A GA BASED META-HEURISTIC FOR CAPACITATED VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICK-UP AND DELIVERIES

by

ARİF VOLKAN VURAL

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

August 2003

A GA BASED META-HEURISTIC FOR CAPACITATED VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICK-UP AND DELIVERIES

APPROVED BY:

Assist. Prof. Bülent Çatay …….

(Thesis Advisor)

Assist. Prof. Berrin Yanıkoğlu …….

Assist. Prof. Tonguç Ünlüyurt …….

DATE OF APPROVAL: …….

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

I feel utterly indebted to my thesis advisor Assist. Prof. Bülent Çatay for his invaluable suggestions and supervision throughout the study. Without him, there would be no way to produce this study.

I would also like to thank to my committee members Assist. Prof. Tonguç Ünlüyurt and Assist. Prof. Berrin Yanıkoğlu for their excellent suggestions and remarks.

This thesis would never give out solutions without the help of my friends Alisher Kholmatov and Zerrin Işık. They taught me not only how to trace and debug a code, but also patiently did them for me when I felt helpless in overcoming code mishaps.

Last, but not least I would express my gratitude to my family. Without their financial and spiritual support, I would not be able to accomplish my studies.

ABSTRACT

In this study, we focus on the theoretical framework of a decision model for a real world problem. The problem reveals itself as simultaneous distribution of commodities and recollection of empty packages the same size as the initial state with a single depot and a fleet of uniform vehicles with limited capacities. Resembling instances pile a profound literature under the category of "pick-up and delivery problems with backhauls" and "rural postman problem." To solve the arousing NP-hard problem we use genetic algorithm approach. Computational efficiency and a good solution performance are sought.

We have studied the wide literature of the vehicle routing problems, classified and briefly introduced the previous asserted algorithms, which provide considerably high quality solutions.

We have developed a genetic algorithm based meta-heuristic on a linear IP model proposed by Dethloff (2001) and conducted tests to come up with a robust heusritic producing results with a reasonable quality. The models we studied were mainly taken from the machine scheduling literature and adapted to handle our problem. Our research has revealed that no resembling problem has ever been proposed to be solved using the genetic algorithms approach. Thus, this work is a first in its field.

The improvement algorithm is found to be considerably good performing while the random keys method failed to produce reasonable solutions. We have tested our algorithm on two benchmark problems introduced by Min (1989) and Dethloff (2001). The latter is composed of 40 problem instances generated. We have performed parameter tests to tune our algorithm and shown that our algorithm produced the best ever solution for the first problem and considerably good solutions for the second one.

ÖZET

Bu çalışmada gerçek hayatta örnekleri görülebilen özel bir karar problemi için kuramsal bir yapı oluşturma üzerine odaklanılmıştır. Problemin kendisi aynı büyüklükte olan dağıtılacak ve iade malların kapasiteleri belirlenmiş özdeş araçlardan oluşan bir filo ile dağıtılması ve aynı anda depoya götürülmek üzere toplanması olarak tanımlanabilir. Literatürde benzer problemler "geri seferli toplama ve iade problemleri" ve "kırsal kesim postacı problemleri" başlıkları altında incelenmektedir. Oluşan *NP*-hard problemin çözümü için genetik algoritma (GA) kullanılmıştır. İşlemsel verimlilik ve iyi çözüm performansı aranmıştır.

Araç rotalama problemleri üzerine olan geniş literatür taranmış, en çok kullanılan çözüm yöntemleri kendi aralarında sınıflandırılmış ve kısaca tanıtılmıştır.

Dethloff (2001) tarafından önerilen doğrusal tamsayı programlama modeli baz alınarak genetic algoritma esaslı bir model geliştirilmiş ve bunun üzerinde denemeler gerçekleştirilerek yüksek kalitede sonuç üretebilen sağlam bir yaklaşım geliştirilmesine çalışılmıştır. Üzerinde çalıştığımız modeller benzer özellikler sergileyen makine çizelgeleme literatüründen alınarak problemimize uyarlanmıştır. Araştırmamız, üzerinde çalıştığımız problem için henüz GA'ları kullanarak çözüme ulaşmış herhangi bir çalışma olmadığını göstermiştir. Dolayısıyla yaptığımız çalışma mevcut haliyle alanında bir ilktir.

İkinci olarak önerilen geliştirilmiş algoritmamız oldukça başarılı sonuçlar üretirken rastsal anahtarlama metodu ile üretilen sonuçların oldukça kötü olduğu gözlenmiştir. Çözümlerimiz, yayınlanmış ve akademik literature girmiş olan Min'in (1989) tek ve Dethloff'un (2001) kırk örnekli problemlerinin girdileri esas alınarak üretilmiş ve yayınlanmış, bilinen en iyi sonuçlarla karşılaştırılmıştır. Parametre testleri

sonucunda uyarlamaları yapılmış olan algoritmamızın ilk problem için en iyi, ikincisi içinse göreceli olarak daha iyi sonuçlar bulduğu gösterilmiştir.

TABLE OF CONTENTS

1. INTRODUCTION 1

2. LITERATURE REVIEW 3

2.1. Vehicle Routing Problem Definition 3

2.2. Categories of Vehicle Routing Problems 4

2.3. VRP with Simultaneous Pick-up and Delivery 6

2.3.1. Problem Definition 6

2.3.2. Mathematical Formulation of the Problem 8

2.4. Optimal Algorithms for VRP 10

2.4.1. Dynamic Programming 10

2.4.2. Langrangean Relaxation Based Methods 11

2.4.3. Column Generation Based Methods 12

2.5. Approximation Algorithms and Heuristics for VRP 13

2.5.1. Construction Algorithms 14

2.5.2. Route Improvement Heuristics 17

2.5.3. Meta-Heuristics 20

3. A DUAL GA APPROACH FOR THE VRPSPD 25

3.1. Random Keys Method 27

3.2. Improvement Heuristic 29

4. COMPUTATIONAL STUDY 33

4.1. Benchmark Problems 33

4.2. Parameters and Analysis 34

4.3. Comparison of Results with the Benchmarks 37

5. CONCLUSION 40

REFERENCES 42

APPENDICES 45

Appendix A: Pseudo-Code for Random Keys Method 46

Appendix B: Pseudo-Code For the Second Method 49

LIST OF FIGURES

Figure 1.1 General representation of the Vehicle Routing Problem..... 4

Figure 2.1 Main steps of Ulusoy's algorithm (Lacomme *et al*., 2001)..... 16

Figure 2.2 2-Opt, 1-1 exchange, 1-0 exchange moves for single and multiple routes (Tarantilis et al., 2002)..... 19

Figure 3.1 An example of replications and omitting of genes through crossover 26

Figure 3.2 Two schedules (chromosomes) and an output of crossover operator applied to these chromosomes (Topcuoglu and Sevilmis, 2002) 30

Figure 3.3 Brief pseudo code for the second heuristic 31

Figure 4.1 Graph of Total Tour Length versus Generations for the SCA 3 Problems ... 36

Figure 4.2 Graph of Total Tour Length versus Generations for the SCA 8 Problems ... 36

Figure 4.3 Graph of Total Tour Length versus Generations for the CON 3 Problems .. 36

Figure 4.4 Graph of Total Tour Length versus Generations for the CON 8 Problems .. 37

LIST OF TABLES

Table 4.1 Hybrid heuristics generated with the proposed algorithms 38

A GA BASED META-HEURISTIC FOR CAPACITATED VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICK-UP AND DELIVERIES

by

ARİF VOLKAN VURAL

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

August 2003

A GA BASED META-HEURISTIC FOR CAPACITATED VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICK-UP AND DELIVERIES

APPROVED BY:

Assist. Prof. Bülent Çatay …….

(Thesis Advisor)

Assist. Prof. Berrin Yanıkoğlu …….

Assist. Prof. Tonguç Ünlüyurt …….

DATE OF APPROVAL: …….

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

I feel utterly indebted to my thesis advisor Assist. Prof. Bülent Çatay for his invaluable suggestions and supervision throughout the study. Without him, there would be no way to produce this study.

I would also like to thank to my committee members Assist. Prof. Tonguç Ünlüyurt and Assist. Prof. Berrin Yanıkoğlu for their excellent suggestions and remarks.

This thesis would never give out solutions without the help of my friends Alisher Kholmatov and Zerrin Işık. They taught me not only how to trace and debug a code, but also patiently did them for me when I felt helpless in overcoming code mishaps.

Last, but not least I would express my gratitude to my family. Without their financial and spiritual support, I would not be able to accomplish my studies.

ABSTRACT

In this study, we focus on the theoretical framework of a decision model for a real world problem. The problem reveals itself as simultaneous distribution of commodities and recollection of empty packages the same size as the initial state with a single depot and a fleet of uniform vehicles with limited capacities. Resembling instances pile a profound literature under the category of "pick-up and delivery problems with backhauls" and "rural postman problem." To solve the arousing NP-hard problem we use genetic algorithm approach. Computational efficiency and a good solution performance are sought.

We have studied the wide literature of the vehicle routing problems, classified and briefly introduced the previous asserted algorithms, which provide considerably high quality solutions.

We have developed a genetic algorithm based meta-heuristic on a linear IP model proposed by Dethloff (2001) and conducted tests to come up with a robust heusritic producing results with a reasonable quality. The models we studied were mainly taken from the machine scheduling literature and adapted to handle our problem. Our research has revealed that no resembling problem has ever been proposed to be solved using the genetic algorithms approach. Thus, this work is a first in its field.

The improvement algorithm is found to be considerably good performing while the random keys method failed to produce reasonable solutions. We have tested our algorithm on two benchmark problems introduced by Min (1989) and Dethloff (2001). The latter is composed of 40 problem instances generated. We have performed parameter tests to tune our algorithm and shown that our algorithm produced the best ever solution for the first problem and considerably good solutions for the second one.

ÖZET

Bu çalışmada gerçek hayatta örnekleri görülebilen özel bir karar problemi için kuramsal bir yapı oluşturma üzerine odaklanılmıştır. Problemin kendisi aynı büyüklükte olan dağıtılacak ve iade malların kapasiteleri belirlenmiş özdeş araçlardan oluşan bir filo ile dağıtılması ve aynı anda depoya götürülmek üzere toplanması olarak tanımlanabilir. Literatürde benzer problemler "geri seferli toplama ve iade problemleri" ve "kırsal kesim postacı problemleri" başlıkları altında incelenmektedir. Oluşan *NP*-hard problemin çözümü için genetik algoritma (GA) kullanılmıştır. İşlemsel verimlilik ve iyi çözüm performansı aranmıştır.

Araç rotalama problemleri üzerine olan geniş literatür taranmış, en çok kullanılan çözüm yöntemleri kendi aralarında sınıflandırılmış ve kısaca tanıtılmıştır.

Dethloff (2001) tarafından önerilen doğrusal tamsayı programlama modeli baz alınarak genetic algoritma esaslı bir model geliştirilmiş ve bunun üzerinde denemeler gerçekleştirilerek yüksek kalitede sonuç üretebilen sağlam bir yaklaşım geliştirilmesine çalışılmıştır. Üzerinde çalıştığımız modeller benzer özellikler sergileyen makine çizelgeleme literatüründen alınarak problemimize uyarlanmıştır. Araştırmamız, üzerinde çalıştığımız problem için henüz GA'ları kullanarak çözüme ulaşmış herhangi bir çalışma olmadığını göstermiştir. Dolayısıyla yaptığımız çalışma mevcut haliyle alanında bir ilktir.

İkinci olarak önerilen geliştirilmiş algoritmamız oldukça başarılı sonuçlar üretirken rastsal anahtarlama metodu ile üretilen sonuçların oldukça kötü olduğu gözlenmiştir. Çözümlerimiz, yayınlanmış ve akademik literature girmiş olan Min'in (1989) tek ve Dethloff'un (2001) kırk örnekli problemlerinin girdileri esas alınarak üretilmiş ve yayınlanmış, bilinen en iyi sonuçlarla karşılaştırılmıştır. Parametre testleri

sonucunda uyarlamaları yapılmış olan algoritmamızın ilk problem için en iyi, ikincisi içinse göreceli olarak daha iyi sonuçlar bulduğu gösterilmiştir.

TABLE OF CONTENTS

1. INTRODUCTION 1

2. LITERATURE REVIEW 3

2.1. Vehicle Routing Problem Definition 3

2.2. Categories of Vehicle Routing Problems 4

2.3. VRP with Simultaneous Pick-up and Delivery 6

2.3.1. Problem Definition 6

2.3.2. Mathematical Formulation of the Problem 8

2.4. Optimal Algorithms for VRP 10

2.4.1. Dynamic Programming 10

2.4.2. Langrangean Relaxation Based Methods 11

2.4.3. Column Generation Based Methods 12

2.5. Approximation Algorithms and Heuristics for VRP 13

2.5.1. Construction Algorithms 14

2.5.2. Route Improvement Heuristics 17

2.5.3. Meta-Heuristics 20

3. A DUAL GA APPROACH FOR THE VRPSPD 25

3.1. Random Keys Method 27

3.2. Improvement Heuristic 29

4. COMPUTATIONAL STUDY 33

4.1. Benchmark Problems 33

4.2. Parameters and Analysis 34

4.3. Comparison of Results with the Benchmarks 37

5. CONCLUSION 40

REFERENCES 42

APPENDICES 45

Appendix A: Pseudo-Code for Random Keys Method 46

Appendix B: Pseudo-Code For the Second Method 49

LIST OF FIGURES

Figure 1.1 General representation of the Vehicle Routing Problem..... 4

Figure 2.1 Main steps of Ulusoy's algorithm (Lacomme *et al*., 2001)..... 16

Figure 2.2 2-Opt, 1-1 exchange, 1-0 exchange moves for single and multiple routes (Tarantilis et al., 2002)..... 19

Figure 3.1 An example of replications and omitting of genes through crossover 26

Figure 3.2 Two schedules (chromosomes) and an output of crossover operator applied to these chromosomes (Topcuoglu and Sevilmis, 2002) 30

Figure 3.3 Brief pseudo code for the second heuristic 31

Figure 4.1 Graph of Total Tour Length versus Generations for the SCA 3 Problems ... 36

Figure 4.2 Graph of Total Tour Length versus Generations for the SCA 8 Problems ... 36

Figure 4.3 Graph of Total Tour Length versus Generations for the CON 3 Problems .. 36

Figure 4.4 Graph of Total Tour Length versus Generations for the CON 8 Problems .. 37

LIST OF TABLES

Table 4.1 Hybrid heuristics generated with the proposed algorithms 38

1. INTRODUCTION

The transportation of goods or humans from one location to another has been a major problem to solve during distribution network design. The well-known Vehicle Routing Problem (VRP) arises through this major design need to utilize the resources of a distribution network to its utmost capacity. There exists a huge number of qualitative as well as quantitative criteria to take care of during analyzing phase of the problem. However, due to modeling concerns, only a set of aspects may be studied in a single problem.

Since it was first formulated in 1959 by Dantzig and Ramser, the VRP has appealed much attention by the operations research academia. The VRP is a resembling problem to the "Traveling Salesman Problem" (TSP). In the TSP, the aim is to find the shortest trip for a salesman who is supposed to cover all customers starting at an initial location and obliged to return to some definite location, which is usually the initial location. The "VRP" is the name given to the class of problems that not only comprises the TSP but also adds some new features that add up to the current complexity of the TSP.

In this study, one special configuration of the VRP is discussed. The problem comprises many customers or "nodes" to be served by a fleet of vehicles of homogeneous type and limited capacity. The vehicles deliver items to customers from the depot and pick-up loads to be delivered back to the depot at the end of the trip. The size of the picked up and delivered items are identical and they consume the same amount of capacity on each truck. Delivery and pick-up locations are unique and feeding a customer with anything picked up at a node other than the main depot is strictly avoided. The objective is to minimize the total distance covered by the fleet during service. Some instances of this type of problem may be observed in distribution networks of bottled spring water in re-collectable containers, industrial gas distribution-

collection in refillable tanks, liquefied petroleum gas distribution in commercial containers from wholesalers to retailers, and so on.

2. LITERATURE REVIEW

2.1. Vehicle Routing Problem Definition

The vehicle routing problem is represented as the following graph theoretic problem. Let $G = (V, A)$ be a complete graph where $V = \{0, 1, \ldots, n\}$ is the vertex set and A is the arc set. Vertices $j = 1, \ldots, n$ correspond to the customers, each with a known non-negative demand, d_j, to be delivered whereas vertex 0 correspond to the depot. A non-negative cost, c_{ij} , is associated with each arc $(i, j) \in A$ and represents the travel cost to go from vertex i to vertex j. If the cost values satisfy the symmetry, such that for any i and $j \in V$, $c_{ij} = c_{ji}$, then the problem is said to be symmetric VRP, else, it is called an asymmetric VRP. In several practical cases the cost matrix satisfies the triangle inequality, such that $c_{ik} + c_{kj} \geq c_{ij}$ for any $i, j, k \in V$ (Toth and Vigo, 1998).

The VRP consists of finding a collection of k simple circuits, each corresponding to a vehicle route, with minimum cost which is defined as the sum of the costs of the arcs belonging to the circuits, and such that:

i. each circuit visits vertex 0, i.e., the depot;

ii. each vertex $j \in V \setminus \{0\}$ is visited by exactly one circuit;

iii. for the case with limited vehicle capacity, C, and each vertex with a non-negative demand value, d_i, the sum of the demand of the vehicles visited by a circuit does not exceed the vehicle capacity, C.

Figure 1.1 illustrates how a solution to a VRP would look like after routes are generated. The sketch shows the vertices to be served (customers), the edges (route segments), and the depot.

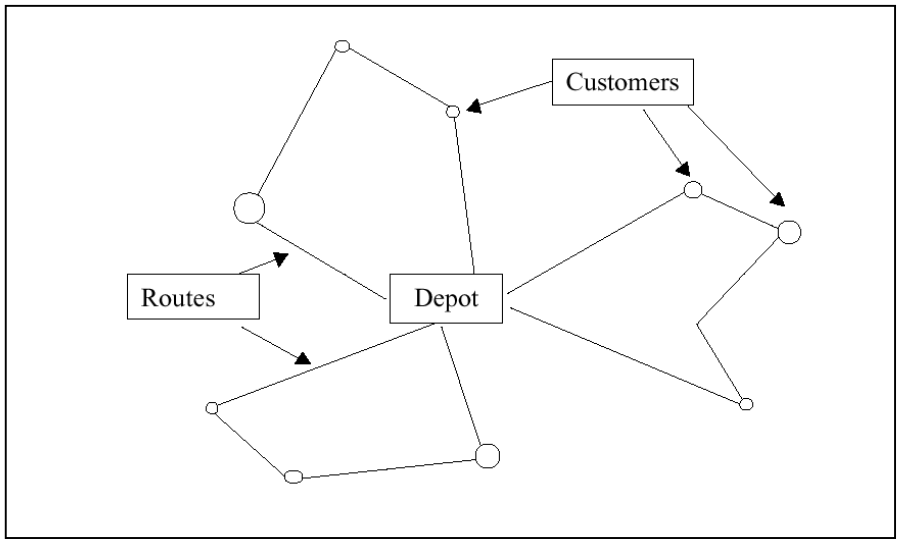

Figure 1.1 General representation of the Vehicle Routing Problem.

2.2. Categories of Vehicle Routing Problems

The VRP designates a wide range of set-ups on the original problem of TSP rather than addressing a specific problem. Several versions of the problem may be defined, depending on a number of factors, constraints, and objectives addressed in the problem context. Crainic and Laporte (1997) emphasize some questions in order to discriminate the problem through the tremendous VRP literature by discovering or highlighting the borders and content. The questions are listed as follows:

1) Does the problem involve deliveries, collections, or a combination of both? Are there precedence relations between deliveries or collections?
2) Does distribution take place through a single depot or from several centers?
3) How many vehicles are involved? Is the number fixed or does it constitute a decision variable? Is the vehicle fleet homogeneous or heterogeneous? What are the capacity, speed, operating costs of these vehicles?

4) What are the work conditions of the drivers? What is the pay structure? What is the length of a normal workday? What are the conditions on overtime? Are multiple same day trips allowed?

5) Is the demand known in advance or is it revealed in dynamic fashion during the course of operations?

6) How often or when must each customer be visited during the planning period? On a given day, must customers be visited within specific time windows?

It is fairly important to clarify the borders and content of the problem prior to developing an analytical approach towards a solution. From these questions, the main attributes within the configuration of most VRP problems ever published in literature are listed as follows:

- *Number of vehicles*: The upper limit on number of vehicles available for routing.
- *Vehicles' homogeneity/heterogeneity*: The condition on the vehicles' capacity, whether it is uniform for all vehicles or not.
- *Time windows*: The imposed time constraint for servicing a customer.
- *Backhauls*: Besides feeding a customer with its demand, the customer loads the truck with some load to be carried to another destination.
- *Splitting/Unsplitting of load*: The load to be delivered or picked up at any node may be divided into any number of groups in the splitting case and this is strictly forbidden in the unsplitting case. This puts forward multiple trips to any node rather than a single one during the routing process.
- *Single Depot/Multi Depot*: The distribution or collection process is constructed considering a single depot or multiple ones; even distribution and collection centers may be different.
- *Static/Dynamic Service Needs*: The demand values are either known in whole or unknown to some level, even in whole prior to establishing a route for the service vehicle.
- *Precedence/Coupling Constraints*: This is the case if a node's demand is satisfied with anything picked up at a node other than the depot (coupling

constraint), the latter node has a precedence in service to the prior node (precedence constraint).

2.3. VRP with Simultaneous Pick-up and Delivery

In this thesis, we focus on the theoretical framework of a decision model for a real world problem. The problem reveals itself as distribution of commodities and simultaneous recollection of empty packages of the same size as the initially delivered ones with a single depot and a fleet of uniform vehicles with limited capacities. The problem is kept stripped from many of the attributes listed above in order to attain simplicity. With its current content and configuration, it is named as the vehicle routing problem with simultaneous pick-ups and deliveries (VRPSPD).

2.3.1. Problem Definition

The graph theoretical definition of the VRPSPD problem is as follows:

Instance: A graph $G = (V, E)$, edge weights w_e for all $e \in E$ and vertex weights d_v and p_v for all $v \in V$, a distinguished node, depot-d and a parameter either given or not k for upper limit to the number of vehicles available, and a parameter C denoting uniform capacity of each of the trucks.

Objective: Find a partition of the nodes in $V\backslash\{d\}$ to $V_1, \ldots, V_k$ and a subset of edges $T_k \subseteq E$ forming k tours each containing node d and each node of V_i exactly once, so *that* $\sum_{e \in Tk} w_e$ is minimized without violating $\sum_{j \in Vh} d_j \leq C$, $\sum_{j \in Vh} p_j \leq C$ for $h \in \{1, \ldots, k\}$ and $p_{vt}^* + d_{vt}^* + p_v \leq C$ for $v \in V$, $t \in \{1, \ldots, k\}$, p_{vt}^* denoting all the load picked up at some partition V_t prior to some definite node $v \in V_t$; d_{vt}^* denoting all the load to be delivered at some partition V_t after some definite node $v \in V_t$.

To our knowledge, there has been little attention to the VRPSPD. This problem is first introduced to the literature by Min (1989). In his work, Min studied book distribution and recollection activity between a central library and 22 remote libraries at a county in Ohio. Each and every day, a central depot is responsible for supplying remote libraries with ordered books and recollecting previously delivered books from them in return. There are two trucks, which are assigned for this distribution and re-collection activity, with limited capacity of 10500 pounds each. Thus, capacity invasion of books is given in pounds. The article also supplies the cost matrix in terms of distances between these libraries. The cost matrix is symmetric.

Halse (1992) studies this special case VRPSPD problem as well as many others in the VRP literature. In the work, cases with a single depot and multiple vehicles and number of nodes varying between 22 and 150 are studied.

Gendreau, Laporte and Vigo (1999) study the VRPSPD for a single vehicle case. They derive 26 problem instances based on some formerly published instances and they test the performance of their two newly developed heuristics with previously introduced in the VRP literature. In their problem instances, the number of nodes vary between 6 and 261 including the depot.

Dethloff (2001) also studies the VRPSPD problem. In his work, he develops 40 instances to test his algorithm. He also reports an improvement on the solution published by Min (1989). Then, he compares the results of his algorithm with those found by Salhi and Nagy (1999), based on their problem instances and problem structure. In the problem structure in Salhi and Nagy (1999), nodes are separated into disjoint delivery or pick-up nodes with 0 distance vector in between and they are provided either delivery or pick-up service, but not both at the same instant. Thus, a node may be visited more than once when the coupling of nodes in the solution is collapsed into single ones. Besides, the problem puts a limit on the maximum route length and introduces multiple depots rather than single depot case.

2.3.2. Mathematical Formulation of the Problem

The notion and the mathematical formulation of VRPSPD is as follows (Dethloff, 2001):

Sets

J: Set of nodes

J_0: Set of nodes including the depot such that $J_0=J \cup \{0\}$

V : Set of vehicles

Parameters:

C: Vehicle capacity

c_{ij}: Distance between nodes $i \in J_o$, $i \neq j$, $c_{ii} = M$, $i \in J$, $c_{00} = 0$

D_j: Delivery amount of customer $j \in J$ from the depot

n: Number of nodes, i.e., $n = |J_0|$

P_j : Pick-up amount from customer $j \in J$

M: Large number, e.g. $M = \max\left\{ \sum_{j \in J} (D_j + P_j), \sum_{i \in J_0} \sum_{j \in J_0, j \neq i} C_{ij} \right\}$

Decision Variables

l'_v: Load of vehicle $v \in V$ when leaving the depot (which can be eliminated from the model)

l_j: Load of vehicle after having serviced customer $j \in J$

π_j: Variable used to prohibit sub-tours (which can be interpreted as position of node $j \in J$ in the route)

x_{ijv}: Binary variable indicating whether vehicle $v \in V$ travels directly from node $i \in J_0$ to node $j \in J_0$ ($x_{ijv} = 1$) or not ($x_{ijv} = 0$)

Model

$$\text{Minimize } z = \sum_{i \in J_0} \sum_{j \in J_0} \sum_{v \in V} c_{ij} x_{ijv} \qquad (1)$$

Subject to

$$\sum_{i \in J_0} \sum_{v \in V} x_{ijv} = 1 \qquad j \in J \qquad (2)$$

$$\sum_{i \in J_0} x_{isv} = \sum_{j \in J_0} x_{sjv} \qquad s \in J,\ v \in V \qquad (3)$$

$$l_v^{'} = \sum_{i \in J_0} \sum_{j \in J} D_j x_{ijv} \qquad v \in V \qquad (4)$$

$$l_j \geq l_v^{'} - D_j + P_j - M(1 - x_{0jv}) \qquad j \in J,\ v \in V \qquad (5)$$

$$l_j \geq l_i - D_j + P_j - M\left(1 - \sum_{v \in V} x_{ijv}\right) \qquad i \in J,\ j \in J,\ i \neq j \qquad (6)$$

$$l_v^{'} \leq C \qquad v \in V \qquad (7)$$

$$l_j \leq C \qquad j \in J \qquad (8)$$

$$\pi_j \geq \pi_i + 1 - n\left(1 - \sum_{v \in V} x_{ijv}\right) \qquad i \in J,\ j \in J,\ i \neq j \qquad (9)$$

$$\pi_j \geq 0 \qquad j \in J \qquad (10)$$

$$x_{ijv} \in \{0, 1\} \qquad i \in J_0,\ j \in J_0,\ v \in V \qquad (11)$$

In the model above, the objective function (1) aims to minimize the total travel distance. Constraints (2) assure servicing each node exactly once. Constraints (3) assure that if a vehicle arrives at a customer, then the same vehicle must also leave it. Initial vehicle loads are determined by constraint set (4), while the initial loads after serving the first customer are defined with constraint set (5). The constraint set (6) introduces limits for vehicle loads "en route." The constraints (7) and (8) ensure load amount of a truck stay under the capacity limits. Constraints (9) are sub-tour elimination constraints and (10) are the related non-negativity constraints.

A relaxation of the VRPSPD may be obtained by separating pick-up and delivery processes such that at any node, either pick-up or a delivery occurs. This relaxation has

been commented to be at least as hard as the NP-hard problems to solve (Mosheiov, 1998). Thus, the VRPSPD is also NP-hard in the strong sense.

2.4. Optimal Algorithms for VRP

The exact algorithms have proved to reveal deteriorating results with exponentially increasing solution times as the size of the problem increases or additional constraints are introduced, folding the complexity of the problem. Although, in the VRP literature, some results have been published with good performance, even reaching to optimality, the bound on the number of nodes for such problems is usually shallow, not exceeding 60 nodes. Various researchers studied these methods in case of the VRPSPD identical to ours or in other cases of the VRP. In what follows is the discussion of three exact methods in the literature.

2.4.1. Dynamic Programming

In dynamic programming problems, there is a given initial state of the system, x_o and discrete time dynamic system of N stages. The system takes value x_k at the k^{th} stage of the problem such that x_k is a member of a given finite set. During the k^{th} stage, the state of the system changes from x_k to x_{k+1} according to an equation of the form

$$x_{k+1} = f_k(x_k, u_k) \tag{12}$$

where u_k is a control that takes values from a given finite set, which may depend on the index k. This transition involves a cost $g_k(x_k, u_k)$.The final transition from x_{N-1} to x_N, involves an additional terminal cost $G(x_N)$. The functions f_k, g_k, and G are given.

Given a control sequence $(u_0,\ldots,u_{N-1})$, the corresponding state sequence $(x_0,\ldots,x_N)$ is determined from the given initial state x_o and the system of equation (12). The objective in dynamic programming is to find a control sequence and a corresponding state sequence such that the total cost

$$G(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \tag{13}$$

is minimized (Bertsekas, 1998).

The application instances of the dynamic programming in the VRP literature are abundant. Dethloff (2001) utilizes dynamic programming to calculate net savings attainable when imbedded the future steps and the course of actions to follow during those steps. The aim is to keep higher residual capacities on the vehicles to attain higher freedom for future servings of nodes while dealing with a current node. Higher residual capacities can be achieved by serving customers with a small (large) delivery amount and large (small) pick-up amount late (early) in the route. Each of those residuals is more advantageous if it is valid for a long part of the route. Additionally, the residual values are prospectively more advantageous if a higher cumulative demand for delivery and pick-up of the yet unrouted customers for future insertions exists. The formulation by Dethloff (2001) is left to the reader's inquiry and will not be mentioned here.

2.4.2. Lagrangean Relaxation Based Methods

Lagrangean Relaxation (LR) based methods have been widely used for obtaining a tight lower (upper) bound for the minimization (maximization) type problems. However, it requires a number of iterations and preferred to be used beside some heuristics or algorithms like the branch-and-bound method. The notion behind is explained as follows: Consider a simple linear programming problem instance with a linear cost function, linear side constraints and integer decision variables. Such a problem's representation would be as follows in the most simplistic way:

$$\text{min} \qquad ax \tag{14}$$

$$\text{subject to} \qquad c_t x \le d_t \qquad t = 1,\ldots, r \tag{15}$$

$$x_{ij} \in X_{ij} \qquad \forall (i,j) \in A \tag{16}$$

where a and c_t are given vectors, d_t are given scalars, and each X_{ij} is a finite subset of integers. In the LR approach, the side constraints $c_t x \le d_t$ is eliminated by adding it to the cost function multiplied by a vector of nonnegative scalars $\mu_t = (\mu_1, \ldots, \mu_r)$, thereby the cost function becomes

$$L(x,\mu) = ax + \sum_{t=1}^{r} \mu_t (c_t x - d_t) \,. \qquad (17)$$

μ_t is a Langrangean multiplier which may be viewed as a penalty per unit violation of the corresponding side constraint $c_t x \le d_t$.

A key idea of LR is that regardless of the choice of μ, the minimization of the $L(x, \mu)$ over the set of remaining constraints yields a lower bound to the optimal cost of the original problem (Bertsekas, 1998).

The main difficulty associated with LR is represented by the cardinality of the relaxed constraints, which does not allow for the explicit inclusion of all of them in the objective function. To overcome this difficulty Toth and Vigo (2002) propose to include only a limited set of the relaxed constraints and iteratively add to the LR constraints which are violated by the current solution of the Langrangean problem. Beside this mechanism, to avoid complexity of the objective function, they also propose to purge the relaxed constraints from the LR in case they become slack by the current solution. This process is iterated until no violated constraints are detected (hence, feasibility is attained) or a prefixed number of subgradient iterations have been executed.The model of Cordone and Calvo (1996) is based on a hierarchical objective function where the main objective is the minimization of the number of vehicles and the second objective is minimizing the total distance traveled.

2.4.3. Column Generation Based Methods

Column generation has turned out to be an efficient method for a range of vehicle routing and scheduling problems. The notion behind is to avoid enumerating all variables (columns) to get a feasible solution to a given problem. Column generation is based on the idea of initializing the linear program with a small subset of variables (by setting all other variables to 0) and computing a solution to this reduced linear program. Given a feasible basis for an LP, the question of whether it is optimal or not is answered through checking all the reduced costs of the decision variables. For optimality case, reduced costs should satisfy non-negativity. This second part of the problem is also known as the pricing problem. If any columns are found with negative reduced costs,

they are included in the basis and solved. This iterative process continues until all reduced costs are found to be non-negative or until a predefined number of successive iterations. The column generation used together with branch-and-bound is denoted as Branch-and-Price algorithm.

Halse (1992) utilizes a LR and column generating approach. A cluster first-route second type heuristic is developed in which nodes are first distributed to vehicles and then the problem is solved using 3-opt approach. Angelelli and Mansini (2001) study the VRPSPD with time windows constraints. They implement a Branch-and-Price approach based on a set covering formulation for the master problem. A relaxation of the elementary shortest path problem with time windows and capacity constraints is used as pricing problem. Branch-and-Bound is applied to obtain integer solutions. Angelelli and Mansini (2001) provide further profound guidance about exact algorithms based on column generation and branch-and-price algorithms.

2.5. Approximation Algorithms and Heuristics for VRP

Heuristics or approximate algorithms are designed to quickly find good but not necessarily optimal solutions. For a variety of problems with LP structure, it is easy to devise heuristic algorithms to find primal and dual feasible solutions. Depending on the quality of the solution required, an approximate solution may be the final answer for a particular problem or may be an input of an exact algorithm. One invaluable contribution of the heuristics may be in cases of solution by branch-and-bound method where they may provide lower and upper bounds in reducing the effort necessary to iterate the whole optimality tree (Nemhauser and Wolsey, 1989).

Though it is difficult to describe completely general heuristic algorithms, Nemhauser and Wolsey (1989) pinpoint three ideas, which are applicable in a wide variety of classes. The first is that of a greedy, alternatively called a steepest ascent/descent or myopic, algorithm.

Greedy algorithms are frequently applied to maximization of set functions. Let $v(Q)$ be a real valued function defined on all subsets of $N = \{1,\ldots, n\}$ and consider the problem $\max\{ v(Q): Q \subseteq N\}$. Then, assume set Q^t is an instance of Q such that it satisfies $Q \subseteq N$. Then, the next element to be added to Q^t should be the one that gives the greatest immediate increase in value, provided that it exists. Moreover, once an element is chosen, it is kept throughout the algorithm.

The second idea highlighted is that of local search or interchange heuristics. As the name implies, a heuristic of this type takes a given feasible solution and, by making only limited changes, tries to find a better solution. Such algorithms are mainly studied under the title of route improving heuristics.

The third general principle is to utilize primal and dual heuristic solutions in pairs. It is particularly desirable to find both primal and dual feasible solutions since the dual solution provides an upper limit for the deviation from optimality of the primal solution (Wolsey and Nemhauser, 1989). Iterating the primal-dual couple associatively, i.e, developing and testing one's solution based on the solution of the other, one may reach a solution to a complex problem with a reasonable quality if not the optimum itself.

2.5.1. Construction Algorithms

Construction type heuristics aim to provide feasible as well as reasonably good solutions to complex problems. These sorts of heuristics are grouped into sequential or parallel heuristics. In the sequential case, a route is initially constructed and the remaining ones are constructed whenever necessary. On the contrary, in the case of parallel heuristics, many routes are constructed simultaneously following a predefined set of rules.

The well-known sweep heuristic (Gillett and Miller, 1974) and the savings heuristic (Clarke and Wright, 1964) are the most cited cases of the sequential construction algorithms. In the former one, routes are constructed as an angle sweeps the location of nodes on a 2D space, while in the latter case routes are constructed in a predefined quantity, then new nodes are added to currently available nodes in order to

attain maximum savings in the total distance covered. Assume that node 0 denotes the depot and nodes $N = \{1,\ldots, n\}$ denote the nodes such that d_{0i} for $i \in N$ denotes the distance from the depot to node i, d_{j0} and d_{ij} the distances from the depot to node j and the distance between nodes i and j for $j \in N$. The savings amount s_{ij} when the arcs between the depot and node j, node i and depot are replaced with a single one between nodes i and j will be as follows:

$$s_{ij} = d_{i0} + d_{0j} - \gamma d_{ij} \tag{18}$$

where γ represents some type of coefficient emphasizing or penalizing the savings attained.

The first algorithm presented by Gendreau, Laporte and Vigo (1999) propose to construct a sequence and serve nodes with positive demands (they describe positive demand as the case when the pick-up quantity is greater than the delivery quantity) until a violation on the residual capacity of the truck during handling the next positive demand customer occurs. Then they quit to serve the customer and begin to serve the next available customer with negative demand. When there is enough room available to serve the next customer, the node where the former capacity violation occurred is returned and the next following node with a positive demand is served.

Building routes sequentially may cause latterly constructed routes to be of poor quality since there are only few alternative points of insertion at the latter iterations of the process. This can be overcome partially by constructing parallel routes. Potvin and Rousseau (1993) propose a parallelization of the Insertion Heuristics by creating many routes simultaneously. For the initialization of each route, the customer that is farthest from the depot is selected as a "center customer." Then the customers are inserted to the best feasible insertion place.

Lacomme *et al.* (2001) emphasize that most Genetic Algorithms (GA) for the TSP instances use permutation chromosomes as constructive heuristics. For the capacitated arc routing case they study (a special type of TSP), a chromosome could be viewed as the order in which the vehicle must perform n tasks, assuming that a single vehicle performs all trips in turn. This encoding type is found appealing because it always has

an optimal sequence. However, that one great trip may be divided into sub-trips considering all the inevitable trip delimiters such as the capacity constraints.

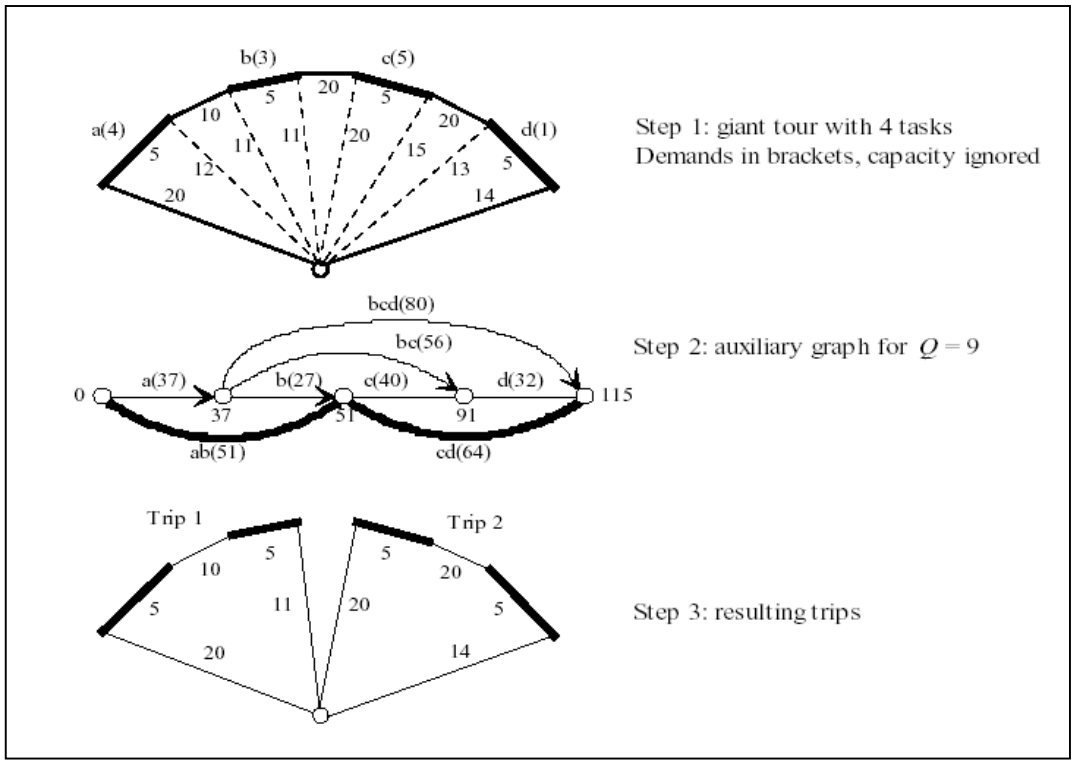

Figure 2.1 Main steps of Ulusoy's algorithm (Lacomme *et al*., 2001)

Ulusoy's algorithm (1985) provides an elegant solution for keeping simple permutation chromosomes. In the capacitated arc routing case, a fleet of vehicles are assigned routes and they are supposed to serve the arcs rather than edges en route, which creates a slight difference from the classical vehicle routing cases. In the algorithm, first the capacity constraints are ignored and a one giant tour *T* covering all the tasks (a,b,c, and d in Figure 2.1) is formed. Second, an auxiliary graph is built in which each arc denotes a subsequence of *T* that can be done by one single trip. Each arc is weighted by the cost of that particular trip constructed. A shortest path in this graph shows where to split *T* into trips and gives the cost of the corresponding solution. At last, the solution is built with one trip per arc of the path. The steps of the algorithm are visualized in Figure 2.1. In our work, the first step is followed and then the giant tour is partitioned with iterating each node's servicing requirements, their comparison with the current vehicle's available capacity, if any violation is detected closing the current vehicle, sending it back to depot and consequently opening a new one and servicing the

first customer a capacity violation is faced until all the nodes are served. This partitioning scheme is a sort of "Iterated Tour Partitioning Heuristic," which is introduced first by Haimovich and Rinnooy Kan (1985) and studied for a special case of pick-up and delivery type vehicle routing problems by Mosheiov (1998). For deeper interest and worst-case analysis of further constructive heuristics, one may refer to the Anily and Bramel (1999).

2.5.2. Route Improvement Heuristics

Route improvement heuristics strive for constructing a better tour starting from a considerably poorer performing one. Such an activity is basically classified as neighborhood search in the optimization literature. Neighborhood search notion has been widely used in the combinatorial optimization problems and especially in TSPs for more than forty years.

The neighborhood of some point is the region of the search space that is "near" some particular point in that space. Consider some abstract search space S together with some particular point $x \in S$. The intuition is that a neighborhood $N(x)$ of x is a set of all points of the search space S that are close in some measurable sense to the given point x.

Many search methodologies are based on the statistics of the neighborhood around a given point; that is, the sequence of points that these techniques generate while searching for the best possible solution relies on local information at each step along the way. These techniques are designed to locate solutions within a neighborhood of the current point that have better corresponding evaluations. Appropriately, they are known as "neighborhood" or "local search" strategies (Michalewicz and Fogel, 2000).

There are two categories of local optimization methods that are used to improve the routes, namely intra-route and inter-route. The intra-route local optimization rearranges the order in which the customers are visited to decrease the total distance traveled by the vehicles. The inter-route method exchanges or moves customers between two routes in order to improve the overall quality of the solution (Thangiah and Petrovic, 1998).

For both inter-route and intra-route optimization methods, there are few number of methods preferred by a wide range of researchers. These are the *k*-opt (Croes, 1958), Or-opt (Or, 1976), 1-0 and 1-1 exchange moves (Waters, 1987).

The *k*-opt heuristic replaces a set of links in the route by another set of *k* links in order to attain a gain in the overall cost function. The complexity of the heuristic is mostly affected by the size of *k*. For larger *k* values, the heuristic tends to give better results, but the computational time increases. The most preferred *k* values are usually 2 or 3. The 2-opt checks two paths, 3-opt checks 8 different paths and 4-opt would check 48 different paths. As *k* reaches the total number of customers in a route, then the improvement procedure leads to an exact iterative search procedure. The Or-opt procedure is a modification of the 3-opt procedure provides a reduction in the computational effort required. The Or-opt only considers a subset of the paths considered for 3-opt that results at most three adjacent customers being inserted between two other customers. In other words, Or-opt removes a chain of at most three customers and relocates them in the remaining chain satisfied that the constraints are not violated and maximum savings is attained. The 1-1 exchange move procedure is swapping arcs between two nodes from the same route. The same procedure is constructed in the case of multiple routes but the swapping of connectors between nodes takes place between two different routes. The 1-0 exchange move transfers a node from its current position in one route to another position in either the same or a different route. Figure 2.2 illustrates examples for a 2-opt move, 1-1 exchange, and 1-0 exchange procedures.

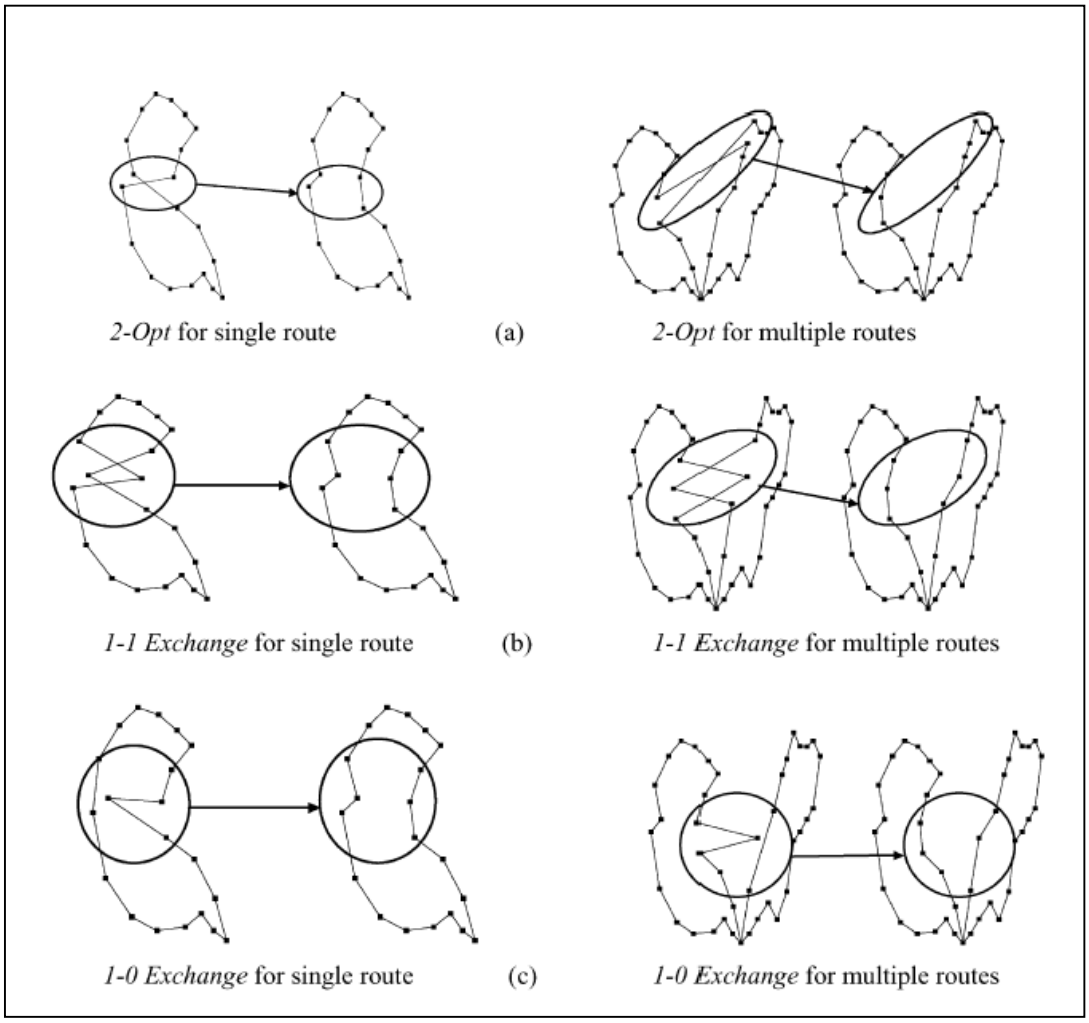

Figure 2.2 2-Opt, 1-1 exchange, 1-0 exchange moves for single and multiple routes (Tarantilis et al., 2002)

In this study, Or-opt local search method based on movement of a single node is utilized. Our work yet comprises two main steps, the constructive step which establishes routes and the finalizing step which reduces the total length of found routes using local search methods, similar to Or-opt. In our work, one single giant tour is established first, and then it is partitioned based on the residual capacities on the vehicles as well as the empty spaces available to accommodate the current node's pick-up requirements. Soon after the initial routes are established, each node's location within a route is checked for a better replacement that will reveal a saving in terms of total distance. Each node is checked for replacing in a location succeeding its current location until the route end, and replaced at a location that will reveal the utmost positive saving if there exists any. Thus, our local search mechanism resembles the Or-opt process with a single node.

2.5.3. Meta-Heuristics

In recent years, there have been significant advances in the theory and applications of meta-heuristics for complex optimization problems. A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high quality solutions. It may combine intelligently different concepts to explore the search space using adaptive learning strategies and structured information (Osman, 2002). Meta-heuristics are general combinatorial optimization techniques, which are designed with the aim of being flexible enough to handle as many different combinatorial optimization problems as possible rather than handling a certain type or configuration of a specific problem. For a good introduction on meta-heuristics, we refer the reader to Osman (1995), Osman and Kelly (1996), and Golden et al. (1998).

For the last two decades, much of the research effort has been devoted to development of meta-heuristics, using mainly two principles: local search and population search. In local search methods, an intensive exploration of the search space is performed by moving at each step from the current solution to another promising solution in its neighborhood. Simulated annealing (SA) and tabu search (TS) are the most popular local search methods. Population search consists of maintaining a pool of good solutions and combining them in order to produce hopefully better solutions. Classical examples are GAs and adaptive memory procedures (Hertz and Widmer, 2003).

Hertz and Widmer provide guidelines for success in adaptation of a meta-heuristic to a combinatorial optimization problem. These basic principles are grouped into two for local search and population search methods. The ones for the local search methods are listed as follows:

- *It should be easy to generate solutions in the solution space of a particular problem*. For the case of NP-hard problems, the search space should be defined by relaxing some constraints of the original problem, and adding some penalty coefficients in the cost function against violation of constraints.
- *For each solution instance in the solution space, there must exists an improvement path available for the meta-heuristic to advance which links the current solution*

instance to an optimal solution instance. If such a condition is not satisfied, then an optimal solution will never be reached.

- *The solutions in the neighborhood of a solution instance should be in some sense close to that particular solution*. It is important to define neighborhoods so that a reasonable solution may be generated in a reasonable time and effort. Taking neighborhoods great in content may lead to a burden of solving the original problem instead. The neighborhood of an instance should be easily obtained by performing simple modifications on the original solution instance.
- *The topology induced by the cost function on the solution set should not be too flat*. It is difficult for a local search to escape from large plateaus since any solution in the boarder of such a plateau should have the same cost value as its neighbors, thus it becomes impossible to guide a search towards an optimal solution. A common way to avoid such cases is to add some components in the cost function, which will discriminate between solutions with same objective function values.

The guidelines addressed for the population search methods are listed as follows:

- *Pertinent information should be transmitted during the co-operation phase*. In the co-operation phase, groups of individuals exchange pieces of information and new offspring solutions are created that should combine the best features of the parent solutions.
- *The combination of two equivalent parent solutions should not produce an offspring that is different form the parents.*
- *Diversity should be preserved in the population*. One of the major difficulties observed when using population search algorithms is the premature convergence of the process, all solutions in the population having a natural tendency to become equal to the best solution in it. If this occurs, then the population search behaves more or less like a local search since there is nothing to gain in combining equivalent solutions. In order to prevent such a phenomenon, it is important to implement operators that preserve diversity in the population. The mutation operator in genetic algorithms is an example of such a tool.

Next, the most commonly used three search meta-heuristics are briefly described: SA, TS and GA.

2.5.3.1 Simulated Annealing

In order to alleviate the problem of becoming trapped at a local optimum, SA allows the selection of some uphill moves in a controlled manner. Let S be the current solution and $N(S)$ be a neighborhood of S. We randomly select $S' \in N(S)$ and compute the difference $D = f(S) - f(S')$, where $f(S)$ is the objective function value of S. If $D < 0$, then S' is is selected as the new solution. If $D > 0$ and $e^{-D/T} > q$ (where $0 < q < 1$ is randomly generated from a uniform distribution), then S is selected as the new solution. T is simply a control parameter known as the temperature. Typically, the temperature is gradually lowered during a search process so that the probability of accepting an uphill move (i.e., $D > 0$) steadily decreases. The SA procedure continues until a stopping condition is met (Golden *et al*., 1998). For further details the interested reader is referred to the Osman and Kelly (1996) and Reeves (1993).

2.5.3.2 Tabu Search

TS aims to avoid getting trapped at a poor local minimum, by accepting on occasion a worse or even infeasible solution from within the current solution (Bertsekas, 1998). The main idea behind TS is quite simple. A "memory" forces the search to explore new areas of the search space. We can memorize some solutions that have been examined recently and these become tabu (forbidden) points to be avoided in making decisions about selecting the next solution (Michalewicz and Fogel, 2000).

The memory concept of the TS is quite crucial. Golden *et al.* (1998) define two types of memory: short-term and long-term memory. Usually, short-term memory is imposed to restrict the search from revisiting solutions that were considered previously

and to discourage the search from cycling between subsets of solutions. On the other hand, long-term memory is used to diversify the search. Usually, it is implemented to discourage frequently made moves and to encourage the search process to explore new regions of the solution space. One other element of the TS is the aspiration criterion, which is defined as the overriding mechanism of the short-term and long-term memory functions.

Golden *et al.* (1998) provide an extensive survey of academic publishings in the tabu search while Michalewicz and Fogel (2000) provide some basic pseudo-codes.

2.5.3.3 Genetic Algorithms

The concept of GAs for solving optimization problems is based on the analogy to evolution theory in population genetics. Holland (1975) adopted the idea of survival of the fittest in a process of cooperation and competition among individuals to combinatorial optimization problems. The solutions of a problem are coded into chromosomes, a sequence of genes. A set of such chromosomes is called a population. Starting from an initial population new chromosomes are generated by standard genetic reproduction operators, crossover and mutation, and are evaluated with respect to a problem specific fitness function (Derigs *et al*., 1999).

Derigs *et al*. (1999) divide the genetic mechanism into two distinct stages. The static stage comprises definition of a coding scheme and developing an appropriate fitness function capturing the main objectives and constraints for the given problem. Any GA incurs a wide range of parameters such as the population size, deviation structure, and termination conditions. These parameters are adjusted at the static stage. The dynamic stage is divided into four phases, which are iteratively applied until a termination condition is satisfied. It is this dynamic stage's resemblance to the evolution in nature, which names this population search type meta-heuristic. These phases are listed as follows:

- Selection phase: A number of individuals of the current population are selected and paired for reproduction.
- Reproduction phase: Applying the principle genetic reproduction operators like crossover and mutation new solutions are generated by sexual reproduction.
- Integration phase: The new individuals are evaluated according to the defined fitness function. Then, it is decided which of these offsprings will be integrated into the new population and which older individuals will be excluded form the older population.
- Control phase: Global metrics of the population are assessed and the communication scheme is updated. The algorithm checks if the termination condition holds.

For to define a GA, beside the iterating structure, a set of operation parameters must be supplied. Usually, the hardest part of the work is to determine these operation parameters. The current practice is to make as many experiments as possible with all combination of a set of predefined parameters and consequently addressing the best performing combination, or applying it to a set of problems. However, current efforts in GA are to develop parameter free structures or reducing parameters to the minimum possible levels. For further knowledge in the GA literature the reader is referred to the Gen and Cheng (2000) and Reeves and Rowe (2003).

3. A DUAL GA APPROACH FOR THE VRPSPD

There have been no efforts detected to solve the VRPSPD or its relaxed cases using the GAs. Thus, the genetic representation and model infrastructures are mainly searched through the machine shop scheduling with multi machines literature, which offers an adequate amount of instances.

The two problems are fairly identical in the fashion that in both cases some items are distributed to some scarce resource, which are the vehicles in the VRP case and the machines in the machine scheduling case. The capacity constraint imposed by the capacitated vehicle routing problem (CVRP) is replaced by the total available time span of the machine and the resource invasion of each of the loads in the CVRP case is replaced by the time consumption of each of the jobs to be scheduled on the machines. The two way constraints (the residual capacity constraints on deliverable goods and empty space constraints availing picking of loads) imposed by the simultaneous delivery and pick of loads in the VRPSPD case could not be substituted with any of the current practices in the machine shop scheduling literature which are usually one way (available time span constraint of the machine). Thus, continuous two-way check of the available capacity is added in this problem.

The resulting problem required an efficient data structure to handle the evolutionary mechanisms' needs with utmost efficiency. In the canonical (binary) representation of genes in the chromosome, some gene sequences become obsolete since they do not provide information correspondending to a meaningful allele (an allele represents the meaning of a gene). For example, when binary strings are required to represent alleles ranging from 1 to 50, some 6 digits are required in the string. However, the strings representing more than 50 correspond to nothing meaningful. Thus, if in a gene a string sequence greater than 50 ever occurs following a crossover operation, then that gene must be iterated once more, which adds up to the complexity and solution

time of the algorithm. Seeking methods to avoid both deficiencies, our method has to be quite efficient. In this research, rather than the binary representation, integer labels are assigned in addressing the alleles of the genes.

One important feature in the GA heuristics is that maintaining the sequence feasibility through the crossover operations. This is mainly keeping the chromosome structures healthy and meaningful right after a reproduction process. In many cases, after the reproduction phase, one may see replication of genes in one parent chromosome while the other parent lacks those in its sequence. This is a crucial mishap if the sequence of genes on the chromosome means one great tour sequence like the one in Figure 2.1. Then replications of genes in a chromosome mean revisiting of some nodes, which is an undesirable outcome. Besides, omitting some genes in a chromosome means no service to those nodes during the great sequence. Fixing those undesirable faults requires extra search efforts, increases computational time and introduces ample amounts of computer codes, which gets harder to trace. Thus, an efficient method should never let replications or omitting of genes on the chromosome structures. The two heuristics that we have studied well satisfy this requirement.

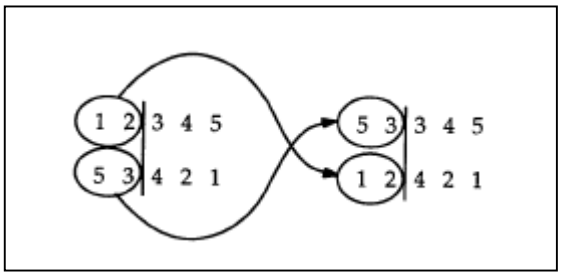

Figure 3.1 An example of replications and omitting of genes through crossover (Bean, 1994)

Through our literature search, two models seemed to provide utmost computational efficiency and coding ease beside their characteristic properties. In an effective GA, two main opposite goals must be satisfied at the highest levels. These are "exploration," which means exploration of the solution space as much as possible, and "exploitation," which means exploiting the inherited memory from the previous iterations to the success of future iterations. Thus, a slight contradiction between the aims of these two goals may be that exploitation requires continuing the search within the small neighborhood of a solution, while exploration requires the neighborhood

widened. The following reveals some different characteristics in terms amount of exploitation as well as exploration they provide.

3.1. Random Keys Method

This method by Bean (1994) mainly focuses to overcome the difficulty of maintaining offspring feasibility through the genetic crossover operations. This method proved to be effective in multiple machine scheduling, resource allocation, and quadratic assignment type problems. However, no application to a VRP type problem has been noticed in the literature.

The random keys representation encodes a solution with random numbers. These values are used to sort keys to decode the solution. Random keys eliminate the offspring feasibility problem by using chromosomal encoding that represents solutions in a soft manner. These encodings are interpreted in the objective evaluation routine in a way that avoids the feasibility problem.

The primary difference between this encoding and those others is the use of random numbers as tags to represent solutions. Random numbers are sampled from some space, typically $[0, 1]^n$. The genetic algorithm searches that space as a surrogate for the literal space. Points in the random keys space are mapped to points in the literal space for evaluation. For this reason, the random keys approach differs the binary encodings. The generation of random numbers in the keys space employs a sense of random search in conjunction with the GA. One advantage of this encoding is its robustness to problem structure (Bean, 1994). In the paper, the algorithm is tested in three different problem instances. For its simplicity and ease to illustrate the overall mechanism, the case for the single machine scheduling will be discussed here.

Consider a problem that requires sequencing of five jobs on a machine. The consecutive chromosome structure will be composed of genes, each of which will correspond to a job to be scheduled. In order to form an instantiation, generate a

uniform (0, 1) random deviate for each allele. Then the chromosome structure would be as follows:

$$(.46, .91, .33, .75, .51) \tag{19}$$

The mapping to the literal space is accomplished by sorting the alleles and sequencing the jobs in the ascending order. The sequence would be then:

$$3 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 2 \tag{20}$$

This sequence may easily be evaluated for the tardiness or total work time it introduces. Crossovers are executed on the chromosomes –the random keys- not on the sequence. Consider two parents

$$(.46, .91, .33, .75, .51) \equiv 3 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 2 \tag{21}$$

and

$$(.84, .32, .64, .04, .48) \equiv 4 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 1 \tag{22}$$

Using a one-point crossover and performing it right after the second gene, the offsprings would be

$$(.84, .32, .33, .75, .51) \equiv 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 1 \tag{23}$$

and

$$(.46, .91, .64, .04, .48) \equiv 4 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow 2 \tag{24}$$

Since any sequence of numbers can be interpreted as a sequence, all offsprings are feasible solutions. The random keys simply serve as tags, which the crossover operator uses to rearrange the jobs.

The above heuristic is applied to the forty VRPSPD problem instances for a tour of fifty nodes (Dethloff, 2001). As tour partitioning approach, the way described beforehand was employed. However, the solution quality has turned out to be very poor compared to the values published by Dethloff (2001). One reason is the high amount of

deviation on the chromosome structures during crossovers. On chromosomes of fifty nodes, amount of passed data from parents to offsprings seemed to diminish a great deal. Thus, this method proved to be strong in the sense of exploration and providing diversity but weak in the sense of exploiting the inherited memory. The pseudo-code for the random keys procedure is provided in Appendix A.

3.2. Improvement Heuristic

The improvement heuristic proposed is based on the GA structure developed by Topcuoglu and Sevilmis (2002) for a task-scheduling problem with multi objectives. Their problem definition is the optimal mapping selection between tasks and processors so that precedence requirements are satisfied, schedule length is minimized as well as the number of processors employed is tried to be minimized.

In their representation, each chromosome is presented by a single string of a predefined length, which denotes the number tasks to be scheduled. Each gene is a tuple such that one element of the pair shows the task number while the other shows the processor this task is assigned. The ordering of tasks on the chromosome also reveals the precedence relations of tasks on the processors.

The most important feature of the representation is the crossover operation presented in that paper. Their single-point order crossover operator randomly generates a crossover point and cuts the selected pair of chromosomes into left and right parts. Then, it copies the left portion with respect to crossover point of the chromosome from the first parent to the first child, which is then appended with the remaining genes (in the form of tuples) from the other parent in the same order. The second child string is generated in the same way. Figure 3.2 illustrates the crossover operation. This strategy guarantees to produce valid schedules considering the precedence constraints. Implication of this result to the case of VRPSPD is that using some variant of this chromosome representation and this crossover mechanism, the great tours covering all the nodes may easily be constructed and generations may be iterated without any congestion due to replication of genes in the chromosomes or omitting any of those.

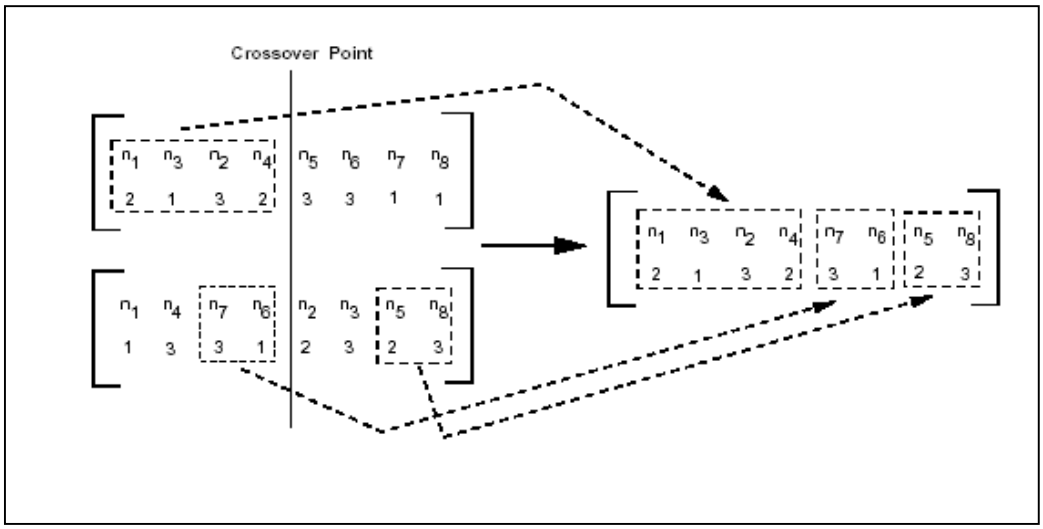

Figure 3.2 Two schedules (chromosomes) and an output of crossover operator applied to these chromosomes (Topcuoglu and Sevilmis, 2002)

This second method has a more potential to pass genetic information to forthcoming generations without dramatic destruction. The reason is that in this second heuristic, during the crossover operation, rather than single relocations of genes in the offsprings, groups of genes are replaced preserving the order they have at the parent chromosome. Thus, inheritage of pertinent information of the parents is more expectable for the offsprings.

Our approach blends the good sides of the both heuristics, such that it strives to exploit the high deviation and initiation procedures as well as the data couple structure of the random keys method and blends it with the crossover mechanism of the second heuristic. The pseudo-code of our heuristic is provided in the Appendix B. Figure 3.3 provides an overall representation of our heuristic.

The first step in our heuristic is to initiate the population. For this purpose random keys method is used. Each gene in our structure consists of two information pieces, the number of the node it represents and the random key assigned. Initialization of the population is done by sorting the genes of each and every chromosome in ascending order of the random keys they comprise. Then after the generations may begin.

Initiate the population by ordering the genes wrt random keys

While Number of generations ≤ Criteria on Generations

For

Partition each giant tour to subtours

Improve subtours by savings based local search (1 node Or-opt)

Evaluate the chromosomes and derive fitness values

End For: each chromosome iterated

Assign matching probabilities wrt fitness values

Replace bad chromosomes with outperforming ones in the same population

Realize crossovers

Realize mutations

End While: Criteria on Generations violated

Figure 3.3 Brief description of the dual GA algorithm

In each generation, the first step is to partition each giant tour (chromosome sequence of genes represent the giant tour) to some sub-routes, which are then assigned a vehicle each. The procedure is iterating the nodes in the chromosome following the sequence they appear. At each gene, two checks are made. The first one is whether the demand of the node may be satisfied with the residual capacity on the truck. The second one is whether the amount to be picked up after the delivery may be accommodated on the truck with the current empty capacity. If both are not violated, then that node is serviced by the vehicle and vehicle advances to the next consecutive node. If any check conditions is violated at that particular node, then the node is left unserviced, the vehicle is sent to the depot and a new vehicle is sent to the unserviced node to satisfy its needs. The fitness function is the total distance covered by all the trucks during servicing all the customers in a giant tour. Then comes the route improvement procedure. Imitating the Or-opt procedure, each sub-tour is taken separate, and then each node in this tour is considered for relocation at an available point following its current location. Again the capacity constraints are considered and maximum positive saving is sought on the current tour at evaluating the location of each node on a particular sub-tour.

The fitness values of each chromosome are determined based on the reduced total tour lengths. Then, these chromosomes are ordered in descending fitness values and the

worst *n* many of them are replaced with the best *n* chromosomes (*n* is a parameter set by the user). Then comes the crossover procedure, which is the same as the one depicted by Topcuoglu and Sevilmis (2002).

The crossover procedure is followed by the deviation procedure. There are three different types of mutations included in our heuristic. The first one is to interchange the locations of two genes on a chromosome with probability values increasing with the gap between the last generation a best solution is found and the current generation. The second type of mutation is altering the random key value of a gene with again a random pattern just similar to the previous mutation. The third mutation is reordering the genes on a chromosome using the random keys procedure. The number of chromosomes to be reordered is determined by two methods. The first one is linearly incrementing the number and the second one is generating Fibonacci numbers beginning from a parameter. Advancing mechanism operates when the difference between the last generation a best solution is found and the current generation divided by some parameter reveals no remainder. The results of comparison of linear proportion to Fibonacci numbers and other parameter tests are analyzed in the following chapter. This overall genetic evaluation procedure is sustained until a violation on the terminating condition of the algorithm is faced.

4. COMPUTATIONAL STUDY

This chapter discusses the parameter setting and computational experiments in comparison with two benchmark studies in the literature.

4.1. Benchmark Problems

The benchmark problems studied in this work is provided form two sources. Min (1989) studies the VRPSPD with the same configuration as ours but imposes a constraint on the maximum number of vehicles available to use, which is strictly two in his problem. His problem studies the case with 22 customers and a single node. Each node requires deliveries and provides some load to be picked and transported to the main depot. The delivery and pick-up quantities are determined from averaging daily transfer quantities and considered to be deterministic. He provides the symmetric distance matrix based on the real geographical measurements and the cost function comprises only the total distance covered.

The second source of problems studied is the one studied by Dethloff (2001). In his work, Dethloff randomly generates test instances with 50 customers and two different geographical scenarios. In the first scenario, SCA, the coordinates of the customers are uniformly distributed over the interval [0, 100]. Scenario CON consists of one half of the customers distributed in the same way as SCA. The coordinates of the other half are more concentrated. They are uniformly distributed over the interval [100/3, 200/3], thus yielding a more "urban" configuration in $1/9^{th}$ of the area. Distances are measured in Euclidean metric in both of the cases.

The delivery amounts of the customers are uniformly distributed over the interval [0,100]. The pick-up demand P_j corresponding to the delivery demand D_j is computed

by using a random number r_j that is uniformly distributed over the interval [0,1] such that $P_j = (0.5+r_j)\, D_j$.

The vehicle capacities are taken into consideration as well. Instances with different vehicle capacities are generated by choosing the minimal number of vehicles required, denoted by μ. Then the corresponding capacity is $C = \sum_{s \in J} D_s / \mu$. In the instance descriptor the digit after the letters for the geographical scenario represents the respective value of μ, which is chosen to be 3 or 8. For each of the resulting configurations 10 random experiments are performed. The last digit of the instance descriptor denotes the number of the experiment (Dethloff, 2001).

4.2. Parameters and Analysis

The static stage of GAs requires existence of some number of parameters to attain a solution quality and sustain controllable evaluation of the process. Robustness requires applicability of a solution heuristic to any kind of problems within the same problem area without further tuning of parameter sets. In order to achieve this, the state of the art trend is to strip a heuristic from any parameters, though it is quite hard. Thus, eliminating as many parameters as possible by replacing them with some self-advancing mechanisms that relies on some parameter or variable is the approach we have followed throughout this research. Through some computational tests, we have reduced the parameters to four.

The parameters that resides yet in the heuristic are described as follows:

- *Discard Quantity*: This parameter defines how many of the worst chromosomes will be discarded and replaced with how many best ones at each generation.
- *Initial Destructive Mutation Quantity (Normmut)*: At each generation, some number of chromosomes is deviated by the destructive mutation process. The upper limit to number of these chromosomes to go under this process is determined dynamically throughout the generations based on the difference between the generation a best solution found and the current solution. However,

this dynamic routine has to initiate from some number and "Normmut" provides this initiation.

- *Population Size*: The population size describes the number of parents that go under reproduction and number of new offsprings generated at each generation of the evolution algorithm. The size of the population is kept still throughout the generations.
- *Increment Size*: To advance the upper limit to the number of destructive mutations that may be performed consecutively at each generation, the difference between the generation a best solution found and the current solution is operated by a mode function. When the result is 0, then the upper limit is advanced some quantity determined by the advancing mechanism. The mode operator value is called as the "Increment size."

One other issue worth mentioning is the total number of generations to be held consecutively. Our current practice is taking it as long as possible in order to watch the improvement of the objective function value through generations. In order to diminish the solution time to some reasonable values, we seek a reasonable value for the maximum number of generations allowed prior to termination condition of the problem. Figure 4.1-4.4 illustrate the improvement process of the solutions of four different families of problem instances developed by Dethloff (2001). The *x* coordinate gives the range of generations from 0 to 50000 replications. The graphs provide insights about when to stop the generations without losing too much from the solution quality. Figures 4.1, 4.2, 4.3 and 4.4 represent each a group of problems' (SCA3, SCA8, CON3, and CON8, respectively) solution result path marked at each 500 generations from the very beginning to the last generation, which is 50000^{th} generation.

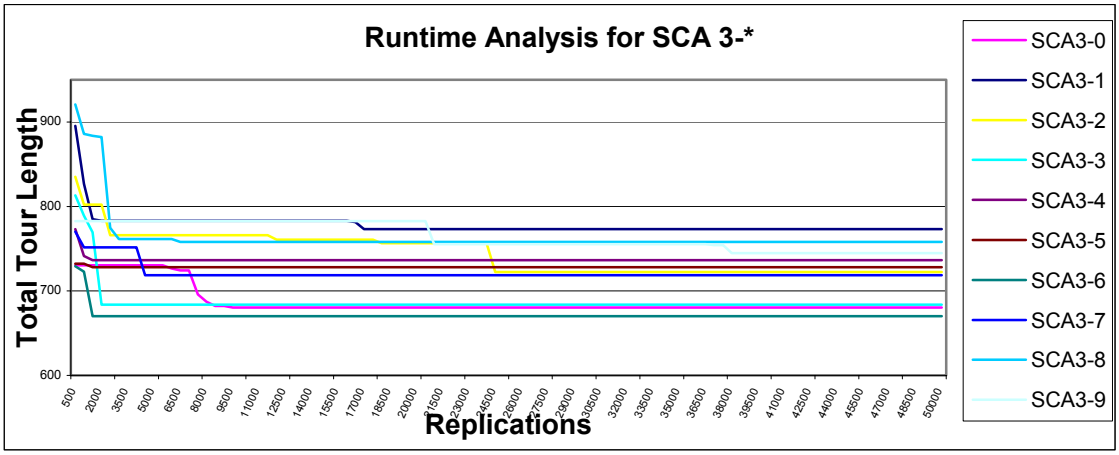

Figure 4.1 Graph of total tour length versus generations for the SCA 3 Problems

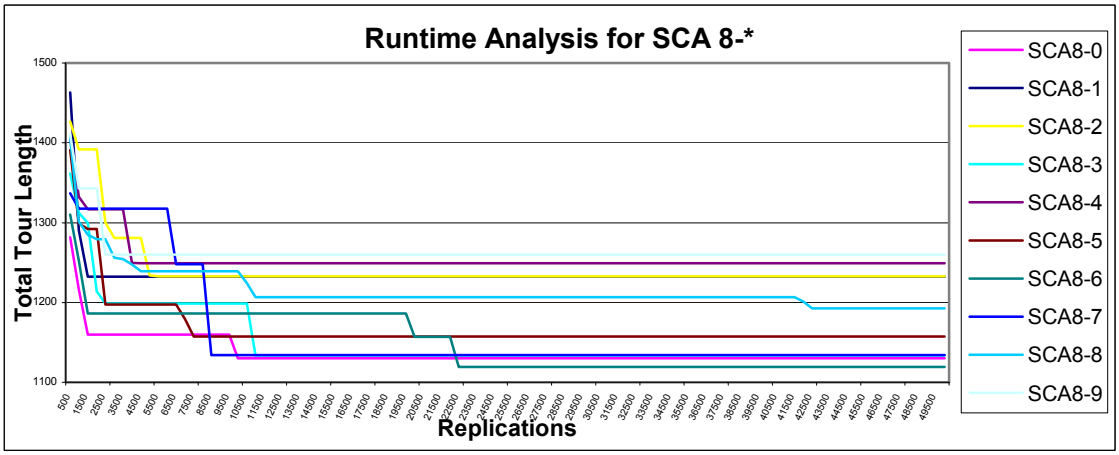

Figure 4.2 Graph of total tour length versus generations for the SCA 8 Problems

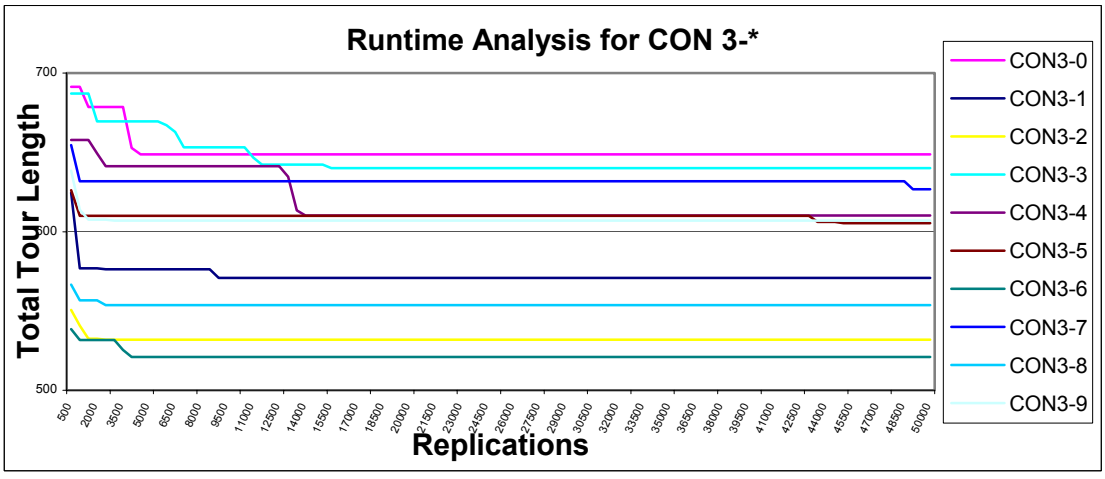

Figure 4.3 Graph of total tour length versus generations for the CON 3 Problems

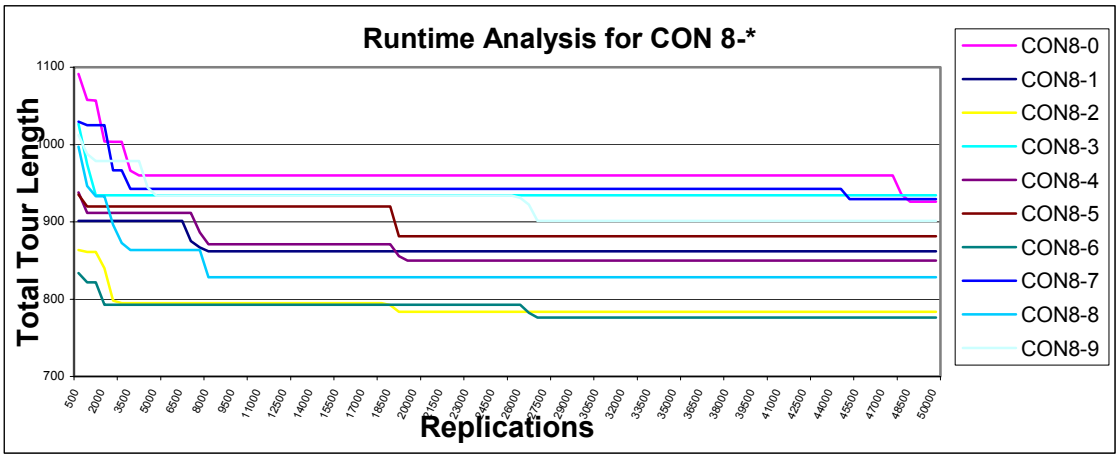

Figure 4.4 Graph of total tour length versus generations for the CON 8 Problems

Extensive tests are performed to obtain the best parameter settings among some number of alternatives generated for each parameter. However, due to high computation duration requirements, all combinations of these parameters could not be iterated. The results revealed to be promising and the best parameter set is found to be (8/200/5/13), with the same order described earlier. The solutions using these parameters are compared to the results of the sample problems in Dethloff (2001) and Min (1989).

4.3. Comparison of Results with the Benchmarks

We have coded and run our algorithm in ANSI C. Min (1989) declares his solution for his problem as 94. He uses a method of clustering the nodes into two groups, then solving a relaxation of the problem using branch and bound, then determining the constraint violations, penalizing moves leading to such violations and then solving the relaxed form of the problem iteratively until no violations of the constraints are faced. Dethloff (2001) declares his best solution to the same problem as 91 using the three heuristics he introduces. He follows a dynamic approach based on savings method by Clarke and Wright (1964) improved by some level of considering future attainable savings at each step. The solution time is declared to be less than a second. Besides, in the paper it is reported that the best-known solution after a hundred hours of computation time using Pentium III 500 Mhz processor is found to be 89, with

a lower bound of 84. Our solution approach provides a value of 88 with a Pentium IV 1.3 Ghz processor in less than a minute.

Table 4.1 Comparison of results to those of Dethloff (2001)

Problem Code	Dethloff's Best	Best Obtained During Parameter Analysis		Results using Parameters (8/5/200/13)	
		Best Soln.	% Gap	Best Soln.	% Gap
SCA3-0	689.0	**640.69**	**-7.01**	**680.25**	**-1.27**
SCA3-1	765.6	**753.11**	**-1.63**	773.18	0.99
SCA3-2	742.8	747.51	0.63	**722.537**	**-2.73**
SCA3-3	737.2	**716.04**	**-2.87**	**683.95**	**-7.22**
SCA3-4	747.1	**698.35**	**-6.53**	**736.329**	**-1.44**
SCA3-5	784.4	**670.44**	**-14.53**	**728.341**	**-7.15**
SCA3-6	720.4	**675.93**	**-6.17**	**670.06**	**-6.99**
SCA3-7	707.9	**711.91**	**0.57**	718.622	1.51
SCA3-8	807.2	**746.21**	**-7.56**	**758.093**	**-6.08**
SCA3-9	764.1	**754.65**	**-1.24**	**744.683**	**-2.54**
SCA8-0	1132.9	**1064.42**	**-6.04**	**1129.25**	**-0.32**
SCA8-1	1150.9	1212.2	5.33	1232.22	7.07
SCA8-2	1100.8	1188.3	7.95	1232.62	11.97
SCA8-3	1115.6	**1098.04**	**-1.57**	1133.826	1.63
SCA8-4	1235.4	**1189.91**	**-3.68**	1249.188	1.12
SCA8-5	1231.6	**1175.52**	**-4.55**	**1157.542**	**-6.01**
SCA8-6	1062.5	1113.39	4.79	1119.119	5.33
SCA8-7	1217.4	**1128.84**	**-7.27**	**1134.23**	**-6.83**
SCA8-8	1231.6	**1178.08**	**-4.35**	**1192.876**	**-3.14**
SCA8-9	1185.6	1273.72	7.43	1259.932	6.27
CON3-0	672.4	**661.23**	**-1.66**	**648.805**	**-3.51**
CON3-1	570.6	**567.58**	**-0.53**	**570.751**	**0.03**
CON3-2	534.8	539.53	0.88	**531.821**	**-0.56**
CON3-3	656.9	**625.81**	**-4.73**	**640.144**	**-2.55**
CON3-4	640.2	**613.06**	**-4.24**	**610.17**	**-4.69**
CON3-5	604.7	**582.28**	**-3.71**	605.357	0.11
CON3-6	521.3	**514.9**	**-1.23**	**521.066**	**-0.04**
CON3-7	602.8	622.69	3.30	626.701	3.96
CON3-8	556.2	**546.75**	**-1.70**	**553.623**	**-0.46**
CON3-9	612.8	623.65	1.77	**606.939**	**-0.96**
CON8-0	967.3	**907.89**	**-6.14**	**926.144**	**-4.25**
CON8-1	828.7	836.59	0.95	862.001	4.02
CON8-2	770.2	779.11	1.16	783.807	1.77
CON8-3	906.7	**884.57**	**-2.44**	934.192	3.03
CON8-4	876.8	**860.33**	**-1.88**	**849.79**	**-3.08**
CON8-5	866.9	881.32	1.66	881.473	1.68
CON8-6	749.1	**729.87**	**-2.57**	776.08	3.60
CON8-7	929.8	**904.31**	**-2.74**	**929.369**	**-0.05**
CON8-8	833.1	**825.15**	**-0.95**	**828.567**	**-0.54**
CON8-9	877.3	892.76	1.76	901.243	2.73

The results obtained during the parameter test and with specified parameter set are listed in the Table 4.1. The first column denotes the problem name, the second column the best solutions presented by Dethloff (2001), and the third column denotes the best found solutions during parameter testing with the percent gap compared to Dethloff, and the last column the results with our last parameter set again with the percent gap. The percent gap is calculated as

$$\{Dual\ GA - Dethloff's\ Best\} / Dethloff's\ Best \quad (25)$$

The bold face values denote the results which are better than those found by Dethloff (2001). As seen on the table, Dual GA algorithm with parameters (8/5/200/13) performed better than Dethloff's Best in 16 problem instances out of 20 with 3 vehicles case and 8 problem instances out of 20 with 3 vehicles case.

5. CONCLUSION

Our work studied a very specialized case of the classical VRP, which is the VRPSPD. Our problem differs mainly from the rest of the literature with its capacity constraints. These require simultaneous pick-ups and deliveries of loads of the same size from the depot to the customers and from the customers to the depot. Besides, in the case of our problem the tour is not segmented to line haul and backhaul clusters and splitting of the demand as well as pick-up quantities is not permitted to occur. Yet, there exists some number of similar instances to our problem in the literature. We have studied and described these problems, approaches, and solution techniques in the literature as well as the general VRP literature.

We have tested our dual GA heuristic, which was improved by ourselves. The heuristics introduced for solving the VRP start at one point of the search space and proceed down one path, accepting an improvement if it exists. That is, no matter how many times one runs the heuristics she would always end up with the same final solution if she started from the same initial solution. Stochastic heuristics or heuristics that tend to change the search path based on upon probabilistic factors, on the other hand yield to give different solutions during different executions, even when starting from the same initial solution (Thangiah and Petrovic, 1998). The heuristics developed in this thesis incurred high level of stochastic content. Thus, it is quite probable that even the runs are replicated with our codes on identical machines, one may not come up with the solutions same as the ones we have supplied at this work.

It was the first time at this paper the random keys method was tested on a VRP instance. Together with the second heuristic this work is the first ever study that uses GAs to solve VRPSPD. We have come out with improvements to the previously declarations as a result of our study. However, computation times, although not expressed explicitly, are not quite short to compete with other heuristics whose

progresses are declared to be in seconds. Future directions to our work may be to eliminate the remaining parameters while improving the best solutions of this study as well as to enhance the speed of the process leading to a satisficing outcome. This would be probable by imbedding the improvements in the exact algorithms, better memory properties, and enhancing the search capabilities of the genetic mechanism with those of the well performing ones such as the tabu search and the simulated annealing heuristics.

REFERENCES

1. Bean, J.C., "Genetic Algorithms and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing*, vol.6, no.2, Spring 1994.

2. Bertsekas, D. P., *Network Optimization: Continuous and Discrete Models,* Athena Scientific, Belmont, Massachusetts, 1998.

3. Clarke, G. and Wright W., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, 12, pp.568-581, 1964

4. Crainic, T.G. and Laporte, G., "Planning Models for Freight Transportation," *European Journal of Operational Research*, vol.97, pp.409-438, 1997.

5. Croes, G.A., "A method for solving the traveling salesman problems," *Operations Research*, vol.6, pp.791-812, 1958.

6. Dantzig, G.B. and Ramser, J.H., "The truck dispatching problem," *Management Science*, vol.6, no.1, pp.58-64, 1959.

7. Derigs, U., Kabath, M., and Zils, M., "Adaptive genetic algorithms: amethodology for dynamic autoconfiguration of genetic search algorithms," *Metaheuristics: Advances and Trends in Local Search for Optimization,* Kluwer Academic Publishers, pp.231-249, Boston, 1999.

8. Dethloff, J., "Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-up," *OR Spektrum*, vol.23, pp.79-96, 2001.

9. Gen, M. and Cheng, R., *Genetic Algorithms and Engineering Optimization*, John Wiley and Sons, Inc., New York, USA, 2000.

10. Gendrow, M., Laporte, G., and Vigo, D., "Heuristics for the Traveling Salesman Problem with Pick-Up and Delivery Delay," *Computers and Operations Research*, vol.26, pp.699-714, 1999.

11. Gillett, B. and Miller L., "A Heuristic for the Vehicle Dispatching Problem," *Operations Research*, vol.22, pp.340-349, 1974.

12. Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I. M., "Meta Heuristics in Vehicle Routing," *Fleet Management and Logistics*, Kluwer Academic Publishers, pp.33-57, Boston, 1998.

13. Mosheiov, G., "Vehicle Routing With Pick Up And Delivery: Tour Partitioning Heuristics", *Computers & Industrial Engineering*, vol.34, no.3, pp.669-684, 1998.

14. Halse, K.*, Modeling and Solving Complex Vehicle Routing Problems*, PhD Thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Lyngby, 1992.

15. Hertz, A. and Widmer, M., "Preface: Guidelines for the Use of Meta Heuristics in Combinatorial Optimization," *European Journal of Operational Research,* Article in press, 2003.

16. Min, H., "The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points," *Transportation Research*, vol.23-A, pp. 377-386, 1989.

17. Holland, J. H., *Adoption in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

18. Lacomme P., Prins C., Ramdane-Cherif. W., "A Genetic Algorithm for the Capacitated Arc Routing Problem and its Extensions," *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, Springer, v. 2037, pp.473-483, Berlin, 2001.

19. Osman, I. H., "Preface: Focused Issue on Applied Meta Heuristics", *Computers and Industrial Engineering,* vol.44, pp. 205-207, 2002.

20. Osman, I. H., "An Introduction to Meta Heuristics," *Operational Research Tutorial Papers*, Operational Research Society Press, pp. 92-122, Birmingham, U.K., 1995.

21. Osman, I. H., and Kelly, J. P., "Meta Heuristics: An overview," *Meta Heuristics: Theory and Applications*, Kluwer, Boston, pp. 1-21, 1996.

22. Potvin, J., and Rousseau J.M., "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows," *European Journal of Operational Research*, vol.66, pp.331, 1993

23. Reeves, C., and Rowe, J. E., *Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory,* Kluwer Academic Publishers, Massachusetts, USA, 2003.

24. Reeves, R.C., *Modern Heuristic Techniques for Combinatorial Problems*, Mc Graw Hill, New York, 1995.

25. Salhi, S. and Nagy, G., "A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems With Backhauling," *Journal of the Operational Research Society*, v.50, pp. 1034-1042, 1999.

26. Tarantilis, C. D., Kirinovdis, C. T., and Markatos, N. C., "Use of the BATA Algorithm and MIS to Solve the Mail Carrier Problem," *Applied Mathematical Modeling*, v.26, pp. 481-500, 2002.

27. Thangiah, S.R. and Petrovic, P., "Introduction to Genetic Heuristics and Vehicle Routing Problems with Complex Constraints", *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search: Interfaces in Computer Science and Operations Research*, Kluwer Academic Publishers, Massachusetts, 1998.

28. Topcuoglu, H. and Sevilmis C., "Task Scheduling with Conflicting Objectives," *Lecture Notes on Computer Science*, Springer-Verlag, vol.2457, pp.346-355, Heidelberg, 2002.

29. Toth, P. and Vigo, D., "Models, Relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics,* v.23, pp. 487-512, 2002.

30. Toth, P. and Vigo, D., "Exact Solution of the Vehicle Routing Problem," *Fleet Management and Logistics*, Kluwer Academic Publishers, Massachusetts, USA, pp.3-4, 1998.

31. Ulusoy, G., "The Fleet Size and Mixed Problem for Capacitated Arc Routing," *European Journal of Operational Research,* v.22, pp. 329-337, 1985.

32. Waters, C. D. J., "A Solution Procedure for the Vehicle Scheduling Problem Based on Iterative Route Improvement," *Journal of Operational Research* Society, v.38, pp. 833-839, 1987.

APPENDICES

Appendix A: Pseudo-Code for Random Keys Method

Initialize the input and data keeping structures

Read and input the distance matrix between all the nodes including the depot.

Read and input the demand and supply amounts at each node excluding the depot.

Sum up the demand and supply values at each node and assess the minimum amount of vehicles required.

While NOT end of 50 nodes
total demand += demand of node i;
total supply += supply of node i;
i← i+1;
If (total demand > total supply)
Minimum number of vehicles = ⌈total demand / vehicle capacity⌉
Else
Minimum number of vehicles = ⌈total supply / vehicle capacity⌉

Establish the initial truck capacity available for distribution of goods and the initial empty capacity.

*fullcapacity = Vehicle capacity * fillrate;*
*emptycapacity = Vehicle capacity * (1-fillrate);*

Initialize the chromosome structure.

- 200 chromosomes are constructed as the initial population and this number is kept constant throughout the generations.
- Each gene of the chromosome possesses two information values. The first one is the number of the node, the second is the random number assigned to this gene.

While NOT End of the Generations

Generate twins of currently generated chromosomes.

Sort the genes in the clone chromosome in ascending order of the random numbers at each gene.

Distribute the nodes to vehicles and calculate the total road covered.

for (total number of total chromosomes, i)
for (total number of genes in the clone chromosome, j)
if (the fullcapacity ≥ demand (i,j)) AND (emptycapacity + demand (i,j) ≥ supply (i,j))
fullcapacity ← fullcapacity – demand(i, j);
emptycapacity ← emptycapacity + demand (i,j) - supply (i,j);
roadcovered ← roadcovered + distance(i^1,$(i-1)^1$);
else
roadcovered ← roadcovered + distance(($i-1)^1$, 0);
*fullcapacity = Vehicle capacity * fillrate;*
*emptycapacity = Vehicle capacity * (1-fillrate);*
fullcapacity ← fullcapacity – demand(i, j);
emptycapacity ← emptycapacity + demand (i,j) - supply (i,j);
roadcovered ← distance(0, i^1);

if (it is the last gene j)
roadcovered ← roadcovered + distance(i^1 , 0);

Record the performance of the clone chromosome.

Compare the found value with the current best practice.

if (current best solution > roadcovered(i))
current best solution = roadcovered(i);

keep the original chromosome sequence i;

Sort the road covered values of each of 200 chromosomes in ascending order.

Assign them weight values with respect to performances obtained.

Generate a random matching between pair of the 200 chromosomes taking into consideration the weight values based on the performance vales assigned.

Realize cross-over between two parents for every pair.

take the original cromosomes
crosspoint =random number less than maximum number of genes;
for (j = crosspoint to maximum number of nodes)
temp = parent1 j^4;
parent1 j^4 = parent2 j^4;
parent2 j^4 = temp;

Realize mutations and alter the random number part of a random gene at an original random chromosome.

check = current generation - the last generation a best solution found;
increase the chance of mutation as check increases;
if (mutation condition satisfied)
choose a random original chromosome, i;
choose a random gene, j;
$(i, j)^4$ = random number;

This is the end of the generations.

Get the best chromosome sequence ever found in the replications, redistribute the nodes to vehicles and output the obtained sequence on the user screen.

Appendix B: Pseudo-Code For the Second Method

Initialize the input and data keeping structures

Read and input the distance matrix between all the nodes including the depot.

Read and input the demand and supply amounts at each node excluding the depot.

Sum up the demand and supply values at each node and assess the minimum amount of vehicles required.

While NOT end of 50 nodes
  total demand += demand of node i;
  total supply += supply of node i;
  i← i+1;
If (total demand > total supply)
  Minimum number of vehicles = ⌈total demand / vehicle capacity⌉
Else
  Minimum number of vehicles = ⌈total supply / vehicle capacity⌉

Establish the initial truck capacity available for distribution of goods and the initial empty capacity.

*fullcapacity = Vehicle capacity * fillrate;*
*emptycapacity = Vehicle capacity * (1-fillrate);*

Initialize the chromosome structure.

- 200 chromosomes are constructed as the initial population and this number is kept constant throughout the generations.
- Each gene of the chromosome possesses two information values. The first one is the number of the node, the second is the random number assigned to this gene.

Sort the genes in the chromosomes in ascending order of the random numbers of each gene.

While NOT End of the Generations

Distribute the nodes to vehicles and calculate the total road covered.

for (total number of total chromosomes, i)
for (total number of genes in the clone chromosome, j)
if (the fullcapacity ≥ demand (i,j)) AND (emptycapacity + demand (i,j) ≥ supply (i,j))
fullcapacity ← fullcapacity – demand(i, j);
emptycapacity ← emptycapacity + demand (i,j) - supply (i,j);
roadcovered ← roadcovered + distance(i^1,$(i-1)^1$);
else
roadcovered ← roadcovered + distance(($i-1)^1$, 0);
Send the sub-route to optimizer
roadcovered ← reduced roadcovered;
total_route_covered ← roadcovered + total_route_covered;
roadcovered ← 0;
*fullcapacity = Vehicle capacity * fillrate;*
*emptycapacity = Vehicle capacity * (1-fillrate);*
fullcapacity ← fullcapacity – demand(i, j);
emptycapacity ← emptycapacity + demand (i,j) - supply (i,j);
roadcovered ← distance(0, i^1);

if (it is the last gene j)
roadcovered ← roadcovered + distance(i^1 , 0);
Send the sub-route to optimizer
roadcovered ← reduced roadcovered;
total_route_covered ← roadcovered + total_route_covered;

The *optimizer* reduces the sub-tour distance by applying Or-opt.

Given a sub-tour with *n* nodes

h← 1, *m*← 2, *ns*← *0, maximum_ns*← *0* ;

While Not All *n* Nodes Iterated

While Not All Nodes Following *h* Considered

Calculate the net saving *ns* if the node *h* is replaced between node *m* and *m*+1;

If (ns ≥ maximum_ns) And (ns ≥0)

maximum_ns ← ns;

best_location_for_insertion ← m;

m ← m+1;

END of While, m = n;

Replace node h between m and m+1;

h← h+1;

END of While, h=n;

Calculate the total sub-tour length;

Give out the sub-tour length

Record the performance of the clone chromosome.

Compare the found value with the current best practice.

if (current best solution > total_route_covered (i))

current best solution = total_route_covered (i);

keep the original chromosome sequence i;

Sort the *total_route_covered* values of each of 200 chromosomes in ascending order.

Assign them probability weight values, *p(i)* with respect to performances obtained.

p(i) ← Average of (total_route_covered) / total_route_covered(i)

Generate a random matching between two of the 200 chromosomes for 100 times taking into consideration the probability weight values.

Realize cross-over between two parents for 100 times.

take the parent cromosomes

crosspoint =random number less than maximum number of genes;

for (i ≤ crosspoint)

locate the gene of the first parent directlyinto the first offspring;

for (j = crosspoint to maximum number of nodes)

search the genes in the second parent but not yet located in the offspring;

locate them in the offspring with the same sequence appearing in the second parent

Realize mutations and alter the random number part of a random gene at an original random chromosome.

check = current generation - the last generation a best solution found;

increase the chance of normal mutation and destructive mutation linearly as check increases in multiples of IncremetSize;

While (mutation condition satisfied)

choose a random original chromosome, c;

choose two random genes, i and j;

interchange their locations on the chromosome;

choose two random gene, s;

change its random key with an arbitrary number;

END While;

While (destructive mutation condition satisfied)

choose a random original chromosome, c;

resort the genes in the chromosome with ascending order of the random keys;

END While;

This is the end of the generations.

Get the best chromosome sequence ever found in the replications, redistribute the nodes to vehicles and output the obtained sequence on the user screen.

REFERENCES

1. Bean, J.C., "Genetic Algorithms and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing*, vol.6, no.2, Spring 1994.

2. Bertsekas, D. P., *Network Optimization: Continuous and Discrete Models,* Athena Scientific, Belmont, Massachusetts, 1998.

3. Clarke, G. and Wright W., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, 12, pp.568-581, 1964

4. Crainic, T.G. and Laporte, G., "Planning Models for Freight Transportation," *European Journal of Operational Research*, vol.97, pp.409-438, 1997.

5. Croes, G.A., "A method for solving the traveling salesman problems," *Operations Research*, vol.6, pp.791-812, 1958.

6. Dantzig, G.B. and Ramser, J.H., "The truck dispatching problem," *Management Science*, vol.6, no.1, pp.58-64, 1959.

7. Derigs, U., Kabath, M., and Zils, M., "Adaptive genetic algorithms: amethodology for dynamic autoconfiguration of genetic search algorithms," *Metaheuristics: Advances and Trends in Local Search for Optimization,* Kluwer Academic Publishers, pp.231-249, Boston, 1999.

8. Dethloff, J., "Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-up," *OR Spektrum*, vol.23, pp.79-96, 2001.

9. Gen, M. and Cheng, R., *Genetic Algorithms and Engineering Optimization*, John Wiley and Sons, Inc., New York, USA, 2000.

10. Gendrow, M., Laporte, G., and Vigo, D., "Heuristics for the Traveling Salesman Problem with Pick-Up and Delivery Delay," *Computers and Operations Research*, vol.26, pp.699-714, 1999.

11. Gillett, B. and Miller L., "A Heuristic for the Vehicle Dispatching Problem," *Operations Research*, vol.22, pp.340-349, 1974.

12. Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I. M., "Meta Heuristics in Vehicle Routing," *Fleet Management and Logistics*, Kluwer Academic Publishers, pp.33-57, Boston, 1998.

13. Mosheiov, G., "Vehicle Routing With Pick Up And Delivery: Tour Partitioning Heuristics", *Computers & Industrial Engineering*, vol.34, no.3, pp.669-684, 1998.

14. Halse, K.*, Modeling and Solving Complex Vehicle Routing Problems*, PhD Thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Lyngby, 1992.

15. Hertz, A. and Widmer, M., "Preface: Guidelines for the Use of Meta Heuristics in Combinatorial Optimization," *European Journal of Operational Research,* Article in press, 2003.

16. Min, H., "The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Points," *Transportation Research*, vol.23-A, pp. 377-386, 1989.

17. Holland, J. H., *Adoption in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

18. Lacomme P., Prins C., Ramdane-Cherif. W., "A Genetic Algorithm for the Capacitated Arc Routing Problem and its Extensions," *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, Springer, v. 2037, pp.473-483, Berlin, 2001.

19. Osman, I. H., "Preface: Focused Issue on Applied Meta Heuristics", *Computers and Industrial Engineering,* vol.44, pp. 205-207, 2002.

20. Osman, I. H., "An Introduction to Meta Heuristics," *Operational Research Tutorial Papers*, Operational Research Society Press, pp. 92-122, Birmingham, U.K., 1995.

21. Osman, I. H., and Kelly, J. P., "Meta Heuristics: An overview," *Meta Heuristics: Theory and Applications*, Kluwer, Boston, pp. 1-21, 1996.

22. Potvin, J., and Rousseau J.M., "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows," *European Journal of Operational Research*, vol.66, pp.331, 1993

23. Reeves, C., and Rowe, J. E., *Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory,* Kluwer Academic Publishers, Massachusetts, USA, 2003.

24. Reeves, R.C., *Modern Heuristic Techniques for Combinatorial Problems*, Mc Graw Hill, New York, 1995.

25. Salhi, S. and Nagy, G., "A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems With Backhauling," *Journal of the Operational Research Society*, v.50, pp. 1034-1042, 1999.

26. Tarantilis, C. D., Kirinovdis, C. T., and Markatos, N. C., "Use of the BATA Algorithm and MIS to Solve the Mail Carrier Problem," *Applied Mathematical Modeling*, v.26, pp. 481-500, 2002.

27. Thangiah, S.R. and Petrovic, P., "Introduction to Genetic Heuristics and Vehicle Routing Problems with Complex Constraints", *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search: Interfaces in Computer Science and Operations Research*, Kluwer Academic Publishers, Massachusetts, 1998.

28. Topcuoglu, H. and Sevilmis C., "Task Scheduling with Conflicting Objectives," *Lecture Notes on Computer Science*, Springer-Verlag, vol.2457, pp.346-355, Heidelberg, 2002.

29. Toth, P. and Vigo, D., "Models, Relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics,* v.23, pp. 487-512, 2002.

30. Toth, P. and Vigo, D., "Exact Solution of the Vehicle Routing Problem," *Fleet Management and Logistics*, Kluwer Academic Publishers, Massachusetts, USA, pp.3-4, 1998.

31. Ulusoy, G., "The Fleet Size and Mixed Problem for Capacitated Arc Routing," *European Journal of Operational Research,* v.22, pp. 329-337, 1985.

32. Waters, C. D. J., "A Solution Procedure for the Vehicle Scheduling Problem Based on Iterative Route Improvement," *Journal of Operational Research* Society, v.38, pp. 833-839, 1987.