Imperial College of Science and Technology
(University of London)
The Management School

ALGORITHMS FOR ROUTING PROBLEMS IN DISTRIBUTION

by

Kyu-Heon Lee MBA

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the
Diploma of Imperial College

November 1990

ABSTRACT

This thesis is concerned with the development of algorithms for the exact solution to the travelling salesman (TSP) and vehicle routing (VRP) problems. We consider :

(a) The pure TSP, where a salesman based at a given location has to visit a given set of customers and finally return to his base.

(b) The VRP, where a set of vehicles of known capacity based at a depot, have to be routed in order to supply customers with known requirements.

In all cases what is required is to design routes, so that the total 'cost' (i.e. total route length, or time duration, etc.) is minimized.

For each of the above problems we provide :

(i) A formulation based on dynamic programming (DP).

(ii) The relaxation of the DP formulation so that the dimensionality of the state-space is reduced thus making the recursions solvable. The relaxation is based on mapping functions which guarantee that the value of the solution of the relaxed recursion is a lower-bound to the value of the solution of the original recursion.

(iii) A derivation of bounds based on (ii) above with bound ascent procedures from subgradient and state-space ascents.

(iv) The incorporation of the above bounds into tree search algorithms to solve the problems.

It is shown, that although for the TSP the resulting algorithm (although novel) is totally uncompetitive with other existing TSP algorithms ; for the VRP the corresponding algorithm is the best exact solution procedure currently known. Computational results show that VRPs with up to 40 customers can be solved optimally with this method.

Dedicated to

my parents

and

my wife and family

who will be glad.

ACKNOWLEDGEMENT

I would particularly like to thank my supervisor Professor Nicos Christofides, of the Management School, Imperial College, for providing me with the opportunity to carry out this research, as well as for his help and advice throughout its course. He was unfailingly generous with his time, his knowledge of combinatorial optimization and his friendship at difficult moments of my research.

In preparing this thesis I have benefited from the help of my government and many people. Many thanks must go to my government and my colleagues of the Korean Army and the Management Science Research Unit of this Department. They have shown friendship and provided the pleasant environment in which my research has been carried out.

This thesis is dedicated to my wife Nam-Hee, who gave me all the help for the completion of my research. I would also like to express my deepest gratitude to my mother, my parents in law, my brothers and sisters for the physical and psychological support. I hope they know that without them the completion of this thesis would not have been possible.

TABLE OF CONTENTS

CHAPTER THREE : THE TSP AND STATE-SPACE RELAXATION

CHAPTER FOUR : BOUNDS FOR THE VRP FROM STATE-SPACE RELAXATION

CHAPTER 1

INTRODUCTION

1.1 An overview of the Travelling Salesman (TSP) and the Vehicle Routing Problems (VRP)

We consider a problem in which a set of geographically dispersed 'customers' with known requirements must be served on routes operated by a fleet of 'vehicles' stationed at a central facility (depot) in such a way as to minimize some distribution objective. It is assumed that all vehicle routes must start and finish at the depot.

The vehicle routing problem (VRP) is a generic name given to a whole class of problems involving the visiting of 'customers' by 'vehicles'. The VRP (also known in the literature as the 'vehicle scheduling', Clarke & Wright [1964], Eilon, Watson-Gandy & Christofides [1971] and Gaskel [1967], 'vehicle dispatching' Christofides & Eilon [1969], Dantzig & Ramser [1959], Gillet & Miller [1974] and Pierce [1970], or simply as the 'delivery' problem Balinski & Quandt [1964], Hays [1967] and Tillman & Cochran [1969]) appears very frequently in practical situations not directly related to the physical delivery of commodities. For example, the collection of mail from mail boxes or coins from telephone boxes, the pickup of children by school buses, house-call tour by a doctor, preventive maintenance inspection tours and the delivery of laundry, etc. are all VRP's in which the 'delivery' operation may be a collection, collection and/or delivery, or neither ; and in which the 'commodities' and 'vehicles' can take a variety of forms, some of which may not even be of a physical nature. In view of the enormous number of practical situations which give rise to VRP's, it is not surprising to find that an equally large number of constraints and/or objectives appear in such problems. Also, because it is very hard to formulate and solve such problems, one can only hope to study the basic problem which is at the core of all vehicle routing problems. We will call this core problem the basic VRP.

The vehicle routing problem defined above is a generalization of the travelling salesman problem (TSP). In the TSP just one vehicle is required to visit all the customers and return to the depot. Although for this latter problem exact methods of solution have been developed which can solve problems of a few hundred customers (Christofides [1979] and Waters & Brodie [1987]), for the VRP no such algorithms exist. In fact, the largest size general VRP's reported solved optimally in the literature involve problems with 10 or 12 customers, Gillet & Miller [1974], 25 customers, Christofides *et al.* [1981 a] or 40 customers (with special conditions), Christofides & Lucena [1986]. VRPs with several hundred customers have been solved by approximate heuristic methods.

Nevertheless vehicle routing problems have received a considerable amount of attention in both theory and practice with many approaches, both exact and heuristic, being put forward for their solution. In the last decade, enormous advances have been made in the field of vehicle routing, especially in the heuristic solution of practical problems due to advances in both algorithm development and computer capability. In fact, the vehicle routing problem, an area of both research and practice, stands out as one of the great success stories of operational research. Innovative algorithmic research has played a major role in aiding the cost-effective movement of goods and delivery of products within a wide variety of firms and organizations.

If vehicle routing does constitute a major success story, a share in this success must be contributed to effective modeling and implementation.

From the standpoint of the underlying methodologies of mathematical programming and combinatorial optimization, one could argue that existing algorithms for the vehicle routing problem are no more technically involved or sophisticated than, say, solution techniques for the classical travelling salesman problem. However, the major advance in the vehicle routing problem has been to capture enough characteristics of the real-world distribution environment to enable the solution procedures to obtain a useful answer, without thereby precluding their computational tractability. In most successful applications, this desirable state of affairs has resulted from a combination of careful modeling, the design of clever heuristics, and an appropriate interactive user interface.

A number of useful surveys in this field include Golden & Assad [1986 a], Bodin *et al.* [1983], Bott & Ballow [1986], and Christofides [1985 a & 1985 b]. A full survey of modeling and implementaion in routing problems would take us beyond the contents and scope of this thesis where we discuss only the vehicle routing problem, and do not enter into a discussion of other problems of transportation such as crew scheduling, ship scheduling, or rail transport where routing plays an important part.

Ultimately, the focus of our discussion and research is based on the objective of developing algorithms to minimize vehicle routing-related costs (travel time or distances etc.) and to solve problems closer in size to real-world problems than is possible at present (within a reasonable computational time).

1.2 Outline of the thesis

This thesis is mainly concerned with the TSP and VRP using vehicles of uniform capacity. Emphasis is given on procedures that guarantee optimal solutions for these problems by using dynamic programming, state space relaxation and tree search methods.

In Chapter 2 a survey of the VRP is described and the various approaches from the literature, both exact and heuristic, to solve the VRP are introduced. A survey of the TSP is not given since many such surveys can be found in the literature (La$\overset{\ell}{\wedge}$wer *et al.* [1985]) and the problem is well known.

In Chapter 3 the basic concept of the dynamic programming formulation and state space relaxation for the TSP are introduced. Three lower bounds are derived, one (B1) directly from the relaxed DP recursions, one (B2) from "through-circuits" and one (B3) from "2-paths". Lagrangean relaxation techniques are subsequently used for improving the bounds and for reducing problem size. A subgradient optimization procedure is applied to update the Lagrangean multipliers. The computational results of the two kinds of bounds are presented and compared on a number of randomly generated test problems for the TSP. A tree search algorithm is then developed into which the bounds are imbedded in order to provide an exact algorithm for solving TSP's. Computa-tional results are given for this tree search algorithm.

Chapter 4 introduces an integer programming formulation and a dynamic programming formulation for the vehicle routing problem (VRP) with vehicle capacity constraints only. Then, two kinds of bounds (a direct and an indirect bound) are derived from the state space relaxation of the dynamic programming formulation. Lagrangean relaxation techniques and subgradient procedures are used in order to improve the bounds. The bounds are compared on a number of randomly generated VRP test problems.

In Chapter 5 tree search algorithms for solving the VRP are described. The final VRP algorithm incorporates into the tree search the best lower bound for the VRP from Chapter 4. The branching strategy is based on the building up a partially completed route with an arc one at a time and the reduction of problem size by various considerations. Computational results for problems of up to 40 customers are given. Many of these problems are from the literature but newly generated test problems are also considered.

Finally, Chapter 6 presents conclusions and considers some problems suitable for further research.

CHAPTER 2

A SURVEY OF THE VEHICLE ROUTING PROBLEM

In this chapter we present a classification of the VRP, the definition of the basic VRP, some published exact and approximate algorithms for the VRP, and the features and structure of the more realistic routing models.

2.1 Classification of vehicle routing problems

Recent research in the field of routing problems includes significant advances in problem formulations and in the construction, analysis and implementation of solution procedures. These advances have important implications for future research in routing problems. From a practical standpoint, the effective routing of vehicles can increase productivity in lots of fields of governmental and industrial sectors.

We outline general characteristics that describe any vehicle routing problem. A specific vehicle routing problem can be classified on the basis of these characteristics in rather obvious ways. The utility of this taxonomy is that it can help the analyst to identify the type of problem that he is confronting. If the characteristics define a well-known problem, then existing algorithms can be applied to solve the problem. A more important benefit is to specify the constraints that govern the route configurations. The summary of the classification for the vehicle routing problems is shown in Table 2.1 below, which is an offshoot of earlier efforts by Bodin [1975], Golden [1978], Golden *et al.* [1977], Bodin & Golden [1981], and Assad [1988].

Table 2.1 General Characteristics of the Vehicle Routing Problem.

1. Objective	A. Minimize routing costs (distances or times) incurred. B. Minimize sum of fixed and variable costs. C. Minimize number of vehicles required.	
2. Depot	A. Single depot. B. Multiple depot.	
3. Vehicle	A. Size of fleet	a. Single vehicle. b. Multiple vehicles (more than one vehicle).
	B. Type of fleet (Capacity)	a. Homogeneous case (all vehicles the same). b. Heterogeneous case (not all vehicles the same). c. Compartments or not.
4. Customer (demand)	A. Number of commodities	a. Single commodity. b. Multiple commodities. c. Mixed or not (in compartments).
	B. Operations	a. Pure pickups or pure deliveries. b. Mixed pickups and deliveries. c. Pickups (deliveries) with backhaul option.
	C. Nature of demand	a. Deterministic or stochastic. b. Must deliver all demands or not.
	D. Priority	a. Priority for customer or not.
5. Time constraints	A. Call time specified in advance. B. Time windows on customers or not. C. Time windows on drivers or not.	

2.2 A classification of solution strategies

Most solution strategies for the vehicle routing problem can be classified as one of the following approaches (refer to Bodin & Golden [1981] and Christofides [1985 b]) : (i) savings and insertion, (ii) cluster first - route second, (iii) route first - cluster second, (iv) improvement and exchange, (v) mathematical programming-based, (vi) interactive optimization, or (vii) exact procedures. The first four (i) - (iv) approaches have been used extensively in the past. The other three (v) - (vii) approaches represent relatively recently developed ideas.

(A) Savings and insertion procedures.

Build a solution in such a way that at each step of the procedure (up to and including the penultimate step) a current configuration that is possibly infeasible is compared with an alternative configuration that may also be infeasible. The alternative configuration is one that yields the largest savings in terms of some criterion function, such as, total cost (distances or times) or that inserts least expensively a customer not in the current configuration into the existing route or routes. The procedure eventually concludes with a feasible configuration. Examples of savings and insertion procedures are described in Clarke & Wright [1964], Gaskel [1967], Yellow [1970], Hinson & Mulherkar [1975], Mole & Jameson [1976], Golden [1977], Golden *et al.* [1980], Williams [1982] and Bodin [1983].

(B) Cluster first - route second procedures.

Group or cluster customers' nodes first and then design economical routes over each cluster as a second step. Examples of this idea are given by Gillet & Miller [1974], Russell [1974], Gillet & Johnson [1976], Karp [1977] and Krolak & Nelson [1978].

(C) Route first - cluster second procedures.

Work in the reverse sequence to the one above. First, a large (usually infeasible) route or cycle is constructed which includes all of the customers. Next, the large route is partitioned into a number of smaller, but feasible, routes. Golden *et al.* [1982] provided an algorithm that typified this approach for a heterogeneous fleet size vehicle routing problem.

Newton & Thomas [1974] and Bodin & Berman [1979] used this approach for routing school buses to and from a single school, and Bodin & Kursh [1978 & 1979] utilized this approach for routing street sweepers. See also the works of Stern & Dorr [1979], Beasley [1983], Mole *et al.* [1983] and Haimovich & Rinnooy Kan [1985].

(D) Improvement and exchange procedures.

The procedures (such as the well-known branch exchange heuristic of Lin [1965] and Lin & Kernighan [1973] for the TSP, and extended by Christofides & Eilon [1969] and Russell [1977] for the VRP) always maintain feasibility and strive towards optimality. At each step, one feasible solution is altered to yield another feasible solution with a reduced overall cost. This procedure continues until no additional cost reductions are possible. Bodin & Sexton [1979] modified this approach in order to schedule minibuses for the subscriber dial-a-ride problem. The well-known procedures using this concept are the 2-opt and 3-opt algorithms. Baker and Schaffer [1986] have conducted a computational study of the 2-opt and 3-opt algorithms applied to heuristically generated initial solutions.

(E) Mathematical programming based heuristics.

These procedures include algorithms that are directly based on a mathematical programming formulation of the underlying routing problem, and can be partitioned into two categories, i.e. (i) generalized assignment and (ii) set partitioning and covering.

(i) An excellent example of generalized assignment-based procedures is given in Fisher & Jaikumar [1978] in which two interrelated components are identified. One component is a TSP and the other is a generalized assignment problem. Their heuristic attempts to take advantage of the fact that these two problems have been studied extensively and powerful mathematical programming approches for their solution have already been devised. Other examples are described in Fisher & Jaikumar [1981], Gavish & Shlifer [1979] and Van Leeuwen & Volganant [1983].

(ii) Balinski & Quandt [1964] give a set covering formulation of the VRP, where variables correspond to the (enumerated) routes. In Cullen *et al.* [1981], a man-machine interactive approach is used for solving a class of routing problems including the vehicle routing problem and the dial-a-ride problem. A set partitioning model forms the basis of the approach, together with a pricing mechanism for generating new routes. The implementation on a colour graphics terminal has produced good results on standard test problems. Forster & Ryan [1976] formulate the vehicle routing problem as a set covering problem and a column generation procedure is suggested, together with heuristic variations to enable reasonable-size problems to be solved.

(F) Interactive optimization.

This is a general-purpose approach in which a high degree of human interaction is incorporated into the problem-solving process. The idea is that the experienced decision-maker should have the capability of setting and revising parameters and injecting subjective assessments based on knowledge and intuition into the optimization model. This almost always increases the likelihood that the model will eventually be implemented and used. Some early adaptations of this approach to the vehicle routing problem are presented by Krolak *et al.* [1971 & 1972]. The paper by Cullen *et al.* [1981] introduces several rather novel interactive optimization heuristics.

(G) Exact procedures.

These procedures for solving the vehicle routing problem include specialized branch and bound and cutting plane algorithms. Some of the more effective exact approaches are described by Held & Karp [1970 & 1971], Crowder & Padaberg [1980], Christofides *et al.* [1981 a], and in the PhD thesis of Lucena [1986]. These procedures are discussed in greater detail in the following sections.

2.3 The basic vehicle routing problem

In view of the enormous number of practical situations which give rise to vehicle routing problems, it is worthwhile to extract a basic VRP which forms the core to these problems, and to study this basic VRP.

2.3.1 Definition

The basic VRP is defined as follows. We consider the VRP for a given graph G = (X, A) which is defined by the set X of its vertices and the set A of its arcs. Let X = $\{ x_i \mid i = 1, \dots , n \}$ be a set of n vertices (depot and customers), i.e. customers are indexed i = 2, ... , n and i = 1 refers to the depot. A set V = $\{ v_k \mid k = 1, \dots , m \}$ vehicles available at the depot is given, i.e. the vehicles are indexed k = 1, ... , m.

A customer i has a demand (requirements of commodity) of q_i. The travel cost between customers i and j is c_{ij}, which can be taken to be either travel distances or travel times between customers. The capacity of vehicle k is Q_k. We will assume that all customers and vehicles are ordered in descending order of q_i and Q_k respectively.

The basic VRP is to route the vehicles (one route per vehicle, starting and finishing at the depot), so that all customers are supplied with their requirements and the total travel cost is minimized. Fig. 2.1 shows the shape of the solution to a VRP.

The basic VRP ignores a large number and variety of additional constraints and extensions that are often found in real-world problems. Some of these constraints and extensions are described in IBM [1970] and Christofides *et al.* [1982] as :

(i) Each vehicle can operate more than one route, provided the total time spent on these routes is less than a given time T (which is related to the operating time period). Note that such a constraint - in common with many of the ones listed below - requires the knowledge of travel times (t_{ij}) between every pair of customers.

(ii) Each customer must be visited only at a time that lies in one of a given number of working time windows during the period.

(iii) The problem may involve both deliveries to and collections from customers. In addition, it may be possible to mix deliveries and collections on a single route, or alternatively, it may be required for a vehicle to first perform all the deliveries in the route

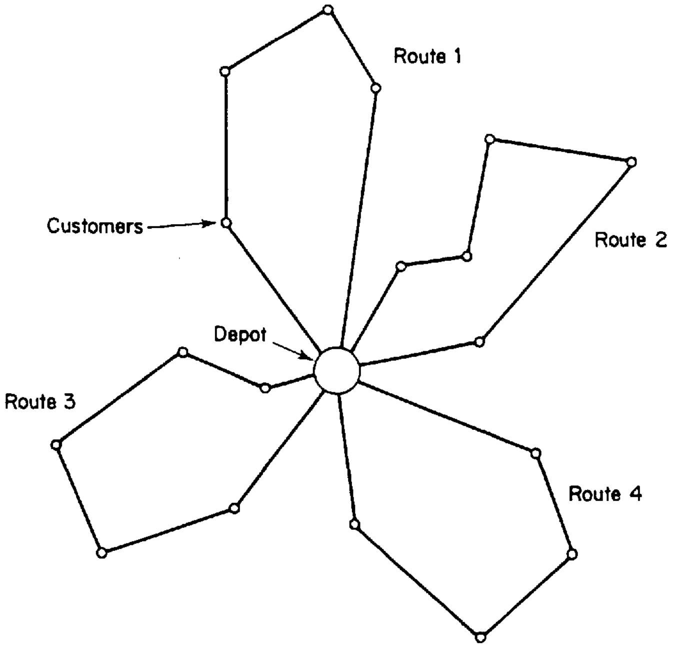


Figure 2.1 Shape of solution to the basic VRP

before performing the collections. This latter case is often referred to as backhauling.

(iv) Just as in (ii) above (every customer has working time windows), vehicles (in fact their drivers) may also have working time windows during the period. The vehicle can then only operate during the specified time windows.

(v) Time-consuming activities other than the travel times (t_{ij}) mentioned above must be also considered. These include : unloading times (or loading times for the case of collections) at the customer premises ; loading times of the vehicles at the depot - both for the first and for any subsequent routes (see (i)) ; queueing times of vehicles for loading at the depot if the number of available loading bays is limited ; etc.

Although the constraints and extensions listed above are only a small fraction of those found in practice (see the classification of the VRP in the previous section), they do not change the essential nature of the basic VRP and can be incorporated in a number of heuristic methods for solving the problem. On the other hand, there are some other practical considerations that also arise frequently, and which do not fit neatly in to the basic

VRP framework, Christofides [1985 b].

2.3.2 Formulation of the basic VRP

Here we introduce some formulations for the basic VRP. However, the purpose of this section not simply to give a comprehensive review of VRP formulations, which are many and varied ; Golden [1976] and Gavish & Srikanth [1979], but to present some formulations which have been used as a basis for solution methods. The formulations of this section include integer programming, set partitioning and dynamic programming. A formulation for the basic VRP with more general objective is described in Christofides *et al.* [1979 c].

(A) Formulation 1 (related to the TSP)

A formulation of the VRP was first given by Golden [1975] as an integer program which is closely related to the TSP. A slightly different formulation is given in Christofides *et al.* [1979 c]. A simplified formulation of the VRP is given below as an integer program.

Let

$$\xi_{ijk} = \begin{cases} 1, & \text{if vehicle k visits customer } x_j \text{ immediately} \\ & \text{after visiting customer } x_i, \\ 0, & \text{otherwise.} \end{cases}$$

The basic VRP is then :

$$\text{Min} \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} \sum_{k=1}^{m} \xi_{ijk}) \tag{1}$$

subject to

$$\sum_{i=1}^{n} \sum_{k=1}^{m} \xi_{ijk} = 1, \qquad j = 1, \dots, n \tag{2}$$

$$\sum_{i=1}^{n} \xi_{ipk} - \sum_{j=1}^{n} \xi_{pjk} = 0, \qquad k = 1, \dots, m, \quad p = 1, \dots, n \tag{3}$$

$$\sum_{i=1}^{n} (q_i \sum_{j=1}^{n} \xi_{ijk}) \leq Q, \qquad k = 1, \dots, m \tag{4}$$

$$\sum_{j=2}^{n} \xi_{1jk} = 1, \qquad k = 1, \dots, m \tag{5}$$

$$y_i - y_j + n \sum_{k=1}^{m} \xi_{ijk} \leq n - 1, \qquad i \neq j = 1, \dots, n \tag{6}$$

$$\xi_{ijk} \in \{ 0, 1 \} \quad \text{for all i, j, k} \tag{7}$$

$$y_i \text{ ; arbitrary}$$

where Q is the capacity (constant) of a vehicle.

Expression (2) states that a customer must be visited exactly once. Expression (3) means that if a vehicle visits a customer, it must also depart from it. Expression (4) is the capacity limitation on each route. Expression (5) states that a vehicle must be used exactly once. Expression (6) is the subtour-elimination condition derived for the travelling salesman problem by Miller *et al.* [1960], and which also forces each route to pass through the depot. Expression (7) are the integrality conditions.

(B) Formulation 2 (Fisher & Jaikumar [1978 & 1981])

This formulation is similar to that of formulation 1 and is also based on integer program.

Let

$$\xi_{ijk} = \begin{cases} 1, & \text{if vehicle k visits customer } x_j \text{ immediately after customer } x_i, \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{if customer } x_i \text{ is visited by vehicle k,} \\ 0, & \text{otherwise.} \end{cases}$$

The basic VRP is then to minimize

$$z = \sum_{i,j} c_{ij} \sum_{k} \xi_{ijk} \tag{8}$$

subject to

$$\sum_{k} y_{ik} = \begin{cases} 1, & i = 2, \dots, n \\ m, & i = 1 \end{cases} \tag{9}$$

$$\sum_{i} q_i y_{ij} \le Q, \tag{10}$$

$$\sum_{j} \xi_{ijk} = \sum_{j} \xi_{jik} = y_{ik}, \quad i = 1, \dots, n, \quad k = 1, \dots, m \tag{11}$$

$$\sum_{i,j \in S} \xi_{ijk} \le |S| - 1, \text{ for all } S \subseteq \{ 2, \dots, n \}, \ k = 1, \dots, m \tag{12}$$

$$y_{ik} \in \{ 0, 1 \}, \quad i = 1, \dots, n, \quad k = 1, \dots, m \tag{13a}$$

$$\xi_{ijk} \in \{ 0, 1 \}, \quad i = 1, \dots, n, \quad k = 1, \dots, m \tag{13b}$$

Constraints (9) ensure that every customer is allocated to some vehicle (except for the depot which is visited by all vehicles), constraints (10) are the vehicle capacity constraints, constraints (11) ensure that a vehicle which visits a customer also leaves that customer, and constraints (12) are the usual subtour elimination constraints for the TSP.

(C) **Formulation 3** (Christofides *et al.* [1981 a])

Let all optimal feasible single routes for vehicle 1 in the VRP be indexed r = 1, ... , $\hat{r}$. Let the index set of customers in route r be M_r and the cost of the route (i.e. the cost of the optimal TSP solution through the customers of the route) be d_r. $N_i = \{ r \mid i \in M_r \}$ will be used.

Let

$$y_r = \begin{cases} 1, & \text{if route r is in the optimal VRP solution,} \\ 0, & \text{otherwise.} \end{cases}$$

The VRP is then to minimize

$$z = \sum_{r=1}^{\hat{r}} d_r y_r \tag{14}$$

subject to

$$\sum_{r \in N_i} y_r = 1, \qquad i = 2, \dots, n \tag{15}$$

$$\sum_{r=1}^{\hat{r}} y_r = m \tag{16}$$

$$y_r \in \{ 0, 1 \}, \qquad r = 1, \dots, \hat{r} \tag{17}$$

Constraints (15) ensure that every customer is visited, and constraints (16) ensure that m routes are chosen for the solution. This problem defined by (15) to (17) is a set partitioning problem with an additional constraint. The problem defined in Christofides *et al.* [1981 a] is more general since it also deals with non-uniform vehicle capacities.

(D) Formulation 4 (Christofides *et al.* [1981 b])

We will now give a dynamic programming formulation of the basic VRP.

Let $X' = \{2, \dots, n\}$ be the set of customers. For any $T \subseteq X'$, let $f(k, T)$ be the minimum cost of supplying the customers in T using only vehicles $1, \dots, k$, let $v(T)$ be the minimum cost of a solution to the TSP defined by the depot and the customers in T, and let $q(T) = \sum_{i \in T} q_i$. The dynamic programming recursion is initialized for k = 1 by f(1,T) = v(T) and defined for $k \geq 2$ by

$$f(k, T) = \min_{S \subset T} [f(k\text{-}1, T\text{-}S) + v(S)] \tag{18}$$

subject to

$$q(T) - (k\text{-}1)\cdot Q \leq q(S) \leq Q \tag{19}$$

$$\frac{1}{m-k} q(X'-T) \leq q(S) \leq \frac{1}{k} q(T) \tag{20}$$

Here, k = 2, ... , m, except for the left-hand side of (20) for which $k \neq m$. The set $T \subseteq X$ to be considered must satisfy

$$q(X') - (m\text{-}k)\cdot Q \leq q(T) \leq k\cdot Q \tag{21}$$

The restrictions on S and T are so as to avoid computing f(.) and v(.) for sets that can only lead to load-infeasible completions. The right-hand side of (19) is the capacity restriction on vehicle k, whereas the left-hand side of (19) is a capacity restriction on the first k-1 vehicles. We have imposed an (arbitrary) order on the routes so that a route with greater load is operated by a vehicle of smaller index than another route with smaller load, i.e. routes are generated in decreasing order of load. Constraints (20) partly imposes this ordering by insisting that the load on route k is greater than the average load on the remaining m-k routes, and less than the average load on the first k-1 routes.

2.4 Exact algorithms for the basic VRP

The exact algorithms for solving the vehicle routing problem are based on the formulations described in the previous section. As with any combinatorial optimization problem, their success or failure is entirely dependent on the degree to which they exploit problem structure. We present here an approach based on Benders decomposition using formulation 2, a branch and bound algorithm using bounds obtained from relaxations of formulation 3 and from state-space relaxation of the recursion of formulation 4. Many exact algoritms are described in Laporte & Nobert [1987], and more details about the solution procedures of these algorithms are described in Christofides *et al.* [1981 a, 1981 b & 1985 a]. In the following chapters we will show the detailed procedures for obtaining bounds for the TSP and the VRP. Before describing these algorithms we will first introduce a well-solved case of the VRP.

2.4.1 A well-solved case of the VRP

Consider a VRP for which $Q_1= Q_2= \dots Q_m= Q$ and with $q_n+ q_{n-1}+ q_{n-2} > Q$, where q_i is the demand of customer x_i and the q_i are assumed ordered in ascending order. For such a problem, all routes contain one or at most two customers only. Form a graph $G= (X', E)$ with a set of vertices $X' = \{x_2, \dots , x_n\}$ and a set of arcs $E = \{\{x_i, x_j\} \mid x_i, x_j \in X', q_i+ q_j\leq Q\}$. Set the cost of arc $\{x_i, x_j\}$ equal to $c_{1i} + c_{ij} + c_{j1}$ and set a penalty p_i of vertex x_i to $2c_{1i}$. The solution of the generalized matching problem on a graph is to find a matching such that the sum of the costs of the arcs in the matching plus the sum of penalties of the vertices that are unmatched is minimum (see Christofides & Thornton [1982]). In the graph G, a vertex x_i left unmatched is interpreted as a route (x_1, x_i, x_1). Note that if s arcs are in the matching then there are n-s routes in the VRP. Thus, if it is required to have exactly m routes, s must be set to be (n-m). Setting the cardinality of a matching does not lead to any additional computational problems.

It has been assumed here that the travel cost matrix is symmetric. Generalization to the asymmetric case is straightforward (Thornton-PhD thesis [1989]).

2.4.2 An algorithm based on Benders decomposition

In formulation 2, a generalized assignment problem is defined by constraints (9), (10) and (13a), and a TSP (in fact, m independent TSPs) is defined by constraints (11), (12) and (13b). Formulation 2 can then be rewritten to bring out this structure, as the nonlinear generalized assignment problem of minimizing

$$\sum_k f_k(y_k) \qquad (22)$$

subject to

constraints (9), (10) and (13a),

where y_k is written for the vector (y_{1k}, y_{2k}, ... , y_{nk}) and $f_k(y_k)$ is the cost of an optimal solution to the TSP defined by the customer set $\{ i \mid y_{ik} = 1 \}$ and the depot, for a given value of k. This function is given by

$$f_k(y_k) = \min \left[\sum_{i,j} c_{ij} \, y_{ijk} \right] \qquad (23)$$

subject to

constraints (11), (12) and (13b).

Obviously, $f_k(y_k)$ is a very complicated function which cannot be written down explicitly. One possible approach is to construct (iteratively) a piece-wise linear approximation of $f_k(y_k)$ by applying Benders decomposition. Each time the generalized assignment problem - defined by (22), (9), (10) and (13a) with some approximation for $f_k(y_k)$ - is solved to obtain y_k, a lower linear support of $f_k(y_k)$ is constructed. This support is derived by solving the m independent TSPs implied by (23), (11), (12) and (13b) for the given y_k and using the dual variables thus obtained. The Benders inequalities describing this lower support are then added to constraints (9), (10) and (13a) to form an extended generalized assignment problem. This problem is now resolved to obtain a new improved y_k, which in turn leads to new TSPs, whose solution provides further Benders inequalities, and so on.

The procedure terminates when the value of the solution to the extended generalized assignment problem (which provides a lower bound to the value of the VRP) coinsides with the sum of the values of the solutions to the TSPs (which provides an upper bound).

Although the overall picture painted above is very much that of a general Benders decomposition, a number of points have to be made.

(A) The TSP subproblems

Since the TSP subproblems defined by (23), (11), (12) and (13b) are integer programs, dual variables cannot be obtained directly. This complication can be removed by replacing constraints (13b) with their linear counterpart

$$0 \leq \xi_{ijk} \leq 1, \qquad \text{for all i, j, k}$$

together with as many linear inequalities of the form

$$\alpha_k x + \beta_k y_k \leq \gamma_k \tag{24}$$

as necessary to ensure that x is naturally integer for any integer y.

Clearly, both the constraint sets (12) and (24) are very large and are best generated as and when required. Fisher & Jaikumar [1978] used Gomory cutting planes to impose integrality on the x, taking care that the constraints of type (24) produced by these cutting planes are valid for all y_k. Constraints (12) are generated as required in the standard way as for any TSP.

(B) The generalized assignment master problem

The generalized assignment problem defined by (22), (9), (10) and (13a) is extended - at some arbitrary iteration - by the addition of the Benders constraints. This problem can be solved to optimality (although this is clearly not necessary at every iteration) by using a branch and bound algorithm using bounds obtained from the Lagrangean relaxation of constraints (2) and the Benders constraints.

2.4.3 An algorithm based on set partitioning

The problem defined by (14) to (17) in formulation 3 is a set partitioning problem with simple additional constraints. Any of the algorithms developed for solving set covering or set partitioning problem (Marston [1974], Balas & Padaberg [1976] and Christofides & Paixao [1982]) could be adapted to deal with the above problem.

The method starts by assuming that the totality of routes which a single vehicle can operate feasibly can be generated. Thus, if $T \subseteq X'$ is a subset of the customers which can be supplied feasibly on a single route by a vehicle, then it is assumed that the total variable cost associated with the optimal way of routing the customers in T can be calculated. Since the problem of routing optimally the customers in T is a TSP, this is not a trivial task if $|T|$ happens to be large.

For a vehicle a family T of all feasible single routes for this vehicle is generated. A matrix $G = [g_{ij}]$ is then produced with row i corresponding to customer x_i and with m blocks of columns. A block of columns corresponds to a vehicle and the column j of this block corresponds to a feasible single route T_j of this vehicle. Let $g_{ij}= 1$ or 0 depending on whether customer x_i is an element of T_j or not respectively, and let $c(T_j)$ be the cost associated with the operation of this route by a vehicle.

The VRP then becomes the problem of choosing at most one column from each block of G so that every row of G has an entry of 1 under exactly one of the chosen columns, and the total cost of columns chosen is minimized. The problem can be easily modified to become a set partitioning problem and the set of columns in the solution contains the optimal routes in the VRP.

However, a basic weakness with the approach is the need to enumerate all routes T_m. Even for very moderate size problems - other than for cases where there are only one or two customers per route - this route generation step is a formidable task. An advantage of this approach is that as the VRP becomes more and more constrained, the number of routes that must be considered becomes smaller and smaller.

2.4.4 A branch and bound algorithm based on state-space relaxation

This algorithm is described in Christofides [1981 a & 1981 b]. Since we will use and extend this algorithm for the rest of this thesis we will give only a very brief description here and examine it in greater detail in the next chapter.

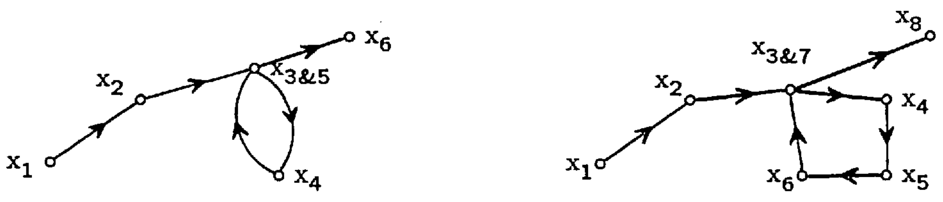
(A) Minimum q-routes

Let W be the set of all possible load (quantities) that could exist on a route operated by a vehicle, i.e.

$$W = \{\, q \mid \sum_i q_i \delta_i = q \le Q,\ \delta_i \in \{\, 0, 1 \,\}\}.$$

Let the elements of W be ordered in ascending order. We will denote by $q(l)$ the value of the lth element of W and by $\lambda(q)$ that l for which $q(l) = q$. If $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ is a path (not necessarily simple), we will call $\sum_{h=1}^{k} q_{i_h}$ the total load on that path. Let $\phi(x_i)$ be the cost of the least cost path from the depot (vertex x_1) to customer x_i with total load $q(l)$. Such a path is called a q-path. It is not easy to impose the condition that no vertex on such a path is visited more than once, but it is simple to impose the less stringent restriction that the path should not contain "loops" formed by three consecutive vertices such as x_{i_α}, x_{i_β}, x_{i_α}. Henceforth when we refer to "loops" we will mean loops of 3 consecutive vertices. Fig. 2.2 shows a path with loops and without. Thus, we will henceforth refer to q-paths and $\phi(x_i)$, implying that these paths are loopless.

Let $\psi_l(x_i)$ be the cost of the least cost route without loops, starting from the depot, passing through customer i and finishing back at the depot with a total load $q(l)$. Such a route will be referred to as a through q-route.



with loops

with no loops

Figure 2.2 A path with loops and without loops.

(B) **Direct bound from state-space relaxation**

Using the dynamic programming formulation of the basic VRP given by expression (18) to (21), we will use a state-space relaxation to compute lower bounds that will subsequently be used in a branch and bound algorithm for solving the VRP. The lower bounds derived in this way are, in general, of excellent quality.

The original state (k, T) appearing in recursion (18) will be relaxed to (k, g(T)) where g is a mapping function from the space of all subsets T to a lower-dimensional space. If we take $g(T) = \sum_{i \in T} q_i \equiv t$ for all $T \subseteq X'$ and similarly $g(S) = \sum_{i \in S} q_i \equiv s$, then the relaxed problem becomes

$$f(k, t) = \min_{s < t} \left[f(k\text{-}1, t\text{-}s) + \overline{v}(s) \right] \tag{18$'$}$$

subject to

$$t - (k\text{-}1)\cdot Q \leq s \leq Q \tag{19$'$}$$

$$\frac{1}{m - k}(q(X') - t) \leq s \leq \frac{1}{k} t \tag{20$'$}$$

$$q(X') - (m\text{-}k)\cdot Q \leq t \leq k\cdot Q \tag{21$'$}$$

where $\overline{v}(s)$ is the minimum cost of a circuit, starting and finishing at depot, with total load s.

A lower bound on $\overline{v}(s)$ is clearly $\min_{x_i} [\phi_l(x_i) + c_{i1}]$, where $l = \lambda(s)$.

After one of the above substitutions is made for $\overline{v}(s)$ in (18$'$), the final value of the recursion, i.e. f(m, q(X$'$)) obtained from (18$'$) to (21$'$), is a lower bound to the VRP.

(C) **Indirect bound from state-space relaxation**

Another bound that can be obtained directly from (18) (and the one recomended by Christofides [1981 a]), is as follows. Recursion (18) implies that the final solution to the

VRP is given by

$$f(m, X') = v(S_1) + v(S_2) + \ldots + v(S_m) \tag{25}$$

for some subsets $S_1, \ldots, S_m$. Consider a subset S and let $l = \lambda(q(S))$. Then $\psi_l(x_i) \leq v(S)$ for any $x_i \in S$ and, in general,

$$\sum_{i \in S} \alpha_i \psi_l(x_i) \leq v(S)$$

for any $\alpha_i \geq 0$ subject to $\sum_{i \in S} \alpha_i = 1$. A choice of α_i which always guarantees the last equality is $\alpha_i = q_i/q(l)$. Thus, an easy lower bound is obtained from (25) as

$$\sum_{i \in X'} \min_{\lambda(q_i) \leq l \leq \lambda(\hat{q})} [\, q_i \psi_l(x_i)/q(l) \,], \tag{26}$$

where $\hat{q}$ is the largest element of W.

The bounds derived above from the state-space relaxation can be improved by penalty methods (using subgradient optimization) in much the same way as bounds derived from Lagrangean relaxation.

(D) Tree - search

There are many branching rules (Garfinkel & Nemhauser [1970] and Balas & Toth [1985], etc.) that can be used in a tree-search scheme using the bounds derived earlier. Possibly the simplest of these involves choosing one as yet unrouted customer to include in or exclude from the currently emerging route. The bound (e.q., that given by expression (26)) can be computed at every node of the branch and bound tree. When a route is completed, the customers in the route are sequenced optimally by solving the corresponding TSP. Note that additional constraints (e.q., delivery time windows) may require the use of a specialized TSP code (Christofides *et al.* [1981 c]).

2.5 Approximate algorithms for the VRP

A great deal of work has been done devising heuristics for the VRP, although much less effort has been spent comparing and drawing conclusions. The possibilities for heuristics are virtually limitless. In this section we will present an outline of some of the best known algorithms and comments on the computational effort.

2.5.1 Criteria for the effectiveness of heuristic algorithms

(i) Quality of solution : in this case, quality is measured in two ways, i.e. the proximity of the objective function value to the optimal value and the ability of the algorithm to generate a feasible solution whenever one exists. A variety of techniques exist for measuring how close is the solution to the optimal value. These include worst case analysis, probabilistic analysis, statistical analysis, characterization of good and bad problems, and a variety of emperical analyses. Many researchers and experts (practitioners) believe that the emperical analysis is the most trusted form of analysis. However, we need to note that there is still a lack of uniformity and no widely accepted guidelines for emperical studies. In particular, there is a definite need for a standard set of easily obtainable test problems.

(ii) Running time : this category applies to all algorithms, not just heuristics. A reasonable running time is a very important element to evaluate algorithms, since implementation of an algorithm is critically dependent on the computing time to solve the vehicle routing problem.

(iii) Difficulty of implementation : two principal difficulties are considered, one is the intricacy of coding, and another is the extent of the data requirement. However, it is difficult to measure these characteristics.

(iv) Flexibility : since heuristics are typically involved in the solution of real world problems it is important that they should be flexible. In particular, they should easily handle changes in the model, constraints and objective function.

(v) **Robustness** : this includes the ability to perform sensitivity analysis and the ability to generate bounds on the solution obtained.

(vi) Simplicity and analyzability : there is significant appeal to algorithms that can be simply stated and that more readily lend themselves to analysis. Extremely complex algorithms are much less likely to be analyzed in terms of flexibility, quality of solution, etc., than a simple algorithm.

(vii) **Interactive computing** : the idea of using man-machine interaction within an algorithm comes up on numerous occasions. It is general opinion that little has been known about this class of algorithms and that other criteria should be developed to evaluate interactive algorithms.

2.5.2 Criteria for route expansion in constructive methods

In the criteria for route expansion, a criterion is considered as a function defined over the customers and which is used to determine which customer should enter the route(s) being constructed and in which position. That customer is chosen (for expanding the route) which optimizes the criterion function. Some of the more often used criteria are as follows :

(i) Savings : the 'saving' of a customer x_l with respect to x_1 (depot) and another customer x_j is given by :

$$s(l, j) = c_{l1} - c_{lj} + c_{1j}$$

$s_1(l, j)$ is the saving in mileage of supplying x_l and x_j together on one route as opposed to supplying them individually directly from the depot, i.e. operating route (x_1, x_l, x_j, x_1) instead of routes (x_1, x_l, x_1) and (x_1, x_j, x_1).

(ii) Extra-mileage : the 'extra-mileage' of an as yet unrouted customer x_l with respect to two consecutive customers x_i and x_j already in an emerging route is given by

$$m(i, l, j) = c_{il} + c_{lj} - c_{ij}$$

(iii) Radial position : the angle $\theta_1(l, j)$ that the ray (x_1, x_l) forms with the ray (x_1, x_j)

for a given routed customer x_j can be used as a criterion function defined over the as yet unrouted customers x_l. Note that this criterion requires customer coordinates to be specified.

(iv) **Composite criteria** : these are composite functions of savings, extra-mileage and radial criteria, and in addition functions of : the quantity q_l to be delivered to an as yet unrouted customer x_l ; the number of other as yet unrouted customers (n_l say) remaining in the 'neighbourhood' of x_l etc. The functions are such that the larger the values of S, 1/m, q, 1/n, etc. are, the larger the criterion value of the customer. The above measures are in most cases specialized to ease computations.

2.5.3 Sequential, parallel and coalescing procedures

In a sequential procedure one route is constructed at a time until all the customers are routed. At no time is the question raised whether a customer x_l should be placed on route R or route S. This consideration is made implicitely by deciding whether to include x_l on route R or not. Such procedures, typically, start a route with a given customer and then expand the route by computing which customer to insert in it next using one of the above evaluation functions.

In a parallel procedure a number of routes is being formed in parallel (fixed a priori to some number, say K). K routes are initiated by choosing K "seed" points to start the routes and expansion of these is then based on the above evaluation functions. At the end of the procedure K routes exist.

In a coalescing procedure a large number of smaller routes (initially routes consisting of one customer only) are coalesced into a smaller number of larger ones until the routes can not be coalesced any more. The number of routes remaining at the end of such an algorithm is not predictable.

2.5.4 The effectiveness of simple criteria

It is quite easy to show that evén for the basic VRP none of the criteria listed above is uniformly better than the others. Consider, for example, a parallel algorithm initialized

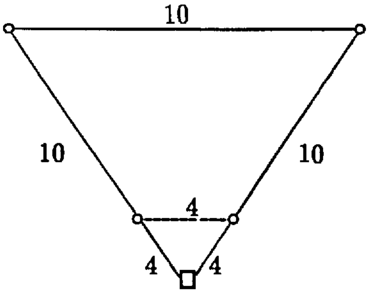


Figure 2.3(a) Savings Total : 42

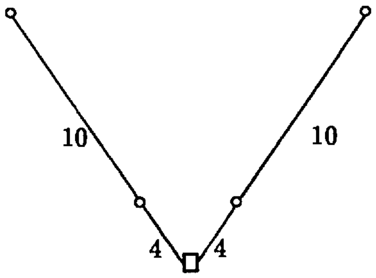


Figure 2.3(b) E. M. Total : 40

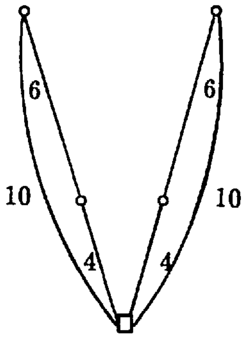


Figure 2.4(a) E. M. Total : 40

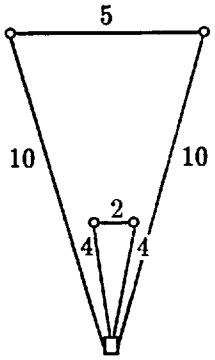


Figure 2.4(b) Savings Total : 35

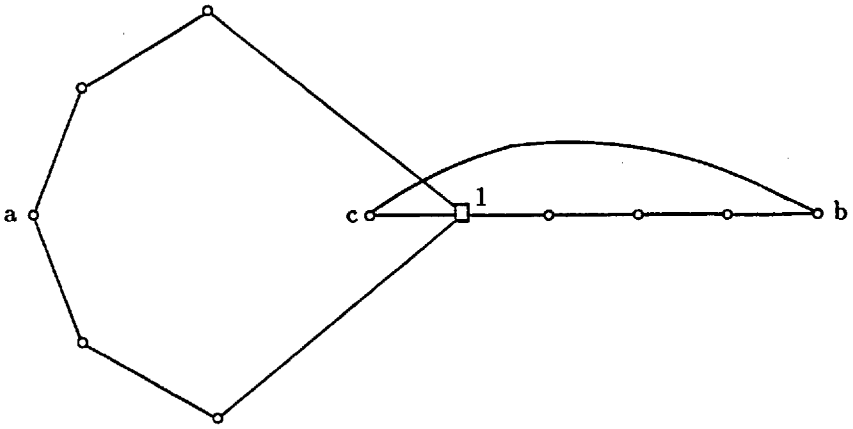


Figure 2.5(a)

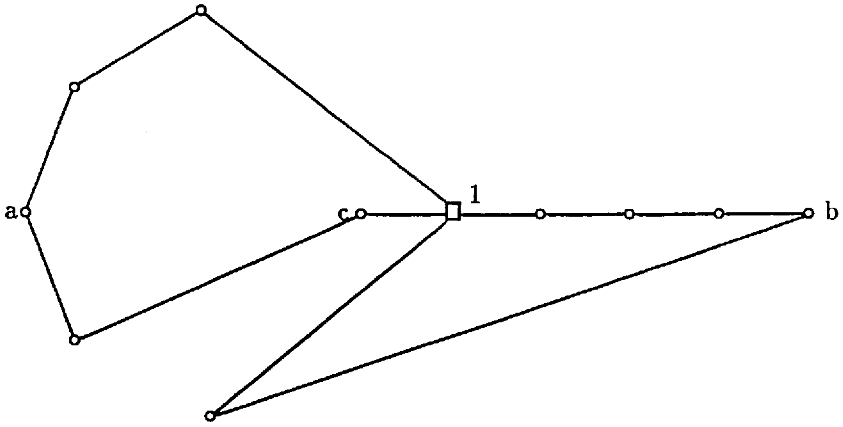


Figure 2.5(b)

with four routes each starting from the depot to a customer and back (it is assumed that a vehicle can take at most 2 customers). In the example in Fig. 2.3, we see the results of the savings and extra-mileage, indicating that the extra-mileage measure is better for this example. For the example in Fig. 2.4 however the comparison is in favour of the savings. Moreover, even for the same example all of the above (Sction 2.5.2) criteria may produce bad solutions. Consider, for example, the problem in Fig. 2.5 where 1 is the depot (x_1) and customer c is on the line (a, 1). We will use a sequential procedure, and assume that a vehicle can take at most 5 customers. If the procedure is initialized with route (x_1, b, x_1) and a savings criterion is used, the solution in Fig. 2.5(b) is obtained. If the procedure is initialized with route (x_1, a, x_1) and the extra-mileage criterion is used, the same solution is obtained. However, a better solution is shown in Fig. 2.5(a) which can be obtained if the procedure is initialized with (x_1, b, x_1) and an extra-mileage criterion is used, or if it is initialized with (x_1, a, x_1) and a savings criterion is employed. This example illustrates the importance of initializing the routes.

2.5.5 Algorithms by constructive methods

(A) The savings algorithm of Clarke & Wright [1964]

This algorithm is one of the earliest ones and is without doubt the most widely known heuristic for the VRP. The algorithm proceeds as follows :

Step 1 : Calculate the saving $s_{ij} = c_{1i} - c_{ij} + c_{j1}$ for all pairs of customers i and j. Note that s_{ij} is the saving in cost that would result if the link (i, j) is made to produce route (x_1, x_i, x_j, x_1) instead of supplying x_i and x_j on two routes (x_1, x_i, x_1) and (x_1, x_j, x_1).

Step 2 : Order the savings in descending order.

Step 3 : Starting at the top of the list, do the following.

(Coalescing version)

Step 4 : If making a given link results in a feasible route according to the constraints of the VRP, then append this link to the solution; if not, reject the link.

Step 5 : Try the next link in the list and repeat step 4 until no more links can be chosen.

(Sequential version)

Step 4 : Find the first feasible link in the list which can be used to extended one of the two ends of the currently constructed route.

Step 5 : If the route can be expanded and remain feasible, make this link, if it cannot be expanded further, terminate the route. Choose the first feasible link in the list to start a new route.

Step 6 : Repeat steps 4 and 5 until no more links can be chosen.

In both the coalescing and sequential versions of this procedure, it is advisable to check the feasibility of the partial solution at every stage, to ensure that the available vehicles can operate the routes being formed. Otherwise, it is quite likely that at the end no feasible solution is found. Also note that the initial starting solution when every customer is on a separate route is infeasible. However, the possibility always exists at the end to leave unrouted some customers on single-customer routes.

Many modified definitions of savings have been proposed to achieve different results (e.q., Gaskell [1967] and Yellow [1970]). In particular, the original Clarke & Wright algorithm produces circumferential routes that were often objected to by schedulers. Golden, Magnanti & Nguyen [1977] substentially reduced the running time of the Clarke & Wright algorithm by sophisticated computer science methods.

(B) The algorithm of Mole & Jameson

Many other consecutive methods exist which use criteria different from savings. We mention in particular the sequential tour building procedure of Mole & Jameson [1976], in which a criterion is used that can change the emphasis from giving preference to circumferential routes, to giving emphasis to radial shaped routes. This criterion contains parameters λ and μ that are user-controlled. For the given value of λ and μ, the following two criteria are used to expand a route under construction.

$$e(i, l, j) = c_{il} + c_{lj} - \mu c_{ij}$$

$$\sigma(i, l, j) = \lambda c_{1l} - e(i, l, j)$$

The algorithm then proceeds as follows :

Step 1 : For each unrouted customer x_l compute the feasible insertion in the emerging route R as :

$$e(i_l, l, j_l) = \min_{\substack{\text{for all adjacent customers} \\ x_r,\ x_s \in R}} [\, e(r, l, s) \,],$$

where x_{i_l} and x_{j_l} are customers between which x_l has the best insertion.

Step 2 : The best customer x_{l^*} to be inserted in the route is computed as the one for which the following expression is maximized.

$$\sigma(i_{l^*}, l^*, j_{l^*}) = \max_{\substack{\text{for } x_l \text{ unrouted} \\ \text{and feasible}}} [\, \sigma(i_l, l, j_l) \,]$$

Step 3 : Insert x_{l^*} in route R between x_{i^*} and x_{j^*}.

Step 4 : Optimize route R using r-optimal (Lin & Kernighan [1973]).

Step 5 : Return to step 1 to start a new route R (see note (a)), either until all customers are routed or no more customers can be routed.

It is easy to see in the above definition of $\sigma(i, l, j)$ and $e(i, l, j)$ that by changing the values of λ and μ it is possible to obtain different criteria to choose the best customer for insertion. Generally, as λ grows the shape of the emerging route tends to be circumferential and as μ grows the presence of long links is discouraged.

Note that the above description explains how a route R is expanded by the addition of customers. Initially, (and each time a new route is to be started), some customer x_s must be chosen to initialize the route R as (x_1, x_s, x_1). Customer x_s may be chosen in a variety of ways, e.q., the furthest unrouted customer, the customer with the largest demand $\overline{q_s}$, the customer with the most stringent delivery time restrictions, etc. Also note that Mole & Jameson describe the procedure for a fleet of identical vehicles. In this case the assignment of a vehicle to an emerging route is trivial except if vehicles are used for second, third etc., trips - in which case the departure times of vehicles from the depot (for this additional trips) will be different for each vehicle and a choice exists as to what vehicle to assign to the current route. More generally, at some stage when a route R is being constructed, different size vehicles with different starting and ending times and different allowable working periods will be available and an assignment of vehicles to routes must be made.

2.5.6 Algorithms by two phase methods

(A) The sweep algorithm of Gillet & Miller [1974 & 1976]

Both the first and second phases of this procedure are of a sequential nature. Assume that the vehicle routing problem is Euclidean and that customers are located by their polar coordinates (r_i, θ_i) with the depot at $r_1 = 0$ and an arbitrary customer i^* at $\theta_{i^*} = 0$. (Other can also be accommodated.) Reorder the customers such that $\theta_2 \leq ... \leq \theta_n$.

Phase I

Step 1 : Choose an unused vehicle k.

Step 2 : Starting from the unrouted customer i with smallest angle θ_i, include consecutive customers i+1, i+2, ... in the route until the capacity constraint of the vehicle k is reached.

Step 3 : If all customers are 'swept' or if all vehicles have been used, go to Phase II, else return to step 1.

Phase II

Step 4 : Solve the travelling salesman problem for every set of customers assigned to a vehicle to form the final routes.

Note that there are a number of possible variations of the sweep algorithm above. Different choices of the 'reference' customer i^* from which to measure the polar coordinate angles, lead to different final routes. The same is true with different rules used to choose the vehicle to consider next.

(B) The algorithm of Christofides, Mingozzi & Toth [1979 c]

The first phase of this heuristic consists of performing a number of clustering trials using a least cost insertion criterion with a user-controlled extra parameter that could produce different solutions in different trials.

Phase I

Step 1 : (Sequential trial). Choose an unrouted customer to be a seed. Choose a vehicle k to allocate to the emerging route.

Step 2 : Enter unrouted customers into the emerging cluster, in increasing order of some insertion cost relative to the seed of the cluster, until the capacity limit of vehicle k is reached. If all customers are clustered, or all vehicles used, go to step 3, else repeat from step 1.

Step 3 : (parallel trial). Using the seeds chosen in the sequential trial, free all customers from their clusters.

Step 4 : For every free customer, compute its insertion cost into a feasible cluster relative to the seed of the cluster. Consider all clusters and keep the best insertion for the customer.

Step 5 : Of the free customers, allocate the one with minimum insertion cost to its corresponding cluster.

Step 6 : Repeat step 4 for any free customer whose previously best insertion is no longer feasible, and continue with step 5 until no further feasible insertions are possible.

Phase II

Step 7 : For both the above two clusterings formed sequentially and in parallel, solve the TSP for each cluster. Keep the best of the two as the VRP solution.

Once again, note that by making use of a user-controlled parameter in the measure of insertion cost, more than two trial clusterings can be produced.

(C) The algorithm of Fisher & Jaikumar [1981]

The first phase of this heuristic performs a parallel clustering by solving optimally a generalized assignment problem.

Phase I

Step 1 : Choose m customers to be seeds of clusters and allocate a vehicle to each.

Step 2 : For each customer i and for each cluster k, compute an insertion cost d_{ik} relative to the seed of the cluster.

Step 3 : Solve the generalized assignment problem min { $\sum_{i,k} d_{ik} y_{ik}$ | expressions (9), (10) and (13a) in the previous section }.

Phase II

Step 4 : Solve the TSP for every set of customers in the clusters implied by the y_{ik}.

2.5.7 Comments

Note that although the last two methods are similar, the latter heuristic solves the clustering phase optimally by using a fast algorithm for the generalized assignment problem (Fisher *et al.* [1979]). Thus, the objective $\sum_{i,k} d_{ik} y_{ik}$ can be considered as an easy-to-compute approximation to the objective in expression (22), and the whole method as a first iteration of the exact method described earlier. It has been noted by several researchers, that none of the above-mentioned heuristics are uniform in their behaviour. In particular, they perform reasonably well when the VRPs are mostly unconstrained, but become progressively worse as more constraints are added.

2.6 The structure of practical vehicle routing problems

We discuss the features that seem to be encountered in real vehicle routing problems referring to Schrage [1981], Christofides [1985 a & 1985 b] and Assad [1988]. We will present these features in six categories referring to the classification of the vehicle routing problems as mentioned in Section 2.1.

2.6.1 Various objectives

The standard objective is to minimize the total distance (or time, etc.) travelled over all routes selected. Actually, there may be some noticeable deviations from this objective. The actual cost/mile may differ on different arcs because of different road conditions or simply because of different rates charged by carriers. Ocasionally, carriers will specify a minimum trip charge and/or a drop charge for each stop. The latter discourages split deliveries. The capability of handling a time-dependent drop charge is useful in time-dependent delivery problems. On the other hand, sometimes, various situations may arise when it is simply infeasible to solve the VRP as given. In practice, this infeasibility is resolved by either (i) hiring more vehicles, and/or (ii) postponing service to some customers beyond the established service level or into the next period. In these cases, the objective may be to minimize (i) the number of extra vehicles hired, and/or (ii) the number of customers not served in the present period, and/or the total distance (or time) travelled. More complex objectives have been utilized in various problem settings in order to capture the flavour of constraints which are difficult to quantify. Genarally, the objective in a vehicle routing problem may be a linear combination of various simpler objectives.

2.6.2 Multiple depots

In companies with more than one depot, it is often the case that each depot is autonomous, with its own fleet of vehicles and its own geographical customer area to serve. In such cases, the company would simply face a number of similar single-depot vehicle routing problems. In other cases, however, depot operations are interdependent and vehicles leaving one depot may, after delivering to customers, end up at another depot, perhaps to load again and continue on a subsequent trip. In these cases each depot cannot be considered in isolation. Bettrami *et al.* [1971] extends the savings algorithm to a routing problem with multiple depots ; Gillet & Johnson [1976] extends the sweep algorithm for the vehicle routing problem to this case of more than one depot ; and Laporte *et al.* [1988] descibes the solving a family of multi-depot vehicle routing problem.

2.6.3 Multiple vehicles and vehicle types

We can consider a fleet consisting of one or more vehicles, and in case of more than one vehicle various vehicle types. It is frequently useful to think of the commodities being transported as having several dimensions, such as weight and volume. For example, in air fleight both weight and volume may play an important role in determining what gets loaded on a given trip. Also multidimensional capacity may mean multicompartment vehicles, such as fuel trucks, which may deliver regular, premium, unleaded etc., fuel, all in one trip.

If an algorithm allows multiple-vehicle types, then one of the vehicle types can correspond to a dummy vehicle and one can thus represent options not to service a particular node or arc based on profitability. The option not to visit an arc or node (in a given period) is especially important in time-dependent routing problems.

2.6.4 Multiple time constraints

The time constraints are classified in two categories, i.e. time windows and the time period. We can find a lot of literature about these fields such as Beltrami & Bodin [1974], Russell & Igo [1979], Raft [1982], Christofides & Beasley [1984], Savelsberg [1985], Golden & Assad [1986 b], Kolen *et al.* [1987], Solomon [1987], Desrochers *et al.* [1988] and Solomon *et al.* [1988], etc.

(A) Time windows

In the routing problem with time windows, the customers requiring service have to be served between certain times. Problems with time windows include snow removal, postal deliveries and bank deliveries. Routes and schedules have to be devised such that the required service is performed during these time windows.

(B) The VRP within a time period

The time period during which the customer requirements must be fulfilled is one of the most important parameters in a vehicle routing problem, and is a measure of the service level. Since customer ordering is a dynamic, non-periodic process, any attempt to define a vehicle routing problem for a given period must, by definition, be an approximation or an arbitrarily imposed order. Some of these approximations are as follows.

(i) Typical period : This is the case when the customers are fixed and their demands are assumed to be typical in a given period. A customer that is expected to order once every t days is required to be visited T/t times during the period of T days, and these visits must be t$\pm$ ϵ days apart, for some small given value of ϵ. The fixed routes that are produced by solving the vehicle routing problem for the period are often made public so that each customer knows when to expect his deliveries. Clearly, problems of feasibility can arise in a real period that is not typical.

(ii) Cut-off time : A frequently used modus operandi is to set a cut-off date for orders. Orders received in the previous T days are delivered in the following T days. The vehicle routing problem for T-day period is then completely specified. However, with such a system, orders received during the current T-day period and which could (or perhaps should) have been delivered in the current period, are ignored until the next T-day period. The result is that infeasibility problems (usually resolved by hiring extra vehicles) may arise in some period.

(iii) Creeping customer priorities : An often used alternative to defining a period, as in (i) or (ii) above, is to allocate a priority to each customer according to the time interval remaining up to the date when the customer must be visited (say T days after receipt of the order). The smaller the time remaining, the higher the customer priority. At any one time the vehicle routing problem would then involve a complex objective of both routing costs and the priorities of the customers that are routed, in an attempt to maintain the customer service within a T-day maximum delay.

(iv) Frequency requirements : In these problems, certain customers have to be covered a specific number of times within a certain time period such as a week. Typical problems of this type are coin collection from parking meters, garbage collection, fuel delivery and sales plan calling. For example, in case of coin collection from parking meters, when a particular heavily used parking meter is emptied is not important, as long as it is emptied every other day say. This is a variation of case (i) above.

(C) Time-dependent travel time

In urban routing problems, travel time may increase dramatically during rush hours, over some bottlenecks, such as bridges and tunnels, implying that the travel time over a route (arc) may depend upon the period in which it occurs.

(D) Design of fixed routes

Fixed routes can be operated unchanged over a given period even though the demand is changing. Christofides [1971] describes this problem, i.e. a set of customer areas and the demand within each area are given for each day of a given period. These routes are required to be feasible for each of the days in the period. Once in an area, a vehicle is assumed to visit all the customers (and supply all the demand) within the area. This problem is similar to the VRP with stochastic demand.

2.6.5 Various demands

(A) Multiple commodities

In some vehicle routing problems, the vehicles are compartmented so that different commodities are stored in segregated compartments. Each customer may require specified quantities of different types of commodity. Such problems appears in the distribution of gasoline fuel, refrigerated (or not) foods, etc. (refer to multiple vehicles in the previous section), and involve - in addition to the routing aspect of the vehicle routing problems - a knapsack or bin-packing problem.

(B) Split deliveries and lumpy cargo

When the requirement of a single customer is large relative to the vehicle capacity, it may be economical to split a customer among several vehicles. For example, suppose vehicle capacity is 8 units, that there is a customer close to the depot with requirements 8 and that there are two distant (from the depot and each other) customers with requirements 4 each. It may be optimal to split the big close customer among the two vehicles making the trips to the distant customers so that each vehicle delivers 4 units to the close customer and 4 units to a distant customer. Without splitting the load, it may be impossible to service the two distant customers with one vehicle because of the travel-time restrictions. Hence, without splitting the load, this problem might require three vehicles rather than two. This problem has been addressed in specific instances by heuristic procedures or by a set covering approach.

When splitting a load is possible, it may be important to take into account the lumpiness or integrality of the cargo. That is, only integral amounts of cargo may be assigned to the vehicles involved in the split.

CHAPTER 3

THE TSP AND STATE-SPACE RELAXATION

3.1 Introduction

We consider a graph G=(X, A) defined by the set X of its vertices, the set A of its arcs and $[c_{ij}]$ the cost matrix for the cost of these arcs. We use $c(x_i, x_j)$ and c_{ij} interchangeably. A typical routing problem on G is the travelling salesman problem (TSP) in which the least cost route passing through every vertex of G is required.

One of the most successful methods of solving routing problems is by use of branch and bound algorithms which are based on bounds, where the effectiveness of the bounds is the most important parameter that determines the efficiency of the complete algorithm. A general methodology for computing bounds is Lagrangean relaxation; see Geofferion [1974] and Fisher [1978], and although it is only one of the several bounding schemes that are possible, it has performed well on many different types of combinatorial problems.

However, when we want to add some additional constraints like time-constraints in the TSP or vehicle capacities and customer's requirement constraints as in the VRP, these constraints tend to destroy whatever structure the original unconstrained problem had. Because of these difficulties, an alternative methodology has been developed in Christofides *et al.* [1979 b] to deal with routing problems. This methodology is based on the two observations that

(i) every routing problem is essentially a shortest path problem on some underlying state graph with additional constraints, and

(ii) dynamic programming is a general procedure for solving shortest path problems subject to constraints, by introducing the constraints into additional state variables in the state vector, and solving an essentially unconstrained shortest path problem on an expanded state-space graph.

Thus, it is quite natural to consider dynamic programming. Consider, for example, the TSP where a shortest route is required passing through every vertex of G once and only once. Let S be a subset of vertices and $f(S, x_i)$ be the least cost of a path starting at vertex x_1, passing through every vertex of S and finishing at vertex $x_i \in S$. A dynamic programming recursion for $f(S, x_i)$ is as follows :

$$f(S, x_i) = \min_{x_j \in S-x_i} [f(S - x_i, x_j) + c(x_j, x_i)] \tag{1}$$

where $S \subseteq X' \equiv X - \{x_1\}$, $\forall x_i \in S$ and the initialization is $f(\{x_i\}, x_i) = c(x_1, x_i)$, $\forall x_i$. The optimum solution to the problem is then given by the expression :

$$\min_{x_i \in X'} [f(X', x_i) + c(x_i, x_1)] \tag{2}$$

Recursion (1) gives a shortest path procedure on the state-space graph whose vertices correspond to the states (S, x_i) and whose arcs represent transitions from one state to another.

It is well-known, see Bellman [1958], that few combinatorial optimization problems can be solved effectively by dynamic programming alone, since the number of vertices of the state-space graph is enormous. Therefore, a general relaxation procedure has been proposed, whereby the state-space associated with a given dynamic programming recursion is relaxed (i.e., the number of states reduced) in such a way that the solution to the relaxed recursion provides a bound (lower bound in the case of minimization , upper bound in the case of maximization) to the value of the true optimum. Such a relaxation could then provide bounds for embedding in general branch and bound algorithms for the solution of the routing problems.

This state-space relaxation is analogous to Lagrangean relaxation in integer programming. Constraints in integer programming formulation appear as state variables in dynamic programming recursions and hence constraint relaxation corresponds to state-space relaxation.

In this chapter, we will not discuss the general principles of state-space relaxation which the reader can find in Christofides *et al.* [1981 b], but will instead concentrate on the application of this procedure to the derivation of bounds for the TSP, and the embedding of these bounds into a tree search algorithm for the solution of TSPs.

3.2 State - space relaxation for the TSP

Consider the dynamic programming formulation of the TSP given by recursion (1) in the previous section. The state variable s in that formulation is (S, x). Let g(.) be a mapping function from the domain of (S, x) to some other smaller vector space (g(S), x). Recursion (1) for the TSP can now be relaxed to the amaller space (g(S), x) and become :

$$f(g(S), x) = \min_{y} [f(g(S - x), y) + c(y, x)] \qquad (3)$$

We wish to choose g(.) is chosen to be a separable function, so that given g(S) and x, g(S-x) can be computed. Also we wish to restrict the minimization to be only over these values of y so that given g(S) and x, we can obtain state (g(S), x) from the state (g(S-x), y).

The initialization is :

$$f(w, y) = c(x_1, y)\ , \quad \text{if } w = g(\{y\})$$
$$= \infty\ , \quad \text{otherwise.}$$

By using the transpose of the cost matrix $[c_{ij}]$ we can define a second 'reverse' function f′(g(S), x) by a recursion exactly analogous to (3). f′(g(S), x) corresponds to a path starting from state (g(X), x_1). For symmetric TSP's, the reverse function f′(.) = f(.). Note that backtracking can produce the solutions corresponding to f(. , .) and f′(. , .).

3.2.1 Forms of the mapping function g(.) for the TSP

We have mentioned in the previous section that we wish g(.) to be any separable function. In this section we will introduce two functions, which we use in this thesis, from a variety of such functions :

(A) g(S) = |S|. (Cardinality relaxation : n-path).

Let k = |S|. We then have $g(S - x_i) = g(S) - 1$. Recursion (11) becomes :

$$f(k, x_i) = \min_{x_j} [\, f(k - 1, x_j) + c(x_j, x_i)\,]. \tag{4}$$

and is initialized by $f(1, x_i) = c(x_1, x_i)$. This recursion is the shortest n-path relaxation of the TSP.

(B) $g(S) = \sum_{x_i \in S} q_i$. (q-path relaxation).

Let us associate an integer number $q_i \geq 1$ with every vertex $x_i \in X$, ($q_1 = 0$, for the depot). Define $g(S) = q \equiv \sum_{x_i \in S} q_i$. We then have :

$$g(S - x_i) = g(S) - q_i.$$

Recursion (3) now becomes :

$$f(q, x_i) \;=\; \min_{x_j} \; [\, f(q - q_i, x_j) + c(x_j, x_i) \,]. \tag{5}$$

and initialized by :

$$\begin{aligned} f(q, x_i) &= c(x_1, x_i), && \text{if } q = q_i \\ &= \infty, && \text{if } q \neq q_i. \end{aligned}$$

In (5) the minimization must be over those values of x_j for which $q_j \leq q - q_i$. This recursion (5) is the shortest q-path relaxation.

3.2.2 Imposing loopless constraints

From the previous section we can see how a mapping function g(.) can be used to reduce the dimensionality of the state space. The introduction of g(.) does not, in general, allow any detailed knowledge of the state and hence one cannot impose additional conditions to ensure that a feasible solution to the original problem is obtained. However, certain specific restrictions can be imposed without increasing the dimensionality of state space and these restrictions improve the quality of the solution generated by solving the relaxed problem. In the case of the TSP relaxation defined by recursion (3), for example, it is possible to impose the condition that the path should not contain loops formed by three consecutive vertices, i.e., to avoid paths of the form like ... x_i, x_j, x_i ... . This can be done in the following way.

Let $p(g(S), x)$ be the vertex just prior to x on the path corresponding to $f(g(S), x)$. Let $\phi(g(S), x)$ be the least cost path from the initial state $(g(\{x_1\}), x_1)$ to state $(g(S), x)$ and with $\pi(g(S), x) \neq p(g(S), x)$, where $\pi(g(S), x)$ is the vertex just prior to vertex x on the path corresponding to $\phi(g(S), x)$.

Recursion (3) now becomes as :

$$f(g(S), x) = \min_{y} \begin{bmatrix} f(g(S\text{-}x), y) + c(y, x), & \text{if } p(g(S\text{-}x), y) \neq x \\ \\ \phi(g(S\text{-}x), y) + c(y, x), & \text{otherwise.} \end{bmatrix} \tag{6a}$$

The value of y producing the above minimum is p(g(S), x).

$$\phi(g(S), x) = \min_{y \neq p(g(S),x)} \begin{bmatrix} f(g(S\text{-}x), y) + c(y, x), & \text{if } p(g(S\text{-}x), y) \neq x \\ \\ \phi(g(S\text{-}x), y) + c(y, x), & \text{otherwise.} \end{bmatrix} \tag{6b}$$

The value of y producing the minimum of the above (6b) is $\pi(g(S), x)$.

The initialization is now

$$\begin{aligned} f(w, y) &= c(x_1, y) \text{ and } p(w, y) = x_1, \quad \text{if } w = g(\{y\}) \\ &= \infty, \qquad \text{otherwise} \end{aligned}$$

and

$$\phi(w, y) = \infty.$$

<u>The relaxed q-path recursion</u>

Let us define $g(S) = q \equiv \sum q_i$. Relaxed recursions (6a) and (6b) can now be rewritten as follows :

$$f(q, x_i) = \min_{\substack{x_j, \\ q-q_i \geq q_j}} \begin{bmatrix} f(q\text{-}q_i, x_j) + c(x_j, x_i), & \text{if } p(q\text{-}q_i, x_j) \neq x_i \\ \\ \phi(q\text{-}q_i, x_j) + c(x_j, x_i), & \text{otherwise.} \end{bmatrix} \tag{7a}$$

$$\phi(q, x_i) = \min_{\substack{x_j,\ x_j \neq p(q,x_i) \\ q-q_i \geq q_j}} \begin{bmatrix} f(q\text{-}q_i, x_j) + c(x_j, x_i), & \text{if } p(q\text{-}q_i, x_j) \neq x_i \\ \\ \phi(q\text{-}q_i, x_j) + c(x_j, x_i), & \text{otherwise.} \end{bmatrix} \tag{7b}$$

The initialization is given below :

$$\left.\begin{array}{l} \left.\begin{array}{l} f(q, x_i) = c(x_1, x_i) \text{ and } p(q, x_i) = x_1, \\ \phi(q, x_i) = \infty \text{ and } \pi(q, x_i) = \text{undefined}, \end{array}\right\} \text{ if } q = q_i \\ \left.\begin{array}{l} f(q, x_i) = \infty \text{ and } p(q, x_i) = \text{undefined}, \\ \phi(q, x_i) = \infty \text{ and } \pi(q, x_i) = \text{undefined}. \end{array}\right\} \text{ if } q \neq q_i \end{array}\right\} \qquad (7c)$$

The value of j (vertex x_j) producing the minimum of recursion (7a) is $p(q, x_i)$, and the value of j producing the minimum of recursion (7b) is $\pi(q, x_i)$. Note that $f(q, x_i)$ and $\phi(q, x_i)$ will remain unchanged for $q \leq q_i$ as is apparent from recursions (7a) and (7b).

3.3 Bounds for the TSP from state-space relaxation (with loops)

It is clear from Christofides *et al.* [1981 b] that the state-space relaxations of the dynamic programming recursions of combinatorial optimization problems can be used to obtain lower bounds on the value of the solution to these problems. For the case of the TSP we will describe how some of these bounds can be obtained. We should note, however, that this is by no means an exhaustive list of bounds that can be derived from the state-space relaxation of the travelling salesman problem.

3.3.1 Direct bound for the TSP from q-path relaxation

A simple bound can be obtained from recursion (5) by noting that $f(q, x_i)$ is the least cost path (q-path) starting from vertex x_1, finishing at vertex x_i and having a weight q $(=\sum q_i)$. The bound is as follows :

$$B1 = \min_{x_i} \left[f(\overline{Q}, x_i) + c(x_i, x_1) \right], \text{ where } \overline{Q} = \sum_{x_i} q_i. \qquad (8)$$

3.3.2 Indirect bound for the TSP from 'through - circuits'

Let's now define a function $\psi(q, x_i)$ as the least cost of a circuit with total load $\overline{Q}$ starting and finishing at vertex x_1 and passing through vertex x_i, when the sum of the q_j of all vertices x_j preceding x_i along the circuit (and including x_i itself) adds up to q. We will call q the 'load position' of x_i. The function, $\psi(q, x_i)$ can be computed as follows :

$$\psi(q, x_i) = f(q, x_i) + f'(\overline{Q}-q+q_i, x_i). \tag{9}$$

We now define b_{iq} to be a lower bound on the cost of the least cost tour starting and finishing at vertex x_1 and passing through vertex x_i, when the load position of x_i is q. The b_{iq} can be computed as :

$$b_{iq} = \psi(q, x_i) \tag{10}$$

Let us now construct an $(n - 1)\times\overline{Q}$ matrix $[b_{iq}]$, where each row corresponds to a vertex x_i $(x_i \neq x_1)$ and each column corresponds to an integer $q = 1, \dots, \overline{Q}$.

Every vertex must be in some load position of a feasible tour and there can only be one vertex in any load position. Thus, a whole family of bounds for the TSP can be derived by the use of the matrix $[b_{iq}]$ as follows :

(i) The value of the solution of the bottleneck assignment problem (Garfinkel & Nemhauser [1970]) defined by $[b_{iq}]$ is a lower bound.

(ii) Any lower bound to the above bottleneck assignment problem is obviously also a bound to the TSP. One such bound is :

$$B2 = \max_{x_i} \left[\min_{q=1, \dots, \overline{Q}} b_{iq} \right] \tag{11}$$

3.3.3 A bound for the TSP based on 2 q-paths

Let us consider two vertices s and t $\in$ X which are maximally distant each other, i.e. for which

$$c_{st} = \max_{i,j} \; [\, c_{ij} \,]$$

If the TSP is a euclidean problem, then s and t are two vertices on the convex hull of vertices. For a symmetric TSP, a tour is composed of two paths (vertex disjoint) from s to t so that every other vertex ($\neq$ s, t) is on exactly one of these paths.

Fig. 3.1 shows two q-paths from s to t and t to s. Path P_1 has total "load" q and cost $f_s(q, t)$ and P_2 has total "load" $q^* = \overline{Q} - q + q_s + q_t$ and cost $f_t(q^*, s)$.

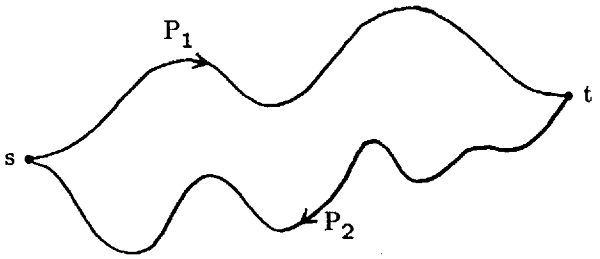


Figure 3.1. Two q-paths

Note that the total load on P_1 and P_2 is $\overline{Q} + q_s + q_t$, (i.e. the loads of vertices s and t are counted twice once on path P_1 and once on path P_2), and this is also the total required load of two paths forming a TSP tour. Thus, a lower bound B3 on the cost of a TSP tour can be derived from P_1 and P_2 as :

$$B3 \; = \min_{q_s + q_t \leq q \leq \overline{Q}} \; [\, f_s(q, t) + f_t(q^*, s) \,]. \tag{12}$$

(A) Use of the q-paths for problem reduction

The computation of the q-paths based on vertices s and t enable the elimination of certain arcs from further consideration without affecting the optimality of any TSP solution. Consider any arc (x_i, x_j) and let us say that the arc is in the TSP solution. The arc must, therefore, lie on either a path from s to t, or on a path from t to s. For a symmetric TSP the two cases are indistinguishable, so we will assume arc (x_i, x_j) to lie on the s to t path with some load, say, q.

Fig. 3.2 shows the situation.

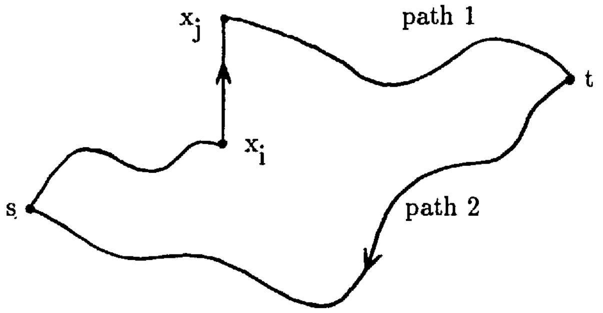


Figure 3.2. Two q-paths with an arc (x_i, x_j)

A lower bound A_{ij} on any TSP solution containing arc (x_i, x_j) is given by :

$$A_{ij} = \min_{q_s+q_i+q_j+q_t \leq q \leq \overline{Q}} \left[\min_{q_s+q_i \leq q' \leq q-q_j-q_t} \begin{array}{l} [\, f_s(q', x_i) + c_{ij} + \\ \quad f_t(q-q', x_j) + f_t(q^*, s) \,]. \end{array} \right] \tag{13}$$

Thus, if ZU is the current upper bound on the value of the optimal TSP solution (obtained, for example, by using a heuristic), then :

If

$$A_{ij} \geq ZU, \tag{14}$$

then arc (x_i, x_j) cannot be in any optimal solution, and can be removed from the set A of arcs of the graph on which the TSP is defined.

Thus, test 14 can be used to reduce the size of the TSP by deleting unnecessary arcs.

3.4 Bounds for the TSP from state-space relaxation (with loopless constraints)

In the case of the direct bound (B1), the bound is the same expression as in (8) simply by using the loopless values for f(. , .). However, in order to calculate the indirect bound (B2), expression (9) should be slightly changed. Let $\psi(q, x_i)$ be the least cost circuit without loops, starting from the depot, passing through x_i and finishing back at the depot with a total load $\overline{Q}$ when the load position of x_i is q. $\psi(q, x_i)$ must be composed of either two best q-paths to x_i whose total loads add up to $(\overline{Q}+q_i)$ or a best path and a second best path to x_i whose total loads add up to $(\overline{Q}+q_i)$. $\psi(q, x_i)$ can then be computed as follows :

$$\psi(q, x_i) = \min \left[\begin{array}{l} f(q, x_i) + f(\overline{Q}\text{-}q+q_i, x_i), \text{ if } p(q, x_i) \neq p(\overline{Q}\text{-}q+q_i, x_i), \\ \\ \min\, [f(q, x_i) + \phi(\overline{Q}\text{-}q+q_i, x_i), \phi(q, x_i) + f(\overline{Q}\text{-}q+q_i, x_i)], \\ \quad \text{if } p(q, x_i) = p(\overline{Q}\text{-}q+q_i, x_i). \end{array} \right] \quad (15)$$

We note here that the computational effort involved in computing the q-paths is linearly related to $\overline{Q}$. Thus, we can reduce the computational effort to almost a half by imposing a constraint, $q_i \leq q \leq 1/2(\overline{Q}+q_i)$.

Indirect bound (B2) is then obtained from expression (11) in the same way, by using the loopless values of $\psi(q, x_i)$ and hence b_{iq}.

For bound B3 expression (12) should also be slightly changed if the TSP is symmetric. In that case, the tour can be considered either as two paths from s to t or as two paths from t to s. Therefore, B3 can be restated in slightly stronger terms as follows :

(i) Consider two paths from s to t

Let

$$D_s(q, t) = \min \begin{bmatrix} f_s(q, t) + f_s(q^*, t), \quad \text{if } p_s(q, t) \neq p_s(q^*, t), \\ \\ \min [\, f_s(q, t)+\phi_s(q^*, t),\ \phi_s(q, t)+f_s(q^*, t) \,], \\ \text{otherwise} \end{bmatrix} \tag{16a}$$

(ii) Consider two paths from t to s

Let

$$D_t(q, s) = \min \begin{bmatrix} f_t(q, s) + f_t(q^*, s), \quad \text{if } p_t(q, s) \neq p_t(q^*, t), \\ \\ \min [\, f_t(q, s)+\phi_t(q^*, s),\ \phi_t(q, s)+f_t(q^*, s) \,], \\ \text{otherwise} \end{bmatrix} \tag{16b}$$

Then,

$$B3 = \min_{q_s+q_t \leq q \leq \bar{Q}} [\, \max \{ D_s(q, t), D_t(q, s) \} \,]. \tag{17}$$

3.5 Lagrangean penalty methods and subgradients to improve the bounds

In the previous section we have seen the simple bounds for the TSP. In this section we describe how a procedure can be used to improve the resulting bound further by using penalties in a Lagrangean fashion. The general objective is to force the solution corresponding to the relaxed problem 'closer' to feasibility.

The lower bound B(0) from the state-space relaxation of the TSP is computed as B1 (equation 8) or B2 (equation 11) or B3 (equation 12). In all cases, the bound corresponds to a circuit starting and finishing at vertex x_1. This circuit H(0), say, is normally infeasible, that is, some vertices (x_i) are not visited, whereas some other vertices (x_j) are visited twice. Fig. 3.3 may, for example, represent H(0) where vertices x_2 and x_5 are not visited, whereas vertices x_3 and x_8 are visited twice. Therefore, by penalizing vertices x_j (by a penalty λ_j) in normal Lagrangean fashion, a new bound $B(\lambda)$ can be obtained by resolving the recursions (for f(. , .) in the case of B1 ; and for f(. , .), ϕ(. , .) and ψ(. , .) in the case of B2 ; and for f(. , .), ϕ(. , .), D_s(. , .) and D_t(. , .) in the case of B3) with the updated cost matrix $[c'_{ij}]$, where $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$. A new circuit $H(\lambda)$ and new bound $B(\lambda)$ are then obtained. We wish to choose λ^* for which :

$$B(\lambda^*) = \max_{\lambda} [B(\lambda)] \tag{18}$$

Here we can use the normal subgradient optimization methods to compute λ^*, and more details are described below.

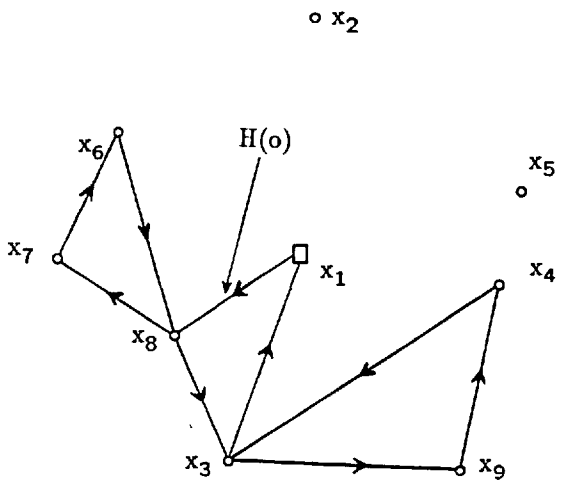


Figure 3.3 Circuit H(o) corresponding to bound B(o).

3.5.1 The subgradient method

Consider maximizing the bound in a Lagrangean fashion as a function of the multipliers. The strongest Lagrangean relaxation is obviously given by $\lambda = \lambda^*$. The subgradient optimization method for solving (18) Held *et al.* [1974], Sandi [1979] starts with some arbitray $\lambda = \lambda^o$ (say the zero vector) and at the *k*th iteration updates λ^k. Let $H(\lambda^k)$ be the optimal solution and let UB be the upper bound (the best solution value so far) for the problem. If $H(\lambda^k)$ is a tour, or if $Z(H(\lambda^k)) \geq$ UB, stop. Otherwise, for $x_i \in X'$, let d_i be the degree of vertex x_i in $H(\lambda^k)$. Then the n-vector with components $d_i^k - 2$ is a subgradient of $B(\lambda)$ at λ^k. Set

$$\lambda_i^{k+1} = \lambda_i^k + t^k(d_i^k - 2), \qquad x_i \in X', \tag{19}$$

where t^k is the 'step length' defined by :

$$t^k = \alpha \cdot \frac{(UB - B(\lambda^k))}{\sum_{i \in X'} (d_i^k - 2)^2} \tag{20}$$

with $0 < \alpha \leq 2$. Then set $k = k + 1$ and repeat the procedure.

It can be shown that the method converges if $\sum_{k=1}^{\infty} t^k = \infty$ and $\lim_{k \to \infty} t^k = 0$. These conditions are satisfied if one starts with $\alpha = 2$ and periodically reduces α by some factor.

3.5.2 An algorithm for the lower bound

We will describe an algorithm to improve the lower bounds for the TSP with penalty procedures. This algorithm can be used for the direct bound, the indirect bound and the bound based on two-paths for the TSP in the same way.

Step 0 : (Initialization). Set the best lower bound $ZL^* = 0$. Let ZU^* be the value of the best solution so far. Set $\alpha = 2.0$ and KOUNT = 0.

Step 1 : **(Initialization).** Set $\lambda_i = 0$, $i = 1, \dots, n$ and $d_i = 0$, $i = 1, \dots, n$.

Step 2 : **(Calculation of lower bound).** Compute the lower bound $B(\lambda)$ using the state-space relaxation as mentioned in the previous section. Let ZL be a updated lower bound on the value of the solution to the TSP, $ZL = B(\lambda) - 2\sum\lambda_i$. If $ZL^* < ZL$, set $ZL^* = ZL$. If $ZL^* \geq ZU^*$ or KOUNT = maximum number of iterations allowed, stop. Else if $ZL^* < ZU^*$ and KOUNT $\neq$ maximum number of iterations allowed, KOUNT = KOUNT + 1, and go to step 3.

Step 3 : **(Backtracking).** Backtrack in order to find the circuit $H(\lambda)$ corresponding to the above lower bound using $f(q, x_i)$, $p(q, x_i)$, $\phi(q, x_i)$ and $\pi(q, x_i)$. Check the degree d_i of vertex x_i with respect to graph produced by $H(\lambda)$. If the degree d_i is 2, for all i ($i = 1, \dots, n$), stop. (In this case ZL^* is the best lower bound that can be obtained by this procedure and is the optimal solution value for the TSP. Otherwise, go to step 4.

Step 4 : **(Penalties).** Compute penalties as given below :

$$\lambda_i \Leftarrow \lambda_i + \alpha \cdot \frac{ZU^* - ZL}{\sum_{j=1}^{n} (d_j - 2)^2} \cdot (d_i - 2) \cdot [\, q_i/\max_i [q_i] \,], \quad i = 2, \dots, n$$

where α is a constant ($0 < \alpha \leq 2$) and can be periodically reduced by some factor. For example, after every 5 iterations α is reduced by a half, i.e. $\alpha = 2.0$ for KOUNT ≤ 5, $\alpha = 1.0$ for $6 \leq$ KOUNT < 10, and so on, and where the expression $[\, q_i/\max_i [q_i] \,]$ gives greater "weight" to those vertices with high demand.

Step 5 : **(Udating the cost matrix).** Modify and update the cost matrix $[c_{ij}]$ as :

$$c'_{ij} = c_{ij} + \lambda_i + \lambda_j.$$

Step 6 : **(Computation of f, p, ϕ and π from the relaxed recursion).** Compute $f(q, x)$, p(q, x), ϕ(q, x) and π(q, x) from the state-space relaxation recursions for the updated cost matrix [c'_{ij}]. Go to step 1.

At the end of the *k*th iteration, ZL* is the best lower bound found so far.

3.6 Computation of bounds with an example

We consider the 9-customer symmetric TSP whose cost matrix and graph for this example are given in Table 3.1 and Fig. 3.4. We will use state-space relaxation to compute lower bounds to the value of the optimal solution to this TSP.

First, we will compute the direct lower bound B1 and the lower bound from through q-paths B2 allowing for loops. An example for bound B3 can be computed in a similar fashion and is not given here. Then better bounds of the above two kinds will be computed after imposing the loopless conditions. Finally we will improve the above bounds by using the penalty procedures.

Table 3.1 Cost(distance) matrix [c_{ij}]

$x_i \backslash x_j$	1	2	3	4	5	6	7	8	9
1	-	28	21	14	17	18	22	15	30
2	28	-	47	36	25	20	35	38	50
3	21	47	-	26	37	30	20	13	18
4	14	36	26	-	15	31	34	25	17
5	17	25	37	15	-	29	39	22	35
6	18	20	30	31	29	-	16	19	45
7	22	35	20	34	39	16	-	12	32
8	15	38	13	25	22	19	12	-	28
9	30	50	18	17	35	45	32	28	-

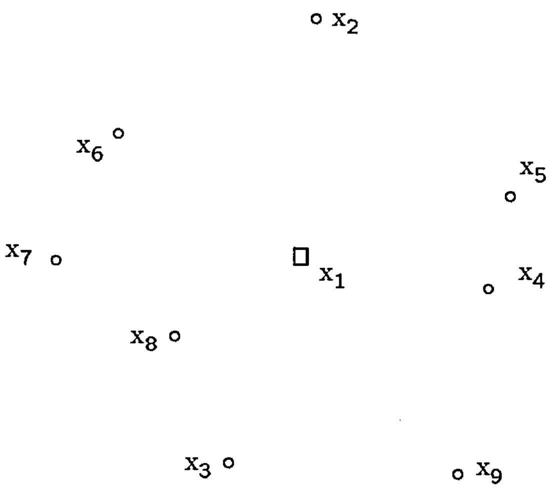


Figure 3.4 Graph of vertices(customers).

3.6.1 Bound B1 (with loops)

In this example, the underlying graph represented by the cost(distance) matrix (Table 3.1) is complete. Recursion (5) can be rewritten as :

$$f(q, x_i) = \min_{x_j,\ q-q_i \geq q_j} [\ f(q - q_i, x_j) + c(x_j, x_i)\ ]. \tag{5'}$$

Note that $f(q, x_i)$ will remain unchanged for $q \leq q_i$ as is apparent from recursion (5′). Let us choose (arbitrarily) a set of weights q_i to the vertices 1, ... , 9. Let these weights be given by :

$x_i =$	1	2	3	4	5	6	7	8	9
$q_i =$	0	2	3	1	1	2	1	3	2

and hence $\overline{Q} = \sum_i q_i = 15$.

We will use the recursion (5′) to compute $f(\overline{Q}, x_i)$ for $\overline{Q} = 15$ and $x_i = \{x_1, \dots, x_9\}$. From expression (5) the initialization is :

for x_2 : $f(2, x_2) = 28$, $p(2, x_2) = x_1$;

$f(q, x_2) = \infty$, $p(q, x_2)$: unspecified for $q \neq 2$.

for x_3 : $f(3, x_3) = 21$, $p(3, x_3) = x_1$;

$f(q, x_3) = \infty$, $p(q, x_3)$: unspecified for $q \neq 3$.

for x_4 : $f(1, x_4) = 14$, $p(1, x_4) = x_1$;

$f(q, x_4) = \infty$, $p(q, x_4)$: unspecified for $q \neq 1$.

for x_5 : $f(1, x_5) = 17$, $p(1, x_5) = x_1$;

$f(q, x_5) = \infty$, $p(q, x_5)$: unspecified for $q \neq 1$.

for x_6 : $f(2, x_6) = 18$, $p(2, x_6) = x_1$;

$f(q, x_6) = \infty$, $p(q, x_6)$: unspecified for $q \neq 2$.

for x_7 : $f(1, x_7) = 22$, $p(1, x_7) = x_1$;

$f(q, x_7) = \infty$, $p(q, x_7)$: unspecified for $q \neq 1$.

for x_8 : $f(3, x_8) = 15$, $p(3, x_8) = x_1$;

$f(q, x_8) = \infty$, $p(q, x_8)$: unspecified for $q \neq 3$.

for x_9 : $f(2, x_9) = 30$, $p(2, x_9) = x_1$;

$f(q, x_9) = \infty$, $p(q, x_9)$: unspecified for $q \neq 2$.

We will use the value of q to index the iterations of recursion (5′), i.e. we will call iteration 2 the iteration which computes all of $f(2, x_i)$, iteration 3 the iteration which computes of all of $f(3, x_i)$, etc.

Iteration 2 ($q = 2$).

The values of $f(2, x_2)$, $f(2, x_3)$, $f(2, x_6)$, $f(2, x_8)$ and $f(2, x_9)$ remain unchanged (as noted earlier).

$f(2, x_4) = \min [\ f(1, x_5) + c(x_5, x_4),\ \ f(1, x_7) + c(x_7, x_4)\ ]$

$= \min [\ (17 + 15),\ (22 + 34)\ ] = 32$

The vertex which produced the minimum of $f(2, x_4)$ is :

$p(2, x_4) = x_5$.

$f(2, x_5) = \min [\ f(1, x_4) + c(x_4, x_5),\ \ f(1, x_7) + c(x_7, x_5)\ ]$

$= \min [\ (14 + 15),\ (22 + 39)\ ] = 29$

$p(2, x_5) = x_4$.

$f(2, x_7) = \min [\ f(1, x_4) + c(x_4, x_7),\ \ f(1, x_5) + c(x_5, x_7)\ ]$

$= \min [\ (14 + 34),\ (17 + 39)\ ] = 48$

$p(2, x_7) = x_4$.

This is the end of iteration 2.

Iteration 3 (q = 3).

The values of $f(3, x_3)$, and $f(3, x_8)$ remain unchanged (as noted earlier).

$f(3, x_2) = \min [\ f(1, x_4) + c(x_4, x_2),\ \ f(1, x_5) + c(x_5, x_2),\ \ f(1, x_7) + c(x_7, x_2)\ ]$

$= \min [\ (14 + 36),\ (17 + 25),\ (22 + 35)\ ] = 42$

$p(3, x_2) = x_5$.

$f(3, x_4) = \min [f(2, x_2) + c(x_2, x_4),\ \ f(2, x_5) + c(x_5, x_4),\ \ f(2, x_6) + c(x_6, x_4),$

$f(2, x_7) + (x_7, x_4),\ \ f(2, x_9) + (x_9, x_4)\ ]$

$= \min [\ (28 + 36),\ (29 + 15),\ (18 + 31),\ (48 + 34),\ (30 + 17)\ ] = 44$

$p(3, x_4) = x_5$.

$f(3, x_5) = \min [f(2, x_2) + c(x_2, x_5),\ \ f(2, x_4) + c(x_4, x_5),\ \ f(2, x_6) + c(x_6, x_5),$

$f(2, x_7) + (x_7, x_5),\ \ f(2, x_9) + (x_9, x_5)\ ]$

$= \min [\ (28 + 25),\ (32 + 15),\ (18 + 29),\ (48 + 39),\ (30 + 35)\ ] = 47$

$p(3, x_5) = x_5$ (or x_6).

$f(3, x_6) = \min [\ f(1, x_4) + c(x_4, x_6),\ \ f(1, x_5) + c(x_5, x_6),\ \ f(1, x_7) + c(x_7, x_6)\ ]$

$$
\begin{aligned}
&= \min\,[\,(14 + 31),\ (17 + 29),\ (22 + 16)\,] = 38\\
p(3, x_6) &= x_7.\\
f(3, x_7) &= \min\,[f(2, x_2) + c(x_2, x_7),\ \ f(2, x_4) + c(x_4, x_7),\ \ f(2, x_5) + c(x_5, x_7),\\
&\qquad f(2, x_6) + (x_6, x_7),\ \ f(2, x_9) + (x_9, x_7)\,]\\
&= \min\,[\,(28 + 35),\ (32 + 34),\ (29 + 39),\ (18 + 16),\ (30 + 32)\,] = 34\\
p(3, x_7) &= x_6.\\
f(3, x_9) &= \min\,[\,f(1, x_4) + c(x_4, x_9),\ \ f(1, x_5) + c(x_5, x_9),\ \ f(1, x_7) + c(x_7, x_9)\,]\\
&= \min\,[\,(14 + 17),\ (17 + 35),\ (22 + 32)\,] = 31\\
p(3, x_9) &= x_4.
\end{aligned}
$$

This is the end of iteration 3. The values of functions f(. , .) and p(. , .) at this point are as follows :

f(q, x)

$x_i \backslash q$	1	2	3	> 4
2	∞	28	42	∞
3	∞	∞	21	∞
4	14	32	44	∞
5	17	29	47	∞
6	∞	18	38	∞
7	22	48	34	∞
8	∞	∞	15	∞
9	∞	30	31	∞

p(q, x)

$x_i \backslash q$	1	2	3	> 3
2	-	x_1	x_5	-
3	-	-	x_1	-
4	x_1	x_5	x_5	-
5	x_1	x_4	x_4,x_6	-
6	-	x_1	x_7	-
7	x_1	x_4	x_6	-
8	-	-	x_1	-
9	-	x_1	x_4	-

Iteration 4 (q = 4).

$$f(4, x_2) = \min\,[\,f(2, x_4) + c(x_4, x_2),\ \ f(2, x_5) + c(x_5, x_2),\ \ f(2, x_6) + c(x_6, x_2),$$

$f(2, x_7) + c(x_7, x_2),\ \ f(2, x_9) + c(x_9, x_2)\ ]$

$= \min\ [\ (32 + 36), (29 + 25), (18 + 20), (48 + 35), (30 + 50)\ ] = 38$

$p(4, x_2) = x_6.$

$f(4, x_3) = \min\ [\ (14 \underset{x_4}{+} 26), (17 \underset{x_5}{+} 37), (22 \underset{x_7}{+} 30)\ ] = 40$

$p(4, x_3) = x_4.$

$f(4, x_4) = \min[(42 \underset{x_2}{+} 36), (21 \underset{x_3}{+} 26), (47 \underset{x_5}{+} 15), (38 \underset{x_6}{+} 31), (34 \underset{x_7}{+} 34), (15 \underset{x_8}{+} 26), (31 \underset{x_9}{+} 17)] = 40$

$p(4, x_4) = x_8.$

$f(4, x_5) = \min[(42 \underset{x_2}{+} 25), (21 \underset{x_3}{+} 37), (44 \underset{x_4}{+} 15), (38 \underset{x_6}{+} 29), (34 \underset{x_7}{+} 39), (15 \underset{x_8}{+} 22), (31 \underset{x_9}{+} 35)] = 37$

$p(4, x_5) = x_8.$

$f(4, x_6) = \min[(28 \underset{x_2}{+} 20), (32 \underset{x_3}{+} 31), (29 \underset{x_5}{+} 29), (48 \underset{x_7}{+} 16), (30 \underset{x_9}{+} 45)] = 48$

$p(4, x_6) = x_2.$

$f(4, x_7) = \min[(42 \underset{x_2}{+} 35), (21 \underset{x_3}{+} 20), (44 \underset{x_4}{+} 34), (47 \underset{x_5}{+} 39), (38 \underset{x_6}{+} 16), (15 \underset{x_8}{+} 12), (31 \underset{x_9}{+} 32)] = 27$

$p(4, x_7) = x_8.$

$f(4, x_8) = \min[(14 \underset{x_4}{+} 25), (17 \underset{x_5}{+} 22), (22 \underset{x_7}{+} 12)] = 34$

$p(4, x_8) = x_7.$

$f(4, x_9) = \min[(28 \underset{x_2}{+} 50), (32 \underset{x_4}{+} 17), (29 \underset{x_5}{+} 35), (18 \underset{x_6}{+} 45), (48 \underset{x_7}{+} 32), (30 \underset{x_8}{+} 28)] = 49$

$p(4, x_9) = x_4.$

This is the end of iteration 4.

Iteration 5 $(q = 5)$.

$f(5, x_2) = \min[(21 \underset{x_3}{+} 47), (44 \underset{x_4}{+} 36), (47 \underset{x_5}{+} 25), (38 \underset{x_6}{+} 20), (34 \underset{x_7}{+} 35), (15 \underset{x_8}{+} 38), (31 \underset{x_9}{+} 50)] = 53$

$p(5, x_2) = x_8.$

$f(5, x_3) = \min[(28 \underset{x_2}{+} 42), (32 \underset{x_4}{+} 26), (29 \underset{x_5}{+} 37), (18 \underset{x_6}{+} 30), (48 \underset{x_7}{+} 20), (30 \underset{x_9}{+} 18)] = 48$

$p(5, x_3) = x_6$ (or x_9).

$f(5, x_4) = \min[(38 \underset{x_2}{+} 36), (40 \underset{x_3}{+} 26), (37 \underset{x_5}{+} 15), (48 \underset{x_6}{+} 31), (27 \underset{x_7}{+} 34), (34 \underset{x_8}{+} 25), (49 \underset{x_9}{+} 17)] = 52$

$p(5, x_4) = x_5.$

$$f(5, x_5) = \min[(38 \underset{x_2}{+} 25), (40 \underset{x_3}{+} 37), (40 \underset{x_4}{+} 15), (48 \underset{x_6}{+} 29), (27 \underset{x_7}{+} 39), (34 \underset{x_8}{+} 22), (49 \underset{x_9}{+} 35)] = 55$$

$p(5, x_5) = x_4$.

$$f(5, x_6) = \min[(42 \underset{x_2}{+} 20), (21 \underset{x_3}{+} 30), (44 \underset{x_4}{+} 31), (47 \underset{x_5}{+} 29), (34 \underset{x_7}{+} 16), (15 \underset{x_8}{+} 19), (31 \underset{x_9}{+} 45)] = 34$$

$p(5, x_6) = x_8$.

$$f(5, x_7) = \min[(38 \underset{x_2}{+} 35), (40 \underset{x_3}{+} 20), (40 \underset{x_4}{+} 34), (37 \underset{x_5}{+} 39), (48 \underset{x_6}{+} 16), (34 \underset{x_8}{+} 12), (49 \underset{x_9}{+} 32)] = 46$$

$p(5, x_7) = x_8$.

$$f(5, x_8) = \min[(28 \underset{x_2}{+} 38), (32 \underset{x_4}{+} 25), (29 \underset{x_5}{+} 22), (18 \underset{x_6}{+} 19), (48 \underset{x_7}{+} 12), (30 \underset{x_9}{+} 28)] = 37$$

$p(5, x_8) = x_6$.

$$f(5, x_9) = \min[(42 \underset{x_2}{+} 50), (21 \underset{x_3}{+} 18), (44 \underset{x_4}{+} 17), (47 \underset{x_5}{+} 35), (38 \underset{x_6}{+} 45), (34 \underset{x_7}{+} 32), (15 \underset{x_8}{+} 28)] = 39$$

$p(5, x_9) = x_3$.

This is the end of iteration 5.

Similarly for iterations q= 6 to q= 15. The final results are shown in Table 3.2a and 3.2b below.

Table 3.2a f(q, x)

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	28	42	38	53	62	54	63	76	73	78	87	80	89	102
x_3	∞	∞	21	40	48	28	47	50	47	52	64	54	71	76	73
x_4	14	32	44	40	52	56	54	64	63	66	78	82	80	90	89
x_5	17	29	47	37	55	59	56	61	75	63	80	85	82	87	99
x_6	∞	18	38	48	34	43	56	53	58	67	60	69	82	79	84
x_7	22	48	34	27	46	49	46	51	65	53	70	75	72	77	89
x_8	∞	∞	15	34	37	34	39	53	41	58	63	60	65	77	67
x_9	∞	30	31	49	39	57	65	46	65	68	65	70	82	72	89

Table 3.2b p(q, x)

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	-	x_1	x_5	x_6	x_8	$x_{5,7}$	x_6	x_6	x_6	x_6	x_6	x_6	x_6	x_6	x_6
x_3	-	-	x_1	x_4	$x_{6,9}$	x_8	$x_{7,8}$	x_8	x_8	x_8	x_9	x_8	x_8	x_8	x_8
x_4	x_1	x_5	x_5	x_8	x_5	x_9	x_3	x_8	x_9	x_8	$x_{3,5}$	x_9	x_3	x_8	x_9
x_5	x_1	x_4	$x_{4,6}$	x_8	x_4	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8
x_6	-	x_1	x_7	x_2	x_8	x_7	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8
x_7	x_1	x_4	x_6	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8	x_8
x_8	-	-	x_1	x_7	x_6	x_3	x_7	x_6	x_3	x_7	$x_{3,7}$	x_3	$x_{3,7}$	x_3	x_3
x_9	-	x_1	x_4	x_4	x_3	x_4	x_8	x_3	x_3	x_3	x_3	x_3	x_3	x_3	x_3

The above tableau for f(q, x) gives the values of $f(\overline{Q}, x_i)$, (i.e. $f(15, x_i)$), for all x_i and can be used in the expression (8) to obtain bound B1 as :

$$\begin{aligned} B1 &= \min_{x_i} [\, f(\overline{Q}, x_i) + c(x_i, x_1)\,] \\ &= \min\ [\, (\underset{x_2}{102 + 28}),\ (\underset{x_3}{72 + 21}),\ (\underset{x_4}{89 + 14}),\ (\underset{x_5}{99 + 17}), \\ &\qquad (\ \underset{x_6}{84 + 18}),\ (\underset{x_7}{89 + 22}),\ (\underset{x_8}{67 + 15}),\ (\underset{x_9}{88 + 30})\,] \\ &= 82, \text{ with the minimum obtained for } x_8. \end{aligned}$$

By backtracking through tableau f(q, x) and p(q, x), we obtain the q-path corresponding to the above value of 82 as follows :

Vertex	Predecessor on path	Comment
$\underline{x_1}$	x_8	from the computation of B1.
x_8	x_3	p(15, x_8)
x_3	x_8	p(12, x_3)
x_8	x_3	p(9, x_8)
x_3	x_8	p(6, x_3)
x_8	$\underline{x_1}$	p(3, x_8)

The q-path is shown diagrammatically Fig. 3.5.

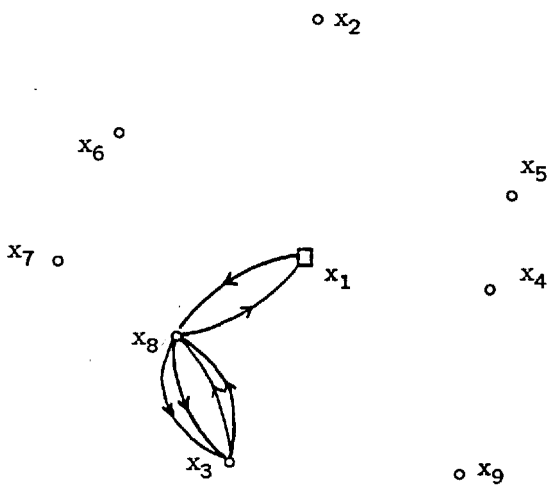


Figure 3.5 q-path corresponding to direct bound, B1 = 82.

3.6.2 Bound B2 (with loops)

Let us compute $\psi(q, x_i)$ from equation (9), i.e. ;

$$\psi(q, x_i) = f(q, x_i) + f(\overline{Q}\text{-}q+q_i, x_i).$$

Thus :

$$\begin{aligned}\psi(2, x_2) &= f(2, x_2) + f(15 - 2 + 2, x_2)\\ &= f(2, x_2) + f(15, x_2)\\ &= 28 + 93 = 121.\\ \psi(3, x_2) &= f(3, x_2) + f(15 - 3 + 2, x_2)\\ &= f(3, x_2) + f(14, x_2)\\ &= 42 + 89 = 131.\\ \psi(4, x_2) &= f(4, x_2) + f(15 - 4 + 2, x_2)\\ &= f(4, x_2) + f(13, x_2)\\ &= 38 + 80 = 118.\end{aligned}$$

etc. The computed tableau of $\psi(q, x_i)$ is as shown below :

Table 3.3 $\psi(q, x)$

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	130	131	$\underline{118}$	140	140	127	139	139	127	140	140	$\underline{118}$	131	130
x_3	∞	∞	94	116	119	82	111	102	94	102	111	82	119	116	94
x_4	103	122	124	122	130	122	117	128	117	122	130	122	124	122	103
x_5	116	116	129	122	135	122	131	122	131	122	135	122	129	116	116
x_6	∞	102	117	130	103	103	123	111	111	123	103	103	130	117	102
x_7	111	125	106	102	116	102	111	102	111	102	116	102	106	125	111
x_8	∞	∞	82	111	102	94	102	111	82	111	102	94	102	111	82
x_9	∞	119	103	131	109	122	133	111	111	133	122	109	131	103	119

If b_{iq} is computed from the equation ($b_{iq} = \psi(q, x_i)$), then the above matrix is also the matrix $[b_{iq}]$, and the value the solution of the bottleneck assignment problem for this matrix is a bound.

A lower bound to this solution value (and hence to the TSP) is given by equation (11) as :

$$\begin{aligned} B2 &= \max_{x_i} [\min_{q} b_{iq}] \\ &= \max_{x_i} [118,\ 82,\ 103,\ 116,\ 102,\ 102,\ 82,\ 103] \\ &= 118, \text{ as indicated in Table 3.3.} \end{aligned}$$

In this example B2 = 118, the minimum shown underlined in the above matrix for $x_i = x_2$ and q = 13.

The value of 118 is obtained from :

$$f(13, x_2) + f(4, x_2).$$

The paths corresponding to each one of these two terms can be obtained by backtracking through tableau p(q, x), and the corresponding through-circuit (composed of those two paths) is shown below in Fig. 3.6.

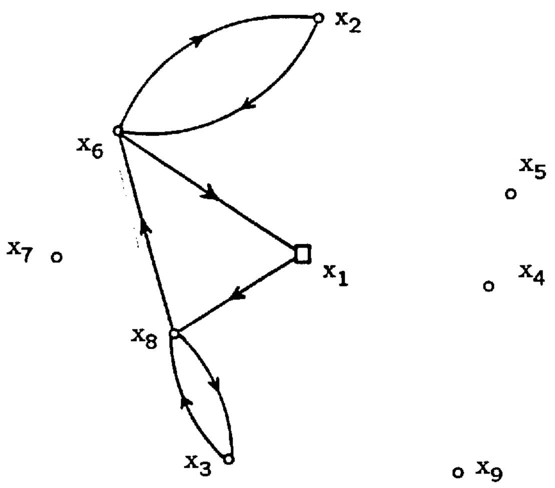


Figure 3.6 Through-circuit corresponding to indirect bound B2 = 118.

3.6.3 Bound B1 (with no loops)

We will use recursions (7a) - (7c) to compute f(. , .), ϕ(. , .), p(. , .) and π(. , .), and then use the value of $f(\overline{Q}, x_i)$ for $\overline{Q} = 15$ to obtain B1.

$$\begin{array}{lrcl} \text{for } x_2 : & f(2, x_2) = 28 & \text{and} & p(2, x_2) = x_1, \\ & \phi(2, x_2) = \infty & \text{and} & \pi(2, x_2) = \text{undefined} ; \end{array}$$

$$\left.\begin{array}{rcl} f(q, x_2) = \infty & \text{and} & p(q, x_2) = \text{undefined}, \\ \phi(q, x_2) = \infty & \text{and} & \pi(q, x_2) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 2$$

$$\begin{array}{lrcl} \text{for } x_3 : & f(3, x_3) = 21 & \text{and} & p(3, x_3) = x_1, \\ & \phi(3, x_3) = \infty & \text{and} & \pi(3, x_3) = \text{undefined} ; \end{array}$$

$$\left.\begin{array}{rcl} f(q, x_3) = \infty & \text{and} & p(q, x_3) = \text{undefined}, \\ \phi(q, x_3) = \infty & \text{and} & \pi(q, x_3) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 3$$

$$\text{for } x_4 : \quad \begin{array}{l} f(1, x_4) = 14 \text{ and } p(1, x_4) = x_1, \\ \phi(1, x_4) = \infty \text{ and } \pi(1, x_4) = \text{undefined} ; \\ \left.\begin{array}{l} f(q, x_4) = \infty \text{ and } p(q, x_4) = \text{undefined}, \\ \phi(q, x_4) = \infty \text{ and } \pi(q, x_4) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 1 \end{array}$$

$$\text{for } x_5 : \quad \begin{array}{l} f(1, x_5) = 17 \text{ and } p(1, x_5) = x_1, \\ \phi(1, x_5) = \infty \text{ and } \pi(1, x_5) = \text{undefined} ; \\ \left.\begin{array}{l} f(q, x_5) = \infty \text{ and } p(q, x_5) = \text{undefined}, \\ \phi(q, x_5) = \infty \text{ and } \pi(q, x_5) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 1 \end{array}$$

$$\text{for } x_6 : \quad \begin{array}{l} f(2, x_6) = 18 \text{ and } p(2, x_6) = x_1, \\ \phi(2, x_6) = \infty \text{ and } \pi(2, x_6) = \text{undefined} ; \\ \left.\begin{array}{l} f(q, x_6) = \infty \text{ and } p(q, x_6) = \text{undefined}, \\ \phi(q, x_6) = \infty \text{ and } \pi(q, x_6) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 2 \end{array}$$

$$\text{for } x_7 : \quad \begin{array}{l} f(1, x_7) = 22 \text{ and } p(1, x_7) = x_1, \\ \phi(1, x_7) = \infty \text{ and } \pi(1, x_7) = \text{undefined} ; \\ \left.\begin{array}{l} f(q, x_7) = \infty \text{ and } p(q, x_7) = \text{undefined}, \\ \phi(q, x_7) = \infty \text{ and } \pi(q, x_7) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 1 \end{array}$$

$$\text{for } x_8 : \quad \begin{array}{l} f(3, x_8) = 15 \text{ and } p(3, x_8) = x_1, \\ \phi(3, x_8) = \infty \text{ and } \pi(3, x_8) = \text{undefined} ; \\ \left.\begin{array}{l} f(q, x_8) = \infty \text{ and } p(q, x_8) = \text{undefined}, \\ \phi(q, x_8) = \infty \text{ and } \pi(q, x_8) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 3 \end{array}$$

$$\text{for } x_9 : \quad \begin{array}{l} f(2, x_9) = 30 \text{ and } p(2, x_9) = x_1, \\ \phi(2, x_9) = \infty \text{ and } \pi(2, x_9) = \text{undefined} ; \\ \left.\begin{array}{l} f(q, x_9) = \infty \text{ and } p(q, x_9) = \text{undefined}, \\ \phi(q, x_9) = \infty \text{ and } \pi(q, x_9) = \text{undefined}. \end{array}\right\} \text{ if } q \neq 2 \end{array}$$

Iteration 2 ($q = 2$)

The values of $f(2, x_2)$, $f(2, x_3)$, $f(2, x_6)$, $f(2, x_8)$, $f(2, x_9)$, $\phi(2, x_2)$, $\phi(2, x_3)$, $\phi(2, x_6)$, $\phi(2, x_8)$ and $\phi(2, x_9)$ remain unchanged (as noted earlier).

In the case of $f(2, x_4)$, $p(q\text{-}q_j, x_j) \neq x_i$, for j, i.e. $p(1, x_5) = x_1$, $p(1, x_5) = x_1$, $p(1, x_5) = p(1, x_7) \neq x_4)$, therefore we use the first term of the recursion (7a).

$$\begin{aligned} f(2, x_4) &= \min [\, f(1, x_5) + c(x_5, x_4), f(1, x_7) + c(x_7, x_4) \,] \\ &= \min [\, (17 + 15), (22 + 34) \,] = 32. \end{aligned}$$

The vertex which produced the minimum of $f(2, x_4)$ is :

$$p(2, x_4) = x_5.$$

In $\phi(2, x_4)$, the condition, $x_j \neq p(q, x_i)$, should be satisfied, therefore we do not have to use the term for the vertex which produced $p(2, x_4)$. Hence, $\phi(2, x_4)$ can be computed as follows :

$$\begin{aligned} \phi(2, x_4) &= \min [\, f(1, x_7) + c(x_7, x_4) \,] \\ &= \min [\, (22 + 34) \,] = 56. \end{aligned}$$

The vertex which produced the minimum of $\phi(2, x_4)$ is :

$$\begin{aligned} \pi(2, x_4) &= x_7. \\ f(2, x_5) &= \min [\, f(1, x_4) + c(x_4, x_5), f(1, x_7) + c(x_7, x_5) \,] \\ &= \min [\, (14 + 15), (22 + 39) \,] = 29. \\ p(2, x_5) &= x_4. \\ \phi(2, x_5) &= \min [\, f(1, x_7) + c(x_7, x_5) \,] \\ &= \min [\, (22 + 39) \,] = 61. \\ \pi(2, x_5) &= x_7. \\ f(2, x_7) &= \min [\, f(1, x_4) + c(x_4, x_7), f(1, x_5) + c(x_5, x_7) \,] \\ &= \min [\, (14 + 34), (17 + 39) \,] = 48. \\ p(2, x_7) &= x_4. \end{aligned}$$

$$\phi(2, x_7) = \min [\, f(1, x_5) + c(x_5, x_7) \,]$$
$$= \min [\, (17 + 39) \,] = 56.$$
$$\pi(2, x_7) = x_5.$$

This is the end of iteration 2.

Iteration 3 (q = 3)

$$f(3, x_2) = \min [\, f(1, x_4) + c(x_4, x_2),\ f(1, x_5) + c(x_5, x_2),\ f(1, x_7) + c(x_7, x_2) \,]$$
$$= \min [\, (14 + 36),\ (17 + 25),\ (22 + 35) \,] = 42.$$
$$p(3, x_2) = x_5.$$
$$\phi(3, x_2) = \min [\, f(1, x_4) + c(x_4, x_2),\ f(1, x_7) + c(x_7, x_2) \,]$$
$$= \min [\, (14 + 36),\ (22 + 35) \,] = 50.$$
$$\pi(3, x_2) = x_4.$$

In case of f(3, x_4), some of p(q-q_i, x_j) are the same as x_i, some of them are not the same, i.e. p(2, x_5) = x_4 and p(2, x_2) = p(2, x_6) = p(2, x_9) $\neq$ x_4. In order to satisfy the conditions of (51a), we should use the first recursion of (51a) for the terms f(2, x_2) + c(x_2, x_4), f(2, x_6) + c(x_6, x_2) and f(2, x_9), and should use the second recursion for vertices x_5 and x_7 like ϕ(2, x_5) + c(x_5, x_2) and ϕ(2, x_7) + c(x_7, x_2). Then we can compute f(3, x_4) and ϕ(3, x_4) as follows :

$$f(3, x_4) = \min [\, f(2, x_2) + c(x_2, x_4),\ \phi(2, x_5) + c(x_5, x_4),\ f(2, x_6) + c(x_6, x_2),$$
$$\phi(2, x_7) + c(x_7, x_4),\ f(2, x_9) + c(x_9, x_4) \,]$$
$$= \min [\, (28 + 36),\ (61 + 15),\ (18 + 31),\ (56 + 35),\ (30 + 17) \,] = 47.$$
$$p(3, x_4) = x_9.$$
$$\phi(3, x_4) = \min [\, f(2, x_2) + c(x_2, x_4),\ \phi(2, x_5) + c(x_5, x_4),\ f(2, x_6) + c(x_6, x_2),$$
$$\phi(2, x_7) + c(x_7, x_4) \,]$$
$$= \min [\, (28 + 36),\ (61 + 15),\ (18 + 31),\ (56 + 35) \,] = 49.$$
$$\pi(3, x_4) = x_6.$$

$$\begin{aligned}
f(3,\, x_5) &= \min\, [\, f(2,\, x_2) + c(x_2,\, x_5),\, \phi(2,\, x_4) + c(x_4,\, x_5),\, f(2,\, x_6) + c(x_6,\, x_5), \\
&\qquad\quad f(2,\, x_7) + c(x_7,\, x_5),\, f(2,\, x_9) + c(x_9,\, x_5)\,] \\
&= \min\, [\, (28 + 25),\, (56 + 15),\, (18 + 29),\, (48 + 39),\, (30 + 35)\,] = 47. \\
p(3,\, x_5) &= x_6. \\
\phi(3,\, x_5) &= \min\, [\, f(2,\, x_2) + c(x_2,\, x_5),\, \phi(2,\, x_4) + c(x_4,\, x_5),\, f(2,\, x_7) + c(x_7,\, x_5), \\
&\qquad\quad f(2,\, x_9) + c(x_9,\, x_5)\,] \\
&= \min\, [\, (28 + 25),\, (56 + 15),\, (48 + 39),\, (30 + 35)\,] = 53. \\
\pi(3,\, x_5) &= x_2. \\
f(3,\, x_6) &= \min\, [\, f(1,\, x_4) + c(x_4,\, x_6),\, f(1,\, x_5) + c(x_5,\, x_6),\, f(1,\, x_7) + c(x_7,\, x_6)\,] \\
&= \min\, [\, (14 + 31),\, (17 + 29),\, (22 + 16)\,] = 38. \\
p(3,\, x_6) &= x_7. \\
\phi(3,\, x_6) &= \min\, [\, f(1,\, x_4) + c(x_4,\, x_6),\, f(1,\, x_5) + c(x_5,\, x_6)\,] \\
&= \min\, [\, (14 + 31),\, (17 + 29)\,] = 45. \\
\pi(3,\, x_6) &= x_4. \\
f(3,\, x_7) &= \min\, [\, f(2,\, x_2) + c(x_2,\, x_7),\, f(2,\, x_4) + c(x_4,\, x_7),\, f(2,\, x_5) + c(x_5,\, x_7), \\
&\qquad\quad f(2,\, x_6) + c(x_6,\, x_7),\, f(2,\, x_9) + c(x_9,\, x_7)\,] \\
&= \min\, [\, (28 + 35),\, (32 + 34),\, (29 + 39),\, (18 + 16),\, (30 + 32)\,] = 34. \\
p(3,\, x_7) &= x_6. \\
\phi(3,\, x_7) &= \min\, [\, f(2,\, x_2) + c(x_2,\, x_7),\, f(2,\, x_4) + c(x_4,\, x_7),\, f(2,\, x_5) + c(x_5,\, x_7), \\
&\qquad\quad f(2,\, x_9) + c(x_9,\, x_7)\,] \\
&= \min\, [\, (28 + 35),\, (32 + 34),\, (29 + 39),\, (30 + 32)\,] = 62. \\
\pi(3,\, x_7) &= x_9. \\
f(3,\, x_9) &= \min\, [\, f(1,\, x_4) + c(x_4,\, x_9),\, f(1,\, x_5) + c(x_5,\, x_9),\, f(1,\, x_7) + c(x_7,\, x_9)\,] \\
&= \min\, [\, (14 + 17),\, (17 + 35),\, (22 + 32)\,] = 31. \\
p(3,\, x_9) &= x_4. \\
\phi(3,\, x_9) &= \min\, [\, f(1,\, x_5) + c(x_5,\, x_9),\, f(1,\, x_7) + c(x_7,\, x_9)\,] \\
&= \min\, [\, (17 + 35),\, (22 + 32)\,] = 52.
\end{aligned}$$

$\pi(3, x_9) = x_5$.

This is the end of iteration 3.

We can compute the rest (from iteration 4 to 15) as the same way. We then have tables of the results of the full computation for f(q, x), p(q, x), ϕ(q, x) and π(q, x) as shown below :

Table 3.4a f(q, x)

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	28	42	38	53	62	54	63	81	73	82	98	106	99	113
x_3	∞	∞	21	40	48	28	47	50	59	66	80	89	73	90	94
x_4	14	32	47	40	52	56	54	71	63	82	85	94	101	99	116
x_5	17	29	47	37	55	59	56	69	81	78	82	96	103	101	114
x_6	∞	18	38	48	34	43	67	53	62	80	86	79	93	101	98
x_7	22	48	34	27	51	49	46	79	69	79	91	86	94	91	108
x_8	∞	∞	15	34	37	34	53	61	62	60	74	81	79	96	102
x_9	∞	30	31	49	39	57	65	46	65	68	77	84	101	109	91

Table 3.4b p(q, x)

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	-	x_1	x_5	x_6	x_8	$x_{5,7}$	x_6	x_6	$x_{5,7}$	x_6	x_6	x_8	x_6	x_6	x_6
x_3	-	-	x_1	x_4	$x_{6,9}$	x_8	$x_{7,8}$	x_8	x_8	x_7	$x_{8,9}$	x_{479}	x_8	x_8	x_8
x_4	x_1	x_5	x_9	x_8	x_5	x_9	x_3	x_5	x_9	x_9	$x_{8,9}$	x_9	x_9	x_3	x_3
x_5	x_1	x_4	x_6	x_8	x_4	x_8	x_8	x_4	x_9	x_4	x_8	x_8	x_8	x_8	x_4
x_6	-	x_1	x_7	x_2	x_8	x_7	x_7	x_8	x_7	$x_{3,8}$	x_7	x_8	x_8	x_8	x_8
x_7	x_1	x_4	x_6	x_8	x_8	x_8	x_8	x_8	x_6	x_3	x_8	x_8	x_8	x_8	x_8
x_8	-	-	x_1	x_7	x_6	x_3	$x_{3,7}$	x_3	x_{367}	x_7	$x_{3,9}$	$x_{6,7}$	x_3	x_3	x_3
x_9	-	x_1	x_4	x_4	x_3	x_4	x_8	x_3	x_3	x_3	x_3	x_3	x_3	x_8	x_3

The above tableau for f(q, x) gives the values of $f(\overline{Q}, x_i)$, i.e. $f(15, x_i)$, for all x_i and can be used in the expression (8) to obtain the lower bounds.

Table 3.4c $\phi(q, x)$

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	∞	50	54	58	72	75	72	87	94	99	100	107	119	117
x_3	∞	∞	∞	42	58	49	66	61	69	80	83	92	106	108	106
x_4	∞	56	49	47	59	62	59	73	76	85	92	97	106	104	121
x_5	∞	61	53	58	56	63	65	75	82	84	97	100	108	110	118
x_6	∞	∞	45	58	51	53	70	58	72	91	89	95	107	102	103
x_7	∞	56	62	41	60	50	48	86	70	88	96	100	95	93	110
x_8	∞	∞	∞	39	51	46	67	67	81	79	77	82	91	105	112
x_9	∞	∞	52	63	43	58	69	62	71	88	90	88	105	110	107

Table 3.4d $\pi(q, x)$

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	-	-	x_4	x_5	x_6	x_8	x_8	x_8	x_6	x_5	x_4	x_6	x_5	x_8	x_8
x_3	-	-	-	x_7	x_4	x_9	x_4	x_9	x_7	x_8	x_6	x_6	x_9	x_9	$x_{7,9}$
x_4	-	x_7	x_6	x_3	x_8	x_8	x_8	x_3	x_3	x_3	x_3	x_5	x_8	x_8	x_8
x_5	-	x_7	x_2	x_3	x_8	x_6	x_3	x_8	x_6	x_8	x_4	x_4	x_6	x_3	x_8
x_6	-	-	x_4	x_5	x_3	x_8	x_8	x_3	x_8	x_9	x_3	x_7	x_7	x_7	x_3
x_7	-	x_5	x_9	x_3	x_3	x_6	x_3	$x_{3,6}$	x_3	x_6	x_6	x_3	x_6	x_3	x_3
x_8	-	-	-	$x_{4,5}$	x_5	x_7	x_6	x_9	x_4	$x_{3,4}$	x_6	x_3	x_7	$x_{6,9}$	$x_{7,9}$
x_9	-	-	x_5	x_6	x_8	x_3	x_4	x_8	x_4	x_4	x_8	x_8	x_8	x_3	x_8

Using expression (8), we can now compute the direct lower bound B1 as follows :

$$\begin{aligned} B1 &= \min_{x_i} [\, f(\overline{Q}, x_i) + c(x_i,x_1) \,] \\ &= \min \, [\, \underset{x_2}{(113+28)},\ \underset{x_3}{(94+21)},\ \underset{x_4}{(116+14)},\ \underset{x_5}{(114+17)}, \\ &\qquad \underset{x_6}{(103+18)},\ \underset{x_7}{(98+22)},\ \underset{x_8}{(102+15)},\ \underset{x_9}{(\ 91+30)} \,] \\ &= 115, \text{ with the minimum obtained for } x_3. \end{aligned}$$

This bound is improved compared with the bound of B1= 82 obtained from the previous section (direct bound with loops). By backtracking through tableau f(q, x), p(q, x), ϕ(q, x) and π(q, x), we can obtain the q-path corresponding to the above value of 115. Two alternative paths are obtained from the results of the backtracking as follows :

Alternative path 1 : $x_1 - x_3 - x_8 - x_7 - x_6 - x_8 - x_3 - x_1$

Alternative path 2 : $x_1 - x_3 - x_8 - x_6 - x_7 - x_8 - x_3 - x_1$

The above two alternatives for the present example are almost identical and the q-paths corresponding to the above value of 115 are as shown in Fig 3.7a and Fig 3.7b.

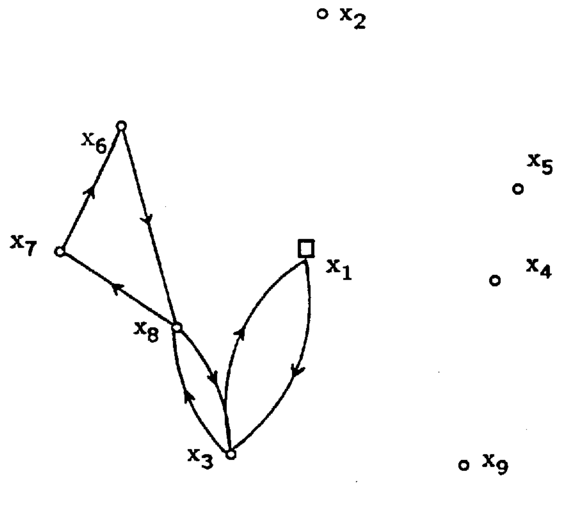


Figure 3.7a Alternative 1 q-path corresponding to B1 = 115.

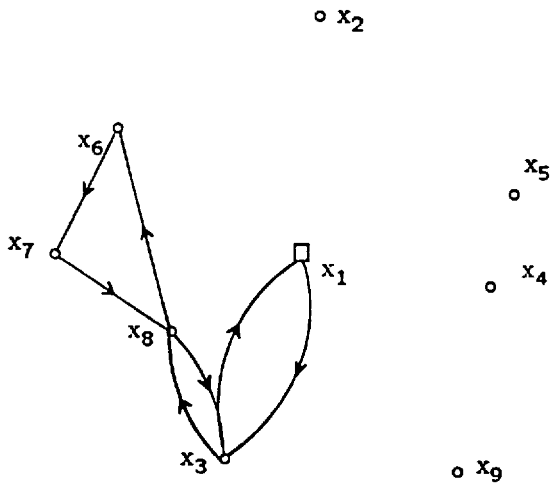


Figure 3.7b Alternative 2 q-path corresponding to B1 = 115.

3.6.4 Bound B2 (with no loops)

Let us compute $\psi(q, x_i)$ from expression (15).

Thus :
$$\psi(2, x_2) = \min [\, f(2, x_2) + f(15 - 2 + 2, x_2) \,], \text{ since } p(2, x_2) \neq p(15, x_2)$$
$$= \min [\, f(2, x_2) + f(15, x_2) \,]$$
$$= 28 + 113 = 141.$$
$$\psi(4, x_2) = \min [\, f(4, x_2) + \phi(13, x_2), \phi(4, x_2) + f(13, x_2) \,],$$
$$\text{since } p(4, x_2) = p(13, x_2)$$
$$= \min [\, (38 + 107), (54 + 106) \,]$$
$$= 145.$$
$$\psi(5, x_2) = \min [\, f(5, x_2) + \phi(12, x_2), \phi(5, x_2) + f(12, x_2) \,]$$
$$= \min [\, (53 + 100), (58 + 98) \,]$$
$$= 153.$$
$$\psi(6, x_2) = \min [\, f(6, x_2) + f(11, x_2) \,]$$
$$= 62 + 82$$
$$= 144.$$

etc. The tableau of $\psi(q, x_i)$ is computed as shown below :

Table 3.5 $\psi(q, x)$

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	$\underline{141}$	$\underline{141}$	145	153	144	148	144	144	148	144	153	145	$\underline{141}$	$\underline{141}$
x_3	∞	∞	115	130	121	117	130	116	128	116	130	117	121	130	115
x_4	130	131	150	134	137	141	117	144	117	141	137	134	150	131	130
x_5	131	130	150	137	137	137	137	144	137	137	137	137	150	130	131
x_6	∞	116	139	141	129	132	147	115	115	147	132	129	141	139	116
x_7	130	139	128	127	147	128	115	165	115	128	147	127	128	139	130
x_8	∞	∞	117	130	116	115	130	121	143	121	130	115	116	130	117
x_9	∞	121	140	150	127	134	133	117	117	133	134	127	150	140	121

As shown in the previous example, we can compute b_{iq} (= $\psi(q, x_i)$), and then obtain an indirect lower bound (B2) for the TSP as follows :

$$\begin{aligned} B2 &= \max_{x_i} [\, \min_{q} b_{iq} \,] \\ &= \max_{x_i} [\, 141,\ 115,\ 117,\ 130,\ 115,\ 115,\ 115,\ 117 \,] \\ &= 141, \text{ as indicated in Table 3.5.} \end{aligned}$$

In this example B2 = 141, which is better than the value of 118 obtained for B2 when loops where allowed. The minimum is shown underlined in the above matrix for $x_i = x_2$ and q = 14 or 15. The value of 141 is obtained from :

$$f(15, x_2) + f(2, x_2) \text{ or } f(14, x_2) + f(3, x_2) .$$

The two alternative paths corresponding to each one of these above two expressions can be obtained by backtracking through tableau p(q, x) and $\pi(q, x)$, and are given as below :

Alternative path 1 : $x_1 - x_8 - x_3 - x_9 - x_8 - x_6 - x_2 - x_1$

Alternative path 2 : $x_1 - x_8 - x_3 - x_7 - x_8 - x_6 - x_2 - x_5 - x_1$

The through-circuits are shown below in Fig. 3.8a and Fig. 3.8b.

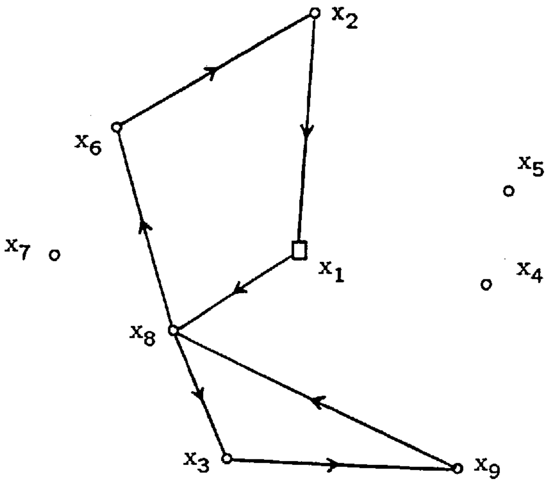


Figure 3.8a Alternative 1 q-path corresponding to B2 = 141.

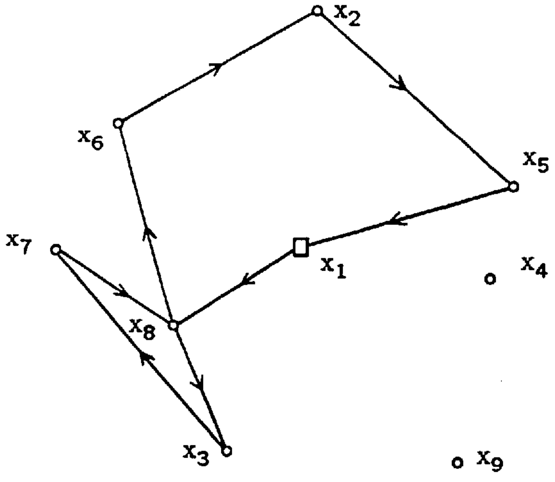


Figure 3.8b Alternative 2 q-path corresponding to B2 = 141.

3.6.5 Bounds B1 and B2 with no loops and applying the penalty procedure

Since the penalty procedure for the indirect bound B2 is the same with that for the direct bound B1, we will illustrate the procedures just for B1 with no loops.

Step 0 : (Initialization).

$ZL^*= 0$. $ZU^* = 194$ (the best solution - obtained by a heuristic - to this example so far).

$\alpha = 2.0$ and KOUNT = 0.

Step 1 : (Initialization).

$\lambda_i = 0, i = 1, \dots , 9$ and $d_i = 0, i = 1, \dots , 9$.

Step 2 : (Calculation of lower bound).

LB = 115.0 (the bound B1 with no loops derived in Section 3.6.3).

$ZL = B1(\lambda^o) - 2\sum\lambda_i = 115.0 - 0 = 115.0$

Since $ZL^* < ZL$, $ZL^* = ZL = 115.0$,

KOUNT = 1, and go to step 3.

Step 3 : (Backtracking).

The q-path corresponding to the value of lower bound, 115.0 is as :

$H(o)$: $x_1 - x_3 - x_8 - x_6 - x_7 - x_8 - x_3 - x_1$ (refer to Fig. 3.7b).

$d_i = (2, 0, 4, 0, 0, 2, 2, 4, 0)$ for $i = 1, \dots , 9$.

Since $d_i \neq 2$ for all i, go to step 4.

Step 4 : (Penalties).

Compute λ_i as follows :

$ZU^* - ZL = 194.0 - 115.0 = 79.0$

$$\sum_{j=2}^{9}(d_j - 2)^2 = (0 - 2)^2 + (4 - 2)^2 + (0 - 2)^2 + (0 - 2)^2 + (2 - 2)^2 + (2 - 2)^2 + (0 - 2)^2 + (0 - 2)^2 = 24$$

$$\lambda_2 = 0 + 2.0 \cdot \frac{79}{24} \cdot (0 - 2) \cdot \frac{2}{3} = - \frac{79}{9}$$

$$\lambda_3 = 0 + 2.0 \cdot \frac{79}{24} \cdot (4 - 2) \cdot \frac{3}{3} = \frac{79}{6}$$

$$\lambda_4 = 0 + 2.0 \cdot \frac{79}{24} \cdot (0 - 2) \cdot \frac{1}{3} = - \frac{79}{18}$$

$$\lambda_5 = 0 + 2.0 \cdot \frac{79}{24} \cdot (0 - 2) \cdot \frac{1}{3} = - \frac{79}{18}$$

$$\lambda_6 = 0 + 2.0 \cdot \frac{79}{24} \cdot (2 - 2) \cdot \frac{2}{3} = 0$$

$$\lambda_7 = 0 + 2.0 \cdot \frac{79}{24} \cdot (2 - 2) \cdot \frac{1}{3} = 0$$

$$\lambda_8 = 0 + 2.0 \cdot \frac{79}{24} \cdot (4 - 2) \cdot \frac{3}{3} = \frac{79}{6}$$

$$\lambda_9 = 0 + 2.0 \cdot \frac{79}{24} \cdot (0 - 2) \cdot \frac{2}{3} = - \frac{79}{9}$$

(Note that $\sum_{i \in H(o)} q_i = \sum_{j \notin H(o)} q_j$).

Step 5 : (Udating the cost matrix).

Since $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$, the results of modifying are as follows :

$$c'_{11} = c_{11} + \lambda_1 + \lambda_1 = c_{11} + 0 + 0 = c_{11}$$

$$c'_{12} = c_{12} + \lambda_1 + \lambda_2 = 28.0 + 0 - \frac{79}{9} \doteq 19.22$$

$$c'_{13} = c_{13} + \lambda_1 + \lambda_3 = 21.0 + 0 + \frac{79}{6} \doteq 34.17$$

....

$$c'_{98} = c_{98} + \lambda_9 + \lambda_8 = 28.0 - \frac{79}{9} + \frac{79}{6} \doteq 32.39$$

Table 3.6 New cost matrix $[c'_{ij}]$

$x_j \backslash x_i$	1	2	3	4	5	6	7	8	9
1	-	19.22	34.17	9.61	12.61	12.61	22.00	28.17	21.22
2	19.22	-	51.39	22.83	11.83	11.22	26.22	42.39	32.44
3	34.17	51.39	-	34.78	45.78	43.17	33.17	39.33	22.39
4	9.61	22.83	34.78	-	6.22	26.61	29.61	33.78	3.83
5	12.61	11.83	45.78	6.22	-	24.61	34.61	30.78	21.83
6	18.00	11.22	43.17	26.61	24.61	-	16.00	32.17	36.22
7	22.00	26.22	33.17	29.61	34.61	16.00	-	25.17	23.22
8	28.17	42.39	39.33	33.78	30.78	32.17	25.17	-	32.39
9	21.22	32.44	22.39	3.83	21.83	36.22	23.22	32.39	-

Step 6 : (Computation of f, p, ϕ and π from the relaxed recursion).

The results of computation for f(q, x), p(q, x), ϕ(q, x) and π(q, x) with the new cost matrix above are as follows :

Table 3.7a f(q, x)

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	19.2	24.4	27.7	27.7	45.9	43.1	56.3	63.9	73.1	75.0	84.8	98.6	100.	106.9
x_3	∞	∞	34.2	44.4	43.6	35.8	45.1	68.3	63.5	76.7	73.5	90.2	95.9	105.	105.4
x_4	9.6	18.8	25.1	37.3	41.5	47.3	56.9	66.7	78.7	79.2	88.8	97.4	109.6	111.	119.6
x_5	12.6	15.8	31.1	31.3	41.1	56.7	62.9	63.2	72.9	84.9	88.2	95.1	103.6	113.	120.1
x_6	∞	18.0	36.2	30.4	35.7	38.9	57.1	54.3	67.6	79.8	86.0	86.2	96.0	112.	111.2
x_7	22.0	39.2	34.0	36.7	45.9	51.7	54.9	73.1	70.3	83.6	95.8	102.0	102.2	112.	128.7
x_8	∞	∞	28.2	43.4	46.6	45.8	55.1	67.8	71.7	80.1	83.5	95.5	105.9	112.	115.4
x_9	∞	21.2	13.4	22.7	45.9	41.1	53.3	51.1	67.8	73.6	82.8	83.0	99.7	105.	113.4

Table 3.7b p(q, x)

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	-	x_1	x_5	x_5	x_9	x_5	x_5	x_6	x_6	x_5	x_5	x_6	x_5	x_5	x_5
x_3	-	-	x_1	x_4	x_9	x_9	x_9	x_9	x_9	x_9	x_9	x_9	x_9	x_9	x_9
x_4	x_1	x_5	x_9	x_5	$x_{5,9}$	x_5	x_9	x_9	x_9	x_5	x_9	x_9	x_5	x_5	x_5
x_5	x_1	x_4	x_2	x_4	x_2	x_4	$x_{4,9}$	x_4	x_9	$x_{2,4}$	x_4	x_4	x_4	x_2	x_4
x_6	-	x_1	x_4	x_2	x_2	x_2	x_2	x_2	x_2	x_2	x_2	x_2	x_2	x_2	x_2
x_7	x_1	x_4	x_6	x_9	x_9	x_6	x_6	x_6	x_6	x_6	x_6	x_6	x_6	x_6	x_6
x_8	-	-	x_1	$x_{4,5}$	x_5	x_9	x_9	x_6	x_6	x_7	x_9	x_7	x_9	x_3	x_9
x_9	-	x_1	x_4	x_4	x_4	x_4	x_4	x_4	x_4	x_4	x_4	x_4	x_4	x_4	x_4

Table 3.7c $\phi(q, x)$

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	∞	∞	32.4	29.2	47.4	51.7	60.9	68.6	74.8	83.6	91.6	101.4	101.6	115.	124.0
x_3	∞	∞	∞	55.2	53.6	59.8	69.8	76.3	82.1	88.1	97.5	103.5	113.9	122.	129.4
x_4	∞	51.6	42.1	38.3	50.5	56.7	63.9	69.7	78.9	81.9	95.8	101.6	110.6	113.	120.7
x_5	∞	56.6	42.6	35.3	44.5	57.7	63.5	72.7	72.9	89.6	95.4	103.4	104.8	116.	127.3
x_6	∞	∞	37.2	40.4	49.7	52.7	61.9	93.9	80.3	87.3	90.3	105.8	112.8	119.	122.2
x_7	∞	47.2	44.4	50.7	46.4	69.1	64.3	77.6	74.3	91.0	96.8	106.0	106.2	123.	133.1
x_8	∞	∞	∞	47.2	50.2	58.8	61.8	71.1	73.5	84.4	86.5	99.7	108.7	115.	118.4
x_9	∞	∞	34.4	37.7	52.9	53.1	62.9	74.9	78.1	85.0	93.6	106.8	110.0	117.	125.4

Table 3.7d $\pi(q, x)$

x\q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_2	-	-	x_4	x_6	x_4	x_6	x_6	x_5	x_5	x_9	x_6	x_5	x_6	x_9	x_6
x_3	-	-	-	x_7	x_4	x_4	x_7	x_4	x_4	x_7	x_6	x_7	x_4	x_8	x_6
x_4	-	x_7	x_2	x_9	x_2	x_9	x_5	x_5	x_5	x_9	x_5	x_5	x_9	x_9	x_9
x_5	-	x_7	x_6	x_9	x_4	x_2	x_2	x_2	x_4	x_9	x_2	x_2	x_9	x_4	x_2
x_6	-	-	x_5	x_5	x_9	x_7	x_7	x_4	x_7	x_5	x_7	x_4	x_5	x_5	x_7
x_7	-	x_5	x_9	x_2	x_6	x_9	x_9	x_9	x_9	x_7	x_9	x_3	x_9	x_9	x_2
x_8	-	-	-	x_7	x_6	x_4	x_7	x_5	x_9	x_3	x_6	x_6	x_7	x_7	x_6
x_9	-	-	x_5	x_5	x_5	x_5	x_5	x_7	x_7	x_5	x_7	x_5	x_5	x_5	x_5

Go to step 1.

At this stage (KOUNT = 1), we obtain a updated lower bound $B1(\lambda^1)$ from expression (8) and Table 3.7a as follows :

$$B1(\lambda^1) = \min_{x_i} [f(\overline{Q}, x_i) + c(x_i, x_1)]$$

$$\begin{aligned} &= \min \ [(106.9 \underset{x_2}{+} 19.22),\ (105.4 \underset{x_3}{+} 34.17),\ (119.6 \underset{x_4}{+} 9.61),\ (120.1 \underset{x_5}{+} 12.61), \\ &\qquad (111.2 \underset{x_6}{+} 12.61),\ (128.7 \underset{x_7}{+} 22.0),\ (115.4 \underset{x_8}{+} 28.17),\ (113.4 \underset{x_9}{+} 21.22)\ ] \\ &= 129.21, \text{ with the minimum obtained for } x_4. \end{aligned}$$

In step 2, since $ZL^* = 115.0 < ZL = B1(\lambda^1) - 2\sum\lambda_i = 129.21$, $ZL^*=129.21$

After 6 iterations of the same procedures, we obtain $d_i = 2$ (for all i), and $ZL^*=152.0$ Because this is a feasible solution to the TSP, this value is an optimal solution value. So, from backtracking, the optimal q-path corresponding to this value is as follows :

$$H(\lambda^*) : x_1 - x_5 - x_2 - x_6 - x_7 - x_8 - x_3 - x_9 - x_4 - x_1.$$

3.7 Computational results for bound calculations

This section deals with the computational performance of algorithms to obtain

Table 3.8 Problem description

Problem	Number of vertices	Total requirement*	Source
1	9	15	Given as an example
2	10	28	Test problem 1 in Appendix A
3	11	93	Christofides *et. al.* [1981 a]
4	15	56	Test problem 2 in Appendix A
5	20	80	Test problem 3 in Appendix A
6	30	105	Test problem 5 in Appendix A
7	40	140	Test problem 7 in Appendix A
8	50	147	Test problem 9 in Appendix A

* This is the sum of the values of q_i. These values are generated randomly except for problem 3 where the values of customer demand in the VRP origins of that problem are used as the q_i for the TSP.

bounds for the TSP. Eight test problems (see Table 3.8 and Appendix A) are used for tests ranging from 9 to 50 vertices. All of these problems are randomly generated, and they are symmetric and uniformly distributed.

Fig. 3.9 shows the bound ascents of the direct lower bound (B1), the indirect lower bound (B2) and the 2-paths bound (B3) for the 50 vertex TSP. Table 3.9 shows a comparison (values and time) of the three bounds after the ascent (at 10, 20 and 30 iterations) for eight test problems.

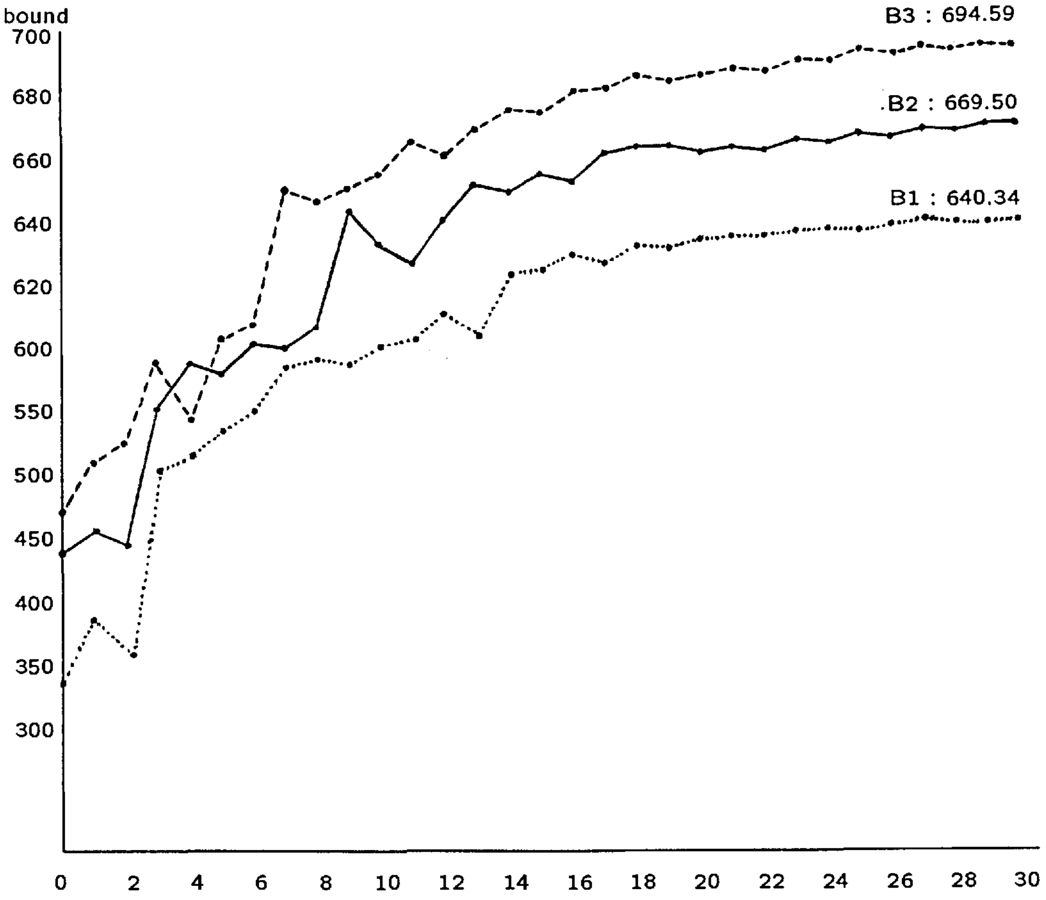


Figure 3.9 The bound ascents for a 50 vertex TSP.

Table 3.9 A comparison of the three bounds : values & times at 10, 20 & 30 iterations

Prob.	B1						B2						B3					
	Number of iterations						Number of iterations						Number of iterations					
	10		20		30		10		20		30		10		20		30	
	values	time	values	time	values	time	values	time	values	time	values	time	values	time	values	time	values	time
1[a]	152.0*	0:01	-	-	-	-	152.0*	0:02	-	-	-	-	152.0*	0:04	-	-	-	-
2[b]	219.0*	0:01	-	-	-	-	219.0*	0:01	-	-	-	-	219.0*	0:04	-	-	-	-
3[c]	155.2*	0:18	-	-	-	-	155.2*	0:16	-	-	-	-	155.2*	0:30	-	-	-	-
4[d]	278.3	0:24	288.7	0:48	291.0*	0:57	283.3	0:27	291.0*	0:53	-	-	276.2	0:48	289.0	1:37	291.0*	2:05
5	320.2	1:09	332.6	2:17	336.2	3:26	324.8	1:13	339.2	2:25	341.8	3:38	322.4	2:12	334.9	4:24	337.4	6:35
6	491.1	3:37	506.8	7:14	509.4	10:51	501.2	3:45	509.8	7:31	511.6	11:16	496.7	7:21	507.8	14:42	511.8	22:02
7	505.2	8:54	536.7	17:44	542.9	26:42	546.0	9:08	569.9	18:16	575.1	27:25	584.8	12:01	601.9	24:02	606.3	52:04
8	607.2	14:53	634.8	29:45	640.3	44:38	632.1	15:10	662.3	30:19	669.5	45:29	656.7	00:10	688.6	00:19	694.6	00:29

* : The optimal solution value obtained without embedding into tree search algorithm.

a : Problem 1 has the optimal solution value at 7th iteration for B1, at 9th for B2 and at 9th for B3.

b : Problem 2 has the optimal solution value at 3rd iteration for B1, at 1st for B2 and at 4th for B3.

c : Problem 3 has the optimal solution value at 10th iteration for B1, at 7th for B2 and at 7th for B3.

d : Problem 4 has the optimal solution value at 24th iteration for B1, at 20th for B2 and at 25th for B3.

3.8 The tree search algorithm for the TSP

We present two tree search algorithms and the computational performance of those algorithms to obtain optimal solution values for the TSP.

The basis of a tree search algorithm is to divide the set of all possible tours into smaller and smaller subsets and to calculate for each subset a lower bound on the cost of the best tour therein. The object of calculating lower bounds is that firstly to be used as guidance for the partitioning of the subsets and secondly to limit the search and also to identify the optimal tour. In constructing such search trees it is necessary to consider a branching strategy. Fig. 3.10 shows a diagram of the basic tree search for the TSP using some branching strategy. We assume the vertices are (x_1, x_2, x_3, ...).

In Fig. 3.10 LB_0 is the initial lower bound, ZU is the upper bound at the root node and we assume that the lower bounds LB_0, LB_1, LB_2, LB_5 and LB_6 on nodes 0, 1, 2, 5 and 6

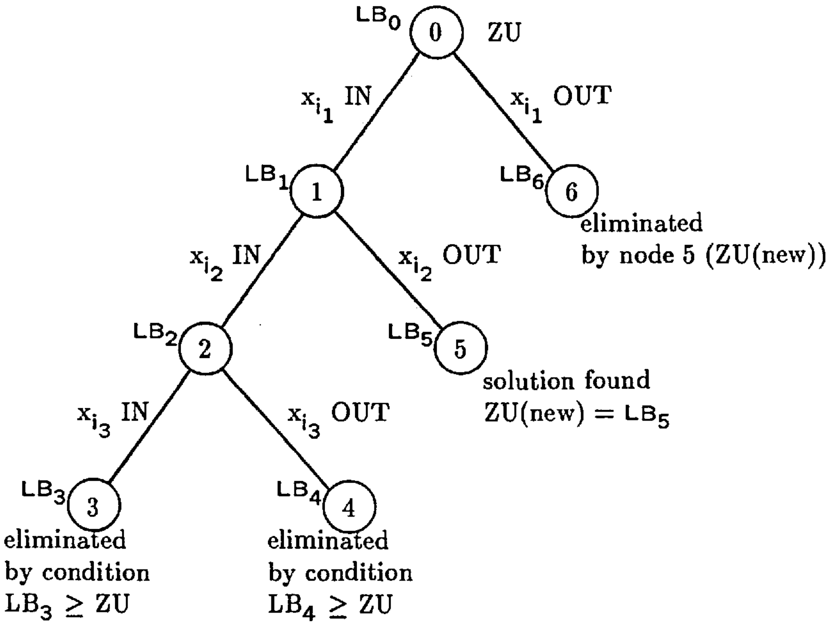


Figure 3.10 Tree of the tree search method for the TSP.

respectively are less than ZU. We also assume that LB_3 and LB_4 are greater than ZU (i.e. nodes 3 & 4 are eliminated by the bounds). If the solution corresponding to LB_5 is found to be feasible, a new solution is obtained and ZU can then be updated when this node is reached. If we now assume that LB_6 although previously less than the initial value of ZU (as stated earlier), is now greater than the updated value of ZU, then node 6 is now eliminated. The tree search proceeds as follows :

Customer x_{i_1} is chosen as the first one for forward branching producing nodes 1 (x_{i_1} IN) and 6 (x_{i_1} OUT). Since LB_1 is less than ZU and not feasible at node 1, x_{i_2} is chosen for the next branching producing nodes 2 (x_{i_2} IN) and 5 (x_{i_2} OUT). The partially completed path is now (x_1, x_{i_1}, x_{i_2}) so far. Similarly nodes 3 and 4 are produced by branching on x_{i_3}. Since LB_3 is greater than ZU, the forward branching is stopped at node 3. Then, backtracking occurs by rejecting the customer x_{i_3} and examining node 4. Since LB_4 is greater than ZU, backtracking continues to node 5. Since LB_5 is less than ZU and the bound corresponds to a feasible solution, this value becomes a new upper bound ZU(new). Backtracking now continues to node 6. Since LB_6 is greater than ZU(new) and there are no more tree nodes for branching, the search tree is terminated at node 6. Thus, the optimal solution value is ZU(new) and the optimal solution was found at node 5. (Note that this optimal solution starts as x_1, x_{i_1}, x_α, ... , where $x_\alpha \neq x_{i_2}$.)

3.8.1 The branching strategy for the TSP

The branching strategy (i.e. deciding which vertex to examine next) is based on arcs, i.e. an arc (x_i, x_j) is chosen for branching at a node of the search tree in order to extend a partially completed path (x_1, x_k, ... , x_i), and the alternative branching is to reject arc (x_i, x_j) as a possible extension of the path. In choosing the arc (x_i, x_j) for branching, (which means that vertex x_j is used to extend the route just after x_i), the following simple branching rule was applied.

Branching rule :

If a partially completed path must be extended, the vertex to be chosen for branching is the vertex nearest to the path end.

By choosing this vertex for the branching, we expect that the alternative branching (i.e. the rejection of the extension of the partially completed path with that vertex) will result in a higher lower bound, and may result in backtracking from the corresponding node.

If an arc (x_i, x_j) is chosen for branching at a node of the tree search in order to extend a partially completed path $(x_1, x_k, \dots , x_i)$, the costs are changed as follows :

$$c_{il} = c_{li} = \infty \text{ for } l = 2, \dots , n \text{ and } l \neq i, j.$$

For the alternative branching (which is to reject arc (x_i, x_j) as a possible extension of the partially completed path) the cost is changed as follows :

$$\text{If } c_{ij} = c_{ji} = \infty.$$

3.8.2 Fathoming

A partially completed path is "fathomed" (i.e. one can backtrack from it) when one of the following situations has arisen ;

(i) if a lower bound at a certain stage is greater than or equal to the current upper bound (the best solution so far),

or (ii) if the solution corresponding to the lower bound at a subproblem, is a feasible solution,

or (iii) if the bottom of the tree is reached and no unvisited vertices remain.

3.8.3 A Tree search algorithm for the TSP

We will show a tree search algorithm for the TSP using the lower bound B2 and the branching strategy in Section 3.8.1.

Description of the algorithm

Step 0 : (Initialization). Initialize level (LEVEL ; depth of tree) and total number of tree nodes (NTNODES) ; LEVEL = 0, NTNODES = 0. Let ZU (upper bound) be the value of the best solution so far, and set Zopt = ZU.

Step 1 : (Choose a vertex). Choose a vertex according to the branching strategy and call it IPICK (chosen vertex). If IPICK does not exist, then go to backtrack (step 4). Else, set NTNODES = NTNODES + 1, LEVEL = LEVEL + 1.

Step 2 : (Update the cost matrix). With the fixed customer (IPICK $\equiv x_j$) as an element of a partially completed path, change the cost matrix as :

$$c_{il} = c_{li} = \infty \text{ for } l = 2, \ldots, n \text{ and } l \neq i, j.$$

Step 3 : (Calculate the lower bound and check feasibility). Compute the updated lower bound with the updated cost matrix in step 2, and then check if the solution corresponding to the bound for this subproblem is feasible or not. Also check the lower bound (ZL) against the best upper bound (Zopt) so far. If a feasible solution is obtained and ZL<Zopt, then record the optimal solution value and the path corresponding to this value and set Zopt = ZL, and then go to step 4 to backtrack. If ZL<Zopt and the solution to this subproblem is not feasible, then go to step 1 to continue the branching. If ZL≥Zopt, then go to step 4 to backtrack.

Step 4 : (Backtrack). If the alternative to the current node at this LEVEL has not been examined, then go to examine that node : Set NTNODES= NTNODES+1, update $c_{ij} = c_{ji} = \infty$, and go to step 3. If the alternative to the current node has been examined, then set LEVEL= LEVEL-1. If LEVEL= 0, stop. Else, reset the costs c_{ij} and c_{ji} to their original values and repeat step 4.

In order to increase the efficiency of the tree search algorithm above, we can consider the following :

(i) since the initial lower bound ZL is close to the optimal solution (normally within 6%), the value k·ZL (where k= 1.3, for example) can be used as the initial upper bound with same confidence. In this case, because the quality of the initial upper bound is closer to the optimal solution value, the number of nodes in the tree search is reduced greatly. (Note that if the initial upper bound estimate is below the optimal solution value, no feasible solution will be obtained by the tree search.)

(ii) we can consider the gap between the lower bound and the upper bound at a certain node. For example, if we assume that the initial costs are integers, the feasible solution value should be integer. Therefore, if the value of a lower bound at a certain node is between the upper bound and the upper bound - 1 (i.e. $0 <$ Zopt-LB < 1), we can backtrack from this node.

3.9 Computational results

In Table 3.10 we show the size of test problems and the computational results (values and time) for the three bounds (B1, B2 and B3) using an IBM PS/2-70 386. All values for the bounds are obtained at the 30th iteration. From Table 3.10 we can see that bounds B2 and B3 are better than the direct bound B1. B3 is better than B2 but requires considerably longer time to compute (see Table 3.9).

Table 3.11 gives the computational performance (computing times and total number of nodes) of the algorithm using bound B2 for the first 6 test problems.

All computing times shown in Table 3.11 are times on the IBM PS/2-70 386 using the

Microsoft fortran 4.0 compiler, and all codes are in FORTRAN 77. In case of small size problems, i.e. problems 1 to 4, the optimal solution value was obtained without using a tree search algorithm, during the subgradient ascent procedure.

Table 3.10 Computational results : values & times (on IBM PS/2-70 386)

Problem	Number of vertices	Optimal solution value	Initial lower bound			
			B1	B2	B3	
			value	value	value	%[a]
1	9	152.0	152.0*	152.0*	152.0*	-
2	10	219.0	219.0*	219.0*	219.0*	-
3	11	155.2	155.2*	155.2*	155.2*	-
4	15	291.0	291.0*	291.0*	291.0*	-
5	20	343.0	336.2	341.8	337.4	12.6
6	30	537.0	509.4	511.6	511.8	47.8
7	40	-	542.9	575.1	606.3	44.1
8	50	-	640.3	669.5	694.6	16.0

* : The optimal solution value obtained without embedding into tree search algorithm.

a : % means the reduction percentage of the original problem and the two opt. TSP is used as the heuristic solution value.

From Fig. 3.11 it can be seen that the algorithm is not competitive with other existing algorithms that can be found in the literature. In view of the above results it was not considered useful to investigate better branching schemes, or to try to improve the algorithm in any way. Indeed, the bounding procedures are reported for the TSP only as an easy introduction to the use of state-space relaxation for the VRP where the power of this bounding procedure (in more complex problems) becomes apparent. However, it is worthwhile to note here, that we believe that the TSP with constraints (e.g. visit time windows, precedences, etc.) may also provide useful applications for state-space relaxation.

Table 3.11 Computational results :
total number of nodes & times (on IBM PS/2-70 386)

	Number of	Total	Algorithm	
Problem	vertices	requirement*	time**	nodes
1	9	15	0: 2	0(9)
2	10	28	0:16	0(7)
3	11	93	0: 1	0(1)
4	15	56	0:53	0(20)
5	20	80	22:23	6
6	30	105	62:20:51	634

( ) : The number of subgradient iterations before the optimal solution was obtained at the root node.

* : This is the value of $\overline{Q} = \sum_i q_i$. The values of q_i were distributed amongst the cities arbitrarily.

** hours : minutes : seconds

CHAPTER 4

BOUNDS FOR THE VRP FROM STATE-SPACE RELAXATION

4.1 Introduction

Consider a graph $G = (X, A)$ defined by the set X of its vertices and the set A of its arcs. Let $X' = \{x_i \mid i = 2, \dots , n\}$ be used for the set of n customers and let x_1 be the depot. $X = X' \cup \{x_1\}$. A customer x_i has an associated quantity q_i of some product to be delivered by a vehicle. We assume that M identical vehicles each of capacity Q are stationed at the depot.

The number of vehicles is assumed to be large enough for a feasible solution to exist. We further assume that the cost of the least cost path from every vertex x_i to every vertex x_j is given as c_{ij}. It is required that the total quantity on each vehicle route is less than or equal to Q. The objective in the VRP that is considered here, is to design feasible routes - one for each vehicle - in order to supply all of the customers and minimize the total cost of all the routes. For the purpose of this section the 'cost' c_{ij} mentioned above can be taken to be either travel distances or travel times between the customers. The VRP defined above is a generalization of the travelling salesman problem discussed earlier.

In this chapter, we will introduce a dynamic programming formulation and the corresponding relaxed recursion for the VRP. This relaxed recursion is then used to derive various lower bounds for the VRP. The bounds are improved by a procedure similar to subgradient optimization and the bound performance and quality is compared. The 'best' lower bound is used in the tree-search of the next chapter to produce a complete solution algorithm for the VRP.

4.2 A dynamic programming formulation for the VRP

Let f(m, S) be the least cost of supplying a set S of customers using only m vehicles and let v(S) be the solution to the TSP defined by the set S of customers and the depot x_1.

With the above definition, the dynamic programming recursion becomes :

$$f(m, S) = \min_{L \subseteq S} [\, f(m - 1,\ S - L) + v(L)\,] \tag{1}$$

$$\text{subject to} \quad \sum_{x_i \in S} q_i - (m - 1)\, Q \leq \sum_{x_i \in L} q_i \leq Q \tag{1a}$$

for m = 2, ... , M, and where $S \subseteq X'$ must satisfy

$$\left.\begin{array}{ll} & \overline{Q} - (M - m)\, Q \leq \sum_{x_i \in S} q_i \leq m \cdot Q \\ \text{where} & \overline{Q} = \sum_{x_i \in X'} q_i. \end{array}\right\} \tag{1b}$$

The restrictions on sets L and S are so as to avoid the computation of f(.) and v(.) for sets that can only lead to load - infeasible completions. For m = M only $S = X'$ need be considered, and the recursion is initialized by f(1, S) = v(S).

4.3 The relaxed recursion and the direct bound for the VRP

Let us consider the mapping function, $g(S) \equiv \alpha = \sum_{x_i \in S} \alpha_i$, where α_i are any 'weights' associated with the customers, and which can be chosen in any arbitrary fashion. We will choose $\alpha_i = q_i$ and denote α by q. The relaxed recursion (1) then becomes :

$$f(m, q) = \min_{q-(m-1)Q \leq q' \leq \min [q, Q]} [\ f(m\text{-}1, q\text{-}q') + \overline{v}(q')\ ], \tag{2}$$

where $\overline{v}(q')$ is given by an expression similar to (6) in Chapter 3, and for this case becomes the least cost of a circuit C for which $\sum_{x_i \in C} q_i = q'$. Since this is itself a hard problem, we will redefine $\overline{v}(q')$ to be, instead, a lower bound on the cost of such a circuit. Such a bound is derived as follows.

$$\overline{v}(q') = \min_{x_i \neq x_1} [\ f(q', x_i) + c(x_i, x_1)\ ], \tag{3}$$

where $f(q', x_i)$ is given by expression (13) in Chapter 3.

The above equation for $\overline{v}(q')$ can be rewritten as :

$$\overline{v}(q') = \min_{x_i} [\ \theta(q', x_i) + c(x_i, x_1)\ ], \tag{3'}$$

where $\theta(. , .)$ is the function f(. , .) for the TSP in the previous chapter, and which has been renamed in order to avoid confusion with the function f(m, q) of recursion (2). We will also use $\gamma(q', x)$ which is the vertex just prior to x on the path corresponding to $\theta(q', x)$ instead of $p(q', x)$.

From expressions (2) and (3′), the 'direct' lower bound (LB1) for the VRP is then given by

$$\text{LB1} = f(M, \overline{Q}) \tag{4}$$

4.4 The indirect bound for the VRP from 'through q - routes'

4.4.1 The through q - routes

Let W be the set of all possible loads (quantities) that could exist on any vehicle route, i.e.

$$W = \{ q \mid \sum_{i=2}^{n} q_i \xi_i = q \leq Q, \text{ for some } \xi, \ \xi_i \in \{0, 1\} \}.$$

Let the elements of W be ordered in ascending order and let $w = |W|$. We will denote by $q(l)$ the value of the lth element of W and by $\tau(q)$ that l^* so that $q(l^*) = q$. The total load on a path $\Phi = (x_1, x_{i_1}, x_{i_2}, \dots , x_{i_k})$ is defined as $\sum_{x_i \in \Phi - \{x_1\}} q_i$. (Note that Φ is not necessarily a simple path.)

Let $f_l(x_i)$ be the cost of the least cost path called a q-path with load $q(l)$. A q-path with the additional arc (x_i, x_1) is called a q-route and has cost $f'_l(x_i) = f_l(x_i) + c(x_i, x_1)$.

The path corresponding to $f_l(x_i)$ is not necessarily simple but it is not easy to impose the condition that no vertex is visited by the path more than once. On the other hand it is quite easy to impose the restriction that the path should not contain loops formed by three consecutive vertices. With this restriction imposed, a better bound can be calculated in much the same way as for the TSP in Chapter 3.

Let $p_l(x_i)$ be the vertex just prior to x_i on the path corresponding to $f_l(x_i)$, and let $\phi_l(x_i)$ be the least cost path from the depot to x_i with load $q(l)$ and with $\pi_l(x_i) \neq p_l(x_i)$, where $\pi_l(x_i)$ is the vertex just prior to x_i on the path corresponding to $\phi_l(x_i)$.

Fig. 4.1 shows two possible paths corresponding to $f_l(x_i)$ and $\phi_l(x_i)$.

For a given value of l, let $g(x_j, x_i)$ be the cost of the least cost path from x_0 to x_i with x_j just prior to x_i and without loops. Then, $g(x_j, x_i)$ is :

$$\left.\begin{aligned} g(x_j, x_i) &= f_{l'}(x_j) + c(x_j, x_i), \quad \text{if } p_{l'}(x_j) \neq x_i \\ &= \phi_{l'}(x_j) + c(x_j, x_i), \quad \text{otherwise} \end{aligned}\right\} \qquad (5)$$

where l' is such that $q(l') = q(l) + q_i$.

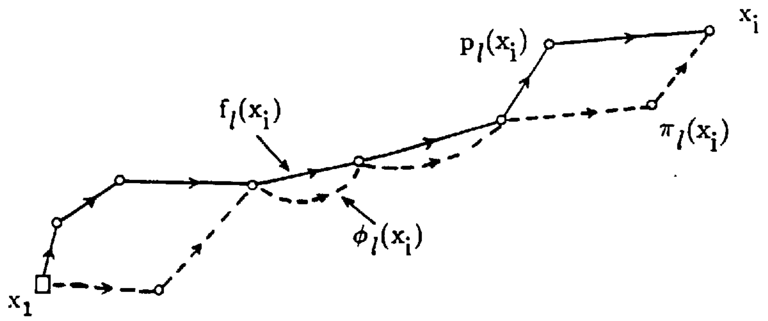


Figure. 4.1 A q-route with no loops.

Given the function g computed from (5), function f and ϕ can be computed for the given l as follows :

$$\left.\begin{aligned} f_l(x_i) &= \min_{x_j} [\, g(x_j, x_i) \,], \\ p_l(x_i) &= x_j^* \end{aligned}\right\} \quad (6a)$$

where x_j^* is the value of x_j corresponding to the above minimum.

$$\left.\begin{aligned} \phi_l(x_i) &= \min_{x_j \neq p_l(x_i)} [\, g(x_j, x_i) \,], \\ \pi_l(x_i) &= x_j^* \end{aligned}\right\} \quad (6b)$$

where x_j^* is the value of x_j corresponding to the above minimum.

From the above expression it is clear that the path corresponding to $f_l(x_i)$ has no end loops.

The initialization of the functions f, ϕ, p and π is as follows :

$$
\begin{aligned}
&f_l(x_i) = \phi_l(x_i) = \infty \text{ for } l \text{ such that } q(l) \neq q_i \\
&f_l(x_i) = c(x_1, x_i) \quad ; \quad p_l(x_i) = x_1 \text{ for } l \text{ such that } q(l) = q_i . \\
&\phi_l(x_i) = \infty
\end{aligned}
$$

Using the above expressions we can now obtain an indirect bound from the through q-routes. Let $\psi_l(x_i)$ be the value of the least cost route, without loops, starting from the depot, passing through x_i and finishing back at the depot with a total load $q(l)$. Such a route will be called a through q-route. $\psi_l(x_i)$ must be composed of either the two best q-paths to x_i whose total loads add up to $q(l)$, or a best path and a second-best path to x_i whose total loads add up to $q'(l)$.

$\psi_l(x_i)$ can then be computed as follows :

$$
\psi_l(x_i) = \min_{q_i \leq q' \leq \frac{1}{2}q(l')} [\, f_{\tau(q')}(x_i) + f_{\tau(q(l')\text{-}q')}(x_i) \,], \qquad \text{if } p_{\tau(q')}(x_i) \neq p_{\tau(q'(l)\text{-}q')}(x_i), \tag{7a}
$$

or

$$
\left.\begin{aligned}
\psi_l(x_i) = \min_{q_i \leq q' \leq \frac{1}{2}q(l')} [\![\min [\, & f_{\tau(q')}(x_i) + \phi_{\tau(q(l')\text{-}q')}(x_i), \\
& \phi_{\tau(q')}(x_i) + f_{\tau(q(l')\text{-}q')}(x_i)] \,]\!], \\
& \text{if } p_{\tau(q')}(x_i) = p_{\tau(q(l')\text{-}q')}(x_i).
\end{aligned}\right\} \tag{7b}
$$

We note that the computational effort involved in computing the q-path is linearly related to w. Thus, if w is large this operation can be quite time consuming.

Now we consider the calculation of the indirect bound from the computed values of $\psi_l(x_i)$.

4.4.2 The indirect bound (LB2) for the VRP

Let the total number of feasible single routes possible in the VRP be indexed by r = 1, ... , $\hat{r}$. Let the index set of customers in route r be M_r, the cost of the route be d_r and total load of the route be $K_r = \sum_{i \in M_r} q_i$. Let N_i be the index set of routes visiting customer x_i.

Let $y_r = 1$, if route r is in the optimal VRP solution,

$= 0$, otherwise.

The integer programming formulation of the VRP is as follows :

$$\text{Min} \quad \sum_{r=1}^{\hat{r}} d_r y_r, \tag{8}$$

$$\text{s.t.} \quad \sum_{r \in N_i} y_r = 1, \quad i = 2, \dots , n \tag{9}$$

$$\sum_{r=1}^{\hat{r}} y_r = M \tag{10}$$

$$y_r \in \{ 0, 1 \}. \tag{11}$$

Let us substitute y_r by the following expression in terms of new variables ξ_{ir} :

$$y_r = \frac{1}{K_r} \sum_{i \in M_r} \xi_{ir} q_i \tag{12}$$

The formulation of the VRP given by equations (8) - (11) now becomes :

$$\text{Min} \quad \sum_{r=1}^{\hat{r}} \frac{d_r}{K_r} \sum_{i \in M_r} \xi_{ir} q_i \tag{13}$$

$$\text{s.t.} \quad \sum_{r \in N_i} \xi_{ir} = 1, \quad i = 2, \dots, n \tag{14}$$

$$\xi_{ir} = \frac{1}{K_r} \sum_{j \in M_r} \xi_{jr} q_j, \quad i \in M_r, \ r = 1, \dots, \hat{r} \tag{15}$$

$$\sum_{r=1}^{\hat{r}} \sum_{i \in M_r} \frac{q_i}{K_r} \xi_{ir} = M \tag{16}$$

$$\xi_{ir} \in \{ 0, 1 \} \tag{17}$$

Constraints (15) ensure that $\xi_{ir} = 1$ if and only if $\xi_{jr} = 1$, $\forall j \in M_r$ and hence $y_r = 1$. Thus, constraints (14) correspond to constraints (9).

Let the above problem be relaxed by (i) removing constraints (15) and (ii) by replacing set M_r for route r by the complete set $I = \{2, \dots, n\}$. The resulting relaxed problem can be somewhat strengthened by adding the constraints as :

$$\sum_{r=1}^{\hat{r}} \sum_{i=2}^{n} q_i \xi_{ir} = \overline{Q}, \tag{18}$$

which was redundant for the formulation given by equations (13) - (17) but which is no longer redundant for the new relaxed problem.

In the relaxed problem defined by equations (13), (14), (16), (17) and (18) (with M_r replaced by I), only one route need by considered for each customer x_i and for which possible value of load q on the route ($q \in W$). This is clear from the fact that if two routes r_1 and r_2 both contain customer x_i and have loads $K_{r_1} = K_{r_2} = q$, then if $d_{r_1} \leq d_{r_2}$, route r_1 dominates route r_2 in the relaxed problem. Let us call the undominated route (i, l) with $l = \tau(q)$. We will denote the cost of this route by d_{il}. There are now w routes to consider for each i. The relaxed problem now becomes :

$$\text{Min} \sum_{i=2}^{n} \sum_{l=1}^{w} \overline{d}_{il} \, \xi_{il} \tag{19}$$

$$\text{s.t.} \quad \sum_{l=1}^{w} \xi_{il} = 1, \quad i = 2, \dots, n \tag{20}$$

$$\sum_{i=2}^{n} \sum_{l=1}^{w} \frac{q_i}{q(l)} \xi_{il} = M \tag{21}$$

$$\sum_{i=2}^{n} \sum_{l=1}^{w} q_i \xi_{il} = \bar{Q} \tag{22}$$

$$\xi_{il} \in \{ 0, 1 \} \tag{23}$$

where $\overline{d}_{il} = d_{il}\, q_i/q(l)$.

Note that if no route passing through i with load $q(l)$ exists, then $d_{il} = \infty$. $d_{il}/q(l)$ represents the marginal cost of supplying customer i, on a route with load $q(l)$, with a unit quantity and hence $\overline{d}_{il}$ is the cost contribution of customer i.

It is quite apparent that the cost $\psi(x_i)$ of the minimum cost through q-route passing through customer x_i and having load $q(l)$ is a lower bound on d_{il}. Thus, the solution of the problem defined by the objective function

$$\text{Min} \sum_{i=2}^{n} \sum_{l=1}^{w} b_{il}\, \xi_{il} \tag{24}$$

and constraints (20) - (23), where $b_{il} = \psi_l(x_i) \cdot q_i/q(l)$, is a lower bound to the VRP. b_{il} is a lower bound on $\overline{d}_{il}$ and is obtained by relaxing the restrictions that in a feasible solution the degree of every vertex is 2.

Therefore, a simple bound can be computed by ignoring constraints (21) and (22), and minimizing (24) subject to only (20) and (23). The resulting bound is as follows :

$$\text{LB2} = \sum_{i=2}^{n} \min_{l=1,\dots,w} [\, b_{il}] \tag{25}$$

We will show a detailed example of how to compute this indirect lower bound for the VRP with loops and without loops in Section 4.6.

4.5 Penalty procedures for improving the bounds

For the VRP, we can use the penalty procedures and the subgradient method (see Chapter 3). As we have seen in the previous section, the lower bounds for the VRP from state-space relaxation are computed by the expressions :

$$\mathrm{LB1(o)} = f(M, \overline{Q}) \text{ ; for the direct bound of the VRP,}$$

$$\mathrm{LB2(o)} = \sum_{i=2}^{n} \min_{l=1,\ldots,w} [\, b_{il} \,] \text{ ; for the indirect bound of the VRP.}$$

Let R(o) be the routes producing the above minima, R1(o) and R2(o) corresponding to LB1(o) and LB2(o) can be obtained from $f(q, x_i)$, $p(q, x_i)$, $\phi(q, x_i)$ and $\pi(q, x_i)$ by backtracking respectively. For example, Fig. 4.2 and Fig. 4.3 show the routes R1(o) and R2(o) respectively. Such routes can clearly be infeasible. As we can see in Fig. 4.2, vertices

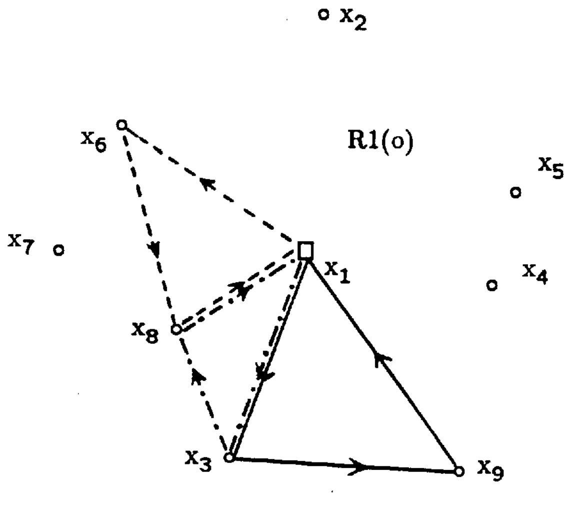


Figure 4.2 Routes R1(o) corresponding to bound LB1(o).

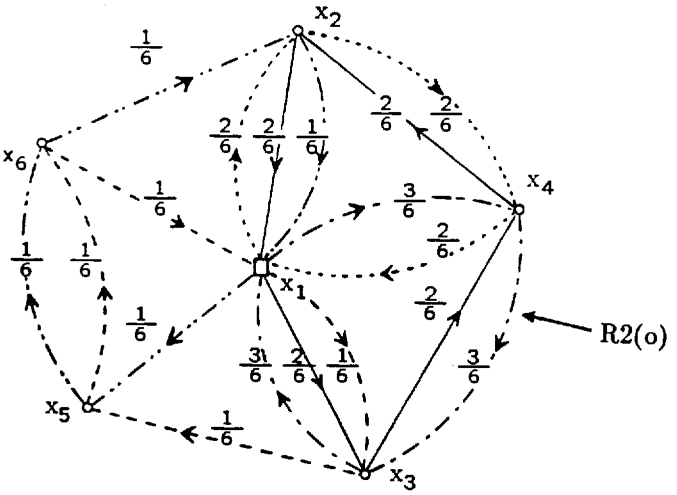


Figure 4.3 Routes R2(o) corresponding to bound LB2(o).

x_2, x_4, x_5 and x_7 are not visited, whereas vertices x_3 and x_8 are visited twice. Similarly in Fig. 4.3 $d_k = \sum_i \delta_k q_i/q(l_i) \neq 2$, for some k. Here, values on arcs mean weights $(q_i/q(l))$ on arcs ; for example, suppose that $\psi_6(x_2)$ for the vertex x_2 is composed of f(5, x_2) and f(2, x_2), and its through q-path is x_1-x_3-x_4-x_2-x_1. Hence, value $\frac{2}{6}$ on arcs (line in Fig. 4.2) is derived from $q_2=2/q(l)=6$. In Fig. 4.2 we can penalize vertices x_2, x_3, x_4, x_5, x_7 and x_8 (by penalties λ_2, λ_4, λ_5 and $\lambda_7 < 0$, and λ_3 and $\lambda_8 > 0$) in the normal Lagrangean fashion, and we can then modify c_{ij} as : $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$. And a new lower bound LB1(λ) can then be obtained by resolving the recursions (in the same way as in the previous section) with an updated cost matrix $[c'_{ij}]$. New routes R(λ) are then obtained with respect to the new LB1(λ). We wish to choose λ^* for which :

$$\text{LB1}(\lambda^*) = \max_{\lambda} [\ \text{LB1}(\lambda)\ ].$$

In this penalty procedure, we can use the subgradient optimization method to compute λ^* (see Held, Wolfe & Crowder [1974] or Sandi [1979]).

A similar procedure can be used to obtain the best bound LB2(λ^*).

4.5.1 An algorithm for improving the bounds for the VRP

We will describe an algorithm to improve the lower bounds for the VRP using penalty procedures. This algorithm can be used for the direct bound (LB1) and the indirect bound (LB2) for the VRP in the same way except for the step of backtracking.

Step 0 : (Initialization). Set the best lower bound $ZL^*= 0$. Let ZU^* be the value of the best solution so far. Set $\alpha = 2.0$ and KOUNT = 0.

Step 1 : (Initialization). Set $\lambda_i = 0, i = 1, \dots , n$ and $d_i = 0, i = 1, \dots , n$.

Step 2 : (Calculation of lower bound). Compute the lower bound (LB1 or LB2) using state-space relaxation as mentioned in the previous section. Let ZL be the updated lower bound on the value of the solution to the VRP, $ZL = LB - 2\sum\lambda_i$. If $ZL^* < ZL$, set $ZL^* = ZL$. If $ZL^* \geq ZU^*$ or KOUNT = maximum number of iterations allowed, stop. Else if $ZL^* < ZU^*$ and KOUNT $\neq$ maximum number of iterations allowed, KOUNT = KOUNT + 1, and go to step 3a (for the direct bound), or step 3b (for the indirect bound).

Step 3a: (Backtracking for the direct bound). Backtrack in order to find the q-routes, which are not necessarily pairwise vertex disjoint, corresponding to the above direct lower bound using $f(q, x_i)$, $p(q, x_i)$, $\phi(q, x_i)$ and $\pi(q, x_i)$, where $q = Q$. Check the degree d_i of vertex x_i with respect to graph G corresponding to the updated lower bound. If the degree d_i is 2, for all i ($i = 2, \dots , n$), stop. (In this case ZL^* is the best lower bound that can be obtained by this procedure and is the optimal solution value for the VRP). Otherwise, go to step 4.

Step 3b: (Backtracking for the indirect bound). Backtrack in order to find the q-routes, which are not necessarily pairwise vertex disjoint, from $\psi_{l_i}(x_i)$ corresponding to the above indirect lower bound using $f(q, x_i)$, $p(q, x_i)$, $\phi(q, x_i)$ and $\pi(q, x_i)$ for $q = Q$, where l_i is the value of l producing the minimum in the expression below :

$$\min_{l=1, \dots ,w} \left[\frac{\psi_l(x_i)}{q(l)} \right].$$

Let δ^i_k be the degree of x_k with respect to graph G_i (refer to Fig. 4.3). Then, compute

$$d_k = \sum_{i=2}^{n} \delta^i_k \cdot q_i/q(l_i). \tag{26}$$

The degree d_k should be equal to 2 in any feasible solution to the VRP. Check the degree d_k of vertex x_i with respect to graph G corresponding to the updated lower bound. If the degree d_k is 2, for all i (i = 2, ... , n), stop. (In this case ZL* is the best lower bound that can be obtained by this procedure and is the optimal solution value for the VRP). Otherwise, go to step 4.

Step 4 : (Penalties). Compute penalties as below :

$$\lambda_i = \lambda_i + \alpha \cdot \frac{ZU^* - ZL}{\sum_{j=2}^{n} (d_j - 2)^2} \cdot (d_i - 2) \cdot q_i/\max[q_i], \quad i = 2, \dots , n$$

where α is a constant $(0 < \alpha \leq 2)$ and can be periodically reduced by some factor. For example, after every 5 iterations α is reduced to a half, i.e. $\alpha = 2.0$ for KOUNT ≤ 5, and $\alpha = 1.0$ for $6 \leq$ KOUNT < 10, and so on.

Step 5 : (Udating the cost matrix). Modify and update the cost matrix $[c_{ij}]$ as :

$$c'_{ij} = c_{ij} + \lambda_i + \lambda_j.$$

Step 6 : (Computation of f, p, ϕ and π from the relaxed recursion). Compute f(q, x), p(q, x), ϕ(q, x) and π(q, x) for q = Q, from the Dynamic Programming recursions using the state-space relaxation for the updated cost matrix $[c'_{ij}]$.

Go to step 1 (for the direct bound). Go to step 7 (for the indirect bound).

Step 7 : (Computation of $\psi(x_i)$ and matrix $[b_{il}]$). With the updated value of f, p, ϕ and π, and using the expression (7), compute $\psi(x_i)$. And then, compute b_{il} as :

$$b_{il} = \psi_l(x_i) \cdot q_i/q(l), \qquad \text{for } i = 2, \dots , n,\ l = 1, \dots , Q \tag{27}$$

Go to step 1.

At the end we can obtain an updated lower bound which is close to the best lower bounds obtainable from these procedures as :

$$LB1(\lambda^*) = \max_{\lambda} [\, LB1(\lambda)\,] = f(M, \overline{Q}) - 2\sum\lambda_i,$$

$$LB2(\lambda^*) = \max_{\lambda} [\, LB2(\lambda)\,] = \min_{l=1, \ldots, w} [\, \frac{\psi_l(x_i)}{q(l)} \cdot q_i] - 2\sum\lambda_i.$$

4.6 An example

Consider a 1-depot, 8-customers, 3-vehicles VRP, where x_1 refers to the depot, and x_2, ..., x_9 refer to the customers. We will use the same example as in Chapter 3, Refer to Table 3.1 and Fig. 3.2 in Chapter 3 for the intercustomer (and depot) distances. The vehicles are of capacity Q = 6 units and the customer demands are as shown below :

x_i =	1	2	3	4	5	6	7	8	9
q_i =	0	2	3	1	1	2	1	3	2

The total demand is $\overline{Q} = 15$.

We will use state-space relaxation to compute lower bounds on the value of the VRP solution.

4.6.1 The simple bounds from state-space relaxation

(A) Direct bound 1 (LB1)

From expressions (2), (3) and (3′), and the mapping function $g(S) = \sum_{i \in S} q_i$ (where we will choose q_i, i = 2, ... , 9, so that $g(S) = \sum_{x \in S} q_i \equiv q$), the relaxed recursion for this example can be rewritten as :

$$f(m, q) = \min_{q-6(m-1) \leq q' \leq \min [q,6]} [f(m\text{-}1, q\text{-}q') + \overline{v}(q')], \tag{26a}$$

with $6m\text{-}3 \leq q \leq \min [15, 6m]$, and

$$\overline{v}(q') = \min_{x_i} [\theta(q', x_i) + c(x_i, x_1)]. \tag{26b}$$

From Tables 3.2a and 3.2b of Chapter 3, Tables 4.1a and 4.1b are obtained as :

Table 4.1a $\theta(q', x)$

x\q	1	2	3	4	5	6
x_2	∞	28	42	38	53	62
x_3	∞	∞	21	40	48	28
x_4	14	32	44	40	52	56
x_5	17	29	47	37	55	59
x_6	∞	18	38	48	34	43
x_7	22	48	34	27	46	49
x_8	∞	∞	15	34	37	34
x_9	∞	30	31	49	39	57

Table 4.1b $\gamma(q', x)$

x\q	1	2	3	4	5	6
x_2	-	x_1	x_5	x_6	x_8	$x_{5,7}$
x_3	-	-	x_1	x_4	$x_{6,9}$	x_8
x_4	x_1	x_5	x_5	x_8	x_5	x_9
x_5	x_1	x_4	$x_{4,6}$	x_8	x_4	x_8
x_6	-	x_1	x_7	x_2	x_8	x_7
x_7	x_1	x_4	x_6	x_8	x_8	x_8
x_8	-	-	x_1	x_7	x_6	x_3
x_9	-	x_1	x_4	x_4	x_3	x_4

From the above expressions, the direct bound 1 to this VRP is then given by $f(M, \overline{Q}) = f(3, 15)$. The computation steps to compute f(3, 15) are :

The initialization is given by $f(1, q) = \overline{v}(q)$, and hence we will start by first computing $\overline{v}(.)$ from equation (26b) using Tables 4.1a and 3.1 of Chapter 3. Here, $\pi(q)$ is the vertex just prior to deopt (x_1) on the route corresponding to $\overline{v}(q)$ with load position q.

$$
\begin{aligned}
\overline{v}(1) &= \min\ [\theta(1,x_4)+c(x_4,x_1),\ \theta(1,x_5)+c(x_5,x_1),\ \theta(1,x_7)+c(x_7,x_1)] \\
&= \min\ [(14+14),\ (17+17),\ (22+22)] \\
&= 28, \text{ with the minimum obtained for } \pi(1) = x_4. \\
\overline{v}(2) &= \min\ [\theta(2,x_2)+c(x_2,x_1),\ \theta(2,x_4)+c(x_4,x_1),\ \theta(2,x_5)+c(x_5,x_1),\ \theta(2,x_6)+c(x_6,x_1), \\
&\qquad \theta(2,x_7)+c(x_7,x_1),\ \theta(2,x_9)+c(x_9,x_1)] \\
&= \min\ [(28+28),\ (32+14),\ (29+17),\ (18+18),\ (48+22),\ (30+30)] \\
&= 36, \text{ for } \pi(2) = x_6. \\
\overline{v}(3) &= \min\ [\theta(3,x_2)+c(x_2,x_1),\ \theta(3,x_3)+c(x_3,x_1),\ \theta(3,x_4)+c(x_4,x_1),\ \theta(3,x_5)+c(x_5,x_1), \\
&\qquad \theta(3,x_6)+c(x_6,x_1),\ \theta(3,x_7)+c(x_7,x_1),\ \theta(3,x_8)+c(x_8,x_1),\ \theta(3,x_9)+c(x_9,\ x_1)] \\
&= \min\ [(42+28),\ (21+21),\ (44+14),\ (47+17),\ (38+18),\ (34+22),\ (15+15),\ (31+30)] \\
&= 30, \text{ for } \pi(3) = x_8. \\
\overline{v}(4) &= \min\ [\theta(4,x_2)+c(x_2,x_1),\ \theta(4,x_3)+c(x_3,x_1),\ \theta(4,x_4)+c(x_4,x_1),\ \theta(4,x_5)+c(x_5,x_1), \\
&\qquad \theta(4,x_6)+c(x_6,x_1),\ \theta(4,x_7)+c(x_7,x_1),\ \theta(4,x_8)+c(x_8,x_1),\ \theta(4,x_9)+c(x_9,\ x_1)] \\
&= \min\ [(38+28),\ (40+21),\ (40+14),\ (37+17),\ (48+18),\ (27+22),\ (34+15),\ (49+30)] \\
&= 49, \text{ for } \pi(4) = x_7 \text{ or } x_8. \\
\overline{v}(5) &= \min\ [\theta(5,x_2)+c(x_2,x_1),\ \theta(5,x_3)+c(x_3,x_1),\ \theta(5,x_4)+c(x_4,x_1),\ \theta(5,x_5)+c(x_5,x_1), \\
&\qquad \theta(5,x_6)+c(x_6,x_1),\ \theta(5,x_7)+c(x_7,x_1),\ \theta(5,x_8)+c(x_8,x_1),\ \theta(5,x_9)+c(x_9,\ x_1)] \\
&= \min\ [(53+28),\ (48+21),\ (52+14),\ (55+17),\ (34+18),\ (46+22),\ (37+15),\ (39+30)] \\
&= 52, \text{ for } \pi(5) = x_6 \text{ or } x_8. \\
\overline{v}(6) &= \min\ [\theta(6,x_2)+c(x_2,x_1),\ \theta(6,x_3)+c(x_3,x_1),\ \theta(6,x_4)+c(x_4,x_1),\ \theta(6,x_5)+c(x_5,x_1), \\
&\qquad \theta(6,x_6)+c(x_6,x_1),\ \theta(6,x_7)+c(x_7,x_1),\ \theta(6,x_8)+c(x_8,x_1),\ \theta(6,x_9)+c(x_9,\ x_1)] \\
&= \min\ [(62+28),\ (28+21),\ (56+14),\ (59+17),\ (43+18),\ (49+22),\ (34+15),\ (57+30)] \\
&= 49, \text{ for } \pi(6) = x_3 \text{ or } x_8.
\end{aligned}
$$

The functions $\overline{v}(.)$ and $\pi(.)$ are summarized in Table 4.2.

Table 4.2 $\overline{v}(q)$ and $\pi(q)$

q	1	2	3	4	5	6
$\overline{v}(q)$	28	36	30	49	52	49
$\pi(q)$	x_4	x_6	x_8	$x_{7,8}$	$x_{6,8}$	$x_{3,8}$

We will now use the recursion (26a) to compute f(2, q). In case of computing f(2, q), we need to compute only for 9 ≤ q ≤ 12. Thus,

$$\begin{aligned} f(2, 9) &= \min [\, f(1, \underbrace{6) + \overline{v}(3)}_{q'=\,3},\ f(1, \underbrace{5) + \overline{v}(4)}_{q'=\,4},\ f(1, \underbrace{4) + \overline{v}(5)}_{q'=\,5},\ f(1, \underbrace{3) + \overline{v}(6)}_{q'=\,6} \,] \\ &= \min [\, (49+30),\ (52+49),\ (49+52),\ (30+49) \,] \\ &= 79, \text{ with the minimum obtained for } \pi(9) \text{ composed of } \pi(3) \text{ and } \pi(6). \end{aligned}$$

$$\begin{aligned} f(2, 10) &= \min [\, f(1, \underbrace{6) + \overline{v}(4)}_{q'=\,4},\ f(1, \underbrace{5) + \overline{v}(5)}_{q'=\,5},\ f(1, \underbrace{4) + \overline{v}(6)}_{q'=\,6} \,] \\ &= \min [\, (49+49),\ (52+52),\ (49+49) \,] \\ &= 98, \text{ for } \pi(10) \text{ composed of } \pi(4) \text{ and } \pi(6). \end{aligned}$$

$$\begin{aligned} f(2, 11) &= \min [\, f(1, \underbrace{6) + \overline{v}(5)}_{q'=\,5},\ f(1, \underbrace{5) + \overline{v}(6)}_{q'=\,6} \,] \\ &= \min [\, (52+49),\ (49+52) \,] \\ &= 101, \text{ for } \pi(11) \text{ composed of } \pi(5) \text{ and } \pi(6). \end{aligned}$$

$$\begin{aligned} f(2, 12) &= \min [\, f(1, \underbrace{6) + \overline{v}(6)}_{q'=\,6} \,] \\ &= \min [\, (49+49) \,] \\ &= 98, \text{ for } \pi(12) \text{ composed of } \pi(6) \text{ and } \pi(6). \end{aligned}$$

Since iteration 3 to compute f(3, q) is the last iteration for the present example (since M = 3 and $\overline{Q}$ = 15), we do not need to compute f(3, q) for all q, but need only compute f(3, 15) since this is the only value required. Therefore,

$$\begin{aligned} f(3, 15) &= \min_{3 \leq q' \leq 6} [\, f(2, 15 - q') + \overline{v}(q')\,] \\ &= \min [\, \underbrace{f(2, 12)+\overline{v}(3)}_{q'=3},\ \underbrace{f(2, 11)+\overline{v}(4)}_{q'=4},\ \underbrace{f(2, 10)+\overline{v}(5)}_{q'=5},\ \underbrace{f(2, 9)+\overline{v}(6)}_{q'=6}\,] \\ &= \min [\, (98+30), (101+49), (98+52), (79+49)\,] \\ &= 128, \text{ with the minimum obtained for } \pi(15) \text{ composed of } \pi(6), \pi(6) \text{ and } \pi(3). \end{aligned}$$

Note that, in general, since $\overline{v}(q)$ represents the value of a route (not necessarily simple) with load q on it, the function f(M, $\overline{Q}$), in general, represents a combination of M such routes (with total load $\overline{Q}$, as required).

In the present example, the lower bound (LB1) of 128 is produced by two terms in square brackets above, indicating there are two alternative combinations of pairs of routes whose total values are 128.

The routes corresponding to f(3, 15) can be derived by backtracking (see below) through $\pi(q)$ of Table 4.2 and 4.1b giving the values of the predecessor indices $\gamma(q', x)$.

Route 1 : $x_1 - x_8 - x_3 - x_1$

$$\begin{tikzcd} \underline{x_1} & x_3 \arrow[dl] & \\ x_3 & x_8 \arrow[dl] & \gamma(6, x_3) \\ x_8 & \underline{x_1} & \gamma(3, x_8) \end{tikzcd}$$

Route 2 : This is the same as route 1.

Route 3 : $x_1 - x_8 - x_1$

$$\begin{tikzcd} \underline{x_1} & x_8 \arrow[dl] & \\ x_8 & \underline{x_1} & \gamma(3, x_8) \end{tikzcd}$$

For the present example, the corresponding triple of routes are as shown in Fig. 4.4. The route with load 6 is repeated twice 6.

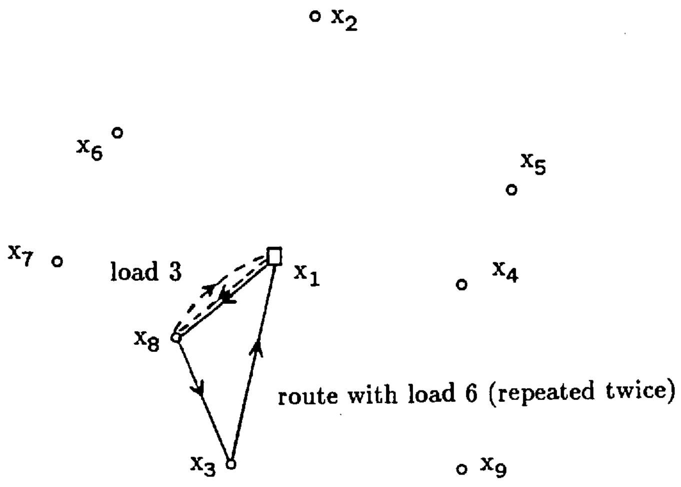


Figure 4.4 Three routes of value LB1=128 (= 49+49+30).

(B) Indirect bound (LB2)

In the former section we computed a direct lower bound (LB1) for the VRP solution value. In this example we compute the indirect bound (LB2) as given by expression (25). In Section 4.4, a procedure is given for computing the cost(distance) $f_l(x_i)$ of the q-path starting at x_1, finishing at x_i with total load $q(l)$ and with no loops.

These q-paths are then used to compute through q-routes of value $\psi_l(x_i)$. Clearly, imposing the restriction that the q-paths contain no loops improves the quality of the bound finally obtained, but at some additional computational cost.

In the example of the TSP, for the same data (but with the aim of computing a TSP bound on that occasion) we have already computed the cost of all q-paths starting from vertex x_1 (see Table 3.2a of Chapter 3), without the 'no-loops' restriction. Howerever, we need the values only for $q = 6$, since the capacity of vehicle, $Q = 6$ in this example (see Table 4.1a). Thus, we can compute the indirect bound LB2 for the VRP as given by the expression (25), by again ignoring the 'no-loops' improvement and making use of the table for f(q, x) already computed. Clearly, we will obtain a bound worse than what we could have obtained had we recomputed $f(q, x_i)$ with the 'no-loops' restriction, but in this way we will also be able to compare this indirect bound with the direct bound obtained for the same VRP in the previous part.

For the VRP in this example we have :

$W = \{ 1, 2, 3, 4, 5, 6 \}$, $w = 6$.

(i.e. every load value from 1 unit to 6 units can be achieved as the load on a route).

$q(l) = [1, 2, 3, 4, 5, 6]$.

(i.e. load level 3 implies a load of 3 units).

$\tau(q) = [1, 2, 3, 4, 5, 6]$.

(i.e. a load of 3 units on the route corresponds to load level 3, etc.)

Thus, the load level and the actual load (in units) correspond to the same number for this example. i.e. $l \equiv \tau(q) = q(l)$.

In order to compute the matrix $[b_{il}]$ with $\psi_l(x_i) \cdot q_i / q(l)$, we must first compute $\psi_l(x_i)$; the minimum cost of a through-route passing through x_i and with total load level l. The expression for $\psi_l(x_i)$ is given in Section 4.4 for the general case (no-loops). When the 'no-loops' restriction is ignored, only the first expression (7a) applies, and hence, for the present example, (where $l = \tau(q) = q$), this term can be rewritten as :

$$\psi_q(x_i) = \min_{q_i \le q' \le \frac{1}{2}(q+q_i)} [\, f(q', x_i) + f(q+q_i-q', x_i) \,]. \qquad (7'a)$$

Thus, from Table 4.1a and the expression (7'a), we compute $\psi_q(x_i)$ as :

<u>For x_2</u> : $\psi_2(x_2) = \min [\, \underbrace{f(2, x_2)}_{q'=\,2} + f(2, x_2) \,] = 28+28 = 56$

$$\psi_3(x_2) = \min [\, \underbrace{f(2, x_2)}_{q'=\,2} + f(3, x_2) \,] = 28+42 = 70$$

$$\psi_4(x_2) = \min\,[\, \underbrace{f(2, x_2) + f(4, x_2)}_{q'=2},\ \underbrace{f(3, x_2) + f(3, x_2)}_{q'=3} \,]$$

$$= \min\,[(28+38),\ (42+42)] = 66$$

$$\psi_5(x_2) = \min\,[\, \underbrace{f(2, x_2) + f(5, x_2)}_{q'=2},\ \underbrace{f(3, x_2) + f(4, x_2)}_{q'=3} \,]$$

$$= \min\,[(28+53),\ (42+38)] = 80$$

$$\psi_6(x_2) = \min\,[\underbrace{f(2,x_2)+f(6,x_2)}_{q'=2},\ \underbrace{f(3,x_2)+f(5,x_2)}_{q'=3},\ \underbrace{f(4,x_2)+f(4,x_2)}_{q'=4}]$$

$$= \min\,[(28+62),\ (42+53),\ (38+38)] = 76$$

<u>For x_3</u> :

$$\begin{aligned}
\psi_3(x_3) &= \min\,[\, f(3, x_3) + f(3, x_3) \,] = 21+21 = 42\\
\psi_4(x_3) &= \min\,[\, f(3, x_3) + f(4, x_3) \,] = 21+40 = 61\\
\psi_5(x_3) &= \min\,[\, f(3, x_3) + f(5, x_3),\ f(4, x_3) + f(4, x_3) \,]\\
&= \min\,[\, (21+48),\ (40+40) \,] = 69\\
\psi_6(x_3) &= \min\,[\, f(3, x_3) + f(6, x_3),\ f(4, x_3) + f(5, x_3) \,]\\
&= \min\,[\, (21+28),\ (40+48) \,] = 49
\end{aligned}$$

<u>For x_4</u> :

$$\begin{aligned}
\psi_1(x_4) &= \min\,[\, f(1, x_4) + f(1, x_4) \,] = 14+14 = 28\\
\psi_2(x_4) &= \min\,[\, f(1, x_4) + f(2, x_4) \,] = 14+32 = 46\\
\psi_3(x_4) &= \min\,[\, f(1, x_4) + f(3, x_4),\ f(2, x_4) + f(2, x_4) \,]\\
&= \min\,[\, (14+44),\ (32+32) \,] = 58\\
\psi_4(x_4) &= \min\,[\, f(1, x_4) + f(4, x_4),\ f(2, x_4) + f(3, x_4) \,]\\
&= \min\,[\, (14+40),\ (32+44) \,] = 54\\
\psi_5(x_4) &= \min\,[\, f(1, x_4)+f(5, x_4),\ f(2, x_4)+f(4, x_4),\ f(3, x_4)+f(3, x_4) \,]\\
&= \min\,[\, (14+52),\ (32+40),\ (44+44) \,] = 66\\
\psi_6(x_4) &= \min\,[\, f(1, x_4)+f(6, x_4),\ f(2, x_4)+f(5, x_4),\ f(3, x_4)+f(4, x_4) \,]
\end{aligned}$$

$$= \min [\,(14+56), (32+52), (44+40)\,] = 70,$$

etc. for x_5, ... , x_9.

The whole table is shown in Table 4.3a below.

Table 4.3a $\psi_q(x_i)$

l= q=	1	2	3	4	5	6
x_2	-	56	70	66	80	76
x_3	-	-	42	61	69	49
x_4	28	46	58	54	66	70
x_5	34	46	58	54	66	76
x_6	-	36	56	66	52	61
x_7	44	70	56	49	68	61
x_8	-	-	30	49	52	49
x_9	-	60	61	62	69	70

The matrix $[b_{iq}]$ is then computed as shown in Table 4.3b.

Table 4.3b $[\,b_{iq}]$

l= q=	1	2	3	4	5	6
x_2	-	56.00	46.67	33.00	32.00	$\underline{25.33}$
x_3	-	-	42.00	45.75	41.40	$\underline{24.50}$
x_4	28.00	23.00	19.33	13.50	13.20	$\underline{11.67}$
x_5	34.00	23.00	19.33	13.50	13.20	$\underline{12.67}$
x_6	-	36.00	37.33	33.00	20.80	$\underline{20.33}$
x_7	44.00	35.00	18.67	12.25	13.60	$\underline{10.17}$
x_8	-	-	30.00	36.75	31.20	$\underline{24.50}$
x_9	-	60.00	40.67	31.00	27.60	$\underline{23.33}$

The minimum value of b_{iq} for each row x_i is shown underlined, and hence the indirect bound LB2 is computed from equation (25) to be :

$$\begin{aligned} LB2 &= \sum_{i=1}^{9} \min_{q=1,\dots,6} [\, b_{iq} \,] \\ &= 25.33 + 24.5 + 11.67 + 12.67 + 20.33 + 10.17 + 24.5 + 23.33 \\ &= 152.5 \end{aligned}$$

(Note) In this example, the indirect bound LB2 is better than the direct bound derived in the previous section.

The graph of this solution from backtracking is as shown in Fig. 4.5, and the routes picked for each customer are as follows :

Route for $\psi_6(x_2)$: x_1- x_6- x_2- x_6- x_1.

Route for $\psi_6(x_3)$: x_1- x_3- x_7- x_6- x_1.

Route for $\psi_6(x_4)$: x_1- x_4- x_3- x_6- x_1.

Route for $\psi_6(x_5)$: x_1- x_4- x_5- x_4- x_3- x_1.

Route for $\psi_6(x_6)$: x_1- x_6- x_7- x_3- x_1.

Route for $\psi_6(x_7)$: x_1- x_6- x_7- x_3- x_1.

Route for $\psi_6(x_8)$: x_1- x_8- x_3- x_1.

Route for $\psi_6(x_9)$: x_1- x_4- x_9- x_3- x_1.

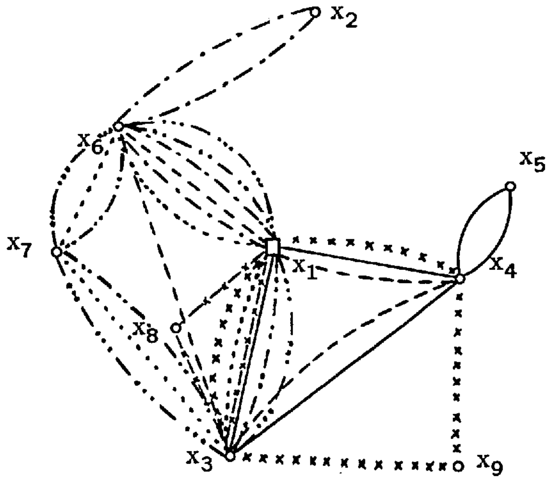


Figure 4.5 The graph of solution value LB2=152.5

4.6.2 The bounds with the " no - loops " restrictions

We will refer to the previous example. Since we are here considering the case with no loops, we need $\theta(q', x)$, $\gamma(q', x)$, $\phi(q', x)$ and $\pi(q', x)$ just for $q = 6$ from Tables 3.4a, b, c, d in Chapter 3 which has been already computed, and hence we will consider those as Tables 4.4a, b, c, d as follows :

Table 4.4a $\theta(q', x)$

x\q	1	2	3	4	5	6
x_2	∞	28	42	38	53	62
x_3	∞	∞	21	40	48	28
x_4	14	32	47	40	52	56
x_5	17	29	47	37	55	59
x_6	∞	18	38	48	34	43
x_7	22	48	34	27	51	49
x_8	∞	∞	15	34	37	34
x_9	∞	30	31	49	39	57

Table 4.4b $\gamma(q', x)$

x\q	1	2	3	4	5	6
x_2	-	x_1	x_5	x_6	x_8	$x_{5,7}$
x_3	-	-	x_1	x_4	$x_{6,9}$	x_8
x_4	x_1	x_5	x_9	x_8	x_5	x_9
x_5	x_1	x_4	x_6	x_8	x_4	x_8
x_6	-	x_1	x_7	x_2	x_8	x_7
x_7	x_1	x_4	x_6	x_8	x_8	x_8
x_8	-	-	x_1	x_7	x_6	x_3
x_9	-	x_1	x_4	x_4	x_3	x_3

Table 4.4c $\phi(q', x)$

x\q	1	2	3	4	5	6
x_2	∞	28	50	54	58	72
x_3	∞	∞	21	42	58	49
x_4	14	56	49	47	59	62
x_5	17	61	53	58	56	63
x_6	∞	18	45	58	51	53
x_7	22	56	62	41	60	50
x_8	∞	∞	15	39	51	46
x_9	∞	30	52	63	43	58

Table 4.4d $\pi(q', x)$

x\q	1	2	3	4	5	6
x_2	-	x_1	x_4	x_5	x_6	x_8
x_3	-	-	x_1	x_7	x_4	x_9
x_4	x_1	x_7	x_6	x_3	x_8	x_8
x_5	x_1	x_7	x_2	x_3	x_8	x_6
x_6	-	x_1	x_4	x_5	x_3	x_8
x_7	x_1	x_5	x_9	x_3	x_3	x_6
x_8	-	-	x_1	$x_{4,5}$	x_5	x_7
x_9	-	x_1	x_5	x_6	x_8	x_3

(A) Direct bound 1 (LB1)

The initialization is given by $f(1, q) = \overline{v}(q)$, and hence we will start by first computing $\overline{v}(.)$ from the expression (26b) using Table 4.5a.

$$\begin{aligned}
\overline{v}(1) &= \min\,[\theta(1,x_4)+c(x_4,x_1),\ \theta(1,x_5)+c(x_5,x_1),\ \theta(1,x_7)+c(x_7,x_1)]\\
&= \min\,[(14+14),\ (17+17),\ (22+22)]\\
&= 28,\ \text{ with the minimum obtained for } \pi(1) = x_4.\\
\overline{v}(2) &= \min\,[\theta(2,x_2)+c(x_2,x_1),\ \theta(2,x_4)+c(x_4,x_1),\ \theta(2,x_5)+c(x_5,x_1),\ \theta(2,x_6)+c(x_6,x_1),\\
&\qquad \theta(2,x_7)+c(x_7,x_1),\ \theta(2,x_9)+c(x_9,x_1)]\\
&= \min\,[(28+28),\ (32+14),\ (29+17),\ (18+18),\ (48+22),\ (30+30)]\\
&= 36,\ \text{ for } \pi(2) = x_6.\\
\overline{v}(3) &= \min\,[\theta(3,x_2)+c(x_2,x_1),\ \theta(3,x_3)+c(x_3,x_1),\ \theta(3,x_4)+c(x_4,x_1),\ \theta(3,x_5)+c(x_5,x_1),\\
&\qquad \theta(3,x_6)+c(x_6,x_1),\ \theta(3,x_7)+c(x_7,x_1),\ \theta(3,x_8)+c(x_8,x_1),\ \theta(3,x_9)+c(x_9,\ x_1)]\\
&= \min\,[(42+28),\ (21+21),\ (44+14),\ (47+17),\ (38+18),\ (34+22),\ (15+15),\ (31+30)]\\
&= 30,\ \text{ for } \pi(3) = x_8.\\
\overline{v}(4) &= \min\,[\theta(4,x_2)+c(x_2,x_1),\ \theta(4,x_3)+c(x_3,x_1),\ \theta(4,x_4)+c(x_4,x_1),\ \theta(4,x_5)+c(x_5,x_1),\\
&\qquad \theta(4,x_6)+c(x_6,x_1),\ \theta(4,x_7)+c(x_7,x_1),\ \theta(4,x_8)+c(x_8,x_1),\ \theta(4,x_9)+c(x_9,\ x_1)]\\
&= \min\,[(38+28),\ (40+21),\ (40+14),\ (37+17),\ (48+18),\ (27+22),\ (34+15),\ (49+30)]\\
&= 49,\ \text{ for } \pi(4) = x_7 \text{ or } x_8.\\
\overline{v}(5) &= \min\,[\theta(5,x_2)+c(x_2,x_1),\ \theta(5,x_3)+c(x_3,x_1),\ \theta(5,x_4)+c(x_4,x_1),\ \theta(5,x_5)+c(x_5,x_1),\\
&\qquad \theta(5,x_6)+c(x_6,x_1),\ \theta(5,x_7)+c(x_7,x_1),\ \theta(5,x_8)+c(x_8,x_1),\ \theta(5,x_9)+c(x_9,\ x_1)]\\
&= \min\,[(53+28),\ (48+21),\ (52+14),\ (55+17),\ (34+18),\ (46+22),\ (37+15),\ (39+30)]\\
&= 52,\ \text{ for } \pi(5) = x_6 \text{ or } x_8.\\
\overline{v}(6) &= \min\,[\theta(6,x_2)+c(x_2,x_1),\ \theta(6,x_3)+c(x_3,x_1),\ \theta(6,x_4)+c(x_4,x_1),\ \theta(6,x_5)+c(x_5,x_1),\\
&\qquad \theta(6,x_6)+c(x_6,x_1),\ \theta(6,x_7)+c(x_7,x_1),\ \theta(6,x_8)+c(x_8,x_1),\ \theta(6,x_9)+c(x_9,\ x_1)]\\
&= \min\,[(62+28),\ (28+21),\ (56+14),\ (59+17),\ (43+18),\ (49+22),\ (34+15),\ (57+30)]\\
&= 49,\ \text{ for } \pi(6) = x_3 \text{ or } x_8.
\end{aligned}$$

The functions $\overline{v}(.)$ and $\pi(.)$ are summarized in Table 4.5.

Table 4.5 $\overline{v}(q)$ and $\pi(q)$

q	1	2	3	4	5	6
$\overline{v}(q)$	28	36	30	49	52	49
$\pi(q)$	x_4	x_6	x_8	$x_{7,8}$	$x_{6,8}$	$x_{3,8}$

We will now use the recursion (26a) to compute f(2, q). In case of computing f(2, q), we need to compute only for $9 \leq q \leq 12$. Thus,

$$
\begin{aligned}
f(2, 9) &= \min [\, f(1, \underbrace{6) + \overline{v}}_{q'=3}(3),\ f(1, \underbrace{5) + \overline{v}}_{q'=4}(4),\ f(1, \underbrace{4) + \overline{v}}_{q'=5}(5),\ f(1, \underbrace{3) + \overline{v}}_{q'=6}(6)\,] \\
&= \min [\, (49+30), (52+49), (49+52), (30+49)\,] \\
&= 79, \text{ with the minimum obtained for } \pi(9) \text{ composed of } \pi(3) \text{ and } \pi(6). \\
f(2, 10) &= \min [\, f(1, \underbrace{6) + \overline{v}}_{q'=4}(4),\ f(1, \underbrace{5) + \overline{v}}_{q'=5}(5),\ f(1, \underbrace{4) + \overline{v}}_{q'=6}(6)\,] \\
&= \min [\, (49+49), (52+52), (49+49)\,] \\
&= 98, \text{ for } \pi(10) \text{ composed of } \pi(4) \text{ and } \pi(6). \\
f(2, 11) &= \min [\, f(1, \underbrace{6) + \overline{v}}_{q'=5}(5),\ f(1, \underbrace{5) + \overline{v}}_{q'=6}(6)\,] \\
&= \min [\, (52+49), (49+52)\,] \\
&= 101, \text{ for } \pi(11) \text{ composed of } \pi(5) \text{ and } \pi(6). \\
f(2, 12) &= \min [\, f(1, \underbrace{6) + \overline{v}}_{q'=6}(6)\,] \\
&= \min [\, (49+49)\,] \\
&= 98, \text{ for } \pi(12) \text{ composed of } \pi(6) \text{ and } \pi(6).
\end{aligned}
$$

Since iteration 3 to compute f(3, q) is the last iteration for the present example (since M = 3 and $\overline{Q}$ = 15), we do not need to compute f(3, q) for all q, but need only compute f(3, 15) since this is the only value required. Therefore,

$$\begin{aligned} f(3, 15) &= \min_{3 \le q' \le 6} [\, f(2, 15 - q') + \overline{v}(q') \,] \\ &= \min [\, \underbrace{f(2, 12)+\overline{v}(3)}_{q'=3}, \underbrace{f(2, 11)+\overline{v}(4)}_{q'=4}, \underbrace{f(2, 10)+\overline{v}(5)}_{q'=5}, \underbrace{f(2, 9)+\overline{v}(6)}_{q'=6} \,] \\ &= \min [\, (98+30), (101+49), (98+52), (79+49) \,] \\ &= 128, \text{ with the minimum obtained for } \pi(15) \text{ composed of } \pi(3), \pi(6) \text{ and } \pi(6). \end{aligned}$$

As we can see the results of computation of f(m, q) above, if the initial values have not changed, the direct bound is not changed. Therefore, we do not need to proceed further, and the direct lower bound (LB1) for this example is 128 which is the same as the previous one with loops. (Note that even without loops being explicitely excluded, no loops were formed in this particular numerical example.)

(B) Indirect bound (LB2)

The procedures of computation for this case are almost the same to the previous one ignoring the condition with no loops. Now we will compute the indirect bound (LB2) considering the 'no-loops' restriction. Then, we can compare this bound with the former one with loops.

First, we will compute $\psi_q(x_i)$ from the Tables 4.4a and 4.4c using expressions (7$'$a) and (7$'$b) which are rewritten expressions (7a) and (7b), where $l = \tau(q) = q$.

$$\psi_q(x_i) = \min_{q_i \le q' \le \frac{1}{2}(q+q_i)} [\, f(q', x_i) + f(q+q_i-q', x_i) \,], \quad \text{if } p(q', x_i) \ne p(q+q_i-q', x_i), \qquad (7'a)$$

or

$$\left.\begin{aligned}\psi_q(x_i) = \min_{q_i \leq q' \leq \frac{1}{2}(q+q')} & \llbracket \min [f(q', x_i)+\phi(q+q_i-q', x_i), \\ & \qquad \phi(q', x_i)+f(q+q_i-q', x_i)] \rrbracket, \\ & \text{if } p(q', x_i) = f(q+q_i-q', x_i).\end{aligned}\right\} \qquad (7'b)$$

<u>For x_2</u> :

$$\begin{aligned}
\psi_2(x_2) &= \min [\min [\underbrace{f(2, x_2) + \phi(2, x_2)}_{q'=2}, \underbrace{\phi(2, x_2) + f(2, x_2)}_{q'=2}] \\
&= \min [(28+\infty), (\infty+28)] = \infty \\
\psi_3(x_2) &= \min [\underbrace{f(2, x_2) + f(3, x_2)}_{q'=2}] = 28+42 = 70 \\
\psi_4(x_2) &= \min [\underbrace{f(2,x_2)+f(4,x_2)}_{q'=2}, \min [\underbrace{f(3,x_2)+\phi(3,x_2)}_{q'=3}, \underbrace{\phi(3,x_2)+f(3,x_2)}_{q'=3}]] \\
&= \min [(28+38), \min[(42+50), (50+42)]] = 66 \\
\psi_5(x_2) &= \min [\underbrace{f(2, x_2) + f(5, x_2)}_{q'=2}, \underbrace{f(3, x_2) + f(4, x_2)}_{q'=3}] \\
&= \min [(28+53), (42+38)] = 80 \\
\psi_6(x_2) &= \min [\underbrace{f(2,x_2) + f(6,x_2)}_{q'=2}, \underbrace{f(3,x_2) + f(5,x_2)}_{q'=3}, \\
&\qquad \min [\underbrace{f(4,x_2) + \phi(4,x_2)}_{q'=4}, \underbrace{\phi(4,x_2) + f(4,x_2)}_{q'=4}]] \\
&= \min [(28+62), (42+53), (38+54)] = 90
\end{aligned}$$

<u>For x_3</u> :

$$\begin{aligned}
\psi_3(x_3) &= \min [f(3, x_3) + \phi(3, x_3), \phi(3, x_3) + f(3, x_3)] = \infty \\
\psi_4(x_3) &= \min [f(3, x_3) + f(4, x_3)] = 21+40 = 61 \\
\psi_5(x_3) &= \min [f(3, x_3) + f(5, x_3), \\
&\qquad \min [f(4, x_3) + \phi(4, x_3), f(4, x_3) + f(4, x_3)]] \\
&= \min [(21+48), \min[(40+42), (42+40)]] = 69 \\
\psi_6(x_3) &= \min [f(3, x_3) + f(6, x_3), f(4, x_3) + f(5, x_3)]
\end{aligned}$$

$$= \min [(21+28), (40+48)] = 49$$

For x_4 :

$$\begin{aligned}
\psi_1(x_4) &= \min [\min[f(1, x_4) + \phi(1, x_4),\ \phi(1, x_4) + f(1, x_4)]] = \infty \\
\psi_2(x_4) &= \min [f(1, x_4) + f(2, x_4)] = 14+32 = 46 \\
\psi_3(x_4) &= \min [f(1, x_4) + f(3, x_4), \\
&\qquad \min [f(2, x_4) + \phi(2, x_4), \phi(2, x_4) + f(2, x_4)]] \\
&= \min [(14+47), (32+56)] = 61 \\
\psi_4(x_4) &= \min [f(1, x_4) + f(4, x_4), f(2, x_4) + f(3, x_4)] \\
&= \min [(14+40), (32+44)] = 54 \\
\psi_5(x_4) &= \min [f(1, x_4) + f(5, x_4), f(2, x_4) + f(4, x_4), \\
&\qquad \min [f(3, x_4) + \phi(3, x_4), \phi(3, x_4) + f(3, x_4)]] \\
&= \min [(14+52), (32+40), (47+49)] = 66 \\
\psi_6(x_4) &= \min [f(1, x_4) + f(6, x_4), \min [f(2, x_4) + \phi(5, x_4), \phi(2, x_4) \\
&\qquad + f(5, x_4)], f(3, x_4) + f(4, x_4)] \\
&= \min [(14+56), \min[(32+59), (56+52)], (47+40)] = 70,
\end{aligned}$$

etc. for $x_5, \ldots , x_9$.

The whole table is shown in Table 4.6a below.

Table 4.6a $\psi_q(x_i)$

l= q=	1	2	3	4	5	6
x_2	-	-	70	66	80	90
x_3	-	-	-	61	69	49
x_4	-	46	61	54	66	70
x_5	-	46	64	54	66	76
x_6	-	-	56	66	52	61
x_7	-	70	56	49	73	61
x_8	-	-	-	49	52	49
x_9	-	-	61	79	69	70

The table for the matrix $[b_{iq}]$ is then computed as shown in Table 4.6b.

Table 4.6b $[\, b_{iq}]$

l= q=	1	2	3	4	5	6
x_2	-	-	46.67	33.00	32.00	$\underline{30.00}$
x_3	-	-	-	45.75	41.40	$\underline{24.50}$
x_4	-	23.00	20.33	13.50	13.20	$\underline{11.67}$
x_5	-	23.00	21.33	13.50	13.20	$\underline{12.67}$
x_6	-	-	37.33	33.00	20.80	$\underline{20.33}$
x_7	-	35.00	18.67	12.25	14.60	$\underline{10.17}$
x_8	-	-	-	36.75	31.20	$\underline{24.50}$
x_9	-	-	40.67	39.50	27.60	$\underline{23.33}$

The minimum value of b_{iq} for each row x_i is shown underlined, and hence the indirect lower bound LB2 is computed from equation (25) to be :

$$\mathrm{LB2} = \sum_{i=1}^{9} \min_{q=1,\dots,6} [\, b_{iq} \,]$$

$$= 30.0 + 24.5 + 11.67 + 12.67 + 20.33 + 10.17 + 24.5 + 23.33$$

$$= 157.17$$

(Note) In this example, the indirect bound LB2 is improved compared with the former one obtained in the previous section.

The graph of this solution from backtracking is as shown in Fig. 4.6, and the routes picked for each customer are as follows :

Route for $\psi_6(x_2)$: x_1- x_2- x_7- x_8- x_1.	Route for $\psi_6(x_3)$: x_1- x_3- x_8- x_1.
Route for $\psi_6(x_4)$: x_1- x_4- x_9- x_3- x_1.	Route for $\psi_6(x_5)$: x_1- x_5- x_8- x_6- x_1.
Route for $\psi_6(x_6)$: x_1- x_6- x_7- x_8- x_1.	Route for $\psi_6(x_7)$: x_1- x_6- x_7- x_8- x_1.
Route for $\psi_6(x_8)$: x_1- x_8- x_3- x_1.	Route for $\psi_6(x_9)$: x_1- x_4- x_9- x_3- x_1.

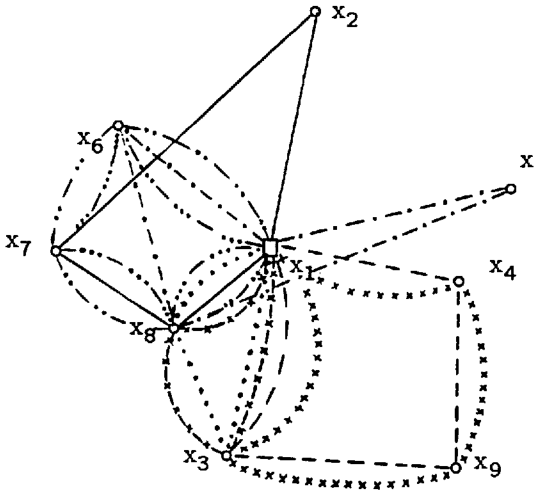


Figure 4.6 The graph of solution value LB2=157.2

4.6.3 The bounds from the algorithm with penalty procedures

(A) Direct bound (LB1) with no loops

The solution corresponding to the value of the direct bound (LB1) can be obtained by backtracking using the results of recursions in the previous sections as mentioned. This solution represents a graph such as the one in Fig. 4.2, which shows three q-routes. Therefore, placing penalties λ_i (i = 2, ... , n) on the vertices x_i (for $d_i \neq 2$) for the solution of the direct bound, we can obtain the new cost matrix $[c'_{ij}]$, and then obtain a updated lower bound by a allowed number of iterations repeatedly. The computational steps are as follows :

Step 0 : (Initialization). $ZL^* = 0$.

$ZU^* = 210$ (the best solution for the VRP to this example so far).

$\alpha = 2.0$ and KOUNT = 0.

Step 1 : (Initialization).

$\lambda_i = 0$ and $d_i = 0, i = 1, \ldots, 9$.

Step 2 : (Calculation of lower bound).

LB1= 128.0 (the direct lower bound derived from state-space relaxation).

$ZL = LB1 - 2\sum\lambda_i = 128.0 - 0 = 128.0$

Since $ZL^* < ZL$, $ZL^* = ZL = 128.0$,

KOUNT = 1, and go to step 3a.

Step 3a: (Backtracking).

The q-routes for the value corresponding to the lower bound, 128.0 are as :

Route 1 : $x_1 - x_3 - x_8 - x_1$

Route 2 : $x_1 - x_3 - x_8 - x_1$

Route 3 : $x_1 - x_8 - x_1$ (refer to Fig. 4.4 for this graph).

$d_i = (\,6, 0, 4, 0, 0, 0, 0, 6, 0\,)$ for $i = 1, \ldots, 9$.

Since $d_i \neq 2$, go to step 4.

Step 4 : (Penalties). Let's compute λ_i as follows :

$ZU^* - ZL = 210.0 - 128.0 = 82.0$

$$\sum_{j=2}^{9}(d_j - 2)^2 = (0 - 2)^2 + (4 - 2)^2 + (0 - 2)^2 + (0 - 2)^2 + (0 - 2)^2 + (0 - 2)^2 + (6 - 2)^2 + (0 - 2)^2 = 44$$

$$\lambda_2 = 0 + 2.0 \cdot \frac{82}{44} \cdot (0 - 2) \cdot \frac{2}{3} = -\frac{164}{33}$$

$$\lambda_3 = 0 + 2.0 \cdot \frac{82}{44} \cdot (4 - 2) \cdot \frac{3}{3} = \frac{82}{11}$$

$$\lambda_4 = 0 + 2.0 \cdot \frac{82}{44} \cdot (0 - 2) \cdot \frac{1}{3} = -\frac{82}{33}$$

$$\lambda_5 = 0 + 2.0 \cdot \frac{82}{44} \cdot (0 - 2) \cdot \frac{1}{3} = -\frac{82}{33}$$

$$\lambda_6 = 0 + 2.0 \cdot \frac{82}{44} \cdot (0 - 2) \cdot \frac{2}{3} = -\frac{164}{33}$$

$$\lambda_7 = 0 + 2.0 \cdot \frac{82}{44} \cdot (0 - 2) \cdot \frac{1}{3} = -\frac{82}{33}$$

$$\lambda_8 = 0 + 2.0 \cdot \frac{82}{44} \cdot (6 - 2) \cdot \frac{3}{3} = \frac{164}{11}$$

$$\lambda_9 = 0 + 2.0 \cdot \frac{82}{44} \cdot (0 - 2) \cdot \frac{2}{3} = - \frac{164}{33}$$

(Note that $\sum_{i \in R(o)} q_i = \sum_{j \notin R(o)} q_j$ and $\sum_{i \in R(o)} \lambda_i = \sum_{j \notin R(o)} \lambda_j$).

Step 5 : (Updating the cost matrix). Calculate a updated cost matrix $[c'_{ij}]$.

Since $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$, the results of modifying are as follows :

$$c'_{12} = c_{12} + \lambda_1 + \lambda_2 = 28.0 + 0 - \frac{164}{33} \doteq 23.03$$

$$c'_{13} = c_{13} + \lambda_1 + \lambda_3 = 21.0 + 0 + \frac{82}{11} \doteq 28.45$$

....

$$c'_{98} = c_{98} + \lambda_9 + \lambda_8 = 28.0 - \frac{164}{33} + \frac{82}{11} \doteq 37.94$$

The updated cost matrix $[c'_{ij}]$ is then as follows :

Table 4.7 New cost(distance) matrix $[c'_{ij}]$

$x_i \backslash x_j$	1	2	3	4	5	6	7	8	9
1	-	23.03	28.45	11.52	14.52	13.03	19.52	29.91	25.03
2	23.03	-	49.48	28.55	17.55	10.06	27.55	47.94	40.06
3	28.45	49.48	-	30.97	41.97	32.48	24.97	35.36	20.48
4	11.52	28.55	30.97	-	10.03	23.55	29.03	37.42	9.55
5	14.52	17.55	41.97	10.03	-	21.55	34.03	34.42	27.55
6	13.03	10.06	32.48	23.55	21.55	-	8.55	28.94	35.06
7	19.52	27.55	24.97	29.03	34.03	8.55	-	24.42	24.55
8	29.91	47.94	35.36	37.42	34.42	28.94	24.42	-	37.94
9	25.03	40.06	20.48	9.55	27.55	35.06	24.55	37.94	-

Step 6 : (Computation of θ, γ, ϕ and π from the relaxed recursion).

The results of computation for $\theta(q', x)$, $\gamma(q', x)$, $\phi(q', x)$ and $\pi(q', x)$ with the new cost matrix are shown in Tables 4.8a,b,c and d. Then, goto step 1.

Table 4.8a $\theta(q', x)$

x\q	1	2	3	4	5	6
x_2	∞	23.0	32.1	23.1	38.1	53.2
x_3	∞	∞	28.5	42.5	45.5	41.6
x_4	11.5	24.6	34.6	44.6	51.6	50.7
x_5	14.5	21.6	34.6	44.6	40.6	55.7
x_6	∞	13.0	28.1	33.1	42.1	49.2
x_7	19.5	40.6	21.6	43.6	41.6	50.7
x_8	∞	∞	29.9	43.9	42.0	46.0
x_9	∞	25.0	21.1	34.1	46.1	54.2

Table 4.8b $\gamma(q', x)$

x\q	1	2	3	4	5	6
x_2	-	x_1	x_5	x_6	x_6	x_6
x_3	-	-	x_1	x_4	$x_{6,9}$	x_9
x_4	x_1	x_5	x_9	x_5	x_2	x_5
x_5	x_1	x_4	x_6	x_4	x_2	x_2
x_6	-	x_1	x_7	x_2	x_2	x_2
x_7	x_1	x_4	x_6	x_6	x_6	x_6
x_8	-	-	x_1	x_7	x_6	x_7
x_9	-	x_1	x_4	x_4	$x_{4,7}$	x_4

Table 4.8c $\phi(q', x)$

x\q	1	2	3	4	5	6
x_2	∞	∞	40.1	39.1	49.1	62.2
x_3	∞	∞	∞	44.5	55.5	46.6
x_4	∞	48.6	36.6	50.6	56.6	58.5
x_5	∞	53.6	40.6	48.6	54.6	61.7
x_6	∞	∞	35.1	43.1	56.1	54.2
x_7	∞	48.6	49.6	45.6	50.6	65.7
x_8	∞	∞	∞	48.9	56.0	57.0
x_9	∞	∞	42.1	48.1	48.9	63.2

Table 4.8d $\pi(q', x)$

x\q	1	2	3	4	5	6
x_2	-	x_1	x_4	x_5	x_7	x_5
x_3	-	-	x_1	x_7	x_4	x_7
x_4	x_1	x_7	x_6	x_7	x_6	x_9
x_5	x_1	x_7	x_2	x_9	x_6	x_4
x_6	-	x_1	x_4	x_5	x_9	x_7
x_7	x_1	x_5	x_9	x_9	x_2	x_2
x_8	-	-	x_1	x_4	x_5	x_6
x_9	-	x_1	x_5	x_6	x_3	x_3

In this stage (KOUNT= 1), we obtain the updated lower bound, i.e. LB1 = 163.45

Therefore,

$$ZL = LB1 - 2\sum\lambda_i$$

$$= 163.45$$

In step 2, since $ZL^* < ZL$, $ZL^* = 163.45$

At the end of the 30th iteration we obtain $ZL^* = 198.73$ Because d_i for this value are not 2, the solution to this example corresponding to this lower bound is infeasible. Therefore, we can consider this lower bound as the best lower bound so far. We show the graph of the bound ascent for this example in Fig. 4.7.

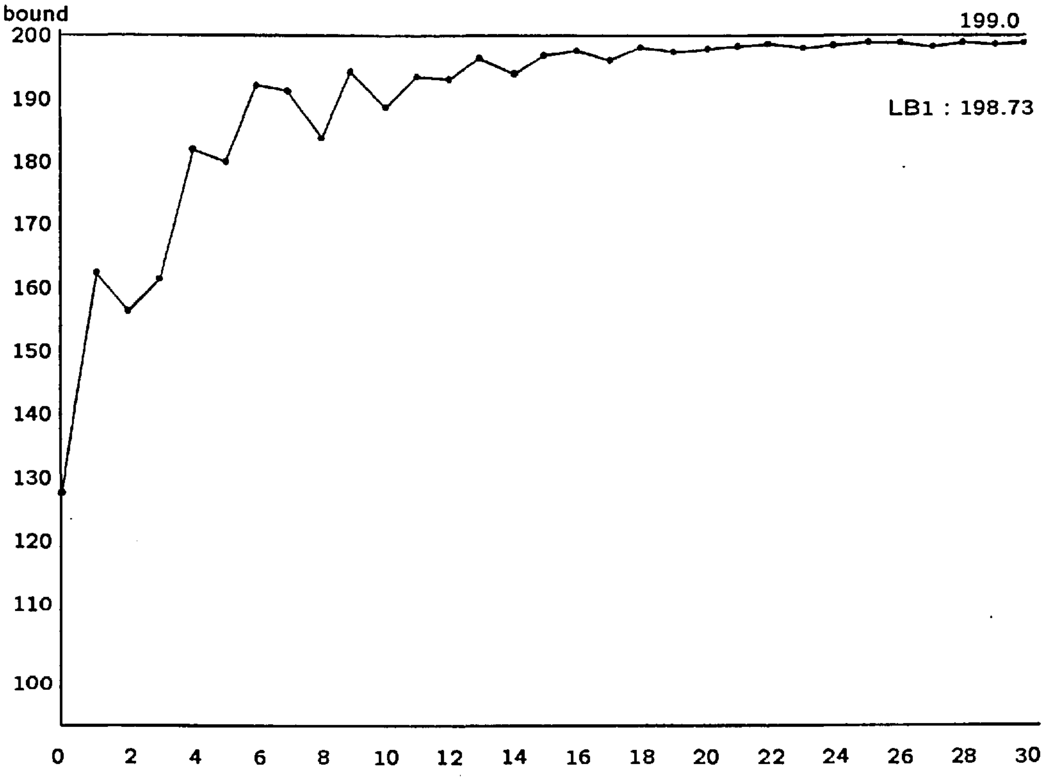


Figure 4.7 The direct bound ascents for the example (9 customers & 3 vehicles VRP).

(B) Indirect bound (LB2) with no loops

The solution corresponding to the value of the indirect bound (LB2) can be obtained by backtracking using the results of recursions in the previous sections as mentioned. This solution represents a graph such as the one in Fig. 4.3, which shows eight q-routes (one per customer). Therefore, placing penalties λ_i (i = 2, ... , n) on the vertices x_k (for $d_k \neq 2$) for the solution of the indirect bound, we can obtain the new cost matrix $[c'_{ij}]$, and then obtain a updated lower bound by a allowed number of iterations repeatedly. The computational steps are as follows :

Step 0 : (Initialization).

$ZL^* = 0$.

$ZU^* = 210.0$ (the best solution for the VRP to this example so far).

$\alpha = 2.0$ and KOUNT = 0.

Step 1 : (Initialization).

$\lambda_i = 0$ and $d_i = 0$, i = 1, ... , 9.

Step 2 : (Calculation of lower bound).

LB3 = 157.17 (the indirect lower bound derived from state-space relaxation).

$ZL = LB3 - 2\sum\lambda_i = 157.17 - 0 = 157.17$

Since $ZL^* < ZL$, $ZL^* = ZL = 157.17$,

KOUNT = 1, and go to step 3b.

Step 3a: (Backtracking).

The q-routes G_i corresponding to $\psi_{q_i}(x_i)$ is obtained by backtracking $\psi_6(x_2)$, $\psi_6(x_3)$, $\psi_6(x_4)$, $\psi_6(x_5)$, $\psi_6(x_6)$, $\psi_6(x_7)$, $\psi_6(x_8)$ and $\psi_6(x_9)$ respectively, and the results as follows :

Route for $\psi_6(x_2)$: $x_1 - x_2 - x_5 - x_8 - x_1$

Route for $\psi_6(x_3)$: $x_1 - x_3 - x_8 - x_1$

Route for $\psi_6(x_4)$: x_1 - x_4 - x_9 - x_3 - x_1

Route for $\psi_6(x_5)$: x_1 - x_5 - x_8 - x_6 - x_1

Route for $\psi_6(x_6)$: x_1 - x_6 - x_7 - x_8 - x_1

Route for $\psi_6(x_7)$: x_1 - x_6 - x_7 - x_8 - x_1

Route for $\psi_6(x_8)$: x_1 - x_8 - x_3 - x_1

Route for $\psi_6(x_9)$: x_1 - x_4 - x_9 - x_3 - x_1

Compute d_k for each vertex as follows :

$\sum\delta_2^i = 2$, $\sum\delta_3^i = 8$, $\sum\delta_4^i = 4$, $\sum\delta_5^i = 4$, $\sum\delta_6^i = 6$, $\sum\delta_7^i = 4$, $\sum\delta_8^i = 12$, and $\sum\delta_9^i = 4$. Then we can compute d_k from the expression (26).

$$\begin{aligned}
d_2 &= \sum_{i=2}^{9} \delta_2^i \cdot q_i/q(l_i) \\
&= \delta_2^2 \cdot q_2/q(l_2) + \delta_2^3 \cdot q_3/q(l_3) + \dots + \delta_2^9 \cdot q_9/q(l_9) \\
&= 2 * 2/6 + 0 * 3/6 + \dots + 0 * 2/6 \\
&= 4/6 = 0.67 \\
d_3 &= \sum_{i=2}^{9} \delta_3^i \cdot q_i/q(l_i) \\
&= 2 * 3/6 + 2 * 1/6 + 2 * 3/6 + 2 * 2/6 \\
&= 18/6 = 3. \\
d_4 &= 2 * 1/6 + 2 * 2/6 = 6/6 = 1. \\
d_5 &= 2 * 2/6 + 2 * 1/6 = 6/6 = 1. \\
d_6 &= 2 * 1/6 + 2 * 2/6 + 2 * 1/6 = 8/6 = 1.33 \\
d_7 &= 2 * 2/6 + 2 * 1/6 = 6/6 = 1. \\
d_8 &= 2 * 2/6 + 2 * 3/6 + 2 * 1/6 + 2 * 2/6 + 2 * 1/6 + 2 * 3/6 \\
&= 24/6 = 4. \\
d_9 &= 2 * 1/6 + 2 * 2/6 = 6/6 = 1.
\end{aligned}$$

And hence ;

$[d_i] = (\ 0,\ 0.67,\ 3,\ 1,\ 1,\ 1.33,\ 1,\ 4,\ 1\ )$.

Since $d_i \neq 2$ for several i, go to step 4.

Step 4 : (Penalties).

Let's compute λ_i as follows :

$ZU^* - ZL = 210.0 - 157.17 = 52.83$

$$\sum_{j=2}^{9}(d_j - 2)^2 = (0.67 - 2)^2 + (3 - 2)^2 + (1 - 2)^2 + (1 - 2)^2 + (1.33 - 2)^2$$
$$+ (1 - 2)^2 + (4 - 2)^2 + (1 - 2)^2 \doteq 11.22$$

$$\lambda_2 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (0.67 - 2) \cdot \frac{2}{3} = - 8.37$$
$$\lambda_3 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (3. - 2) \cdot \frac{3}{3} = 9.42$$
$$\lambda_4 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (1. - 2) \cdot \frac{1}{3} = - 3.14$$
$$\lambda_5 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (1. - 2) \cdot \frac{1}{3} = - 3.14$$
$$\lambda_6 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (1.33 - 2) \cdot \frac{2}{3} = - 4.18$$
$$\lambda_7 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (1. - 2) \cdot \frac{1}{3} = - 3.14$$
$$\lambda_8 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (4. - 2) \cdot \frac{3}{3} = 18.$$
$$\lambda_9 = 0 + 2.0 \cdot \frac{52.83}{11.22} \cdot (1. - 2) \cdot \frac{2}{3} = - 6.28$$

Step 5 : (Udating the cost matrix). Calculate a updated cost matrix $[c'_{ij}]$.

Since $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$, the results of modifying are as follows :

$c'_{12} = c_{12} + \lambda_1 + \lambda_2 = 28.0 + 0 - 8.37 \doteq 19.63$

$c'_{13} = c_{13} + \lambda_1 + \lambda_3 = 21.0 + 0 + 9.42 \doteq 30.42$

....

$c'_{98} = c_{98} + \lambda_9 + \lambda_8 = 28.0 - 6.28 + 18. \doteq 40.55$

We can obtain the new cost matrix as following the above ways.

Step 6 : (Computation of f, p, ϕ and π from the relaxed recursion).

The results of computation for f(q', x), p(q', x), ϕ(q', x) and π(q', x) with the new cost matrix above are shown in below Tables. Then, go to step 7.

Table 4.9a f(q′, x)

x\q	1	2	3	4	5	6
x_2	∞	19.6	27.4	21.3	35.0	48.7
x_3	∞	∞	30.4	43.1	44.9	39.6
x_4	10.9	22.6	31.3	41.8	45.8	43.5
x_5	13.9	19.6	33.1	40.0	34.8	48.5
x_6	∞	13.8	27.5	27.1	34.8	40.5
x_7	18.9	38.6	22.5	41.0	35.8	43.5
x_8	∞	∞	33.8	46.6	47.5	50.2
x_9	∞	23.7	18.5	30.2	45.1	49.4

Table 4.9b p(q′, x)

x\q	1	2	3	4	5	6
x_2	-	x_1	x_5	x_6	x_6	x_6
x_3	-	-	x_1	x_4	x_9	x_9
x_4	x_1	x_5	x_9	x_5	x_2	x_5
x_5	x_1	x_4	x_2	x_4	x_2	x_2
x_6	-	x_1	x_7	x_2	x_2	x_2
x_7	x_1	x_4	x_6	x_9	x_6	x_6
x_8	-	-	x_1	x_7	x_6	x_7
x_9	-	x_1	x_4	x_4	$x_{4,7}$	x_4

Table 4.9c ϕ(q′, x)

x\q	1	2	3	4	5	6
x_2	∞	∞	35.4	33.1	46.0	53.5
x_3	∞	∞	∞	45.1	49.1	48.8
x_4	∞	46.6	37.5	47.0	50.8	58.5
x_5	∞	51.6	35.5	44.0	48.8	54.5
x_6	∞	∞	34.5	41.3	51.8	49.7
x_7	∞	46.6	43.1	43.2	44.8	58.5
x_8	∞	∞	∞	51.6	57.3	59.0
x_9	∞	∞	39.5	45.2	51.6	56.6

Table 4.9d π(q′, x)

x\q	1	2	3	4	5	6
x_2	-	x_1	x_4	x_5	x_7	x_5
x_3	-	-	x_1	x_7	x_6	x_7
x_4	x_1	x_7	x_6	x_9	x_6	x_6
x_5	x_1	x_7	x_6	x_9	x_6	x_4
x_6	-	x_1	x_4	x_5	x_7	x_7
x_7	x_1	x_5	x_2	x_6	x_2	x_2
x_8	-	-	x_1	x_4	x_5	x_9
x_9	-	x_1	x_5	x_5	x_3	x_2

Step 7 : (Computation of $\psi_q(x_i)$ and matrix $[b_{iq}]$). With the updated values of f, p, ϕ and π, and using expressions (7a) and (7b), compute $\psi_q(x_i)$, the results are as in Table 4.10a.

Table 4.10a $\psi_q(x_i)$

l= q=	1	2	3	4	5	6
x_2	-	-	46.98	40.89	48.61	54.00
x_3	-	-	-	73.55	75.28	70.00
x_4	-	33.45	42.17	52.71	56.61	54.34
x_5	-	33.45	46.98	52.71	48.61	54.34
x_6	-	-	41.35	40.89	48.61	54.34
x_7	-	57.45	41.35	58.89	54.61	62.34
x_8	-	-	-	80.39	81.29	84.02
x_9	-	-	42.17	53.89	63.00	70.00

And then, using expression (27), compute b_{iq} as :

$b_{23} = \psi_3(x_2)\cdot q_2/q(3)$

$= 46.98 * 2/3 = 31.32$, and so on.

The results for the matrix $[b_{iq}]$ is then as shown in Table 4.10b.

Then, go to step 1.

Table 4.10b $[\, b_{iq}]$

l= q=	1	2	3	4	5	6
x_2	-	-	31.32	20.45	19.45	18.00
x_3	-	-	-	55.17	45.17	35.00
x_4	-	16.72	14.06	13.18	11.32	9.06
x_5	-	16.72	15.66	13.18	9.72	9.06
x_6	-	-	27.57	20.45	19.45	18.11
x_7	-	28.72	13.78	14.97	10.92	10.39
x_8	-	-	-	60.29	48.78	42.01
x_9	-	-	28.11	26.95	25.20	23.33

At the first iteration (KOUNT= 1), we obtain a updated lower bound with the tables above as follows :

$$LB2 = \sum \min [\, b_{iq} \,] - 2 \sum \lambda_i, \text{ for } i=2, \dots ,9 \text{ and } q=1, \dots ,6.$$

$$= 164.96$$

Therefore, ZL = 164.96 Since ZL* < ZL, ZL*= 164.96

At the end of 30 iterations we obtain ZL*= 195.3 Because d_i for this bound are not 2, the solution to this example corresponding to this lower bound is infeasible. Therefore, ZL* is the best lower bound. We show the graph of the bound ascent for this example in Fig. 4.8.

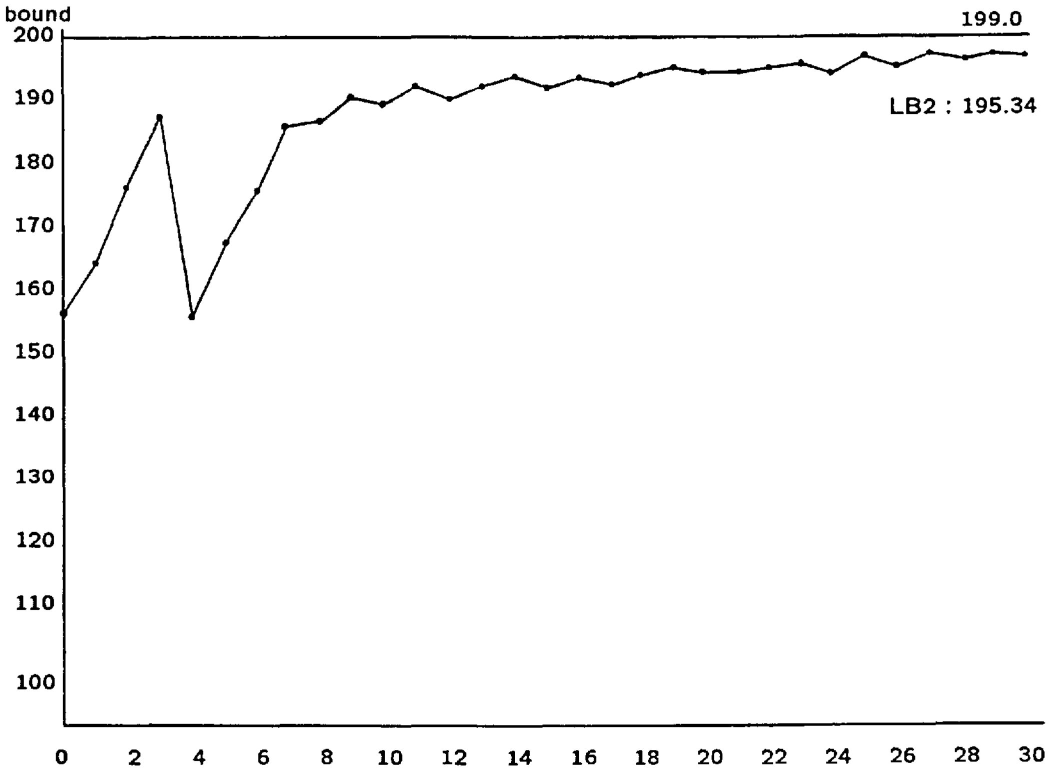


Figure 4.8 The indirect bound ascents for the example (9 customers & 3 vehicles VRP).

4.7 Computational Results

In this section, we will present the computational performance of the algorithm illustrated by the previous example. Ten test problems are used for the tests ranging from n = 9 to n = 50 customers (see Table 4.11 and Appendix A). All of these problems are randomly generated.

The values of the lower bound LB1 and LB2 as a function of the number of penalty iterations in the bound ascent procedure are given in Fig. 4.9 and Fig. 4.10 for the 25 and 50 customer problems respectively. From the figures and Table 4.11, we can easily see that the indirect lower bounds are better than the direct lower bounds, and that the lower bounds are much improved from the initial lower bounds during the ascent iterations.

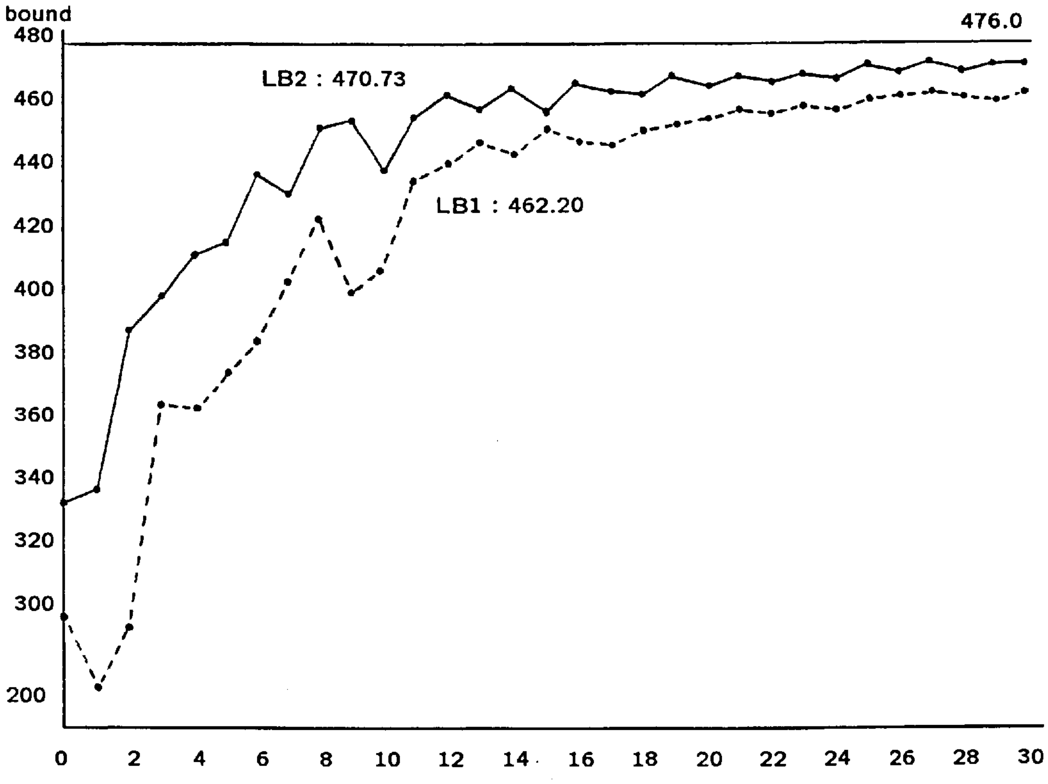


Figure 4.9 The bound ascents for 25 customers & 4 vehicles VRP.

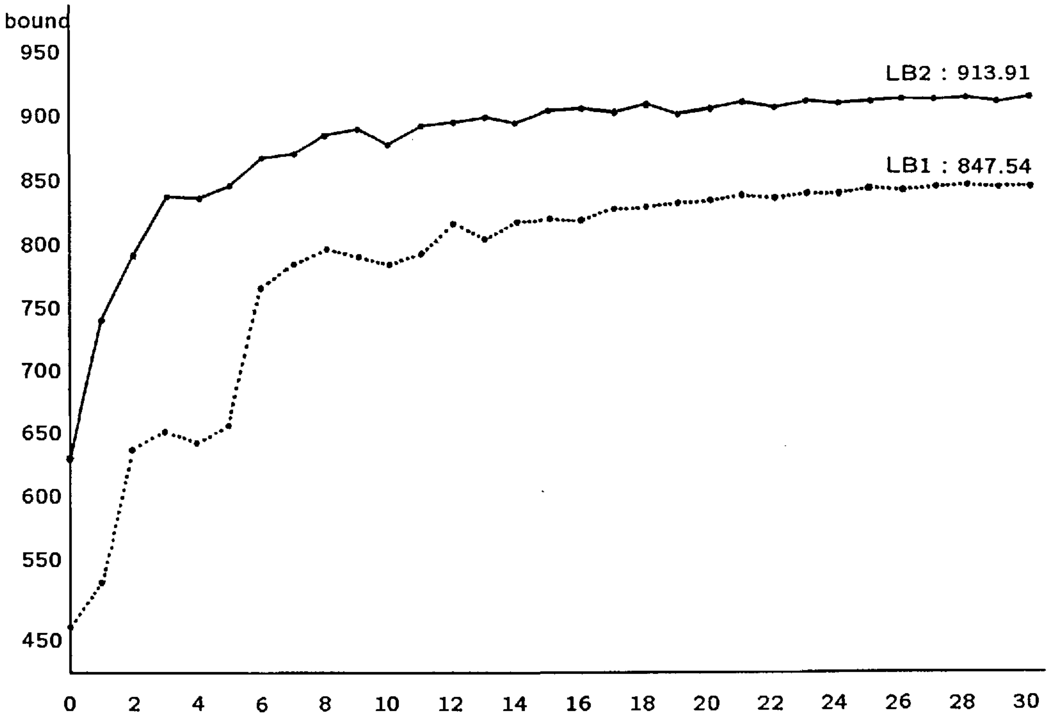


Figure 4.10 The bound ascents for 50 customers & 5 vehicles VRP.

In Table 4.11, we show the size of the test problems and the computational results for two kinds of bounds (LB1 and LB2). All values for the bounds are obtained at the 30th iteration. In case of small size problems, sometimes, the optimal solution value can be obtained without using the tree search to get the optimal solution. In this chapter, we only show the derivation of the lower bounds to embed into a tree search procedure to get the optimal solution for the VRP. In order to obtain the optimal solutions of these test problems, we will describe the tree search algorithm for solving the VRP in next chapter.

Table 4.11 Test problems & computational results for bounds : values & times (on IBM PS/2-70 386)

Problem	Number of vertices	Total demand	Number of vehicles	Vehicle capacity	Initial lower bounds			
					LB1		LB2	
					values	time	values	time
1	9	15	3	6	198.7	0 : 2	195.3	0 : 3
2	10	28	3	10	294.6	0 : 4	295.0*	0 : 3
3	15	56	3	20	350.4	0 : 22	357.4	0 : 28
4	20	80	3	30	387.2	1 : 08	393.3	1 : 28
5	25	97	4	30	462.2	1 : 50	470.7	2 : 13
6	30	105	4	30	610.8	2 : 42	617.4	3 : 14
7	35	108	4	30	637.2	3 : 50	651.5	4 : 28
8	40	140	5	30	713.1	5 : 02	767.9	5 : 43
9	45	141	5	30	782.8	6 : 23	849.2	7 : 13
10	50	147	5	30	847.5	8 : 00	913.9	8 : 58

* : The optimal solution value was obtained without embedding into the tree search algorithm.
Time (00:00) means (minutes : seconds).

CHAPTER 5

A TREE SEARCH ALGORITHM FOR THE VRP

5.1 Introduction

The tree search method or branch and bound method (Balas & Toth [1985]) is based on the idea of intelligently enumerating all the feasible solutions of a combina-torial optimization problem. The qualification "intelligently" is important here because, while solving combinatorial problems, it is a hopeless task to look at all feasible solutions.

The efficiency of all the tree search algorithms depends on two factors which are the quality of bounds and the branching strategies. In any tree search algorithm, the calculation of a bound on the value of the optimal solution to a subproblem corresponding to some nodes of the search tree is the most important factor affecting the efficiency of the algorithm.

In this chapter, we will describe a basic branching strategy and tree search algorithms for the VRP using the lower bounds derived from Chapter 4, and give some computational results for VRP's of small to medium size.

5.2 A basic tree search algorithm and branching strategy for the VRP

5.2.1 A basic depth-first tree search agorithm for the VRP

The basis of any tree search algorithm is to divide the set of all possible tours into smaller and smaller subsets and to calculate for each subset a lower bound on the cost of the best tour therein. The object of calculating lower bounds is that firstly to be used as guidance for the partitioning of the subsets and secondly to limit the search and also to identify the optimal tour. In constructing such search trees it is necessary to consider a branching strategy. Fig. 5.1 shows a flow diagram of the basic depth-first tree search for the VRP using some branching strategy (see Section 5.2.2).

The functions of the various boxes in Fig. 5.1 are explained further below.

(1) Initialization : Read in the cost matrix C, number the node N from which branching will continue, number the level of tree L, set Zopt (the cost of the best route so far) = ZUB (a heuristic upper bound), and set Z(N) = LBN which is the lower bound at root node N (= 0).

(2) Branching test : Check whether the set of customers x_i (i=2, ..., n) to choose for branching to next, is empty or not. If the unrouted customer set is empty, go to step 10 ; otherwise continue.

(3) Branching : Choose a customer (x_i) for branching to next according to the branching rule, and number the node N and the tree-depth level L.

(4) Branch forward to node N and modify the cost matrix C accordingly.

(5) Bound : Compute the lower bound of the node Z(N)= LBN.

(6), (7), (8) and (9) If this solution corresponding to the bound is feasible and the cost is less than Zopt, record it and go to step 10 to backtrack. If the cost is greater than or equal to Zopt, go to step 10 to backtrack. If the cost is less than Zopt, go to step 2 in order to branch forward.

(10) Backtrack (10a, 10b and 10c) : If the alternative to the current node N has not been examined, then go to examine that node. Set N= N+1, update the cost matrix accordingly and go to step 5. If the alternative to the current node has been examined,

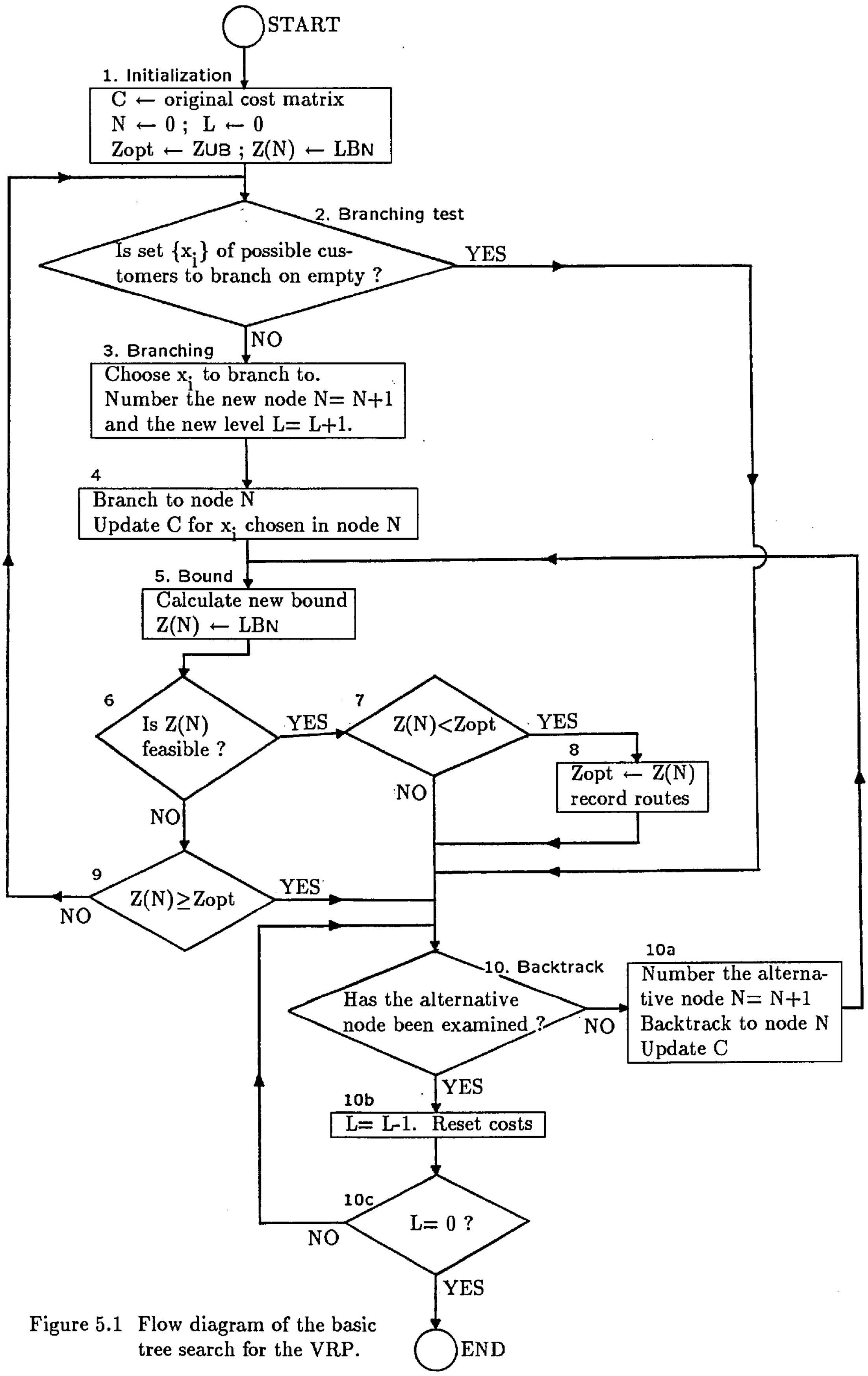


Figure 5.1 Flow diagram of the basic tree search for the VRP.

then set L= L-1. If L= 0, stop ; Else reset the costs accordingly (to their original values) and repeat step 10.

5.2.2 The branching strategy

The branching strategy (i.e. deciding which customer to examine next) is based on arcs, i.e. an arc (x_i, x_j) is chosen for branching at a node of the search tree in order to extend a partially completed route $(x_1, x_k, \dots, x_i)$, and the alternative branching is to reject arc (x_i, x_j) as a possible extension of the partially completed route.

In choosing the arc (x_i, x_j) for branching, (which means that customer x_j is used to extend the route just after x_i), the following simple branching rules were applied.

(A) The starting of the first route

In this case, the arc (x_1, x_j) chosen for branching may be the arc from the depot to the unrouted customer x_j nearest the depot so far. (If there are more than one customer, choose the one with the biggest demand.) Such a customer has a good chance to be the first customer in this route.

(B) The extension of a partially completed route

If a partially completed route must be extended, the customer to be chosen for branching is the nearest customer to the route end, without violation of the vehicle capacity. By choosing this type of customer as the branching customer, we expect that the alternative branching (the rejection of the extension of the partially completed route with that customer) will result in a higher lower bound, and result in an early backtracking from the corresponding node.

(C) The finishing of a partially completed route

A partially completed route is "finished" when one branches on the arc (x_i, x_1).

(D) The starting of a new route

If the branchings so far have produced a set of completed routes, a new route must be started again. In this case, the customer chosen for branching is again the customer not yet routed, and nearest from the depot without violating the vehicle capacity.

If an arc (x_i, x_j) is chosen for branching at a node of the tree search in order to extend a partially completed route $(x_1, x_k, \ldots, x_i)$, the cost is changed as follows :

$$c_{il} = c_{li} = \infty \text{ for } l = 2, \ldots, n \text{ and } l \neq i, j.$$

For the alternative branching which is to reject arc (x_i, x_j) as a possible extension of the partially completed route, the cost is changed as follows :

$$c_{ij} = c_{ji} = \infty.$$

5.2.3 Fathoming

A node of the search tree is "fathomed" when the following situation has arisen :

(i) if a lower bound at a certain stage is greater than or equal to the current upper bound (the best solution so far), and backtracking can then occur,

or (ii) if the solution corresponding to the lower bound at a subproblem, is a feasible solution,

or (iii) if the bottom of the tree is reached and no unvisited customers remain.

5.3 Comments on the algorithm

In order to increase the efficiency of the tree search algorithm above, we can consider the following :

(i) Whenever a route is completed, we can reduce the problem size by removing all customers of the completed route from the original problem. Any subsequent computation of the lower bound is then based on the reduced problem.

(ii) Since the initial lower bounds derived from the algorithm in Chapter 4 are close to the optimal solution (normally within 4%), the quantity k*ZL (for k= 1.3 say) can be used as the initial upper bound. In this case, because the quality of the initial upper bound is closer to the optimal solution value than any heuristic upper bound, the number of node is reduced greatly.

(iii) We can use the TSP algorithm for the last route in order to avoid to build the last partially completed route. We can also apply a TSP algorithm whenever a route is completed to determine whether the partial solution constructed so far is optimal or not.

(iv) We can consider the gap between the lower bound and the upper bound at a certain node. For example, if we assume that the initial costs are integers, the feasible solution value should be integer. Therefore, if the value of a lower bound at a certain node is between the upper bound and the upper bound - 1 (i.e. 0 < Zopt-LB < 1), we can backtrack from this node.

5.4 An example

We will use the same example as in Section 4.6 of the previous chapter. We will show the computational procedures only for algorithm 2 and the resulting tree of the tree search algorithm for two versions.

(A) Algorithm 1 (with bound LB1)

Fig. 5.2 shows the resulting tree search of algorithm 1 without any details. The number inside a tree node indicates the customer picked for branching on that node. A number $\overline{k}$ indicates that customer k is rejected as a candidate customer to extend the route at the stage corresponding to that node. The number on the top of the left to a node is the computed bound LB1.

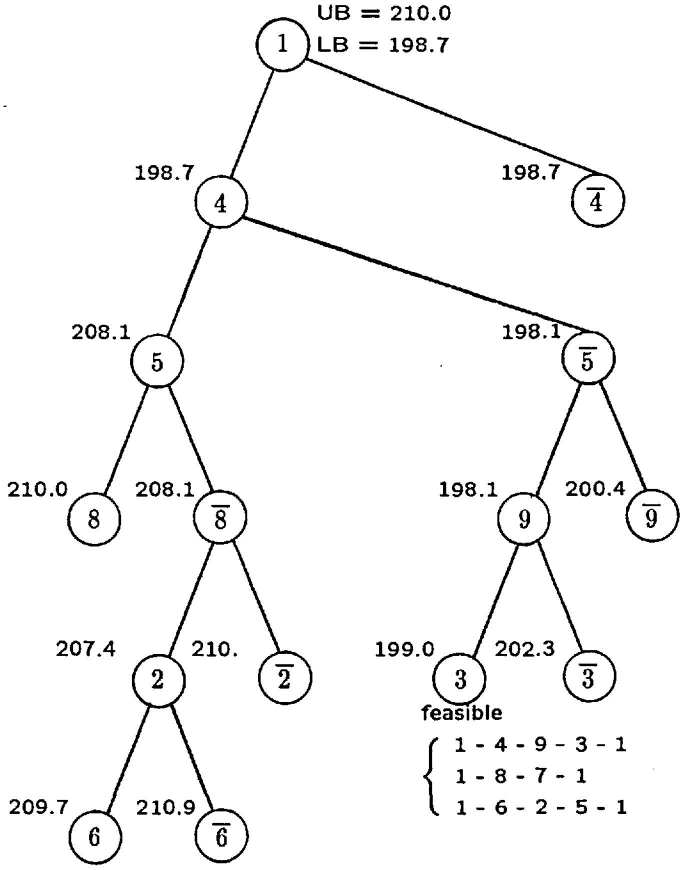


Figure 5.2 B & B tree for the example using the algorithm with bound LB1.

(B) Algorithm 2 (with bound LB2)

Step 1 : (Initialization).

Set L= 0, N = 0,

Zopt= 204.0 by using consideration (ii) in Sec. 5.3, i.e. ZL*1.04= 203.1→204

Z(0) = LB_0= 195.34 (the best lower bound at the root node).

Step 2 : (Check whether the set of possible customers to branch on is empty or not).

Since the set of possible customers to branch on is not empty, go to step 3.

Step 3 : (Branching).

Customer x_4 is chosen to branch to node 1.

Number N= N+1 = 0+1= 1 and L= L+1 = 0+1= 1.

Then, go to step 4.

Step 4 : (Branch to node N and update the cost matrix).

Branch to node 1 and update the cost matrix for the partially completed route (x_1- x_4) so far. Then, go to step 5.

Step 5 : (Calculate the lower bound).

$Z(1) = LB_1= 195.34$, then go to step 6 to 9.

Step 6 to 9 : (Decide branching forward or backtracking).

Since Z(1) does not correspond to a feasible solution and Z(1)<Zopt, go to step 2.

Step 2 : Since the set of possible customers to branch on is not empty, go to step 3.

Step 3 : Choose another customer to extend this partially completed route (x_1, x_4) so far. Customer x_5 is chosen to branch to node 2.

Number N= N+1 = 1+1= 2 and L= L+1 = 1+1= 2.

Then, go to step 4.

Step 4 : Branch to node 2 and update the cost matrix for the partially completed route (x_1- x_4- x_5) so far as follows :

$c_{i4} = c_{4i} = \infty$, i = 2, ... , 9 and $c_{45} = c_{54} = 15$. Then, go to step 5.

Step 5 : $Z(2) = LB_2= 205.9$, then go to step 6 to 9.

Step 6 to 9 : Since Z(2) does not correspond to a feasible solution and Z(2)>Zopt, go to step 10 to backtrack.

Step 10 : (Backtracking).

Since the alternative node has not been examined, go to step 10a.

Step 10a : Number N= N+1= 2+1 = 3 and backtrack to node 3. Update cost matrix for the partially completed route (x_1- x_4) and rejection customer x_5 so far as follows :

$c_{i4} = c_{4i} = \infty$, i = 2, ... , 9 and $c_{45} = c_{54} = \infty$. Then, go to step 5.

Step 5 : Z(3) = LB_3= 195.3, then go to step 6 to 9.

Step 6 to 9 : Since Z(3) does not correspond to a feasible solution and Z(1)<Zopt, go to step 2.

Step 2 : Since the set of possible customers to branch on is not empty, go to step 3.

Step 3 : Customer x_9 is chosen to branch to node 4.

Number N= N+1 = 3+1= 4 and L= L+1 = 2+1= 3.

Then, go to step 4.

Step 4 : Branch to node 4 and update the cost matrix. Then, go to step 5.

Step 5 : Z(4) = LB_4= 195.3, then go to step 6 to 9.

Step 6 to 9 : Since Z(4) does not correspond to a feasible solution and Z(4)<Zopt, go to step 2.

Step 2 : Since the set of possible customers to branch on is not empty, go to step 3.

Step 3 : Customer x_3 is chosen to branch to node 5.

Number N= N+1 = 4+1= 5 and L= L+1 = 3+1= 4.

Then, go to step 4.

Step 4 : Branch to node 5 and update the cost matrix. Then, go to step 5.

Step 5 : Z(5) = LB_5= 195.4, then go to step 6 to 9.

Step 6 to 9 : Since Z(5) does not correspond to a feasible solution and Z(5)<Zopt, go to step 2.

....

Step 2 : Since the set of possible customers to branch on is not empty, go to step 3.

Step 3 : Customer x_8 is chosen to branch to node 7.

Number N= N+1 = 6+1= 7 and L= L+1 = 5+1= 6.

Then, go to step 4.

Step 4 : Branch to node 7 and update the cost matrix after removing the elements of the completed route (x_1- x_4- x_9- x_3- x_1) from the original problem by using consideration (i) in Section 5.3. Then, go to step 5.

Step 5 : Z(7) = LB_7= 195.4, then go to step 6 to 9.

Step 6 to 9 : Since Z(7) does not correspond to a feasible solution and Z(7)<Zopt, go to step 2.

Step 2 : Since the set of possible customers to branch on is not empty, go to step 3.

Step 3 : Customer x_7 is chosen to branch to node 8.

Number N= N+1 = 7+1= 8 and L= L+1 = 6+1= 7.

Then, go to step 4.

Step 4 : Branch to node 8 and update the cost matrix after removing the elements of the completed route (x_1- x_4- x_9- x_3- x_1) from the original problem by using consideration (i) in Section 5.3. Then, go to step 5.

Step 5 : Z(8) = LB_7= 199.0, then go to step 6 to 9.

Step 6 to 9 : Since Z(8) corresponds to a feasible solution and Z(8)<Zopt, set Zopt= Z(8)= 199.0 and record the routes corresponding to this solution value.

Then, go to step 10 to backtrack.

The q-routes corresponding to the current feasible solution value (199.0) to this subproblem are as :

Route 1 : x_1 - x_6 - x_2 - x_5 - x_1

Route 2 : x_1 - x_8 - x_7 - x_1

Route 3 : x_1 - x_4 - x_9 - x_3 - x_1

Step 10 : Since the alternative node has not been examined, go to step 10a.

Step 10a : Number N= N+1= 8+1 = 9, backtrack to node 9 and update cost matrix.

Then, go to step 5.

Step 5 : $Z(9) = LB_9 = 200.8$, then go to step 6 to 9.

Step 6 to 9 : Since Z(9) does not correspond to a feasible solution and Z(9)>Zopt, go to step 10 to backtrack.

....

Step 10 : Since the alternative node has not been examined, go to step 10a.

Step 10a : Number N= N+1= 12+1 = 13, backtrack to node 13 and update costs. Then, go to step 5.

Step 5 : $Z(13) = LB_{13} = 199.7$, then go to step 6 to 9.

Step 6 to 9 : Since Z(13) does not correspond to a feasible solution and Z(13)>Zopt, go to step 10 to backtrack.

Step 10 : Since the alternative node has been examined, go to step 10b.

Step 10b : Set L= L-1= 3-1 = 2 and reset the costs accordingly. Then, go to step 10c.

Step 10c : Since $L \neq 0$, go to step 10.

Step 10 : Since the alternative node has been examined, go to step 10b.

Step 10b : Set L= L-1= 2-1 = 1 and reset the costs accordingly. Then, go to step 10c.

Step 10c : Since $L \neq 0$, go to step 10.

Step 10 : Since the alternative node has not been examined, go to step 10a.

Step 10a : Number N= N+1= 13+1 = 14, backtrack to node 14 and update costs. Then, go to step 5.

Step 5 : $Z(14) = LB_{14} = 202.0$, then go to step 6 to 9.

Step 6 to 9 : Since Z(14) does not correspond to a feasible solution and Z(14)>Zopt, go to step 10 to backtrack.

Step 10 : Since the alternative node has been examined, go to step 10b.

Step 10b : Set L= L-1= 1-1 = 1 and reset the costs accordingly. Then, go to step 10c.

Step 10c : Since L= 0, stop.

Thus, this algorithm 2 is terminated at the 14th node with the optimal solution value 199.0 The complete tree search for this example VRP is shown in Fig. 5.3.

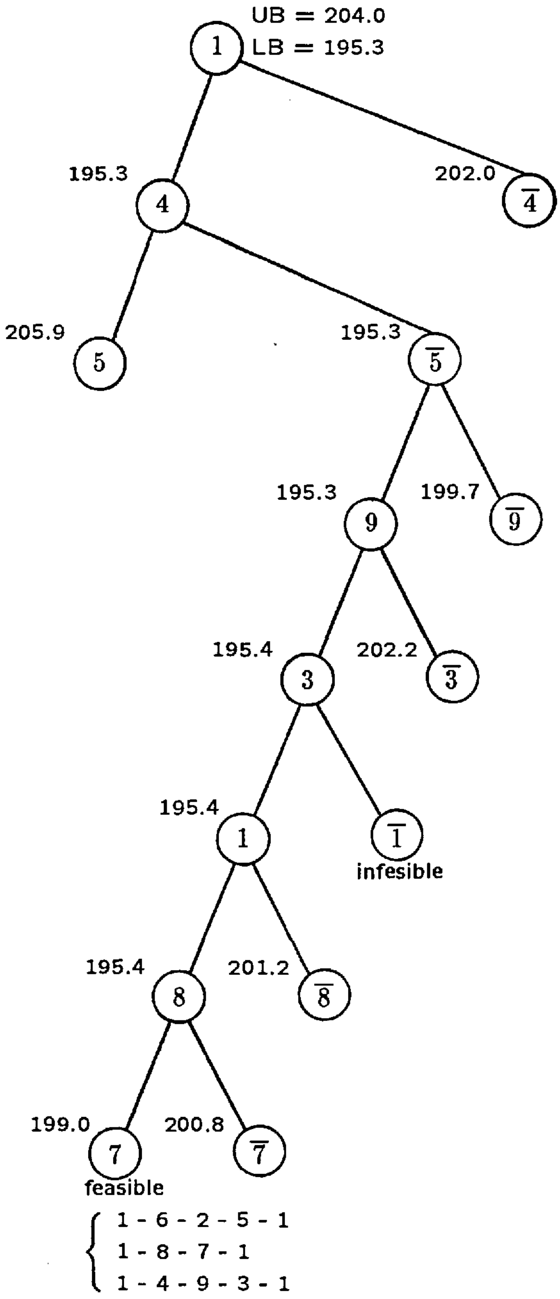


Figure 5.3 B & B tree for the example using the algorithm with bound LB2.

5.5 Computational results

In this section we show the computational performance of the algorithm using bounds LB1 and LB2. Sixteen problems are used for the tests ranging from N= 9 to N= 100 vertices (1 depot + customers). One of these problems (problem 2) is from the literature (see Christofides *et. al.* [1981 a]), problems 12 to 16 are from the 100 - vertex problem (except for the changed quantity of demand) in Eilon *et. al.* [1971] and others are randomly generated.

In Table 5.1, we show the size of the test problems, and Table 5.2 shows the values of the optimal solution to the test problems, together with the values of the initial lower bounds (LB1 and LB2) and the computing times of a initial lower bound (using an IBM PS/2 70-386) for the root node.

Table 5.3 gives the computational performance of the two algorithms (algorithm 1 and 2) for nine test problems (problem 1 to 9).

In Table 5.4a (problem description) and Table 5.4b, we compare the results of the initial lower bounds and number of nodes in the branch and bound tree for algorithm 1 and 2 with those of the literature (Christofides *et. al.* [1981a]) for five test problems.

Table 5.3 shows the computing times and total number of nodes in the branch and bound tree for each algorithm. All computing times shown in Table 5.2 and 5.3 are times on the IBM PS/2 - 70 386 using the Microsoft fortran 4.0 compiler.

It should be noted from Table 5.2 that bound LB2 is on average within 1.10% of the optimum solution value and on no occasion is the bound worse than 3.12%. This would suggest that on many practical occasions, a currently available solution to the VRP may be guaranteed (by using the bound) to be close enough to the optimal not to warrant the continuation of the search for an improved solution. It should also be noted that all the values of the initial lower bounds for the direct bound (LB1) and the indirect bound (LB2) and the number of tree nodes required to obtain the optimal solution value are better than those found in the literature. Also we note that general 40-customer VRPs can be solved with algorithm 2 developed here, and this is larger than what is in the literature.

Table 5.1 Test problems.

Problem	Number of vertices	Total demand	Number of vehicles	Vehicle capacity	Source	
1	9	15	3	6	Given as an example	
2	10	28	3	10	Test problem 1 in Appendix A	
3	11	93	4	24	Christofides *et. al.* [1981 a]	
4	15	56	3	20	Test problem 2 in Appendix A	
5	20	80	3	30	Test problem 3 in Appendix A	
6	25	97	4	30	Test problem 4 in Appendix A	
7	30	105	4	30	Test problem 5 in Appendix A	
8	35	108	4	30	Test problem 6 in Appendix A	
9	40	140	5	30	Test problem 7 in Appendix A	
10	45	141	5	30	Test problem 8 in Appendix A	
11	50	147	5	30	Test problem 9 in Appendix A	
12	60	192	5	40	No. 1 to 59	in the 100 customer problem in Eilon *et. al.* [1971]*.
13	70	220	6	40	No. 1 to 69	
14	80	275	6	50	No. 1 to 79	
15	90	316	7	50	No. 1 to 89	
16	100	339	7	50	No. 1 to 99	

* For the problem 12 to 16, demand of each customer is shown in Appendix A.

Table 5.2 Computational results : values and times.

Problem	Optimal solution value	Initial lower bounds			
		LB1		LB2	
		bound	time**	bound	time**
1	199.0	198.73	0 : 2	195.34	0 : 3
2	295.0	294.57	0 : 4	295.0*	0 : 3
3	222.7	222.68	0 : 8	222.7*	0 : 8
4	361.0	350.41	0 : 22	357.36	0 : 28
5	406.0	387.22	1 : 08	393.32	1 : 28
6	476.0	462.20	1 : 50	470.73	2 : 13
7	623.0	610.79	2 : 42	617.36	3 : 14
8	659.0	637.18	3 : 50	651.53	4 : 28
9	783.0	713.44	5 : 02	767.85	5 : 43
10	-	782.85	6 : 23	849.23	7 : 13
11	-	847.54	8 : 00	913.91	8 : 58
12	-	463.50	15 : 50	561.81	17 : 54
13	-	497.81	21 : 47	655.34	24 : 04
14	-	442.26	36 : 12	665.24	40 : 19
15	-	466.49	45 : 52	705.11	50 : 20
16	-	488.61	56 : 45	726.93	61 : 12

* : Optimal solution value was obtained during the computation of the initial lower bound without embedding into the tree search.

** : minutes : seconds on IBM PS/2 70-386.

Table 5.3 Computational results of algorithms 1 and 2 : values & times

Problem	Algorithm 1 (LB1)		Algorithm 2 (LB2)	
	nodes	time	nodes	time
1	14	0:26	14	0:17
2	8	0:22	0	0:03
3	6	0:34	0	0:08
4	84	23:26	24	7:06
5	1112	15:56:55	264	3:46:06
6	-	-	352	5:46:18
7	-	-	620	14:43:41
8	-	-	890	19:33:51
9	-	-	550	22:46:19

Time : hours:nimutes:seconds on IBM PS/2 70-386 (using the Microsoft Fortran 4.0 compiler).
- : Time limit 24 hrs.

Table 5.4a Test problems for results of Table 5.4b.

Problem	Number of vertices	Total demand	Number of vehicles	Vehicle capacity
1	11	93	4	24
2	16	258	5	55
3	16	258	3	90
4	21	329	6	58
5	21	329	4	85

- Problem 1 is from Christofides *et. al.* [1981a].
- Other problems (2 to 5) are from 50-customer problem in Eilon *et. al.* [1971].
- Problem 2 & 3 : customers are the first 15 of the 50-customer problem.
- Problem 4 & 5 : customers are those numbered 11 to 30 in the 50-customer problem.

Table 5.4b Comparison of the results of algorithm 1 & 2 with those of literature : values (bounds and total number of nodes).

Prob.	Optimal	Initial lower bounds				Nodes			
	solution	Algo. 1	a	Algo. 2	b	1	a	2	b
1	222.7	222.6	211.0	222.7*	222.7*	8	49	0	0
2	334.1	323.9	298.1	325.5	321.4	536	3336	86	194
3	277.9	266.9	252.1	267.8	265.5	252	2148	188	498
4	429.9	413.6	381.2	429.9*	429.7	1382	-	0	6
5	357.6	341.5	260.0	346.8	346.4	-	-	208	886

- a and b are algorithms from Christofides *et. al.* [1981a].

* : optimal solution obtained during computation of the initial bound.

CHAPTER 6

CONCLUSIONS

The travelling salesman problem (TSP) and vehicle routing problem (VRP) are two, both theoretically and practically important, problems in the class of combinatorial optimization problems. Both problems have been studied extensively over the last 25 years and there is a voluminous literature on the subject. For the TSP exact algorithms have continuously improved over this period, and problem sizes that can be solved optimally have increased from 50 or so vertices in the early 1970's to several hundred vertices now. For the VRP (which is the more practically useful of the two problems), the situation has been very different. 25-customer VRP's could be solved optimally 10 years ago and the problem size that can now be solved optimally involves no more than 30 or so customers ; although occasionally very particular VRP's (which are in one way or other "easy") have been solved for sizes up to 100 customers. Thus, although virtually identical algorithmic principles are used for the VRP as for the TSP the results obtained are very different. This situation arises from the fact that many algorithmic developments (e.g. Lagrangean Relaxation) are developed and refined on the TSP and - based on the premise that the TSP and VRP are closely related - are then applied to the VRP. This process guarantees that only procedures that have worked well for the TSP are ever tried on the VRP. The process has failed to produce major advances to the solution of VRP's because of the dramatic degradation of algorithm performance resulting when the structure and constraints imposed by the VRP are encountered.

This thesis develops algorithms for the TSP and VRP based on tree-searches incorporating bounds computed from state-space relaxation. State-space relaxation (which is to Dynamic Programming what Lagrangean relaxation is to Integer programming) is used to develop bounds for the TSP and VRP which for the case of the VRP is superior to the bounds that can be found in the literature. In particular the VRP bound derived in the thesis is superior to bounds that are obtained from lagrangean relaxation for this problem. The TSP bound derived in the thesis is, on the other hand, far inferior both in terms of quality and in computational effort than bounds for the TSP that can be found in the literature and which are based on lagrangean relaxation.

As a result of the above comments, the computational results obtained in the thesis are as one would expect. The performances of the TSP and VRP algorithms derived from state-space relaxation are not too different from one another, and in fact larger VRP's (40 customers) than TSP's (30 vertices) can be solved. This performance is very poor for the TSP (when compared to other methods in the literature) but is much better for general VRP's than what can be found in the literature for that problem. Indeed, the TSP is discussed in the thesis only because it provides an easy introduction to the state-space relaxation methodology.

The major contribution of this thesis, is to derive better bounds for the VRP and incorporate these into an algorithm that enables the solution of VRP's of medium (40 customers) size. No attempt has been made to improve the complete algorithm (by investigating various ways of branching, for example) and this is suggested as a useful avenue for future research. It is expected that VRP's with more than 50 customers could be solved optimally using the bounds derived in this thesis, and incorporating an improved branching scheme.

APPENDIX A : Details of test problems

Test problem 1 : 10 vertices

No.	1	2	3	4	5	6	7	8	9	q
1	-									
2	22									2
3	32	23								3
4	22	31	22							5
5	31	50	45	23						2
6	35	58	60	40	21					4
7	20	41	51	36	31	22				6
8	27	41	58	49	51	41	20			1
9	30	28	50	50	61	57	37	21		3
10	36	20	41	50	67	70	50	42	20	2

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 10 units. q means demand in units.

Test problem 2 : 15 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	20														3
3	40	20													7
4	33	18	17												2
5	22	22	36	20											6
6	16	29	46	35	16										1
7	40	45	56	38	21	25									2
8	35	49	67	52	32	21	23								8
9	30	50	70	61	44	29	48	31							3
10	14	31	51	47	35	24	52	41	22						3
11	36	51	66	67	58	47	73	60	32	21					5
12	21	28	44	45	41	35	60	54	36	13	21				4
13	34	38	49	55	55	51	75	69	49	28	25	15			1
14	41	35	39	51	56	55	77	76	62	37	41	25	14		6
15	28	15	21	30	37	43	61	64	57	34	47	26	27	20	5

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 20 units. q means demand in units.

Test problem 3 : 20 vertices

No.	x_j	dist.	No.	x_j	dist.	No.	x_j	dist.	No.	x_j	dist.	No.	x_j	dist.
1	2	20	3	6	44	5	14	56	8	14	45	12	15	17
	3	35		7	47		15	26		15	55		16	22
	4	14		8	73		16	45		16	67		17	32
	5	35		9	65		17	39		17	60		18	35
	6	34		10	49		18	49		18	68		19	41
	7	22		11	68		19	60		19	77		20	53
	8	43		12	56		20	46		20	85	13	14	23
	9	31		13	76	6	7	19	9	10	16	(2)	15	21
	10	15		14	53	(5)	8	38	(1)	11	14		16	35
	11	33		15	57		9	40		12	22		17	49
	12	23		16	43		10	34		13	35		18	48
	13	42		17	28		11	53		14	28		19	50
	14	21		18	35		12	50		15	39		20	74
	15	28		19	44		13	71		16	42	14	15	10
	16	20		20	24		14	53		17	48	(1)	16	15
	17	17	4	5	21		15	62		18	54		17	26
	18	26	(7)	6	26		16	54		19	63		18	28
	19	37		7	22		17	48		20	74		19	35
	20	42		8	43		18	59	10	11	20		20	54
2	3	16		9	40		19	70	(5)	12	15	15	16	14
(4)	4	14		10	25		20	65		13	36	(2)	17	30
	5	26		11	45	7	8	25		14	20		18	27
	6	38		12	35	(12)	9	22		15	30		19	29
	7	36		13	57		10	16		16	29		20	53
	8	61		14	35		11	33		17	31	16	17	14
	9	51		15	42		12	31		18	39	(15)	18	13
	10	36		16	31		13	51		19	48		19	19
	11	52		17	22		14	35		20	58		20	38
	12	41		18	34		15	46	11	12	13	17	18	11
	13	60		19	45		16	41	(3)	13	21	(3)	19	22
	14	37		20	41		17	39		14	20		20	27
	15	42	5	6	24		18	48		15	27	18	19	12
	16	28	(1)	7	35		19	59		16	35	(1)	20	26
	17	14		8	60		20	63		17	45	19	20	29
	18	23		9	57	8	9	16		18	48	(2)		
	19	35		10	44	(8)	10	28		19	55	20	-	-
	20	26		11	65		11	29		20	73	(5)		
3	4	25		12	57		12	37	12	13	21			
(2)	5	22		13	78		13	50	(1)	14	7			

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. (q) means demand in units.

Test problem 4 : 25 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	12														3
3	17	12													6
4	38	26	27												1
5	41	39	26	19											2
6	31	27	13	26	14										2
7	47	43	34	53	37	27									10
8	22	27	15	41	34	20	24								1
9	44	51	40	65	54	40	21	25							3
10	17	28	23	49	47	32	36	14	28						4
11	31	41	34	61	56	41	35	22	17	12					4
12	23	37	36	61	62	48	50	30	37	17	19				3
13	37	50	47	74	71	57	54	38	35	25	13	13			2
14	48	56	58	82	83	70	70	51	52	38	36	22	16		10
15	31	42	47	68	73	61	68	45	55	32	36	16	25	16	3
16	10	22	27	48	52	41	52	28	45	18	29	17	30	35	3
17	20	32	38	58	63	51	61	37	50	25	33	16	27	26	5
18	22	30	39	54	63	54	68	44	61	35	45	29	41	37	1
19	54	62	72	84	95	86	98	75	87	62	69	50	57	44	2
20	43	45	58	62	77	71	90	66	86	59	70	54	65	58	4
21	45	42	56	53	70	67	89	67	89	62	75	61	74	70	3
22	19	18	31	38	50	44	64	41	64	37	50	39	52	53	11
23	32	26	40	37	54	51	74	52	76	50	63	53	66	66	2
24	42	34	47	35	55	56	80	61	85	60	79	64	78	78	5
25	24	12	24	18	33	32	56	37	62	40	54	49	62	67	7

No.	15	16	17	18	19	20	21	22	23	24
16	21									
17	12	10								
18	21	17	11							
19	33	46	38	31						
20	42	41	39	26	28					
21	53	46	47	34	46	15				
22	37	22	28	19	47	28	27			
23	50	36	41	29	53	27	18	13		
24	61	48	53	42	63	35	19	27	11	
25	52	33	43	37	67	45	38	20	22	25

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. q means demand in units.

Test problem 5 : 30 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	45														4
3	23	22													1
4	57	32	37												2
5	55	50	43	24											15
6	31	34	20	29	25										2
7	17	36	14	41	38	14									3
8	25	48	26	45	34	17	13								2
9	45	64	30	49	28	29	34	21							4
10	22	64	41	69	59	41	30	21	39						3
11	27	59	28	56	42	28	23	11	21	18					7
12	51	81	47	70	49	47	46	32	21	35	24				4
13	63	101	70	95	75	69	65	53	46	42	43	25			1
14	53	95	66	94	76	66	58	50	50	31	38	33	18		3
15	34	78	54	83	71	54	43	49	48	13	30	39	37	21	12
16	50	92	80	108	101	80	66	68	83	44	60	73	68	50	2
17	45	81	76	102	99	76	62	67	85	47	64	81	80	63	3
18	27	71	56	85	78	77	44	45	61	23	40	56	58	41	4
19	12	56	43	70	66	43	30	34	53	20	33	54	62	48	7
20	45	68	57	94	98	73	61	70	91	58	71	93	98	82	2
21	19	50	35	72	73	48	35	44	66	36	46	69	78	64	5
22	42	59	51	87	93	68	57	68	89	59	70	93	101	86	1
23	30	50	40	77	82	57	45	66	77	49	59	82	91	77	2
24	13	34	16	53	56	31	20	32	53	35	37	62	77	66	5
25	38	38	35	68	78	55	46	59	80	60	65	89	102	90	1
26	27	32	23	59	67	42	39	47	67	49	53	77	91	80	2
27	30	16	11	41	51	29	21	39	58	51	48	72	90	82	1
28	61	56	58	88	100	78	70	83	103	81	88	112	124	110	3
29	64	47	54	80	95	74	69	82	102	86	90	114	128	116	2
30	53	20	35	52	71	53	51	65	83	76	74	97	116	107	2

No.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
16	35														
17	45	16													
18	20	20	25												
19	28	35	34	15											
20	61	42	25	41	40										
21	43	41	32	27	14	27									
22	65	50	35	45	40	10	25								
23	56	48	34	38	30	18	13	12							
24	47	56	49	37	22	42	19	38	26						
25	69	66	53	54	42	33	27	23	19	28					
26	60	62	52	47	33	37	21	30	20	14	11				
27	64	75	66	56	41	54	35	47	37	19	28	19			
28	88	78	62	71	62	37	47	28	33	51	23	36	49		
29	95	89	75	80	68	50	53	40	43	53	26	37	45	16	
30	87	91	80	76	62	60	51	50	46	41	29	30	26	38	26

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. q means demand in units.

Test problem 6 : 35 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	50														2
3	30	24													4
4	49	21	35												7
5	32	35	25	17											3
6	47	26	42	16	17										1
7	14	37	20	35	19	35									5
8	34	45	41	39	19	18	26								4
9	9	52	34	48	30	43	15	27							2
10	21	48	36	39	22	31	17	15	13						2
11	21	60	45	50	34	42	25	14	12	13					5
12	45	66	61	50	39	45	42	21	36	26	25				1
13	36	74	61	63	47	51	41	32	28	26	12	22			3
14	56	85	78	70	58	55	58	40	48	41	35	20	22		1
15	34	82	65	75	58	66	45	47	30	36	24	41	20	38	1
16	53	94	81	82	67	71	60	52	47	46	35	39	19	22	10
17	45	95	75	88	71	80	57	62	43	50	38	75	33	49	7
18	64	114	91	112	95	106	77	89	65	75	65	85	63	80	4
19	50	110	78	98	81	93	63	75	50	60	50	72	50	70	3
20	44	95	72	94	61	90	59	73	47	59	50	73	53	73	8
21	28	78	56	78	61	74	41	57	31	43	35	60	41	63	1
22	20	69	51	63	46	55	32	38	17	25	14	38	20	43	2
23	10	61	39	60	43	56	25	41	14	27	23	49	35	56	1
24	41	85	60	89	73	89	54	75	47	60	56	82	64	86	2
25	26	71	46	74	58	73	38	60	32	45	41	67	51	74	2
26	13	55	32	58	42	58	23	46	21	33	32	57	45	66	4
27	40	76	51	83	68	85	49	74	48	60	58	83	69	91	2
28	21	49	25	56	42	59	24	51	29	39	42	65	56	77	1
29	13	42	19	45	30	47	12	38	19	27	32	53	47	67	1
30	42	56	35	70	59	76	44	70	50	60	63	85	78	98	3
31	57	71	49	85	74	92	60	86	65	75	78	101	92	113	1
32	44	44	26	60	52	69	42	66	52	58	64	84	80	99	4
33	62	63	46	80	73	89	61	87	70	78	82	104	97	117	2
34	58	42	32	60	57	73	51	74	63	68	76	93	92	109	6
35	50	25	20	44	42	56	40	60	55	57	67	81	82	98	2

test problem 6 (cont.)

No.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
16	24														
17	15	31													
18	44	60	31												
19	32	51	20	14											
20	34	56	27	22	10										
21	26	49	26	35	22	16									
22	15	34	27	52	38	37	23								
23	28	50	36	53	39	33	14	15							
24	49	73	46	41	32	23	23	45	33						
25	40	63	41	46	34	26	15	32	18	16					
26	40	61	46	58	45	39	24	28	14	31	16				
27	57	81	57	55	45	36	31	49	35	14	18	28			
28	52	74	59	79	57	49	35	40	24	35	23	15	27		
29	47	66	57	73	59	53	37	33	21	44	29	14	38	12	
30	72	93	77	81	70	60	51	60	44	41	36	33	27	21	32
31	84	107	86	85	76	67	60	73	57	44	45	46	30	36	48
32	78	98	84	92	81	72	60	65	48	53	45	37	40	25	32
33	92	114	96	98	88	79	71	80	64	58	55	53	43	41	50
34	92	110	99	107	96	87	75	78	63	69	61	52	55	39	45
35	85	102	94	107	94	87	72	71	58	72	61	49	60	37	37

No.	30	31	32	33	34
31	16				
32	14	25			
33	20	15	20		
34	29	35	16	23	
35	35	47	32	38	27

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. q means demand in units.

Test problem 7 : 40 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	40														3
3	51	25													10
4	64	41	16												8
5	28	22	21	36											2
6	42	37	22	25	22										2
7	65	54	31	21	43	22									1
8	13	29	36	49	13	29	51								3
9	45	53	38	39	35	17	25	36							1
10	34	47	41	45	29	20	36	27	12						1
11	20	44	45	53	26	28	49	17	26	15					2
12	44	59	47	49	40	25	35	38	9	11	24				4
13	45	70	62	65	50	40	51	45	26	21	27	16			5
14	13	50	60	72	36	49	70	23	49	37	22	45	43		20
15	20	55	59	68	38	43	63	27	39	28	15	32	27	16	2
16	30	69	75	85	54	60	78	41	53	42	31	46	36	21	1
17	17	57	62	72	41	48	69	27	45	33	19	39	34	9	2
18	42	75	74	79	57	53	67	47	41	34	31	32	18	36	1
19	27	55	53	60	36	35	53	26	28	17	10	21	17	26	3
20	57	92	82	85	69	60	70	60	47	41	42	36	20	51	4
21	46	83	83	90	65	65	79	54	48	45	39	44	30	38	2
22	64	98	96	100	80	75	86	69	61	55	53	51	35	56	3
23	69	101	96	99	82	75	85	73	60	55	57	50	34	63	1
24	49	90	96	105	74	80	97	60	72	61	51	64	51	39	2
25	36	75	85	96	62	72	91	49	67	56	43	61	52	25	3
26	51	98	102	115	79	92	112	65	88	77	63	83	74	42	3
27	66	100	117	130	94	110	130	81	107	96	83	104	95	60	1
28	52	83	102	116	79	95	117	67	96	84	70	92	86	47	6
29	42	75	93	106	69	84	105	55	84	73	58	81	74	35	7
30	28	63	79	92	55	70	92	42	71	60	45	68	63	22	1
31	51	76	96	111	74	93	115	64	97	85	71	95	90	49	1
32	42	60	83	98	61	81	103	53	87	76	61	86	85	42	4
33	12	48	64	78	41	58	80	29	61	50	35	59	58	20	3
34	30	49	70	85	49	69	91	39	75	65	50	75	74	31	2
35	21	33	54	70	33	54	76	26	62	53	40	64	66	27	10
36	47	50	76	92	58	80	101	53	90	80	67	91	93	51	3
37	28	21	44	61	27	50	70	25	60	53	43	64	70	38	1
38	56	40	66	82	53	74	95	51	86	78	67	89	94	56	5
39	69	44	70	84	60	83	99	62	94	88	78	99	105	70	1
40	46	15	40	55	33	53	70	40	67	63	56	74	82	56	6

test problem 7 (cont.)

No.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
16	17														
17	7	14													
18	21	22	26												
19	11	28	18	20											
20	36	36	41	15	32										
21	27	18	28	12	30	20									
22	44	37	46	23	43	15	18								
23	48	45	51	26	46	14	26	10							
24	36	20	34	33	45	42	22	34	44						
25	30	16	24	37	40	50	30	48	56	17					
26	51	37	45	59	62	70	50	65	74	30	23				
27	70	59	63	81	81	93	72	87	96	52	44	22			
28	61	51	54	73	73	87	66	83	92	49	36	22	16		
29	48	39	41	61	59	75	55	72	81	39	25	13	25	14	
30	36	30	29	52	47	67	48	67	74	37	20	25	38	25	14
31	62	58	55	80	74	94	76	93	102	61	45	34	29	16	23
32	58	56	51	77	68	92	75	93	100	64	48	43	43	28	26
33	30	34	25	52	39	67	52	71	77	50	33	43	57	42	29
34	48	48	42	68	56	83	67	85	93	60	42	45	50	35	30
35	42	48	38	64	47	78	66	84	90	65	47	56	65	50	43
36	67	70	62	88	75	103	88	106	114	81	63	63	63	48	46
37	50	59	47	71	53	85	75	93	98	76	60	70	79	63	56
38	71	77	66	92	76	107	95	113	118	91	73	76	78	63	61
39	85	90	81	105	90	119	109	126	131	106	88	91	92	77	76
40	65	76	64	86	66	100	93	108	112	95	80	89	97	81	76

No.	30	31	32	33	34	35	36	37	38	39
31	28									
32	26	15								
33	17	40	31							
34	22	25	13	25						
35	31	42	28	19	16					
36	43	33	19	39	21	27				
37	45	54	39	31	39	13	32			
38	54	49	34	46	32	28	16	28		
39	69	63	49	60	47	42	30	35	15	
40	64	70	55	50	46	32	41	19	30	29

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. q means demand in units.

Test problem 8 : 45 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	35														5
3	56	22													2
4	58	29	18												8
5	66	44	35	17											11
6	44	25	29	18	22										1
7	50	35	36	22	18	9									3
8	54	42	43	28	19	17	7								2
9	28	17	33	30	38	16	24	30							2
10	35	25	36	27	30	9	14	20	11						5
11	42	34	41	29	28	12	9	12	19	8					1
12	48	42	46	33	29	7	10	7	27	16	8				1
13	36	30	40	31	32	13	14	18	14	5	5	14			4
14	20	38	55	50	53	33	36	38	22	23	26	32	21		4
15	41	57	71	61	57	42	40	38	39	35	31	31	30	22	3
16	16	50	71	70	75	54	57	60	40	44	48	53	42	21	1
17	32	60	79	74	74	56	56	56	46	46	46	48	42	23	2
18	35	65	84	80	80	61	63	63	51	52	52	55	48	29	1
19	41	70	89	83	83	64	64	64	56	55	55	56	51	33	2
20	39	73	92	88	89	70	71	72	59	60	61	64	57	37	2
21	45	76	94	88	88	70	70	69	61	61	60	61	57	38	1
22	49	77	94	87	85	68	67	66	61	59	58	58	55	38	2
23	57	80	96	86	82	68	65	63	64	60	57	55	55	43	7
24	49	80	99	95	95	77	77	77	66	67	68	70	64	44	1
25	52	83	100	94	93	76	76	75	68	67	66	67	63	45	15
26	48	83	104	102	105	85	87	88	72	75	77	81	73	51	3
27	70	98	113	105	101	87	84	82	81	78	76	75	74	60	2
28	70	100	122	127	137	115	120	124	98	106	112	118	106	86	3
29	62	94	116	120	128	106	111	115	90	98	102	108	97	76	3
30	44	79	101	103	109	88	92	95	72	79	86	88	78	56	1
31	42	74	95	99	108	86	91	96	70	77	84	89	78	57	2
32	54	77	97	105	117	95	102	108	79	88	95	102	89	72	4
33	25	58	80	84	92	70	76	80	54	62	67	73	61	42	2
34	15	41	63	68	78	56	63	68	40	49	56	63	50	35	5
35	5	36	58	61	70	48	54	59	36	40	46	54	40	25	3
36	24	43	64	71	83	62	69	75	45	55	63	70	57	43	1
37	56	68	85	97	111	91	99	106	75	86	94	102	88	76	1
38	13	34	56	61	72	49	57	62	33	42	50	58	44	31	3
39	11	27	49	54	64	42	49	55	25	36	43	50	37	27	2
40	37	46	66	76	89	69	77	83	53	64	72	80	67	56	6
41	19	31	53	60	72	50	58	64	34	44	53	60	47	36	2
42	28	23	42	52	66	47	55	62	32	42	51	59	46	42	3
43	49	40	52	67	83	66	75	82	54	64	73	80	68	65	4
44	78	63	68	85	103	90	99	106	80	90	98	106	94	93	2
45	51	21	24	40	57	45	53	61	39	46	54	62	51	58	3

test problem 8 (cont.)

No.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
16	36														
17	22	20													
18	28	19	7												
19	27	26	9	6											
20	36	24	16	8	9										
21	32	31	15	11	5	10									
22	28	34	16	15	8	16	6								
23	25	44	24	26	20	18	19	13							
24	40	32	21	15	13	8	8	14	27						
25	36	37	22	18	12	14	7	8	19	9					
26	54	33	33	25	27	18	25	31	44	17	26				
27	44	55	37	35	29	33	25	21	19	27	19	43			
28	99	65	79	73	78	68	77	84	96	70	80	53	96		
29	88	54	67	60	66	56	65	71	84	58	67	40	83	13	
30	67	34	46	40	45	36	45	51	64	39	48	23	66	32	21
31	72	37	53	49	54	46	56	61	74	51	59	36	78	28	21
32	90	54	73	69	75	68	77	83	95	73	82	59	100	29	31
33	59	23	41	38	44	38	47	53	64	45	53	35	72	45	36
34	56	24	44	44	50	47	55	59	69	55	61	50	80	60	53
35	46	19	36	37	45	42	49	52	61	50	55	49	73	67	59
36	65	34	54	53	60	56	64	68	78	64	70	56	89	58	53
37	98	64	85	83	89	83	92	97	109	90	98	79	117	54	57
38	53	25	44	45	51	49	56	60	69	58	63	54	80	67	60
39	49	27	43	45	51	51	57	59	67	59	63	58	81	74	67
40	78	48	68	68	74	70	78	72	93	78	85	70	103	62	60
41	59	33	51	53	59	57	64	67	76	65	71	61	88	70	64
42	64	44	60	63	69	68	73	76	84	76	81	74	98	81	76
43	88	64	82	84	90	88	95	98	106	96	102	91	120	85	84
44	116	93	112	113	120	117	124	127	135	125	131	119	149	105	106
45	78	67	80	84	89	90	95	96	100	99	102	98	117	107	103

No.	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
31	14														
32	36	22													
33	22	16	32												
34	39	32	39	18											
35	42	38	48	22	10										
36	42	33	34	22	9	19									
37	58	44	26	45	41	51	32								
38	45	39	45	24	7	7	13	44							
39	51	46	53	31	15	10	20	51	8						
40	53	41	34	34	23	31	14	22	25	30					
41	51	43	46	30	12	14	12	41	7	11	20				
42	64	55	56	42	24	25	23	45	19	17	24	13			
43	76	65	57	56	41	45	34	36	38	39	24	31	23		
44	102	90	76	83	70	75	62	50	67	69	49	60	51	29	
45	90	82	80	69	51	49	49	64	45	40	46	39	27	29	46

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. q means demand in units.

Test problem 9 : 50 vertices

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	q
1	-														
2	40														3
3	51	12													1
4	52	25	20												1
5	68	46	40	22											2
6	62	45	41	21	8										5
7	42	31	33	17	26	20									1
8	21	19	30	31	50	46	26								3
9	28	25	32	24	40	34	14	13							2
10	42	39	40	24	28	22	8	30	17						1
11	56	55	55	37	31	23	23	47	33	17					1
12	27	39	46	36	46	39	21	23	14	17	28				2
13	10	41	51	47	62	55	36	21	23	34	47	18			2
14	42	57	62	48	50	42	30	42	31	23	23	19	32		18
15	55	69	73	57	54	47	40	55	44	33	24	32	45	13	3
16	15	49	59	55	69	62	43	30	31	40	50	23	9	31	5
17	33	58	66	54	61	53	38	40	34	33	36	19	24	14	2
18	36	67	75	66	73	65	50	48	43	44	46	30	28	24	4
19	53	79	86	73	74	66	56	61	54	48	44	40	44	25	1
20	50	80	88	78	81	74	61	62	56	54	53	42	42	31	2
21	64	94	102	90	92	84	73	76	70	66	62	56	57	42	1
22	43	80	89	82	90	82	66	60	58	61	63	45	39	41	1
23	58	95	104	96	103	95	80	75	73	74	74	60	54	53	10
24	70	106	115	105	109	101	88	87	83	82	80	69	66	59	9
25	20	60	71	70	85	78	59	41	46	56	67	39	22	48	5
26	44	84	95	92	103	96	78	65	68	75	81	57	45	58	2
27	41	81	93	93	107	100	81	63	69	79	88	61	45	66	2
28	57	95	108	109	123	116	98	78	85	95	104	78	62	82	3
29	44	82	95	97	112	106	86	61	73	85	94	67	50	74	2
30	64	98	110	115	132	126	106	83	92	105	116	88	71	97	4
31	78	110	122	128	146	140	120	96	106	120	131	103	85	112	2
32	64	94	106	114	132	126	106	82	92	106	118	89	72	100	1
33	57	82	93	103	122	118	98	72	83	99	113	84	66	98	1
34	43	74	86	92	111	106	85	60	71	86	99	70	52	82	3
35	28	63	75	79	96	91	71	48	56	70	83	54	36	66	3
36	17	56	68	69	85	79	60	39	45	58	70	41	24	51	1
37	41	64	75	85	104	100	81	54	66	82	96	68	51	82	2
38	45	64	76	86	106	102	83	57	68	85	100	72	55	87	4
39	30	50	62	71	91	87	67	40	53	70	85	56	40	71	5
40	15	43	55	60	79	74	54	29	39	55	70	41	25	56	4
41	41	50	60	74	95	91	73	46	59	76	92	65	51	82	1
42	14	32	43	49	69	64	45	19	30	47	63	35	22	52	2
43	45	41	50	66	87	85	68	43	56	73	90	65	53	83	2
44	35	29	39	53	74	72	55	29	43	60	76	52	42	70	4
45	58	48	56	74	95	93	78	54	67	84	100	76	65	95	1
46	31	14	26	37	59	56	40	17	29	45	62	40	35	59	1
47	45	18	25	43	64	64	49	30	41	56	73	53	49	72	1
48	60	36	41	60	81	81	68	48	60	75	92	72	65	90	8
49	73	52	56	76	96	96	84	63	75	90	107	87	80	106	2
50	56	21	20	40	60	61	51	38	46	58	74	59	59	77	1

test problem 9 (cont.)

No.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
16	43														
17	23	20													
18	31	21	12												
19	22	38	21	18											
20	32	35	23	14	11										
21	38	49	36	28	17	14									
22	45	30	28	17	26	16	25								
23	54	45	41	30	32	22	22	15							
24	57	58	50	39	35	27	19	27	14						
25	58	17	35	30	48	41	54	29	42	56					
26	65	36	45	35	48	37	45	22	26	39	24				
27	75	38	53	45	60	51	60	35	41	54	22	15			
28	91	55	69	60	75	65	71	48	50	62	39	26	16		
29	84	45	61	53	70	61	70	45	50	64	28	25	10	13	
30	107	67	84	77	93	84	91	67	72	84	50	46	32	22	23
31	122	81	99	92	108	98	106	82	85	96	65	60	47	34	38
32	111	69	88	82	99	90	100	74	80	93	53	54	40	32	30
33	110	66	86	83	101	94	105	80	88	102	53	62	47	46	38
34	93	50	70	66	84	76	87	62	70	84	36	45	30	33	22
35	77	33	53	50	68	61	73	47	58	72	20	34	22	33	20
36	64	21	41	38	56	50	62	38	50	64	8	30	24	39	27
37	95	51	72	70	88	82	95	70	80	94	40	56	42	47	36
38	100	56	77	76	93	88	100	75	86	100	46	62	48	53	42
39	85	43	62	63	81	76	89	66	78	92	36	56	45	54	41
40	70	28	47	49	66	62	76	53	67	80	24	48	40	53	40
41	95	55	74	76	93	89	103	79	92	105	49	70	58	66	53
42	66	28	45	50	66	64	78	57	72	85	31	55	50	64	51
43	97	59	77	82	98	95	109	87	100	114	58	81	70	80	67
44	84	49	66	71	87	85	99	78	92	106	51	74	67	78	65
45	108	73	90	93	110	107	122	100	113	126	71	92	82	90	77
46	72	44	57	63	78	78	92	74	89	101	51	75	70	83	71
47	85	58	71	78	92	92	106	88	103	115	64	88	82	94	81
48	103	73	88	94	109	108	122	103	117	131	76	100	91	102	88
49	119	86	102	108	123	122	136	166	130	143	88	110	100	109	97
50	90	68	78	86	99	100	114	98	113	125	76	100	94	108	94

test problem 9 (cont.)

No.	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
31	15														
32	12	16													
33	30	34	20												
34	25	36	21	17											
35	35	49	36	33	16										
36	46	61	48	46	29	13									
37	38	48	31	18	15	22	32								
38	42	50	34	17	20	28	38	6							
39	50	61	44	31	25	22	27	14	16						
40	55	68	53	43	32	20	17	27	30	15					
41	58	66	51	34	34	36	41	20	17	14	27				
42	66	79	64	53	43	31	26	36	38	22	11	30			
43	73	81	65	48	48	48	50	34	31	25	34	15	31		
44	75	85	69	54	50	45	44	37	37	25	27	21	21	13	
45	80	87	72	53	57	59	63	42	38	36	46	24	44	12	24
46	84	96	80	67	60	50	46	49	50	36	30	37	20	28	15
47	92	103	87	71	67	61	58	55	54	42	41	38	32	25	17
48	95	104	88	70	71	69	69	57	54	47	52	37	45	23	25
49	100	107	91	72	77	78	80	62	58	55	63	43	58	29	37
50	106	117	100	84	81	74	70	68	67	56	54	51	44	38	30

No.	45	46	47	48	49
46	38				
47	30	14			
48	19	31	18		
49	19	46	34	15	
50	40	25	13	23	36

The distance matrix is symmetrical. The depot is denoted by 1.
Vehicle capacity : 30 units. q means demand in units.

Demand of each customer of problem 12 to 16 in Chapter 5.

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
q	0	3	1	1	2	5	1	3	2	1	1	2	2	18	3
No.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
q	5	2	4	1	2	1	1	10	9	5	2	2	3	2	4
No.	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
q	2	1	1	3	3	1	2	4	5	4	1	2	2	4	1
No.	46	47	48	49	40	51	52	53	54	55	56	57	58	59	60
q	1	1	8	2	1	5	3	2	11	3	3	8	6	2	2
No.	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
q	2	6	1	3	2	4	2	1	2	5	6	3	6	14	1
No.	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
q	7	4	2	2	10	5	3	2	4	6	1	9	3	3	5
No.	91	92	93	94	95	96	97	98	99	100					
q	1	2	3	2	2	3	1	5	2	2					

REFERENCES

S. Agmon [1954], "The relaxation Method for Linear Inequalities," *Canadian Journal of Mathematics*, *Vol.* 6, pp. 383 - 392.

A. A. Assad [1988], "Modeling and implementation Issues in Vehicle Routing," in Vehicle Routing : Methods and Studies, edited by B. L. Golden and A. A. Assad, North-Holand, Amsterdam, pp. 7 - 45.

E. Balas and M. W. Padberg [1976], "Set Partitioning : A Survey," *SIAM Rev. Vol.* 18, pp. 710 - 760.

E. Balas and P. Toth [1985], "Branch and Bound Methods," in *Traveling Salesman Problem*, E. L. Lawler *et al.* Eds., pp. 361 - 401.

M. L. Balinski and R. E. Quandt [1964], "On an Integer Program for a Delivery Problem," *Operations Research*, *Vol.* 12, pp. 300 - 304.

M. Ball, B. L. Golden, A. A. Assad and L. Bodin [1983], "Planning for Truck Fleet Size in the Presence of a Common Carrier Option," *Decision Sciences*, *Vol.* 14, pp. 103 - 120.

M. Ball and M. Magazine [1981], "The Design and Analysis of Heuristics," *Networks*, *Vol.* 11, pp. 215 - 219.

J. Beasley [1983], "Route First-Cluster Second Methods for Vehicle Routing," *Omega*, *Vol.* 11, pp. 403 - 408.

R. Bellman [1958], *Dynamic Programming*, Princeton University Press, Princeton.

E. Beltrami, N. Bhagat and L. Bodin [1971], *A Randomized Routing Algorithm with Application to Barge Dispatching in New York City*, Report 71-15, State University of New York, Stony Brook.

E. Beltrami and L. Bodin [1974], "Networks and Vehicle Routing for Municipal Waste Collection," *Networks*, *Vol.* 4, pp. 65 - 94.

G. E. Bennington and K. Rebibo [1975], *Overview of RUCUS Vehicle Scheduling Problem* (*BLOCKS*), in Preprints : Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, D. Bergman and L. Bodin, Eds.

L. Bodin [1975], "A Taxonomic Structure for Vehicle Routing and Scheduling Problems," *Comput. Urban Soc.*, *Vol.* 1, pp. 11 - 29.

L. Bodin [1983], *Solving Large Vehicle Routing and Scheduling Problems in Small Core*," Working paper MS/S 83-027, University of Maryland, College Park.

L. Bodin and L. Berman [1979], "Routing and Scheduling of School Buses by Computer," *Transportation Sciences*, *Vol.* 13, pp. 113 - 129.

L. Bodin and B. L. Golden [1981], "Classification in Vehicle Routing and Scheduling," *Networks*, *Vol.* 11, pp. 97 - 108.

L. Bodin, B. L. Golden, A. A. Assad and M. Ball [1983], "Routing and Scheduling of Vehicles and Crews : The State of the Art," *Computers and Operations Research*, *Vol.* 10, pp. 63 - 211.

L. Bodin and S. Kursh [1978], "A Computer-Assisted System for the Routing and Scheduling of Street Sweepers," *Operations Research*, *Vol.* 26, pp. 525 - 537.

L. Bodin and S. Kursh [1979], A Detailed Discription of a Computer Systemfor the Routing and Scheduling of Street Sweepers," *Comp. Oper. Res.*, *Vol.* 6, pp.182 - 198.

L. Bodin and T. Sexton [1979], *The Subscriber Dial-a-Ride Problem*, Final report, U.S. Dept. of Transportation, Urban Mass Transportation Administration, Washington, D.C.

T. B. Boffey [1984], *Graph Theory in Operations Research*, Macmillan Press, London.

K. Bott and R. Ballou [1986], "Research Perspectives in Vehicle Routing and Scheduling," *Transportation Research*, *Vol.* 20*A*, pp. 239 - 243.

N. Christofides [1971], "Fixed Routes and Areas for Delivery Operations," *Interact. Journal of Physical Distribution*, pp. 87 - 92.

N. Christofides [1975], *Graph Theory : An Algorithmic Approach*, Academic Press, London.

N. Christofides [1976], "The Vehicle Routing Problem," *Rev. Franc. Res. Oper.*, *Vol.* 10, pp. 55.

N. Christofides [1979], "The Traveling Salesman Problem," in *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth and C. Sandi, Eds., Wiley, Chichester.

N. Christofides, A. Mingozzi and P. Toth [1979 a], *Exact Algorithms for the TSP with Additional Constraints*, Report IC-OR-80-23, September, Imperial College of Science and Technology, London.

N. Christofides [1985 a], "Vehicle Routing," in *Traveling Salesman Problem*, edited by E. Lawler, J. Lenstra, A. Rinnooy Kan and D. Shmoys, Wiley, Chichester, pp. 431 - 448.

N. Christofides [1985 b], "Vehicle Routing," in *Combinatorial Optimization Annotated Bibliographies*, edited by M. O'hEigeartaigh, J. Lenstra, A. Rinnooy Kan, Wiley, Chichester, pp. 148 - 163.

N. Christofides and J. Beasley [1984], "Multiperiod Routing Problems," *Networks*, *Vol.* 14, pp. 237 - 256.

N. Christofides and S. Eilon [1969], "An Algorithm for the Vehicle Dispatching Problem," *Operational Research Q. Vol.* 20, pp. 309 - 318.

N. Christofides, A. Mingozzi and P. Toth [1979 b], *State-Space Relaxations for Combinatorial Problems*, Report IC-OR-79-09, July, Imperial College of Science and Technology, London.

N. Christofides, A. Mingozzi and P. Toth [1979 c], "The Vehicle Routing Problem," in *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth and C. Sandi Eds., Wiely, Chichester, pp. 315 - 338.

N. Christofides, A. Mingozzi and P. Toth [1981 a], "Exact Algorithms for the Vehicle Routing Problem Based on Spanning Trees and Shortest Path Relaxations," *Mathematical Programming*, *Vol.* 20, pp. 255 - 282.

N. Christofides, A. Mingozzi and P. Toth [1981 b], "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," *Networks*, *Vol.* 11, pp. 145 - 164.

N. Christofides, A. Mingozzi and P. Toth [1981 c], *An Algorithm for the Time Constrained TSP*, Report IC-OR-81-12, Imperial College of Science and Technology, London.

N. Christofides, A. Mingozzi and P. Toth [1982], *MOVER* (*Modeling and Optimization of Vehicle Routing*) - *A User Manual*, Imperial College of Science and Technology, London.

N. Christofides and J. Paixao [1982], *State-Space Algorithms for the Set Covering Problem*, Report IC-OR-82-3, Imperial Clloege of Science and Technology, London.

N. Christofides and M. Thornton [1982], *A Shortest Path Algorithm for Generalized Matchings*, Report IC-OR-82-2, Imperial College of Science and Technology, London.

I. Cheshire, A. Malleson and P. Naccache [1982], "A Dual Heuristic for Vehicle Scheduling," *Journal of Operational Research Society*, *Vol.* 33, pp. 51 - 61.

G. Clarke and J. W. Wright [1964], "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, *Vol.* 12, pp. 568.

W. Clavey [1987], *Development of Distribution System for Navistar Unites Routing/ Scheduling with Production/Inventory Control*, presented at the ORSA/TIMS Meeting, St. Louis.

H. Crowder and M. Padberg [1980], "Solving Large-Scale Symetric Traveling Salesman Problems to Optimality," *Management Science*, *Vol.* 26, pp. 495 - 509.

F. Cullen, J. Jarvis and D. Ratliff [1981], "Set Partioning-Based Heuristics for Interactive Routing," *Networks*, *Vol.* 11, pp. 125 - 144.

G. B. Dantzig and K. H. Ramser [1959], "The Truck Dispatching Problem," *Operations Research*, *Vol.* 12, pp. 80.

M. Desrochers, J. K. Lenstra, M. W. P. Savelsbergh and F. Soumois [1988], "Vehicle Routing with Time Windows : Optimization and Approximation," in *Vehicle Routing : Methods and Studies*, edited by B. L. Golden and A. A. Assad, North-Holland, Amsterdam, pp. 65 - 84.

M. Dror and M. Ball [1987], "Inventory/Routing : Reduction from an Annual to a Short-Period Problem," *Naval Research Logistics*, *Vol.* 34, pp. 891 - 905.

J. Edmonds [1971], "Matroids and the Greedy Algorithm," *Mathematical Programming*, *Vol.* 1, pp. 127 - 136.

J. Edmonds and E. L. Johnson [1973], "Matching, Euler Tours and the Chinese Postman," *Mathematical Programming, Vol.* 5, pp. 88 - 124.

J. Edmonds and R. M. Karp [1972], "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. Assoc. Comput. Math., Vol.* 19, pp. 248 - 264.

S. Eilon, C. Watson-Gandy and N. Christofides [1971], *Distribution Management : Mathematical Modeling and Practical Analysis*, Griffin, London.

M. Fisher [1978], *Lagrangean Relaxation Methods for Combinatorial Optimization*, Paper Presented at Summer School in Combinatorial Optimization, Urbino, Italy.

M. Fisher and R. Jaikumar [1978], *A Decomposition Algorithm for Large-Scale Vehicle Routing*, Report 78-11-05, The Wharton School, University of Pennsylvania, Philadelphia.

M. Fisher and R. Jaikumar [1981], "A Generalized Assignment Heuristic for Vehicle Routing," *Networks, Vol.* 11, pp. 109 - 124.

M. Fisher , R. Jaikumar and L. N. Van Wassenhove [1979], *A Multiplier Adjustment Method for the Genelized Assignment Problem*, Report 81-07-06, Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia.

B. Foster and D. Ryan [1976], "An Integer Programming Approach to the Vehicle Scheduling Problem," *Operations Research Quartly, Vol.* 27, pp. 367 - 384.

G. N. Frederickson [1979], "Approximation Algorithms for Some Postman Problems," *J. Assoc. Comput. Math., Vol.* 26, pp. 538 - 554.

G. N. Frederickson, M. S. Hecht and C. E. Kim [1978], "Approximation Algorithms for Some Routing Problems," *SIAM J. Comput., Vol.* 7, pp. 178 - 193.

R. Garfinkel and G. Nemhauser [1970], *Integer Programming*, Wiley, New York.

T. J. Gaskell [1967], "Bases for Vehicle Fleet Scheduling," *Operational Research Quartly, Vol.* 18, pp. 281.

B. Gavish and E. Shlifer [1979], "An Approach for Solving a Class of Transportation Scheduling Problems," *European Journal of Operations Research, Vol.* 3, pp. 122 - 134.

B. Gavish and K. N. Srikanth [1979], *Mathematical Formulations of the Dial-a-Ride Problem*, Working paper 7909, Graduate School of Management, University of Rochester, NY.

A. Geoffrion [1974], "Lagrangean Relaxation and it's Uses in Integer Programming," *Mathematical Programming Study, Vol.* 2, pp. 82 - 114.

F. Gheysens, B. L. Golden and A. A. Assad [1982], *A Relaxation Heuristic for the Fleet Size and Mix Vehicle Routing Problem*, Working paper MS/S 82-029, University of Maryland, College Park.

B. E. Gillet [1976], *Vehicle Dispatching : Sweep Algorithm and Extentions*, ORSA-TIMS, National Meeting.

B. E. Gillet and T. Johnson [1976], "Multi-Terminal Vehicle-Dispatch Algorithm," *Omega, Vol.* 4, pp. 711 - 718.

B. E. Gillet and L. R. Miller [1974], "A Heuristic Algorithm for the Vehicle Dispatch Priblem," *Operations Research, Vol.* 22, pp. 340.

B. L. Golden [1975], *Vehicle Routing Problems : Formulations and Heuristic Solution Techniques*, ORC Technical Report, Vol. 22, MIT.

B. L. Golden [1976], *Recent Developments in Vehicle Routing*, Presented at the Bicentennial Conference, Mathematical Programming, Gaithersberg, MD, Nobember, 1976.

B. L. Golden [1977], "Evaluating a Sequential Vehicle Routing Algorithm," *AIIE Transportation, Vol.* 9, pp. 204 - 208.

B. L. Golden [1978], "Recent Developments in Vehicle Routing," in *Computers and Mathematical Programming*, W. White Ed., National Bureau of Standards Special Publication 502, Washington DC, pp.243 - 240.

B. L. Golden and A. A. Assad [1986 a], "Perspectives on Vehicle Routing : Exciting New Developments," *Operations Research, Vol.* 34, pp. 803 - 810.

B. L. Golden A. A. Assad [1986 b], "Vehicle Routing with Time-Window Constraints," *American ournal of Mathematical and Management Sciences, Vol.* 6, pp. 251 - 260.

B. L. Golden, A. A. Assad, L. Levy and F. Gheysens [1982], *The Fleet Size and Mix Vehicle Routing Problem*, Working paper MS/S 82-020, University of Maryland, College Park.

B. L. Golden, L. Bodin, T. Doyle and W. Stewart [1980], "Approximate Traveling Salesman Algorithms," *Operations Research, Vol.* 28, pp. 694 - 711.

B. L. Golden, T. Magnanti and H. Nguyen [1977], "Implementing Vehicle Routing Algorithms," *Networks, Vol.* 7, pp. 113 - 118.

M. R. Gary, R. L. Graham and D. S. Johnson [1976], *Some NP-Complete Geometric Problems*, Proc. 8th Ann. ACM Symp. Theory of Computing, pp. 10 - 22.

R. M. Gary and D. S. Johnson [1978], "Strong NP-Completeness Results : Motivation, Examples and Implications," *Journal of Assoc. Comput. Math., Vol.* 25, pp. 499 - 508.

M. R. Gary and D. S. Johnson [1979], *Computers and Interactability : A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

M. Haimovich and A. H. G. Rinnooy Kan [1985], "Bounds and Heuristics for Capacitated Routing Problems," *Mathematical Operations Research, Vol.* 10, pp. 527 - 542.

M. Haimovich, A. H. G. Rinnoy Kan and L. Stougie [1988], "Analysis of Heuristics for Vehicle Routing Problems," in *Vehicle Routing : Methods and Studies*, edited by B. L. Golden and A. A. Assad, North-holland, Amsterdam, pp. 47 - 61.

R. Hays [1967], *The Delivery Problem*, Management Science Research Report, No. 106, Carnegie Institution of Technology.

M. Held and R. M. Karp [1970], "The Traveling Salesman Problem and Minimum Spanning Trees : Part I," *Operations Research, Vol.* 18, pp. 1138.

M. Held and R. M. Karp [1971], "The Traveling Salesman Problem and Minimum Spanning Trees : Part II," *Mathematical Programming, Vol.* 1, pp. 6 - 25.

M. Held, R. M. Karp and H. P. Crowder [1974], "Validation of Subgradient Optimization," *Mathematical Programming, Vol.* 6, pp. 62 - 88.

J. Hinson and S. Mulherkar [1975], *Improvements to the Clark and Wright Algorithm as Applied to an Airline Scheduling Problem*, Technical Report, Federal Express Corp.

D. Houck, J. C. Picard, M. Queyranne and R. R. Vemuganti [1977], *Traveling Salesman Problem and Shortest n-path*, Report, University of Maryland, Clooege Park.

IBM Corporation [1970], *Vehicle Sceduling Program Application - VSPX*, Report GH 19-2000/0, White Plains, NY.

P. Jaillet [1986], "Stochastic Routing Problem," in *Proceedings of CISM Advancd School on Stochastics and Optimization*, G. Andeatta and P. Serafini Eds., Springer-Verlag, Berlin.

P. Jaillet and A. R. Odoni [1988], "The Probabilistic Vehicle Routing Problem," in *Vehicle Routing : Methods and Studies*, edited by B. L. Golden and A. A. Assad, North-Holland, Amsterdam, pp. 293 - 318.

J. Jaw, A. Odoni, H. Psaraftis and N. Wilson [1986], "A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-a-Ride Problem with Time-Window," *Transportation Research, Vol. 20B*, pp. 243 - 257.

A. Jezequel [1986], *Probabilistic Vehicle Routing Problem (S. M. hesis)*, Department of Civil Engineering, Massachusetts Institute of Technology, Cambrige, MA 02139 (Unpublished).

R. M. Karp [1972], "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thather, Eds., Plenum, NY, pp. 85 - 103.

R. M. Karp [1975], "On the Complexity of Combinatorial Problems," *Networks, Vol.* 5, pp. 45 - 68.

R. M. Karp [1977], "Probabilistic Analysis of Partitioning Algorithms for the Traveling Salesman Problem in the Plane," *Mathematical Operations Research, Vol.* 2, pp. 209 - 224.

K. Knowles [1967], *The Use of Heuristic Tree-Search Algorithm for Vehicle Routing and Scheduling*, Presented at the OR Conference, Exter, England.

A. Kolen, A. H. G. Rinnooy Kan and H. Trienekens [1987], "Vehicle Routing with Time Windows," *Operations Research, Vol.* 35, pp. 266 - 273.

P. Krolak, W. Felts and G. Marble [1971], "A Man-Machine Approach Toward Solving the Traveling Salesman Problem," *Comm. ACM, Vol.* 14, pp. 327 - 334.

P. Krolak, W. Felts and J. Nelson [1972], "A Man-Machine Approach Toward Solving the Generalized Truck Dispatching Problem," *Transpotation Sciences*, *Vol.* 6, pp. 149 - 169.

P. Krolak and J. Nelson [1978], *A Family of Truck Load Clustering Heuristics for Vehicle Routig Problems*, Technical Report 78-2, Department of Computer Science, Vanderbilt University, Nashville, TN.

A. Langevin and F. Soumis [1989], "Design of Multiple-Vehicle Delivery Tours Satisfying Time Constraints," *Transportation Research*, *B.* (in Press).

G. Laporte and Y. Nobert [1987], "Exact Algoritms for the Vehicle Routing Problem," in *Surveys in Combinatorial Optimization*, S. Martello *et al.* Eds., North-holland, Amsterdam, pp. 147 - 184.

G. Laporte, Y. Nobert and S. Taillefer [1988], "Solving A Family of Multi-Depot Vehicle Routing and Location-Routing Problems," *Transportation Sciences*, *Vol.* 22, pp. 161 - 172.

E. Lawler, J. Lenstra, A. Rinnooy Kan and D. Shmoys [1985], *Traveling Salesman Problem*, Wiley, Chichester.

J. K. Lenstra and A. H. G. Rinnooy Kan [1976], "On General Routing Problems," *Networks*, *Vol.* 6, pp. 273 - 280.

J. K. Lenstra and A. H. G. Rinnooy Kan [1979], "Computational Complexity of Discrete Optimization Problems," *Ann. Discrete Math.*, *Vol.* 4, pp. 121 - 140.

J. K. Lenstra and A. H. G. Rinnooy Kan [1981], "Complexity of the Vehicle Routing and Scheduling Problems," *Networks*, *Vol.* 11, pp. 221 - 227.

S. Lin [1965], "Computer Solutions of the Traveling Salesman Problem," *Bell Syst. Tech. J.*, *Vol.* 44, pp. 2245 - 2269.

S. Lin and B. Kernighan [1973], "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, *Vol.* 21, pp. 498 - 516.

A. P. Lucena Filho [1986], "Exact Solution Approaches for the Vehicle Routing Problems," *Ph.D. Thesis, Dept. Mgmt. Sci., Imperial College.*

R. E. Marsten [1974], "An Algorithm for Large Set Partioning Problems," *Management Science*, *Vol.* 20, pp. 774 - 787.

C. Miller, A. W. Tucker and R. A. Zemlin [1960], "Integer Programming Formulation of the Traveling Salesman Problem," *Journal of the Association for Computing Machinary*, *Vol.* 7.

R. Mole and S. Jameson [1976], "A Sequential Route-Building Algorithm Employing a Generalized Savings Criterion," *Operations Research Quartly*, *Vol.* 27, pp. 503 - 511.

R. Mole, D. Johnson and K. Wells [1983], "Combinatorial Analysis of Route First-Cluster Second Vehicle Routing," *Omega*, *Vol.* 11, pp. 507 - 512.

T. S. Motzkin and I. J. Schoenberg [1954], "The Relaxation Method for Linear Inequalities," *Canadian Journal of Mathematics*, *Vol.* 6, pp. 393 - 404.

R. Newton and W. Thomas [1974], "Bus Routing in a Multi-School System," *Comput. Operations Research, Vol.* 1, pp. 213 - 222.

I. Or [1983], *A Heuristic Solution Procedure for the Inventory Routing Problem*, Working paper MS/S 83-029, University of Maryland, College Park.

C. Orloff [1976], "Route Constrained Fleet Scheduling," *Transportation Sciences, Vol.* 10, pp. 149 - 168.

C. H. Padadimitriou [1976], "On the Complexity of Edge Traversing," *Journal of Assoc. Comput. Math., Vol.* 23, pp. 544 - 554.

C. H. Padadimitriou [1977], "The Euclidean Traveling Salesman Problem Is NP-Complete," *Theor, Comput. Sci., Vol.* 4, pp. 237 - 244.

C. H. Padadimitriou and K. Steiglitz [1982], *Combinatorial Optimization : Algorithms and Complexity*, North-Holland, Amsterdam.

J. F. Pierce [1970], *A Two Stage Approach to the Solution of Vehicle Dispatching Problems*, Presented at 17th TIMS International Conference, London.

B. T. Poljak [1967], "A Genaral Method of Solving Extremum Problems," *Dokl. Akad. Nauk., SSSR* 174, pp. 33 - 36, Translation, *Soviet Mathematics, Dokl., Vol.* 8, pp. 593 - 597, (1967)

O. M. Raft [1982], "A Modular Algorithm for An Extended Vehicle Scheduling Problem," *European Journal of Operations Research, Vol.* 11, pp. 67 - 76.

R. Russell [1974], *Efficient Truck Routing for International Refuse Collection*, Presented at ORSA/TIMS Meeting, Puerto Rico.

R. Russell [1977], "An Effective Heuristic for M-Tour Traveling Salesman Problem with Some Side Conditions," *Operations Research, Vol.* 25, pp. 517 - 524.

R. Russell and W. Igo [1979], "An Assignment Routing Problem," *Networks, Vol.* 9, pp. 1 - 17.

C. Sandi [1979], "Subgradient Optimization," in *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth and C. Sandi, Eds., Wiely, Chichester, PP. 73 - 91.

L. Schrage [1981], "Formulation and Structure of More Complex/Realistic Routing and Scheduling Problems," *Networks, Vol.* 11, pp. 229 - 232.

J. F. Shapiro [1979], "A Survey of Lagrangean Techniques for Discrete Optimization," *Annals of Discrete Mathematics, Vol.* 5 : *Discrete Optimization II*, pp. 113 - 138.

D. K. Smith [1982], *Network Optimization Practice*, Ellis Horwood, Chichester.

M. M. Solomon [1987], "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Operations Research, Vol.* 35, pp. 254 - 305.

M. M. Solomon, E. K. Baker and J. R. Schaffer [1988], "Vehicle Routing and Scheduling Problems with Time Window Constraints," in *Vehicle Routing : Methods and Studies*, edited by B. L. Golden and A. A. Assad, North-Holland, Amsterdam, pp. 85 - 105.

H. Stern and M. Dror [1979], "Routing Electric Meter Readers," *Comput. Operations Research*, *Vol.* 6, pp. 209 - 223.

W. Stewart and B. Golden [1979], "A Vehicle Routing Algorithm Based on Generalized Lagrangean Multipliers," *Proceedings of the AIDS* 1979 *Annual Convention*, L. Moore, K. Monroe and B. Taylor Eds., New Orleans, LA, Vol. 2, pp. 108 - 110.

W. Stewart, B. Golden and F. Gheysens [1982], *A Survey of Stochastic Vehicle Routing*, Working paper MS/S 82-027, University of Maryland, College Park.

W. Wtewart and B. Golden [1983], "Stochastic Vehicle Routing : A Comprehensive Approach," *European Journal of Operations Research*, *Vol.* 14, *no.* 3, pp. 371 - 385.

F. A. Tillman and H. Cochran [1969], "A Heuristic Approach for Solving the Delivery Problem," *Journal of Industrial Engineering*, *Vol.* 19, pp. 354.

H. Trinekens [1982], *The Time Constrained Vehicle Routing Problem*, Unpublished report, Erasmus University, Rotterdam.

P. Van Leeuwen and A. Volgenant [1983], "Solving Symmetric Vehicle Routing Problems Asymmetrically," *European Journal of Operations Research*, *Vol.* pp. 388 - 393.

C. D. J. Waters and G. P. Brodie [1987], "Realistic Sizes for Routing Problems," *Journal of Operationa Research Society*, *Vol.* 38, pp. 565 - 566.

E. Wilhelm [1975], *Overview of the RUCUS Package Driver Run Cutting Program* (*RUNS*), in Bergman and Bodin.

B. Williams [1982], "Vehicle Scheduling : Proximity Priority Searching," *Journal of Operational Research Society*, *Vol.* 33, pp. 961 - 966.

P. Yellow [1970], "A Computational Modification to the Savings Method of Vehicle Scheduling," *Operations Research Quartly*, *Vol.* 21, pp. 281 - 283.