DEVELOPMENT OF SOME LOCAL SEARCH METHODS FOR SOLVING THE VEHICLE ROUTING PROBLEM

ZENG LING

NATIONAL UNIVERSITY OF SINGAPORE
2003

DEVELOPMENT OF SOME LOCAL SEARCH METHODS FOR SOLVING THE VEHICLE ROUTING PROBLEM

ZENG LING

(*B.ENG, DUT*)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF INDUSTRIAL & SYSTEMS ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE
2003

Acknowledgements

First of all, I would like to express my sincere gratitude and appreciation to my supervisor, Associate Professor Ong Hoon Liong, for his painstaking supervision of my research work; and for his invaluable suggestions, support, guidance, and patience throughout this entire research project. His enthusiasm towards research work and his kind personality will always be remembered.

Appreciation must also be accorded to my fellow research students who have given me much encouragement and guidance, and thus facilitated the completion of this thesis.

Last but not least, I would like to extend my sincere gratitude to my family and my husband, for their kind understanding and warm support throughout the course of my research.

Zeng Ling

Table of Contents

Summary

The vehicle routing problem (VRP) is an important class of combinatorial problems. Its economic importance is marked by its presence in many areas of the manufacturing and service industries. The VRP is NP-hard, and therefore, it is unlikely to be solved by a polynomially optimal algorithm. The objective of this thesis is to develop some efficient heuristics for solving the VRP.

In this study, a local search method, called the assignment-based local search (ABLS) method is proposed to solve the capacitated VRP (CVRP) and some of its variants. The ABLS algorithm is a multi-route improvement algorithm that can operate on several routes at a time. In ABLS algorithm, the inserting of nodes into routes at each step is based on the solution of an assignment problem. Several types of local search methods and strategies that can be incorporated into the ABLS procedures are presented and some composite procedures consisting of the ABLS and other heuristics, such as search space smoothing and simulated annealing, are proposed in this study. To evaluate the performance of the proposed methods, extensive computational experiments on the various proposed algorithms applied to a set of benchmark problems are carried out. The results show that the proposed methods, especially the composite procedures, are able to generate some good solutions to the problems tested compared with other efficient heuristics proposed in the literature.

Another proposed method, generalized crossing (GC) method, is also introduced to solve the VRPs. The algorithm proposed in this study is an extension of the normal string crossing method. In this method, more combination of the strings and the order of each string are considered. That is, the new routes are constructed not only by combining the strings in their original direction but also combining the strings with opposite direction in the GC method. Computational results show that its SA implementation combined with a new improvement procedure, middle improvement procedure, outperforms other SA implementations and is comparable with some other meta-heuristic implementations reported in the literature.

To illustrate the effectiveness of the two proposed methods, an application of the two methods to a real-world soft drinks distribution problem is carried out in this study. The objective function of this problem is to minimize the total number of vehicles used. In the application of this study, a bin packing composite procedure is applied to solve a number of problem instances obtained from a soft drinks distribution company. The computational results show that better solutions can be obtained for the proposed methods than other approaches proposed in the literature. For some problem instances tested, the improvement can be more than 40%.

Nomenclature

d_i	Demand for customer i
g_{ij}	The least cost increment when inserting node i to route j
l	Number of fully loaded vehicles when serving the customer whose demand is greater than vehicle capacity
m	Number of vehicles available
M	A very big positive value
n	Number of customers
Q	Capacity for each vehicle
R	Number of all feasible routes
s_{ij}	Cost increment when node i is inserted into edge j in a TSP tour
T	Current temperature for simulated annealing algorithm
T_0	Initial temperature for simulated annealing algorithm
T_f	Final temperature for simulated annealing algorithm
T_r	Remaining time for vehicle r
W_i	Weight of item i of a bin packing problem
$\lfloor x \rfloor$	The greatest integer smaller than or equal to x
$\lceil x \rceil$	The smallest integer greater than or equal to x
VRP	Vehicle routing problem
TSP	Traveling salesman problem
ABLS	Assignment-based local search
GC	Generalized crossing method
ABLS&SA	ABLS and simulated annealing composite method
GC&SA	GC and simulated annealing composite method

ABLS&BP	ABLS and bin packing composite method
GC&BP	GC and bin packing composite method

List of Figures

List of Tables

Chapter 1

Introduction

This thesis focuses on the design and analysis of algorithms for solving the vehicle routing problem (VRP). Two new local search methods, the assignment-based local search (ABLS) algorithm and generalized crossing (GC) method, are proposed and an extensive evaluation and comparison of the two proposed methods with some other algorithms proposed in the literature are conducted. This chapter presents some background information and basic knowledge of the VRP. It concludes by presenting the purpose of this thesis as well as its organization.

1.1 Background

The traveling salesman problem (TSP) can be defined on a complete directed or undirected graph G=(V, A), where V is a set of n vertices and A is a set of arcs or edges. C=(c_{ij}) is a distance (or cost) matrix associated with A. Its objective is to determine a minimum distance circuit passing through each vertex once and only once (Christofides *et al*., 1979 and Laporte, 1992).

The vehicle routing problem (VRP) is a generalization of the TSP. Its purpose is to determine a set of routes with minimum total costs on the condition that the following criteria are satisfied:

- The route starts and ends at a depot;
- The total demand of any route does not exceed the vehicle capacity, Q;
- The total length of any route does not exceed a preset bound L.

In most practical cases, it is only necessary to consider the cases whose cost function is a metric, i.e., it satisfies: $c_{ij} \geq 0$, $c_{ij} = c_{ji}$ and the triangle inequality, $c_{ik} + c_{kj} \geq c_{ij}, \quad i, j, k \in V$.

The VRP was introduced by Dantzig and Ramser in 1959. In their paper, a real-world application concerning the delivery of gasoline to service stations was described and then a mathematical programming model was formulated. They were followed by Clarke and Wright who proposed an effective heuristic to improve the Dantzig-Ramser approach in 1964. After that, hundreds of models and algorithms were proposed to solve many types of VRPs and many successful applications were reported in the literature.

The last 40 years has seen rapid progress made in many aspects of the VRP, such as theory, practice, computer hardware and software, so that the VRP has become one of the greatest success stories of operations research. The successful implementation of vehicle routing software has been aided by the rapid developments in computer science, such as, the development of the geographic information system and interface software which has enabled customers to integrate routing with other key functions such as inventory tracking, forecasting, and so on. Moreover, vehicle routing software can be integrated directly with enterprise resource planning (ERP). It can be seen, for example, that the routing software of one company can interface with the sales and distribution module of Systems Analysis and Program Development (SAP)'s transportation planning system to access information on orders, carriers, geography and transportation requirements (Baker, 2002).

Many applications of the VRP have been proposed in the literature on operations research in recent years. The applications span a wide variety of industries and involve the commercial distribution of many products that range from newspapers (Picard and

Brody, 1997) to soft drinks (Cheong *et al*. 2002); and from groceries (Carter *et al*., 1996) to milk (Basnet *et al*., 1996), on a daily basis. In addition, the applications also involve waste collection, street sweeping and delivery of mail. Most of these applications not only possess the characteristics of the basic VRP model, such as vehicle route and route duration, but also involve many complicated issues, such as time windows and periodic or multi-deliveries to customers.

1.2 Characteristics of the VRP

The typical characteristics (Toth and Vigo, 2002*b*) of customers in the VRP include:

- The vertex of the graph which denotes where the customer is located;
- The amount of goods (demand) which must be delivered or collected;
- The periods of the time (time windows) during which the customer can be served;
- The times needed to serve customers (loading and unloading times); and
- The subset of the available vehicles that can be used to serve the customer because of the possible access limitations or some other requirements.

The typical characteristics of vehicles include:

- The subset of arcs on the graph which can be traversed by the vehicle;
- The costs associated with the utilization of the vehicle;
- The capacity of the vehicle, expressed as the maximum weight or volume the vehicle can load;
- The devices available for the loading and unloading operations; and
- The possible subdivision of the vehicle into compartments, each characterized by its capacity and by the types of goods that can be carried.

Moreover, all routes must satisfy some operational constraints, such as the nature of the transported goods and the quality of service. Drivers must also satisfy the constraints imposed by the company and customers, such as the maximum working time, the number and duration of breaks during the service and the maximum duration of driving periods, and so on.

The typical objectives of the VRP are:

- To minimize the total transportation costs which are dependent on the distance traveled and the fixed costs associated with the vehicles used and the corresponding number of drivers;
- To minimize the number of vehicles used to serve the customers;
- To balance the routes with respect to the travel time and the vehicle load;
- To minimize the penalties; or
- Any combination of the above objectives.

There has been a steady evolution in the design of solution methodologies, resulting in exact and approximate methods for the VRP, since it was introduced by Dantzig and Ramser in 1959. Several researchers (Golden *et al.*, 1998 and Naddef and Rinaldi, 2002) have noted that no known exact algorithm is capable of consistently finding optimal solutions for the problems with more than 50 customers. Hence, in practice, heuristics are used in most cases.

1.3 Basic Types of VRPs

The VRP can be classified into the capacitated VRP (CVRP), the distance-constrained and capacitated VRP (DCVRP), the VRP with time windows (VRPTW), the VRP with backhauls (VRPB), the VRP with pickup and delivery (VRPPD), and some combination cases, such as the VRPB with time windows (VRPBTW) and VRPPD with time windows (VRPPDTW). A summary of these problems and the relation between them is shown in Figure 1.1.

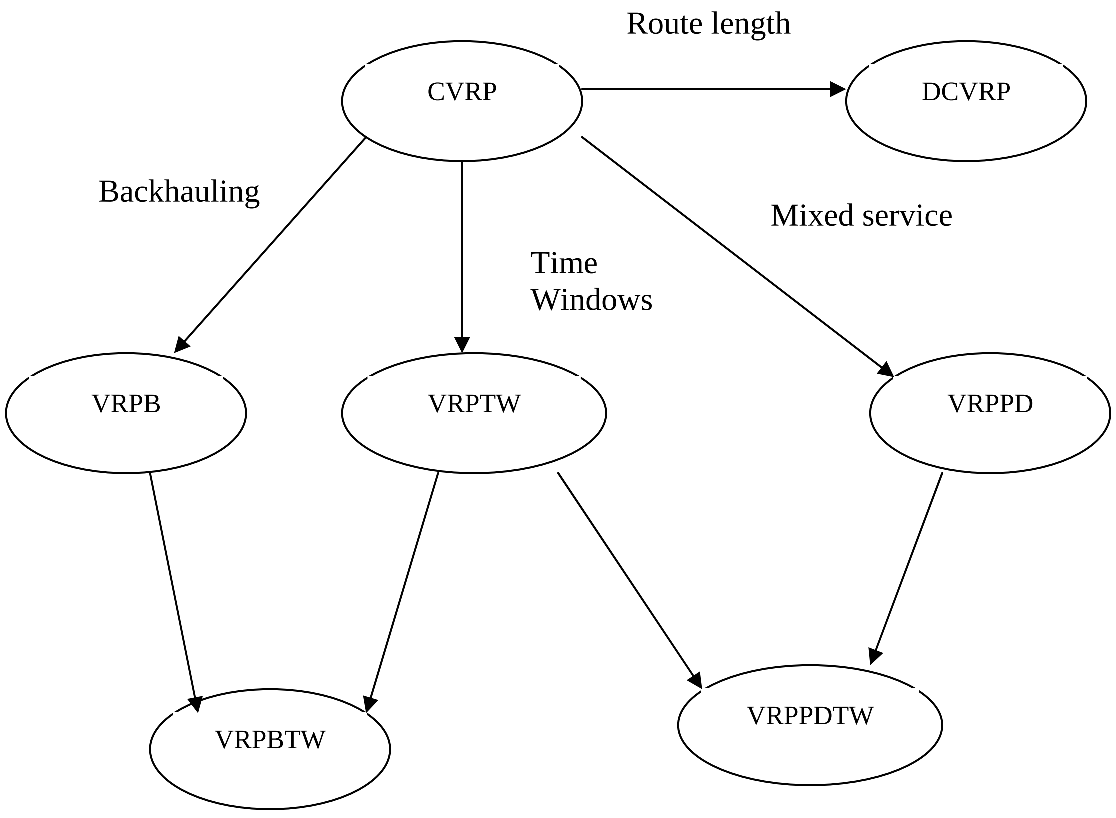


Figure 1.1 Basic types of VRPs and their relations

1.3.1 Capacitated VRP and Distance-Constrained VRP

The capacitated VRP is the simplest and most studied member of the family of VRPs. In this problem, all the customers and demands are deterministic, i.e. known in advance and may not be split. For the CVRP, it is assumed that there is only one depot, all the vehicles are identical and the only constraint imposed is the capacity constraint. The objective of this problem is to design a set of vehicle routes at minimum total costs with all routes starting from and ending at the depot, such that each customer is

)

visited once and the total capacity for each route does not exceed the vehicle’s capacity, *Q*.

The minimum number of vehicles needed may be determined by solving the bin packing problem (BPP). In the BPP, the objective is to determine the minimum number of bins, each with identical capacity, *Q*, to load all items with nonnegative weights. In the implementation of the BPP to the CVRP, vehicles are bins, customers are items and the demands of customers are the weights of the items. Although the BPP is a NP-hard problem (Martello and Toth, 1990), instances involving hundreds of items can be solved to optimality very effectively.

The CVRP is known to be NP-hard (Achuthan *et al*., 1998). Normally, the CVRP can be classified into two categories: the symmetric and the asymmetric CVRP. If costs are symmetric, it is known as a symmetric CVRP (SCVRP); otherwise, it is known as an asymmetric or directed CVRP (ACVRP).

An extensive survey on the exact methods of the VRP was conducted by Laporte and Nobert in 1992. Many other researchers, such as Christofides *et al*. (1979), Bodin *et al*. (1983), and Christofides (1985), have formulated numerous heuristic methods to solve the CVRP.

One of the variants of the CVRP is the distance-constrained VRP (DVRP), where for each route, the capacity constraint is replaced by a maximum route length (or time) constraint. In the case where the vehicle capacity and the maximum distance constraints are present, it is called the distance-constrained CVRP.

1.3.2 VRP with Time Windows

The VRP with time windows is an extension of the CVRP in which capacity constraints are imposed and each customer *i* is associated with a service time interval

[a_i, b_i], called time window. In this problem, the customers have to be served by a fleet of vehicles initially located at the depot. Each customer has a load that must be picked up, and the customer specifies a period of time, called the time window, in which this pick up must occur. The objective is to find a set of routes for the vehicles to serve a set of customers without violating the capacity and time window constraints, while minimizing the total distance traveled by the vehicles (Bramel and Simchi-Levi, 1996; Cordeau *et al*., 2001).

In the VRPTW, soft time windows can be violated at a cost, while hard time windows do not allow a visit to the customer outside the desired time windows. VRPTW is NP-hard, and even finding a feasible solution to the VRPTW with a fixed fleet size is itself an NP-complete problem (Savelsbergh, 1985).

1.3.3 VRP with Backhauls

The VRP with backhauls is another extension of the VRP. In this problem, the customer set is partitioned into two subsets: the first subset, *L*, contains *p* linehaul customers, each requiring a given number of products to be delivered; while the second part, *B*, contains *q* backhaul customers, from whom a certain number of inbound goods must be picked up. This problem is frequently encountered in practice. In the grocery industry, for example, the supermarkets and shops are the linehaul customers and the grocery suppliers are the backhaul customers.

In the VRPB, a precedence constraint between linehaul and backhaul exists, i.e., when the route needs to serve the two types of customers, it must first serve all the linehaul customers before any backhaul customer may be served.

The VRPB can be classified into the symmetric and asymmetric categories. In the symmetric VRPBs, the distance between each pair of locations is the same in the two directions, while in the asymmetric VRPB (AVRPB) the symmetric assumption does not hold.

The objective of the VRPB is to find the minimum costs for a collection of vehicle routes such that each route visits the depot and each customer is served exactly once, provided that the sum of the demands of the linehaul and backhaul customers visited by one route does not exceed the vehicle capacity, and that the linehaul customers precede the backhaul customers.

Let K_L and K_B denote the minimum number of vehicles needed to serve all the linehaul and backhaul customers, respectively. K_L and K_B can be calculated by solving the bin packing problem. The number of vehicles needed to serve all the customers cannot be smaller than the maximum number between K_L and $K_{B.}$

The VRPB and AVRPB are both NP-hard since they generalize the basic version of SCVRP and ACVRP when the subsets of backhaul customers are empty (Toth and Vigo, 1999).

1.3.4 VRP with Pickup and Delivery

In the VRP with pickup and delivery, each customer i is associated with two quantities d_i and p_i, representing the demands of homogeneous commodities to be delivered and picked up, respectively, at customer i. In this problem, it is assumed that the delivery is performed before the pickup for each customer location, and therefore, the current load of the vehicle before arriving at a given location can be calculated by the initial load minus all the demands already delivered plus all the demands already picked up.

The objective of the VRPPD is to minimize the total traveling costs without violating the capacity constraints and precedence constraints between pick up and delivery for each customer location.

In the VRPPD, a heterogeneous vehicle fleet based on multiple terminals must satisfy a set of transportation requirements. Each requirement is defined by a pickup point, a corresponding delivery point, and a demand to be transported between these locations. The required transport could involve goods and persons. The latter case is called dial-a-ride. It was first investigated by Wilson *et al*. in 1971.

The VRPPDTW is a generalization of the VRPTW and has a variety of applications, including the sealift and airlift of cargo and troops. Many researchers, such as Solomon and Desrosiers (1988) as well as Savelsbergh and Sol (1995), have highlighted the perspectives of this growing field. It is noted that both the VRPPD and VRPPDTW are NP-hard.

1.4 Purpose of this Thesis

The purpose of this study is to develop new local search methods to solve the VRP. It is well known that the VRP is NP-hard. Therefore, it is unlikely that a polynomially-bound optimal algorithm for solving the VRP exists. As a result, many researchers have focused on developing heuristics to solve the VRP. Among the well known algorithms are the savings method of Clarke and Wright (1964); the generalized assignment problem algorithm of Fisher and Jaikumar (1981); the sweep algorithm of Gillett and Miller (1974); Lin's (1965) λ-opt mechanism; the sequential insertion of the Mole and Jameson (1976) heuristics, and so on. Most of these heuristics can result in relatively good solutions within a reasonable amount of computational time.

This thesis is concerned with the development of heuristics for solving the CVRP. In particular, two local search methods, the ABLS algorithm and the GC method, are proposed to solve this problem. To evaluate the performance of the proposed algorithms, computational experiments are carried out to compare the algorithms against several algorithms described in the literature. The results of the comparisons demonstrate that the proposed algorithms, especially GC method, are able to generate good solutions to the problems tested. It can be matched with the results obtained by other algorithms reported in the literature.

To illustrate the effectiveness of the proposed algorithms, the two proposed algorithms are applied to a real-world soft drinks distribution problem. Computational results show that these algorithms are able to provide better solutions than the existing method.

1.5 Organization of this Thesis

This thesis focuses on the design and analysis of heuristics for solving the VRP.

In Chapter 2, a literature survey of the methods used to solve the different varieties of VRPs is presented.

In Chapter 3, the first proposed ABLS algorithm, strategies that can be incorporated in the ABLS procedure, and some composite procedures consisting of the ABLS and other heuristics are described in detail. In addition, computational results and analysis are also proposed and presented.

The second GC method is presented in Chapter 4. A new improvement procedure, middle improvement procedure, is presented. This is followed by a more thorough analysis of computational results and comparison with other heuristic methods.

An application of the two proposed algorithm to a real-world soft drinks distribution problem is presented in Chapter 5. In this problem, the objective is to minimize the total number of vehicles used. A bin packing composite procedure is applied to solve a number of problem instances obtained from the soft drinks distribution company. Computational results show that this composite algorithm can improve the existing approaches effectively. It can be seen that for some of the problem instances tested, the improvement can be more than 40%.

Finally, in Chapter 6, some concluding remarks and suggestions for future research work are provided.

Chapter 2

Literature Survey

The vehicle routing problem (VRP) is an important type of combinatorial problem and has been the focus of operations researchers and combinatorial analysts for many years. Many exact and approximate methods have been proposed to solve the VRP in recent years. In this chapter, a review of the various methods proposed for solving the TSP and the VRP in the literature is provided.

2.1 Approaches for Solving the TSP

The TSP is a VRP in its simplest form. To date, it remains one of the most challenging combinatorial optimization problems. The problem’s statement is simple and the objective is to determine a minimal cost cycle that passes through each node or customer location exactly once. It can be classified into two broad categories, i.e., the symmetric TSP and the asymmetric TSP, depending on whether the costs between two locations are dependent on the direction of travel or not. A symmetric TSP is one where the traveling cost does not depend on the direction of the travel. Otherwise, it is defined as an asymmetric TSP.

Hundreds of articles have been published to solve the TSP. A comprehensive survey of the TSP can be found in Bodin *et al.* (1983), Laporte (1992) as well as Johnson and Mcgeoch (1997). Some exact and heuristic algorithms are reviewed in the following section.

2.1.1 Exact Methods for the TSP

Many exact algorithms have been proposed in the literature to solve the TSP. In this section, some commonly used methods, such as the integer linear programming formulations and the branch and bound methods, are reviewed.

Integer linear programming formulations

Dantzig et al. presented one of the earliest formulations of TSP in 1954. In their formulation, $c_{ii} = +\infty$ for $i = 1, 2, \ldots n$, where *n* denotes the number of vertices.

The integer linear programming problem can be formulated as follows:

$$\text{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1,2,\ldots,n\,, \tag{2.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1,2,\ldots,n\,, \tag{2.3}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - 1, \quad 2 \le |S| \le n-2\,, \tag{2.4}$$

$$x_{ij} = \begin{cases} 1 & \text{if salesman travels directly from node } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}. \tag{2.5}$$

In the above formulation, *S* is a subset of the set of *n* vertices, i.e., $S \subset V$. Constraints (2.2) and (2.3) specify that every vertex is entered exactly once and left exactly once. Constraint (2.4) is a subtour elimination constraint, i.e., it can prohibit the formation of a subtour. If there is a subtour on a subset *S*, it should contains |*S*| arcs. Then constraint

(2.4) is violated because |*S*| is not less than or equal to |*S*| -1. Constraint (2.4) has some alternative equivalent forms, such as:

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1, \quad 2 \leq |S| \leq n-2 \qquad (2.4')$$

where $\bar{S} = V \setminus S$. Constraint (2.4') can be derived from (2.4).

Several alternative formulations, such as that of Miller *et al.* (1960), have been proposed and compared since the first formulation proposed by Dantzig *et al.* in 1954. However, Langevin *et al.* (1990) showed that none of these alternative formulations had a stronger linear relaxation than the method formulated by Dantzig *et al.* (1954).

The above formulations can provide insights into the complexity of the TSP and its relationship to other routing problems. Moreover, they can also suggest some algorithms that are obtained by dualizing with respect to certain constraints (Bodin *et al.*, 1983).

Branch and bound algorithms

Branch and bound algorithms are commonly used to solve the TSP. The idea is to relax some of the problem constraints first and then regain feasibility through an enumerative process. The quality of a branch and bound algorithm is directly related to the quality of the bound provided by the relaxation (Laporte, 1992). In the branch and bound algorithms for solving the TSP, many relaxations, such as the assignment problem relaxation, the shortest spanning tree relaxation and 2-matching relaxation, can be used. To date, several branch and bound algorithms based on assignment problem relaxation have been proposed for the TSP, such as the algorithms by Carpaneto and Toth (1980), Balas and Christofides (1981) and Miller and Pekny (1991). Christofides (1970) proposed the first branch and bound algorithm based on the shortest spanning tree relaxation to solve the TSP. Since then, improvements and refinements were also provided by some researchers, such as Volgenant and Jonker (1982) and Carpaneto *et al*. (1989). Some algorithms based on 2-matching relaxation have been proposed by Padberg and Rinaldi (1990) and some other researchers.

Branch and cut algorithms

There are some drawbacks for the branch and bound method, such as some new information cannot be exploited during the enumeration phase because the cutting plane phase and the enumeration phase are completely separated and the enumeration process has to be repeated from scratch if the branch and bound algorithm terminates with a sub-tour solution (Padberg and Rinaldi, 1991). To overcome the drawbacks, the branch and cut method was provided by Padberg and Rinaldi in 1987. Padberg and Rinaldi (1987) reported a branch-and-cut algorithm for solving symmetric traveling salesman problem. This algorithm consists of four major components: a heuristic procedure, a linear program solver, a constraint or cut generator and a branch and bound procedure. The 532-city symmetric TSP problem was solved by this algorithm with 358 minutes of computing time on the CYBER 502 Supercomputer. Two real-world problems, 1002-city and 2392-city problems are solved on the same computer with 438 minutes and 1640 minutes respectively. Padber and Rinaldi (1991) presented another branch-and-cut algorithm for solving the TSP problem. The core of their algorithm is a polyhedral cutting-plane procedure that exploits a subset of the system of linear inequalities defining the convex hull of the incidence vectors of the hamiltonian cycles of a complete graph. Whenever the cutting-plane procedure does not terminate with an optimal solution the algorithm uses a tree-search strategy that keeps on producing cuts after branching. Fischetti *et al*. (1997) proposed another exact algorithm which partitions the customer nodes into clusters and the salesman has to visit at least one node for each clusters. Exact and heuristic separation procedures from some classes of facet-defining inequalities are used within the branch-and-cut algorithm for the proposed algorithm. One branch-and-cut algorithm is proposed by Fischetti and Toth to solve the asymmetric TSP problem, in which new separation algorithm for some classes of facet-defining cuts and new variable-prices techniques for dealing with highly degenerate primal linear programming problem are proposed and applied. The computational analysis on several random and real-world problem instances demonstrates that the algorithm outperforms the best assignment problem based algorithms from the literature. Applegate *et al*. (2003) introduced an algorithm for solving very large scale TSP instance. In their algorithm, separation algorithms are devised for subtour inequalities. Methods for adjusting cutting planes to respond to changes in the optimal LP solution are also designed. The computational tests show that the algorithm is quite effective to solve the TSP problem.

2.1.2 Heuristic Methods for the TSP

Karp (1972) showed that TSP is NP-complete. Therefore, it is unlikely that the TSP can be solved by a polynomially optimal algorithm. As a result, it is natural to tackle it by means of heuristic algorithms.

Broadly speaking, heuristic methods can be classified into three categories: construction heuristics, improvement heuristics and composite heuristics. Construction heuristics are meant to involve the gradual building of a solution by adding a new vertex at each step. Improvement methods attempt to improve any initial solution by exchanging edges or nodes. Composite algorithms combine the features of tour construction procedures and improvement procedures.

Construction algorithms

Construction heuristics are algorithms that gradually build a feasible solution while keeping an eye on solution costs (Laporte and Semet, 2002).

The nearest-neighbor algorithm (Rosenkrantz *et al*., 1977)

The nearest-neighbor algorithm is a greedy one because it considers the nearest node at each step of augmenting the incomplete tour. The basic steps of the algorithm are as follows:

Step 1. Select an arbitrary node as a starting point.

Step 2. Find the nearest node to the last selected node and include it in the tour. Repeat Step 2 until all the nodes are included in the tour.

Step 3. Connect the last node to the first one.

The complexity of this procedure is $O(n^2)$ if c_{ij} is metric. The worst case bound for nearest-neighbor algorithm is:

$$\frac{\text{length of nearest-neighbor tour}}{\text{length of optimal tour}} \leq \frac{1}{2}\lceil \lg(n) \rceil + \frac{1}{2},$$

where lg denotes the logarithm of base 2, $\lceil x \rceil$ denotes the smallest integer equal to or greater than x, and n is the number of nodes.

Insertion algorithm (Rosenkrantz *et al*., 1977; Stewart, 1977; Norback and Love, 1977)

The procedures for insertion algorithms are as follows:

Step 1. Construct a first tour consisting of two vertices.

Step 2. Select a chosen node which is not in the current tour and insert it into the tour with respect to one of the following criteria:

- The node resulting in the least distance increment
- The node nearest to the current tour
- The node farthest from the current tour
- The node forming the largest angle with two consecutive nodes of the tour.

The complexity of the insertion algorithm varies between $O(n^2)$ and $O(n^2 \lg n)$ depending on the given criterion.

Savings algorithm (Clarke and Wright, 1964)

The basic idea of the Clarke and Wright algorithm is to calculate the savings and rank them from the largest to the smallest to form a larger subtour until a tour is formed.

The procedure is:

Step 1. Select an arbitrary node, denoted as node 0, as the central depot.

Step 2. Compute savings $S_{ij} = c_{0i} + c_{0j} - c_{ij}$ for i, j =1, 2, 3…, n-1.

Step 3. Rank the savings from largest to smallest in a savings list.

Step 4. Starting from the top of the savings list, form larger subtours by linking the selected nodes i and j. Repeat until a tour is formed.

The complexity of this algorithm is $O(n^2)$.

Improvement algorithms

Improvement algorithms can be used to improve an initial solution, which is generated randomly or constructed by construction algorithms. It can be classified into two categories: local search methods and meta-heuristics.

Local search methods

Lin (1965)'s *r*-opt algorithm is one of the most famous local search methods.

r-opt algorithm (Lin, 1965)

Step 1. Construct an initial tour.

Step 2. Replace *r* arcs by another *r* arcs such that a shorter tour can be obtained. Repeat Step 2 until no further improvement can be achieved.

The major drawback of an *r*-opt algorithm is that the value of *r* must be specified in advance. It is difficult to decide which *r* can give a good balance between solution quality and computational running time. To overcome this drawback, Lin and Kernighan (1973) presented another method to improve the *r*-opt algorithm, known as the variable *r*-opt method. The computational results show that the variable *r*-opt method is one of the best algorithms for solving the TSP.

In the last fifteen years, several meta-heuristics have been developed to solve the TSP. The outstanding feature of these methods is that it allows deterioration and even infeasible intermediate solutions in the exploration of the solution space. The popular methods include simulated annealing, tabu search, genetic algorithm, ant system algorithm, space smoothing algorithm and so on.

Simulated annealing

Simulated annealing (SA) was introduced by Kirkpatrick *et al.* in 1983. It is a technique that first became popular about a decade ago and has since proved itself as an effective approach to a large number of problems. It works by searching the set of all possible solutions, thus reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior solutions under the control of a randomized scheme. Specifically, if a move from one solution s_0 to another neighboring but inferior solution *s* results in a change in objective function value δ, the move to *s* can be accepted if

$$\exp(-\delta / t) < a\,,$$

where *t* is a control parameter, and $a \in [0,1]$ is a uniformly distributed random number. The parameter *t* is initialize to be large hence allowing many inferior moves to be accepted, and is slowly reduced to a small value where inferior moves are nearly always rejected. There is a close analogy between this approach and the thermodynamic process of annealing in physics; it was this analogy that originally motivated the development of the method.

This approach can be regarded as a variant of the well-known heuristic technique of local search, in which a subset of the feasible solutions is explored by repeatedly moving from the current solution to a neighboring solution. In a local search, the main disadvantage of this method is the likelihood of it finding a local, rather than global optimum. By allowing some uphill moves in a controlled manner, SA offer a way of alleviating this problem.

Some components of SA

Initial temperature: The process must start in such a way that most if not all moves can be accepted, that is, the initial temperature, t_0, must be ‘high’. In practice, this may require some knowledge of the magnitude of neighboring solutions. What is meant by a ‘suitably high’ acceptance rate will vary from one situation to another, but some literature shows that an acceptance rate of between 40% and 60% seems to give good results (Dowsland, 1993).

Cooling schedule: Perhaps the most important factor in practical application is the cooling schedule. Here it should be noted that there are basically two types of schedules, each having analogies to homogeneous and inhomogeneous Markov chains, respectively. In the homogeneous case, annealing is carried out at a fixed temperature until ‘equilibrium’ is reached. In the inhomogeneous case, the temperature is reduced after every move. This is less complicated than the homogeneous case, and is the one more commonly used in practice.

In either case, one has to decide on the ‘shape’ of the cooling curve. Two methods are popular. The first one is a geometric schedule:

$$t \leftarrow \alpha \times t,$$

where α is a constant close to 1 (typically in the range 0.9 to 0.99). The other method is

$$t \leftarrow \frac{t}{1+\beta t},$$

where β is a constant near to zero.

Final temperature: In theory the procedure should be continued until the final temperature t_f is zero, but in practice it is sufficient to stop when the chance of accepting an uphill move has become negligible. To some extent, this is problem-dependent. Literature suggests stopping when

$$t \leq \frac{\varepsilon}{\ln[(|S|-1)/\theta]},$$

where S is the solution space. This is designed to produce a solution which is within ε of the optimum with a probability of θ.

Number of iterations: It should be noted that the number of iterations N_{it} is effectively fixed by the above three choices. In the homogeneous case, it also depends on how equilibrium is detected at each state, but in the inhomogeneous case α, β and N_{it} are related by

$$N_{it} = \frac{\log t_f - \log t_0}{\log \alpha},$$

and

$$N_{it} = \frac{t_0 - t_f}{\beta t_0 t_f},$$

respectively.

The acceptance function is usually assumed to be exponential:

$$\exp(-\delta / t) < a$$

It has been pointed out that calculating this function is comparatively expensive on a digital computer. Approximating by $1 - \delta / t$ often saves computer time at little cost in effectiveness.

Re-annealing is a concept that has been used successfully. It involves recording the temperature at which the best solution is found during the time of carrying out an initial SA pass. Then the process is reheated and a lengthier search is carried out at this temperature until some stopping condition is satisfied.

SA has been applied to solve the TSP by several researchers, such as Golden and Skiscim (1986) and Rossier *et al*. (1986). The computational results of Laarhoven (1988) show that SA outperforms some local search methods, such as the 2-opt algorithm. However, the algorithm formulated by Lin and Kernighan (1973) can generate better solutions than the SA.

Tabu search

The tabu search (TS) method was proposed by Glover in 1986 and became one of the most popular local search methods for combinatorial optimization problems. The basic idea is that it prevents the process from cycling over a sequence of solutions when moving from one solution to the best neighboring solution. One way is to forbid the process from going back to previously encountered solutions, but the disadvantage is that it requires excessive bookkeeping. Another method is to register some attributes of past solutions, in such a way that any solution possessing these attributes may not be considered for a given number of iterations. This mechanism is called short-term memory. Long-term memory, often referred to as diversification, is often implemented. The purpose of diversification is to ensure that the search process will not be restricted to a limited portion of the solution. It keeps track of the past solutions and penalizes frequently performed moves. Contrary to diversification, intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found to be good. They may initiate a return to attractive regions to search them more thoroughly.

Some components of TS

Move: A move characterizes the process of generating a feasible solution to the problem that is related to the current solution (i.e. a move is a procedure by which a new solution is generated from the current one).

Aspiration Condition: These are rules that override tabu restrictions. In other words, if a certain move is forbidden by tabu restrictions, then the aspiration criteria, when satisfied, can make this move allowable.

Tabu list: In order to prevent a return to the local optimum just visited, the reverse move that is detrimental to achieve the optimum solution must be forbidden. This is done by storing this move in a tabu list, in which the attributes of some moves made are recorded. The elements of the tabu list are called tabu moves. The reverse moves are restricted from regions that the search explored. The condition for a move to be a tabu move or not can be problem specific. For instance, a move may be tabu if it leads to a solution that has already been considered in the last few iterations.

Shigeru and Evans (1998) investigated the effect of the tabu list in TS for the TSP and showed that the best tabu list size is about $n/4$ for 2-opt based TS and in the range of $n/16$ to $n/8$ for the 3-opt based TS where n is the total number of nodes in the TSP. Knox (1994) showed that the TS which incorporates the 2-opt outperforms the 2-opt and 3-opt algorithms in most cases, especially for large-size problems.

Genetic algorithm

The genetic algorithm (GA) is another type of well-known modern heuristics. It can also be viewed as a form of neighborhood search, although its original inspiration comes from population genetics. Unlike SA and TS, GA makes use of a population of solutions, from which, using selective breeding and recombination strategies, better and better solutions can be produced. Simple genetic 'operators' such as crossover and mutations are used to construct new solutions from pieces of old ones, in such a way that for many problems, the population steadily improves.

GA is an iterative algorithm that maintains a pool of solutions at each iteration (Reeves, 1993). Initially, the pool of solutions is generated randomly and at each iteration, a new pool of solutions is formed by genetic operators that mimic the principles of evolution and heredity. Each solution is evaluated with an objective function, and this process is repeated until some form of convergence is achieved.

Some components for GA

Crossover: All the individuals that have been selected for reproduction are randomly paired. For each pair, a crossover point is randomly chosen. The crossover point is the point where the string representing an individual is split into two parts. Two new individuals are created by swapping the second parts of the pair.

The crossover operators include the single point crossover, two point crossover, uniform crossover, partially matched crossover operator (PMX), order crossover (OX) and cycle crossover (CX), and so on.

Gene: The basic unit of an individual which is represented as a string over a finite alphabet.

Mutation: A mutation provides the opportunity to reach parts of the search space which perhaps cannot be reached by a crossover alone. Each gene of a string is examined in turn, and with a small probability, its current allele is changed.

Fitness function: The fitness function can be a performance measure or reward function, or a critic, or anything at all that can be framed as an optimization problem.

Selection strategy: It is usually randomized, with the probability of selection proportional to the fitness. For instance, if individual *X* scores twice as high as individual *Y* on the fitness function, then *X* is twice more likely to be selected for reproduction than *Y*.

Reproduction: A process in which individuals are copied according to their fitness values. The fitter an individual, the more copies it has. This operator is an artificial version of natural selection.

The basic idea of GA was initially proposed by Holland in 1975, but it was only fully recognized in the research community 10 years later. Jog *et al.* (1989) as well as Schmitt and Amini (1998) showed that GAs with large population size could result in good quality TSP solutions, but at the expense of running time. Chatterjee *et al*. (1996) obtained a near optimal solution for some sample problems of TSP.

Ant system algorithm

The ant system algorithm (AS) is inspired by an analogy related to real ants which are capable of finding the shortest path from a food source to the nest without visual cues (Colorni, 1991). In other words, they are capable of adapting to changes in the environment, *e.g*. finding a new shortest path once the old one is no longer feasible due to a new obstacle.

Ant algorithm was first proposed by Colorni *et al.* in 1991. Then Dorigo *et al.* (1996) and Dorigo and Gambardella (1997) refined the method and then applied it to solve the TSP. In these applications, the artificial ants got three ideas from natural ant behavior:

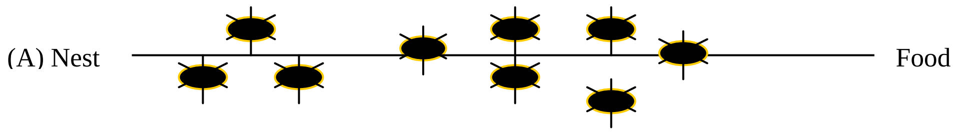
(1) The ants had a preference for paths with a high pheromone level;

(2) There was a higher rate of growth of the amount of pheromone on the shorter paths; and

(3) The trail mediated communication among ants.

The artificial ants were also given a few capabilities which were not found in their natural counterparts, but which were observed to be well suited to the TSP application, i.e., artificial ants can determine how far away cities are, and they are endowed with a working memory used to memorize cities already visited. The working memory is emptied at the beginning of each new tour, and is updated after each time step by adding the newly visited city.

The following figures show the procedure of adapting behavior.



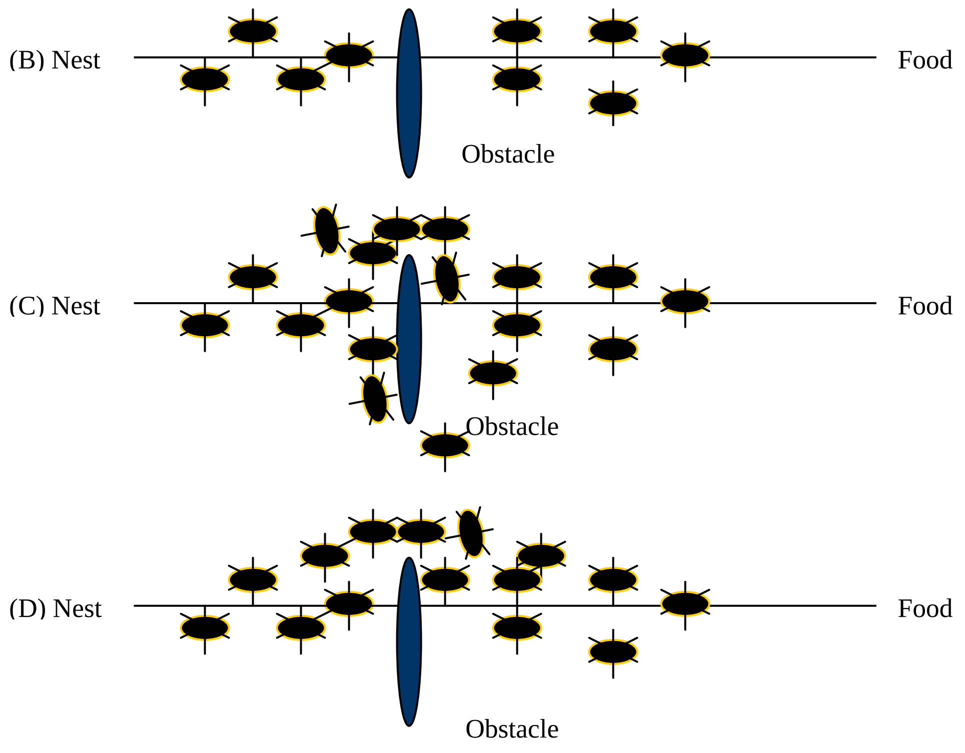


Figure 2.1 Procedure of adapting behavior for ants

It is well known that the primary means for ants to form and maintain a line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to follow a direction rich in pheromone. This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path. In Figure 2.1 (A), ants are moving on a straight line that connects a food source to their nest. In Figure 2.1 (B), once an obstacle has appeared, those ants which are just in front of the obstacle cannot continue to follow the pheromone trail and therefore they have to choose between turning right or left. In this situation, about half the ants choose to turn right and the other half choose to turn left. In Figure 2.1 (C), it is interesting to note that the number of ants which choose the shorter path around the obstacle will increase, because it can reconstitute the interrupted pheromone trail more rapidly, compared to those who choose the longer path. Thus, the shorter path will receive a greater amount of pheromone per time unit and in turn a larger number of ants will choose the shorter path (Figure 2.1 (D)).

Figure 2.1 shows that although all ants move at approximately the same speed and deposit a pheromone trail at approximately the same rate, it is a fact that it is more difficult to make the pheromone trail accumulate quickly on the longer side than the shorter one. Therefore, ants prefer the shorter path because of its higher pheromone trail.

Search space smoothing method

The local search method is effective for combinatorial optimization problems. However, it often gets stuck at a local optimum. A local search method, when coupled with some search space smoothing techniques, can smooth the rugged terrain surface of the search space, and hence it may be able to escape from a local optimum to find a good solution. One of the first heuristics to use data smoothing was developed by Gu

and Huang in 1994. This method tries to avoid being stuck in a poor local optimum by using smoothed distance value instead of the original one. Their computational experiments showed that this method worked well when applied to the TSP. The smoothing function is as follows:

- GH (Gu and Huang, 1994)

The smoothing function is defined as:

$$d_{ij}(\alpha) = \begin{cases} \bar{d} + (d_{ij} - \bar{d})^{\alpha} & \text{if } d_{ij} \geq \bar{d} \\ \bar{d} - (d_{ij} - \bar{d})^{\alpha} & \text{otherwise,} \end{cases},$$

where $\bar{d}$ is the average intercity distance. d_{ij} is the normalized intercity distance and α is the smoothing factor.

In Gu and Huang's search space smoothing method, the schedule for the smoothing factor α is $\alpha = 5, 4, 3, 2, 1$. The smoothing factor is set to 5 for the first step, smooth the distances according to the above smoothing function. Then improve the TSP based on the smoothed distances, followed by updating the smoothing factors 4, 3, 2, 1 to smooth the distance, and improve the TSP solution using the smoothed distance. Repeat it until the value of α is 1. When $\alpha = 1$, the distance is original according to the smoothing function.

The space smoothing search algorithm of Gu and Huang can be described as follows:

Step 1. Let $d_{ij} =$ the distance from city *i* to city *j*. Normalize all distances so that $0 \le d_{ij} \le 1$. Specify the schedule for the smoothing factor (α) from 5 to 1.

Step 2. Generate a random starting tour.

Step 3. Set α to the next value in the smoothing schedule and then smooth the distances according to the following function:

$$d_{ij}(\alpha) = \begin{cases} \bar{d} + (d_{ij} - \bar{d})^{\alpha} & \text{if } d_{ij} \ge \bar{d} \\ \bar{d} - (d_{ij} - \bar{d})^{\alpha} & \text{otherwise} \end{cases},$$

where $\bar{d}$ is the average distance.

Step 4. If α = 1, stop. The current tour is the final tour. Otherwise, using the current tour, go to Step 3.

Gu and Huang (1994) showed that the search space smoothing method can improve the conventional local search algorithm effectively. Schneider *et al.* extended the work of Gu and Huang in 1997 by providing four other smoothing functions (exponential, hyperbolic, sigmoidal, and logarithmic) to solve the TSP. Subsequently, Coy *et al.* (1999) extended the results of Gu and Huang and proposed three other smoothing functions to improve the algorithm.

Coy's functions are defined as follows:

- The concave method (Coy *et al.* 1999)

  The smoothing function is:

  $$d_{ij}(\alpha) = d_{ij}^{1/\alpha}.$$

- The convex method (Coy *et al.* 1999)

  The smoothing function is:

  $$d_{ij}(\alpha) = d_{ij}.$$

- The sequential method (Coy *et al.* 1999)

This sequential method is a combination of concave and convex heuristics. Firstly, the distances between customers are smoothed with a convex function and then the local search heuristics are applied. Secondly, the distances are smoothed with a concave function and then the local search heuristics are applied.

Among the smoothing heuristics proposed by Gu and Huang (1994), Schneider *et al.* (1997) and Coy *et al.* (1999), the computational study of Coy *et al.* (1999) showed that the sequential smoothing method generated the least-cost tours for the TSP. In addition, the sequential smoothing method can avoid getting trapped in poor local minima by alternating between concave smoothing and convex smoothing.

Composite algorithms

A composite algorithm is the combination of the construction algorithm and the improvement algorithm. It constructs one initial solution using some construction algorithms and then improves it by adding one or more improvement procedures. In recent years, several composite algorithms have been developed. Some effective methods, such as the GENIUS algorithm by Gendreau *et al.* (1992) and the nested partitions algorithm by Shi *et al*. (1999) are reviewed in this section.

The GENIUS algorithm (Gendreau *et al*., 1992)

Gendreau *et al.* proposed the GENIUS algorithm in 1992. The GENIUS algorithm consists of two parts: a generalized insertion phase and a post-optimization phase. After the insertion phase, a post-optimization algorithm known as Unstring and Stringing is used to remove and insert nodes to the tour.

Gendreau *et al*. (1992) showed that excellent solutions are obtained when the GENIUS algorithm is used to solve the TSP. For some sample problems, GENIUS is even able to find the optimal solutions.

The nested partitions algorithm (Shi *et al*., 1999)

The nested partitions algorithm was presented by Shi *et al*. in 1999. The basic idea of the nested partitions algorithm is to identify good regions of the solution space and concentrate the search effort in these regions to find good solutions.

The computational results show that the nested partitions algorithm outperforms 2-opt and 3-opt algorithms in solution quality and running time, and in some cases, it can obtain near optimal solutions.

2.2 Approaches for Solving the VRP

2.2.1 Exact Methods for the VRP

Exact algorithms for solving the VRP proposed in the literature, such as set covering based algorithms, branch and bound methods and branch and cut algorithms are reviewed in this section. Most of these exact algorithms are to solve the CVRP, because CVRP is the simplest and most studied type of the VRP family.

Set covering based algorithms

One of the exact methods for solving the VRP is based on a set covering formulation of the problem. Let the index set of all feasible routes be {1, 2, …, *R*} and let c_r be the length of route *r*. Let

$$\alpha_{ir} = \begin{cases} 1 & \text{if customer } i \text{ is served in route } r \\ 0 & \text{otherwise} \end{cases}$$

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

In the set covering formulation of the VRP, the objective is to select a minimum cost set of feasible routes such that each customer is included in some routes. The problem can be formulated as follows:

Problem S: Minimize $\sum_{r=1}^{R} c_r y_r$

subject to

$$\sum_{r=1}^{R} \alpha_{ir} \; y_r \geq 1, \forall i = 1,2,...,n,$$

$$y_r \in \{0,1\}, \forall r = 1,2,...,R.$$

This formulation was first used successfully by Cullen *et al.* (1981) to design heuristic methods for the VRP. Desrochers *et al.* (1992) applied it in conjunction with a branch and bound method to find optimal or near optimal solutions to the VRP.

The set of all feasible routes is extremely large and one cannot expect to generate it completely. To solve the linear relaxation of Problem *S* without enumerating all the routes, Desrochers *et al.* (1992) used the celebrated column generation technique. The general idea is as follows: a portion of all possible routes is enumerated, and the resulting linear relaxation with this partial route set is solved. The solution to this linear program is then used to determine if there are any routes that are not included, which can reduce the objective function value. Using the values of the optimal dual variables (with respect to the partial route set), a new route is generated and the linear relaxation is resolved. This process is continued until one can show that an optimal solution to the linear program is optimal for the complete route set. This method can be combined with a polyhedral approach that generates an optimal or near-optimal solution to the VRP.

Branch and bound algorithms

The branch and bound algorithm has been the most effective method for the VRP in the last few decades (Toth and Vigo, 2002*a*). Up to the end of the last decade, most of the branch and bound methods used basic relaxations such as the assignment problem and the shortest spanning tree. In recent years, problems of larger sizes have been solved to optimality by using more sophisticated bounds based on Lagrangian relaxations or the additive approach. Among the methods, only a few algorithms were proposed to solve the ACVRP. They included an algorithm whose lower bound was based on the assignment problem by Laporte and Nobert (1986); and an algorithm whose lower bound was based on the additive approach by Fischetti *et al.* (1994). Normally, the basic combinatorial relaxations can be used to solve for the optimal solution of the problem with small size only. To improve the bounding technique, some researchers, such as Fisher (1994) and Miller (1995), proposed to strengthen the basic relaxation by dualizing some of the relaxed constraints. An exact algorithm is proposed by Fisher (1994) to solve the CVRP problem. The CVRP can be modeled as the problem of finding a minimum cost *K*-tree (*K* is equal to number of vehicles) with two *K* edges incident on the depot and subject to some side constraints that impose vehicle capacity and the requirement that each customer be visited exactly once. The side constraints are dualized to obtain a Lagrangian problem that provides lower bounds in a branch-and-bound algorithm. The author's computational results show that the exact algorithm has produced proven optimization solutions for a number of difficult problem in the literature and server real problems with 25-71 customers. Miller (1995) presented another branch-and-bound CVRP algorithm. The lower bounds are derived by relaxing the subtour elimination and vehicle capacity constraints to yield a perfect *b*-matching problem. The subtour elimination and vehicle capacity constraints are expressed by a single family of inequalities called generalized subtour elimination constraints. Bounds are strengthened by using Lagrange multipliers to enforce subtour elimination and capacity constraints. This method can increase the size of the instances solvable by branch and bound effectively.

Branch and cut algorithms

The branch and cut algorithm has been extremely successful in finding optimal solutions for large-size TSPs, while its application to the CVRP is still in the initial stages of its development (Naddef and Rinaldi, 2002). A better understanding of the underlying polytope as well as further effort in designing efficient separation routines are needed to provide better solutions for the CVRP. Many researchers are working on this approach for solving the VRPs, and some of them have been successful in solving VRPs of certain sizes. Cornuejols and Harche (1993), for example, studied the facial structure of a set of feasible solutions and suggested a branch and cut procedure.

Augerat *et al.* (1995) implemented a branch and cut algorithm to solve some large problems which have never been solved before.

2.2.2 Heuristic Methods for the VRP

Heuristic Methods for the VRP can be classified into three categories: construction heuristics, improvement heuristics and composite heuristics (Laporte and Semet, 2002). The construction heuristics are used to build a feasible solution while keeping an eye on solution costs. Improvement methods attempt to improve any feasible solution by exchanging edges or nodes within or between vehicle routes. Composite heuristics are divided into two classes: cluster-first, route-second methods and route-first, cluster-second methods (Laporte and Semet, 2002). In the first case, vertices are divided into feasible clusters and then a route in each cluster is constructed. In the second one, a route visiting all the customers is constructed first and then segmented into feasible vehicle routes.

Construction algorithms

Generally, two techniques have been used on constructive methods, i.e., merging existing routes using a savings criterion and assigning vertices to vehicle routes using insertion costs.

The Clarke and Wright (1964) algorithm can be used to solve not only the TSP (See Section 2.1.2) but also the VRP. Its basic idea is to calculate the savings and rank them from the largest to smallest, to form a larger sub-tour. The only difference between these two implementations is that the vehicle capacity should be considered in the implementation of the VRP.

The savings methods of Clarke and Wright produces good routes at the beginning but less interesting routes are likely to be produced towards the end. To improve on it, Yellow (1970) proposed a generalized savings formula: $S_{ij} = C_{i0} + C_{0j} - \lambda C_{ij}$, where λ is a route shape parameter. It was found that the larger the λ, the more emphasis was put on the distance between the vertices to be connected.

Mole and Jameson (1976) proposed one type of sequential insertion heuristics. Their algorithm uses two parameters λ and μ to expand a route under construction:

$$\alpha(i,k,j) = C_{ik} + C_{kj} - \lambda C_{ij}$$

$$\beta(i,k,j) = \mu C_{0k} - \alpha(i,k,j)$$

For each un-routed vertex *k*, construct an emerging route (*0, k, 0)* and then compute the feasible insertion cost $\alpha^*(i_k, k, j_k) = \min\{\alpha(r,k,s)\}$ for all adjacent vertices *r* and *s* of the emerging route, where i_k and j_k are the two vertices yielding α^*. If the insertion is feasible, the best vertex *k** to insert into the emerging route is the vertex yielding $\beta^*(i_{k^*}, k^*, j_{k^*}) = \max\{(\beta(i_k, k, j_k)\}$ over all un-routed vertices *k* that can be feasibly inserted. Insert *k** between i_{k^*} and j_{k^*}. Improve the current route by means of a 3-opt procedure (Lin, 1965).

Procedure of the sequential insertion algorithm:

Step 1. If all vertices belong to a route, stop. Otherwise, construct an emerging route (*0, k, 0*), where *k* is any un-routed vertex.

Step 2. Compute for each un-routed vertex *k*, the feasible insertion cost $\alpha^*(i_k, k, j_k) = \min\{\alpha(r,k,s)\}$ for all adjacent vertices *r* and *s* of the emerging route, where i_k and j_k are the two vertices yielding α^*. If no insertion is feasible, go to Step 1. Otherwise, the best vertex *k** to insert into the emerging route is the vertex yielding $\beta^*(i_{k^*}, k^*, j_{k^*}) = \max\{(\beta(i_k, k, j_k)\}$ over all un-routed vertices k that can be feasibly inserted. Insert k^* between i_{k^*} and j_{k^*}.

Step 3. Optimize the current route by means of a 3-opt procedure (Lin, 1965) and go to Step 2.

Christofides *et al.* (1979) developed a more sophisticated two-phase insertion that also used controlled parameters: λ and μ. The computational results showed that it performed better than the sequential insertion method of Mole and Jameson (1976).

Improvement algorithms

Improvement algorithms can be used on vehicle routes that are taken separately or where several routes are taken at a time.

Thompson and Psaraftis (1993) proposed a multi-route improvement algorithm. This is a general *b*-cyclic, *k*-transfer scheme in which a circular permutation of *b* routes is considered and *k* customers from each route are shifted to the next route of the cyclic permutation.

It has been almost 40 years since the publication of the savings heuristic for the VRP, and during this period, many classical heuristics have been proposed. Some comparisons show that the solution quality of the classical heuristics based on simple construction and local descent improvement techniques do not compete with the best meta-heuristic implementations. Thus, as there was little room left for significant improvement in the area of classical heuristics, researchers next turned their attention to some other methods, such as the meta-heuristic methods.

Simulated annealing

Two known methods using SA to solve the CVRP have emerged in recent years: One is based on Osman's (1993) λ-interchange method. The λ-interchange method can be described thus: two routes, route *p* and route *q*, are first selected from the current solution. Then subsets of customers S_p and S_q satisfying $|S_p| \leq \lambda$ and $|S_q| \leq \lambda$ are chosen, one from each route. The customers in S_p and the customers in S_q are swapped as long as the solution remains feasible. In his implementation, Osman used the λ-interchange ($\lambda=2$) local search method, allowing for a mix of single and double vertex moves, and single and double vertex swaps between vehicle routes.

The other method is based on Breedam's (1995) string cross, string exchange and string relocation methods. Breedam classified the improvement operations as 'string cross', 'string exchange', 'string relocation', and 'string mix', which could all be viewed as special cases of 2-cyclic exchanges of the Thompson and Psaraftis (1993) cyclic transfer algorithm. A string cross is one in which two strings of vertices are exchanged by crossing the edges of two different routes. The string exchange attempts to improve the VRP solution by exchanging customers or strings of customers between every two routes. String relocation is used to insert a customer or a string of customers from one route into another route. A string mix is a mix of string cross, string relocation and string exchange procedures and this procedure tries to cross, exchange or relocate customers or strings of customers, depending on which yields the greatest savings. The computational analysis on some test problems was also provided in this study.

Tabu search

Over the last 10 years or more, tabu search has been applied to the CVRP by several researchers. Some of the earliest tabu implementations did not yield impressive results, while subsequent implementations were much more successful (Gendreau *et al.*, 2002).

Pureza and França (1991) defined the neighbors of a solution by moving one node to another route or swapping the nodes between two routes while preserving feasibility.

One of the most successful methods was presented by Osman in 1993. In this implementation, the λ-interchange ($\lambda=2$) local search method was used and two strategies for selecting a neighbor solution were proposed. One strategy, known as 'best improve'(BI), means that the best non-tabu solution is selected. In the second strategy, known as 'first improve' (FI), the first admissible improving solution is selected if one exists, otherwise, the best admissible solution is retained. The computational results showed that the FI is slight better than BI in solution quality. Gendreau *et al.* also presented an implementation of tabu search for solving the CVRP in 1994. The neighbor solutions were obtained by moving a vertex from its current route to another route containing one of its closest neighbors. The route inserted was performed concurrently with a local re-optimization using the GENI mechanism for the TSP (Gendreau *et al.*, 1992). Taillard's (1993) tabu method shares some features of Gendreau's method, such as the diversification strategy. However, in Gendreau's algorithm, Osman's λ-interchange mechanism was used to generate neighbors, and standard insertions, rather than using the insertions with GENI, were used to reduce running time.

One important concept, the adaptive memory procedure (AMP) was presented by Rochat and Taillard in 1995. An adaptive memory is a pool of good solutions that is updated throughout the search process. The basic idea of this procedure is that some elements of these solutions are extracted from the pool and combined to produce new good solutions. The procedure is mostly used in tabu search, but its applicability is not limited to this type of meta-heuristics. Another promising concept, the granular tabu search, was proposed by Toth and Vigo in 1998. The basic idea is that longer edges do not belong to the optimal solution for most cases. The unpromising solutions are not considered by eliminating the edges whose length exceeds a threshold.

Xu and Kelly (1996) and Rego and Roucairol (1996) applied more sophisticated neighborhood structures to tabu search to solve the VRP. Computational experience shows that tabu search is the best heuristic for the CVRP to date (Gendreau *et al.*, 2002). Its success is due to some key implementation ideas. These include the allowance of infeasible solutions during the search procedure, the use of self-adjusting parameters, diversification and intensification, and so on. These ideas have contributed to both the success of improving the quality of solution and the saving of running time.

Genetic algorithm

Breedam (1996) made a comparison of a GA with some SA and TS algorithms on some types of CVRPs. In the local search of the GA, he used the local search descent operator based on four different types of exchange moves and applied it only to the best solutions in the current solution pool. The computational results showed that the solution produced by the GA is comparable to the solutions obtained by the SA and TS.

Cases and applications of the GA are rather limited, and most research is focused on solving TSPs and VRPTWs. In fact, some very effective implementations of the GA to VRPTWs were reported by Potvin and Bengio (1996) and some other researchers.

Ant system algorithm

Only a few papers have been presented on the application of the AS to solve the VRP. In 1998, Kawamura *et al.* proposed one complex hybrid variant of AS that involved 2-opt improvement procedures and probabilistic acceptance. Bullnheimer *et al.* (1998, 1999) provided two applications of the AS to the VRP. In these applications, they developed two hybrid ant systems in which each vehicle route produced in a given iteration was improved by the 2-opt heuristic before the trail update. In their procedure, they used a number of 'elitist ants' to account for the best solutions. The results of the above methods were encouraging.

Overall, meta-heuristic methods can produce some excellent solutions, sometimes optimal solutions, to the instances with a few hundred customers. Among them, TS has given the best performance to date. The performances of the GA and AS have not been competitive with other methods, but they could be improved in the future because these two methods have not been fully exploited.

Composite algorithms

Generally, there are two types of composite methods: the cluster-first, route-second method and the route-first, cluster-second method. Most of the methods belong to the first type.

The cluster-first, route-second method

Gillett and Miller (1974) proposed one cluster-first, route-second method, known as the sweep algorithm. In this method, the locations that are used to make up each route are determined according to the polar-coordinate angle for each location. Assume that each vertex *i* is represented by its polar coordinates (θ_i, ρ_i), where θ_i is the angle and ρ_i is the ray length. Assign a value $\theta_{i*} = 0$ to an arbitrary vertex *i** and compute the remaining angles from *(0, i*)*. Rank the vertices in increasing order of their θ_i and then assign vertices to an unused vehicle, from the un-routed vertex having the smallest angle, as long as its capacity or the maximal route length is not exceeded. In other words, feasible clusters are formed by rotating a ray centered at the depot. Then improve each vehicle route by solving the TSP.

Procedure of the Gillett and Miller sweep algorithm

Step 1. Choose an unused vehicle *k*.

Step 2. Starting from the un-routed vertex having the smallest angle, assign vertices to vehicle *k* as long as its capacity or the maximal route length is not exceeded.

Step 3. Optimize each vehicle route separately.

The algorithm of Fisher and Jaikumar (1981) is one of the most famous cluster-first, route-second methods. In this method, they used a generalized assignment problem (GAP) to form clusters. The cost matrix for the GAP can be calculated by

$$d_{ik} = \min\{C_{0i} + C_{i\,j_k} + C_{j_k 0}, C_{0 j_k} + C_{j_k i} + C_{i0}\} - (C_{0 j_k} + C_{j_k 0})$$

where j_k are the seed vertices in V to initialize each cluster k, d_{ik} is the cost of allocating each customer i to each cluster k.

Procedure of the Fisher and Jaikumar GAP algorithm

Step 1. Choose seed vertices j_k to initialize each cluster k.

Step 2. Compute the cost d_{ik} of allocating each customer i to each cluster k as

$$d_{ik} = \min\{C_{0i} + C_{i\,j_k} + C_{j_k 0}, C_{0 j_k} + C_{j_k i} + C_{i0}\} - (C_{0 j_k} + C_{j_k 0}).$$

Step 3. Solve the GAP with cost d_{ik}.

Step 4. Solve a TSP for each cluster corresponding to the GAP solution.

Bramel and Simchi-Levi (1995) proposed a two-phase composite algorithm in which the seeds are determined by solving a capacitated location problem and the remaining vertices are gradually included into their allotted route in the second stage. In this method, first locate *k* seeds, known as concentrators, from the *n* customer locations to minimize the total distance of customers to their closest seed. Then vehicle routes are constructed by inserting the customer that is assigned to that route seed and has the least insertion cost at each step.

It is assumed that the number of vehicles is a fixed number in the Fisher and Jaikumar GAP algorithm and in the method by Bramel and Simchi-Levi (1995).

The route-first, cluster-second method

In the route-first, cluster-second method, a giant TSP tour is constructed in the first phase, disregarding side constraints, and then this tour is decomposed into feasible vehicle routes in the second phase. Normally, this idea is applied to the problem with an unlimited number of vehicles. However, some researchers, such as Dijkstra (1959) as well as Bertsimas and Simchi-Levi (1996), when discussing this type of algorithm, found from their computational experiences that route-first, cluster-second heuristics are not competitive with other approaches.

2.3 Stochastic Vehicle Routing Problem

The stochastic vehicle routing problem (SVRP) is a generalization of the deterministic VRP. The SVRP differs from the VRP in the introduction of some element of variability within the system in question (Hadjiconstantinou and Roberts, 2002). For the SVRP, the following modifications to the VRP are required (Bodin *et* a*l*., 1983):

- Customer demand must be a random variable with a known probability distribution;

- Routes must be designed before the actual demands become known; and
- Other costs might be considered when calculating the expected travel distance.

There are two basic types of SVRPs: the SVRPs with stochastic traveling times (VRPST) and the SVRPs with stochastic demands (VRPSD) (Hadjiconstantinou and Roberts, 2002). A comprehensive review of SVRPs can be found in Gendreau *et al*. (1996).

Kao (1978) presented two heuristic methods to solve the VRPST: one is based on dynamic programming and the other on implicit enumeration. Sniedovich (1981) and Carraway *et al*. (1989) continued to improve Kao's (1978) dynamic programming method. Laporte and Loubeaus (1993) developed an integer L-shaped method that can give exact results for the VRPST with 20 customers.

Tillman (1969) developed an algorithm that was adapted from the Clarke and Wright (1964) algorithm to solve the VRPSD. Following that, Teodorović and Pavković (1992) presented a simulated annealing heuristics and Gendreau *et al*. (1994) proposed a tabu search algorithm to solve the VRPSD. A comparative study of meta-heuristics, including simulated annealing, tabu search and threshold accepting, was conducted by Teng *et al*. in 2003. The computational results show that the solution quality of tabu search outperforms the other heuristics for the sample of the tested VRPSD.

Chapter 3

Assignment-Based Local Search Method

In this chapter, a new local search method called the assignment-based local search (ABLS) is proposed. This algorithm is thus named because the inserting of nodes into routes at each step is based on the solution of an assignment problem. The basic idea, classification, parameter setting and computational experiments for the proposed ABLS algorithm are discussed in detail in this chapter.

3.1 Introduction to the ABLS Method

An improvement algorithm tries to improve a feasible solution by performing a sequence of edge or vertex exchanges within or between vehicle routes for the VRP. It can operate on each vehicle route taken separately or on several routes at a time. For single route improvement, all improvement algorithms developed to solve the TSP can be applied. To date, neighborhood search algorithms for the VRP and other scheduling problems have focused almost exclusively on single-route problems. The most popular algorithm for improving the single route is Lin's (1965) λ-opt heuristic. Besides this, Croes (1958), Lin and Kernighan (1973) and Stewart (1987) also proposed some algorithms for the single route improvement problem. By contrast, only a few researchers have focused on multi-route improvement. Thompson and Psaraftis (1993) presented a general '*b*-cyclic, *k*-transfer' scheme in which a circular permutation of *b* routes is considered and *k* customers from each route are shifted to the next route of the cyclic permutation. Breedam (1995) classified the improvement operations as 'string cross', 'string exchange', 'string relocation' and 'string mix'. And all of these operations can be viewed as the special cases of '2-cyclic' exchanges.

Kinderwater and Savelsbergh (1997) defined similar operations and performed experiments mostly in the context of the VRP with time windows.

However, both Thompson and Psaraftis (1993)'s '*b*-cyclic, *k*-transfer' scheme and Breedam (1995)'s 'string cross', 'string exchange', 'string relocation', 'string mix' operations just relocate the chosen nodes randomly. To improve the performance of the multi-route improvement algorithm, a local search method, ABLS algorithm, is proposed. In this section, the ABLS algorithm, which can operate on several routes at a time, will be discussed in detail.

3.1.1 Basic idea of the ABLS Method

The basic idea of the ABLS method is to exchange or move several nodes to other routes. In other words, for any feasible initial solution, choose a certain number of nodes and delete them from the current routes. Then try to insert back the chosen nodes by solving an assignment problem whose corresponding cost matrix is obtained by some insertion procedure. It is noted that the number of nodes chosen should be less than or equal to the number of routes to guarantee that the chosen nodes can be inserted into the vehicle routes.

Assignment problem

The assignment problem is a special type of linear programming problem, in which, the objective is to determine how to assign assignees to perform tasks to minimize the total costs needed while the following assumptions are satisfied:

- The number of assignees is equal to the number of tasks;
- Each assignee is to be assigned to exactly one task;
- Each task is to be performed by exactly one assignee; and

- There is a cost c_{ij} associated with assignee i performing task j.

When the number of assignees is not equal to the number of tasks, dummy assignees and dummy tasks will be added to construct one assignment problem. Then the corresponding cost in the cost matrix is set to zero to avoid this assignment in the optimal solution.

Let v denote the number of tasks.

Then let $x_{ij} = \begin{cases} 1 & \text{if assignee } i \text{ performs task } j \\ 0 & \text{otherwise} \end{cases}$.

The assignment problem can be formulated as a linear programming problem as follows (Hillier and Lieberman, 1995):

$$\text{Minimize } Z = \sum_{i=1}^{v}\sum_{j=1}^{v} c_{ij} x_{ij}$$

subject to

$$\sum_{j=1}^{v} x_{ij} = 1 \quad \text{for } i = 1,2,\ldots,v,$$

$$\sum_{i=1}^{v} x_{ij} = 1 \quad \text{for } j = 1,2,\ldots,v,$$

$$x_{ij} = 0 \text{ or } 1, \quad \text{for } all\ i \text{ and } j.$$

The cost matrix for the assignment problem of the VRP can be calculated by some insertion procedures, such as the nearest insertion, cheapest insertion, arbitrary insertion, farthest insertion and the greatest angle insertion. With regard to the CVRP case, the cheapest insertion is applied because of its efficiency of computation of insertion cost. The basic idea of the cheapest insertion procedure is to try to insert one node between two adjacent nodes that can result in the least increment in distance. For instance, suppose three nodes are deleted from four routes, and attempts are made to insert them back to these routes. The cost matrix of the associated assignment problem is shown in Table 3.1. The value of each cell, g_{ij}, is the least cost increment, if node i is to be inserted into the route j. If it is infeasible to insert node i into route j, the g_{ij} is set to a very large positive number, M. For the CVRP, if the demand of the node is greater than the remaining capacity of the current route, the corresponding cost in the cost matrix should be set to be M. In the case where the node to be inserted is a dummy node, g_{ij} is set to zero.

Table 3.1 Cost matrix of the assignment problem

	Route 1	Route 2	Route 3	Route 4
Assignee 1	g_{11}	g_{12}	g_{13}	g_{14}
Assignee 2	g_{21}	g_{22}	g_{23}	g_{24}
Assignee 3	M	M	g_{33}	g_{34}
Assignee 4 (Dummy)	0	0	0	0

The assignment problem can be solved by the well known Hungarian algorithm (Kuhn, 1955) or some other methods. In this thesis, the shortest augmenting path algorithm by Jonker and Volgenant (1987) is used because it has been shown to be more efficient than Hungarian algorithm.

3.1.2 An Example of the ABLS Method

An illustration on how the ABLS algorithm can be used to solve the VRP is given in the following example.

Assume that there are 10 customers whose demands are to be served by 3 identical vehicles. The capacity for each vehicle is 3500. The demands and coordinates of each customer are given in Table 3.2.

Table 3.2 Demands and coordinates of customers for the CVRP example

Index of Customer	*x*-coordinate	*y*-coordinate	Demand
0	145	215	0
1	151	264	1100
2	159	261	700
3	130	254	800
4	128	252	1400
5	163	247	2100
6	146	246	400
7	161	242	800
8	142	239	100
9	163	236	500
10	148	232	600

Note: Customer 0 denotes the depot.

Step 1. Suppose that the initial or current solution is given as shown in Figure 3.1. In the solution, there are 3 routes, and each of them begins and ends at depot, 0:

Route 1: 0-9-2-1-6-8-10-0, Remaining capacity is 100 and distance is 114.7.

Route 2: 0-7-3-4-0, Remaining capacity is 500 and distance is 108.2.

Route 3: 0-5-0, Remaining capacity is 1400 and distance is 73.4.

Total distance is 296.3.

Step 2. Choose some nodes and delete them from the current solution. In this instance, assume that node 9 is deleted from route 1, while no node is chosen from route 2 and node 5 is chosen from route 3. The solution after deleting nodes 9 and 5 is shown in Figure 3.2. The routes after deletion are as follows:

Route 1: 0-2-1-6-8-10-0, Remaining capacity is 600 and distance is 109.9.

Route 2: 0-7-3-4-0, Remaining capacity is 500 and distance is 108.2.

Route 3: Empty, Remaining capacity is 3500 and distance is 0.

Total distance is 218.1.

Step 3. Calculate the cost matrix for the assignment problem to reassign the nodes deleted into the routes. The cost matrix for this example is given in Table 3.3.

Step 4. Solve the assignment problem with the cost matrix shown in Table 3.3, and insert the nodes to the corresponding location of the route according to the solution. The optimal solution to the assignment problem is: node 9 is assigned to the location between the depot and node 7 in route 2, while node 5 will be inserted back to route 3. The solution after the insertion is shown in Figure 3.3. Then the solution after insertion becomes:

Route 1: 0-2-1-6-8-10-0, Remaining capacity is 600 and distance is 109.9.

Route 2: 0-9-7-3-4-0, Remaining capacity is 0 and distance is 110.8.

Route 3: 0-5-0, Remaining capacity is 1400 and distance is 73.4.

Total distance is 294.1.

After one iteration, it can be seen that the total distance is reduced from 296.3 to 294.1.

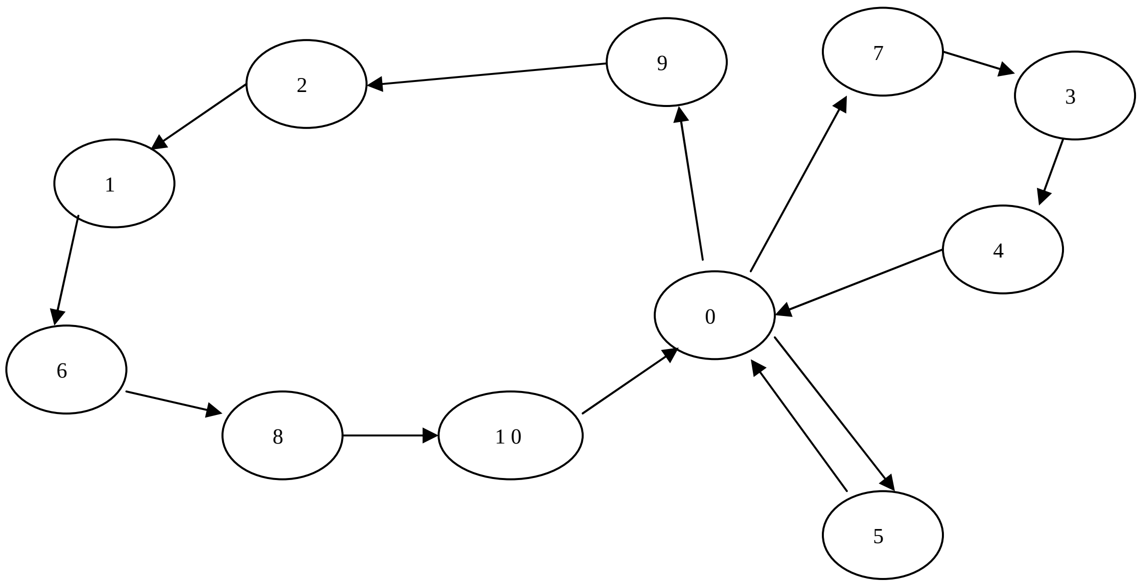


Figure 3.1 Current routes of the CVRP example

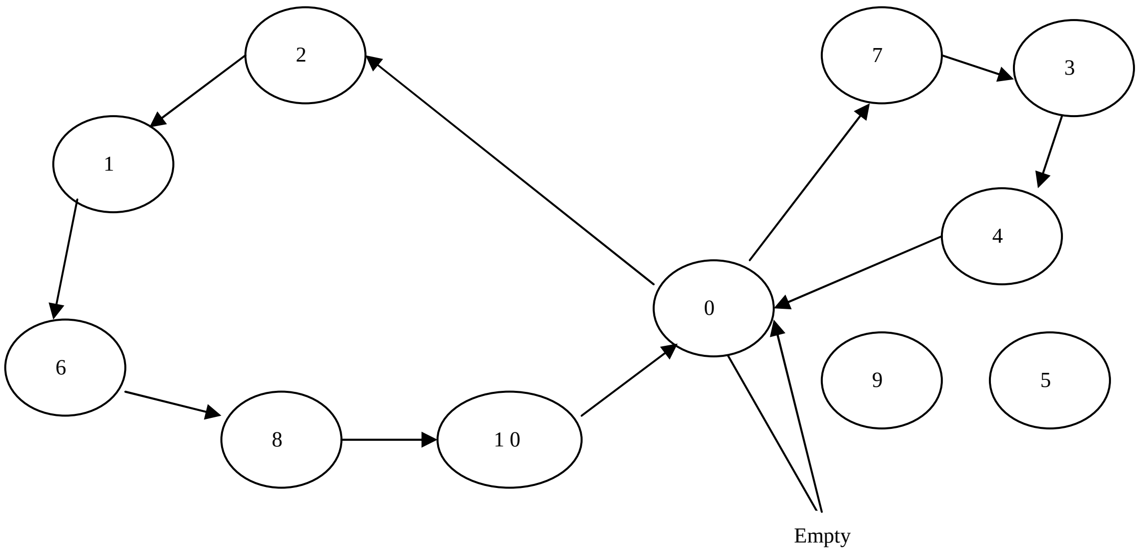


Figure 3.2 Routes after deletion of the CVRP example

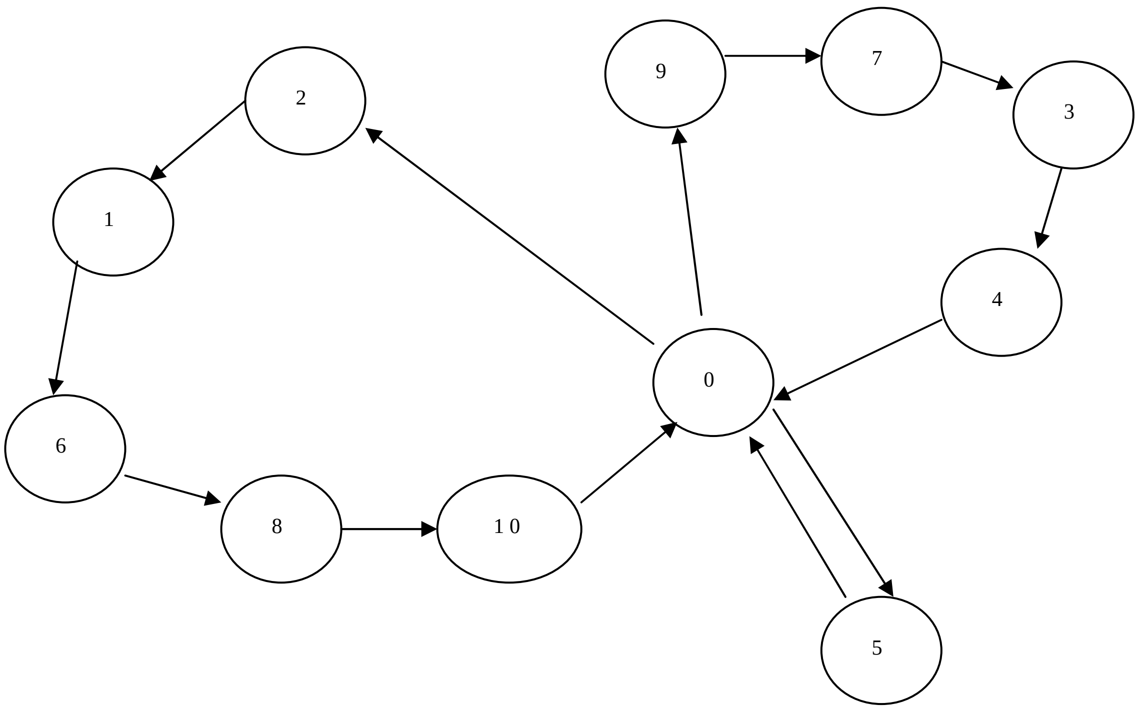


Figure 3.3 Routes after reassignment of the CVRP example

Table 3.3 Cost matrix of the CVRP example

	Route 1	Route 2	Route 3
Assignee 1 (node 9)	4.89	2.59	55.31
Assignee 2 (Dummy)	0	0	0
Assignee 3 (node 5)	M	M	73.43

3.1.3 Types of Problems That Can Be Solved by the ABLS

The ABLS method can be applied to solve the CVRP as well as some variants of the CVRP, such as the DCVRP and the CVRP with multiple trips, and so on.

For the CVRP, only one constraint, i.e., the capacity constraint, needs to be considered. In other words, only the capacity constraint is considered when calculating the cost matrix. In the cost matrix, the cost value should be set to *M* if the demand of the node is greater than the remaining capacity of the corresponding route.

To solve the DCVRP, one more distance-limit constraint should be considered when calculating the cost matrix for the assignment problem. In the case where the demand of the node to be inserted is not greater than the remaining capacity of one particular route, but the length of the route after insertion will exceed the limit length for the route, the corresponding cost in the cost matrix is set to *M* to prevent the node from being inserted into the route. This guarantees the feasibility of the solution obtained.

The CVRP with multiple trips is similar to the CVRP except that each vehicle is allowed to be used more than once. The ABLS algorithm can be used to solve a CVRP with multiple trips effectively because only routes in the solution, and not vehicles, are considered in the improvement procedure. Throughout the whole improvement procedure, one can simply disregard which vehicle is to be assigned to which route temporarily.

In case of the demand is greater than the capacity of vehicles, the demand can be split into two parts. The first part is served by vehicles with full loads until the remaining demand is not greater the capacity of the vehicle. The route with full loads will be untouched in the subsequent improvement procedure if the route contains exactly one customer.

For other practical VRPs with different constraints, such as the case in which only a subset of vehicles can be used to serve some particular customers, the procedure of improvement, especially the cost matrix for the assignment problem, can be revised accordingly.

3.2 Classifications of the ABLS Method

In this section, the proposed ABLS algorithm is classified into two basic types depending on the way the nodes are chosen. They are:

- Type A: Choose zero or one node from each route according to a preset probability, *p*. In other words, for each route a node is to be chosen randomly with probability, *p* and no node will be chosen with probability 1- *p*.

To improve some solutions obtained further, Type A of ABLS algorithm is provided. The strength of this algorithm is that Type A of ABLS algorithm cannot result in worse solutions when only feasible solutions are allowed in the improvement procedure. Therefore, it is possible to improve the solutions when it is difficult to be improved using other algorithms.

- Type B: Choose nodes regardless of which routes they belong to. In this case, a predetermined number of nodes are to be chosen randomly from the set of all customers.

To make the algorithm more flexible, the constraint of choosing nodes has been relaxed in Type B of the algorithm. The nodes can be chosen regardless of the routes they come from. That is, more than one node can be chosen from one route at a time.

There are several variants for each type of ABLS algorithm, depending on which strategies are to be implemented.

In this study, the following four strategies are proposed:

Strategy 1: An infeasible solution is not allowed in the improvement procedure

When using this strategy, infeasible solution is not allowed in the improvement procedure. The advantage of the implementation of this strategy to Type A is that it will never generate a solution that is worse than the current solution.

Strategy 2: An infeasible solution is allowed in the improvement procedure

This strategy can accept an infeasible solution in the improvement procedure. With this modification, the procedure may avoid getting stuck at a locally optimum solution, and hence the algorithm may be able to find a better solution. However, for the cases that result in infeasible solutions easily, such as for the case in which the remaining capacities of routes are quite small compared with the demands of most nodes, the algorithm with this strategy is not quite effective.

Strategy 3: The best solution within a neighborhood of the current solution is selected

In this strategy, the best improvement solution is implemented in the neighborhood search. In other words, only the move that can result in the best solution is accepted in the neighborhood. The weakness of this strategy is longer running time are needed.

Strategy 4: The first improved solution in the neighborhood of the current solution is selected

The first improved solution found in the neighborhood of the current solution is adopted in *Strategy* 4. For the algorithm with this strategy, the running time needed is less than the algorithm with strategy 3 for most cases.

The neighborhood structure for Strategies 3 and 4 can be defined as follows: given a current solution, its neighborhood can be considered as the set of solutions that can be obtained from the current solution by making one move. For the ABLS, the move consists of choosing and deleting some nodes from the current route and then inserting them into the deleted routes. For instance, for Type B of the ABLS, the neighborhood is the set of solutions that is obtained by choosing and deleting 3 nodes from the current solution and then inserting them into the deleted routes assuming that the number of nodes is preset to 3. The neighborhood size can be determined by experiments.

By using different ways of choosing nodes and adopting different strategies in the improvement procedure, the following eight implementations of the ABLS method can be applied to solve several types of VRPs. A summary of the classifications of the ABLS methods is given in Table 3.4.

Table 3.4 Summary of classifications of the ABLS

Type	Strategy 1	Strategy 2	Strategy 3	Strategy 4
Type A1	Y			
Type A2	Y		Y	
Type A3	Y			Y
Type A4		Y		
Type A5		Y	Y	
Type A6		Y		Y
Type B1	Y			
Type B2	Y		Y	
Type B3	Y			Y
Type B4		Y		
Type B5		Y	Y	
Type B6		Y		Y

Composite algorithms

In practice, either a simple type of algorithm or a combination of different types of algorithms can be used to solve the VRP. There are many combinations of the various types of algorithms and the most effective combination can be different for different problems.

As an illustration, two composite algorithms are given below.

```
Composite algorithm 1

{
        Type A5;
        Type A1;
}
```

Composite algorithm 1 includes two parts: Type A5 and Type A1. It means that some nodes are chosen according to the preset probability, p_1, and deleted from the routes. The assignment problem is solved with the cost matrix that is calculated by the cheapest insertion method. Then the nodes are inserted into the routes according to the solution of the assignment problem. Normally, for Type A5, the preset probability, p_1, is relatively large as Type A5 is the first part of the algorithm. Such a parameter setting can help to improve the initial solution quickly. For the neighborhood structure, only the move that can result in the best solution is adopted.

In part 2, zero or one node from each route is chosen according to a preset probability, p_2. Then the cost matrix is calculated and the chosen nodes are inserted into the corresponding routes according to the solution of the assignment problem. It is noted that p_2 can be different from p_1. The values of the probabilities will be determined by experiments.

```
Composite algorithm 2

{
        Type B6;
        Type A1;
 }
```

Composite algorithm 2 includes two parts: Type B6 and Type A1. The first part is different from that of composite algorithm 1 while part 2 of both composite algorithms are similar.

In the first part, one node is chosen from some routes. Then they are inserted into the corresponding route according to the solution of the assignment problem. In the improvement procedure, Strategy 4 is used. In other words, for each neighborhood structure, the first solution that can improve the current solution is accepted. Infeasible solutions are allowed throughout the entire procedure.

For most composite algorithms, Type A1 is always used as the last part of the whole algorithm because it cannot result in a worse solution than the current one.

For each algorithm, the same type of algorithm can be used more than once with different parameters. For example, the number of nodes chosen can be changed, that is, the number of nodes is relatively large if it is at the beginning of the algorithm and the number is small if it is near the end of the algorithm.

3.3 Computational Results and Analysis

To evaluate the performance of the ABLS algorithm, extensive computational experiments on the proposed algorithms applied to a set of benchmark problems are carried out. The computation results and analysis for various implementations of the ABLS algorithms are presented in the following subsections.

3.3.1 Test Instances and Initial Solution

To evaluate the performance of ABLS algorithm, fourteen traditional test instances taken from literature (Christofides, and Eilon, 1969, Christofides *et al*., 1979) are used in the computational study. The test instances are all Euclidean SCVRP instances and for all of the cases, the traveling times are assumed to be equal to the distances. The characteristics of the test problems are summarized in Table 3.5.

Table 3.5 Characteristics of the test instances

Problem	Problem size	No of vehicles	Vehicle capacity	Max total route length	Service time	Capacity ratio	Reference
Problem 1	50	5	160	Infinite	0	0.97	Christofides, and Eilon, 1969
Problem 2	75	10	140	Infinite	0	0.97	Christofides, and Eilon, 1969
Problem 3	100	8	200	Infinite	0	0.91	Christofides, and Eilon, 1969
Problem 4	150	12	200	Infinite	0	0.93	Christofides, *et al.*, 1979
Problem 5	199	16	100	Infinite	0	0.98	Christofides, *et al.*, 1979
Problem 6	50	6	160	200	10	0.80	Christofides, and Eilon, 1969
Problem 7	75	11	140	160	10	0.88	Christofides, and Eilon, 1969
Problem 8	100	9	200	230	10	0.81	Christofides, and Eilon, 1969
Problem 9	150	14	200	200	10	0.80	Christofides, *et al.*, 1979
Problem 10	199	18	200	200	10	0.88	Christofides, and Eilon, 1969
Problem 11	120	7	200	Infinite	0	0.98	Christofides, *et al.*, 1979
Problem 12	100	10	200	Infinite	0	0.90	Christofides, *et al.*, 1979
Problem 13	120	11	200	720	50	0.62	Christofides, *et al.*, 1979
Problem 14	100	11	200	1040	90	0.82	Christofides, *et al.*, 1979

To compare the performances of various implementations of the ABLS algorithms, the same initial solution should be used for each ABLS algorithm. In this study, the initial solution inspired by Brandao and Mercer (1998) is used to carry out the comparison. This algorithm can be used to solve all of the above instances regardless of the customers has service time or not. In the initial solution, a vehicle is allowed to make more than one trip and the demands of some customers can be greater than the capacity of the vehicles.

In the initial solution, the customers whose demands are greater than the capacity of the vehicle are preprocessed first. Each customer in this group will be visited

$l = \left\lfloor \frac{customer's\ demand}{vehicle\ capacity} \right\rfloor$ times by the fully loaded vehicle, where $\lfloor x \rfloor$ denotes the greatest integer smaller than or equal to *x*. The demand of this customer is then reset to the remaining demand, i.e., the customer's demand minus *l* times the vehicle capacity. The routes that serve this customer will be assigned to the same vehicle if possible; otherwise, it will be assigned to other vehicles with the largest remaining time. The fully loaded routes that serve only one customer should be untouched in the following improvement procedure.

After the preprocessing of the customers with demands larger than the vehicle capacity, the node that is farthest from the depot is chosen and assigned to the vehicles with the largest remaining time to construct the current route. This is followed by the insertion of the nodes that least increase the routing time among the un-routed customers that belong to the neighborhood of each customer in the route, provided the constraints are not violated. This step is repeated until no more nodes can be inserted and then a new route is started. The procedure is repeated until all the customers are visited. The neighborhood of one customer is defined to be the set of a number of customers that are nearest to the customer.

In this method, the remaining time of each vehicle, T_r, will be used to determine which vehicle will be used. *r* is the index of vehicles. The procedure of the initial solution is described as follows:

Step 1. Initialization: set the remaining time of each vehicle, T_r.

Step 2. Serving customers with demand larger than the vehicle capacity:

Step 2.1. Select a customer *i* with demand $d_i \geq Q$. If the remaining demands of all customers are less than *Q*, go to Step 3.

Step 2.2. Choose the vehicle which served customer *i* previously and whose remaining time is enough to serve customer *i*. Otherwise, another vehicle with the largest remaining time will be chosen.

Step 2.3. Assign Q to the selected vehicle, then update the remaining time of the chosen vehicle and the remaining demand of the customer *i*. If $d_i \geq Q$, go to Step 2.2.

Step 2.4. Go to Step 2.1 to select the next customer.

Step 3. If all the customers have been served, stop; otherwise, choose a remaining customer that is farthest away from the depot.

Step 4. Choose a vehicle with the largest remaining time and assign the customer selected in Step 3 to the vehicle.

Step 5. Select the node that least increases the routing time among the un-routed customers that belong to the neighborhood of each customer in the route, and insert the selected node to the route if the constraints are not violated.

Step 6. Repeat Step 5 until no other admissible nodes can be inserted.

Step 7. Go to Step 3 and construct the next vehicle trip.

The advantage of the assignment in Step 2 is that it can ensure that the driver is able to serve a group of customers that he is familiar with. This can improve the relationship between the customers and the company, and hence improve the service level.

3.3.2 Computational Results and Analysis of Type A of the ABLS Algorithm

The basic idea of Type A of the ABLS is as follows: one node is chosen from some routes and no node is chosen from other routes. The nodes chosen are deleted from the current routes and inserted back to the routes according to the solution of the assignment problem. For one particular node, the route it will be inserted into can be different from the route that the node comes from. The cost matrix for the assignment problem can be calculated by the cheapest insertion (Rosenkrantz *et al*., 1977). In other words, the value for each cell is the least distance increment for one particular node and one particular route.

The parameters of Type A of the ABLS algorithm depend on: the probability of one node to be chosen from each route; whether an infeasible solution is allowed in the improvement procedure; whether the best improvement strategy or the first improvement strategy is used; the number of iterations, and so on.

To compare the performance of the different combinations of Type-Strategy, all the combinations have been tested. The same number of iterations is set for each algorithm. For the figures of Type A of ABLS algorithm, the *x*-value is the probability of choosing one node from each route, and *y*-value is the average total distance of ten runs.

Probability of one node being chosen from each route

To study the effect of the different probabilities of one node being chosen from each route, computational runs on test instances with probabilities ranging from zero to one are carried out. The results for Problem1, Problem2 and Problem3 are illustrated in Figure 3.4, Figure 3.5 and Figure 3.6, respectively.

As shown in Figures 3.4 to 3.6, the *x*-value is the probability of choosing one node from each route, and *y*-value is the average distance for ten runs. For the *x*-axis, the left point refers to the extreme case that no node is chosen from all the routes, that is, the initial solution is not changed. Therefore, the corresponding *y*-value is the distance for the initial solution because none of the nodes is chosen and hence no improvement is obtained. Similarly, the right node of the *x*-axis refers to the extreme case when one node is chosen from each route in the solution.

Analysis of results implemented with Strategy 1 (infeasible solution not allowed)

As shown in Figures 3.4 to 3.6, two curves are plotted: one is labeled ‘infeasible solution not allowed’, the other is labeled ‘infeasible solution allowed’. The ‘infeasible solution not allowed’ label means that only feasible solutions are allowed in the improvement procedure, while the ‘infeasible solution allowed’ label means that feasible and infeasible solutions are both allowed in the improvement procedure, but that only the best feasible solutions are recorded.

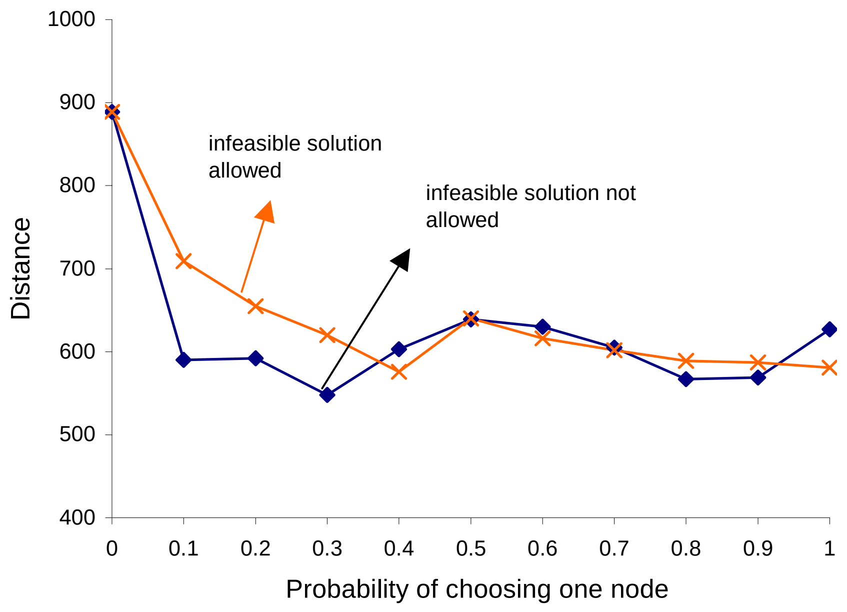


Figure 3.4 Graph of solution values against probability of choosing one node for Problem1 (Type A)

Problem 1 (*Type A, Strategy* 1)

As shown in Figure 3.4, in the curve labeled ‘infeasible solution not allowed’, the biggest distance value occurred at the left end of the *x*-axis. This is the case when none of the nodes is chosen for the improvement procedure, and hence no improvement is obtained. As the probability of choosing one node increases, the distance fluctuates within a certain range. The best solution is obtained when the probability is about 0.3.

Problem 2 (*Type A, Strategy* 1)

As shown in Figure 3.5, the ‘infeasible solution not allowed’ line is relatively flat in the range of 0.1 to 0.9 and the best value is obtained at 0.6.

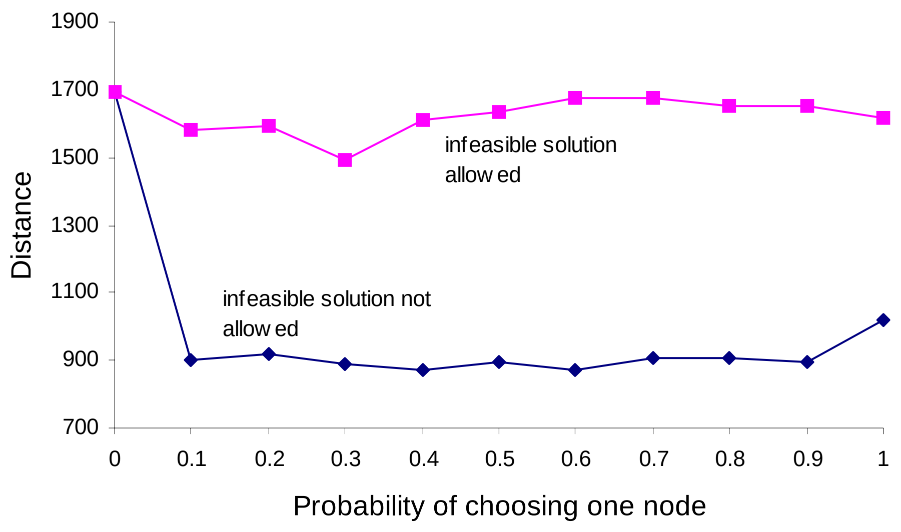


Figure 3.5 Graph of solution values against probability of choosing one node for Problem 2 (Type A)

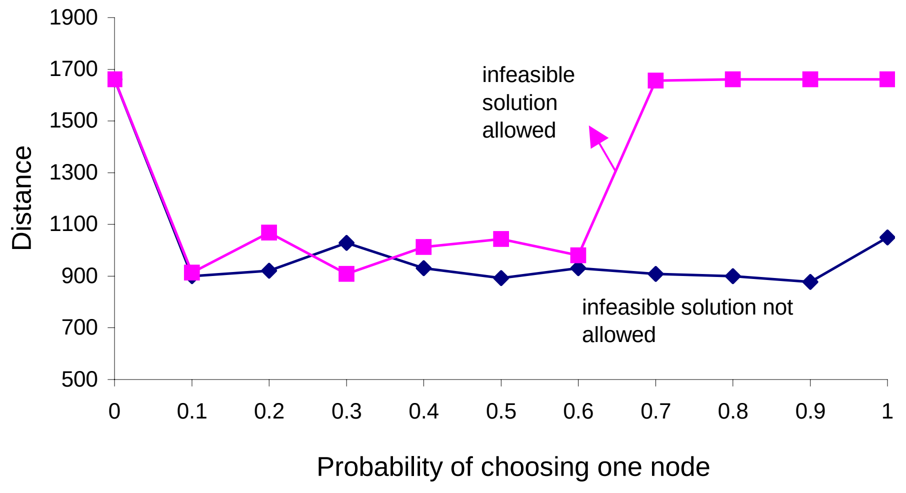


Figure 3.6 Graph of solution values against probability of choosing one node for Problem 3 (Type A)

Problem 3 (*Type A*, *Strategy* 1)

Figure 3.6 shows that the best solution can be obtained within a wide range, i.e., between 0.1 and 0.9.

According to the analysis mentioned above, in most cases, the best solution for the ABLS with Strategy 1 is obtained at a probability between 0.1 and 0.9. A case with a probability that is either too low or too high, will not result in a good solution.

Analysis of results implemented with Strategy 2 (infeasible solution allowed)

Problem 1 (*Type A*, *Strategy* 2)

As shown in Figure 3.4, the results for problem1 with Strategy 2 is similar to that of strategy 1.There is no obvious difference on the performances of these two strategies.

Problem 2 (*Type A*, *Strategy* 2)

In Figure 3.5, the performance of the algorithm in the infeasible solution allowed procedure is much worse than the case when only feasible solutions are allowed. How is the difference explained? The procedure for the case when an infeasible solution is allowed is examined first. In the procedure, it is noted that most of the solutions obtained in the improvement procedure are infeasible. Why are so many infeasible solutions produced? To get the answer, the initial solution and the source data are checked. From the analysis of the initial solution, it is found that the remaining capacities of 9 out of 10 routes are less than 10. In other words, the remaining capacities of 90% of the total routes are less than 10. However, from the source data file, it can be seen that only 9 out of 75 demands are less than 10. In other words, 88% of the demands are greater than 10. This explains why so many infeasible solutions are produced and hence the performance of Strategy 2 is much worse than that of Strategy 1 for Problem 2.

Problem 3 (*Type A, Strategy* 2)

As shown in Figure 3.6, it appears strange that when the probability is less than 0.6, there is not much difference between the two strategies. However, when the probability value is greater than 0.6, the performance of Strategy 2 becomes markedly worse. Almost no improvements can be obtained when the probability is larger than 0.6. A possible reason for this result is that, when the probability is low, the number of nodes chosen is small, and hence there is little chance for an infeasible solution. However, when the probability becomes higher, more nodes can be chosen and hence the infeasible solutions are likely to occur more frequently. Although many infeasible solutions are produced, only the best feasible solution is recorded. This results in the solution becoming worse when the probability is higher than 0.6.

Generally speaking, the performance of the algorithm with Strategy 1 (infeasible solution not allowed) is better than the algorithm with Strategy 2 (infeasible solution allowed) for the instances whose remaining capacity is relatively small; there are no much difference when the remaining capacity is large compared with the demands of each nodes.

Analysis of results implemented with Strategy 3 and Strategy 4 (best improvement and first improvement)

The ABLS algorithm is an iterative improvement method and can be modified as a neighborhood search method. The neighborhood of a given solution can be defined as the subset of all neighbors generated by the ABLS algorithm. For instance, when Type A of the ABLS algorithm is used, the neighborhood of the current solution can be defined as the set of nodes that are chosen according to the preset probability. The size of a neighborhood can be determined by experiments. For the neighborhood structure, two selection strategies were proposed by Osman (1993): BI and FI strategies. In the BI strategy, all the possible solutions in the neighborhood are checked and then the one that can produce the best solution in the neighborhood is accepted. In the FI strategy, only the first neighborhood move that can improve the current solution is accepted.

Computational experience shows that the performance of ABLS with Strategies 3 and 4 is not much better than the performance of ABLS with Strategy 1 and Strategy 2. However, the running time for ABLS with Strategies 3 and 4 is much more than the algorithm with Strategies 1 and 2, especially the ABLS with Strategy 3.

For Type A ABLS algorithm, the computational time increases with the probability of choosing one node from each route becomes bigger. For example, when the Type A1 algorithm is used to solve Problem 1, the running time is 6 seconds when the probability of choosing one node is 0.1; it increases to 32 seconds when the probability becomes 0.5; it reaches 69 when the probability is 1.

There are no much difference between the computational time for the algorithms with Strategy 1 (infeasible solution not allowed) and Strategy 2 (infeasible solution allowed). The running time for the algorithm with Strategy 4 (first improvement) is less than the time for the algorithm with Strategy 3 (best improvement).

3.3.3 Computational Results and Analysis of Type B of the ABLS Algorithm

The basic idea of Type B of the ABLS is similar to that of Type A, the only difference between them is that it is not necessary to consider where the nodes come from. In other words, one can choose more than one node from one route as long as the total number of nodes is not greater than the total number of routes.

For Type B of the ABLS algorithm, the factors which may affect the performance of the algorithm include the number of nodes chosen; whether an infeasible solution is allowed in the improvement procedure; whether the best improvement strategy or the first improvement strategy is used; the number of iterations, and so on.

Number of nodes chosen

In this study, it was observed that the most important factor to influence the behavior of Type B of the ABLS algorithm is the number of nodes chosen. If the number of nodes chosen is too small, the improvement in the solution is also marginal. On the other hand, if too many nodes are chosen, there is a higher chance of getting the infeasible solution in the improvement procedure, and hence there is little chance to get good solutions. Computational experience shows that most of the best solutions can be obtained when the number of nodes chosen is neither too large nor too small.

Analysis of results implemented with strategies 1, 2, 3 and 4

Computational experience also shows that the performance of Type B ABLS with Strategies 1 and 2 is not much different when the number of chosen nodes is relatively small. However, the performance of Strategy 2 is not good when infeasible solution is easy to produce, such as for the case that most routes' remaining capacity is relatively large compared with the demand of each node. Therefore, ABLS with Strategy 1 is more stable than with Strategy 2.

It can be noted that the 'FI' strategy gives better results than the 'BI' strategy for Type B of the ABLS algorithm. Similarly to the Type A of ABLS algorithm, the running time for the algorithm with Strategy 4 (first improvement**)** is less than the time for the algorithm with Strategy 3 (best improvement).

3.3.4 Summary of ABLS Algorithm and Composite Algorithm

There are some drawbacks for some commonly used local search methods. For example, for insertion and swap moves, only a few nodes and a few locations which can be inserted are concerned in each iteration and the rearrangement of the chosen nodes is random. To overcome the drawbacks above, the ABLS algorithm is provided.

The basic idea of the ABLS method is to exchange or move several nodes to other routes. Then try to insert back the chosen nodes by solving an assignment problem whose corresponding cost matrix is obtained by some insertion procedure. In the ABLS algorithm, more nodes and more locations are concerned for each iteration of the algorithm compared with some other local search method. Moreover, there is an optimization for each iteration, that is, the arrangement of the nodes selected is based on the solution of an assignment problem, instead of random arrangement. However, for some cases, more running time is needed because an assignment problem is solved for each iteration.

The effectiveness of the different implementations of the ABLS algorithm depends on the characteristics of the data for each problem. For instance, if the remaining capacity for most of the routes is very low in the current solution, it is more efficient to implement the Type A ABLS algorithm than the Type B ABLS algorithm.

Both the basic Type A and Type B ABLS algorithms can be implemented separately. However, in most cases, better solutions are obtained using some composite algorithms that combine some other strategies and the ABLS algorithm. For example, the ABLS method is incorporated in the smoothing method by adding the following step between Steps 3 and 4 of the smoothing method described in Chapter 2:

Step 3A. Apply the ABLS heuristic with the smoothed distance to produce the current tour.

The computational experiments show that the implementation of this type of composite procedure can improve the results effectively.

3.4 Implementation of SA to the ABLS Algorithm

Generally, local search method will lead to a local optimum and the quality of the final solution depends on the quality of the initial solution. To overcome the shortcomings of the local search method, some meta-heuristics can be used on the local search method. In this section, the details of the implementation of SA to the ABLS algorithm (ABLS&SA) are presented.

In the literature survey, two methods based on SA were used to solve the CVRP, i.e., Osman’s (1993) λ-interchange method, and Breedam’s (1995) string exchange and relocation method.

In this section, one composite method, ABLS&SA, is presented and described in detail.

The procedure of the ABLS&SA composite method can be described as follows:

Step 1. Generate an initial solution.

Step 2. Select an initial temperature T_0 and a final temperature T_f. Set the number of iterations at each temperature *Iter_T*, Reduction Factor *Red_F*, maximum number of reheat *Max_R* and the initial temperature for each reheat, T_{reh} to some predetermined values. Set *No_Iter* = 0 and *No_R* = 0.

Step 3. Choose some nodes from the current solution and delete them from the solution. Insert the nodes to the corresponding routes according to the solution of the assignment problem with the cost matrix calculated by the cheapest insertion method.

Step 4. Check the feasibility of the solution. If it is infeasible, go to Step 3.

Step 5. If the total distance of this solution is less than all the previous ones, set this as the best solution. Go to Step 7.

Step 6. Calculate the difference δ between the current distance and the previous distance. If δ < 0 or $\exp(-\delta / T)$ is less than a random value $a \in [0,1]$, set this solution as the current solution.

Step 7. Set *No_Iter* = *No_Iter*+1. If *No_Iter* < *Iter_T*, go to Step 3.

Step 8. Set $T = T \times$ *Red_F*.

Step 9. If the current temperature, T, is greater than the predetermined T_f, go to Step 3.

Step 10. Set *No_R* = *No_R* +1. If *No_R* < *Max_R*, T is set to be T_{reh} and then go to Step 3.

Step 11. Stop and return the best solution found.

A flowchart for the above procedure is presented in Figure 3.7

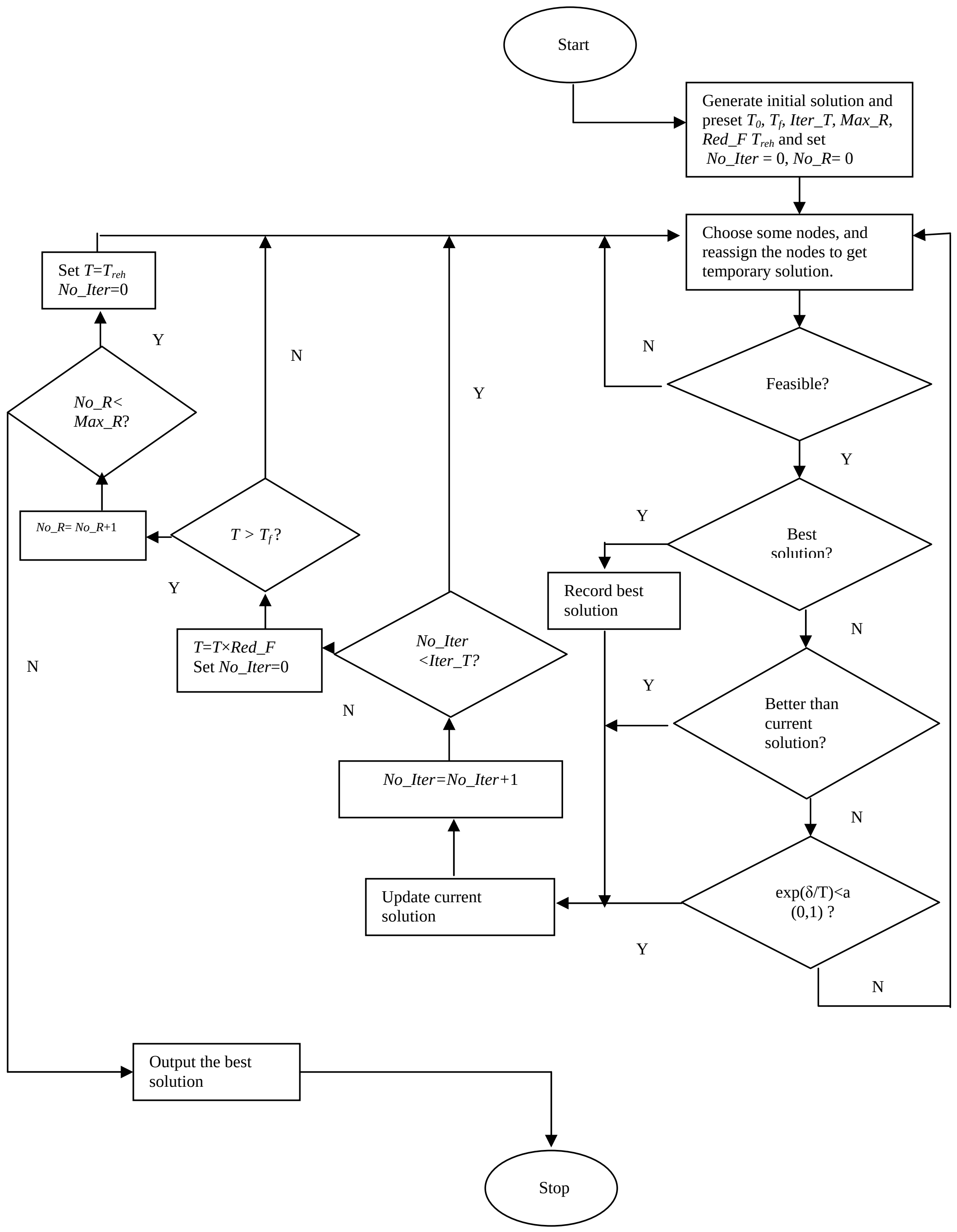

Start
Generate initial solution and preset T_0, T_f, *Iter_T*, *Max_R*, *Red_F* T_{reh} and set *No_Iter* = 0, *No_R*= 0
Choose some nodes, and reassign the nodes to get temporary solution.
Feasible?
Best solution?
Record best solution
Better than current solution?
exp(δ/T)<a (0,1) ?
Update current solution
No_Iter=*No_Iter*+1
No_Iter <*Iter_T*?
T=*T*×*Red_F* Set *No_Iter*=0
$T > T_f$?
No_R= *No_R*+1
No_R< *Max_R*?
Set T=T_{reh} *No_Iter*=0
Output the best solution
Stop
Y
N

Figure 3.7 Flowchart for the composite ABLS&SA procedure

In this study, the following parameters are used in the computational runs for the ABLS&SA composite algorithm.

Initial temperature $T_0 = 300$.

Cooling rate, *Red_F*, is in the range from 0.8 to 0.9.

Final temperature $T_f = 0.001$.

Maximum number of reheat, *Max_R*, = 3.

Iterations for each temperature: *Iter_T* = 4000

Table 3.6 Computational results of the ABLS(Type A)&SA and ABLS(Type B)&SA composite procedures

Problem	ABLS(Type A)&SA		ABLS(Type B)&SA	
	Value	Time[1]	Value	Time[1]
Problem 1	544	79	524	54
Problem 2	866	226	841	278
Problem 3	872	357	829	562
Problem 4	1120	616	1042	863
Problem 5	1490	1413	1429	996
Problem 6	555	170	555	189
Problem 7	943	512	914	443
Problem 8	892	283	865	373
Problem 9	1255	572	1169	641
Problem 10	1485	744	1414	1533
Problem 11	1202	904	1115	1187
Problem 12	912	489	819	613
Problem 13	1558	920	1556	264
Problem 14	866	160	866	483

1. Computation time in seconds based on a 2.1GHz Pentium IV computer

As shown in Table 3.6, the performance of Type B is much better than Type A for the SA based ABLS algorithm.

To compare the performance of the proposed SA based ABLS method, the algorithm has been applied to 14 CVRP instances. The results are compared against the Osman's SA based algorithm (OSA) and The Breedam's SA based algorithm (BSA). The computational results of these algorithms applied to 14 problem instances are summarized in Table 3.7.

Table 3.7 Comparison the ABLS&SA composite procedure with OSA and BSA methods

Problem	OSA	BSA	ABLS(Type B)&SA
Problem 1	528	**521**	524
Problem 2	**838**	841	841
Problem 3	**829**	830	**829**
Problem 4	1058	1063	**1042**
Problem 5	1378	**1360**	1429
Problem 6	555	**548**	555
Problem 7	**909**	920	914
Problem 8	866	870	**865**
Problem 9	**1164**	1197	1169
Problem 10	1417	1462	**1414**
Problem 11	1176	**1042**	1115
Problem 12	826	821	**819**
Problem 13	**1545**	1568	1556
Problem 14	890	867	**866**

From Table 3.7, OSA can result in 5 best solutions in the 14 problems, BSA finds 4 best solutions and ABLS finds 6 best solutions of the 14 problems. The results show that the proposed ABLS&SA composite procedure is able to match the performances of the other two SA implementations on most of the problem instances tested. For some of the problem instances tested, the ABLS&SA composite procedure is able to match the performance of OSA and BSA.

A further comparative study of the ABLS method against some other meta-heuristics is given and discussed in Chapter 4.

3.5 Conclusions and Some Possible Applications of ABLS Methods

In this chapter, we propose an ABLS method for solving the VRPs. The ABLS is an iterative algorithm. Given any feasible initial solution, the algorithms choose a certain number of nodes and delete them from the current routes and then try to insert them back according to the solution of an assignment problem. The ABLS can be used as a multi-route improvement algorithm that can operate on several routes at a time. Therefore, it can be used to solve the CVRP and some of its variants.

ABLS may also be used to solve some other types of problems. One possible application is to solve the CVRP consisting of various clusters of customers. In a CVRP, if some customers are distributed in various clusters, the problem can be simplified by treating the customers in each cluster as a big customer. The demand of the big customer is the sum of the demands of customers in each cluster. However, the distance inside the 'big' node must be considered when calculating the distance of the routes.

An ABLS algorithm may also be applied to solve the TSP. The basic procedure of applying the ABLS algorithm to the TSP is as follows: choose some nodes and delete them from the current tour. Then calculate the insertion cost for each chosen node and each edge in the tour after deletion to obtain the cost matrix. Reassign the chosen nodes to the edges according to the solution of the assignment problem with regard to the cost matrix. To maintain the feasibility of the assignment problem, the number of nodes should be less than or equal to half of the total number of nodes. However, for Type B of the algorithm, there are not all the solution space will be considered, because the deleted nodes cannot be inserted as a neighbor to each other.

Chapter 4

Generalized Crossing Method

In this chapter, we present a multi-route improvement local search called generalized crossing (GC) method for solving the VRPs. The generalized crossing mechanism, the local search method and the implementation with simulated annealing are described in detail in the subsequent sections of this chapter.

4.1 Introduction to GC Method

"String Cross" is an improvement method commonly used in local search procedure for solving VRPs. Among the various multi-route improvement methods proposed in the literature, two of them use string cross as a local search method. The first one was proposed by Breedam in 1994. The other string cross was proposed by Kinderwater and Savelbergh (1997). Although this idea is simple and easy to implement, the computational experiments show that it is not easy to produce good solutions. Based on the idea of string crossing, a generalized crossing method is proposed and tested.

The generalized crossing method is a generalization of the normal string crossover operator proposed by Kinderwater and Savelbergh (1997). Their string crossing is conducted by crossing two edges of two different routes. For example, for the two following original routes shown in Figure 4.1, the two routes are cut into two parts. Route 1 is cut into two parts at the edge (i, i+1) and generates two strings: String 1 and String 2. Similarly, Route 2 is cut into String 3 and String 4 at edge (j, j+1). In String 1, i_s is the first node, and i is the last one. Similarly, i+1, j_s, j+1 are the first nodes, and i_f, j and j_f are the last nodes of String 2, String 3 and String 4 respectively. The first node and the last node can also refer to same node. In this case, there is only one node in the string.

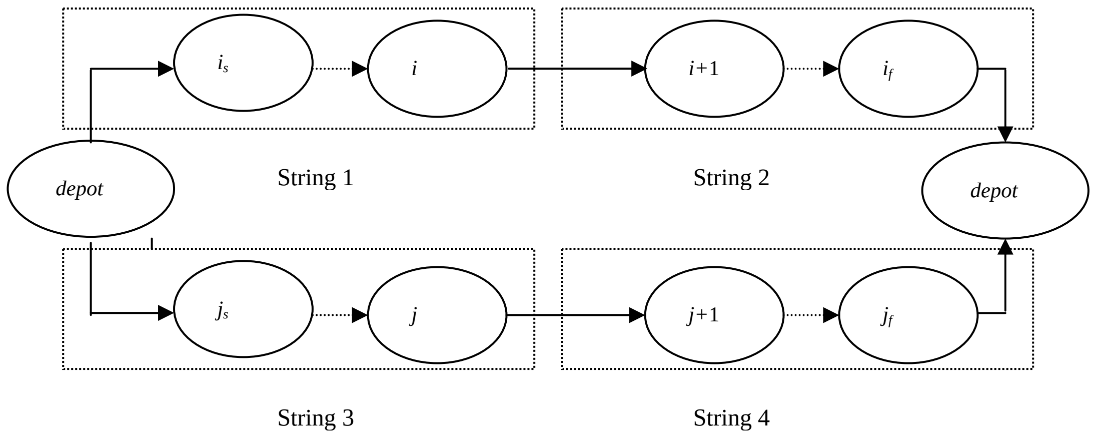


Figure 4.1 Original routes of the example

According to the cross operator proposed by Kinderwater and Savelbergh (1997), node *i* will be connected with node *j*+1, and node *j* will connect with node *i*+1. Then the routes will become:

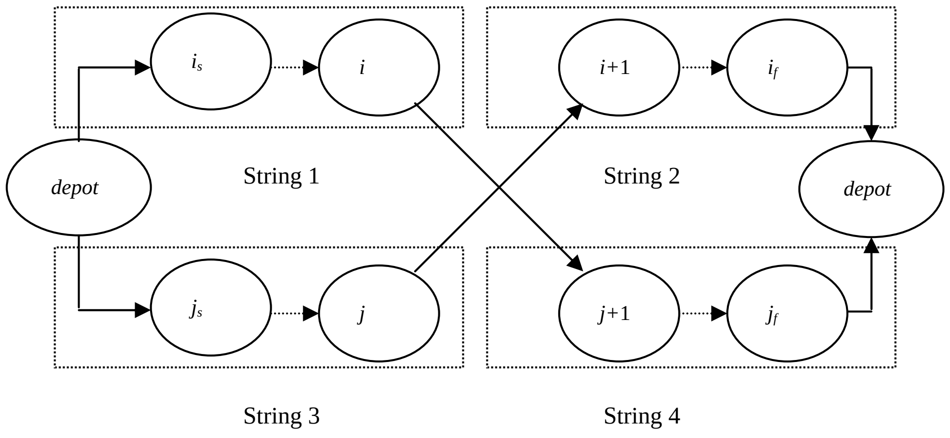


Figure 4.2 Routes after “String cross” of the example

The generalized crossing algorithm proposed in this study is an extension of the string crossing method Kinderwater and Savelbergh (1997), in which more combination of the strings and the order of each string are considered. In the GC method, the new routes are constructed not only by combining the strings in their original direction but also combining the strings with opposite direction in the GC method. For example, for the 4 strings in Figure 4.1, there are three possible combinations to generate new routes:

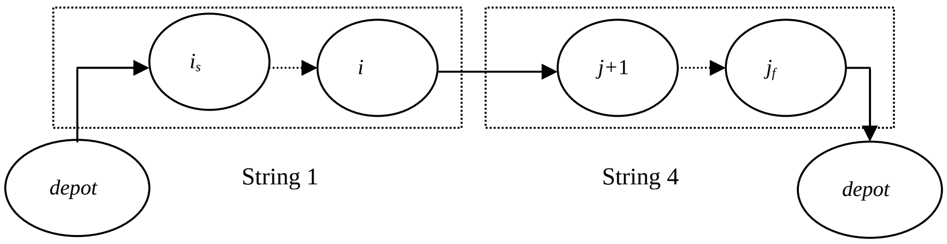
$$\text{Combination 1:} \begin{cases} \text{Route 1 : String 1 and String 2} \\ \text{Route 2 : String 3 and String 4} \end{cases}$$

$$\text{Combination 2:} \begin{cases} \text{Route 1 : String 1 and String 3} \\ \text{Route 2 : String 2 and String 4} \end{cases}$$

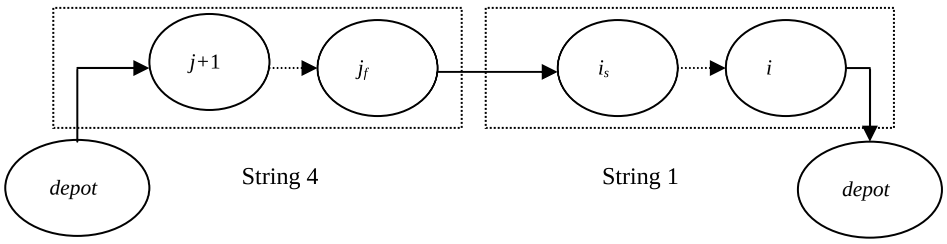
$$\text{Combination 3:} \begin{cases} \text{Route 1 : String 1 and String 4} \\ \text{Route 2 : String 2 and String 3} \end{cases}$$

For each combination of two strings, 4 possible routes can be generated depend on the order of the two strings. For example, for Route 1 of Combination 3, node *i* can connect with *j*+1, but it can also connect with j_f. Then the 4 possible routes are as follows:

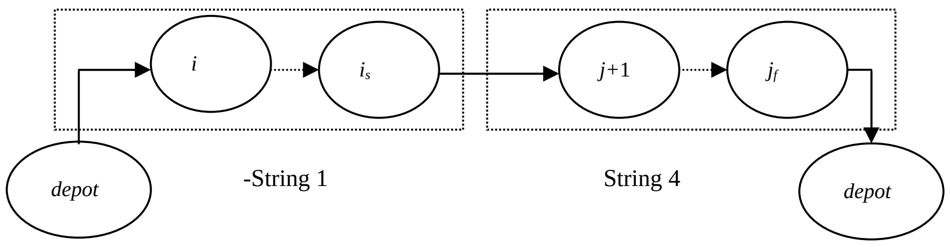
Possible route 1: String 1 -> String 4



Possible route 2: String 4 -> String 1



Possible route 3: -String 1 -> String 4



where “-String1” denote the string of nodes in the opposite direction of String 1.

Possible route 4: String 1 -> -String 4

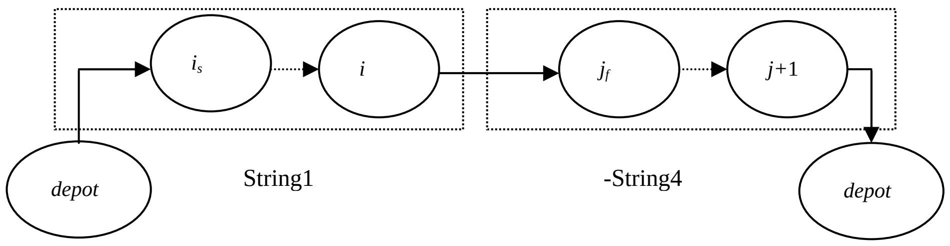


Figure 4.3 Possible routes for Route 1 of Combination 3 of the example

Therefore, there are 16 possible cases for one combination and up to 48 possible solutions can be generated from the original two routes shown in Figure 4.1.

Some swap procedure can also be used in the procedure to enlarge the neighborhood of the current solution. For example, swap node i and node j to get new solution. Our computational experiments show that this can improve the performance of GC algorithm further.

4.2 GC Local Search Method

In this study, we use the most widely known heuristic method, Clarke and Wright method (1964) to construct the initial solution. In the Clarke and Wright algorithm, calculate the savings and rank them from the largest to the smallest to form a larger subtour until a tour is formed.

In this section, some descent implementations of this local search method are considered. It is a systematic search method. The pseudo-code of the first improvement local search method is as follows:

First improvement local search procedure

```
begin
    STARTLOOP
    for Route1:=1 to No_Route do
        for i1:=1 to #(Route1) do
            for Route2 := 1 to No_Route and Route 1≠Route 2 do
                for i2:=1 to #(Route2) do
                    begin
                        $R_{new}$=move ()
                        if cost($R_{new}$)<cost ($R$)
                            begin
                                $R= R_{new}$
                                go to STARTLOOP
                            end
                    end
                next
            next
        next
    next
end
```

The above method is a first improvement method, that is, the systematic search will start from the beginning if an improved solution is found. It is noted that the solution R_{new} is the solution that is obtained by carrying out the best move among the 48 combination. Similarly, Osman (1993)'s best improve strategy can also be implemented. In this case, the pseudo-code is as follows:

Best improvement local search procedure

```
begin
    STARTLOOP
    for Route1:=1 to No_Route do (Loop 4)
        for i1:=1 to #(Route1) do (Loop 3)
            for Route2 := 1 to No_Route and Route 1≠Route 2 do (Loop 2)
```

```
                        for i2:=1 to #(Route2) do (Loop 1)
                            begin
                                $R_{new}$=move ()
                                if cost($R_{new}$)<cost ($R_{best}$)
                                begin
                                    $R_{best}$= $R_{new}$
                                end
                            end
                            Location 1
                        next (end of Loop 1)
                        Location 2
                    next (end of Loop 2)
                    Location 3
                next (end of Loop 3)
                Location 4
    next (end of Loop 4)
    Location 5
    begin
      if cost($R_{best}$)<cost (R)
      begin
         R= $R_{best}$
         go to STARTLOOP
      end
    end
end
```

It is noted that there are 4 loops in the procedure. The first improvement and best improvement are two special cases. In the first improvement procedure, if a better solution is found, the best solution is updated in the Loop 1. That is, if any improved solution is found, it is updated immediately. In the best improvement solution case, the best solution is updated after running Loop 4. That is, the best solution found in the 4 loops is chosen and the current solution is then updated to this best solution. Beside the above two special cases, we can also consider other cases. That is, the following update procedure:

Update procedure

```
    begin
      if cost($R_{best}$)<cost (R)
      begin
```

```
            R= $R_{best}$
            go to STARTLOOP
        **end**
    **end**
```

can be implemented at ***Location* 2, *Location* 3** or ***Location* 4.** We call such procedure as middle improvement procedure. To test the effectiveness of middle improvement strategy, the three middle improvement procedures, the first improvement and best improvement procedure are run for the 14 test instances and the computational results are given in Table 4.1.

Table 4.1 Computational results for the middle improvement strategies for GC local search method

Problem	Initial solution	First improvement	Middle improvement (Location 2)	Middle improvement (Location 3)	Middle improvement (Location 4)	Best improvement
Problem 1	584	572	572	572	572	572
Problem 2	900	886	886	886	886	886
Problem 3	889	885	885	885	885	885
Problem 4	1140	1135	1135	1135	1135	1135
Problem 5	1395	1393	1393	1393	1393	1393
Problem 6	618	618	618	618	618	618
Problem 7	975	972	972	972	972	972
Problem 8	973	973	973	973	973	973
Problem 9	1287	1286	1286	1286	1286	1286
Problem 10	1538	1535	1535	1535	1535	1535
Problem 11	1068	1068	1068	1068	1068	1068
Problem 12	833	829	829	829	830	830
Problem 13	1592	1591	1589	1589	1585	1585
Problem 14	875	873	873	873	873	873

All the tests are running on a Pentium IV(2.1G Hz) computer and the running time for each instance is less than 0.02 seconds. The computational results show that the local search methods are not sensitive to improvement procedure being implemented; there is no much difference on either the solution in quality or computational time taken by for the five improvement methods.

4.3 SA Based GC Method

It is possible that the search may be trapped at local optima in local search. To improve it further, some meta-heuristics can be applied to this method. For example, the simulated annealing can escape poor quality local optimum effectively because it has different randomized search and can accept non-improved solution. In this section, the implementation of SA to the GC is discussed.

The pseudo-code of the implementation of SA to the GC algorithm (GC&SA) is as follows.

GC&SA procedure

```
begin
Set the initial values to the parameters
repeat
    repeat
        begin
            $R_{best}$=R
            No_Iter = No_Iter+1
            for Route1:=1 to No_Route do (Loop 4)
                Initialize $R_{best\_move}$ to be a very poor solution
                for i1:=1 to #(Route1) do (Loop 3)
                    for Route 2:=1 to No_Route and Route 1≠Route 2 do (Loop 2)
                        for i2:=1 to #(Route2) do (Loop 1)
                            begin
                                $R_{new}$=move ()
                                if cost($R_{new}$)<cost ($R_{best\_move}$)
                                    begin
                                        $R_{best\_move}$= $R_{new}$
                                    end
                            end
                            Location 1
                        next (end of Loop 1)
                        Location 2
                    next (end of Loop 2)
                    Location 3
                next (end of Loop 3)
                Location 4
            next (end of Loop 4)
            Location5
            begin
                $\delta$=cost($R_{best\_move}$)-cost (R)
```

```
                    **if** ($\delta$<0)
                        **begin**
                            R= $R_{best\_move}$
                            Update tabu list
                            $i_1$= $M$; $i_2$=$M$; $Route2$=$M$
                        **end**
                        **if** cost(R) <cost($R_{best}$) $R_{best}$=R
                        **else if**( $\exp(-\delta/T)$ < random number $a \in [0,1]$ ) **then**
                            **begin**
                                R= $R_{best\_move}$
                                Update tabu list
                                $i_1$= $M$; $i_2$=$M$; $Route2$=$M$
                            **end**
                **end**
            **end**
        **until** ($No\_Iter$ >$Iter\_T$ )
        $T = T * Red\_F$
        $No\_Iter=0$
**until**($T < T_f$)
**end**
```

It is noted that the solution R_{new} is the solution that is obtained by carrying out the best move among the 47 combinations to prevent returning to the original solution. The combination 1 with the original direction for the four strings 1 to 4 is discarded.

The value *M* in the above procedure denotes a very large positive number. It is noted that the best improvement strategy is implemented in the above procedure.

Similarly, we can also employ other improvement strategies by implementing the solution update procedure at ***Location*** **1,** ***2, 3*** or **4**.

In the tabu list, we record Route 1, Route 2 and the locations in the routes of crossing. The tabu list is used to prevent the solutions reversing in following several iterations. Therefore, it is possible that the current solution is worse than the previous one. The best solutions are recorded throughout the whole search procedure,

The computational experience shows that the tabu list size can be set based on the following formula:

Tabu_List_Size=3 × *No_Route* + Random (-*No_Route* /2, *No_Route* /2), where Random (-*No_Route* /2, *No_Route* /2) is the random integer value between -*No_Route* /2 and *No_Route* /2.

To compare the Middle improvement strategies for the SA based GC method, comprehensive computational experiments are carried out and the 14 instances are tested and summarized in Table 4.2.

The parameter setting for the SA algorithm is given below:

Parameter setting

Initial temperature $T_0 = 15$.

Cooling rate *Red_F* = 0.85

Final temperature $T_f = 0.001$.

Maximum number of reheat, *Max_R*, = 3.

Iterations for each temperature: *Iter_T* = 4000.

Table 4.2 Computational results for the middle improvement strategies for the GC&SA method

Problem	Best improvement (Location 5)	Middle improvement (Location 4)	Middle improvement (Location 3)	Middle improvement (Location 2)	First improvement (Location 1)
Problem1	527(302)	524(237)	524(171)	524(113)	524(96)
Problem2	840(766)	835(572)	835(463)	835(276)	848(276)
Problem3	836(1286)	826(1019)	827(508)	827(305)	829(211)
Problem4	1049(2962)	1033(2271)	1042(926)	1045(472)	1046(362)
Problem5	1316(4838)	1308(5171)	1325(1737)	1320(863)	1318(632)
Problem6	561(454)	555(469)	560(224)	560(150)	558(150)
Problem7	910(1088)	909(1092)	913(688)	911(443)	912(459)
Problem8	875(1883)	865(1895)	865(803)	865(543)	874(518)
Problem9	1176(3999)	1169(3990)	1204(1322)	1205(788)	1201(822)
Problem10	1416(6674)	1409(6248)	1429(2251)	1419(1412)	1423(1424)
Problem11	1042(1582)	1042(1557)	1046(571)	1046(309)	1046(185)
Problem12	819(1770)	819(1748)	819(895)	821(524)	819(347)
Problem13	1543(3049)	1545(3019)	1567(1401)	1567(894)	1567(767)
Problem14	866(1987)	866(1936)	867(1056)	867(641)	867(475)

In Table 4.2, the first number in each cell is the total distance and the number in bracket indicates the computation time in second taken by a Pentium IV (2.1G Hz) computer.

The computational results show that the running time of first improvement method is the fastest method, the next one is the middle improvement at ***Location* 2**, the third one is the middle improvement at ***Location* 3** strategy. The best improvement is most time-consuming one. In terms of solution quality, the middle improvement at ***Location* 4** outperforms the other 4 methods.

4.4 Computational Results and Comparison

To evaluate the performance of the proposed SA based GC method (GC&SA), the algorithm has been applied to solve the 14 CVRP instances. The results and comparison with Osman's (1993) SA based λ -interchange method, and Breedam's (1995) SA based string exchange and string relocation method are summarized in Table 4.3.

Table 4.3 Computational results of GC&SA and the Osman and Breedam methods

Problem	OSA	BSA	GC&SA
Problem 1	528	**521**	524
Problem 2	838	841	**835**
Problem 3	829	830	**826**
Problem 4	1058	1063	**1033**
Problem 5	1378	1360	**1308**
Problem 6	555	**548**	555
Problem 7	**909**	920	**909**
Problem 8	866	870	**865**
Problem 9	**1164**	1197	1169
Problem 10	1417	1462	**1409**
Problem 11	1176	**1042**	**1042**
Problem 12	826	821	**819**
Problem 13	**1545**	1568	**1545**
Problem 14	890	867	**866**

Table 4.3 shows that the GC&SA is able to find 11 SA best solutions out of the 14 instances tested. Therefore, the performance of the GC&SA is comparable with or better than the other two SA methods.

To test the performance of the GC&SA further, comprehensive computational experiment is carried out to compare the performance of the GC&SA method with several good VRP heuristics in the literature. OTS is a tabu search method with interchange mechanism proposed by Osman in 1993. TTS (Tailand, 1993) is another tabu search method in which interchange mechanism is used. Both PF (Pureza and França, 1991) and GHL (Gendreau *et al*., 1994) implemented the mechanism similar to the string relocation and string exchange to the tabu search.

In the comparison, Osman and Salhi (1997)'s measures are used:

1) Relative average percentage deviation over the best solution is defined by

   RAPD= $\sum[(\text{Solution - Best solution known})/\text{Best solution known}]$ /number of test problems × 100.

2) Number of best solutions found against the instances tested (NBSF).

The comparison of GC&SA with some other heuristics is summarized as follows:

Table 4.4 Comparison GC&SA with other meta-heuristic methods

Problem	Best Solution Known	OSA	OTS	TTS	PF	GHL	BSA	ABLS&SA	GC&SA
Problem 1	521	528	524	524	536	524	521	524	524
Problem 2	835	838	844	835	842	835	841	841	835
Problem 3	826	829	835	826	851	826	830	829	826
Problem 4	1028	1058	1044	1028	1081	1031	1063	1042	1033
Problem 5	1291	1378	1334	1298	------	1311	1360	1429	1308
Problem 6	548	555	555	555	560	555	548	555	555
Problem 7	909	909	911	909	929	909	920	914	909
Problem 8	865	866	866	865	887	865	870	865	865
Problem 9	1162	1164	1184	1162	1227	1162	1197	1169	1169
Problem 10	1395	1417	1422	1397	------	1404	1462	1414	1409
Problem 11	1042	1176	1042	1042	1049	1042	1042	1115	1042
Problem 12	819	826	819	819	826	819	821	819	819
Problem 13	1541	1545	1545	1541	1631	1545	1568	1556	1545
Problem 14	866	890	866	866	866	866	867	866	866
RAPD		2.26	0.95	0.18	2.65	0.33	1.55	1.82	0.40
NBSF		1	3	10	1	8	3	3	7

As shown in Table 4.4, ABLS&SA outperforms OSA and BSA when RAPD is considered. When NBSF is concerned, ABLS&SA can match the performance of

OSA, OTS, PF, BSA. GC&SA is ranked third among the 8 algorithms compared when the either RAPD or NBSF is concerned.

4.5 Conclusions

In this chapter, a local search method called generalized crossing method, is proposed. The generalized crossing is a generalization of the normal string cross operator. In this method, new routes are constructed not only by combining the strings in their original direction but also combining the strings with opposite direction. Several improvement strategies have also been proposed and incorporated into the proposed GC method. Computational result show that the SA implementation of the GC method combined with the middle improvement procedure outperforms some other SA implementations and is comparable with some other meta-heuristic methods, such as tabu search method.

Chapter 5

An Application Study of Proposed Methods

In recent years, it can be seen that an increasing number of companies are paying more attention to their core activities and are contracting their product delivery activities to some third-party logistics (3PL) companies. This has led to the rapid growth of 3PLs. In this chapter, an application study of the proposed algorithms to a company to solve its soft drinks distribution problem is presented.

A description of the soft drinks distribution company (SDC) problem is given in Section 5.1. An overview of some procedures for solving the problem is presented in Section 5.2. In Section 5.3, a procedure for solving the problem is proposed. The computational results for the proposed method are discussed in Section 5.4. Finally, some concluding remarks are given in Section 5.5.

5.1 Description of the SDC Problem

The problem addressed in this application study arises from a transportation company providing product delivery services for some manufacturing companies in Singapore. The company has a fleet of vehicles with different capacities to serve a pre-specified set of customers with known delivery demands. Each vehicle starts from the depot, and after serving some customers, it is allowed to return to the depot for replenishment. This would allow the vehicle to run several trips in a day as long as the total time taken is within a certain number of hours. The time taken by each vehicle includes travel time, loading and unloading times, as well as the time for the vehicle driver’s meal break.

A key requirement of the SDC is that vehicles are to be assigned to serve a fixed set of districts. Currently, customers are aggregated into districts based on the postal code classification. The purpose of this requirement is to allow each driver to know his district well enough and to establish a good relationship with the customers assigned to him. Furthermore, fixed assignment of vehicles to serve customers within each district will make it easier for the company to keep track of each vehicle's delivery activities. The main objective of the SDC is to determine a minimum vehicle fleet size and a schedule of vehicles that would be able to serve its customers on a daily basis.

5.2 Procedures for Solving the SDC VRP

In this section, two procedures to solve the SDC VRP in the literature are described. The first procedure is the approach used by the SDC while the second procedure is an improved approach proposed by Cheong *et al.* (2002).

In the SDC approach, the customers are aggregated into districts based on postal codes. The highest customer demand that occurs in each district obtained from historical data is used to estimate the demand for that district to determine the fleet size. Based on these estimates, an approximate fleet of vehicles is assigned to serve the customers in various districts. As there are quite a number of customers in each district and their demands change from day to day, the assignments made on the previous day might not be applicable for the following day. To overcome the dynamic changes in customers and their demands, the SDC provides a small number of standby vehicles to serve the customers that cannot be served on a daily basis by the fixed fleet of vehicles.

The SDC problem can be formulated as a fleet size and mixed vehicle routing problem (FSMVRP) in the literature. Cheong *et al.* (2002) re-examined the fixed assignment used by the SDC and proposed a way to model and solve it. It is a two-phase method.

Phase I is used to determine the vehicle fleet size and their assignments to districts on a long-term basis. Phase II is used to solve the VRP related to individual customers in each district on a daily basis.

The core of Phase I is to formulate the problem as a set-covering type of problem. All feasible patterns and routes are generated as columns and then solved by the column generation algorithm. The procedures used to generate the solutions of Phase I are contained in four steps. The first step is to merge districts. As mentioned earlier, each district is treated as one customer, with the demand of each district varying from day to day. Therefore, to simplify the problem further and reduce the variance of demands estimated, the districts with coefficients of variations that are higher than a certain limit can be merged with other districts in order to reduce the coefficient of variation. The districts to be merged are those close to each other, i.e., the distance between two centers of districts is less than 3 km. The second step is to estimate the demand for each district. The last two steps are to generate all the feasible routes for each vehicle and then obtain the solution by the column generation algorithm.

Based on the results of Phase I, Phase II is used to solve the daily VRP. The purpose of Phase II is to reroute some helper vehicles to serve problem districts. The demand for each district varies from day to day, and as a result, some vehicles assigned to a certain district cannot serve all the customers on certain days or some vehicles are underutilized on certain days. The districts where all customers cannot be served are called problem districts. To solve such problems, Phase II is used to reroute these vehicles to satisfy all the customers. In this phase, three assignment procedures were presented by Cheong *et al*. (2002). These three procedures are independent from one another and each can be used separately. The basic idea of these three procedures is to identify the helper vehicles and problem districts first, and then try to utilize the helper vehicles to serve the districts that need help. The helper vehicles are those vehicles whose spare time is greater than a certain value. Each of these three assignment procedures consists of three stages: (1) Use the helper vehicle assignment procedure to reassign any problem districts to helper vehicles; (2) If this help procedure cannot serve all of them, spare vehicles will be used until all of them are reassigned; (3) Finally, the possibility of releasing any underutilized vehicles from the fleet is checked. In this context, the problem districts refer to those districts whose spare time is less than 0.

The first assignment procedure (M1) selects the helper vehicle by calculating additional time. One disadvantage is that it merely guarantees that the problem district will be served with the least amount of additional time incurred, but it provides no information on whether the helper vehicle has enough time to serve the problem district.

To overcome the shortcomings of the first method, the second method (M2) was proposed. The second method’s assignment strategy is based on P_k which measures the proportion of spare time spent by vehicle k in order to serve the problem district. If P_k is less than or equal to 1, the vehicle k can serve the problem district completely, so that no other helper vehicles are needed.

In the third method (M3), helper vehicles are assigned to problem districts by computing opportunity cost. Through the use of opportunity cost, priority is given to certain helper vehicles to serve the problem districts. Some problem districts are better served by only certain helper vehicles because of their capacity and interactions between the locations of surrounding problem districts and helper vehicles.

5.3 Description of a Bin Packing Composite Method

The computational results show that the three methods proposed by Cheong *et al*. can improve the SDC method effectively. However, the two-phase procedure is complicated, especially for daily operations. To simplify the daily operations and improve the solution further, a bin packing composite method is proposed.

Bin packing composite method

In the bin packing composite method, two bin packing problems are implemented in the procedure of the composite algorithm, one at the start, and the other at the end of the procedure. The first bin packing problem is solved when the initial solution is formed. For the first bin packing problem, only the demand constraint is considered, while, the maximum time constraint is ignored. Therefore, the initial solution may not be feasible as it is possible that the maximum time constraint is violated. Thus, the initial solution is improved by using the proposed methods. If a certain route of the current solution is not feasible, i.e., the total time needed for the customers is longer than the time constraint, this route will be cut into several parts in such a way that each of them satisfies the time constraint. The current solution will be improved again if necessary before the second bin packing problem is used. In the second bin packing problem, only the time needed to serve the customers for each route will be considered. The number of vehicles to be used can be determined by solving this bin packing problem.

The bin packing problem approach for solving the VRP has also been used by other researchers in the literature. For instance, Fleischmann (1990) used the bin packing problem in the final part of the algorithm for the VRP with multiple trips. Taillard *et al.* (1996) also applied the bin packing problem to their algorithm.

Procedure for the bin packing composite method

Step 1. Determine the number of routes by solving it as a bin packing problem.

- Step 1.1. For the customers whose demands are greater than the vehicle capacity, assign routes to the customers until their remaining capacity is less than the vehicle capacity. Record the number of vehicles with full capacity.
- Step 1.2. Assign the customers to the routes by solving the bin packing problem.

Step 2. Improve the initial routes obtained in Step 1 by using the proposed algorithm. In this improved procedure, the feasibility of the capacity must be maintained for each route.

Step 3. Check the feasibility of the distance for each route.

- Step 3.1. Calculate the total time taken for each route.
- Step 3.2. For those routes which violate the time constraint, cut them into several parts by means of some route cutting procedures.

Step 4. Repeat Steps 2 and 3 until the solution is feasible and no further improvement can be made.

Step 5. Assign these routes to the vehicles by solving the bin packing problem to determine the number of vehicles needed. In this bin packing procedure, only the time needed to serve the customers assigned to each route is considered.

In Step 2, any proposed procedures can be used. Because bin packing problem is a NP-hard problem, some simple and effective heuristic methods, such as First-fit decreasing and Best-fit decreasing methods (Coffman *et al*., 1995), are recommended to be used in the bin packing composite method. In the two heuristic methods, the items are sorted first and then are placed to the bins according some rules. In First-fit decreasing method, each item is placed in the lowest indexed bin which has enough capacity to contain it. In the Best-fit decreasing method, each item is placed in the bin with largest current content but can contain it.

Route cutting procedure

In Step 3.2 of the bin packing composite procedure, some routes are still longer than the maximum time constraint. In this case, these long routes will have to be cut into several parts. Let (0, n_1, n_2, …, n_r, 0) be a long route to be cut into several parts. The cutting procedure can be described as follows:

Step 1. Start from a current route (0, n_1, 0) consisting of the first node only.

Step 2. Check whether the next node, n_j, on the long route can be added to the current route without violating the time limit constraint. If not, go to Step 3; otherwise, add node n_j to the current route and go to Step 4.

Step 3. Cut the long route at the point between node n_{j-1} and n_j. Start a new current route (0, n_j, 0) consisting of the node n_j only. Go to Step 2.

Step 4. Add node n_j to the current route. If n_j is the last node on the long route, stop; otherwise go to Step 2 to examine the next node.

An illustration of the bin packing composite method

A simple example is used to illustrate the application of the bin packing composite method in which ABLS is used as the improvement method. Assume that there are 10 customers in the example. All vehicles are identical and each vehicle’s capacity is 10. The maximum time constraint for each vehicle is 8 hours. The average speed of each vehicle is 40 km per hour, i.e., the maximum distance each vehicle can travel is $8\times40 = 320$ km. The coordinates and demands for each of the 10 customers are given in Table 5.1.

Table 5.1 Demands and coordinates of customers for the bin packing composite example

No. of Customer	*x*-coordinate	*y*-coordinate	Demand
0	0	0	0
1	9	16	5
2	40	2	8
3	73	-53	3
4	62	75	5
5	5	2	2
6	94	-29	7
7	119	-7	6
8	139	-49	1
9	-90	95	6
10	-97	87	3

In Step 1, the following routes are constructed:

Route 1: 0-1-4-0, Remaining capacity is 0 and distance is 194.9.

Route 2: 0-7-0, Remaining capacity is 4 and distance is 238.4.

Route 3: 0-9-3-8-0, Remaining capacity is 0 and distance is 564.5.

Route 4: 0-6-10-0; Remaining capacity is 0 and distance is 452.1.

Route 5: 0-2-5-0; Remaining capacity is 0 and distance is 80.4.

In Step 2, the routes constructed in Step 1 are improved by the ABLS algorithm. The 5 routes formed after the improvement are given below:

Route 1: 0-1-4-0, Remaining capacity is 0 and distance is 194.9.

Route 2: 0-3-8-7-0, Remaining capacity is 0 and distance is 322.0.

Route 3: 0-10-9-0, Remaining capacity is 1 and distance is 271.7.

Route 4: 0-6-0, Remaining capacity is 3 and distance is 196.7.

Route 5: 0-2-5-0, Remaining capacity is 0 and distance is 80.4.

The distance of Route 2 is greater than the maximum distance 320. Therefore it should be cut by the cutting procedure in Step 3 into the following parts:

Route 2a: 0-3-8-0, Remaining capacity is 6 and distance is 303.7.

Route 2b: 0-7-0, Remaining capacity is 4 and distance is 238.4.

Next, the number of vehicles to be used is determined by solving the bin packing problem. There are 6 items in the bin packing problem, and their weights are: 194.9, 303.7, 238.4, 271.7, 196.7 and 80.4, respectively. The size of each bin is 320. The solution of the bin packing problem shows that 5 vehicles are needed to serve all customers.

5.4 Computational Results and Analysis

To evaluate the performance of the proposed bin packing method, 14 problem instances of customer demand data collected by the SDC over a certain period are used in the computational study. For each of the problem instances, the information on customer demands as well as their locations in (x, y) coordinates are obtained from the SDC. Approximations of the distances traveled are based on the Euclidean distance. In addition, the vehicles are assumed to be identical, i.e., the capacity and the maximum time constraints are identical for all vehicles.

The computational results of the three methods, M1, M2, and M3 proposed by Cheong *et al*. (2002) and the bin packing methods implemented with the two proposed methods in this thesis, the ABLS&BP method and the GC&BP method, are summarized in Table 5.2.

Table 5.2 Computational results of the Cheong *et al*. method (2002) and the bin packing method

Problem instance	Total customer demands	No. of vehicles required				
		Cheong's method (M1)	Cheong's method (M2)	Cheong's method (M3)	ABLS&BP	GC&BP
1	197396	40	40	40	32	31
2	179874	36	36	35	28	26
3	223168	41	41	41	33	32
4	192241	37	37	37	24	22
5	196717	40	40	40	27	26
6	154069	37	36	36	21	21
7	308339	42	42	42	36	34
8	178436	30	30	30	22	21
9	225722	40	40	40	27	27
10	254774	42	41	41	32	31
11	199028	33	33	33	22	22
12	308339	42	42	42	30	30
13	281432	41	41	41	21	21
14	258612	42	42	42	31	29

In the ABLS&BP method, Type B of the ABLS algorithm is used for improvement. The parameter settings are as follows:

Initial temperature $T_0 = 200$.

Cooling rate *Red_F* = 0.85

Final temperature $T_f = 0.001$.

Maximum number of reheat, *Max_R*, = 3.

Iterations for each temperature: *Iter_T* = 4000

Computational experience shows that the number of chosen nodes cannot be too large in the ABLS&BP method. For most of the cases tested, good results are obtained when the numbers of nodes chosen are set to about 10% of the total routes.

In the GC&BP method, the parameter settings are as follows:

Initial temperature $T_0 = 15$.

Cooling rate *Red_F* = 0.85

Final temperature T_f = 0.1.

Maximum number of reheat, *Max_R,* = 3.

Iterations for each temperature: *Iter_T* = 4000

The results of the comparison show that the proposed bin packing method can produce a significantly better solution than those obtained by other methods for the 14 instances tested. In some instances, the solution can be improved by more than 40%.

5.5 Conclusions

In this chapter, an application of the proposed algorithms, the ABLS method and the GC method, to a real-world soft drinks distribution problem is presented. A bin packing composite algorithm is applied to solve the problem. The results show that it can obtain better solutions than other approaches proposed in the literature. In fact, for some problem instances, the improvement can be more than 40%.

Chapter 6

Summary and Conclusions

It is well known that the VRP is an NP-hard problem, which means that a polynomially-bound optimal algorithm is unlikely to exist for it. Therefore, many heuristic algorithms have been proposed to deal with this hard combinatorial problem in the literature. In this thesis, an ABLS method and a GC method for solving several types of VRPs is proposed. The performance of the proposed algorithms is compared against several algorithms proposed in the literature. In addition, an application study of the proposed algorithms to a real-world soft drinks distribution problem is provided. The computational experiments show that the proposed algorithms together with the bin packing composite algorithm are able to obtain very good solutions to this problem.

6.1 Summary and Conclusions

In Chapter 1, the background of this study is described briefly and the definition of the VRP is introduced. In this chapter, the characteristics and the basic classifications of VRPs are introduced. The VRPs can be classified into 4 main problems: capacitated VRPs and distance-constrained VRPs; VRPs with time windows; VRPs with backhauls; and VRPs with pickup and delivery.

The literature review of the approaches for the VRPs proposed by various researchers is given in Chapter 2. The approaches described include exact algorithms, classical heuristics and meta-heuristics.

In Chapter 3, the proposed ABLS method is introduced. The basic idea, classification of the ABLS algorithm, computational tests and analysis are described in detail. In addition, some possible applications of the ABLS method are discussed. Comparisons with other existing algorithms, such as Osman (1993) and Breedam (1995)'s SA implementations, are also presented in this chapter.

Another proposed method, the GC method, is introduced in Chapter 4. Computational results show that its SA implementation combined with a new improvement procedure, middle improvement procedure, performs better than other SA implementations and is comparable with some other tabu search implementations reported in the literature.

In Chapter 5, the proposed bin packing composite algorithm is applied to solve a real-world soft drinks distribution problem. The computational results show that this composite algorithm is able to obtain good solutions to this problem. The following conclusions can be drawn in this study:

(1) The proposed ABLS algorithm is an effective local search method for solving the VRP. Its performance can match some other algorithms proposed in the literature.

(2) The effectiveness of different implementations of the ABLS algorithm depends on the characteristics of the data for each problem. For instance, if the remaining capacity for most of the routes is very low in the current solution, it is more efficient to implement the Type A ABLS algorithm than the Type B ABLS algorithm.

(3) The second proposed algorithm, the GC method, is also an effective algorithm for solving the VRP. Its SA implementation combined with the middle improvement procedure performs much better than several SA and tabu search methods.

(4) The bin packing composite method is an effective method for solving some real-world problems, especially for the VRP with multiple trips.

6.2 Main Contributions of this Study

The main contributions of this study are briefly summarized as follows:

(1) Two local search methods for VRPs, the ABLS method and the GC method, are proposed for solving the CVRP and some of its variants.

(2) Some composite algorithms consisting of the proposed ABLS and other local search procedures are proposed for solving various types of CVRPs.

(3) A new effective improvement procedure, middle improvement procedure, is proposed. For some problem instances, this improvement procedure is able to generate better solutions to the problem.

(4) An extensive computational study on the performance of the two proposed algorithms, the ABLS and the GC, and some composite procedures is conducted. The computational results show that these procedures are either comparable with or superior to some other efficient heuristics proposed in the literature.

(5) An application study of the two proposed methods to a real-world soft drinks distribution problem is conducted. A bin packing composite procedure is developed to solve this problem. The computational results show that the bin packing procedure is able to obtain better solutions than other approaches proposed in the literature. For some problem instances tested, the improvement can be more than 40%.

6.3 Suggestions for Future Research

The VRP has generated practical interest and has gained the attention of many researchers to develop various solution procedures. The two proposed methods, the ABLS method and the GC method, and some composite methods are able to solve several types of VRPs. Some suggestions for future research are:

(1) To explore the application of the ABLS method and the GC method to solve other types of VRPs, such as the VRPs with time windows constraints;

(2) To apply the ABLS method and the GC method to solve the stochastic VRPs;

(3) To design other ABLS composite procedures for solving the various types of VRPs; and

(4) To implement the middle improvement procedure to other algorithms; and

(5) To apply the basic ideas of the ABLS and GC methods to other types of combinatorial optimization problems.

References

Achuthan, N. R., Caccetta, L. and Hill, S. P., “Capacitated vehicle routing problem: some new cutting planes”, Asia-Pacific Journal of Operational Research, Vol. 15, pp. 109-123. 1998.

Applegate, D., Bixby, R., Chvátal, V. and Cook, W., “Implementing the Dantzig–Fulkerson–Johnson algorithm for large traveling salesman problems”, Mathematical Programming, Series B 97, pp.91-153. 2003.

Augerat, P., Belengure, J. M., Benavent E., Corberan, A., Naddef, D. and Rinaldi, G. “Computational results with a branch and cut code for the capacitated vehicle routing problem”, Technical Report RR949-M, Universitė Joseph Fourier Grenoble, France. 1995.

Baker, E. K., Evolution of microcomputer-based vehicle routing software: Case studies in the United States, In Toth, P. and Vigo, D., editors, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, Philadelphia, pp. 1-26. 2002.

Balas, E., and Christofides, N., “A restricted lagrangean approach to the traveling salesman problem”, Mathematical Programming, Vol. 21, pp. 19-46. 1981.

Basnet, C., Foulds, L. and Igbaria, M., “Fleet manager: a microcomputer-based decision support system for vehicle routing ”, Decision Support Systems, Vol. 16, pp.195-207. 1996.

Bertsimas, D. J. and Simchi-Levi, D., “A new generation of vehicle routing research: Robust algorithms addressing uncertainty”, Operations Research, Vol. 44, pp. 286-304. 1996.

Bodin,. L. D., Golder, B. L., Assad, A. A. and Ball, M., “Routing and scheduling of vehicles and crews, the state of art”, Computers and Operations Research, Vol. 44, pp. 63-212. 1983.

Bramel, J. and Simchi-Levi, D., “A location based heuristic for general routing problems”, Operations Research, Vol. 43, pp.649-660. 1995.

Bramel, J. and Simchi-Levi, D., “Probabilistic analyses and practical algorithms for the vehicle routing problem with time windows”, Operations Research, Vol. 44, pp. 501-509. 1996.

Brandao, J. and Mercer, A., “The multi-trip Vehicle routing problem”, Journal of Operational Research Society, Vol. 49, pp. 799-805. 1998.

Breedam, A. V., “An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints”, Ph.D. Thesis, University of Antwerp.1994.

Breedam, A. V., "An analysis of the effect of local improvement operators in generic algorithms and simulated annealing for the vehicle routing problem", RUCA Working Paper 96/14, University of Antwerp, Belgium. 1996.

Breedam, A. V., "Improvement heuristics for the vehicle routing problem based on simulated annealing", European Journal of Operational Research, Vol. 86, pp. 480-490. 1995.

Bullnheimer, B., Hartl, R. F. and Strauss, C., "An improved ant system for the vehicle routing problem", Annals of Operations Research, Vol. 89, pp. 319-328. 1999.

Bullnheimer, B., Hartl, R. F. and Strauss, C., Applying the ant system to the vehicle routing problem, In Voss, S., Martello, S., Osman, I. H., and Roucairol, C., editors, Meta-heuristics: Advances and Trends in Local search Paradigms for Optimization, Kluwer Boston, MA, pp. 109-120. 1998.

Carpaneto, G. and Toth, P., "Some new branching and bounding criteria for the asymmetric traveling salesman problem", Management Science, Vol. 26, pp. 736-743. 1980.

Carpaneto, G., Fischetti, M. and Toth, P., "New lower bounds for the symmetric traveling salesman problem", Mathematical Programming, Vol. 45, pp. 233-254. 1989.

Carraway, R. L., Morin, T. L. and Moskowitz, H., "Generalized dynamic programming for stochastic combinatorial optimization", Operations Research, Vol. 37, pp. 819-829. 1989.

Carter, M., Farvolden, J., Laporte, G. and Xu, J., "Solving an integrated logistics problem arising in grocery distribution", INFOR, Vol. 34, pp.290-306. 1996

Chatterjee, S., Carrera, C. and Lynch, L. A., "Genetic algorithms and traveling salesman problem", European Journal of Operational Research, Vol. 93, pp.490-510. 1996.

Cheong, Y. M., Ong, H. L. and Huang, H. C., "Modeling the vehicle routing problem for a soft drink distribution company", Asia-Pacific Journal of Operational Research, Vol. 19, pp. 17-34. 2002.

Christofides, N, Mingozzi, A. and Toth, P., The vehicle routing problem, In Christofides, N., Mingozzi, A., Toth, P. and Sandi, C., editors, Combinatorial Optimization, Wiley, Chichester, UK, pp. 315-338. 1979.

Christofides, N, "The shortest Hamiltonian chain of a graph", SIAM Journal on Applied Mathematics, Vol. 19, pp. 689-696. 1970.

Christofides, N, The traveling salesman problem, In Christofides, N., Mingozzi, A., Toth, P. and Sandi, C., editors, Combinatorial Optimization, Wiley, Chichester, UK, pp. 131-149. 1979.

Christofides, N., and Eilon. S., “An algorithm for the vehicle dispatching problem”, Operational Research Quarterly, Vol. 20, pp. 309-318. 1969.

Christofides, N, Vehicle routing, In Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B., editors, The Traveling Salesman Problem, Wiley, Chichester, UK, pp. 431-448. 1985.

Clarke, G. and Wright, J., “Scheduling of vehicles from a central depot to a number of delivery points”, Operations Research, Vol. 12, pp. 568-581. 1964.

Coffman, E. G., Garey, M. R. and Johnson, Jr. D. S., Approximation algorithms for bin packing: A survey, In Hochbaum, D. S., editors, Approximation Algorithms for NP-hard Problems, PWS Pub. Co., Boston, pp.46-93, 1995.

Colorni, A. Dorigo, M. and Maniezzo, V., “ Distributed optimization by ant colonies”, In F. Varela and P. Bourgine, editors, Proceedings of the European Conference on Artificial Life, Elsevier, Amsterdam. 1991.

Cordeau, J. Laporte, F G. and Mercier A., “A unified tabu search heuristic for vehicle routing problems with time windows”, Journal of the Operational Research Society, Vol. 52, pp. 928-936. 2001.

Cornuejols, G. and Harche, F. “Polyhedral study of the capacitated vehicle routing problem”, Mathematical Programming, Vol. 60, pp. 21-52. 1993.

Coy, S. P., Golden, B. L. and Wasil, E. A., “A computational study of smoothing heuristics for the traveling salesman problem”, European Journal of Operational Research, Vol. 124, pp. 15-27. 1999.

Croes, G. A., “A method for solving Traveling Salesman Problems”, Operations Research, Vol. 6, pp. 791-812. 1958.

Cullen, F., Jarvis, J. and Ratliff, D., “Set partitioning based heuristics for interactive routing”, Networks, Vol. 11, pp. 125-144. 1981.

Dantzig, G. B. and Ramser, J. H., “The truck dispatching problem”, Management Science, Vol. 6, pp. 80-91. 1959.

Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., “Solution of a large-scale traveling-salesman problem”, Operations Research, Vol. 2, pp. 393-410. 1954.

Desrochers, M., Desrosiers, J. and Solomon, M. M., “A new optimization algorithm for the vehicle routing problem with time windows”, Operations Research, Vol. 40, pp. 342-354. 1992.

Dijkstra, E. W., “ A note on two problems in connection with graphs”, Numerische Mathematic, Vol. 1, pp. 269-271. 1959.

Dorigo, M. and Gambardella, L. M., "Ant colony system: A cooperative learning approach for the traveling salesman problem", IEEE Transactions on Evolutionary Computation, Vol. 1, pp. 53-66. 1997.

Dorigo, M., Maniezzo, V. and Colorni, A., "Ant system: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetics Part B, Vol. 26, pp. 29-41. 1996.

Dowsland, K. A., Simulated annealing, In Reeves, C. R., editors, Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, UK, pp. 20-69. 1993.

Fischetti, M., Gonzalez, J. and Toth, P., "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem", Operations Research, Vol. 45, pp.378-394. 1997.

Fischetti, M. and Toth, P., "A polyhedral approach to the asymmetric traveling salesman problem", Management Science, Vol. 43, pp.1520-1536. 1997.

Fischetti, M., Toth, P., and Vigo, D., "A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs", Operations Research, Vol. 42, pp. 846-859. 1994.

Fisher, M. L. and Jaikumar, R., " A generalized assignment heuristic for the vehicle routing problem", Networks, Vol. 11, pp. 109-124. 1981.

Fisher, M. L., "Optimal solution of vehicle routing problems using minimum k-trees", Operations Research, Vol. 42, pp. 626-642. 1994.

Fleischmann, B., "The vehicle routing problem with multiple use of vehicles", Working paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg. 1990.

Gendreau, M., Hertz, A. and Laporte, G., "A tabu search heuristic for the vehicle routing problem", Management Science, Vol. 40, pp. 1276-1290. 1994.

Gendreau, M., Laporte, G. and Potvin, J. Y., Metaheuristics for the capacitated VRP, Toth, P. and Vigo, D., editors, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, Philadelphia, pp. 129-154. 2002.

Gendreau, M., Laporte, G. and Seguin, R., "New insertion and post-optimization procedures for the traveling salesman problem", Operations Research, Vol. 40, pp. 1086-1094. 1992.

Gendreau, M., Laporte, G. and Seguin, R., "Stochastic vehicle routing", European Journal of Operational Research, Vol. 88, pp. 3-12. 1996.

Gillett, B. E. and Miller, L. R., "A heuristic algorithm for the vehicle dispatch problem", Operations Research, Vol. 22, pp. 340-349. 1974.

Glover, F., “Future paths for integer programming and links to artificial intelligence”, Computers & Operations Research, Vol. 13, pp. 533-549. 1986.

Golden, B. L., and Skiscim, C. C., “Using simulated annealing to solve routing and location problems”, Naval Research Logistics Quarterly, Vol. 13, pp. 261-280. 1986.

Golden, B. L., Wasil, E. A., Kelly, J. P. and Chao, I-M., Meta-heuristics in vehicle routing, In Fleet Management and Logistics, Crainic, T. G. and Laporte, G., editors, Kluwer, Boston, pp. 33-56. 1998.

Gu, J., Huang, X., “Efficient local search with search space smoothing: a case study of the traveling salesman problem”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 5, pp. 728-735. 1994.

Hadjiconstantinou, E. and Roberts, D., Routing Under Uncertainty: an application in the scheduling of field service engineers, In Toth, P. and Vigo, D., editors, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, Philadelphia, pp. 331-352. 2002.

Hillier, F. S. and Lieberman, G. J., The transportation and assignment problems, In Hillier, F. S. and Lieberman, G. J., editors, Introduction to Operations Research, McGraw-Hill, pp.303-352. 1995.

Holland, J. H., Adaptation in Natural and Artificial Systems, The university of Michigan Press, Ann Arbor, MI. 1975.

Jog, P., Auh, J. Y. and Van, G. D., “The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem”, Proceedings of the Third International Conference on Algorithms, Morgan Kaufmann, pp. 110-115. 1989.

Johnson, D. C. and Mcgeoch, L. A., The traveling salesman problem: A case study. In Aarts, E. H. L. and Lenstra, J. K., editors, Local Search in Combinatorial Optimization, Wiley, NY, pp. 216-310. 1997.

Jonker, R. and Volgenant, A., “A shortest augmenting path algorithm for dense and sparse linear assignment problems”, Computing, Vol. 38, pp. 325-340. 1987.

Kao, E. P. C., “A preference order dynamic program for a stochastic traveling salesman problem”, Operations Research, Vol. 26, pp. 1033-1045. 1978.

Karp, R., Reducibility among combinatorial problems, In Miller, R. and Thatcher, J., editors, Complexity of Computer Computations, Plenum Press, New York, pp. 85-104. 1972.

Kawamura, H., Yamamoto, M., Mitamura, T., Suzuki, K., and Ohuchi, A., “ Cooperative search on pheromone communication for vehicle routing problems”, IEEE Transactions on Fundamentals, E81-A, pp. 1089-1096. 1998.

Kinderwater, G. A. P. and Savelsbergh, M.W. P., Vehicle routing: Handling edge exchanges. In Aarts, E. H. L. and. Lenstra, J. K., editors, Local Search in Combinatorial Optimization, Wiley, NY, pp. 337-360. 1997.

Kirkpatrick, S., Gelatt, Jr. C. D., and Vecchi, M. P., “Optimization by simulated annealing”, Science, Vol. 220, pp. 671-680. 1983.

Knox, J., “ Tabu search performance on the symmetric traveling salesman problem”, Computers and Operations Research, Vol. 21, No. 8, pp. 867-876. 1994.

Kuhn, H. W., “The Hungarian method for the assignment problem”, Naval Research Logistics Quarterly, Vol. 22, pp. 83-97. 1955.

Laarhoven, P. J. M. V., “Theoretical and computational aspects of simulated annealing”, Ph.D. Thesis, Erasmus University, Rotterdam. 1988.

Langevin, A., Soumis, F. and Desrosiers, J., “Classification of traveling salesman problem formulations”, Operations Research Letters, Vol. 9, pp.127-132. 1990.

Laporte, G and Loubeaus, F., “The integer L-shaped method for stochastic integer programs with complete recourse”, Operations Research Letters, Vol. 13, pp. 133-142. 1993.

Laporte, G and Semet, F., Classical heuristics for the capacitated VRP, In Toth, P. and Vigo, D., editors, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, Philadelphia, pp. 109-128. 2002.

Laporte, G., “The traveling salesman problem: An overview of exact and approximate algorithms”, European Journal of Operational Research, Vol. 59, pp. 231-247. 1992.

Laporte, G., Mercure, H. and Nobert, Y., “An exact algorithm for the asymmetrical capacitated vehicle routing problem”, Networks, Vol. 16, pp. 33-46. 1986.

Lin, S. and Kernighan, B., “An effective heuristic algorithm for the traveling salesman problem”, Operations Research, Vol. 11, pp. 972-989. 1973.

Lin, S., “Computer solutions of the traveling salesman problem”, Bell System Technical Journal, Vol. 44, pp. 2245-2269. 1965.

Martello, S. and Toth, P., Knapsack Problems: Algorithms and Computer Implementations, Wiley, Chichester, UK. 1990.

Miller, C. E., Tucher, A. W. and Zemlin, R. A., “Integer programming formulations and traveling salesman problem”, Journal of the Association for Computing Machinery, Vol. 7, pp.326-329. 1960.

Miller, D. L. and Pekny, J. F., “Exact solution of large asymmetric traveling salesman problems ”, Science, Vol. 251, pp. 754-761. 1991.

Miller, D. L., “A matching based exact algorithm for capacitated vehicle routing problems”, ORSA Journal on Computing, Vol. 7, No. 1, pp. 1-9. 1995.

Mole, R. H. and Jameson, S.R., “A sequential route-building algorithm employing a generalized savings criterion, Operational Research Quarterly, Vol. 27, pp. 503-511. 1976.

Naddef, D. and Rinaldi, G., Branch-and cut algorithms for the capacitated VRP. In Toth, P. and Vigo, D., editors, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, Philadelphia, pp. 53-84. 2002.

Norback, J. and Love, R., “Geometric approaches to solving the traveling salesman problem”, Management Science, Vol.23, pp. 1208-1223. 1977.

Osman, I. H., “Meta-strategy simulated annealing and tabu search algorithms for the vehicle routing problem”, Annals of Operations Research, Vol. 41, pp. 421-451. 1993.

Osman, I. H. and Salhi, S., Local Search Strategies for the Vehicle Fleet Mix Problem. In Rayward-Smith, V. J., Osman, I. H., Reeves, C. R. and. Smith, G. D., editors, Modern Heuristic Search Methods, John Wiley and Sons, pp. 131-153. 1997.

Padberg, M. W. and Rinaldi, G., “A branch and cut algorithm for the resolution of Large-scale symmetric traveling salesman problems”, SIAM Review, Vol.33, pp.60-100. 1991.

Padberg, M. W. and Rinaldi, G., “Facet identification for the symmetric traveling salesman problem”, Mathematical Programming, Vol. 47, pp.219-257. 1990.

Padberg, M. W. and Rinaldi, G., “Optimization of a 532-city symmetric traveling salesman problem by branch and cut”, Operations Research Letters, Vol.6, pp.1-7. 1987.

Picard, R. G. and Brody, J. H., The newspaper publishing industry, Allyn and Bacon, Boston, MA.1997.

Potvin, J. Y. and Bengio, S., “The vehicle routing problem with time windows Part II: Genetic search”, INFORMS Journal on Computing, Vol. 8, pp. 165-172. 1996.

Pureza, V. M. and França, P. M., “Vehicle routing problems via tabu search metaheuristic”, Publication CRT-747, Centre de recherché sur les transports, Montréal, 1991.

Reeves, C. R., Genetic algorithm, In Reeves, C. R., editors, Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, UK, pp.151-196. 1993.

Rego, C. and Roucairol, C., “A parallel tabu search algorithm using ejection chains for the vehicle routing problem”, In I. H. Osman and J. P. Kelly, Meta-heuristics: Theory and Applications, Kluwer, Boston, MA, pp. 661-675. 1996.

Rochat, Y. and Taillard, E. D., “ Probabilistic diversification and intensification in local search for vehicle routing”, Journal of Heuristics, Vol. 1, pp. 147-167. 1995.

Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M., “An analysis of several heuristics for the traveling salesman problem”, SIAM Journal on Computing, Vol. 6, pp. 563-581. 1977.

Rossier, Y., Troyon, M. and Liebling, T. M., “Probabilistic exchange algorithms and the Euclidean traveling salesman problems”, Operations Research Spektrum, Vol. 8, pp. 151-164. 1986.

Savelsbergh, M. W. P. and Sol, M., “The general pickup and delivery problem”, Transportation Science, Vol. 29, pp. 17-29. 1995.

Savelsbergh, M. W. P., “Local search in routing problems with time windows”, Annals of Operations Research, Vol. 4, pp. 285-305. 1985.

Schmitt, L. J. and Amini, M. M., “Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study”, European Journal of Operational Research, Vol. 108, pp.551-570. 1998.

Schneider, J., Dankesreiter, M., Fettes, W., Morgenstern, I., Schmid, M. and Singer, J., “Search-space smoothing for combinatorial optimization problems”, Physica A, Statistical and Theoretical Physics 243. 1/2, pp. 77-112. 1997.

Shi, L., Olafsson, S. and Sun, N., “New parallel randomized algorithms for the traveling salesman problem”, Computers and Operations Research, Vol.26, pp. 371-394. 1999.

Shigeru, T. and Evans, J. R., “Optimizing tabu list size for the traveling salesman problem”, Computers and Operations Research, Vol.25, pp. 91-97. 1998.

Sniedovich, E., “Analysis of a preference order traveling salesman problem”, Operations Research, Vol. 29, pp. 1234-1237. 1981.

Solomon, M. M. and Desrosiers, J., “Time window constrained routing and scheduling problems”, Transportation Science, Vol. 22, pp. 1-13. 1988.

Stewart, W. R., “A computationally efficient heuristic for the traveling salesman problem”, Proceedings of the 13th Annual Meeting of Southeastern TIMS, pp.75-85. 1977.

Stewart, W. R., “Accelerated branch exchange heuristics for symmetric traveling salesman problems”, Networks, Vol. 17, pp. 423-437. 1987.

Taillard, E. D., “Parallel iterative search methods for vehicle routing problems”, Networks, Vol. 23, pp. 661-673. 1993.

Taillard, E. D., Laporte, G. and Gendreau, M., “Vehicle routing with multiple use of vehicles”, Journal of Operational Research Society, Vol. 47, pp. 1065-1070. 1996.

Teng, S. Y., Ong, H. L. and Huang, H. C., “A comparative study of metaheuristics for vehicle routing problem with stochastic demands ”, Asia-Pacific Journal of Operational Research, Vol. 20, pp.1-17. 2003.

Teodorović, D. and Pavković, G., “A simulated annealing technique approach to the VRP in the case of stochastic demand”, Transportation Planning and Technology, Vol. 16, pp. 261-273. 1992.

Thompson, P. M. and Psaraftis, H. N., “Cyclic transfer algorithms for multi-vehicle routing and scheduling problems”, Operations Research, Vol. 41, pp. 935-946. 1993.

Tillman, F., “The multiple terminal delivery problem with probabilistic demands”, Transportation Science, Vol. 3, pp. 192-204. 1969.

Toth, P. and Vigo, D., “Models, relaxations and exact approaches for the capacitated vehicle routing problem”, Discrete Applied Mathematics, Vol. 123, pp. 487-512. 2002*a*.

Toth, P. and Vigo, D., “A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls”, European Journal of Operational Research, Vol. 113, pp. 528-543. 1999.

Toth, P. and Vigo, D., “The granular tab search (and its application to the vehicle routing problem)”, Technical Report OR/98/9, DEIS, University of Bologna, Italy. 1998.

Toth, P. and Vigo, D., An overview of vehicle routing problems, In Toth, P. and Vigo, D., editors, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, Philadelphia, pp. 1-26. 2002*b*.

Volgenant, T. and Jonker, R., “A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation”, European Journal of Operational Research, Vol. 9, pp. 83-89. 1982

Wilson, H., Sussman, J., Wang, H. and Higonnet, B., “Scheduling algorithms for dial-a-ride system”, Technical Report USL TR-70-13, Urban Systems laboratory, MIT, Cambridge, MA. 1971.

Xu, J. and Kelly, J, P., “A network flow–based tabu search heuristic for the vehicle routing problem”, Transportation Science, Vol. 30, pp. 379-393. 1996.

Yellow, P., “A computational modification to the savings method of vehicle scheduling”, Operational Research Quarterly, Vol. 21, pp. 281-283. 1970.