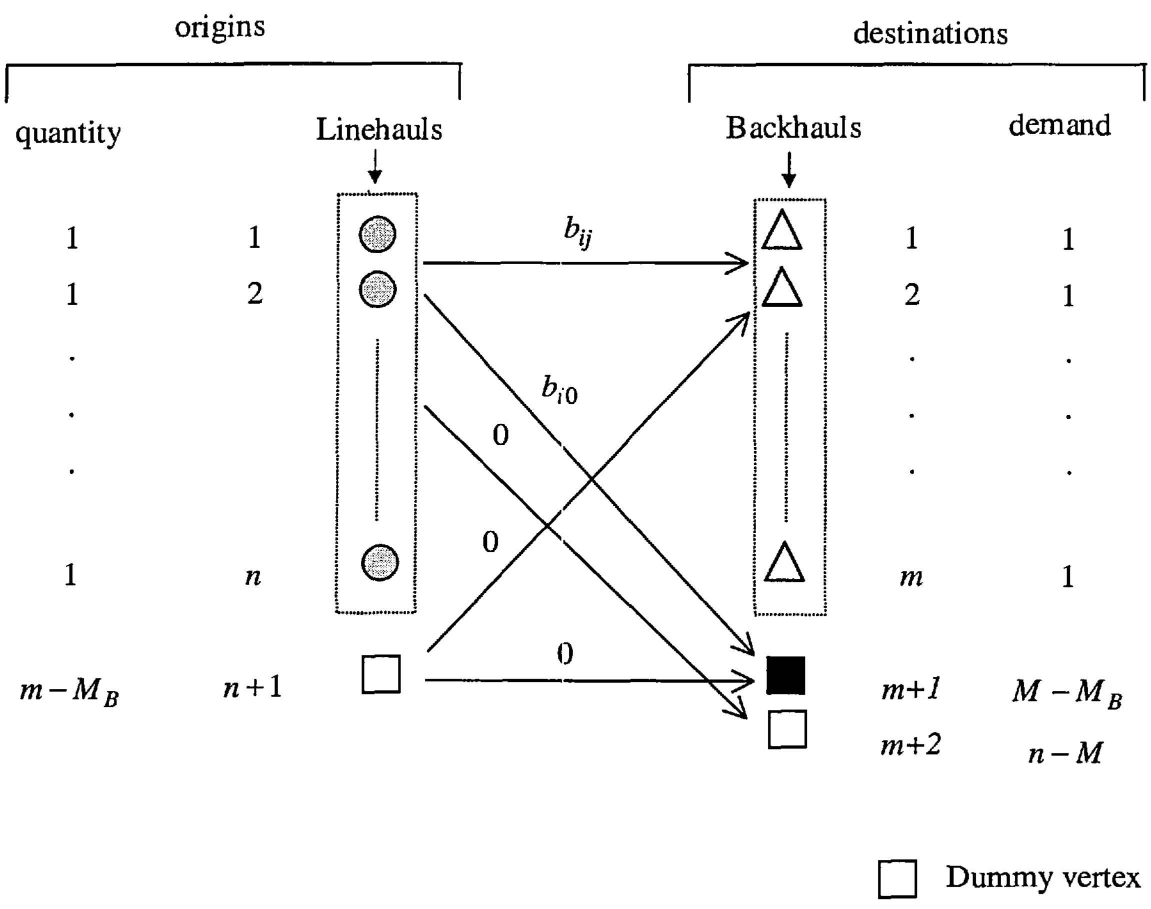


Figure 5.4. Transportation problem TP(λ,μ)

5.4.1.3 Solving problems PR(λ,μ) and DPR(λ,μ)

Problem PR(λ,μ) can be transformed into a balanced transportation problem TP(λ,μ) with $(n+1)$ origins and $(m+2)$ destinations as follows (see Figure 5.4).

Origins $\{1,\ldots,n\}$ correspond to the Linehaul customer set L while origin $n+1$ is a dummy one. Destinations $\{1,\ldots,m\}$ correspond to the Backhaul customer set B while destination $m+1$ represent the depot and $m+2$ is a dummy vertex.

A cost matrix γ_{ij} , $i=1,\ldots,n+1$, $j=1,\ldots,m+2$ is defined as follows :

$$\left.\begin{array}{ll} \gamma_{ij}=b_{ij} & i=1,\ldots,n,\ j=1,\ldots,m \\ \gamma_{ij}=b_{i0} & i=1,\ldots,n,\ j=m+1 \\ \gamma_{ij}=0 & i=n+1,\ j=1,\ldots,m+1 \\ \gamma_{ij}=0 & i=1,\ldots,n,\ j=m+2 \\ \gamma_{ij}=\infty & i=n+1,\ j=m+2 \end{array}\right\} \quad (5.49)$$

Note that the set of arcs $\{(i,j),\ i=1,\ldots,n,\ j=1,\ldots,m+1\}$ corresponds to the arcs set A_0.

We assume that a quantity equal to 1 is available at each origin i, $i=1,\ldots,n$, while a quantity equal to $m-M_B$ is available at origin n+1. The demand of each destination j, $j=1,\ldots,m$ is equal to 1, while the demands of destinations m+1 and m+2 are equal to $M-M_B$ and $n-M$, respectively.

Let χ_{ij} denote the quantity transported from origin i to destination j.

Problem TP(λ,μ) is as follows.

$$\text{TP}(\lambda,\mu)\quad z(\text{TP}(\lambda,\mu))=Min \sum_{i=1}^{n+1}\sum_{j=1}^{m+2}\gamma_{ij}\chi_{ij} \quad (5.50)$$

$$subject\ to \quad \sum_{j=1}^{m+2}\chi_{ij}=1, \qquad i=1,\ldots,n \quad (5.51)$$

$$\sum_{j=1}^{m+1}\chi_{n+1j}=m-M_B \quad (5.52)$$

$$\sum_{i=1}^{n+1}\chi_{ij}=1, \qquad j=1,\ldots,m \quad (5.53)$$

$$\sum_{i=1}^{n+1}\chi_{im+1}=M-M_B \quad (5.54)$$

$$\sum_{i=1}^{n}\chi_{im+2}=n-M \quad (5.55)$$

$$\chi_{ij} \geq 0, \qquad i=1,\ldots,n+1, j=1,\ldots,m+2 \tag{5.56}$$

Note that constraints (5.51) and (5.53) impose $\chi_{ij} \leq 1$, $i=1,\ldots,n, j=1,\ldots,m+2$ and $\chi_{ij} \leq 1$, $i=n+1, j=1,\ldots,m$. Constraint (5.55) impose that $n-M$ units are transported from the origins $\{1,\ldots,n\}$ to the dummy destination m+2. Due to constraints (5.51) we have that M units must be transported from the M distinct origins $\{1,\ldots,n\}$ to the destinations $\{1,\ldots,m+1\}$ and, due to constraint (5.54), at least M_B units are forced to go from the origins $\{1,\ldots,n\}$ to the destinations $\{1,\ldots,m\}$. Hence, any feasible TP(λ,μ) solution contains M arcs of set A_0 and at least M_B arcs of set $\{(i,j):(i,j)\in A \text{ s.t. } i\in L, j\in B\}$.

Theorem 5.3. Problems PR(λ,μ) and TP(λ,μ) are equivalent and

$$z(\mathrm{PR}(\lambda,\mu)) = z(\mathrm{TP}(\lambda,\mu)) + \sum_{i\in L}\lambda_i + \sum_{j\in B}\mu_j \tag{5.57}$$

Proof. We first prove that any solution χ of TP(λ,μ) can be transformed into a solution ξ of PR(λ,μ). Set $\xi_{ij} = \chi_{ij}, i=1,\ldots,n, j=1,\ldots,m$ and $\xi_{i0} = \chi_{im+1}$, $i=1,\ldots,n$. Note that in any optimal solution we have $\chi_{n+1\,m+2} = 0$.

From expressions (5.49) we have $\sum_{(i,j)\in A_0} b_{ij}\xi_{ij} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+2}\gamma_{ij}\chi_{ij}$, hence, equation (5.57).

It is easy to see that ξ satisfy inequalities (5.32) and (5.33). By adding equations (5.51) we obtain :

$$\sum_{i=1}^{n}\sum_{j=1}^{m+1}\chi_{ij} + \sum_{i=1}^{n}\chi_{im+2} = n. \tag{5.58}$$

From equations (5.55) and (5.58) we obtain

$$\sum_{i=1}^{n}\sum_{j=1}^{m+1}\chi_{ij} + n - M = n. \tag{5.59}$$

From equation (5.59) and the definition of ξ we have equation (5.34).

Any solution ξ of PR(λ,μ) can be transformed into a solution χ of problem TP(λ,μ) by setting :

$$\left.\begin{array}{ll}\chi_{ij}=\xi_{ij}\,, & i=1,\ldots,n\,,\ j=1,\ldots,m\\ \chi_{im+1}=\xi_{i0}\,, & i=1,\ldots,n\\ \chi_{n+1j}=1-\sum_{i=1}^{n}\xi_{ij}\,, & j=1,\ldots,m\\ \chi_{n+1m+1}=M-M_B-\sum_{i=1}^{n}\xi_{i0} & \\ \chi_{im+2}=1-\sum_{j=0}^{m}\xi_{ij}\,, & i=1,\ldots,n\end{array}\right\} \quad (5.60)$$

It is easy to show that this solution χ satisfies all the constraints of problem $\mathrm{TP}(\lambda,\mu)$ ■

Problem $\mathrm{TP}(\lambda,\mu)$ can be solved by the Hungarian method that also provides an optimal solution of the dual of $\mathrm{TP}(\lambda,\mu)$. We can show that an optimal dual solution of $\mathrm{TP}(\lambda,\mu)$ can be transformed into an optimal solution of $\mathrm{DPR}(\lambda,\mu)$.

The dual of problem $\mathrm{TP}(\lambda,\mu)$, called $\mathrm{DTP}(\lambda,\mu)$, is as follows :

$$\mathrm{DTP}(\lambda,\mu)\ z(\mathrm{DTP}(\lambda,\mu))=Max\ \sum_{i=1}^{n}\rho_i+(m-M_B)\rho_{n+1}+\sum_{j=1}^{m}\sigma_j+ (M-M_B)\sigma_{m+1}+(n-M)\sigma_{m+2} \quad (5.61)$$

$$\textit{subject to}\quad \rho_i+\sigma_j\le\gamma_{ij}\,,\quad \begin{array}{l} i=1,\ldots,n+1\\ j=1,\ldots,m+2\end{array} \quad (5.62)$$

$$\left.\begin{array}{ll}\rho_i \text{ unrestricted}, & i=1,\ldots,n+1\\ \sigma_j \text{ unrestricted}, & j=1,\ldots,m+2\end{array}\right\} \quad (5.63)$$

Let (ρ^*,σ^*) be an optimal $\mathrm{DTP}(\lambda,\mu)$ solution and let us define :

$$\rho^{\max}=\underset{1\le i\le n}{Max}\left[\rho_i^*\right] \text{ and } \sigma^{\max}=\underset{1\le j\le m+1}{Max}\left[\sigma_j^*\right] \quad (5.64)$$

We denote by $i_{\max}$ the index such that $\rho_{i_{\max}}=\rho^{\max}$.

We assume that $\sigma_{m+1}^*=\sigma^{\max}$ (this is possible since the constraints of problem $\mathrm{TP}(\lambda,\mu)$ are not independent).

Theorem 5.4. In any optimal $\mathrm{DTP}(\lambda,\mu)$ solution (ρ^*,σ^*) such that $\sigma_{m+1}^*=\sigma^{\max}$ we have :

$$\left.\begin{aligned} \rho^*_{n+1} &= -\sigma^{\max} \\ \sigma^*_{m+2} &= -\rho^{\max} \end{aligned}\right\} \qquad (5.65)$$

Proof. By definition $1 \le i_{max} \le n$ (see expressions (5.64)), therefore, from the dual constraint (5.67) and expression (5.49), we have :

$$\left.\begin{aligned} &\rho^*_{n+1} + \sigma^{\max} \le 0 \qquad && i = n+1\,,\, j = m+1 \\ &\rho^{\max} + \sigma^*_{m+2} \le 0 \qquad && i = i_{\max}\,,\, j = m+2 \end{aligned}\right\} \qquad (5.66)$$

hence, $\rho^*_{n+1} \le -\sigma^{\max}$ and $\sigma^*_{m+2} \le -\rho^{\max}$. Since both variables ρ_{n+1} and σ_{m+2} have a positive coefficient in the objective function (5.61) we have $\rho^*_{n+1} = -\sigma^{\max}$ and $\sigma^*_{m+2} = -\rho^{\max}$ ■

Theorem 5.5. Any optimal DTP(λ,μ) solution (ρ^*,σ^*) of cost $z(\mathrm{DTP}(\lambda,\mu))$ can be transformed into an optimal DPR(λ,μ) solution $(\eta^*,\upsilon^*,\delta^*)$ of cost

$$z(\mathrm{DPR}(\lambda,\mu)) = z(\mathrm{DTP}(\lambda,\mu)) + \sum_{i\in L}\lambda_i + \sum_{j\in B}\mu_j$$

by setting :

$$\left.\begin{aligned} &\eta^*_i = \rho^*_i - \rho^{\max}\,, \qquad && i \in L \\ &\upsilon^*_j = \sigma^*_j - \sigma^{\max}\,, \qquad && j \in B \\ &\delta^* = \rho^{\max} + \sigma^{\max} && \end{aligned}\right\} \qquad (5.67)$$

Proof. By substituting η^*, υ^* and δ^* into the dual constraints (5.37) we obtain :

$$\rho^*_i - \rho^{\max} + \sigma^*_j - \sigma^{\max} + \rho^{\max} + \sigma^{\max} \le b_{ij}\,,\ (i,j) \in A_0$$

or $\rho^*_i + \sigma^*_j \le b_{ij}$, $(i,j) \in A_0$. These correspond to the dual constraints of DTP(λ,μ) that, by definition, are satisfied by ρ^* and σ^*. It is obvious that the values of η^*, υ^* and δ^* satisfy constraints (5.38).

The cost $z(\mathrm{DPR}(\lambda,\mu))$ of a DPR(λ,μ) solution $(\eta^*,\upsilon^*,\delta^*)$ can be written as follows:

$$z(\mathrm{DPR}(\lambda,\mu)) = z_A + \sum_{i\in L}\lambda_i + \sum_{j\in B}\mu_j$$

where $z_A = \sum_{i \in L} \eta_i^* + \sum_{j \in B} \upsilon_j^* + M\delta^*$. From expression (5.67) we obtain :

$$z_A = \sum_{i \in L} \rho_i^* + \sum_{j \in B} \sigma_j^* - (n-M)\rho^{\max} - (m-M)\sigma^{\max}. \tag{5.68}$$

From Theorem 5.4 we have $\rho_{n+1}^* = -\sigma^{\max}$, $\sigma_{m+2}^* = -\rho^{\max}$ and, therefore :

$$z_A = \sum_{i \in L} \rho_i^* + \sum_{j \in B} \sigma_j^* + (n-M)\sigma_{m+2}^* + (m-M)\rho_{n+1}^*. \tag{5.69}$$

As $m - M = (m - M_B) - (M - M_B)$, equation (5.69) becomes

$$z_A = \sum_{i \in L} \rho_i^* + \sum_{j \in B} \sigma_j^* + (n-M)\sigma_{m+2}^* + (m - M_B)\rho_{n+1}^* - (M - M_B)\rho_{n+1}^*. \tag{5.70}$$

From Theorem 5.4, we have $\sigma_{m+1}^* = -\rho_{n+1}^*$, hence from equation (5.70) we deduce that $z_A = z(\text{DTP}(\lambda,\mu))$ ∎

5.4.2 Procedure H^2

Let $(\mathbf{u}^1, \mathbf{v}^1, \alpha^1, \beta^1, w^1)$ be a feasible solution of D^1 of cost $z(\text{D}^1)$ produced by procedure H^1. The reduced costs of the variables of problem IP are given by :

$$\left.\begin{array}{ll} c_\ell^2 = c_\ell - \sum_{k \in P_\ell} u_k^1 - \alpha_{t_\ell}^1, & \ell \in \mathcal{L} \\ c_\ell^2 = c_\ell - \sum_{k \in P_\ell} v_k^1 - \beta_{t_\ell}^1, & \ell \in \mathcal{B} \\ d_{ij}^2 = d_{ij} + \alpha^1{}_i + \beta^1{}_j - w^1, & (i,j) \in A_0 \end{array}\right\} \tag{5.71}$$

We denote by D^2 the problem obtained from D by replacing $\{c_\ell\}$ with $\{c_\ell^2\}$ and $\{d_{ij}\}$ with $\{d_{ij}^2\}$.

Problem D^2 cannot be solved directly as the number of constraints may be too large. In this section we describe a heuristic procedure, called H^2, for reducing the number of constraints of D^2 so that the resulting problem, called $\overline{\text{D}}^2$, can be solved directly and any solution of $\overline{\text{D}}^2$ is a feasible D^2 solution. Problem $\overline{\text{D}}^2$ is obtained from D^2 as follows:

(i) reduce the number of constraints (5.11) and (5.12) by replacing $\mathcal{L}$ and $\mathcal{B}$ with subsets $\overline{\mathcal{L}} \subseteq \mathcal{L}$ and $\overline{\mathcal{B}} \subseteq \mathcal{B}$ of limited size;

(ii) add constraints to force any $\overline{\mathrm{D}}^2$ solution to satisfy constraints (5.11) for any $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$ and constraints (5.12) for any $\ell \in \mathcal{B} \setminus \overline{\mathcal{B}}$.

Reduced problem $\overline{\mathrm{D}}^2$

Let $\overline{\mathcal{L}} \subseteq \mathcal{L}$ and $\overline{\mathcal{B}} \subseteq \mathcal{B}$ be the subsets of paths satisfying the following conditions :

$$\left.\begin{array}{ll}\underset{\ell \in \overline{\mathcal{L}}}{Max}\left[c_\ell^2\right] \le \underset{\ell \in \mathcal{L}\setminus\overline{\mathcal{L}}}{Min}\left[c_\ell^2\right] & \text{(a)} \\ \underset{\ell \in \overline{\mathcal{B}}}{Max}\left[c_\ell^2\right] \le \underset{\ell \in \mathcal{B}\setminus\overline{\mathcal{B}}}{Min}\left[c_\ell^2\right] & \text{(b)} \\ c_\ell^2 < z(\mathrm{UB}) - z(\mathrm{D}^1), \ell \in \overline{\mathcal{L}} \cup \overline{\mathcal{B}} & \text{(c)}\end{array}\right\} \quad (5.72)$$

We set $\overline{\mathcal{L}}_i^E = \overline{\mathcal{L}} \cap \mathcal{L}_i^E$, $i \in L$ and $\overline{\mathcal{B}}_j^S = \overline{\mathcal{B}} \cap \mathcal{B}_j^S$, $j \in B$.

For generating the two sets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ we used a procedure similar to the one used by Mingozzi et al. (1995) for the VRP which will be summarised in section 5.4.2.2. Note that real-world VRPB constraints can, at this stage, be easily considered by removing from $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ any infeasible path. The reduced problem $\overline{\mathrm{D}}^2$ is as follows:

$(\overline{\mathrm{D}}^2)$ $$z(\overline{\mathrm{D}}^2) = Max \sum_{i \in L} u_i + \sum_{j \in B} v_j + Mw \qquad (5.73)$$

subject to
$$\sum_{k \in P_\ell} u_k + \alpha_i \le c_\ell^2 , \qquad \forall \ell \in \overline{\mathcal{L}}_i^E , \forall i \in L \qquad (5.74)$$

$$\sum_{k \in P_\ell} v_k + \beta_j \le c_\ell^2 , \qquad \forall \ell \in \overline{\mathcal{B}}_j^S , \forall j \in B \qquad (5.75)$$

$$-\alpha_i - \beta_j + w \le d_{ij}^2 , \qquad \forall (i,j) \in A_0 \qquad (5.76)$$

$$u_i + \delta_i \le U_i , \qquad \forall i \in L \qquad (5.77)$$

$$\alpha_i - \delta_i \le 0 , \qquad \forall i \in L \qquad (5.78)$$

$$v_j + \theta_j \le V_j , \qquad \forall j \in B \qquad (5.79)$$

$$\beta_j - \theta_j \le 0 , \qquad \forall j \in B \qquad (5.80)$$

$$\left.\begin{array}{ll} u_i\ ,\ \alpha_i \text{ unrestricted}\ ,\ \delta_i \geq 0 & \forall i \in L \\ v_j\ ,\ \beta_j \text{ unrestricted}\ ,\ \theta_j \geq 0 & \forall j \in B \\ w \qquad \text{unrestricted} & \end{array}\right\} \quad (5.81)$$

Constraints (5.77), (5.78), (5.79) and (5.80) ensure that any solution of $\overline{\mathrm{D}}^2$ is a feasible D^2 solution if the upper bounds U_i , $i \in L$ and V_j , $j \in B$ are chosen such that :

$$\left.\begin{array}{ll} \sum_{i \in P_\ell} U_i \leq c_\ell^2\ ,\ \ell \in \mathcal{L} \setminus \overline{\mathcal{L}} & \text{(a)} \\ \sum_{j \in P_\ell} V_j \leq c_\ell^2\ ,\ \ell \in \mathcal{B} \setminus \overline{\mathcal{B}} & \text{(b)} \end{array}\right\} \quad (5.82)$$

Theorem 5.6. Any feasible solution to $\overline{\mathrm{D}}^2$ is also a feasible solution of D^2 with the same objective function value.

Proof. Let us consider the dual constraint (5.11) of path $\ell \in \mathcal{L}_i^E \setminus \overline{\mathcal{L}}_i^E$ for a given $i \in L$.

From inequalities (5.77) and (5.78) we have :

$$\sum_{k \in P_\ell} u_k + \alpha_i \leq \sum_{k \in P_\ell} U_k - \sum_{k \in P_\ell} \delta_k + \delta_i$$

and from inequalities (5.82.a), since $\delta_i \geq 0$, $\forall i \in L$, we have :

$$\sum_{k \in P_\ell} u_k + \alpha_i \leq c_\ell^2 - \sum_{k \in P_\ell} \delta_k + \delta_i \leq c_\ell^2 .$$

Hence, any solution of $\overline{\mathrm{D}}^2$ satisfies the dual constraint (5.11) for any $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$. In a similar way we can show that a feasible $\overline{\mathrm{D}}^2$ solution satisfies constraints (5.12) for any $\ell \in \mathcal{B} \setminus \overline{\mathcal{B}}$ ■

5.4.2.1 THE COMPUTATION OF U_i , $i \in L$ AND V_j , $j \in B$

In computing U_i , $\forall i \in L$, we must consider two cases :

A) $c_\ell^2 \geq z(\mathrm{UB}) - z(\mathrm{D}^1)$, $\forall \ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$. From Corollary 5.1 no path $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$ can belong to an optimal VRPB solution, hence, we can set $U_i = \infty$, $\forall i \in L$.

B) $c_\ell^2 < z(\mathrm{UB}) - z\left(\mathrm{D}^1\right)$, for some $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$. In this case every optimal VRPB solution might contain some path $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$. We can set :

$$U_i = q_i \hat{c}^L / Q, \quad \forall i \in L \tag{5.83}$$

where

$$\hat{c}^L = \underset{\ell \in \overline{\mathcal{L}}}{Max}\left[c_\ell^2\right]. \tag{5.84}$$

It is easy to show that the $\{U_i\}$ defined according to expression (5.83) and (5.84) satisfy inequalities (5.82-a). For any $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$ we have :

$$\sum_{i \in P_\ell} U_i = \sum_{i \in P_\ell} q_i \hat{c}^L / Q. \tag{5.85}$$

Since from (5.84) we know that $\hat{c}^L \le c_\ell^2$, $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$, equation (5.85) then becomes :

$$\sum_{i \in P_\ell} U_i \le c_\ell^2 \sum_{i \in P_\ell} q_i / Q \le c_\ell^2.$$

Also for the computation of V_j , $\forall j \in B$, we must consider two cases.

C) $c_\ell^2 \ge z(\mathrm{UB}) - z\left(\mathrm{D}^1\right)$, $\forall \ell \in \mathcal{B} \setminus \overline{\mathcal{B}}$. This case is analogous to case (A) above. We can set $V_j = \infty$, $\forall j \in B$.

D) $c_\ell^2 < z(\mathrm{UB}) - z\left(\mathrm{D}^1\right)$, for some $\ell \in \mathcal{B} \setminus \overline{\mathcal{B}}$. We can set :

$$V_j = q_j \hat{c}^B / Q, \quad \forall j \in B \tag{5.86}$$

where

$$\hat{c}^B = \underset{\ell \in \overline{\mathcal{B}}}{Max}\left[c_\ell^2\right]. \tag{5.87}$$

The proof that the $\{V_j\}$ computed according to expressions (5.86) and (5.87) satisfy inequalities (5.82-b) is similar to the one given in case (B) above. Problem $\overline{\mathrm{D}}^2$ can be considered to be the dual of the following problem $\overline{\mathrm{IP}}^2$.

$$(\overline{\mathrm{IP}}^2) \quad z\left(\overline{\mathrm{IP}}^2\right) = Min \quad \sum_{\ell \in \overline{\mathcal{L}}} c_\ell^2 x_\ell + \sum_{\ell \in \overline{\mathcal{B}}} c_\ell^2 y_\ell + \sum_{(i,j) \in A_0} d_{ij}^2 \xi_{ij} + \sum_{i \in L} U_i x_i^\alpha + \sum_{j \in B} V_j y_j^\beta \tag{5.88}$$

$$\textit{subject to} \quad \sum_{\ell \in \bar{L}_i} x_\ell + x_i^\alpha = 1, \qquad \forall i \in L \tag{5.89}$$

$$\sum_{\ell \in \bar{\mathcal{B}}_j} y_\ell + y_j^\beta = 1, \qquad \forall j \in B \tag{5.90}$$

$$\sum_{\ell \in \bar{L}_i^E} x_\ell - \sum_{j \in B_0} \xi_{ij} + \xi_i^\alpha = 0, \qquad \forall i \in L \tag{5.91}$$

$$\sum_{\ell \in \bar{\mathcal{B}}_j^S} y_\ell - \sum_{i \in L} \xi_{ij} + \xi_j^\beta = 0, \qquad \forall j \in B \tag{5.92}$$

$$\sum_{(i,j) \in A_0} \xi_{ij} = M \tag{5.93}$$

$$x_i^\alpha - \xi_i^\alpha \geq 0, \qquad \forall i \in L \tag{5.94}$$

$$y_j^\beta - \xi_j^\beta \geq 0, \qquad \forall j \in B \tag{5.95}$$

$$\mathbf{x}, \mathbf{y}, \mathbf{x}^\alpha, \mathbf{y}^\beta, \xi, \xi^\alpha, \xi^\beta \geq 0 \tag{5.96}$$

Procedure H^2 consists of finding an optimal solution $\left(\mathbf{x}^*, \mathbf{y}^*, \mathbf{x}^{\alpha*}, \mathbf{y}^{\beta*}, \xi^*, \xi^{\alpha*}, \xi^{\beta*}\right)$ of $\overline{\mathrm{IP}}^2$ of cost $z\left(\overline{\mathrm{IP}}^2\right)$ and the corresponding optimal dual variables $\left(\mathbf{u}^*, \mathbf{v}^*, \alpha^*, \beta^*, w^*\right)$. Hence, we have $z\left(\mathrm{D}^2\right) = z\left(\overline{\mathrm{D}}^2\right)$ and $\mathbf{u}^2 = \mathbf{u}^*$, $\mathbf{v}^2 = \mathbf{v}^*$, $\alpha^2 = \alpha^*$, $\beta^2 = \beta^*$, $w^2 = w^*$.

Procedure HDS finds a solution $\left(\mathbf{u}', \mathbf{v}', \alpha', \beta', w'\right)$ of D of cost $z'(\mathrm{D}) = z\left(\mathrm{D}^1\right) + z\left(\mathrm{D}^2\right)$ by setting $\mathbf{u}' = \mathbf{u}^1 + \mathbf{u}^2$, $\mathbf{v}' = \mathbf{v}^1 + \mathbf{v}^2$, $\alpha' = \alpha^1 + \alpha^2$, $\beta' = \beta^1 + \beta^2$, $w' = w^1 + w^2$.

An optimal VPRB solution

We observe that an optimal $\overline{\mathrm{IP}}^2$ solution, under certain conditions, corresponds to an optimal VRPB solution. In fact the following cases may arise:

(A) $\mathbf{x}^*, \mathbf{y}^*, \xi^*$ integer and $\mathbf{x}^{\alpha*} = \xi^{\alpha*} = \mathbf{0}$, $\mathbf{y}^{\beta*} = \xi^{\beta*} = \mathbf{0}$.

This solution is an optimal VRPB solution of cost $z(\mathrm{IP}) = z'(\mathrm{D})$.

(B) $\mathbf{x}^*$ or $\mathbf{y}^*$ or ξ^* not integer and $U_i = \infty$, $i \in L$, and $V_j = \infty$, $j \in B$.

In this case all paths of any optimal VRPB solution are contained in the two sets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ and an optimal VRPB solution can be obtained by solving problem IP after having replaced sets $\mathcal{L}$ and $\mathcal{B}$ with $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$.

(C) $\mathbf{x}^{\alpha *} \neq \mathbf{0}$ or $\mathbf{y}^{\beta *} \neq \mathbf{0}$.

This $\overline{\mathrm{IP}}^2$ solution is not feasible for the VRPB. Furthermore, sets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ might not contain any feasible and/or optimal VRPB solution.

5.4.2.2 GENERATING SETS $\overline{\mathcal{L}}$ AND $\overline{\mathcal{B}}$

In this section we describe a method for generating sets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ satisfying conditions (5.72) with the additional restrictions that $|\overline{\mathcal{L}}| \leq Maxsize$ and $|\overline{\mathcal{B}}| \leq Maxsize$, where *Maxsize* is a predefined positive integer to ensure that no memory overflow occurs. In the following we give the description of an algorithm for generating set $\overline{\mathcal{L}}$, being obvious that an equivalent procedure can be used for computing $\overline{\mathcal{B}}$.

The method for generating set $\overline{\mathcal{L}}$ is based on the following observation.

Let P be a feasible path in G_L from vertex 0 to some vertex $t(P)$. We denote with $LB(P)$ a lower bound to the reduced cost of any feasible path in G_L that can be obtained by expanding P from $t(P)$ to some other vertex $i \in L_0$.

Theorem 5.7. Let $\mathcal{P}$ a subset of the path set $\mathcal{L}$ such that :

$$\underset{\ell \in \mathcal{P}}{Max}\left[LB(P_\ell)\right] \leq \underset{\ell \in \mathcal{L} \setminus \mathcal{P}}{Min}\left[LB(P_\ell)\right]. \tag{5.97}$$

The subset $\overline{\mathcal{L}}$ of $\mathcal{P}$ defined as :

$$\overline{\mathcal{L}} = \left\{\ell : \ell \in \mathcal{P} \text{ s.t. } c_\ell^2 \leq \underset{\ell \in \mathcal{L} \setminus \mathcal{P}}{Min}\left[LB(P_\ell)\right]\right\} \tag{5.98}$$

satisfies inequality (5.72-a).

Proof. From the definitions of $\overline{\mathcal{L}}$ and of $LB(P)$, we have :

$$\underset{\ell \in \overline{\mathcal{L}}}{Max}\left[c_\ell^2\right] \leq \underset{\ell \in \mathcal{L} \setminus \mathcal{P}}{Min}\left[LB(P_\ell)\right] \leq \underset{\ell \in \mathcal{L} \setminus \mathcal{P}}{Min}\left[c_\ell^2\right]. \tag{5.99}$$

Moreover, from the definition of $\overline{\mathcal{L}}$ we have :

$$\underset{\ell \in \overline{\mathcal{L}}}{Max}\left[c_\ell^2\right] \leq \underset{\ell \in \mathcal{P} \setminus \overline{\mathcal{L}}}{Min}\left[c_\ell^2\right]. \tag{5.100}$$

Hence, from inequalities (5.99) and (5.100) we obtain :

$$\underset{\ell \in \overline{\mathcal{L}}}{Max}\left[c_\ell^2\right] \leq \underset{\ell \in (\mathcal{L} \setminus \mathcal{P}) \cup (\mathcal{P} \setminus \overline{\mathcal{L}})}{Min}\left[c_\ell^2\right]. \tag{5.101}$$

Inequality (5.101) corresponds to inequality (5.72-a) since $(\mathcal{L} \setminus \mathcal{P}) \cup (\mathcal{P} \setminus \overline{\mathcal{L}}) = \mathcal{L} \setminus \overline{\mathcal{L}}$ ■

In the following we describe an algorithm, called GEN$\mathcal{P}$, for generating set $\mathcal{P}$ that is analogous to the Dijkstra's algorithm in an expanded state-space graph. Let $S(P)$ be the set of vertices visited by path P.

In algorithm GEN$\mathcal{P}$ we use $LB(P)$ as a label and set T to denote a temporary set of paths (i.e. it is not known if $P \in T$ is a least cost path starting at 0, visiting the subset of vertices $S(P)$ and ending at $t(P)$).

Algorithm 5.3: GEN$\mathcal{P}$

We use T_j and $\mathcal{P}_j$ to denote the subsets of all paths terminating at vertex j of sets T and $\mathcal{P}$, respectively.

Step 0. Set $T = \{(0)\}$, $LB((0)) = 0$ and $\mathcal{P} = \varnothing$.

Step 1. If $T = \varnothing$, then go to *Step 6.*

Step 2. Let $P^* \in T$ be such that $LB(P^*) = \underset{P \in T}{Min}[LB(P)]$ and denote with i^* the terminal vertex of P^*. Update $T = T \setminus \{P^*\}$ and $\mathcal{P} = \mathcal{P} \cup \{P^*\}$.

If $|\mathcal{P}| = Maxsize$, then go to *Step 6.*

Step 3. For any vertex $j \in \left\{ j : j \in L \setminus S(P^*) \text{ s.t. } \sum_{i \in P^*} q_i + q_j \leq Q \right\}$ repeat *Step 4.*

Step 4. Let P' be a path obtained by appending vertex j to the end of path P^*.

We have two cases :

1. $LB(P') \geq z(\text{UB}) - z(\text{D}^1)$, then P' is rejected.

2. $LB(P') < z(\mathrm{UB}) - z(\mathrm{D}^1)$, then we have three subcases :

   (i) $S(P') \neq S(P),\ \forall P \in \mathcal{P}_j \cup T_j$, then update $T = T \cup \{P'\}$.

   (ii) $S(P') = S(P)$, for some $P \in \mathcal{P}_j$, then P' is dominated by P (i.e. $LB(P) \leq LB(P')$, hence, P' is rejected).

   (iii) $S(P') = S(P)$, for some $P \in T_j$. If $LB(P') \geq LB(P)$ then P' is dominated by P and is not stored in T, otherwise update $T = T \setminus \{P\}$ and $T = T \cup \{P'\}$.

Step 5. Return to *Step 1.*

Step 6 Let $LBMAX = \underset{P \in \mathcal{P}}{Max}[LB(P)]$. Remove from $\mathcal{P}$ any path P such that $\sum_{i \in P} q_i \leq Q^L_{\min}$ and/or $c(P) - \sum_{i \in P} u^1_i - \alpha^1_{t(P)} > LBMAX$.

Set $\overline{\mathcal{L}}$ corresponds to the indices of the paths contained in $\mathcal{P}$ at the end of GEN$\mathcal{P}$.

Concerning the computation of U_i, $i \in L$, we note that if GEN$\mathcal{P}$ terminates with $T = \varnothing$, then $c^2_\ell \geq LB(P_\ell) > z(\mathrm{UB}) - z(\mathrm{D}^1)$, $\forall \ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$, hence, we set $U_i = \infty$, $\forall i \in L$. If GEN$\mathcal{P}$ terminates with $|\mathcal{P}| = Maxsize$, then we set $U_i = q_i\ LBMAX/Q$.

Reducing sets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$

Once the two sets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ have been generated, we can remove from $\overline{\mathcal{L}}$, according to Corollary 5.2, any path ℓ such that :

$$c^2_\ell + \underset{j \in B_0}{Min}\left[d^2_{t_\ell j} + \underset{r \in \overline{\mathcal{B}}^S_j}{Min}\left[c^2_r\right]\right] \geq z(\mathrm{UB}) - z(\mathrm{D}^1)$$

and, according to Corollary 5.3, we can remove from $\overline{\mathcal{B}}$ any path ℓ such that :

$$\underset{i \in L}{Min}\left[\underset{r \in \overline{\mathcal{L}}^E_i}{Min}\left[c^2_\ell\right] + d^2_{i t_\ell} \right] + c^2_\ell \geq z(\mathrm{UB}) - z(\mathrm{D}^1).$$

Computing lower bound $LB(P)$

In the following we use $Q(P)$ to denote the load of path P (i.e. $Q(P) = \sum_{i \in P} q_i$).

Associate with each arc (i, j) of graph G_L the following cost $\tilde{d}_{ij}$:

$$\tilde{d}_{ij} = \begin{cases} d_{ij} - u_j^1, \; i \in L_0 \;, j \in L \\ -\alpha_i^1 \;, \quad i \in L \,, j = 0 \end{cases}$$

Let $f_i^{-1}(q)$ be the cost of the least cost q-path from vertex $i \in L$ to depot 0 and let $\pi_i^{-1}(q)$ be the vertex just prior to i on the path corresponding to $f_i^{-1}(q)$.

Let $g_i^{-1}(q)$ be the cost of the least cost q-path from vertex $i \in L$ to depot 0 with $\rho_i^{-1}(q) \neq \pi_i^{-1}(q)$, where $\rho_i^{-1}(q)$ is the vertex just prior to i on the path corresponding to $g_i^{-1}(q)$.

Functions $f_i^{-1}(q)$ and $g_i^{-1}(q)$ can be computed by means of a dynamic programming algorithm in a similar way as functions $f_i(q)$ described in section 5.4.1.2.

Let P be a feasible path in G_L starting at depot 0 and ending at vertex $i = t(P)$. We denote with $\tilde{c}(P)$ the cost of P using arc costs $\{\tilde{d}_{ij}\}$.

A lower bound $LB(P)$ for the reduced cost $\tilde{c}(P')$ of any feasible path P' in G_L that can be obtained by expanding path P is given by :

$$LB(P) = \tilde{c}(P) + \underset{q_{\min}(P) \le q \le q_{\max}(P)}{Min} \begin{cases} f_i^{-1}(q) \;, \text{ if } \pi_i^{-1}(q) \notin P \\ g_i^{-1}(q) \;, \text{ if } \pi_i^{-1}(q) \in P \end{cases} \tag{5.102}$$

where $q_{\min}(P) = Max\left[Q_{\min}^L - \sum_{k \in P} q_k , q_i \right]$ and $q_{\max}(P) = Q - \sum_{k \in P} q_k + q_i$.

A computationally better method for calculating $LB(P)$, avoiding the minimization problem required by expression (5.102), involves the definition of the following functions $F_i(q)$, $\sigma_i(q)$ and $G_i(q)$ that are computed, for each $q \; (q_i \le q \le Q)$, as follows.

$$F_i(q) = \underset{Max[Q_{\min} - q, q_i] \le q' \le Q - q + q_i}{Min} \left[f_i^{-1}(q') \right]$$

$\sigma_i(q) = \pi_i^{-1}(q^*)$, where q^* is such that $F_i(q) = f_i^{-1}(q^*)$

$$G_i(q) = \underset{Max[Q_{\min}-q, q_i] \le q' \le Q-q+q_i}{Min} \begin{cases} f_i^{-1}(q'), & \text{if } \pi_i^{-1}(q) \neq \sigma_i(q) \\ g_i^{-1}(q'), & \text{if } \pi_i^{-1}(q) = \sigma_i(q) \end{cases}$$

The lower bound $LB(P)$ can be computed as :

$$LB(P) = \begin{cases} \tilde{c}(P) + F_i(Q(P)), & \text{if } \sigma_i(Q(P)) \notin P \\ \tilde{c}(P) + G_i(Q(P)), & \text{if } \sigma_i(Q(P)) \in P \end{cases} \tag{5.103}$$

5.5 AN EXACT METHOD FOR SOLVING THE VRPB

In this section we describe an exact method for the VRPB, called EHP, that consists of reducing the number of variables of the integer program IP so that the resulting problem can be solved by an integer programming solver (CPLEX (1993)). This method may terminate, under certain circumstances, without having found an optimal solution.

Let $(\mathbf{u}', \mathbf{v}', \alpha', \beta', w')$ be the solution of D of cost $z'(\mathrm{D})$ obtained by procedure HDS and let c'_ℓ , $\ell \in \mathcal{L} \cup \mathcal{B}$, and d'_{ij} , $(i, j) \in A_0$ be the reduced costs corresponding to this dual solution. We could attempt to solve IP as indicated by Corollary 5.1, that is, we might generate $\mathcal{L}'$, $\mathcal{B}'$ and A'_0 according to expressions (5.22) and then solve IP using $\mathcal{L}'$, $\mathcal{B}'$ and A'_0 instead of $\mathcal{L}$, $\mathcal{B}$ and A_0. However, the size of $\mathcal{L}'$ and/or $\mathcal{B}'$ may be too large, hence we propose generating $\mathcal{L}'$ and $\mathcal{B}'$ so that their size is limited and the resulting problem IP′ becomes solvable. By means of the procedure described in section 5.4.2.2, we generate $\mathcal{L}'$, $\mathcal{B}'$ satisfying conditions (5.72) where the reduced costs $\{c^2_\ell\}$ are replaced with $\{c'_\ell\}$ and $z(\mathrm{D}^1)$ is substituted with $z'(\mathrm{D})$. Note that the size of each set $\mathcal{L}'$ and $\mathcal{B}'$ is limited by the value of *Maxsize* used in algorithm GEN$\mathcal{P}$. Moreover, the sets $\mathcal{L}'$, $\mathcal{B}'$ and A'_0 can be further reduced by applying Corollaries 5.2, 5.3 and 5.4 of section 5.3.1.1.

Let $\mathbf{x}^*, \mathbf{y}^*, \xi^*$ be an optimal solution of IP′ of cost $z(\mathrm{IP}')$ (we assume $z(\mathrm{IP}') = \infty$ if the sets $\mathcal{L}'$ and $\mathcal{B}'$ do not contain any optimal VRPB solution).

If $z(\text{IP}') < \infty$ then solution $\mathbf{x}^*, \mathbf{y}^*, \xi^*$ is a feasible VRPB solution and it may be also an optimal one. Let $\Delta = Min\left\{ \underset{\ell \in \mathcal{L}'}{Max}\left[c'_\ell\right], \underset{\ell \in \mathcal{B}'}{Max}\left[c'_\ell\right] \right\}$.

We have the following cases :

1. $z(\text{IP}') \leq z'(\text{D}) + \Delta$. In this case the optimal solution of IP′ is also an optimal VRPB solution. This derives from Theorem 5.1 as any VRPB solution involving at least one path of $\mathcal{L} \setminus \mathcal{L}'$ or $\mathcal{B} \setminus \mathcal{B}'$ has a cost greater or equal to $z'(\text{D}) + \Delta$.
2. $z(\text{IP}') > z'(\text{D}) + \Delta$. The optimal solution of IP′ might not be an optimal VRPB solution. However, it is easy to note that, in this case, $z'(\text{D}) + \Delta$ is a valid lower bound to any optimal VRPB solution.

The optimal solution of IP′ is obtained by means of the integer programming code CPLEX 3.0.

5.5.1 A NUMERICAL EXAMPLE

In this section we show a numerical example to illustrate the new exact procedure EHP. The test problem used is problem eilA10166 (see section 5.6). The number n of Linehaul customers is 67 (i.e. $|L| = 67$), while the number m of Backhaul customers is 33 (i.e. $|B| = 33$). The number M of vehicles is 6, each one with a capacity Q=200. The total demand of the Linehaul and Backhaul customers is 1003 and 455, respectively. Therefore, the minimum number of vehicles needed to visit the Linehaul customers and the Backhaul customers is 6 and 3, respectively (i.e. $M_L = 6$ and $M_B = 3$). The data corresponding to this problem test can be found in Appendix A.2. The cost of the heuristic solution found by the algorithm of Toth and Vigo (1996) is 879.

The value of lower bound $z(\text{D}^1)$ computed by procedure H^1 was equal to 841.2. The lower bound has been obtained by a total number of 300 iterations of procedure H^1 in 201 seconds on a Silicon Graphics Indy (MIPS R4400/200 Mhz processor). The M-vertex disjoint arcs of A_0 in the lower bound solution and the corresponding Linehaul and Backhaul q-paths are given in Table 5.1. Figure 5.5 shows the lower bound solution, where the arcs forming the solution are indicated as bold.

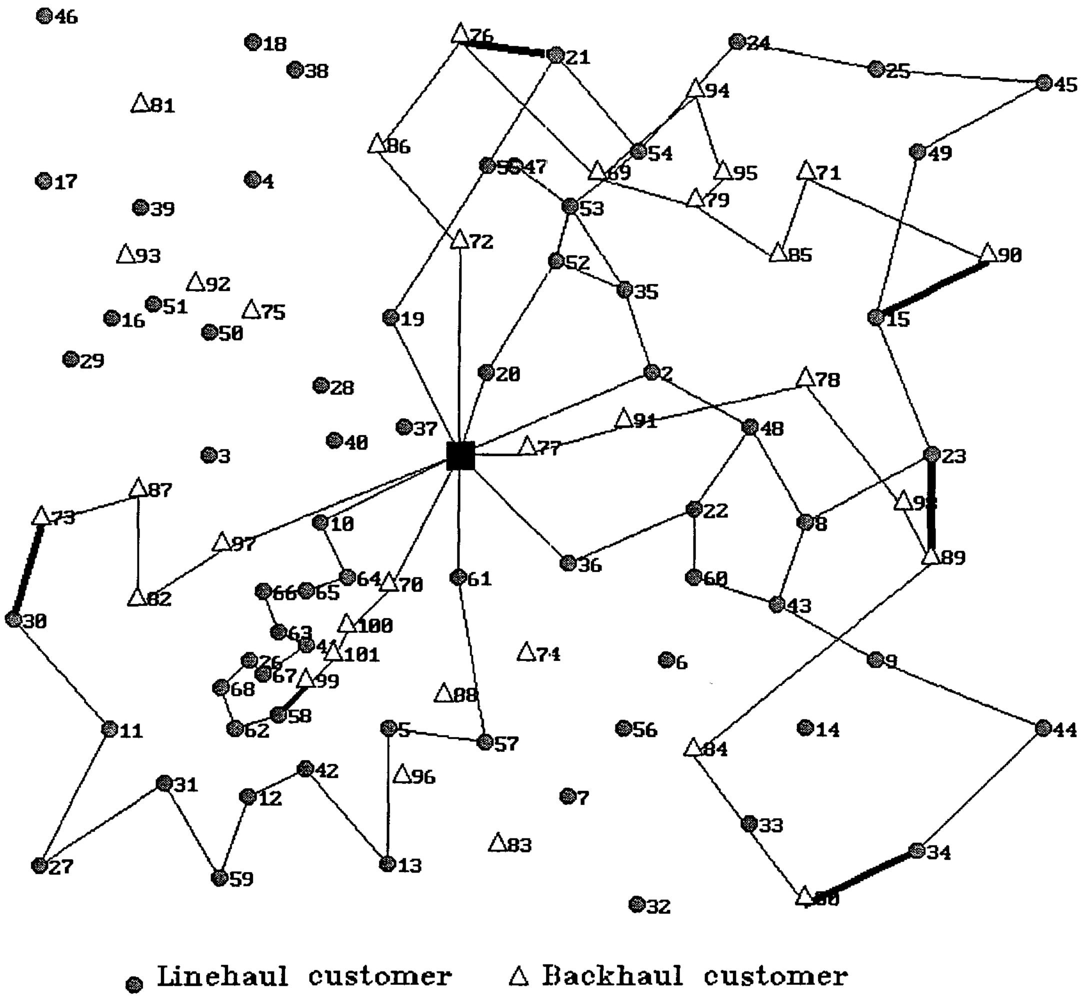


Figure 5.5. Example: lower bound solution to procedure H^1 of cost 841.2

Table 5.1. Example: details of the lower bound solution obtained by procedure H^1

Arc	Delivery q-path	Collection q-path
(15,90)	(1,20,52,35,2,48,22,60,43,8,23,15)	(90,71,85,79,95,94,69,79,95,94,69,76,86,72,1)
(21,76)	(1,36,22,48,8,43,60,22,48,2,35,52,53,54,21)	(76,86,72,1)
(23,89)	(1,2,35,53,47,55,21,54,24,25,45,49,15,23)	(89,98,78,91,77,1)
(30,73)	(1,61,57,5,13,42,12,59,31,27,11,30)	(73,87,82,97,1)
(34,80)	(1,19,55,47,53,52,35,2,48,8,43,9,44,34)	(80,84,89,98,78,91,77,1)
(58,99)	(1,10,64,65,66,63,41,67,26,68,62,58)	(99,101,100,70,1)

The value $z\left(D^2\right)$ of the lower bound obtained by procedure H^2 was 2.1, hence, the value of the final lower bound obtained by procedure HDS was $z'(D) = 843.3$. The total computing time for procedure HDS was 373.0. Table 5.2 reports the details of the lower bound solution produced by H^2 and Figure 5.6 shows the solution, where the bold arcs represent the $\{\xi_{ij}\}$ variables that have a value greater than 0 in the lower bound solution.

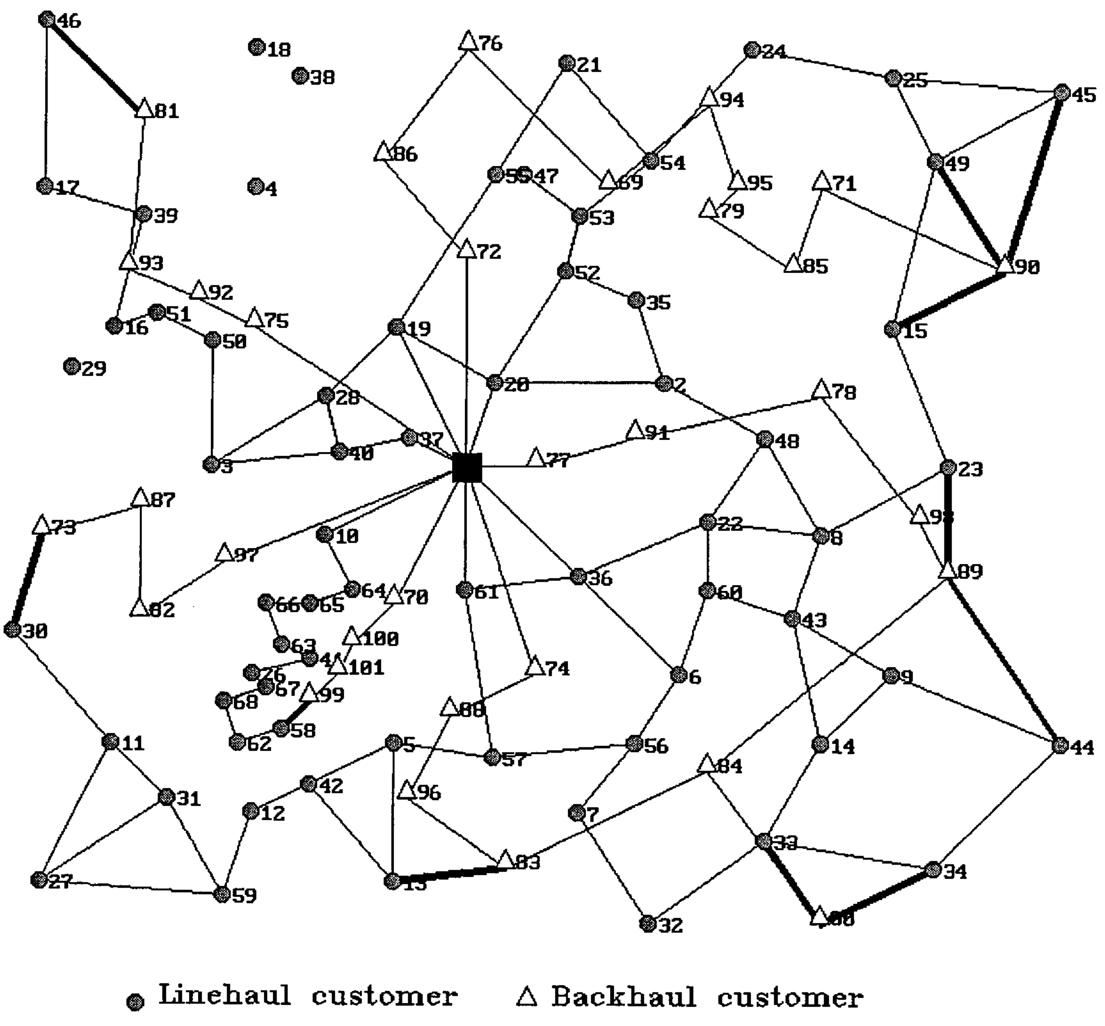


Figure 5.6. Example: lower bound solution to procedure H^2 of cost 2.1

Table 5.2. Example: details of the lower bound solution obtained by procedure H^2

$\{x_\ell\}$ variables

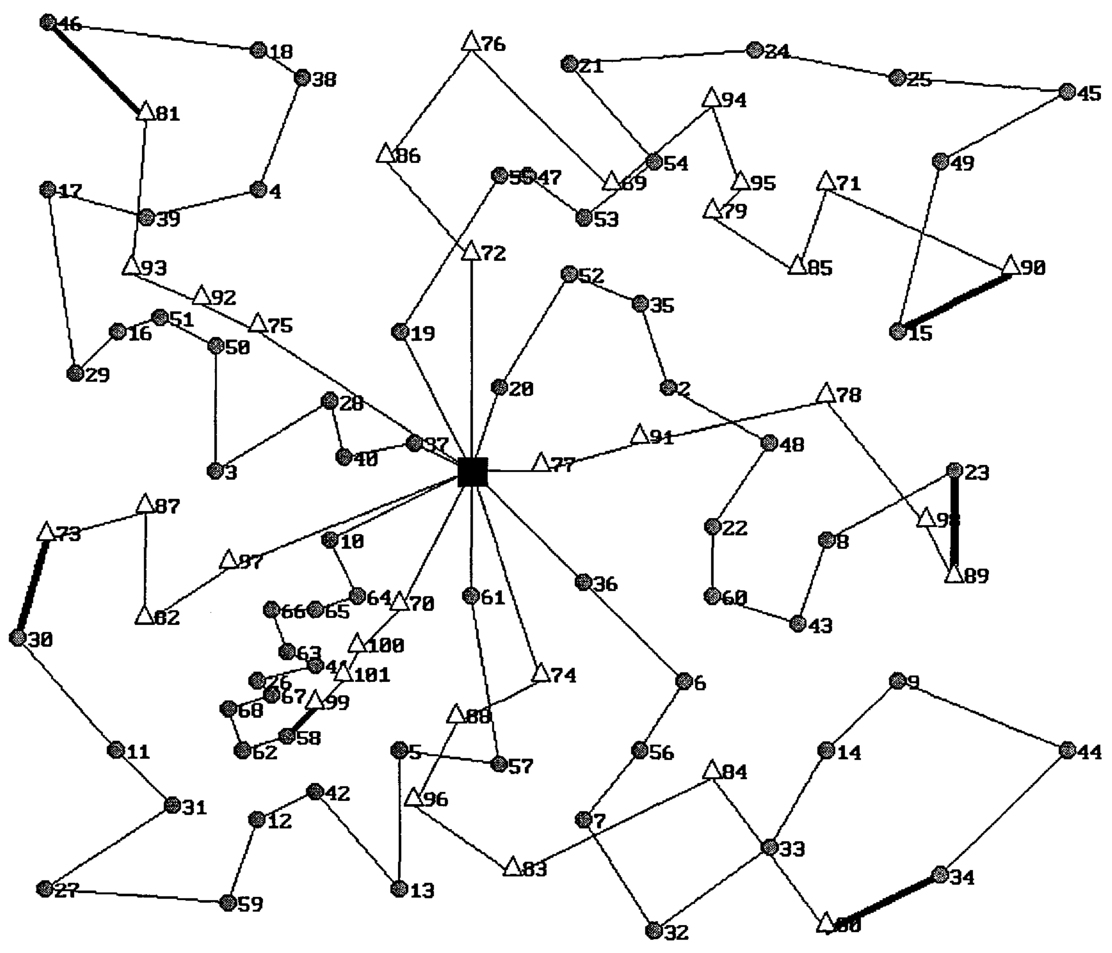
Value	Coefficient	Path
0.20	0.01	(1,20,52,35,2,48,22,60,43,8,23,15)
0.20	0.34	(1,36,22,60,6,56,7,32,33,14,9,43,8,23,15)
0.60	0.02	(1,19,55,47,53,54,24,25,45,49,15,23)
0.60	0.00	(1,61,57,5,13,42,12,59,31,27,11,30)
0.20	0.01	(1,61,57,5,42,12,59,27,31,11,30)
0.20	0.01	(1,36,6,56,57,5,13,42,12,59,31,27,11,30)
0.20	0.00	(1,19,55,47,53,52,35,2,48,8,43,9,44,34)
0.20	0.00	(1,20,52,35,2,48,22,8,43,14,9,44,34)
0.20	0.02	(1,61,36,22,60,6,56,7,32,33,14,9,44,34)
0.20	0.08	(1,36,22,60,6,56,7,32,33,14,9,44,34)
0.20	0.30	(1,20,52,35,2,48,8,43,60,6,56,7,32,33,34,44)
0.20	0.73	(1,20,2,35,52,53,47,55,21,54,24,25,49,45)
0.20	0.40	(1,20,19,28,40,3,50,51,16,39,17,46)
0.80	0.51	(1,37,40,28,3,50,51,16,39,17,46)
1.00	0.00	(1,10,64,65,66,63,41,26,67,68,62,58)

$\{y_\ell\}$ variables

Value	Coefficient	Path
1.00	0.00	(73,87,82,97,1)
0.20	0.00	(80,84,89,98,78,91,77,1)
0.80	0.00	(80,84,83,96,88,74,1)
1.00	0.00	(81,93,92,75,1)
0.20	0.00	(83,96,88,74,1)
0.80	0.00	(89,98,78,91,77,1)
1.00	0.03	(90,71,85,79,95,94,69,76,86,72,1)
1.00	0.00	(99,101,100,70,1)

$\{\xi_{ij}\}$ variables

Arc	Value	Coefficient
(13,83)	0.20	0.04
(15,90)	0.40	0.00
(23,89)	0.60	0.00
(30,73)	1.00	0.00
(33,80)	0.20	0.00
(34,80)	0.80	0.00
(44,89)	0.20	0.07
(45,90)	0.40	0.01
(46,81)	1.00	0.00
(49,90)	0.20	0.01
(58,99)	1,00	0,00



● Linehaul customer △ Backhaul customer

Figure 5.7. Example: optimal solution of cost 846

The cost of the integer solution found by CPLEX was 846, which has been obtained in 61 seconds. The cardinality of the sets of Linehaul and Backhaul paths generated (i.e. $|\mathcal{L}'|$ and $|\mathcal{B}'|$) were 20187 and 15371, respectively. The cardinality of set $|A_0|$ was equal to 1076. The value of Δ was equal to $\Delta = Min\left\{\underset{\ell\in\mathcal{L}'}{Max}[c'_\ell], \underset{\ell\in\mathcal{B}'}{Max}[c'_\ell]\right\} = Min\{3.31, 9.66\} = 3.31$, hence, LS=846.6. Therefore, the solution found is also the optimal solution of the problem since $z(\text{IP}') \leq LS$. Figure 5.7 shows the optimal solution of the problem. The total computing time of procedure EHP was 434 seconds.

5.6 COMPUTATIONAL RESULTS

The algorithm EHP described in Section 5.5 has been coded in Fortran 77 and run on a Silicon Graphics Indy (MIPS R4400/200 Mhz processor) on two classes of test problems. We have used CPLEX 3.0 as the LP-solver in procedure H^2 and as the integer programming solver in EHP.

The problems of class A correspond to a subset of the randomly generated Euclidean VRPB instances proposed by Goetschalckx and Jacobs-Blecha (1989). The problems of class B have been generated by Toth and Vigo (1996) from VRP problems known in the literature. For each VRP problem three VRPB instances have been generated, each one corresponding to a Linehaul customer percentage of 50%, 66% and 80%, respectively. Problem input data of class B have been kindly provided by Toth and Vigo.

To our knowledge, the only exact method presented in the literature for solving these problems has been proposed by Toth and Vigo (1997).

The tables show the following columns :

$z(\mathrm{IP})$:	cost of the optimal VRPB solution (or cost of the best known solution).
$z(\mathrm{UB})$:	cost of the VRPB solution found by the heuristic algorithm of Toth and Vigo (1996).
$z(\mathrm{D}^1)$:	lower bound produced by procedure H^1 after 200 subgradient iterations.
t_{H^1} :	computing time spent by the bounding procedure H^1.
$z'(\mathrm{D})$:	final lower bound produced by procedure HDS.
t_{HDS} :	total computing time of procedure HDS.
$\%\mathrm{E}_{\mathrm{HDS}}$:	percentage error of the lower bound $z'(\mathrm{D})$ computed by procedure HDS.
$\lvert\mathcal{L}'\rvert$:	number of Linehaul paths generated in EHP.
$\lvert\mathcal{B}'\rvert$:	number of Backhaul paths generated in EHP.
LS	$= z'(\mathrm{D}) + \Delta$, where Δ is the value defined in Section 5.5 and it is used by EHP to show the optimality of $z(\mathrm{IP})$ (we set $LS = \infty$ if $c'_\ell > z(\mathrm{UB}) - z'(\mathrm{D}),\ \forall \ell \in (\mathcal{L} \cup \mathcal{B}) \setminus (\mathcal{L}' \cup \mathcal{B}')$).

t_{EHP} : total computing time of procedure EHP including t_{HDS}. We impose a time limit of 25000 CPU seconds. If the time limit is reached, the instance is marked with an asterisk.

$\%\mathrm{E}_{\mathrm{TV}}$: percentage error of the lower bound produced by Toth and Vigo (1997).

t_{TV}: computing time of the exact method TV proposed by Toth and Vigo (seconds of a Pentium 60 Mhz personal computer). If an imposed time limit of 6000 CPU seconds has been reached, the instance is marked with an asterisk. Instances not attempted by Toth and Vigo are marked with **n.a.**.

The percentage errors $\%\mathrm{E}_{\mathrm{HDS}}$ and $\%\mathrm{E}_{\mathrm{TV}}$ are computed as the ratio of the lower bound divided by $z(\mathrm{IP})$ and multiplied by 100. The parameter *Maxsize*, used in GEN$\mathcal{P}$, has been set to 70000 in both procedures H^2 and EHP.

Tables 5.3 and 5.5 show the quality of the lower bounds produced by procedure HDS and by Toth and Vigo for the two classes of problems. Columns $\%\mathrm{E}_{\mathrm{HDS}}$ and $\%\mathrm{E}_{\mathrm{TV}}$ of both tables show that the lower bound obtained by HDS is greater than the lower bound produced by Toth and Vigo, the average values being $\%\mathrm{E}_{\mathrm{HDS}} = 98.2$ and $\%\mathrm{E}_{\mathrm{TV}} = 97.4$ for problems of class A and $\%\mathrm{E}_{\mathrm{HDS}} = 98.3$ and $\%\mathrm{E}_{\mathrm{TV}} = 96.8$ for problems of class B. In fact, out of 64 cases for which comparison is possible, only in three of these did the procedure of Toth and Vigo gave a superior lower bound. Tables 5.4 and 5.6 report the results obtained by the exact method EHP and the exact algorithm of Toth and Vigo. Note that it is difficult to compare directly the computing times required by the two methods since they are relative to different machines. In our experience the Pentium 60 Mhz used by Toth and Vigo is about four times slower than the Silicon Graphics Indy we used. Tables 5.4 and 5.6 indicate that EHP is capable of solving problems up to 90 customers of class A and up to 100 customers of class B. For some problems EHP cannot prove the optimality of the solution produced (this happens when $LS < z(\mathrm{IP})$), however, the distance between $z(\mathrm{IP})$ and LS is small. The computing time required by CPLEX in procedure EHP to solve problem IP' is given by $t_{\mathrm{EHP}} - t_{\mathrm{HDS}}$. We can observe that the CPU time consumed by CPLEX becomes the main component of the total time required by EHP to solve some problems of both classes A and B. We note here that for algorithm EHP it is better to have only a few customers per path (say, an average of 15 customers/path) and that the problem should be "tight" (i.e. the ratios $\left(\sum_{i \in L} q_i\right) / \left(M_L Q\right)$ and $\left(\sum_{i \in B} q_i\right) / \left(M_B Q\right)$ should be greater than, say, 0.9). In this case the sizes of $\mathcal{L}'$ and $\mathcal{B}'$ are small and EHP can be a potentially useful tool for solving practical VRPB problems.

5.7 Summary

In this chapter we have described an exact algorithm for the basic Vehicle Routing Problem with Backhauls (VRPB) based on a new (0-1) integer programming formulation. We have presented a method for computing the lower bound by finding a feasible solution of the dual of the LP-relaxation of its integer program. Such a dual solution is obtained by combining two different bounding procedures where the structure of the second bound is such that additional constraints found in real-world VRPB's can be considered. The exact method uses the dual solution and a method for limiting the variables of the integer program so that the resulting problem can be solved by CPLEX. The overall bounding procedure proved to be effective, being able to produce a lower bound whose value on average was at least 98.2% of the optimum. Computational results show that the proposed method is able to solve exactly VRPB's of size up to 100 customers within the imposed time limit of 25000 seconds.

Table 5.3. Problem class A : lower bounds

PROBLEM DATA						HDS					TV
Prob	n	m	M	M_B	$z(\mathrm{IP})$	$z(\mathrm{D}^1)$	t_{H^1}	$z'(\mathrm{D})$	t_{HDS}	$\%\mathrm{E}_{\mathrm{HDS}}$	$\%\mathrm{E}_{\mathrm{TV}}$
A1	20	5	8	2	229886	215233	2.1	227079	4.0	98.8	98.3
A2	20	5	5	1	180119	170474	1.2	177869	3.4	98.8	98.1
A3	20	5	4	1	163405	154512	5.8	163405	9.1	100.0	100.0
A4	20	5	3	1	155796	148452	5.2	155796	11.6	100.0	100.0
B1	20	10	7	4	239080	233869	10.6	233869	13.2	97.8	96.0
B2	20	10	5	3	198048	193176	29.9	193880	39.1	97.9	97.4
B3	20	10	3	2	169372	169372	3.9	169372	3.9	100.0	100.0
C1	20	20	7	6	249448	236825	9.4	244857	13.6	98.2	95.7
C2	20	20	5	4	215020	207305	10.8	208495	13.9	97.0	96.5
C3	20	20	5	3	199346	197522	18.8	199346	24.5	100.0	99.8
C4	20	20	4	3	195366	193542	18.7	195367	24.4	100.0	100.0
D1	30	8	12	3	322530	306565	3.7	318671	6.2	98.8	97.0
D2	30	8	11	3	316709	292534	3.6	310929	8.5	98.2	94.5
D3	30	8	7	2	239479	224657	4.5	231931	17.6	96.8	95.9
D4	30	8	5	2	205832	194225	21.9	198301	49.4	96.3	95.4
E1	30	15	7	3	238880	229614	5.7	238880	11.2	100.0	95.1
E2	30	15	4	2	212263	206362	21.3	212263	40.9	100.0	97.9
E3	30	15	4	2	206659	199031	32.8	204360	62.2	98.9	98.2
F1	30	30	6	6	263173	248195	6.4	256287	66.4	97.4	96.6
F2	30	30	7	6	265213	254285	6.5	262342	27.6	98.9	98.3
F3	30	30	5	4	241120	229452	9.1	238221	74.8	98.8	98.0
F4	30	30	4	3	233861	221136	11.2	227576	91.3	97.3	97.3
G1	45	12	10	3	306305	292859	14.2	299522	43.3	97.8	91.1
G2	45	12	6	2	245441	237618	24.3	242423	63.6	98.8	93.3
G3	45	12	5	2	229507	221566	30.7	223205	80.7	97.3	96.2
G4	45	12	6	2	232521	223271	30.9	226712	71.6	97.5	96.5
G5	45	12	5	1	221730	213131	38.2	217204	81.7	98.0	97.9
G6	45	12	4	1	213457	204187	49.4	207116	102.8	97.0	96.6
H1	45	23	6	3	268933	262397	98.7	264609	130.5	98.4	96.6
H2	45	23	5	3	253365	249237	67.4	251972	143.7	99.5	99.4
H3	45	23	4	2	247449	242391	79.6	245860	171.3	99.4	99.2
H4	45	23	5	2	250221	244114	32.1	249239	176.4	99.6	99.7
H5	45	23	4	2	246121	239537	94.8	244450	263.4	99.3	99.3
H6	45	23	5	2	249135	243664	74.1	247832	169.4	99.5	99.4
I1	45	45	10	9	353021	338580	56.5	342376	193.2	97.0	n.a.
I2	45	45	7	7	309943	301904	80.2	305923	198.3	98.7	n.a.
I3	45	45	5	5	294833	281061	122.7	285158	274.0	96.7	n.a.
I4	45	45	6	5	295988	286849	121.1	289314	299.6	97.7	n.a.
I5	45	45	7	5	301226	293773	120.6	295935	304.6	98.2	n.a.
J1	75	19	10	3	335006	323922	94.6	329466	150.5	98.3	n.a.
J2	75	19	8	2	315644	295532	123.4	299069	217.2	94.7	n.a.
J3	75	19	6	2	282447	268495	185.2	271767	362.8	96.2	n.a.
J4	75	19	7	2	300548	281414	147.2	285203	259.8	94.9	n.a.
K1	75	38	10	5	394637	379113	104.9	385215	187.2	97.6	n.a.
K2	75	38	8	4	362360	351581	117.7	357327	223.3	98.6	n.a.
K3	75	38	9	4	365693	354651	115.7	360365	219.4	98.5	n.a.
K4	75	38	7	3	358308	336260	142.3	340958	264.1	95.2	n.a.
								Average %dev		98.2	97.4
								Minimum %dev		94.7	91.1

Table 5.4. Problem class A : exact method EHP

PROBLEM DATA							EHP				TV
Prob	n	m	M	M_B	$z(\text{UB})$	$z(\text{IP})$	LS	$\lvert\mathcal{L}'\rvert$	$\lvert\mathcal{B}'\rvert$	t_{EHP}	t_{TV}
A1	20	5	8	2	229886	229886	∞	125	7	5	902
A2	20	5	5	1	180119	180119	∞	242	13	4	209
A3	20	5	4	1	163405	163405 a	∞	-	-	10	3
A4	20	5	3	1	155796	155796 a	∞	-	-	12	3
B1	20	10	7	4	239080	239080	∞	307	69	14	148
B2	20	10	5	3	198048	198048	∞	386	126	40	151
B3	20	10	3	2	169372	169372 a	∞	-	-	4	1
C1	20	20	7	6	253318	249448	∞	945	574	17	227
C2	20	20	5	4	215020	215020	∞	1144	772	18	322
C3	20	20	5	3	199346	199346 a	∞	-	-	25	84
C4	20	20	4	3	195367	195366 a	∞	-	-	25	5
D1	30	8	12	3	322705	322530	∞	339	32	9	289
D2	30	8	11	3	318476	316709	∞	1158	47	13	491
D3	30	8	7	2	239479	239479	∞	4132	160	51	*
D4	30	8	5	2	205832	205832	∞	14696	191	161	*
E1	30	15	7	3	238880	238880 a	∞	-	-	12	476
E2	30	15	4	2	212263	212263 a	∞	-	-	41	788
E3	30	15	4	2	206659	206659	∞	996	288	64	482
F1	30	30	6	6	263929	263173	268630	7201	12019	2049	756
F2	30	30	7	6	265214	265213	∞	805	978	44	891
F3	30	30	5	4	241121	241120	246458	1115	1981	76	468
F4	30	30	4	3	233862	233861	234671	22708	33442	173	3523
G1	45	12	10	3	306959	306305	308396	24678	271	3556	*
G2	45	12	6	2	245441	245441	247176	13705	105	229	*
G3	45	12	5	2	230170	229507 b	227049	38180	351	967	4225
G4	45	12	6	2	232647	232521 b	230648	21336	115	89	*
G5	45	12	5	1	221899	221730 b	220508	17556	434	157	3433
G6	45	12	4	1	213457	213457 c	209922	18946	763	103	840
H1	45	23	6	3	270719	268933 b	265930	2202	374	454	1344
H2	45	23	5	3	253365	253365	256154	6654	534	221	5020
H3	45	23	4	2	247536	247449	249200	5987	1724	177	1465
H4	45	23	5	2	250221	250221	253120	2194	872	179	1287
H5	45	23	4	2	246121	246121	247526	13356	2156	277	406
H6	45	23	5	2	249135	249135	250351	3462	1086	173	416
I1	45	45	10	9	354410	353021 b	349787	55702	57332	20225	n.a.
I2	45	45	7	7	315184	309943	310965	16854	16678	6395	n.a.
I3	45	45	5	5	298367	294833 b	285787	37767	19714	18045	n.a.
I4	45	45	6	5	295988	295988 b	293375	46873	40119	20055	n.a.
I5	45	45	7	5	302709	301226 b	300060	48245	40870	8642	n.a.
J1	75	19	10	3	343476	335006 b	331204	1298	9769	1640	n.a.
J2	75	19	8	2	315644	315644 c	300485	1318	29849	218	n.a.
J3	75	19	6	2	282447	282447 c	272889	827	26266	363	n.a.
J4	75	19	7	2	300548	300548 c	286404	504	25603	260	n.a.
K1	75	38	10	5	408303	394637 b	387804	3713	58698	*	n.a.
K2	75	38	8	4	372423	362360 b	359157	2693	54446	2618	n.a.
K3	75	38	9	4	374417	365693 b	362516	4556	52029	3812	n.a.
K4	75	38	7	3	358308	358308 c	342184	1166	47759	265	n.a.

(a) Optimal solution obtained by procedure HDS.

(b) $z(\text{IP})$ is the cost of the best VRPB solution found by procedure EHP.

(c) No solution found by algorithm EHP of cost smaller than z(UB).

Table 5.5. Problem class B : lower bounds

PROBLEM DATA						HDS					TV
Prob	n	m	M	M_B	$z(\mathrm{IP})$	$z(\mathrm{D}^1)$	t_{H^1}	$z'(\mathrm{D})$	t_{HDS}	$\%\mathrm{E}_{\mathrm{HDS}}$	$\%\mathrm{E}_{\mathrm{TV}}$
eil2250	11	10	3	2	371	369	2.8	371	5.1	100.0	100.0
eil2266	14	7	3	1	366	366	1.0	366	3.0	100.0	100.0
eil2280	17	4	3	1	375	366	3.5	372	5.8	99.2	98.9
eil2350	11	11	2	1	682	682	0.4	682	0.4	100.0	100.0
eil2366	15	7	2	1	649	604	4.5	645	7.6	99.4	98.8
eil2380	18	4	2	2	623	610	5.5	615	8.9	98.7	98.1
eil3050	15	14	2	2	501	473	7.7	501	7.7	100.0	100.0
eil3066	20	9	3	1	537	492	6.4	524	14.1	97.6	98.5
eil3080	24	5	3	1	514	488	7.7	503	24.7	97.9	100.0
eil3350	16	16	3	2	738	737	21.8	738	45.4	100.0	98.4
eil3366	22	10	3	1	750	746	15.3	750	26.6	100.0	94.8
eil3380	26	6	3	1	736	727	18.0	731	42.2	99.3	93.9
eil5150	25	25	3	3	559	550	38.7	557	65.2	99.6	99.3
eil5166	34	16	4	2	548	541	40.2	544	60.6	99.3	97.8
eil5180	40	10	4	1	565	552	47.5	554	104.0	98.1	98.0
eilA7650	37	38	6	5	739	730	67.0	733	110.0	99.2	98.2
eilA7666	50	25	7	4	768	756	75.0	760	135.0	99.0	95.4
eilA7680	60	15	8	2	781	758	89.3	763	195.0	97.7	90.5
eilB7650	37	38	8	7	801	794	45.0	795	62.5	99.3	97.6
eilB7666	50	25	10	5	873	860	54.8	864	97.7	99.0	91.2
eilB7680	60	15	12	3	919	908	65.6	914	115.0	99.5	85.2
eilC7650	37	38	5	4	713	699	88.5	705	186.0	98.9	98.9
eilC7666	50	25	6	3	734	725	100.0	728	196.0	99.2	97.6
eilC7680	60	15	7	2	733	713	62.3	717	131.0	97.8	93.7
eilD7650	37	38	4	3	690	684	109.3	688	182.0	99.7	99.7
eilD7666	50	25	5	2	715	704	119.0	705	236.0	98.6	98.5
eilD7680	60	15	6	2	694	683	140.0	687	310.0	99.0	95.6
eilA10150	50	50	4	4	843	800	167.5	812	363.2	96.3	96.3
eilA10166	67	33	6	3	846	841	201.0	843	373.0	99.6	99.2
eilA10180	80	20	6	2	908	830	222.9	833	430.9	91.7	89.5
eilB10150	50	50	7	7	933	888	96.0	892	210.0	95.6	**n.a.**
eilB10166	67	33	9	5	1056	937	118.8	941	292.6	89.1	**n.a.**
eilB10180	80	20	11	3	1022	992	132.5	993	306.9	97.2	**n.a.**
								Average %dev		98.3	96.8
								Minimum %dev		89.1	85.2

Table 5.6. Problem class B : exact method EHP

Problem Data							EHP				TV
Prob	n	m	M	M_B	$z(\text{UB})$	$z(\text{IP})$	LS	$\lvert\mathcal{L}'\rvert$	$\lvert\mathcal{B}'\rvert$	t_{EHP}	t_{TV}
eil2250	11	10	3	2	389	371 a	∞	-	-	6	3
eil2266	14	7	3	1	366	366 a	∞	-	-	3	6
eil2280	17	4	3	1	375	375	∞	196	6	6	55
eil2350	11	11	2	1	682	682 a	∞	-	-	1	2
eil2366	15	7	2	1	649	649	∞	157	33	7	65
eil2380	18	4	2	2	625	623	∞	657	4	9	36
eil3050	15	14	2	2	501	501 a	∞	-	-	8	3
eil3066	20	9	3	1	542	537	∞	2442	136	17	119
eil3080	24	5	3	1	519	514	∞	7948	12	31	13
eil3350	16	16	3	2	764	738 a	763	-	-	46	292
eil3366	22	10	3	1	763	750	∞	-	-	27	1338
eil3380	26	6	3	1	761	736	748	6745	93	44	1655
eil5150	25	25	3	3	561	559	∞	420	925	66	441
eil5166	34	16	4	2	551	548	553	4366	463	68	2754
eil5180	40	10	4	1	584	565	566	32322	236	691	4436
eilA7650	37	38	6	5	756	739	743	5127	14206	884	15931
eilA7666	50	25	7	4	776	768	770	35299	5635	1205	13464
eilA7680	60	15	8	2	839	781 b	772	44942	718	596	*
eilB7650	37	38	8	7	836	801	∞	2669	9183	124	16345
eilB7666	50	25	10	5	897	873	∞	20246	1879	2918	12990
eilB7680	60	15	12	3	951	919	927	28181	349	821	10413
eilC7650	37	38	5	4	714	713	715	12061	27537	16659	10344
eilC7666	50	25	6	3	748	734	736	44074	2100	952	*
eilC7680	60	15	7	2	757	733 b	724	42230	1314	*	*
eilD7650	37	38	4	3	704	690	695	4753	9050	197	401
eilD7666	50	25	5	2	730	715 b	711	37745	16371	5023	*
eilD7680	60	15	6	2	715	694 b	691	41252	827	20148	*
eilA10150	50	50	4	4	849	843 c	816	4402	8835	364	*
eilA10166	67	33	6	3	879	846	847	20587	15371	434	10913
eilA10180	80	20	6	2	908	908 c	835	1159	55640	431	*
eilB10150	50	50	7	7	954	933 b	900	22571	16152	*	**n.a.**
eilB10166	67	33	9	5	1056	1056 c	946	4331	28666	293	**n.a.**
eilB10180	80	20	11	3	1076	1022 b	996	4939	64198	20199	**n.a.**

(a) Optimal solution obtained by procedure HDS.

(b) $z(\text{IP})$ is the cost of the best VRPB solution found by procedure EHP.

(c) No solution found by algorithm EHP of cost smaller than z(UB).

CHAPTER 6

AN APPLICATION OF MULTI-DEPOT PERIOD VEHICLE ROUTING TO EFFICIENT RESOURCE PLANNING

6.1 INTRODUCTION

The Period Vehicle Routing Problem (PVRP) involves the design of effective vehicle routes that satisfy customer service frequencies over a specified planning horizon. The PVRP generalizes the classical VRP by extending the planning period from a single day to *p* days. Over a *p*-day period, each customer must be visited at least once with some customers requiring several visits. For example, a customer might requires two visits during the period (say a 5-day week) and the allowable combinations for the visits might be Monday-Friday or Monday-Thursday.

The PVRP has been used to model many practical problems. Beltrani and Bodin (1974) and Russell and Igo (1979) encountered the PVRP in refuse collection where the routing is planned weekly and each site requires a different number of collections during the week. Ball (1988) and Dror and Ball (1987) discussed applications of the PVRP to fuel oil delivery and industrial gas distribution problems. Golden and Wasil (1987) discussed an application of the PVRP to the distribution of soft drinks where customer demands may be stochastic. In addition, the PVRP can be applied to mail collection and delivery problems, as well as scheduled retail and wholesale delivery problems. In one variant of the PVRP that is proposed by Raft (1982), customer demand is stochastic.

Whether or not a customer will be visited during the p-day period is also stochastic, and the fleet of vehicles may be stationed at more than one depot. Gaudioso and Paletta (1992) model a variant of the PVRP that determines the scheduling and routing policies of vehicles so that the maximum number of vehicles (i.e., the fleet size) simultaneously deployed over the entire planning period is minimized.

Algorithms for the PVRP have been published by Christofides and Beasley (1984), Tan and Beasley (1984), Russell and Gribbin (1991), Chao et al. (1995) and Cordeau at al. (1995). Christofides and Beasley (1984) developed two heuristics that consist of an inizialization step followed by an improvement procedure. The first heuristic is based upon the idea that minimizing the sum of the radial distances from the customers to a *center* that is specified for each day of the planning period would also tend to minimize the distance travelled in the underlying PVRP. The second heuristic is based upon the idea that minimizing the total distance for TSP's on each day of the planning period would also serve to minimize the total distance for the PVRP. Christofides and Beasly generated 11 test problems which they solved using both heuristics. Tan and Beasley (1984) developed a solution procedure for the PVRP that uses the generalized assignment heuristic of Fisher and Jaikumar (1981b) initially designed to solve the VRP. Tan and Beasley used their procedure to solve five problems drawn from Christofides and Beasley (1984). Russell and Gribbin (1991) proposed a four-phase solution approach for the PVRP that makes use of the interchange heuristic of Christofides and Beasley (1984) to solve a surrogate TSP. Russell and Gribbin solved eight problems taken from Christofides and Beasley (1984) and two new problems that they generated. Chao et al. (1995) presented a new heuristic for the PVRP based on the notion of *record-to-record* improvement. These authors use integer linear programming to assign a visit combination to each customer. They then solve a VRP for each day by means of a modified version of the Clarke and Wright algorithm (1964). Local improvements are then obtained by using the record-to-record approach of Dueck (1990) and 2-opt interchanges. Reinitializations are finally performed to diversify the search. Chao et al. have applied the heuristic to problems from the literature as well as to new test problems. Courdeau et al. (1995) describe a tabu search heuristic method for the PVRP. Computational results show that the algorithm proposed by Courdeau et al.

outperform the heuristics of Tan and Beasley (1984), Christofides and Beasley (1987), Russell and Gribbin (1991) and Chao et al. (1995).

The application described in this chapter is concerned with a utility company, which is responsible for the preventive maintenance of a geographically dispersed network of customers. In the specific area examined, a fleet of 17 mobile gangs is dispatched from 9 depots to call on 162 customers with frequency that can vary from once per day to once every four weeks. Each gang, consisting of two field service workers in a van, visits on average four customers per day. A depot is assumed, for planning purposes, to be both the start and finish point for a gang's daily route. The number of mobile gangs based at each depot varies at present between 1 and 3. The depots operate independently and serve their own designated set of customers. There are limits on the length of the working day imposed by working practices. The company's objectives are to examine its mobile resource planning system in view of making it more cost effective, that is, provide improved customer service at lower cost.

In this chapter we present an algorithm for solving the above problem, which is formulated as a Multi-Depot Period Vehicle Routing Problem (MDPVRP). The computational implementation of the complete planning model is described with reference to the pilot study and results for selected model runs are presented.

The chapter is organized as follows. In the next section we describe the MDPVRP. In the third section, we develop a new heuristic for solving the MDPVRP. Section 6.4 describes the computational implementation of the complete planning model. In Section 6.5 we explore the performance of the MDPVRP algorithm and evaluate the results of various runs of the model. Finally, conclusions and future work are presented in Section 6.6.

6.2 PROBLEM OVERVIEW

There are three major decision problems associated with the maintenance operations provided by the utility company that involve different levels of planning: strategic and tactical/operational. The first problem is to find the most efficient boundaries of the geographical area served by each depot in order to achieve a specified level of service.

The second problem is to determine, over a given time period, optimal visiting patterns for the fleet of gangs, that is, to plan the scheduling of the gang visits to all customers within each depot service area in the best possible way. The third problem is to perform efficient route planning for the mobile gangs within the optimal depot territories subject to a variety of constraints. Some of the goals identified in these problems may conflict and this must be accounted for in the solution methodology. The underlying theme is the integration in a unified model of all the decision problems mentioned above because, we believe, that efficient solutions to tactical and operational problems such as scheduling and route planning may be successfully incorporated in satisfactory solutions to strategic problems. If service and maintenance operations are improperly planned, this may result in poor customer service, waste and inefficiency.

The problem studied in this chapter includes many quantitative restrictions on the gangs as well as those defined by the customers. The complexity of such planning problems affects the algorithms that can be employed in practical situations to generate and evaluate feasible solutions. In this chapter, we address this problem as a MDPVRP and we solve it using the heuristic method presented in the next section. The solution methodology will enable the users to generate the best proposals for the mobile resource planning system and/or evaluate proposed scenarios with respect to certain practical aspects or constraints of the system. In our approach, particular attention is paid to obtaining a well-balanced use of the resources (minimum number of mobile gangs) on different days of the planning period. The proposed methodology is applicable in a broader context to many real distribution problems that exhibit a similar underlying network structure.

6.3 A HEURISTIC ALGORITHM FOR THE MDPVRP

In this section we present a heuristic algorithm for the MDPVRP.

Within the context of the mobile resource planning problem described in the previous section, the classical (single-day) VRP involves a given set of customers that must be visited by mobile gangs operating from a *single* depot (terminal). Each customer has a known service requirement that must be satisfied by one visit of a gang. Each gang route starts and finishes at the depot. The objective is to sequence the visits of each gang so that customer requirements are satisfied and travel costs are minimized.

The MDPVRP generalises the classical VRP by allowing the gangs to operate from one of several depots instead of only one and by extending the planning horizon from a single day to several days. It is important to note that the MDPVRP is a multilevel combinatorial optimisation problem. At the first level we need to define boundaries for each depot service area. At the second level, we need to solve a PVRP for each depot which involves determining a set of customers to be visited on each day of a planning horizon. At the third level, we need to solve a classical VRP for each depot and for each day of the given period. At the fourth level, we need to solve a classical TSP for each route. The classical TSP has been shown to be NP-hard, so the MDPVRP is at least as difficult. A number of exact and heuristic methods have been developed to solve large-scale TSP's. For a large-scale MDPVRP, solving this nearly intractable problem using an exact method would be very difficult and time-consuming, not least because for a given depot and a given choice of customer visit-day combinations there is a large number of resulting VRP's which are difficult to solve. Consequently, in this section we present a new heuristic procedure for the MDPVRP in which a PVRP is solved for each depot.

Basic notation

The PVRP for a single depot is defined as follows.

Given the service requirement of each customer and a set of allowable visit combinations, we need to simultaneously select a visit combination for each customer and establish feasible gang routes for each day of a planning horizon, according to the VRP rules. The objective in the PVRP is to service all customers the required number of times over the planning horizon so that the travel costs are minimised. The following notation is used:

n	the number of customers served from a specific depot
p	the number of days in the planning horizon
H	the length of a working day (hours)

f_i	frequency of service required by customer i, i.e. the number of times customer i is visited over the planning horizon $(i = 1,\dots,n)$
q_i	the mean service time requirement of customer i, $(i = 1,\dots,n)$, $0 \le q_i \le H$
S_i	the set of allowable combinations of days for visiting customer i, $(i = 1,\dots,n)$
R_t	the set of customers scheduled to be visited on day t $(t = 1,\dots,p)$

If a customer i requires f_i visits during a planning period of p days, then these visits may only occur in one of a given number of allowable f_i-day *combinations*. For example, if $f_i = 2$, $p = 5$ (i.e. a planning period of one week) and $S_i = \{(1,3),(1,5),(2,4),(3,5)\}$, then customer i must be visited twice a week and these visits could take place either on Mondays and Wednesdays (i.e. days 1 and 3 of the week) or on Mondays and Fridays or on Tuesdays and Thursdays or on Wednesdays and Fridays, with no other day combinations being acceptable.

Below we present a new heuristic for the PVRP, which includes the following specific constraints:

i) Only one allowable service combination is chosen for each customer.

ii) Each customer must be visited by only one gang on a given day but may be visited by different gangs during the given p-day period.

iii)The number of gangs available daily at a given depot has an upper bound. The total number of gangs available for all depots has to be less than or equal to 17.

iv)The travel times associated with a route plus the service times required by the customers visited on the route must not exceed the maximum length of a working day.

Algorithm 6.1: PVRP

Step 0. Initialization Step

Since there is no easy method of clustering customers based on the frequency of visits, an initial depot-customer allocation is obtained by assigning each customer to its closest depot.

Step 1. Form an ordering of the customers

Order the customers already assigned to each depot, according to some heuristic ordering rule. Let U denote the resulting list of customers for a specific depot. For example, arrange the customers in decreasing order of "importance" expressed both in terms of visiting frequencies and mean service times. First sort the customers in non-increasing order with respect to the f_i's and order the customers that are characterised by the same value of f_i, in non-increasing order with respect to the service times q_i. All customers with a fixed visit combination are placed at the top of this list.

Step 2. Assign and evaluate visit combinations

Define R_t to be the set of customers to be visited from a specific depot on day t of the given planning period. A set R_t is obtained using a least-cost insertion heuristic in which all the customers in U are considered, one at a time, in the order listed. The heuristic involves inserting a specific customer $i \in U$ into the emerging cluster R_t of day t only if an allowable combination $r \in S_i$ includes a visit to customer i on day t. Customer visit combinations are assigned in this way provided all days remain feasible, that is, the total number of gangs required by the emerging customer cluster on each day of the planning period does not exceed the number of depot-based gangs available.

Each time an additional customer is inserted into the emerging cluster R_t of day t, a VRP needs to be solved in order to find the associated least travel cost for day t. This is computationally very difficult to do even for a small number of customers per day. Therefore, we use a heuristic algorithm to solve a single day's problem and choose the combination set that gives the lowest overall travel cost (all days feasible). In particular, a 2-opt and a 3-opt interchange procedures (see Lin and Kernighan (1973)) are applied to evaluate the increase in cost incurred when a customer i is added to the emerging cluster R_t of day t. A local optimisation procedure is then performed to improve each day's solution. This procedure is based on interchanges involving customers that are near to each other, but which are served on different routes operating during the same day.

Step 3. Solve daily VRP's

Once the final sets R_t of customers scheduled to be visited on each day t of the planning period have been determined, we apply a tabu search algorithm to solve the resulting VRP for each day. tabu search is a local search metaheuristic proposed independently by Glover (1986) and Hansen (1986). For recent surveys of this method, see Glover and Laguna (1993a) and Glover et al. (1993b). In this step, the tabu search algorithm constructs least-cost gang routes that satisfy the constraints placed on the time duration of a daily route. When the number of gangs allocated to each depot is a decision variable, the tabu solution may show variations in the number of gangs required by a given depot during the planning period. In this case, the maximum number obtained over this period will determine the final number of gangs assigned to that specific depot.

Step 4. Perform customer interchanges

We attempt to improve the solution by evaluating interchanges of customer combinations and performing those that reduce the total cost. We select (i) a subset of customers that are served by different routes from the same depot on different days of the planning period and (ii) a subset of customers that are assigned to different depots and may be served on the same or different days of the planning period. Customer interchanges of type (ii) can improve the initial depot-customer allocation obtained at the *initialization step* taking into account local knowledge already available from PVRP solutions of existing depots. Repeated applications of this step allow an evolutionary depot-customer reallocation to be achieved by the algorithm.

The visit combination possibilities are subsequently enumerated for subsets of type (i) and (ii), seeking an improved overall solution. We use a 3-opt procedure to evaluate the cost of interchanges. Clearly, for such a procedure to be computationally practical only a restricted number of customer subsets can be considered (for example, customers who are visited once or twice a week).

Step 5. Seek improved customer-visit combinations

A different ordered list U of customers is obtained at Step 1 and a new iteration of Steps 2 to 4 is performed. The criterion for ordering is based on an alternative function of customer visiting frequencies and mean service times.

The heuristic algorithm is terminated after a fixed number of iterations specified by the operator.

6.4 Major elements of the computer system

In this section we describe the computational implementation of the complete planning model which permits the identification and evaluation of:

a) the efficient scheduling of customers and routing of mobile gangs within existing depot territories;
b) new boundaries on each depot service area in view of improving the performance of the existing depot configuration;
c) the effect of closing down some existing depots or opening new depots in desired locations proposed by the user.

The computer system uses the following input data files: a *customer file*, a *resource file* and a *time/distance file*.

The *customer* file

The data supplied by the company is stored in the following files.

The *customer* file contains details of the 162 customers served by the company in the pilot study area including information on customer name, location, the frequency of visits and the mean service time required for each customer. The mean service time recorded for each customer is the actual average working time of a mobile gang during a customer visit. This includes the time required to drive in and out of the customer's site and perform all the necessary routine maintenance operations. Table 6.1 summarises the data on the frequency of customer service currently provided by the company. The first two columns give the number of customers currently being served from each depot and the number of mobile gangs based at each depot. Columns (3)-(8) show the number of customers within each depot service area visited with varying frequency. The last three columns show the average frequency of customer visits per week, the average service time per customer visit and the average number of customers in a gang route within each depot territory.

The location of each customer or depot is defined by (x,y) geographic co-ordinates recorded using an 8-figure grid reference system. The graphical layout of the customer and depot locations (depots D1-D9) and the service area for each depot (i.e. the set of customers currently being served from each depot) are shown in Figure 6.1.

The *resource* file

The *resource* file contains a description of each mobile gang in the system, including its capacity, its base depot (this is the present origin/ destination point for the gang) and the list of customers currently assigned to the gang. The capacity of a mobile gang is defined by the maximum length of a working day which, including travel times, is taken to be 8 hours.

The *time/distance* file

The travel time and distance between pairs of customer and depot locations is an important input to the route planning model. The theoretical travel times are mainly used in strategic planning to show how routes vary with different circumstances. For day-to-day planning a computerised vehicle scheduling system is intended to help schedulers in their effort to produce manual schedules, rather than replace them; they will have to cope with difficulties and real traffic and travel times manually.

The common approach of computing straight line distances from grid reference co-ordinates and then adjusting them by some factor to approximate road miles (or kilometres) was judged to be inaccurate for our application and it provides no information about travel times. In the pilot study area there are about 14500 customer-to-customer and depot-to-customer travel times or distances. To achieve the necessary level of accuracy for distances and travel times in a practical manner, a vendor was contracted to develop to our specifications a comprehensive computerised network representation of the road map in the region concerned. This road network contains about 2700 road segments (links) and about 2000 intersections of road segments (nodes identified by grid references). Figure 6.2 shows a graphical representation of the road network; the intersections of the road segments are identified by small dark dots, the depots by large rectangles and the customers by grey circles. Each road segment has a length and a classification (or category) such as motorway, dual carriageway, rural road or urban road, so that different average travel speeds of the van used by a mobile gang can be established.

Each possible intercustomer distance and travel time is determined by finding the shortest path between pairs of customers through the road network. The criterion used to determine the shortest path is not solely distance but is a travel-cost function expressed as a linear combination of travel time and distance. The time-related component of travel cost reflects an hourly labour cost (drivers' hourly rate) while the distance component (expressed as cost per km) primarily reflects vehicle operating costs such as fuel, tyres, and maintenance. The travel-cost function represents an equally cost weighted combination of both factors.

The shortest route algorithm itself is an efficient dynamic programming algorithm developed by Dijkstra (1959).

6.5 RESULTS

In this section we explore the performance of the MDPVRP algorithm and evaluate the results of various runs of the model. The development and experimentation was undertaken on an IBM 486 machine. The source code is written in FORTRAN.

The model can be used both as a tool for evaluating management strategies and also as part of a computerised vehicle scheduling system at each depot. The output from model runs include:

a) Boundaries of the geographic region served by each depot. This output includes the customer list assigned to each depot and the associated lowest cost of service.
b) Detailed work schedules over a given planning horizon showing the list of customers visited daily by each mobile gang based at each depot.
c) Least-cost routes (order of work) for each mobile gang based at each depot.

d) Evaluation of alternative depot configurations.

Results are reported for the scenarios described in Table 6.2 and are based on a two-week (or ten-day) planning horizon.

Base run: Depot D9

Depot D9, with one mobile gang serving an area of 18 customers, was selected as the base run to evaluate the route planning module of the model. The model computed minimum cost routes for the mobile gang indicating customer visiting sequences, together with the expected route distances and time durations. Table 6.3 compares the model's schedule with the manual schedule (see Figure 6.3). All computations are based on the use of the road network. From these results, it is noted that, should the computed routes be made operational, a 14% reduction in the travel cost for this depot service area would be achieved (the weekly distance being reduced by 21% and the associated travelling time by 9%) through a reorganisation of the gang's current weekly schedule.

Scenario 1: Efficient route planning within existing depot service areas

Based on the current boundaries of the geographic areas served by each depot, the model efficiently allocates customers to the mobile gangs based at each depot and designs least-cost routes for each gang.

Two versions of this scenario were examined: scenario 1a allows a given customer to be visited by different mobile gangs on different days of the planning period; scenario 1b evaluates the effect of imposing the constraint that each customer must be visited by the same gang throughout the period. Table 6.4 shows the results of these two scenarios for all depots. The main conclusion emanating from this comparison is that the additional constraint results in an 11% increase in the travel cost incurred.

Scenario 2: New depot service areas

One of the most notable results of the model is the impact of the overall reorganisation of current depot service areas on the performance of the current system. The model finds the most effective layout of depot territories, which satisfies a specified level of customer service and service requirements. Tables 6.5 and 6.6 summarise the results of this run and Figure 6.3 shows the graphical layout of the new depot service areas. The new depot territories have considerable operational implications for the local management of the depots. The most significant changes occurred for depots D5, D8 and D9. The acceptance of the new and the ceding of the old customers by depots will result in very different patterns of visits if the existing levels of service to customers are to be maintained. Thus, it is quite evident that substantial changes in the traditional routes operated by the depots will be necessary.

Two versions of this scenario were examined; scenario 2b evaluates the same constraint imposed on scenario 1b. It can be seen from Table 6.6 that, for the new depot service areas, scenario 2b incurs a 16.8% increase in the travel cost compared to that for scenario 2a. Comparing the results of scenarios 1a and 2a (see tables 6.4 and 6.6, respectively), the model indicates that effective reorganisation of current depot service areas saves the company at least 12.5% in travel cost. This is an additional saving to that achieved in scenario 1a through improvement of current weekly schedules.

Scenario 3: An alternative depot configuration

An important feature of the model is its ability to evaluate the benefits or otherwise of closing down one or more of the existing depots or establishing a new depot at any desired location in the region. An example of a scenario proposed by management and evaluated by the model was the closure of depot D1. This run resulted in a minor reduction in the total distance and time obtained for scenario 2a, as the customers originally assigned to depot D1 are now served from depot D5 on shorter routes which have been appropriately modified to serve the additional set of customers.

6.6 SUMMARY

The results of this study illustrate the significant reductions in travel cost that can be achieved by finding new boundaries for the depot service areas and making more effective decisions in daily mobile resource planning. It also shows how the mathematical model can be used as a planning tool to evaluate a number of "what-if" scenarios and the resulting benefits.

The main findings are summarised as follows:

1. Given the current boundaries of the region served by each depot, effective reorganisation in scheduling visits and routing of gangs can lead to substantial savings in travel cost (e.g. a 14% reduction can be achieved in the case of one depot, namely depot D9 - base run).
2. Further reductions of at least 12.5% in the travel cost can be achieved for all depots by dividing the whole territory under study into *new* depot service areas (as shown by the results of scenario 2a).

Future work may involve the development and implementation of a computerised vehicle scheduling system in order to complement, rather than replace, the efforts of the existing schedulers. This is possible mostly by recent advances in computer technologies, graphic interfaces and geographic information systems. The system can be used at each depot to produce a detailed schedule for a given planning horizon. Visiting frequencies and customer requirements should be updated interactively as they are received. Such an on-line system requires a user-friendly interface that allows the scheduler to query and update data, execute the scheduling module and change the schedule as desired.

Table 6.1. Problem Data: Frequency of Customer Service

Depot service area	(1) # customers assigned	(2) # mobile gangs	# customers visited:						(9) Average # weekly visits per customer	(10) Average service time per customer visit (hrs)	(11) Average # customers per gang route
			(3) daily	(4) three times/ week	(5) twice/ week	(6) once/ week	(7) once/ two weeks	(8) once/ four weeks			
D1	3	1	-	3	-	-	-	-	3.0	1.50	1.80
D2	30	2	3	1	17	9	-	-	2.03	1.08	6.10
D3	12	1	1	1	6	4	-	-	2.0	1.29	4.80
D4	21	2	-	3	12	6	-	-	1.86	1.52	3.90
D5	20	3	9	-	1	10	-	-	2.85	1.50	3.80
D6	19	2	4	6	-	9	-	-	2.47	1.26	4.70
D7	9	2	3	3	1	-	-	2	2.94	2.16	2.65
D8	30	3	6	4	-	-	20	-	1.73	1.46	3.46
D9	18	1	1	6	-	11	-	-	1.89	0.78	6.80
All Depots	**162**	**17**	**27**	**27**	**37**	**49**	**20**	**2**	**2.15**	**1.36**	**4.22**

Table 6.2. List of Scenarios evaluated by the Model

BASE CASE: Depot D9 Comparison between the model's routes and actual routes performed in practice
SCENARIO 1*: Existing depot service areas Nine depots Given allocation of customers to depots Given number of mobile gangs per depot Improved allocation of customers to mobile gangs Improved route planning within existing depot service areas
SCENARIO 2*: New depot service areas Nine depots Improved allocation of customers to depots Minimum number of mobile gangs per depot Improved allocation of customers to mobile gangs Improved route planning within new depot service areas
SCENARIO 3: An alternative depot configuration Closure of depot D1, Eight depots New depot service areas
*** Two versions of these scenarios, (a) and (b), are considered.** **Scenarios 1(b) and 2(b) include the additional constraint that:** Each customer must be served by the SAME mobile gang throughout the planning period.

Table 6.3. Results of the Base Run for Depot D9

Total # of customers assigned = 18 # of mobile gangs = 1, Planning horizon (weeks) = 1	
MANUAL WORK SCHEDULE	
Day	**Order of Work**
Monday	01-02-03-16-04-05-06-07-01
Tuesday	01-11-09-10-15-01
Wednesday/ Friday	01-02-03-04-05-06-07-01
Thursday	01-08-12-14-18-13-17-01
***Total Travel Distance** = 292.8 km , **Total Travel Time** = 5.5 hours*	
MODEL'S WORK SCHEDULE	
Day	**Order of Work**
Monday	01-02-03-04-05-17-06-07-01
Tuesday	01-15-10-16-11-13-18-14-08-12-09-01
Wednesday/ Friday	01-02-03-04-05-06-07-01
Thursday	NO ROUTE
***Total Travel Distance** = 231.5 km , **Total Travel Time** = 5.0 hours*	

Table 6.4. Scenario 1: Efficient Route Planning within Existing Depot Areas

Planning horizon: two weeks				
Depot	**Scenario 1a**		**Scenario 1b**	
	Distance (km)	**Time (hrs)**	**Distance (km)**	**Time (hrs)**
D1	154.2	3.42	154.2	3.42
D2	685.0	14.98	838.2	18.12
D3	655.6	13.10	655.6	13.10
D4	606.2	13.04	731.2	15.72
D5	1085.6	22.96	1091.2	23.40
D6	545.0	10.80	579.8	11.36
D7	558.0	11.14	561.9	11.62
D8	1357.3	28.40	1695.7	35.57
D9	463.0	10.00	463.0	10.00
All Depots	**6109.9**	**127.84**	**6770.8**	**142.31**

Table 6.5. Existing and New Depot Service Areas

Depot	**Existing depot areas**		**New depot areas**	
	# customers assigned	**# mobile gangs**	**# customers assigned**	**# mobile gangs**
D1	3	1	5	1
D2	30	2	29	2
D3	12	1	10	1
D4	21	2	18	2
D5	20	3	15	2
D6	19	2	20	2
D7	9	2	9	2
D8	30	3	22	2
D9	18	1	34	3
All Depots	***162***	***17***	***162***	***17***

Table 6.6. Scenario 2: Efficient Route Planning within New Depot Areas

Planning horizon: two weeks				
Depot	**Scenario 2a**		**Scenario 2b**	
	Distance (km)	**Time (hrs)**	**Distance (km)**	**Time (hrs)**
D1	334.0	7.36	334.0	7.36
D2	708.0	15.10	789.4	16.88
D3	261.6	5.92	261.6	5.92
D4	498.2	10.64	507.4	10.64
D5	641.6	13.62	704.6	14.96
D6	626.2	12.52	728.8	14.54
D7	586.4	11.92	591.2	12.22
D8	680.5	14.56	773.4	16.51
D9	972.9	20.43	1517.4	31.83
All Depots	**5309.4**	**112.07**	**6207.8**	**130.86**

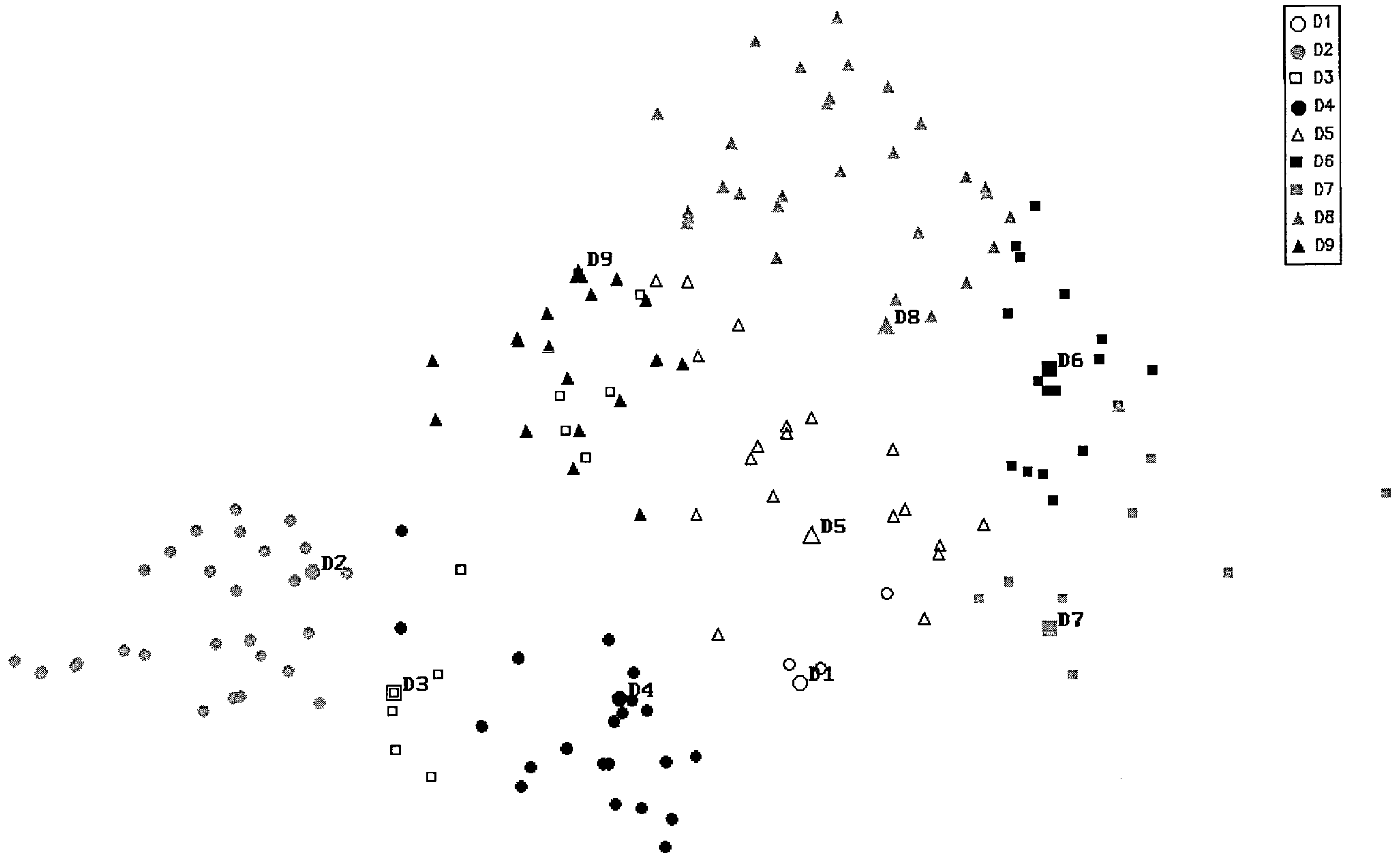


Figure 6.1. Scenario 1: Existing Depot Service Areas

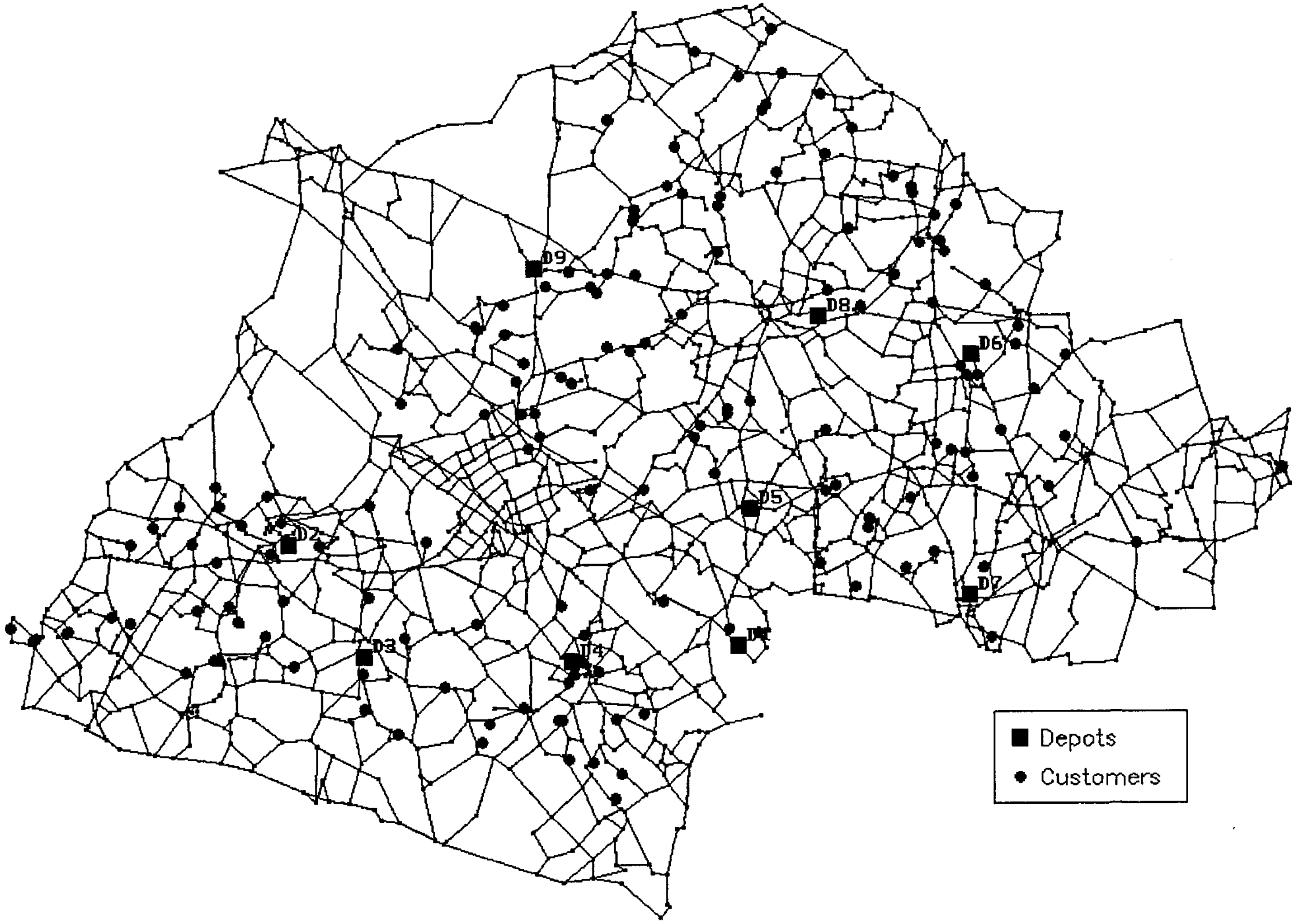


Figure 6.2. The Road Network

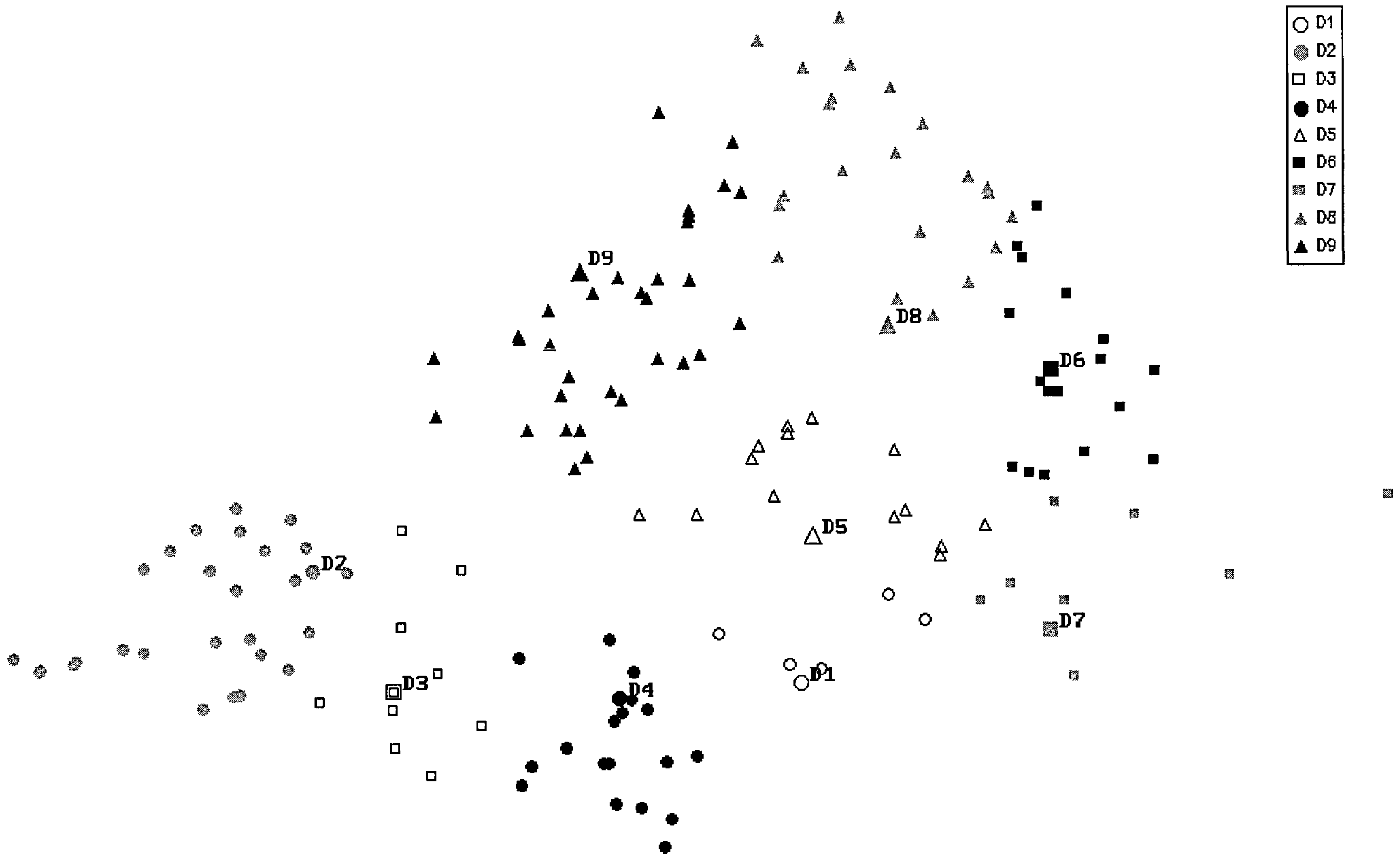


Figure 6.3. Scenario 2: New Depot Service Areas

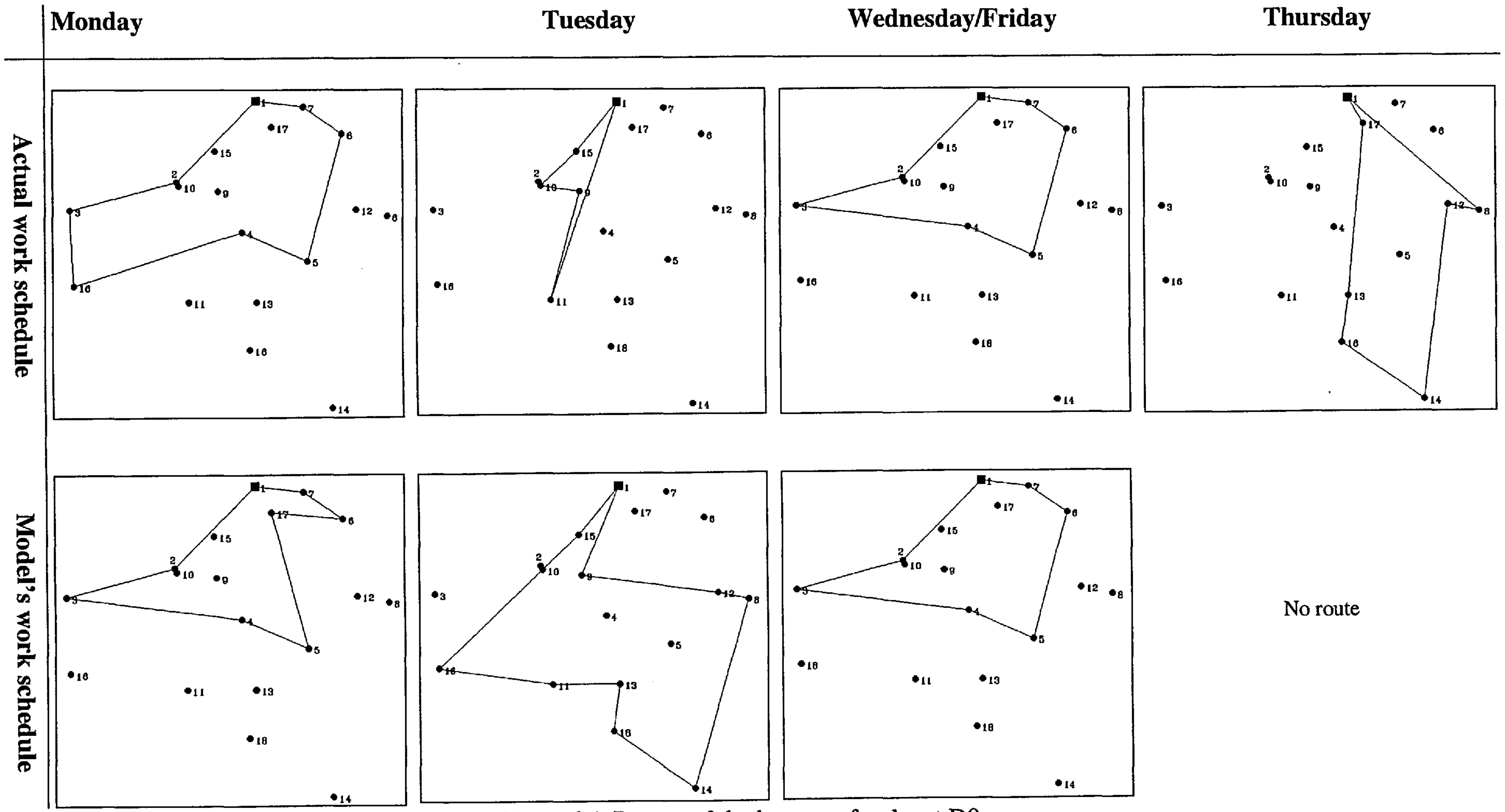


Figure 6.4. Routes of the base run for depot D9

CHAPTER 7

CONCLUSIONS

In this chapter we provide a summary of the entire thesis highlighting the main contributions of the completed work and some suggestions for further research.

This thesis is concerned with the study of location and routing problems in distribution systems. The research objective was to develop both exact and heuristic solution algorithms for a class of different location and routing problems.

The thesis began by describing a new metaheuristic technique, called the Bionomic Algorithm, to solve the Capacitated p-Median Problem (CPMP). Bionomic Algorithms are evolutionary metaheuristic algorithms that update a whole set of solutions (a population of solutions) at each main cycle. They differ from similar previously presented algorithms, namely Genetic Algorithms and Evolution Strategies, because they explicitly direct the choice of solutions to combine in order to define an offspring, that is, a solution in the population of the next iteration. This feature introduces a diversification mechanism for clustering by reference to maximal independent sets, carried out over progressively smaller domains, to provide a specific refinement of the scatter search proposal for generating parents from clustering strategies. The parent selection process, together with the use of problem-specific ways to produce an offspring from the parents, makes Bionomic Algorithms well-suited to combinatorial optimization applications. In the Bionomic Algorithm developed for the CPMP, the problem-specific steps, maturation and propagation, were implemented as follows. Maturation is based on a state-of-the-art heuristic for the Generalized Assignment Problem, a problem to which CPMP reduces once the p medians are chosen. Propagation, specifically the definition of a child solution once a parent set is assembled, is based on the computation of a Lagrangean lower bound for the CPMP. The computational results, presented both on standard data sets from the literature and on more difficult symmetric and asymmetric cost instances, attest the effectiveness of the approach. Our findings can motivate future research that could examine additional types of clustering strategies such as incorporating intensification criteria and diversification criteria for choosing and combining multiple parents. Furthermore, since the results have shown that the Bionomic Algorithm is computationally competitive with other sophisticated heuristic methods, its application to other combinatorial optimization problems, such as Vehicle Routing Problems (VRPs), can be investigated.

A significant contribution to the CPMP literature has also been made by the new exact method we developed based on a Set Partitioning formulation of the problem. A valid lower bound to the optimal solution cost has been obtained by combining two different heuristic methods for solving the dual of the LP-relaxation of the exact formulation. The dual solution obtained has been used for generating a reduced set partitioning problem that can be solved by an integer programming solver. The solution achieved might not be an optimal CPMP solution, however the new method allows to estimate its maximum distance from optimality. The computational performance of the new exact algorithm has been evaluated on two classes of test problems proposed in the literature and on two new classes of difficult CPMP instances with additional constraints. The results show that the exact algorithm has been able to solve exactly CPMP's including up to 100 customers.

One of the main objectives of the research was to develop new efficient solution algorithms for routing problems. For this purpose, we have investigated new integer programming formulations for such problems which are based on the two-commodity network flow formulation of the Traveling Salesman Problem (TSP) described by Finke et al. (1984). This formulation is interesting in many different ways. It can be shown that its LP-relaxation satisfies a weak form of the subtour elimination constraints. The formulation can also be modified to accommodate different constraints and, therefore, is capable of being extended to different routing problems. A new two-commodity network flow formulation for the symmetric TSP (STSP) has been derived and extended to derive new integer programming formulations for a class of different routing problems. The VRP has been examined in which a fleet of *M* vehicles stationed at a central depot is to be optimally routed to supply customers with known demands subject to vehicle capacity constraints. We investigated a new integer programming formulation for the VRP and a new lower bound based on the linear relaxation of the two-commodity formulation. The lower bound, strengthened by a set of valid inequalities, has been embedded in a branch and cut procedure to solve the problem optimally. The computational results on a set of problem instances derived from the literature have shown that the lower bound obtained is tight and that the branch and cut algorithm has been able to solve to optimality problems up to 100 customers. The STSP formulation has also been extended to deal with other routing constraints such as delivery and collection constraints. We considered the TSP with mixed deliveries and collections (TSPDC) in which a vehicle located at a central depot must be optimally used to serve a set of customers partitioned in two subsets of delivery and collection customers. The vehicle capacity must not be exceeded along the tour and the total length of the tour must be minimized. A new mathematical formulation has been derived for the TSPDC and another one for the special case, known as TSP with Backhauls, where in any feasible solution all delivery customers must precede the collection customers. New lower bounds have been obtained from the linear relaxation of these formulations which have been further strengthened by valid inequalities and embedded in a branch and cut procedure to solve the problems optimally. The resulting branch and cut procedure has been applied to a set of instances taken from the literature and involving problems up to 150 customers. The results have shown that the branch and cut algorithm has been able to solve to optimality problems involving up to 150 customers.

The computational results of the new two-commodity formulation presented in this thesis have shown the effectiveness of the exact methods derived from this formulation. Future research can focus on the investigation of other valid inequalities for strengthening the lower bound and on the extension of the two-commodity network flow model to other routing problems such as the VRP with Backhauls (VRPB) and the Multi-Depot Vehicle Routing Problem.

Following the success of the exact algorithm developed for the CPMP in providing a sound and efficient solution structure, we have constructed an exact algorithm for the basic VRPB based on a new (0-1) integer programming formulation of the problem. We have presented a method for computing the lower bound by finding a feasible solution of the dual of the LP-relaxation of its integer program. Such a dual solution has been obtained by combining two different bounding procedures where the structure of the one of these bounds is such that additional constraints found in real-world VRPB's can be considered. The exact method uses the dual solution and a method for limiting the variables of the integer program so that the resulting problem can be solved by an integer programming solver. The overall bounding procedure proved to be effective, being able to produce a lower bound whose value on average was at least 98.2% of the optimum. The computational results have shown that the proposed method is able to solve exactly VRPB's of size up to 100 customers within the imposed time limit of 25000 seconds.

The computational results of the new exact methods for the CPMP and the VRPB presented in this thesis have shown the effectiveness of a general technique for solving to optimality complex locations and routing problems. Future work can focus on the development of new exact methods for other complex routing problems, such as the VRP with time windows.

Finally, an application of a mathematical model developed for a real-life routing problem has been presented in the thesis. We considered the resource planning problem of a utility company, which provides preventive maintenance services to a set of customers using a fleet of mobile gangs based at some depots. The results of this study illustrate the significant reductions in travel cost that can be achieved by finding new boundaries for the depot service areas and making more effective decisions in daily mobile resource planning. It also shows how the mathematical model can be used as a planning tool to evaluate a number of "what-if" scenarios and the resulting benefits. The main findings of the case-study are summarised as follows:

1. Given the current boundaries of the region served by each depot, effective reorganisation in scheduling visits and routing of gangs can lead to savings in travel cost of the order of 14%.
2. Further reductions of at least 12.5% in the travel cost have been achieved for all depots by dividing the whole territory under study into new depot service areas.

APPENDIX A

THE CPMP AND VRPB TEST PROBLEMS

In this appendix we report the details of the test problems used in the numerical examples of Section 3.5.3 for the CPMP and of Section 5.5.1 for the VRPB.

A.1 CPMP test problem

The test problem CCPX16 is composed of n=100 customers, among which p=10 must be chosen as medians. The capacity Q of each median is equal to 120 and the total demand of customers (i.e. $\sum_{i \in N} q_i$) is equal to 1060. Table A.1 displays the (x,y) coordinates and the demand for each vertex.

A.2 VRPB test problem

The test problem eilA10166 is composed of n=67 Linehaul customers i.e. ($|L|=33$) and m=33 Backhaul customers (i.e. $|B|=33$). The number M of vehicles is 6, each one with a capacity Q=200. The total demand of the Linehaul and Backhaul customers is 1003 and 455, respectively. Therefore, the minimum number of vehicles needed to visit the Linehaul customers and the Backhaul customers is 6 and 3, respectively (i.e. $M_L = 6$ and $M_B = 3$). Table A.2 displays the (x,y) coordinates and the demand for each vertex.

Table A.1. CPMP test problem

	x	*y*	*demand*		*x*	*y*	*demand*
1	6	29	4	51	55	56	9
2	84	85	11	52	38	59	5
3	44	25	7	53	10	58	10
4	27	68	4	54	99	25	18
5	45	43	19	55	68	54	8
6	55	89	16	56	31	11	4
7	64	73	1	57	52	53	12
8	80	85	1	58	58	87	8
9	53	52	10	59	6	98	19
10	29	46	17	60	44	74	14
11	22	22	13	61	97	68	19
12	99	87	16	62	15	20	6
13	51	22	19	63	86	37	20
14	17	49	19	64	84	69	2
15	76	63	3	65	56	6	9
16	94	38	9	66	50	7	13
17	5	23	7	67	51	52	10
18	29	70	12	68	14	20	12
19	38	84	19	69	49	34	6
20	30	69	3	70	21	97	7
21	7	87	6	71	83	69	3
22	37	45	19	72	31	66	9
23	55	60	3	73	81	67	16
24	73	53	7	74	77	55	11
25	16	75	6	75	85	68	20
26	57	96	7	76	57	3	4
27	69	28	8	77	79	49	17
28	53	33	2	78	78	61	4
29	79	29	7	79	27	75	16
30	46	6	4	80	44	78	8
31	41	71	20	81	2	19	10
32	54	87	8	82	44	42	16
33	7	13	19	83	65	65	2
34	30	28	6	84	9	27	13
35	41	76	8	85	17	83	18
36	15	39	6	86	17	65	1
37	36	59	18	87	29	41	10
38	33	47	5	88	78	52	15
39	10	6	7	89	44	69	12
40	51	11	3	90	39	51	20
41	38	99	19	91	76	21	11
42	38	78	8	92	55	44	6
43	94	48	12	93	74	81	19
44	98	25	15	94	58	56	20
45	19	94	3	95	16	10	8
46	64	13	20	96	1	52	8
47	37	69	12	97	49	45	18
48	70	45	11	98	45	77	9
49	73	56	9	99	2	70	19
50	55	10	6	100	44	92	12

Table A.2. VRPB test problem

	x	*y*	*demand*		*x*	*y*	*demand*
1	35	35	0	52	49	42	13
2	41	49	10	53	53	43	14
3	35	17	7	54	57	48	23
4	55	20	19	55	56	37	6
5	15	30	26	56	15	47	16
6	20	50	5	57	14	37	11
7	10	43	9	58	16	22	41
8	30	60	16	59	4	18	35
9	20	65	12	60	26	52	9
10	30	25	23	61	26	35	15
11	15	10	20	62	15	19	1
12	10	20	19	63	22	22	2
13	5	30	2	64	26	27	27
14	15	60	17	65	25	24	20
15	45	65	9	66	25	21	12
16	45	10	18	67	19	21	10
17	55	5	29	68	18	18	17
18	65	20	6	69	55	45	13
19	45	30	17	70	25	30	3
20	41	37	16	71	55	60	16
21	64	42	9	72	50	35	19
22	31	52	27	73	30	5	8
23	35	69	23	74	20	40	12
24	65	55	14	75	45	20	11
25	63	65	8	76	65	35	3
26	20	20	8	77	35	40	16
27	5	5	16	78	40	60	21
28	40	25	9	79	53	52	11
29	42	7	5	80	2	60	5
30	23	3	7	81	60	12	31
31	11	14	18	82	24	12	5
32	2	48	1	83	6	38	16
33	8	56	27	84	13	52	36
34	6	68	30	85	49	58	10
35	47	47	13	86	57	29	18
36	27	43	9	87	32	12	7
37	37	31	14	88	17	34	3
38	63	23	2	89	27	69	10
39	53	12	6	90	49	73	25
40	36	26	18	91	37	47	6
41	21	24	28	92	47	16	25
42	12	24	13	93	49	11	18
43	24	58	19	94	61	52	3
44	15	77	9	95	55	54	26
45	62	77	20	96	11	31	7
46	67	5	25	97	28	18	26
47	56	39	36	98	31	67	3
48	37	56	5	99	18	24	22
49	57	68	15	100	22	27	11
50	44	17	9	101	20	26	9
51	46	13	8				

REFERENCES

AARDAL, K. 1994. Capacitated facility location: separation algorithms and computational experience. *Technical Report, Dept. Of Econometrics*, No. 9480. Tilburg University.

AGARWAL, Y., K. MATHUR, AND H. M. SALKIN. 1989. Set partitioning approach to vehicle routing. *Networks*, **7**, 731-749.

ANILY, S. AND G. MOSHEIOV. 1994. The traveling salesman problem with delivery and backhauls. *Operations Research Letters*, **16**, 11-18.

ANILY, S. 1996. The vehicle-routing problem with delivery and back-haul options. *Naval Research Logistics*, **43**, 415-434.

APPLEGATE, D., R. BIXBY, V. CHVATAL, AND W. 1994. Cook. Special session on TSP. In *15th International Symposium on Mathematical Programming*. University of Michigan, USA.

ARAQUE, J. R. 1990. Solution of a 48-city vehicle routing problem by branch and cut. *Research Memorandum*, **90-19**, Purdue University.

ARAQUE, J. R., L. HALL, AND T. MAGNANTI. 1990. Capacitated trees, capacitated routing and associated polyhedra. Discussion paper 9061, CORE, Louvain La Neuve.

ARAQUE, J. R., G. KUDVA, T. L. MORIN, AND J. F. PEKNY. 1994. A branch-and-cut for vehicle routing problems. *Annals of Operations Research*, **50**,37-59.

AUGERAT, P., J. M. BELENGUER, E. BENAVENT, A. CORBERAN, D. NADDEF, AND G. RINALDI. 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem. Rapport de recherche 1 RR949-M, ARTEMIS-IMAG, Grenoble France.

AUGERAT, P. AND Y. POCHET. 1995. New valid inequalities for the vehicle routing problem. In preparation.

BÄCK, T., F. HOFFMEISTER, AND H. P. SCHWEFEL. 1991. A survey of evolution strategies. *Fourth International Conference on Genetic Algorithms*, Morgan Kauffman.

BALAS, E. 1975. Bivalent programming by implicit enumeration. In *Encyclopaedia of Computer Science and Technology* 2, J. Belzer, A.G. Holzman and A. Kent (eds.), Dekker, New York, 479-494.

BALAS, E. AND P. TOTH. 1985. Branch and bound methods. In *The Traveling Salesman Problem*, E. L. Lawler , J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley and Sons, 361-401.

BALL, M. 1988. Allocation/routing: models and algorithms. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. Assad (eds.), North-Holland, Amsterdam.

BAUSCH, D. O., G. G. BROWN, AND D. RONEN.1995. Consolidating and dispatching truck shipments of Mobil heavy petroleum products. *Interfaces*, **25**, 1-17.

BEASLEY, J. E. 1988. An algorithm for solving large capacitated warehouse location problems. *European Journal of Operational Research*, **33**, 314-325.

BEASLEY, J. E. 1993. Lagrangean relaxation. In *Modern Heuristic Techniques for Combinatorial Problems*, C.R. Reeves (eds.), Blackwell Scientific, 243-303.

BECK, M. P. AND J. M. MULVEY. 1982. Constructing optimal index funds. Rep. EES-82-1. School of Engineering and Applied Science, Princeton University, Princeton, New Jersey.

BELTRANI, E. AND L. BODIN. 1974. Networks and vehicle routing for municipal waste collection. *Networks*, **4**, 65-94.

BIANCO, L., A. MINGOZZI, AND S. RICCIARDELLI. 1994. A set partitioning approach to the multiple depot vehicle scheduling problem. *Optimization Methods and Software*, **3**, 163-194.

BODIN, L.D., B. L. GOLDEN, A. A. ASSAD, AND M. O BALL. 1983. Routing and scheduling of vehicles and crews. The State of the Art. *Computers & Operations Research*, **10**, 69-211.

BODIN, L.D. 1990. Twenty years of routing and scheduling. *Operations Research*, **38**, 571-579.

CAMPOS, V., A. CORBERAN, AND E. MOTA. 1991. Polyhedral results for a vehicle routing problem. *European Journal of Operational Research*, **52**, 75-85.

CASCO, D. O., B. L. GOLDEN, AND E. A. WASIL. 1988. Vehicle routing with backhauls: models, algorithms, and case studies. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 127-147.

CHAO, I-M., B. L. GOLDEN, AND E. A. WASIL. 1995. An improved heuristic for the period vehicle routing problem. *Networks*, **26**, 24-44.

CHRISTOFIDES, N. AND S. EILON. 1969. An algorithm for the vehicle dispatching problem. *Operations Research Quarterly*, **20**, 309-318.

CHRISTOFIDES, N. 1975. Graph theory: an algorithmic approach. Academic Press, London.

CHRISTOFIDES, N., A. MINGOZZI, P. TOTH, AND C. SANDI. 1979a. Combinatorial optimization. John Wiley & Sons, Chichester.

CHRISTOFIDES, N., A. MINGOZZI, AND P. TOTH. 1979b. The vehicle routing problem. In *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth and C. Sandi, (Eds), J. Wiley, 315-338.

CHRISTOFIDES, N., A. MINGOZZI, AND P. TOTH. 1981a. Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxation. *Mathematical Programming*, **10**, 255-280.

CHRISTOFIDES, N., A. MINGOZZI, AND P. TOTH. 1981b. State space relaxation procedures for the computation of bounds to routing problems. *Networks* , **11**, 145-164.

CHRISTOFIDES, N. 1981. Uses of a vehicle routing and scheduling system in strategic distribution planning. *Scand. J. Mat. Admin.*, **7**(2), 39-55.

CHRISTOFIDES, N. AND J. E. BEASLEY. 1983. Extensions to a lagrangean relaxation approach for the capacitated warehouse location problem. *European Journal of Operational Research*, **12**, 19-28.

CHRISTOFIDES, N. AND J. E. BEASLEY. 1984. The period routing problem. *Networks*, **14**, 237-256.

CHRISTOFIDES, N. 1985. Vehicle Routing. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The traveling salesman problem: a guided tour of combinatorial optimization*, 431-448. John Wiley & Sons Ltd., Chichester.

CHRISTOFIDES, N. AND A. MINGOZZI. 1990. Vehicle routing: practical and algorithm aspects. In *LOGISTICS: Where Ends Have to Meet*, C.F.H. van Rijn, Pergamon Press.

CHRISTOFIDES, N. 1994. The Bionomic algorithm. *AIRO'94 Conference*, Savona, Italy.

CHVATAL, V. 1973. Edmons polytopes and weakly Hamiltonian graphs. *Mathematical Programming*, **5**, 29-40.

CLARKE, C. AND J. Q. WRIGHT. 1964. Scheduling of vehicle from a central depot to a number of delivery points. *Operations Research*, **12**, 568-581.

CORDEAU, J-F., M. GENDREAU, AND G. LAPORTE. 1995. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Report CRT-95-76, Centre de Recherche sur les Transports, University of Montreal.

CORNUEJOLS, G., M. L. FISHER, AND G. L. NEMHAUSER. 1977. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Science*, **23**, 789-810.

CORNUEJOLS, G. AND F. HARCHE. 1993. Polyhedral study of the capacitated vehicle routing. *Mathematical Programming*, **60**, 21-52.

CPLEX OPTIMIZATION INC. 1993-1996. Using the cplex callable library and cplex mixed integer library. 930 Tahoe Blvd #802-297, Incline Village, NV 89451, U. S. A.

CROWDER, H. AND M. W. PADBERG. 1980. Solving large-scale symmetric traveling salesman problems to optimality. *Management Science*, **26**, 495-509.

DANTZIG, G. B., D. R. FULKERSON, AND S. M. JOHNSON. 1954. Solution of a large scale traveling salesman problem. *Operations Research*, **2**, 393-410.

DANTZIG, G. B. AND J. H. RAMSER. 1959. The truck dispatching problem. *Management Science*, **6**, 81-91.

DANTZIG, G. B., D. R. FULKERSON, AND S. M. JOHSIN. 1959. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, **7**, 58-66.

DEIF, I. AND L. BODIN. 1984. Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. In *Proceedings of the Babson Conference on Software Uses in Transportation and Logistics Management*, A. Kidder (eds), Babson Park (U.S.A), 75-96.

DIJKSTRA, E. W. 1959. A note on two problems in connection with graphs. *Numer. Math.*, **1**, 269-271.

DROR, M. AND M. BALL. 1987. Inventory/routing: reduction from an annual to a short-period problem. *Naval Research Logistics Q.*, **34**, 891-905.

DUECK, G. 1990. New optimization heuristics, the great deluge algorithm and the record-to-record travel. Technical report, IBM Germany, Heidelberg Scientific Center.

DUHAMEL, C., J.Y. POTVIN, AND J.M. ROUSSEAU. 1994. A tabu search algorithm for the vehicle routing problem with backhauls and time windows. *Transportation Science* (to appear).

EILON, S., C. WATSON-GANDY, AND N. CHRISTOFIDES. 1971. Distribution management: Mathematical Modelling and Practical Analysis. Hafner, New York.

ERLENKOTTER, D. 1978. A dual-based procedure for uncapacitated facility location. *Operations Research*, **26**, 992-1009.

FINKE, G., A. CLAUS, AND E. GUNN. 1984. A two-commodity network flow approach to the traveling salesman problem. *Congress. Numerantium*, **41**, 167-178.

FISCHETTI, M. AND P. TOTH. 1989. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, **37**, 319-328.

FISHER, M. L. 1981. The lagrangean relaxation method for solving integer programming problems. *Management Science*, **27**, 1-18.

FISHER, M. L. AND R. JAIKUMAR. 1981. A generalized assignment heuristic for vehicle routing. *Networks*, **11**, 109-124.

FISHER, M. L. 1994. Optimal solution of vehicle routing problems using minimum *K*-Trees. *Operations Research*, **42** , 626-642.

FISHER, M. L. 1995. Vehicle routing. In *Network Routing*, *Handbooks in Operations Research and Management Science*, M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser (eds.), North-Holland, Amsterdam, **8**, 1-33.

GAREY, M. R. AND D. S. JOHNSON. 1979. Computers and intractability: a guide to the theory of np completeness. W.H. Freeman and Co, San Francisco.

GARFINKEL, R. AND G. NEMHAUSER. 1972. Integer programming. John Wiley & Sons Inc., New York.

GARFINKEL, R. S. 1979. Branch and bound methods for integer programming. In *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (eds.), John Wiley & Sons, Chichester, 1-20.

GAUDIOSO, M. AND G. PALETTA. 1992. A heuristic for the periodic vehicle routing problem. *Transportation Science*, **26**, 86-92.

GÉLINAS, S., M. DESROCHERS, J. DESROSIERS, AND M.M. SOLOMON. 1995. A new branching strategy for the time constrained routing problem with application to backhauling. *Annals of Operations Research*, **61**, 91-110.

GENDREAU, M., A. HERTZ, AND G. LAPORTE. 1996. The travelling salesman problem with backhauls. *Computers & Operations Research*, **23**, 501-508.

GENDREAU, M., G. LAPORTE, AND D. VIGO. 1997. Heuristics for the traveling salesman problem with pickup and delivery. Technical Report DEIS-OR-97-5, University of Bologna, Bologna, Italy.

GENDREAU, M., A. HERTZ, AND G. LAPORTE. 1997. An approximation algorithm for the traveling salesman problem with backhauls. *Operations Research* , **45**, 639-641.

GEOFFRION, A. M. 1974. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, **2**, 82-114.

GILMORE, P. C. AND R. E. GOMORY. 1961. A linear programming approach to the cutting stock problem. *Operations Research*, **9**, 849-859.

GLOVER, F. 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, **8**, 156-166.

GLOVER, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, **13**, 533-549.

GLOVER, F. 1989. Tabu search -- Part I. *ORSA Journal on Computing*, **1**, 190-206.

GLOVER, F. 1990. Tabu search -- Part II. *ORSA Journal on Computing*, **2**, 4-32.

GLOVER, F AND M. LAGUNA. 1993. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*, C. R. Reeves (eds.), Halsted press, New York, 70-150.

GLOVER, F., E. TAILLARD, AND D. DE WERRA. 1993. A user's guide to tabu search. *Annals of Operations Research*, **41**, 3-28.

GLOVER, F. 1997. A template for scatter search and path relinking. To appear in *Lecture Notes in Computer Science*, J.K.Hao, E.Lutton, E.Ronald, M.Schoenauer, D.Snjers (Eds.).

GLOVER F. AND M. LAGUNA. 1997. Tabu search. Kluwer, Norwell, MA.

GOETSCHALCKX, M. AND C. JACOBS-BLECHA. 1989. The vehicle routing problem with backhauls. *European Journal of Operational Research*, **42**, 39-51.

GOETSCHALCKX, M. AND C. JACOBS-BLECHA. 1993. The vehicle routing problem with backhauls: properties and solution algorithms. Technical Report MHRC-TR-88-13, Georgia Institute of Technology.

GOLDBERG, A. V. AND R. E. TARJAN. 1988. A new approach to the maximum flow problem. *Journal of the ACM*, **35**, 921-940.

GOLDBERG, D. E. 1989. Genetic algorithms and Walsh functions: part I, a gentle introduction. *Complex Systems*, **3**, 129-152.

GOLDBERG, D. E. 1989. Genetic algorithms and Walsh functions: part II, deception and its analysis. *Complex Systems*, **3**, 153-171.

GOLDEN, B.L., E. BAKER, J. ALFARO, AND J. SCHAFFER. 1985. The vehicle routing problem with backhauling: two approaches. In *Proceedings of the XXI Annual Meeting of S.E. TIMS* (R.D. Hammesfahr editor), Myrtle Beach (SC, U.S.A.), 90-92.

GOLDEN, B. AND C. SKISCIM. 1986. Using simulated annealing to solve routing and location problems. *Naval Research Logistic Quarterly*, **33**, 261-279.

GOLDEN, B. L. AND A. A. ASSAD. 1986. Perspectives on vehicle routing: exciting new developments. *Operations Research*, **34**, 803-810.

GOLDEN, B. L. AND E. WASIL. 1987. Computerized vehicle routing in the soft drink industry. *Operations Research*, **35**, 6-17.

GOLDEN, B. L. AND A. A. ASSAD. 1988. Vehicle routing: methods and studies. North-Holland, Amsterdam.

GOMORY, R. E. AND T. C. HU. 1961. Multi-terminal network flows. *SIAM Journal on Applied Mathematics*, **9**, 551-570.

GRÖTSCHEL, M. AND M. W. PADBERG. 1979. On the symmetric traveling salesman problem: I and II. *Mathematical Programming*, **16**, 265-280.

GRÖTSCHEL, M. AND M. W. PADBERG. 1985. Polyhedral theory, in: E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, eds., The Traveling Salesman Problem, John Wiley & Sons, Chichester, 251-305.

GRÖTSCHEL, M. AND O. HOLLAND. 1991. Solution of large-scale symmetric traveling salesman problems. *Mathematical Programming*, **51**, 141-202.

HADJICONSTANTINOU, E., N. CHRISTOFIDES, AND A. MINGOZZI. 1995. A new exact algorithm for the vehicle routing problem based on q-paths and K-shortest paths relaxations. *Annals of Operations Research*, **61**, 21-43.

HADJICONSTANTINOU, E. AND R. BALDACCI. 1998. A multi-depot period vehicle routing problem arising in the utilities sector. *Journal of Operations Research Society*, **49**, N°. 12, 1239-1248.

HALSE, K. 1992. Modelling and solving complex vehicle routing problems. Ph.D. Thesis n. 60, IMSOR, The Technical University of Denmark.

HANSEN, P. 1986. The steepest ascent mildest descent heuristic for combinatorial programming. *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.

HANSEN, P. AND B. JAUMARD. 1990. Algorithms for the maximum satisfiability problem. *Computing*, **44**, 279-303.

HANSEN, P., B. JAUMARD, AND E. SANLAVILLE. 1994. Weight constrained minimum sum-of-star clustering, *Gerad Technical Report G-93-38.*

HANSEN, P. AND B. JAUMARD. 1997. Cluster analysis and mathematical programming. *Mathematical Programming*, **79**, 191-215.

HAO, J. AND J. B. ORLIN. 1992. A faster algorithm for finding the minimum cut in a graph. Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms, Orlando, Florida, 165-174.

HARCHE, F. AND G. RINALDI. 1991. Vehicle routing. private communication.

HELD, M. AND R. M. KARP. 1970. The travelling-salesman problem and minimum spanning trees. *Operations Research*, **18**, 1138-1162.

HELD, M. AND R. M. KARP. 1971. The travelling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, **1**, 6-25.

HELD, M., P. WOLFE, AND H. P. CROWDER. 1974. Validation of subgradient optimization. *Mathematical Programming*, **6**, 62-88.

HOLLAND, J. H. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press.

JÜNGER, M., G. REINELT, AND G. RINALDI. 1995. The traveling salesman problem. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds). Network Models, Handbooks in Operations Research and Management Science 7. North- Holland, Amsterdam, 225-330.

KARGER, D. R. 1993. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms, 21-30.

KARGER, D. R. AND C. STEIN. 1993. An $\tilde{O}(n^2)$ algorithm for minimum cuts. Proceedings of the 25th ACM Symposium on the Theory of Computing, San Diego, CA, 757-765.

KARP, R. M. 1972. Reducibility among combinatorial problems. R. E. Miller, J. W. Thatcher (eds.). Complexity of Computer Computations, Plenum Press, New York, 85-103.

KONTOVRADIS, G. AND J. BARD. 1995. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, **7**, 10-23.

KRARUP, J. AND P. M. PRUZAN. 1983. The simple plant location problem: survey and synthesis. *European Journal of Operations Research*, **12**, 36-81.

KRAUS, A., C. JANSSEN, AND A. MCADAMS. 1970. The lock-box location problem, a class of fixed charge transportation problems. *Journal of Bank Research*, **1**, 51-58.

LAND, A. H., AND A. G. DOIG. 1960. An automatic method for solving discrete programming problems. *Econometris*, **28**, 497-520.

LANGEVIN, A., M. DESROCHERS, J. DESROSIERS, S. GÉLINAS AND F. SOUMIS. 1993. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, **23**, 631-640.

LAPORTE, G. AND Y. NOBERT. 1984. Comb inequalities for the vehicle routing problem. *Methods of Operations Research*, **51**, 271-276.

LAPORTE, G., Y. NOBERT, AND M. DESROCHERS. 1985. Optimal routing under capacity and distance restrictions. *Operations Research*, **33**, 1058-1073.

LAPORTE, G. 1992a. The traveling salesman problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, **59**, 231-247.

LAPORTE, G. 1992b. The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, **59**, 345-358.

LAPORTE, G. AND I. H. OSMAN. 1995. Routing problems: a bibliography. *Annals of Operations Research*, **61**, 227-262.

LAPORTE, G. 1997. Vehicle routing. In: Dell'Amico, M., Maffioli, F., Martello, S. (eds). Annotated Bibliographies in Combinatorial Optimization. Wiley, Chichester. Forthcoming.

LAWLER, E. L., J. K. LENSTER, A. H. G. RINNOOY KAN, AND D. B SHMOYS. 1985. The traveling salesman problem. A guided tour of combinatorial optimization. Wiley, Chichester.

LEUNG, J. M. Y. AND T. L. MAGNANTI. 1989. Valid inequalities and facets of the capacitated plant location problem. *Mathematical Programming*, **44**, 271-291.

LEWIS, H. R. AND C. H. PAPADIMITRIOU. 1981. Elements of the theory of computation. Prentice-Hall.

LIN, S. AND B. W. KERNIGHAN. 1973. An effective heuristic algorithm for the travelling salesman problem. *Operations Research*, **21**, 498-516.

LUCENA, A. 1986. Exact solution approaches for the vehicle routing problem. Ph.D. Thesis, Management Science Dept., Imperial College, London.

MACQUEEN, J. B. 1967. Some methods for classification and analysis of multivariate observations. *5th Berkeley Symposium on Mathematics, Statistics and Probability.*

MAGNANTI, T. L. 1981. Combinatorial optimization and vehicle fleet planning: perspectives and prospects. *Networks*, **11**, 179-213.

MANIEZZO, V., A. MINGOZZI, AND R. BALDACCI. 1988. A Bionomic approach to the capacitated p-median problem. *Journal of Heuristics*, **4**, 263-280.

MARSTEN, R.E. AND F. SHEPARDSON. 1981. Exact solution of crew scheduling problems using the set partitioning model: recent successful applications. *Networks*, **11**, 165-177.

MARTELLO, S. AND P. TOTH. 1990. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Chichester.

MINGOZZI, A., N. CHRISTOFIDES, AND E. HADJICONSTANTINOU. 1994. An exact algorithm for the vehicle routing problem based on the set partitioning formulation. Internal report, Department of Mathematics, University of Bologna, Bologna, Italy.

MINGOZZI, A., M. BOSCHETTI, S. RICCIARDELLI, AND L. BIANCO. 1995. A set partitioning approach of the crew scheduling problem. Internal Report Department of Mathematics, University of Bologna, Italy.

MINGOZZI, A., R. BALDACCI, AND S. GIORGI. 1999. An exact method for the vehicle routing problem with backhauls. *Transportation Science* (to appear).

MIRCHANDANI, P.B AND R.L. FRANCIS. 1990. Discrete location theory. John Wiley & Sons, Chichester.

MOSHEIOV, G. 1994. The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research*, **79**, 299-310.

MULVEY, J. M. AND H. L. CROWDER. 1979. Cluster analysis: an application of lagrangian relaxation. *Management Science*, **25**, 329-240.

MULVEY, J. M. AND M. P. BECK. 1984. Solving capacitated clustering problems. *European Journal of Operational Research*, **18**, 339-348.

NAGAMOCHI, H. AND T. IBARAKI. 1992a. A linear time-algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, **7**, 583-596.

NAGAMOCHI, H. AND T. IBARAKI. 1992b. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM Journal on Discrete Mathematics*, **5**, 54-66.

NEEBE, A. W. AND M. R. RAO. 1983. An algorithm for the fixed charge assigning users to sources problem. *Journal of the Operational Research Society*, **34**, 11, 1107-1113.

NEMHAUSER, G. L. AND L. A. WOLSEY. 1988. Integer and combinatorial optimization. John Wiley & Sons, Chichester.

OSMAN, I. H. 1993. Vehicle routing and scheduling: applications, algorithms and developments. Technical report, Institute of Mathematics and Statistics, University of Canterbury.

OSMAN, I. H. AND N. CHRISTOFIDES. 1994. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, **1**, 3, 317-336.

OSMAN, I. H. AND G. LAPORTE. 1996. Methaheuristics. a bibliography. *Annals of Operational Research*, **63**, 513-628.

PADBERG, M. W. AND M. GRÖTSCHEL. 1985. Polyhedral computations. In *The Traveling Salesman Problem*, E. L. Lawler , J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds.), John Wiley and Sons, 307-360.

PADBERG, M. W. AND G. RINALDI. 1990. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming*, **47**, 19-36.

PADBERG, M. W. AND G. RINALDI. 1991. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, **33**, 60-100.

PAPADIMITRIOU, C. D. AND K. STEIGLITZ. 1982. Combinatorial optimizations: algorithms and complexity. Prentice-Hall, Englewood Cliffs, New York.

PIRKUL, H. 1987. Efficient algorithms for the capacitated concentrator location problem. *Computers and Operations Research*, **14**, 3, 197-208.

RAFT, O. M. 1982. A modular algorithm for an extended vehicle scheduling problem. *European Journal of Operational Research*, **11**, 67-76.

RECHENBERG, I. 1973. Evolutionsstrategie, Fromman-Holzbog.

REEVES, C. R. 1993. Modern heuristic techniques for combinatorial problems. Blackwell Scientific.

REINELT, G. 1991. TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, **3**, 376-384.

ROCKAFELLAR, R. T. 1970. Convex analysis. Princeton University Press, Princeton, New York.

RUSSELL, R. A. AND W. IGO. 1979. An assignment routing problem. *Networks*, **9**, 1-17.

RUSSELL, R. A. AND D. GRIBBIN. 1991. A multiphase approach to the period routing problem. *Networks*, **21**, 747-765.

SANDI, C. 1979. Subgradient optimization. In *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (eds.), John Wiley & Sons, Chichester, 73-91.

SPIELBERG, K. 1979. Enumerative methods in integer programming. *Annals of Discrete Mathematics*, **5**, 139-183.

TAN, C. C. R. AND J. E. BEASLEY. 1984. A heuristic algorithm for the period vehicle routing problem. *Omega*, **12**, 497-504.

THANGIAH, S. R., JEAN-YVES POTVIN, AND TONG SUN. 1996. Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research*, **23**, 1043-1057.

TOTH, P. AND D. VIGO. 1996. A heuristic algorithm for the vehicle routing problem with backhauls. In *Advanced Methods in Transportation Analysis: Proc. Of the Second TRISTAN Conference*, L. Bianco and P. Toth (eds), Springer-Verlag, Berlin, 585-608.

TOTH, P. AND D. VIGO. 1997. An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, **31**, 372-385.

VAN ROY, T.J. 1986. A cross decomposition algorithm for capacitated facility location. *Operations Research*, **34**, 145-163.

YANO, C. A., T. J. CHAN, L. K. RICHTER, T. CUTLER, K. G. MURTY, AND D. MCGETTIGAN. 1987. Vehicle routing at quality stores. *Interfaces*, **17**(2), 52-63.

[29] D.J.Bishop and J.D.Reppy. *Phys.Rev.*, B**22**, 5171, (1980).

[30] G.Agnolet, D.F.McQueeney, and J.D.Reppy. *Phys.Rev.*, B**39**, 8934, (1989).

[31] P.W.Adams and V.Pant. *Phys. Rev. Lett.*, **68**, 2350, (1992).

[32] P.A.Crowell and J.D.Reppy. *Phys. Rev. Lett.*, **70**, 3291, (1993).

[33] P.A.Crowell and J.D.Reppy. *Physica B*, **197**, 269, (1994).

[34] P.A.Crowell and J.D.Reppy. *Phys. Rev.*, B**53**, 2701, (1996).

[35] B.L.Maschhoff and J.P.Cowin. *J. Chem. Phys.*, **101**, 8138, (1994).

[36] Seldon B. Crary. *Ph.D. Thesis*, University of Washington, (1978).

[37] P.Mohandas, C.Lusher, B.Cowan, and J.Saunders. *J.Low Temp.Phys.*, **89**, 613, (1992).

[38] P.Mohandas, C.P.Lusher, B.Cowan, and J.Saunders. *J. Low Temp. Phys.*, **101**, 481, (1995).

[39] E.Cheng, M.W.Cole, W.F.Saam, and J.Treiner. *Phys. Rev. Lett.*, **67**, 1007, (1991).

[40] C.Ebner and W.F.Saam. *Phys. Rev. Lett.*, **38**, 1486, (1977).

[41] J.W.Cahn. *J. Phys. Chem.*, **66**, 3667, (1977).

[42] J.Finn and P.A.Monson. *Phys. Rev.*, A **39**, 6402, (1989).

[43] D.E.Sullivan and M.M.Telo Da Gama. *Fluid Interfacial Phenomena, editor C.A. Croxton*, (Wiley, New York), (1986).