DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

AN ADAPTIVE LOCAL SEARCH ALGORITHM FOR VEHICLE ROUTING PROBLEMS WITH SIMULTANEOUS AND MIXED PICKUPS AND DELIVERIES

by
Mustafa AVCI

January, 2014
İZMİR

AN ADAPTIVE LOCAL SEARCH ALGORITHM FOR VEHICLE ROUTING PROBLEMS WITH SIMULTANEOUS AND MIXED PICKUPS AND DELIVERIES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Industrial Engineering, Industrial Engineering Program

by
Mustafa AVCI

January, 2014
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"AN ADAPTIVE LOCAL SEARCH ALGORITHM FOR VEHICLE ROUTING PROBLEMS WITH SIMULTANEOUS AND MIXED PICKUPS AND DELIVERIES"** completed by **MUSTAFA AVCI** under supervision of **ASSOC. PROF. DR. ŞEYDA AYŞE TOPALOĞLU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Şeyda Ayşe TOPALOĞLU

Supervisor

Assoc. Prof. Dr. Ali Serdar TAŞAN

(Jury Member)

Asst. Prof. Dr. Ahmet ÖZKURT

(Jury Member)

Prof. Dr. Ayşe OKUR
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to Assoc. Prof. Dr. Şeyda Topaloğlu, for her guidance, support and understanding. She always supported my decisions and gave me freedom in my research while helping me to overcome challenges and giving advices.

I would also like to express my thanks to all professors and colleagues in the Industrial Engineering Department of Dokuz Eylül University for their support, encouragement and tolerance. Especially, I am grateful to my colleague Res. Assist. Alper Hamzadayı for his invaluable support.

Mustafa Avcı

AN ADAPTIVE LOCAL SEARCH ALGORITHM FOR VEHICLE ROUTING PROBLEMS WITH SIMULTANEOUS AND MIXED PICKUPS AND DELIVERIES

ABSTRACT

As the importance of reverse logistics activities increases, integration of pickup and delivery operations of goods becomes more crucial. The vehicle routing problem with simultaneous pickup and delivery (VRPSPD) and mixed pickup and delivery (VRPMPD) have attracted research interest due to their applicability in numerous reverse logistic systems involving bi-directional flow of goods. The VRPSPD is an extension to the classical Vehicle Routing Problem (VRP), where customers may both receive and send goods simultaneously. The common assumption that goods may only be picked up after all deliveries have been completed is not made in these problems. The VRPMPD differs from the VRPSPD in that the customers have either pickup or delivery demand. Therefore, the solution approaches proposed for the VRPSPD can be directly applied to the VRPMPD. From the theoretical point of view, the VRPSPD is an NP-hard combinatorial optimization problem because it generalizes the standard version of the VRP. Large-scale VRPSPD instances encountered in real-life business activities cannot be efficiently tackled by exact solution approaches. In this study, an adaptive local search solution approach is developed for the VRPSPD and the VRPMPD. In this local search methodology, we hybridize a Simulated Annealing (SA) inspired algorithm with Variable Neighborhood Descent (VND). The algorithm uses an adaptive threshold function which makes the algorithm self-tuning. The proposed approach is tested on well-known VRPSPD and VRPMPD benchmark instances derived from the literature. The computational results indicate that the proposed algorithm is effective in solving the problems in reasonable computation time.

Keywords: Vehicle routing problem with simultaneous pickup and delivery, local search algorithms, variable neighborhood descent algorithm, reverse logistics

EŞ ZAMANLI VE KARIŞIK DAĞITIM VE TOPLAMALI ARAÇ ROTALAMA PROBLEMLERİ İÇİN BİR ADAPTİF LOKAL ARAMA ALGORİTMASI

ÖZ

Tersine lojistik aktivitelerinin önemi arttıkça, toplama ve dağıtım operasyonlarının entegre edilmesi işlemi çok daha önemli hale gelmektedir. Eş zamanlı toplamalı dağıtımlı araç rotalama problemi (EZTDARP) ve karışık toplamalı dağıtımlı araç rotalama problemi (KTDARP) çift yönlü mal akışını içeren birçok tersine lojistik sisteminde uygulanabilirlikleri sebebiyle araştırmacıların ilgisini çekmektedir. EZTDARP problemi klasik araç rotalama probleminin farklı türlerinden biridir, bu problemde müşteriler eş zamanlı olarak mal alımını ve gönderimini yapabilmektedirler. KTDARP sadece mal gönderen veya mal alan müşteriler içermesi nedeniyle EZTDARP probleminden farklılaşmaktadır. Bu yüzden, EZTDARP için önerilen bir çözüm yöntemi doğrudan KTDARP için de uygulanabilmektedir. Yaygın olarak kullanılan yük toplama işlemlerinin tüm dağıtım işlemeleri tamamlandıktan sonra yapılacağı varsayımı bu problemlerde yapılmamaktadır. Teorik açıdan bakıldığında, EZTDARP iyi bilinen bir NP-hard problem olan standart araç rotalama problemini genelleştirmesi sebebiyle bir NP-hard kombinatoryal optimizasyon problemdir. Gerçek hayat problemlerinde karşılaşılan büyük ölçekli EZTDARP örnekleri kesin çözüm yöntemleriyle etkin bir şekilde çözülememektedir. Bu çalışmada, EZTDARP ve KTDARP problemleri için bir adaptif lokal arama algoritması geliştirilmiştir. Bu çözüm yönteminde, tavlama benzetimi türü bir algoritmayla değişen komşuluklar algoritması melezlenmiştir. Geliştirilen algoritma, kendi kendine ayarlanabilir bir adaptif eşik değer fonksiyonu kullanmaktadır. Önerilen yöntemin etkinliği literatürden türetilmiş iyi bilinen ölçüt problemleri üzerinde test edilmiştir. Sayısal sonuçlar önerilen algoritmanın makul hesaplama süreleri içinde problemleri çözmede etkin olduğunu ortaya çıkarmaktadır.

Anahtar sözcükler: Eş zamanlı toplamalı dağıtımlı araç rotalama problemi, lokal arama algoritmaları, değişen komşuluklar algoritması, tersine lojistik

CONTENTS

Page

M.Sc THESIS EXAMINATION RESULT FORM ..... **ii**
ACKNOWLEDGMENTS ..... **iii**
ABSTRACT ..... **iv**
ÖZ ..... **v**
LIST OF FIGURES ..... **viii**
LIST OF TABLES ..... **ix**

CHAPTER ONE - INTRODUCTION 1

CHAPTER TWO - VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY 4

2.1 The Classical Vehicle Routing Problem (VRP) 4
2.1.1 Vehicle Routing Problems with Pickups and Deliveries 5
2.1.1.1 Vehicle Routing Problems with Backhauls 6
2.1.1.2 Vehicle Routing Problems with Mixed Pickup and Delivery 6
2.2 The Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) 8
2.2.1 Mathematical Model 9
2.2.2 Literature Review 11

CHAPTER THREE - AN OVERVIEW ON METAHEURISTICS 15

3.1 Simulated Annealing (SA) 16
3.2 Genetic Algorithm (GA) 17
3.3 Tabu Search (TS) 18
3.4 Particle Swarm Optimization (PSO) 18
3.5 Ant Colony Optimization (ACO) 19

CHAPTER FOUR - SOLUTION METHODOLOGY 21

4.1 Self-Adaptive Local Search (SALS) 21
4.2 Variable Neighborhood Descent (VND) 24
4.3 Proposed Local Search Algorithm (SALS-VND) 24
4.3.1 Solution Representation 25
4.3.2 Generating Initial Solution 26
4.3.3 Checking Feasibility of the Capacity Constraint 27
4.3.4 Neighborhood Structures 28
4.3.4.1 Adjacent Swap 29
4.3.4.2 General Swap 29
4.3.4.3 Single Insertion 29
4.3.4.4 Block Insertion 29
4.3.4.5 2-opt 29
4.3.4.6 Shift(1,0) 30
4.3.4.7 Swap(1,1) 30
4.3.4.8 Shift(2,0) 30
4.3.4.9 Swap(2,1) 30
4.3.4.10 Swap(2,2) 30
4.3.5 Stage I: Integration of SALS with VND (hybrid_SALS) 33
4.3.6 Applied VND Algorithm (p_VND) 36
4.3.7 Stage II: Application of VND with a Perturbation Mechanism (VND_perturbation) 37
4.4 Numerical Study 42
4.4.1 Computational Results 44
4.4.2 An Illustrative Example 52

CHAPTER FIVE - CONCLUSION 56

REFERENCES 58

LIST OF FIGURES

Page

Figure 2.1 Illustration of the VRPB 7
Figure 2.2 Illustration of the VRPMPD 7
Figure 2.3 An example of non-monotonic changing of the vehicle load 9
Figure 3.1 A general scheme for metaheuristics 16
Figure 4.1 Changing of the value of t during iterations 22
Figure 4.2 Steps of SALS algorithm 23
Figure 4.3 Steps of VND algorithm 24
Figure 4.4 Basic steps of the proposed algorithm 25
Figure 4.5 Solution representation 26
Figure 4.6 Initial solution construction procedure 27
Figure 4.7 Feasibility checking procedure 28
Figure 4.8 Intra-route neighborhood structures 31
Figure 4.9 Inter-route neighborhood structures 32
Figure 4.10 Visualization of the first phase of the algorithm. Improved solution points, x_{lb}, found by SALS from different regions are used as initial solutions for VND. By applying VND, the solution points, x_{gb}, yielding less cost are obtained. .. 33
Figure 4.11 Flowchart of the first stage of the algorithm (hybrid_SALS) 34
Figure 4.12 First stage of the proposed algorithm (hybrid_SALS) 35
Figure 4.13 Main steps of the applied VND algorithm (p_VND) 38
Figure 4.14 Visualization of the second phase of the algorithm. Starting from an initial solution x, VND is implemented and a local minimum point x' is found. By applying the perturbation mechanism, different search regions can be explored. 39
Figure 4.15 Flowchart of the second stage of the algorithm (VND_perturbation) 39
Figure 4.16 Second phase of the proposed algorithm (VND_perturbation) 40
Figure 4.17 Proposed perturbation mechanism (perturbation) 41
Figure 4.18 Illustration of the vehicle routes obtained by hybrid_SALS 53
Figure 4.19 Illustration of the vehicle routes obtained by SALS-VND 53

LIST OF TABLES

Page

Table 4.1 Computational results of different neighborhood orderings 38
Table 4.2 Computational results of SALS-VND and hybrid_SALS 42
Table 4.3 Computational results for the VRPSPD benchmark instances of Dethloff (2001) 46
Table 4.4 Results of paired-t test for the VRPSPD benchmark instances of Dethloff (2001) 46
Table 4.5 Computational results for the VRSPD benchmark instances of Salhi & Nagy (1999) 47
Table 4.6 Results of paired-t test for the VRSPD benchmark instances of Salhi & Nagy (1999) 48
Table 4.7 Computational results for the VRPMPD instances labeled T of Dethloff (2001) 49
Table 4.8 Computational results for the VRPMPD instances labeled Q of Dethloff (2001) 50
Table 4.9 Computational results for the VRPMPD instances labeled H of Dethloff (2001) 51
Table 4.10 Results of paired-t test for the VRPMPD instances of Dethloff (2001) .. 51
Table 4.11 Coordinates, pickup and delivery demands from node 0 to node 25 54
Table 4.12 Coordinates , pickup and delivery demands from node 26 to node 50 55

CHAPTER ONE
INTRODUCTION

Over the last century, continuing increase on energy consumption and emissions of greenhouse gases which mainly stems from industrial activities have brought about serious environmental problems such as global warming and climate change. Consequently, global awareness on environmental issues has increased rapidly in recent years, making the reverse logistics activities more important. The activities in the context of reverse logistics cover recycling and reusing operations that require bi-directional flow of goods. As a result, the design of the transportation systems requiring both pickup and delivery services becomes crucial for companies to minimize transportation costs.

If the distribution and collection activities are operated separately, the transportation costs of the companies increase substantially. Besides, the operations aiming to reduce energy consumption may cause extra damage to the environment and yield no significant benefit. Therefore, all these conditions necessitate the integration of pickup and delivery operations. The Vehicle Routing Problems with Pickups and Deliveries (VRPPD) arise from these requirements and the purpose of these problems is to combine the bi-directional flow of goods effectively and efficiently. There exist three variants of the problem in the literature which are Vehicle Routing Problem with Backhauls (VRPB), Vehicle Routing Problem with Mixed Pickup and Delivery (VRPMPD) and Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD), respectively.

In the VRPB, vehicles can only service backhauls (customers having pickup demand) after visiting all customers having delivery demand, which are called as linehauls. Imposing this type of precedence constraint makes the reorganization of the pickup and delivery loads on the vehicles easier. Moreover, accepting all pickup goods after servicing the delivery demands simplifies the control of the vehicle load. The VRPMPD relaxes the precedence constraint of the VRPB, and customer demands can be satisfied in any order. Making such a relaxation on the problem

constraint can reduce the transportation cost but results in a fluctuating load on the vehicle.

The VRPB and the VRPMPD model the transportation systems in which each customer has either a pickup or a delivery demand. However, in numerous distribution and collection systems, the customers require both a pickup and a delivery demand. Additionally, they want to be serviced once by the same vehicle. The VRPSPD covers such transportation systems where customers have both types of demands and require to be serviced exactly once. Therefore, the VRPSPD can be defined as a generalized version of the VRPMPD and a solution approach constructed for the VRPSPD can be directly applied to the VRPMPD. The problem arises in many real life business activities containing bi-directional flow of goods. One of the most well-known real life example to the VRPSPD occurs in soft drink industry where the operations of delivering full bottles and collecting empty ones are performed simultaneously by the same vehicle. The problem can also be encountered at grocery stores where empty pallets or containers are reused for transportation. Another example to the problem occurs on the collection of used products at the end of their lifecycles for remanufacturing operations.

From the theoretical point of view, the VRPSPD is known to be an NP-hard because it is a variant of the classical Vehicle Routing Problem (VRP) which is a well-known NP-hard problem. Large-scale real life problem instances cannot be solved efficiently by exact solution methods. Thus, heuristic and metaheuristic solution approaches have been generally implemented as a solution method for the VRPSPD since these solution methods are capable of generating high quality solutions to this type of problems within a reasonable computation time.

In this thesis, we develop an adaptive local search algorithm for solving the VRPSPD and the VRPMPD. The applied methodology is constructed by hybridizing a Simulated Annealing (SA) inspired algorithm with Variable Neighborhood Descent (VND) algorithm. The SA-inspired algorithm enables the search process to explore different search regions in the search space while VND is implemented to generate

high-quality solutions from the examined regions. Moreover, an extra refinement procedure consisting of VND and a developed perturbation mechanism is applied to further improve the solution obtained from the first stage. One of the most important features of the developed algorithm is that it is free from parameter tuning which makes the implementation of the algorithm easier. The proposed methodology is tested on well-known VRPSPD and VRPMPD benchmark instances derived from the literature. The computational results show that the proposed algorithm generates competitive results with the most sophisticated algorithms developed so far, for the VRPSPD and the VRPMPD.

The remainder of this thesis is organized as follows: In Chapter 2, we firstly give an overview of the classical VRP and its variants, the VRPB and the VRPMPD, and describe the VRPSPD with a mathematical model proposed by Dethloff (2001). Finally, the studies related to the VRSPD in the literature are reviewed. In Chapter 3, we firstly give information about the general properties of the metaheuristic solution approaches, and then some well-known metaheuristic algorithms are briefly described. In Chapter 4, a detailed information about our proposed methodology is initially given. Then, the computational results of the proposed algorithm obtained from the benchmark instances are presented. Finally, concluding remarks and future research are presented in Chapter 5.

CHAPTER TWO
VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY

In this chapter, we firstly describe the vehicle routing problem with simultaneous pickup and delivery along with the mathematical model of the problem developed by Dethloff (2001) and then the related literature is given.

2.1 The Classical Vehicle Routing Problem (VRP)

The classical Vehicle Routing Problem (VRP) is one of the most well-known problems in combinatorial optimization (Barnhart & Laporte, 2007). Since the VRP has been first introduced by Dantzig & Ramser (1959), it has attracted a great deal of research interest due to its applicability on numerous logistics and transportation systems. Inspired by real-life cases, a lot of variants of the problem are proposed. These variants involve additional constraints such as route duration, time windows, heterogeneous fleet, multiple depots, pickup and delivery services etc.

The VRP can be formally defined as follows: Let G = (V, A) be a graph, where $V=\{v_0, v_1,\ldots,v_n\}$ is the set of vertices in which v_0 represents the central depot at which homogeneous vehicles are located and the other vertices represent the clients. $A = \{ (v_i, v_j): v_i, v_j \in V, i \neq j\}$ is the arc set and each arc $\{i, j\} \in A$ has a nonnegative cost c_{ij}. Each client $\{n_1,n_2,\ldots,n_n\}$ has a non-negative demand quantity d_i. The objective of the VRP is to determine a set of routes minimizing the cost and satisfying the following constraints: (i) every route starts and finishes at the central depot; (ii) each client must be visited exactly once by exactly one vehicle; (iii) the total amount of goods to be carried by a vehicle cannot exceed the vehicle capacity.

The VRP is known to be NP-hard because it generalizes the Travelling Salesman Problem (TSP) which is a well-known NP-hard problem. In the TSP, a salesman, starting from an initial location, services a set of clients by following a route in which the clients are visited only once and returns back to the initial location. The

objective is to determine the visiting order of the customers which minimizes the total traveled distance. The classical VRP and its variants are much more complicated versions of the TSP because it involves extra challenging constraints such as vehicle capacity, time windows, backhauls etc.

Numerous solution methods have been developed for the VRP. These methods can be divided into two categories as exact and heuristic solution methods. A detailed survey on the VRP variants, mathematical formulations and exact solution methods can be found in Toth & Vigo (2002). Another survey on exact solution approaches for the VRP is recently presented by Baldacci, Mingozzi & Roberti (2012).

Comprehensive surveys on heuristic and metaheuristic solution methods for the VRP can be found in Cordeau, Gendreau, Hertz, Laporte & Sormany (2005) and Vidal, Crainic, Gendreau & Prince (2013). Some successful applications of metaheuristic algorithms recently proposed for the VRP can be seen in Prins (2004), Reimann, Doerner & Hartl (2004), Tarantilis (2005), Li, Golden & Wasil (2005), Pisinger & Ropke (2007), Kytöjoki, Nuortio, Bräysy & Gendreau (2007) and Mester and Bräysy (2007).

2.1.1 Vehicle Routing Problems with Pickups and Deliveries

The Vehicle Routing Problem with Pickups and Deliveries (VRPPD) generalizes the classical VRP where pickup and delivery services are performed by the same vehicles. The problem can be classified into three categories according to the type of customers and the backhaul constraints. The extensions of the problem are: the VRP with backhauls (VRPB), the VRP with mixed pickup and delivery (VRPMPD), the VRP with simultaneous pickup and delivery (VRPSPD). Comprehensive reviews about the pickup and delivery problems can be found in Berbeglia, Cordeau, Gribkovskaia & Laporte (2007) and Parragh, Doerner & Hartl (2008).

2.1.1.1 Vehicle Routing Problems with Backhauls

The Vehicle Routing Problem with Backhauls (VRPB) is a variant of the VRPPD where the customers requiring delivery and pickup demands are classified as linehaul and backhaul customers, respectively. The VRPB is also known as delivery first pickup second VRP (Zachariadis, Tarantilis & Kiranoudis, 2009). In this problem, each vehicle must collect the pickup loads from backhaul customers after visiting the linehaul customers to deliver their demands (see Figure 2.1). Total delivery amount of the linehaul customers and the total pickup amount of the backhaul customers served by a vehicle cannot exceed the vehicle capacity.

Many exact and heuristic solution approaches are developed for the VRPB. Exact solution methods proposed for the problem can be found in Toth & Vigo (1997) and Mingozzi, Giorgi & Baldacci (1999). Some well-known heuristic and metaheuristic solution approaches can be found in Goetschalckx & Jacobs-Blecha (1989), Casco, Golden & Wasil (1988), Anily (1996), Toth & Vigo (1999), Dethloff (2002) and Osman & Wassan (2002). Most recent metaheuristic algorithms for the VRPB are proposed by Brandao (2006), Ropke & Pisinger (2006), Wassan (2007), Gajpal & Abad (2009b) and Zachariadis & Kiranoudis (2012).

2.1.1.2 Vehicle Routing Problems with Mixed Pickup and Delivery

The Vehicle Routing Problem with Mixed Pickup and Delivery (VRPMPD) is another variant of the VRPPD where the precedence constraint of linehaul and backhaul customers is not imposed. Thus, the linehaul and the backhaul customers can be ordered in any sequence on a route (see Figure 2.2).

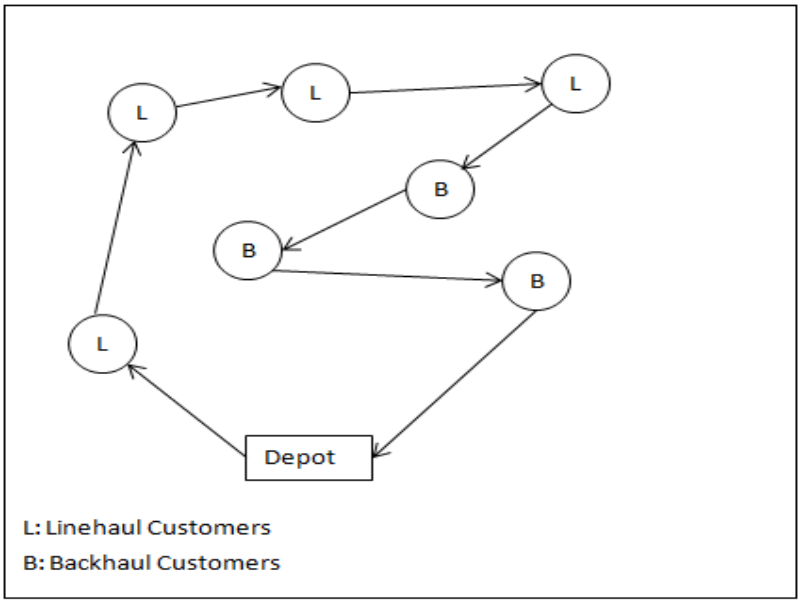


Figure 2.1 Illustration of the VRPB

Regarding the literature of the VRPMPD, there are only a few papers directly related to the problem. Mosheiov (1998) has developed a tour-partitioning heuristic which is based on splitting a travelling salesman tour into feasible vehicle routes. Salhi & Nagy (1999) suggested a cluster insertion heuristic which is an extended version of the classical insertion method. Wassan, Nagy & Ahmadi (2008) proposed a reactive tabu search algorithm for the VRPMPD.

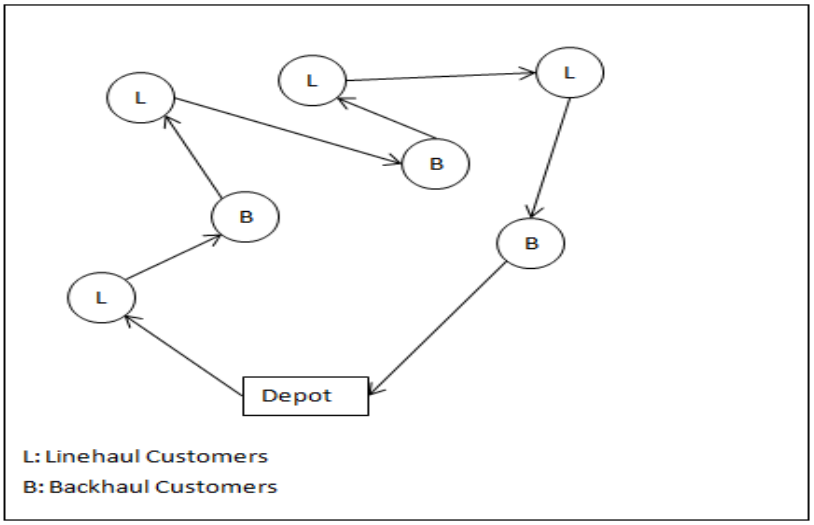


Figure 2.2 Illustration of the VRPMPD

2.2 The Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD)

The VRPSPD can be encountered in a distribution/collection system involving a set of customers requiring delivery and pickup services simultaneously. The customers are serviced by a homogeneous fleet of vehicles which are located at a central depot. Each customer has both pickup and delivery demands and the objective is to generate a feasible set of routes minimizing the total distance traveled.

The problem can be described with graph theory terms as follows: Let G = (N, A) be a complete graph, where $N=\{n_0,n_1,\ldots,n_n\}$ is the set of nodes in which n_0 represents the central depot and the remaining ones the customers. $A = \{ (n_i, n_j): n_i, n_j \in N, i \neq j\}$ is the set of arcs and each arc $\{i, j\} \in A$ has a nonnegative cost c_{ij}. Each customer $\{n_1,n_2,\ldots,n_n\}$ requires a nonnegative delivery quantity d_i and a nonnegative pickup quantity p_i. At the central depot, there are an unlimited number of homogeneous vehicles with capacity Q. The objective of the problem is to construct a set of vehicle routes satisfying the following constraints: (i) every route initiates and finishes at the central depot; (ii) each customer must be serviced once by one vehicle; (iii) all delivered goods must be originated from the depot and all pickup goods must be transported back to the depot; (iv) the transported amount of goods cannot exceed the capacity of a vehicle.

From the theoretical point of view, the VRPSPD is an NP-hard combinatorial optimization problem since it generalizes the standard version of the VRP. Large-scale VRPSPD instances encountered in real-life business activities cannot be efficiently tackled by exact solution approaches. Therefore, heuristic and metaheuristic methods have been proposed to solve the problem. The important aspect of the problem is that as the vehicle services the customers, the load of the vehicle changes non-monotonically due to the fact that the customers require both delivery and pickup services at the same time (see Figure 2.3). Hence, the solution methods developed for the problem must involve a mechanism that controls the load of the vehicle to ensure the capacity constraint.

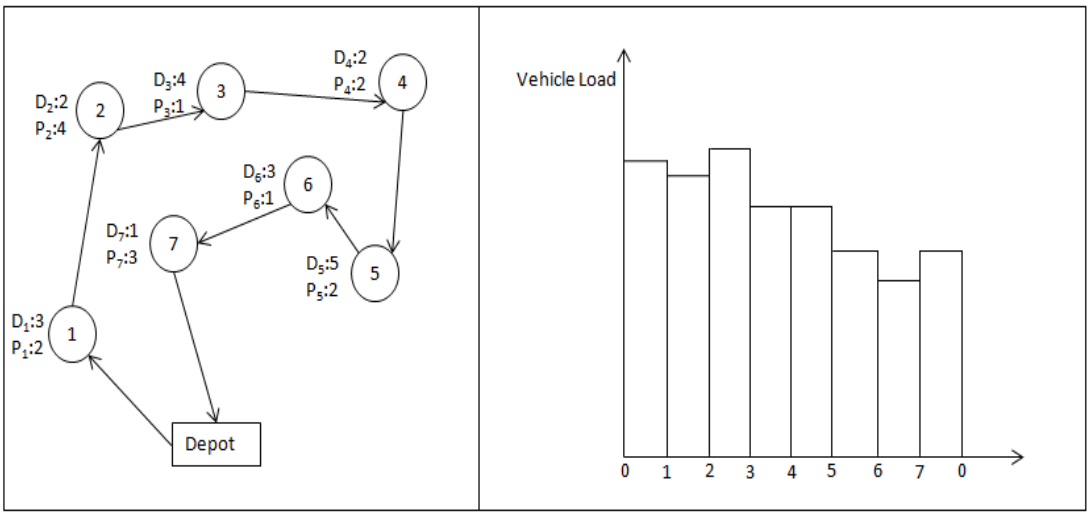


Figure 2.3 An example of non-monotonic changing of the vehicle load

The VRPSPD can arise in many practical applications of reverse logistics, and constructing an effective solution strategy to the problem is one of the most critical issues for designing the transportation system effectively. In the bottled drinks industry, for example, full bottles are delivered and empty ones which are used for recycling are collected simultaneously from the customers. Another practical application of the VRPSPD occurs on the collection of used materials such as car parts, industrial equipments and computers for remanufacturing or dissembling operations (Zachariadis, Tarantilis & Kiranoudis, 2009). Moreover, the problem can be seen at grocery stores in which pallets or boxes can be collected and reused for transportation (Dethloff, 2001).

2.2.1 Mathematical Model

In this section, we present the mathematical model of the VRPSPD proposed by Dethloff (2001).

Notation:

Sets:

J: Set of customer nodes

J_0: Set of all nodes containing the central depot,

V: Set of all vehicles

Parameters:

C: Vehicle Capacity

c_{ij}: Distance between nodes i and j $i, j \in J_0, i \neq j$

d_j: Delivery amount of customer j, $j \in J$

n: Number of nodes

p_j: Pickup amount of customer j, $j \in J$

M: Large number, e.g. $M=\max\{\sum_{j\epsilon J}(d_j+p_j), \sum_{i\epsilon J_0}\sum_{j\epsilon J_0, i\neq j} c_{ij}\}$

Decision Variables:

l'_v: Load of vehicle v when leaving the depot, $v \in V$

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

π_j: Variable used for sub-tour elimination and it can be interpreted as the position of node j, $j \in J$

x_{ijv}: 1, if vehicle $v \in V$ travels directly from i to j, and 0, otherwise, $v \in V$, $i \in J_0$, $j \in J_0$.

Model:

$$\text{Minimize } z = \sum_{i\in J_0}\sum_{j\in j_0}\sum_{v\in V} c_{ij}x_{ijv} \quad (2.1)$$

subject to

$$\sum_{i\in J_0}\sum_{v\in V} x_{ijv} = 1 \qquad (j \in J) \quad (2.2)$$

$$\sum_{i\in J_0} x_{isv} = \sum_{j\in J_0} x_{sjv} \qquad (s \in J, v \in V) \quad (2.3)$$

$$l'_v = \sum_{i\in J_0}\sum_{j\in J} d_j x_{ijv} \qquad (v \in V) \quad (2.4)$$

$$l_j \geq l'_v - d_j + p_j - M(1 - x_{0jv}) \quad (j \in J, v \in V) \quad (2.5)$$

$$l_j \geq l_i - d_j + p_j - M(1 - \sum_{v\in V} x_{ijv}) \qquad (i \in J, j \in J, j \neq i) \quad (2.6)$$

$$l'_v \leq C \qquad (v \in V) \quad (2.7)$$

$$l_j \leq C \qquad (j \in J) \quad (2.8)$$

$$\pi_j \geq \pi_i + 1 - n(1 - \sum_{v\in V} x_{ijv}) \qquad (i \in J, j \in J, j \neq i) \quad (2.9)$$

$$\pi_j \geq 0 \qquad (j \in J) \quad (2.10)$$

$$x_{ijv} \in \{0,1\} \qquad (i \in J_0, j \in J_0, v \in V) \quad (2.11)$$

The objective function of the model is given in (2.1) which minimizes the total distance traveled. Constraint (2.2) guarantees that each customer is visited exactly once. Constraint (2.3) ensures that each customer is visited and left by the same vehicle. The initial load of a vehicle is calculated in (2.4), which is the accumulated demand of the customers assigned to a vehicle on its route. Constraint (2.5) determines a vehicle loads after visiting the first customers. For other customers, the load of the vehicle is calculated as in (2.6). Constraints (2.7) and (2.8) satisfy the capacity restrictions of the vehicles. Constraint (2.9) imposes the sub-tour elimination constraints. Constraint (2.10) guarantees the non-negativity restrictions on π_j variables and constraint (2.11) indicates that x_{ijv} is a binary decision variable.

2.2.2 Literature Review

Over the last decade, Vehicle Routing Problem with Simultaneous Pickup and Delivery has become an interesting topic for researchers because of its applicability on reverse logistic systems containing distribution and collection of goods. In this section, we review the exact and heuristic solution approaches proposed for the VRPSPD.

The VRPSPD is first introduced by Min (1989). In this study, a book distribution and collection problem from a central library to 22 remote libraries is handled. In order to solve the problem, customers are clustered first and then the Traveling Salesman Problem is solved for each cluster. Dethloff (2001) developed an insertion based heuristic algorithm consisting of four different insertion criteria: traveling distance, residual capacity, radial surcharge and combination. Furthermore, the author presented a mathematical model for the problem and discussed the relationship between the VRPSPD and other VRP variants. Nagy & Salhi (2005) proposed an integrated heuristic approach for the VRPSPD. The algorithm comprises of various routines used for feasibility and improvement. The first exact algorithm for the VRPSPD has been developed by Dell'Amico, Righini & Salani (2006). In this study, two different strategies, exact dynamic programming and state space relaxation, are used for sub pricing problem. The algorithm can solve instances up to

40 customers optimally. Gajpal & Abad (2010) presents saving-based heuristics for the VRPSPD. In these heuristics, two existing routes are merged in order to create a new route. The feasibility of the new route is checked by employing a cumulative net-pickup approach. Another exact solution approach has been suggested by Subramanian, Uchoa, Pessoa & Ochi (2011). The authors developed a branch-and-cut algorithm for the VRPSPD. The algorithm finds improved lower bounds and several new optimal solutions.

Metaheuristic algorithms have been widely implemented for the VRPSPD. Especially, Single solution based algorithms based on tabu search (TS) have been commonly used. Crispim & Brandao (2005) are the first researchers to use a metaheuristic algorithm for the VRPSPD. In their study, an algorithm constructed by hybridizing TS and variable neighborhood descent (VND) has been developed. Chen & Wu (2006) presented another hybrid metaheuristic algorithm consisting of TS and record to record travel algorithm. In this algorithm, initial solution is constructed by using an insertion based algorithm and subsequently 2-exchange, swap, shift, 2-opt and or-opt neighborhood generation mechanisms are used to improve the initial solution. Montane & Galvao (2006) developed a TS-based heuristic approach in which Shift, cross and 2-opt routines are employed.

Bianchessi & Righini (2007) proposed constructive and local search heuristics and a TS algorithm that contains variable neighborhood structures. Wassan, Wassan & Nagy (2008) developed a reactive TS algorithm for the VRPSPD. The developed algorithm consists of two phases: generating an initial solution and improving on it. A modified sweep algorithm is used to obtain an initial solution and the neighborhood structures which are shift, swap, local shift and reverse, respectively used for improvement. The authors also constructed a mechanism that dynamically controls the tabu list size to provide an effective balance between the intensification and diversification of the search. Zachariadis, Tarantilis & Kiranoudis (2009) presented a hybrid metaheuristic algorithm that combines TS and guided local search (GLS) heuristics. In this study, an initial solution is generated by a saving based constructive heuristic and then the solution is improved by the hybrid TS-GLS

methodology with the neighborhood structures which are customer relocation, customer exchange, route interchange I and route interchange II, respectively.

Zachariadis, Tarantilis & Kiranoudis (2010) suggested a heuristic algorithm based on adaptive memory methodology. The heuristic algorithm collects properties of good solutions found during the search process. New solutions are generated by combining these properties, and subsequently an improvement procedure based on TS is applied. Moreover, an additional memory mechanism has been proposed to provide appropriate diversification in the search. Zachariadis & Kiranoudis (2011) presented a local search algorithm which is capable of exploring rich solution neighborhoods effectively. In order to examine these neighborhood types, the authors use an algorithmic concept, called Static Move Descriptor (SMD), which statically encodes tentative moves. To diversify the search efficiently, another algorithmic framework, called promises concept, which is a variation of the aspiration criteria of TS has been used. Ropke & Pisinger (2006) proposed a large neighborhood search (LNS) algorithm for some VRP variants involving the VRPSPD. A parallel algorithm is developed by Subramanian, Drummond, Bentes, Ochi & Farias (2010). The algorithm is embedded with a multi start heuristic which comprises of VND procedure integrated in an iterated local search (ILS) framework. The developed algorithm automatically calibrates some parameters, which makes it self-adaptive, avoiding the need of manual tuning. Furthermore, the algorithm has the ability of exploring the high-level of parallelism inherent to recent multi-core clusters.

Recently, Population-based algorithms have been also applied for the VRPSPD. Ai & Kachitvichyanukul (2009) proposed a particle swarm optimization algorithm for solving the VRPSPD. In this study, a random key-based encoding and decoding method is applied, and 2-opt and a heuristic approach based on cheapest insertion are implemented for improvement. Gajpal & Abad (2009a) developed an ant colony optimization (ACO) algorithm for the problem. The authors used the nearest neighbor heuristic to construct an initial solution by means of which the trail intensities and parameters are initialized. Then the trail intensities are used to generate an ant solution for each ant. After that, a local search consisting of 2-opt,

customer insertion/interchange and sub-path exchange is implemented on each ant-solution, and elitist ants and trail intensities are updated.

Another heuristic approach based on ACO is suggested by Çatay (2010). In this study, a new saving-based visibility function and pheromone updating rule are proposed. The nearest neighbor heuristic is implemented to generate an initial solution, and a local search consisting of four routines, namely intra-move, intra-swap, inter-move and inter-swap are performed to improve the solutions. Tasan & Gen (2012) presented a metaheuristic approach based on genetic algorithm (GA). In this algorithm, a permutation-based representation is used. Initial population is constructed randomly and genetic operators, crossover and mutation, are implemented on the members of the population. As a selection rule, the roulette wheel selection is applied. Goksal, Karaoglan & Altiparmak (2013) proposed a hybrid metaheuristic algorithm based on PSO in which VND algorithm is implemented for local search. In the hybrid algorithm, PSO is performed to explore good solutions in the solution space and VND is used to improve random solutions selected from the population in each iteration of the algorithm. In addition, an annealing-like strategy is applied so as to preserve the swarm diversity of the PSO. To represent a solution, the authors use the giant tour representation, by this way the splitting procedure proposed by Prins (2004) is adapted for obtaining a feasible solution from the giant tour for the VRPSPD.

The related literature of the VRPSPD involves powerful heuristic methods which are successfully applied to the problem. However, many of them suffer from the parameter tuning process. Thus, this thesis proposes a simple adaptive local search algorithm that does not need any parameter setting since it is developed by hybridizing two parameter-free algorithms which are a SA-inspired algorithm, Self Adaptive Local Search (SALS) and VND, respectively.

CHAPTER THREE
AN OVERVIEW ON METAHEURISTICS

The term metaheuristic corresponds to a type of approximate solution methods for the optimization problems. A metaheuristic is an algorithmic framework which integrates basic heuristic methods in order to investigate the search space effectively. Although many researchers try to suggest a definition, there is no general definition widely accepted among the researchers. These different definitions of the term metaheuristic can be found in Zèapfel, Braune & Bèogl (2010) and Blum & Roli (2003).

One of the most important factors that affect the efficiency of the metaheuristic algorithms is the balance between diversification and intensification. The term diversification can be defined as the ability of the algorithm to discover different search regions in the search space while the term intensification is related to obtain high-quality solutions from the search regions explored. The main features of the metaheuristcs are listed as follows:

- The aim of the metaheuristics is to efficiently explore the search space to reach near optimal or optimal solutions within a reasonable computation time.
- Metaheuristics are not problem dependent.
- Metaheuristics combine search characteristics of different heuristics to yield high-quality solutions.
- The search techniques which are employed in metaheuristics must involve a mechanism to escape from local optima.
- The search experience is widely used to conduct the search process (Blum & Roli, 2003).

There exist several classifications for the metaheuristic algorithms according to different viewpoints such as nature-inspired or non-nature-inspired, population-based or single solution-based, dynamic or static objective function, construction-based or improvement-based etc. Figure 3.1 shows a general scheme for the metaheuristics

from the viewpoint of the number of solutions used at the same time. In the following sections, we briefly describe well-known metaheuristic algorithms which are used to solve the optimization problems.

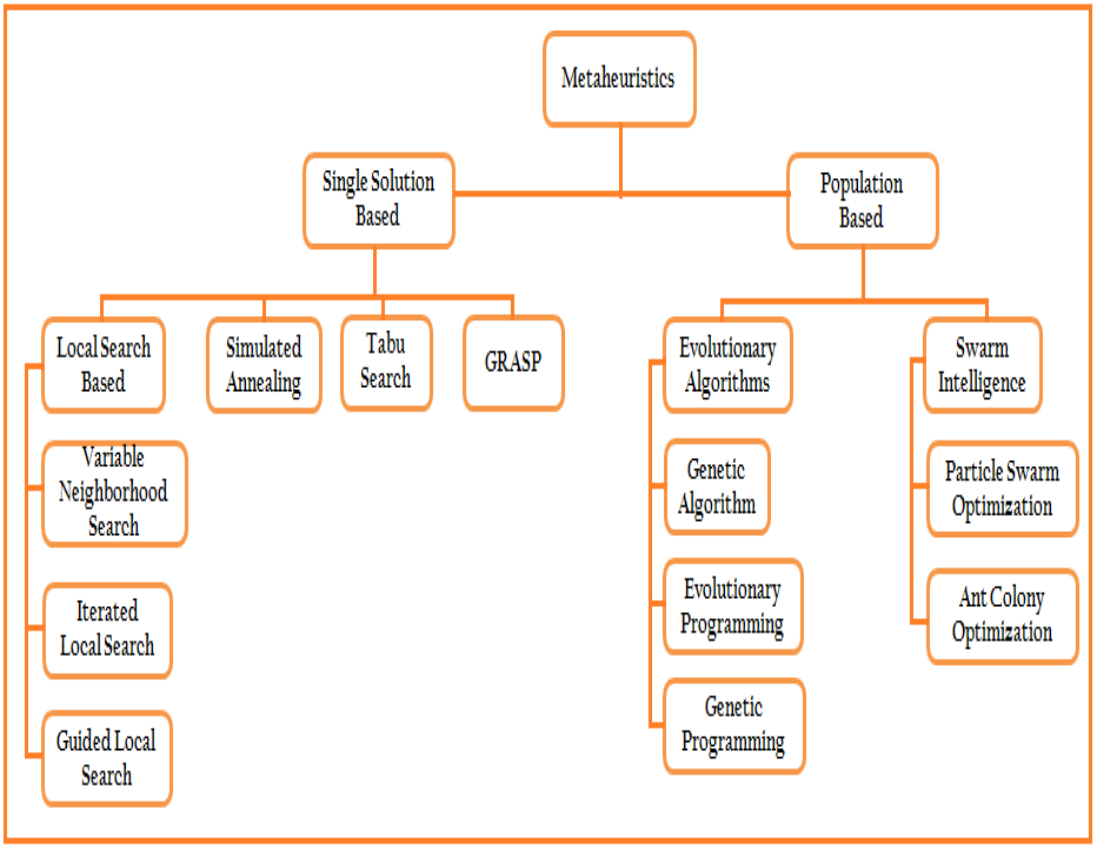


Figure 3.1 A general scheme for metaheuristics

3.1 Simulated Annealing (SA)

Simulated Annealing (SA) is a well-known single solution-based metaheuristic which has been introduced by Kirkpatrick, Gelatt & Vecchi (1983) and Cerny (1985). The algorithm is one of the oldest algorithms to apply a certain mechanism to avoid from local optimum solutions (Blum & Roli, 2003). SA is inspired by the annealing process of materials which requires heating and then slowly cooling a substance to obtain a strong crystalline structure (Talbi, 2009). The strength of the obtained structure is strongly related to the cooling rate. If the cooling process is implemented too fast or the initial temperature is not determined sufficiently high,

the annealing process yields imperfections. Conversely, the annealing process produces strong crystals if the cooling process is applied carefully and slowly.

SA simulates this physical process by using a cooling schedule. From an initial solution, SA iteratively executes the search process. At each iteration, a neighbor solution is generated. If the generated solution is better than the current one, it is accepted as the new current solution. Otherwise, the new solution is accepted with a probability value, $\exp^{-\Delta/T}$, where Δ represents the difference between objective function values of the generated solution and the current one. T represents the current temperature of the algorithm. The higher the value of T, the higher the probability of accepting a worse solution. T is monotonically reduced during iterations and the search is terminated when T decreases to a certain value which is generally called as the final temperature.

One of the most important factors affecting the performance of SA is the design of the cooling schedule. Using a monotone threshold function may not always produce good quality solutions because the algorithm may be trapped in local minima at the end of the search due to the low temperature. Thus, different cooling strategies involving adaptive and non-monotone threshold functions are proposed to overcome this issue. For some applications of non-monotone and adaptive threshold functions, we refer the interested readers to the papers of Hu, Kahng & Tsao (1995), Lin, Haley & Sparks (1995), Azizi & Zolfaghari (2004), Altiparmak & Karaoglan (2008) and Alabas-Uslu & Dengiz (2011).

3.2 Genetic Algorithm (GA)

Genetic Algorithm (GA) is one of the most popular population-based algorithms which simulates the natural behavior of biological systems. The fundamental ideas of GA have been initially proposed by Holland (1975) and the algorithm has become a popular solution method for optimization problems after the study of Goldberg & Holland (1988).

GA involves a population that consists of individuals representing feasible solutions to the problem. Each individual has a fitness value which corresponds to the objective function value of the solution. New individuals, children, are created by implementing genetic operators to the existing individuals, parent solutions. These genetic operators are called as crossover and mutation. Crossover operators create new individuals by combining parts of different individuals while mutation operators produce a new individual by making changes on a single individual. Thus, crossover operators provide characteristics of the parents to the new individuals while mutation operators maintain diversification of the population. After a predetermined number of generations which is also called as iteration number, the algorithm converges to an individual that has the best fitness value. The obtained solution represents an optimum or near optimum solution to the problem.

3.3 Tabu Search (TS)

Tabu Search (TS) is a deterministic search technique inspired by human memory processes to solve optimization problems. The algorithm is first introduced by Glover (1990). TS works based on neighborhood search and uses search experience to escape from local optima. The algorithm employs a tabu list consisting of prohibited solutions. The tabu list prevents the algorithm from cycling and enables the search process to explore different search regions. Another factor that affects the acceptance condition of newly generated solutions is the aspiration criteria. It allows us to accept a solution even if it is available in the tabu list. The most widely applied aspiration criterion is to accept a solution with higher quality than the current best one (Blum & Roli). The main purpose of the aspiration criteria is to make the algorithm more flexible (Altiparmak & Karaoglan, 2008).

3.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is one of the most popular evolutionary algorithms that has been introduced by Kennedy & Eberhart (1995). The development of the algorithm is based on the social behavior of some species and

living animals (Goksal, Karaoglan & Altiparmak, 2013). Each particle in the swarm represents a single solution to the problem. The particles in the swarm move around the search space according to the information which is related with the velocity of the solution, personal best objective function value, current objective function value and the global best objective function value. The aim of all particles is to improve the best objective function value found during the iterations.

3.5 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is another population-based metaheuristic solution method that is widely used for optimization problems. Since it was first introduced by Dorigo, Maniezzo & Colorni (1996), many applications of the algorithm have been proposed for several combinatorial optimization problems such as quadratic assignment problem, scheduling problems, vehicle routing problems etc. (Çatay, 2010). The algorithm is inspired by the behavior of real ant colonies searching for food sources. Real ants spread an aromatic essence which is called as pheromone to the path they walk. The accumulated amount of pheromone on a path is changed according to the length of the path and the quality of the food source. If a path is followed by many ants, the probability of selecting the corresponding path by other ants increases.

In ACO, artificial ants are employed for obtaining high-quality solutions to the optimization problems, which are converted to the problem of finding the best path on a weighted graph (Çatay, 2010). The artificial ants move on the graph and the solutions are constructed gradually by applying a stochastic construction process (Dorigo, 2007).

Considering our developed methodology, it is a single solution-based algorithm based on two parameter-free heuristics which are a SA-inspired algorithm, Self Adaptive Local Search (SALS), and Variable Neighborhood Descent (VND). From the solution construction viewpoint, the algorithm can be classified as an improvement based solution method. Besides, the constructed method presents a

general search procedure for the optimization problems. Therefore, it can be applied to many combinatorial optimization problems. The details of the proposed methodology will be given in the next chapter.

CHAPTER FOUR
SOLUTION METHODOLOGY

In this chapter, our proposed solution methodology for the VRPSPD is introduced. Firstly, we give information about the self-adaptive local search and the variable neighborhood descent algorithms that constitute our hybridized solution method. After that, we describe the details of the developed solution methodology.

4.1 Self-Adaptive Local Search (SALS)

Self-adaptive local search (SALS) is a Simulated Annealing (SA) inspired metaheuristic algorithm proposed by Alabas-Uslu & Dengiz (2011). The algorithm involves a non-monotone threshold accepting function with only one generic parameter called acceptance parameter. The parameter is adjusted automatically during the search process according to the information received from the problem and performance measure of the algorithm. The main feature of the algorithm is that it never requires a parameter tuning, which simplifies its application to the optimization problems.

As we mentioned earlier, SA is a stochastic search method which has a threshold function used to escape from local optima. For a minimization problem, the search begins from an initial solution, x, it then generates a new candidate solution, x', with a specified neighbor generation method. The difference between the objective function values, $\Delta=f(x')-f(x)$, is calculated. If Δ is non-positive, then the candidate solution is accepted as a new current solution. Otherwise, it is accepted with a probability value, $\exp^{-\Delta/T}$, where T is a control parameter that corresponds to the temperature of the annealing schedule. T generally starts from a high value, and monotonically decreases throughout the search. The algorithm is terminated when T reaches a pre-determined value, called final temperature.

In a similar fashion with SA, starting from an initial solution, x, SALS generates a new solution, x', with a neighbor generation procedure at each iteration. Whether the

generated solution is accepted or not is determined according to the following acceptance condition: if $f(x') \leq t.f(x)$, then $x \leftarrow x'$. In this situation, t represents the self-adaptive parameter of SALS. When the candidate solution is accepted, the algorithm proceeds to the next iteration. Otherwise, a new candidate solution is generated. The value of t is increased by the equation of $t=t+a_1.a_2$ when the number of consecutive rejected solutions is equal the neighborhood size of the current solution. (Alabas-Uslu & Dengiz, 2011). The only parameter of SALS, t, is updated during the search according to two criteria: the number of improved solutions obtained during the search process and the ratio of best solution to current solution at each iteration. Therefore, two performance indicators which are obtained by Equations (4.1) and (4.2) are used to manipulate t. In these equations, C_i represents the number of improved solutions found through the search, and x_b and x represent the best solution and the current solution at iteration i, respectively. Parameter t is calculated throughout the search process by Equation (4.3). Whenever a new improved solution is found, a_1 decreases while a_2 increases. The value of C_i gives information about the search space structure. High values of C_i indicate that the search space has many local optimum points. On the other hand, when the search space is smooth, C_i takes low values.

$$a_1=f(x_b)/f(x) \quad (4.1)$$

$$a_2=C_i/i \quad (4.2)$$

$$t=1+a_1.a_2 \quad (4.3)$$

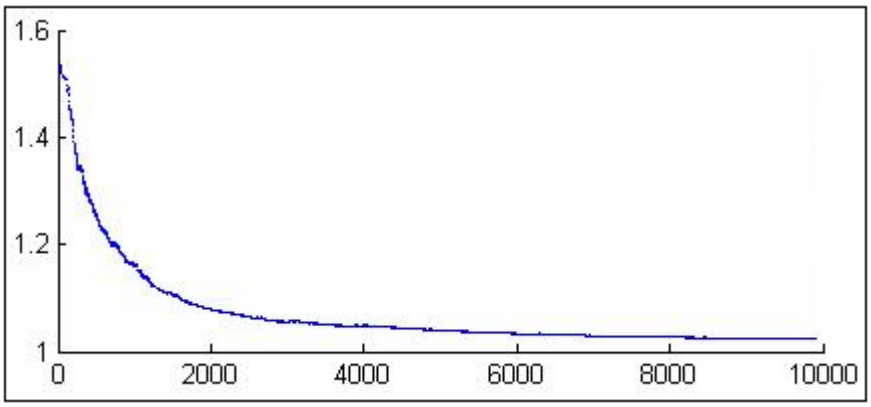


Figure 4.1 Changing of the value of t during iterations

The changing of the value of t during iterations and the main steps of the algorithm are given in Figure 4.1 and Figure 4.2, respectively. At the early stages of the algorithm, the search is nearly random because of the high values of t. As the number of iterations increases, the value of t decreases non-monotonically according to the values of a_1 and a_2. If the search space has many local optima, the value of t decreases gradually and this situation makes a positive impact on the diversification of the algorithm. Conversely, if the number of improved solutions is low, the value of t decreases more rapidly in comparison to the previous situation. Hence, intensification of the algorithm is increased.

```
Procedure SALS
Input: Problem data, Neighborhood Structures
Output: Best solution, xb;
Construct a random initial solution, x;
N←Neighborhood size of x;
Ci←1;
i←1;
xb←x;
age←0;
a1←f(xb)/f(x);
a2←Ci/i;
t←1+a1.a2;
while t≠1
    Generate a random neighbor solution, x' of N(x);
    if f(x')<=t.f(x)
        x←x';
        i←i+1;
        age←0;
        if f(x)<f(xb)
            xb←x;
            Ci←Ci+1;
        end
        a1←f(xb)/f(x);
        a2←Ci/i;
        t←1+a1.a2;
    else
        age←age+1;
        if age==N
            t←t+a1.a2;
            age←0;
        end
    end
```

Figure 4.2 Steps of SALS algorithm

4.2 Variable Neighborhood Descent (VND)

Variable Neighborhood Descent (VND) is a deterministic version of the Variable Neighborhood Search (VNS) which is a well-known local search algorithm proposed by Hansen & Mladenovic (1997). The algorithm is commonly integrated with other metaheuristic algorithms in order to increase the intensification of the search. The main idea of the algorithm is to explore the search region by using different neighborhood structures successively. The basic steps of VND are given in Figure 4.3. At the beginning of the algorithm, VND executes the local search with the first neighborhood structure k=1 until no improved solutions can be found. Then the local search continues with the second neighborhood structure k=2. If an improvement has been found with the second neighborhood structure, VND returns to the first neighborhood structure to restart the search. Otherwise, it proceeds to the third structure, and so on. If the last neighborhood structure $k=k_{max}$ yields no improved solutions, VND is finished and the obtained solution is considered as a local optimum with respect to all neighborhood structures.

```
Procedure VND
Input: x, neighborhood structures
Output: x
k←1
while k≤k_max
   find the best neighbor (x') of x with respect to the k^th
   neighborhood structure;
   if f(x')<f(x)
       x←x';
       k←1;
   else
       k←k+1;
   end
end
```

Figure 4.3 Steps of VND algorithm

4.3 Proposed Local Search Algorithm (SALS-VND)

As mentioned earlier, SALS is a parameter-free algorithm and this special feature makes its integration with other heuristic methods easier. The use of a threshold

function that starts from a high value diversifies the search process and allows the algorithm to explore different regions in the search space. Besides, the non-monotone nature of the threshold function makes it easier to escape from local minima at the end of the search. Although SALS provides sufficient diversification to the search, it may require more intensification as the algorithm encounters different search regions. Thus, hybridizing SALS with another local search heuristic that leads to an intensification effect enables the solution method to obtain more qualified solutions from the search regions examined. Therefore, this thesis proposes a two-stage solution strategy based on SALS and VND. The basic stages of the algorithm are depicted in Figure 4.4. The procedures hybrid_SALS and VND_perturbation represent the first and the second stages of the algorithm, respectively. In the first stage, SALS is integrated with VND in a way that whenever a new improved solution is found by SALS, VND is implemented on the solution to obtain a more qualified solution. In the second stage which can be considered as a post optimizer, VND is used with a perturbation mechanism to further improve the solution obtained from the first stage. In the following subsections, we give information about the details of the proposed algorithm.

```
Procedure SALS-VND
Input: Problem data
Output: x
Construct a random initial solution, x;
x'←hybrid_SALS(x, iter_sals, neighborhood structures);
x''←VND_perturbation(x', iter_vnd, max_d);
x←x'';
```

Figure 4.4 Basic steps of the proposed algorithm

4.3.1 Solution Representation

A solution representation is given for a problem with nine customers and two vehicles in Figure 4.5. A solution x for the VRPSPD is represented as a vector that consists of a sequence of nodes, with dimension D=n+m+1, where n is the number of customers and m is the number of vehicles. The node 0 represents the depot and the other nodes represent customers. The solution x is represented as x=[0 1 2 3 4 0 5 6 7

8 9 0] where the first route starts from the depot and visits customers 1,2,3 and 4, respectively, and returns to the depot. Similarly, the second route, which is initiated and terminated at the depot, services customers 5,6,7,8 and 9, respectively.

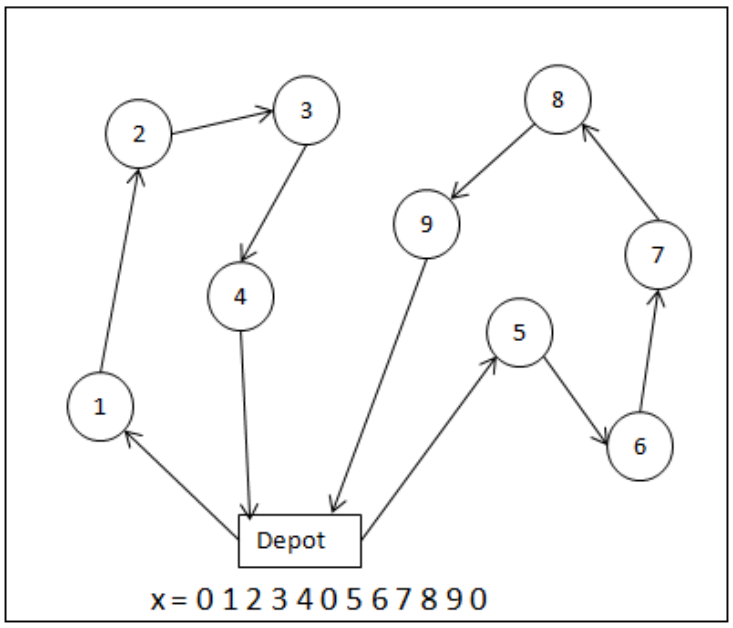


Figure 4.5 Solution representation

4.3.2 Generating Initial Solution

Generally, two different initial solution construction strategies are employed by the researchers for the vehicle routing problems. The first strategy is to use a heuristic method to generate an initial solution and the second one is to construct it randomly. Using a heuristic algorithm initiates the search process from a better quality solution and it can accelerate the convergence of the search method. But, it has a negative effect on the diversification of the search by preventing the algorithm from exploring different search regions. For this reason, we use a random solution construction procedure which is given in Figure 4.6 for the VRPSPD. The procedure starts with an empty route and a customer list, CL. A randomly selected customer from CL is inserted into the empty route, set as the current route and the inserted customer is deleted from CL. The addition of a randomly selected customer to the end of the current route is repeated until it becomes infeasible, and the added customers are deleted from the customer list, CL. The feasibility of the current route

is checked by using feasibility_revealer procedure. When the route becomes infeasible, the current route is adding to the initial solution and another empty route is generated. The selected customer which cannot be added to the previous route is inserted into the new one. The procedure continues until no customer remains in CL.

```
Procedure initial_solution_generator
Input: Customer data
Output: initial solution x
initialize the customer list CL={1,...,n};
create an empty vehicle route and set the route as current route;
select a random customer n_r from CL;
insert n_r into the current route as the first customer;
remove n_r from CL;
exit←0;
while exit==0
   select a random customer n_r from CL;
   generate a trial route tr by adding n_r to the end of the
   current route;
   (feasibility, tr)=feasibility_revealer(tr, customer data,
   VC);
   if feasibility==1 // the trial route is feasible
       set the trial route as current route;
   else
       add the current route to the initial solution;
       create a new empty vehicle route and set the route as current
       route;
       insert n_r into the current route as the first customer;
   end
   remove n_r from CL;
   if CL==0
       exit←1;
   end
end
```

Figure 4.6 Initial solution construction procedure

4.3.3 Checking Feasibility of the Capacity Constraint

As mentioned before, one of the most critical features of the VRPSPD is the non-monotonic change of the amount of good carried by a vehicle during the distribution and collection process. In order to check the feasibility of a new constructed route, we apply a checking procedure described in Figure 4.7. The procedure starts with calculating the total amount of the pickup and delivery amounts of the route examined. If either of them exceeds the vehicle capacity, the solution is infeasible. Otherwise, the loads of the vehicle after servicing the customers are calculated. If the load exceeds the vehicle capacity, we reverse the direction of the route due to the fact

that the reversed version of an infeasible route may be feasible owing to the fluctuating nature of the load. If the reversed version of the route is also infeasible, then the solution is determined to be infeasible. Otherwise, the solution is considered to be feasible with the reversed version of the route.

```
Procedure feasibility_revealer
Input: x, customer data, vehicle capacity (VC)
Output: feasibility, x
for i=1:r // r=number of routes
      P←Cumulative pickup amount of the r_i;
      D←Cumulative delivery amount of the r_i;
      if D<VC && P<VC
          load←D;
          feasibility←1;
          for j=1:m // m=number of customers in the r_i
                load←load+p_j-d_j
                if load>VC
                   r_i←reverse(r_i);
                   load←D;
                   for k=1:m
                         load←load+p_k-d_k; // j^th customer in r_i
                         if load>VC
                            feasibility←0;
                            break;
                         end
                   end
                end
                if feasibility==0
                    break;
                end
          end
      else
          feasibility←0;
      end
      if feasibility==0
          break;
      end
end
```

Figure 4.7 Feasibility checking procedure

4.3.4 Neighborhood Structures

Ten neighborhood structures which are widely used for the VRP are applied in SALS and VND. Five of them are inter-route movements while others are intra-route ones. Figure 4.8 and Figure 4.9 illustrate the intra-route and inter-route neighborhood structures, respectively.

4.3.4.1 Adjacent Swap

The first intra-route neighborhood structure, Adjacent Swap, exchanges the positions of two adjacent nodes. In Figure 4.8 (a), the positions of adjacent nodes 2 and 3 are exchanged.

4.3.4.2 General Swap

The second structure, General Swap, exchanges the positions of any node pair located in the same route. In Figure 4.8 (b), the positions of nodes 1 and 3 are exchanged.

4.3.4.3 Single Insertion

In this structure, a node is removed and inserted between two adjacent nodes located in the same route. In Figure 4.8 (c), the node 1 is removed and inserted between nodes 4 and 5.

4.3.4.4 Block Insertion

The fourth intra-route structure, Block Insertion, removes two adjacent nodes and inserts them between another adjacent node pair. In Figure 4.8 (d), the adjacent nodes 1 and 2 are inserted between node 4 and 5.

4.3.4.5 2-opt

The final intra-route structure, 2-opt, replaces a non-adjacent arc pair with a new one, which reverses the location of nodes lying between these new arcs. In Figure 4.8 (e), the non-adjacent arcs (2,3) and (5,0) are deleted while the arcs (2,5) and (3,0) are inserted.

4.3.4.6 Shift(1,0)

The first inter-route structure, Shift(1,0), removes a customer from its current route and inserts it at a location in another route. In Figure 4.9 (a), the node 9 is removed and inserted between the adjacent nodes 3 and 4.

4.3.4.7 Swap(1,1)

In this structure, the positions of any node pair located in different routes are exchanged. In Figure 4.9 (b), the positions of nodes 4 and 8 are exchanged.

4.3.4.8 Shift(2,0)

This structure removes an adjacent node pair and transfers them to another route. In Figure 4.9 (c), the adjacent nodes 5 and 6 are removed and inserted between nodes 4 and 0.

4.3.4.9 Swap(2,1)

This structure exchanges the positions of two adjacent customers with a customer from a different route. In Figure 4.9 (d), the adjacent nodes 7 and 8 are exchanged with node 4.

4.3.4.10 Swap(2,2)

This structure exchanges the positions of two adjacent node pairs from different routes. In Figure 4.9 (e), the adjacent nodes 1 and 2 are exchanged with the adjacent nodes 8 and 9.

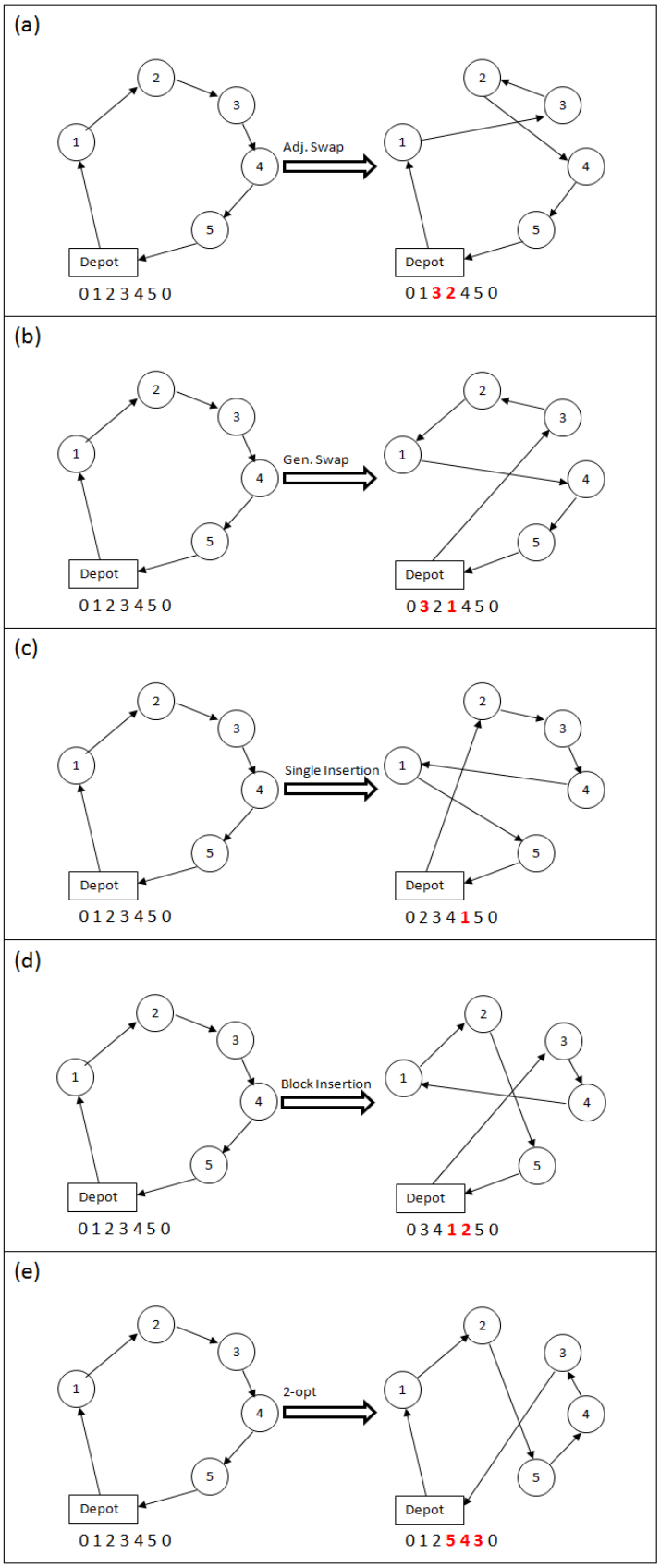


Figure 4.8 Intra-route neighborhood structures

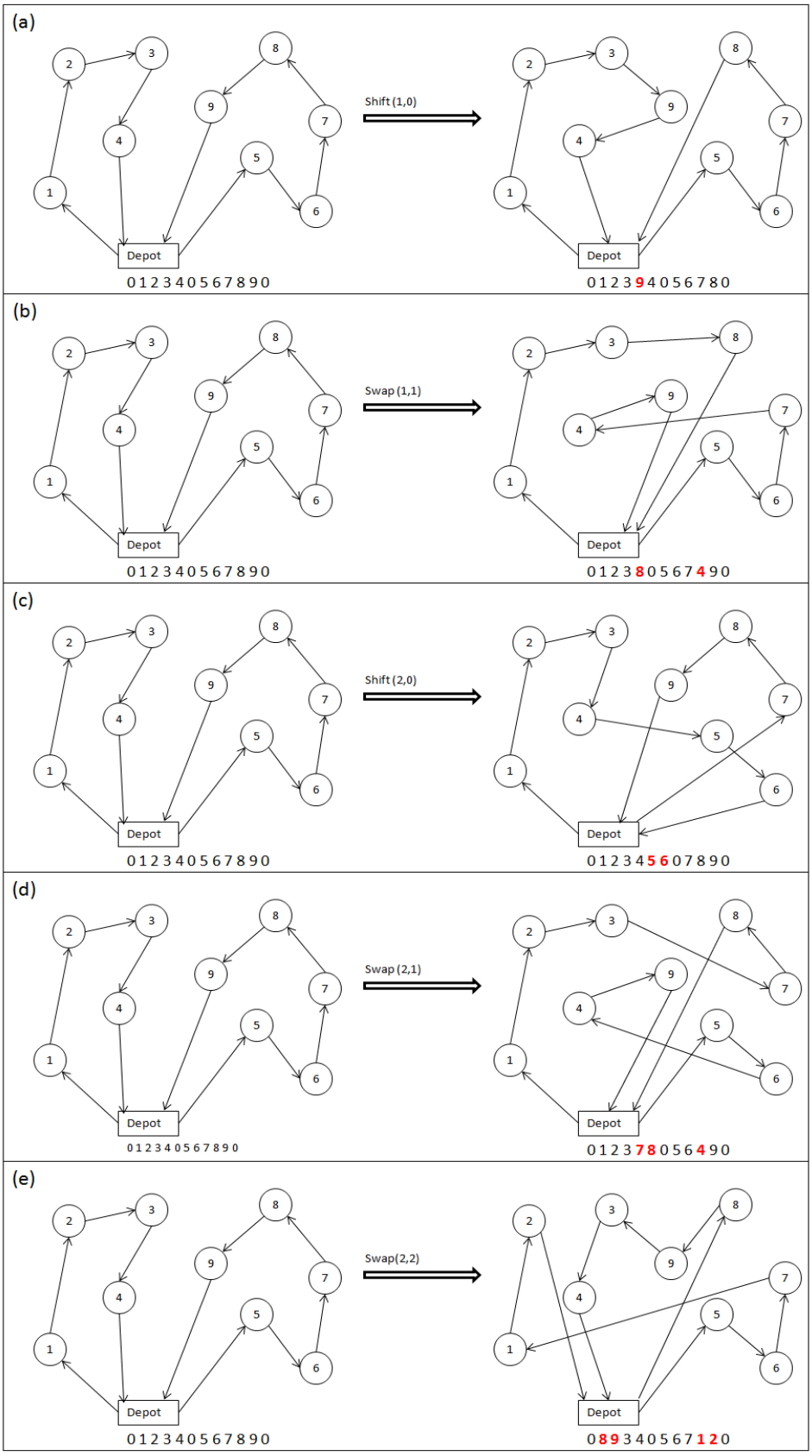


Figure 4.9 Inter-route neighborhood structures

4.3.5 Stage I: Integration of SALS with VND (hybrid_SALS)

In the first stage of the algorithm, we apply SALS for discovering different search regions in the search space whereas VND is used to increase the intensification of the search process (see Figure 4.10). Thus, SALS is employed as our global search technique and VND is used as our regional search method. The flowchart and the main steps and of the first stage of the algorithm are given in Figure 4.11 and Figure 4.12 where x_{lb} represents the best solution obtained from SALS, and x_{gb} represents the global best solution found after applying VND. The neighbor solutions are obtained by applying each neighborhood structure described in Figure 4.8 and Figure 4.9. Whenever x_{lb} is updated, our VND algorithm, described in Figure 4.13, is triggered for improvement. The first phase is terminated when total number of consecutive iterations without improvement, m_1, reaches a predetermined value which is represented as iter_sals in Figure 4.12.

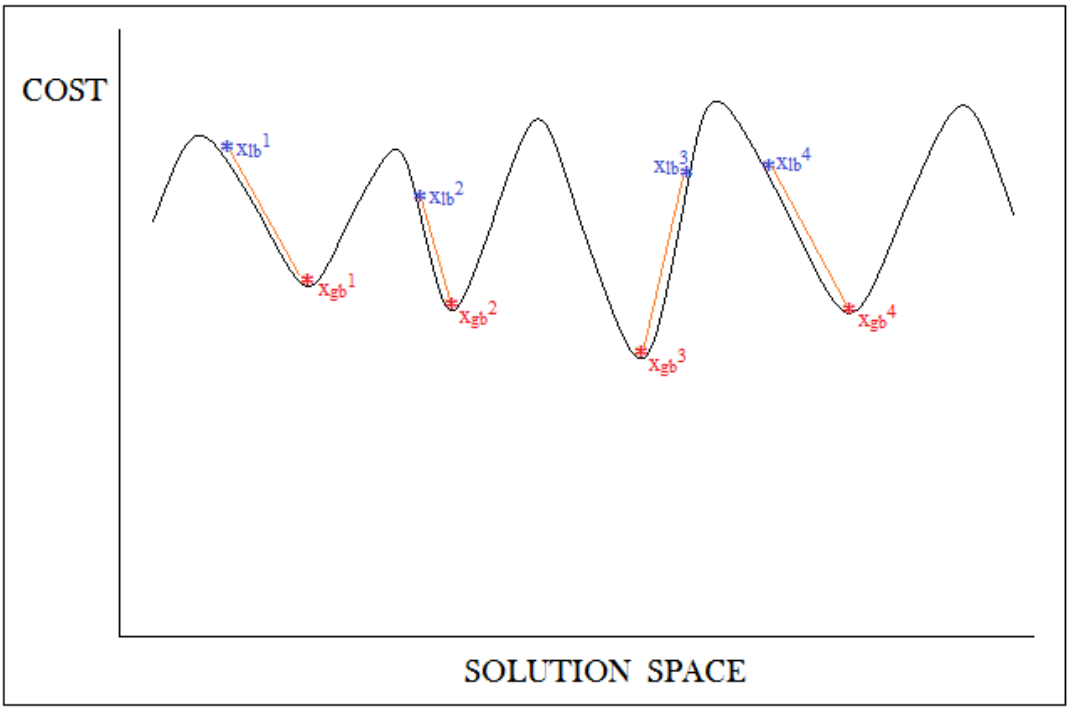


Figure 4.10 Visualization of the first phase of the algorithm. Improved solution points, x_{lb}, found by SALS from different regions are used as initial solutions for VND. By applying VND, the solution points, x_{gb}, yielding less cost are obtained.

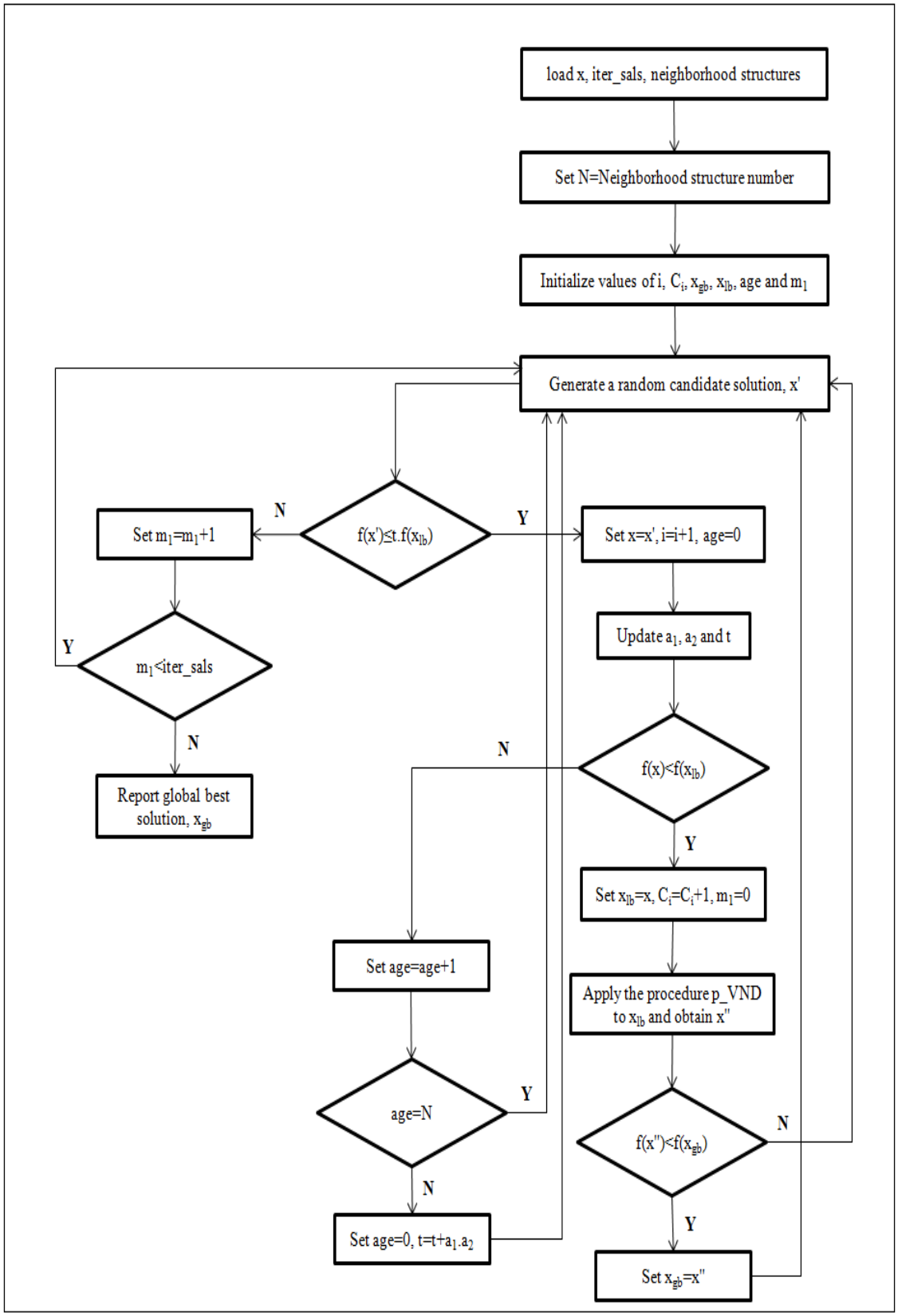


Figure 4.11 Flowchart of the first stage of the algorithm (hybrid_SALS)

```
Procedure hybrid_SALS
Input: x, iter_sals, neighborhood structures
Output: Global best solution, x_gb
N←Neighborhood structure number;
C_1←1;
i←1;
x_gb←x;
x_lb←x;
age←0;
m_1←0;
a_1←f(x_lb)/f(x);
a_2←C_1/i;
t←1+a_1.a_2;
while m1<iter_sals
    Generate random neighbor solutions of x and select the best one as
    candidate solution, x';
    if f(x')<=t.f(x_lb)
        x←x';
        i←i+1;
        age←0;
        if f(x)<f(x_lb)
            m_1←0;
            x_lb←x;
            C_1←C_1+1;
            x''←p_VND(x_lb, NS_inter);
            if f(x'')<f(x_gb)
                x_gb←x'';
            end
        else
            m_1←m_1+1;
        end
        a_1←f(x_lb)/f(x);
        a_2←C_1/i;
        t←1+a_1.a_2;
    else
        age←age+1;
        if age==N
            age←0;
            t←t+a_1.a_2;
        end
    end
end
```

Figure 4.12 First stage of the proposed algorithm (hybrid_SALS)

Two slight modifications are applied to the original version of SALS algorithm. The first modification is implemented on the threshold function. Instead of using the objective function value of the current solution, x_{lb} is employed. Preliminary tests show that this modification produces better results than the original version. The second modification is to decrease the total number of rejected candidate solutions before increasing the value of t. In order to make the search process more sensible to the number of consecutive rejected neighbors, we increase the value of t when the total number of rejected solutions is equal to the number of neighborhood structures

instead of the neighborhood size of the current solution. This modification increases the diversification of the search.

4.3.6 Applied VND Algorithm (p_VND)

VND algorithm is employed at both stages of the algorithm to intensify the search process. The basic steps of the algorithm are illustrated in Figure 4.13. We utilize five inter-route neighborhood structures, Swap(1,1), Shift(1,0), Swap(2,1), Shift(2,0) and Swap (2,2), for VND. Whenever a new improved solution is obtained, the procedure VND_intra is implemented on the routes modified by the current inter-route structure to further improve the quality of the solution. When the procedure is called, four intra-route neighborhood structures, General Swap, 2-opt, Single Insertion and Block Insertion are exhaustively applied to the modified routes. The strategy of using intra-route structures after obtaining an improvement by implementing inter-route ones can be seen in Subramanian et al. (2010) and Goksal, Karaoglan & Altiparmak (2013).

The order of the neighborhood structures which influences the performance of the algorithm is determined based on our experiments. In these experiments, we investigate the performances of different neighborhood orderings by applying them to 20 different initial solution instances. The neighborhood orderings are constructed as follows:

- Ordering 1: Shift(1,0), Swap(1,1), Shift(2,0), Swap(2,2) and Swap (2,1)
- Ordering 2: Swap(1,1), Shift(1,0), Shift(2,0), Swap(2,1) and Swap (2,2)
- Ordering 3: Swap(1,1), Swap(2,2), Swap(2,1), Shift(1,0) and Shift (2,0)
- Ordering 4: Shift(1,0), Swap(2,1), Shift(2,0), Swap(1,1) and Swap (2,2)
- Ordering 5: Swap(2,1), Swap(1,1), Shift(1,0), Swap(2,2) and Shift (2,0)
- Ordering 6: Shift(2,0), Shift(1,0), Swap(2,1), Swap(1,1) and Swap (2,2)
- Ordering 7: Swap(2,2), Shift(2,0), Swap(2,1), Shift(1,0) and Swap (1,1)

The computational results of different neighborhood orderings are presented in Table 4.1. The average solution values for all neighborhood orderings (Avg. sol.) are reported in the last row. As seen from the table, there is no one single neighborhood ordering which outperforms all other orderings in all instances. The performances of the neighborhood orderings change according to the solution point that is used as an initial solution for VND. However, considering the average solution values we observe that the second ordering yields better results than the other ones. Thus, we apply the second neighborhood ordering for VND.

4.3.7 Stage II: Application of VND with a Perturbation Mechanism (VND_perturbation)

In the second stage of the algorithm, VND is integrated with a perturbation mechanism which is developed for this study. In this phase, VND is implemented to reach local optimum points while the perturbation mechanism is applied to escape from the local optimum points found by VND (see Figure 4.14).

Design of the perturbation mechanism in this stage affects the performance of the algorithm significantly. If the mechanism yields too small perturbation, VND may generate the same local optimum points and no further improvement can be obtained. On the other hand, applying an exaggerate perturbation may drive the search process to undesirable regions in the search space. Thus, the characteristics of the moving types and the number of moves applied in the mechanism have crucial importance for obtaining an achievement from the method. Moreover, the required perturbation length changes according to the search space structure and using a fixed perturbation may not produce qualified solutions. In order to overcome these difficulties, we propose an adaptive perturbation mechanism which adjusts the perturbation length according to the structure of the landscape.

```
Procedure p_VND
Input: x, inter-route neighborhood structures NS_inter
Output: x
k←1
while k≤k_max
   find the best neighbor (x') of x with respect to the k^th
neighborhood structure;
   if f(x')<f(x)
        x''←VND_intra(x', NS_intra);
        x←x'';
        k←1;
   else
        k←k+1;
   end
end
```

Figure 4.13 Main steps of the applied VND algorithm (p_VND)

Table 4.1 Computational results of different neighborhood orderings

Sol. P.	Ord-1	Ord -2	Ord -3	Ord -4	Ord -5	Ord -6	Ord -7
1	**684.63**	698.92	694.39	688.12	689.17	702.85	696.18
2	794.65	**650.87**	675.49	659.19	700.35	690.17	719.05
3	660.53	686.14	693.87	666.18	**655.29**	675.02	684.62
4	**675.80**	683.02	715.26	702.34	693.13	680.47	684.61
5	692.71	696.92	698.37	716.34	692.71	704.14	**688.29**
6	696.30	723.20	**689.63**	703.25	690.83	726.22	699.51
7	700.97	**684.81**	713.17	692.73	707.16	688.86	701.09
8	**685.73**	699.54	688.33	717.24	**685.73**	687.14	696.36
9	695.18	**691.26**	705.88	712.76	704.51	699.96	698.93
10	715.72	713.81	725.41	733.81	719.67	**710.31**	715.72
11	666.16	682.35	**661.86**	674.17	695.93	672.17	686.54
12	**658.85**	**658.85**	663.59	667.48	659.14	658.85	666.18
13	683.41	**681.05**	693.81	696.36	691.58	700.03	691.77
14	**681.47**	**681.47**	689.26	695.18	691.56	685.10	686.39
15	702.00	**700.88**	712.60	708.34	708.34	706.13	722.53
16	681.28	684.42	669.48	**660.17**	699.48	714.69	692.91
17	**657.18**	**657.18**	693.54	**657.18**	**657.18**	687.60	662.25
18	671.39	671.39	685.24	706.32	698.67	655.19	689.76
19	705.78	**669.78**	702.44	675.18	691.53	699.18	715.82
20	665.69	687.10	675.28	**657.72**	685.15	677.40	665.69
Avg. sol.	688.77	685.15	692.35	689.50	690.86	691.07	693.21

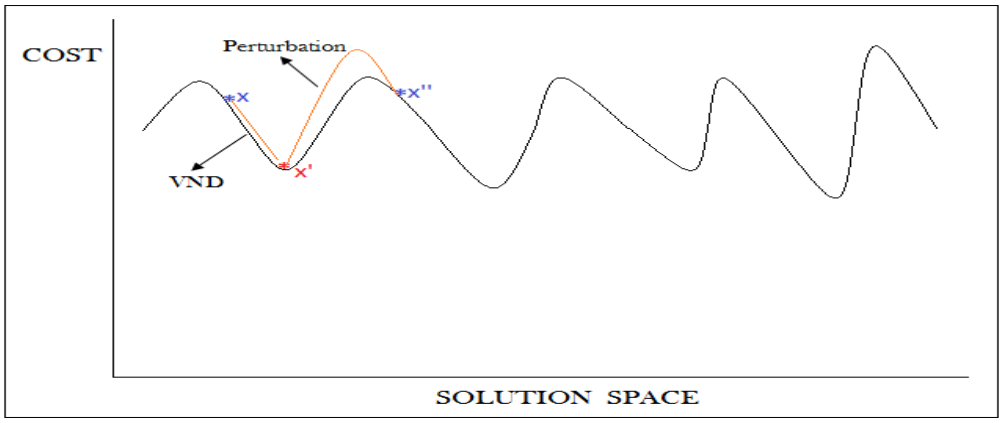


Figure 4.14 Visualization of the second phase of the algorithm. Starting from an initial solution x, VND is implemented and a local minimum point x' is found. By applying the perturbation mechanism, different search regions can be explored.

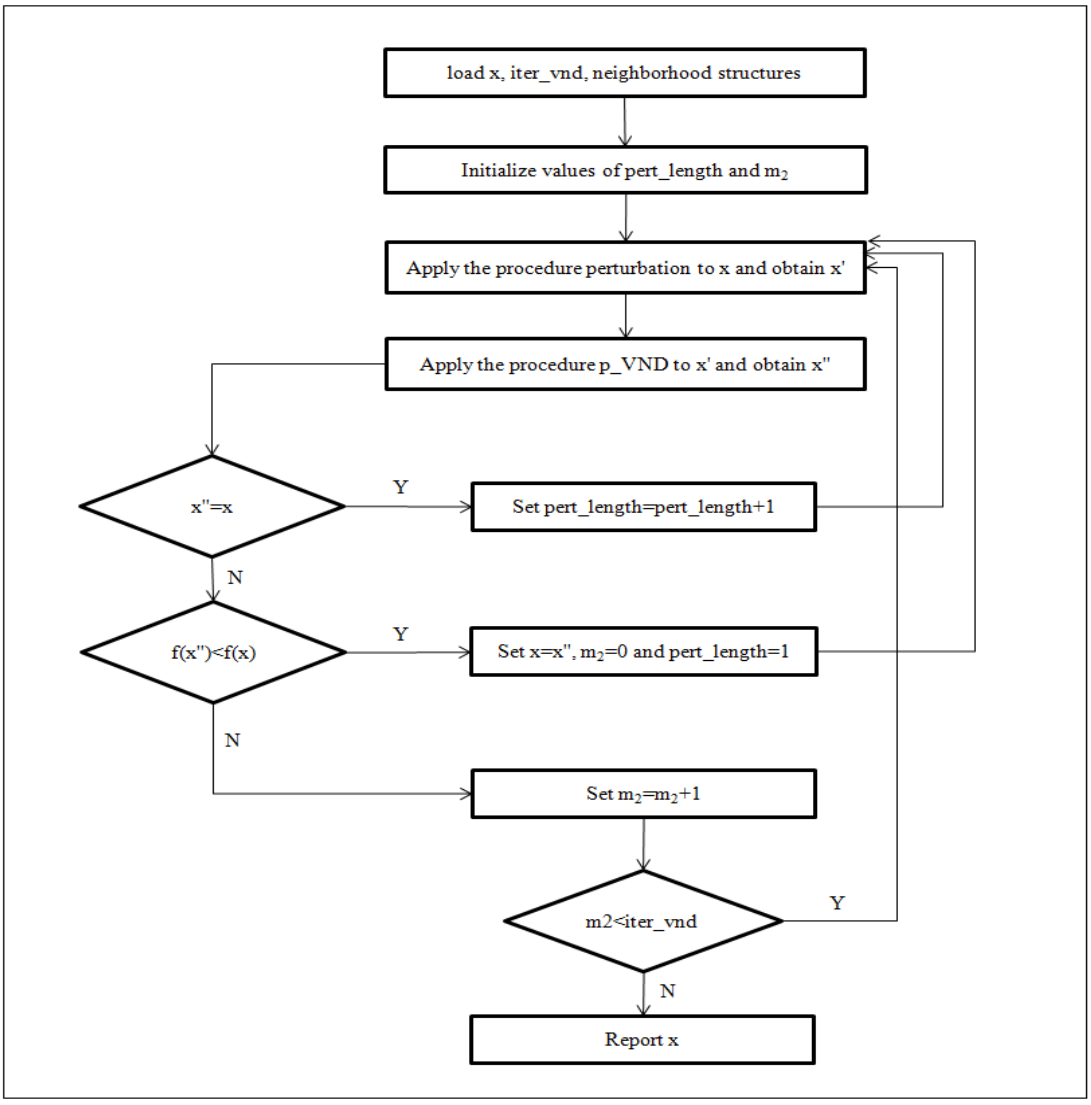


Figure 4.15 Flowchart of the second stage of the algorithm (VND_perturbation)

The flowchart and the basic steps of the second stage are given in Figure 4.15 and Figure 4.16 where m_2 and pert_length represent the number of consecutive rejected solutions and the number of moves to be applied in the perturbation mechanism, respectively. At each iteration of the stage, the perturbation and VND procedures are successively implemented to the current solution, x. If the obtained solution, x", is the same with the current solution, which signals that the applied perturbation length is too small, pert_length is increased by one unit and the same procedures are repeated. This mechanism enables us to calibrate the perturbation length according to the problem specifications. When the solution produced by VND is better than the current one, the values of m_2 and pert_length are initialized. The second stage is terminated when the value of m_2 reaches a pre-determined value which is represented as iter_vnd in Figure 4.16.

```
Procedure VND_perturbation
Input: x, iter_vnd
Output: x
m2←0;
pert_length←1;
while m2<iter_vnd
    x'←perturbation(x, pert_length, λ, max_try);
    x''←p_VND(x', NS_inter);
    if x''==x
        pert_length←pert_length+1;
    else
       if f(x'')<f(x)
           x←x'';
           m2←0;
           pert_length←1;
       else
           m2←m2+1;
       end
    end
end
```

Figure 4.16 Second phase of the proposed algorithm (VND_perturbation)

The proposed perturbation mechanism is presented in Figure 4.17. Only the inter-route neighborhood structures explained in previous sections are used in the mechanism because the intra-route ones cannot produce sufficient diversification

based on our preliminary tests. To adjust the perturbation length, not only determining the number of moves to be applied is required but also the impact of each move should be controlled. Therefore, we use a threshold function by means of which we prevent the perturbation procedure from excessive moves. At each iteration of the mechanism, one of the inter-route neighborhood structures is randomly selected and a candidate solution for perturbation, x', is generated randomly by using the neighborhood structure. The candidate solution is accepted according to the following condition: if $f(x') \leq \gamma . f(x)$, then $x_p \leftarrow x'$. In this condition, x_p and γ represent the perturbed solution and the acceptance parameter, respectively. If the candidate solution is accepted, γ is increased by λ that is a constant value between 0 and 1. Otherwise, a new candidate solution is generated to satisfy the acceptance condition. If the number of consecutive rejected solutions reaches a predetermined value represented as max_try in Figure 4.17, γ is increased by λ again to relax the acceptance condition. When the total number of applied moves equals to pert_length, the procedure is terminated. We set the value of λ as 0.01 and allow maximum 100 consecutive trials before increasing the value of γ.

```
Procedure perturbation
Input: x, pert_length, λ, max_try
Output: xp
p←1;
γ←1+λ;
p_try←0;
xp←x;
while p≤pert_length
   Generate a random solution x';
   if f(x') ≤γ.f(x)
        xp← x';
        p←p+1;
        γ←γ+λ;
        p_try←0;
   else
        p_try←p_try+1;
        if p_try==max_try
           γ←γ+λ;
           p_try←0;
        end
   end
end
```

Figure 4.17 Proposed perturbation mechanism (perturbation)

In order to investigate the effect of the second stage to the performance of the algorithm, six problem instances consisting of 50 to 100 customers are randomly generated. For the instances no. 1, 3 and 5, the coordinate values of the customers are uniformly distributed within [0,100] while the coordinate values of the customers for other instances are uniformly distributed in [0,50]. For each problem instance, the algorithm runs for ten times.

The results of the computational experiments are presented in Table 4.2. For each problem instance, the average solution values of SALS-VND and hybrid_SALS (Avg. sol.), average computation times (ACT) and the percentage of improvement made by the second stage (Imp. Per.) are reported. As can be seen from the table, the second stage of the algorithm improves the solutions obtained from the first stage (hybrid_SALS) by the percentage of 0.975 on average. Considering the average computation times, the second stage constitutes only 10 percent of the computation times on average. Therefore, we can state that the second stage improves the solutions of the first stage while inducing acceptable computational burden.

Table 4.2 Computational results of SALS-VND and hybrid_SALS

Instance	n	hybrid_SALS		SALS-VND		
		Avg. sol.	ACT (min.)	Avg. sol.	ACT (min.)	Imp. Per. (%)
1	50	817.57	9.00	809.42	10.25	1.007
2	50	446.81	8.19	444.56	9.88	0.506
3	75	906.76	21.84	898.06	24.06	0.969
4	75	448.31	18.55	444.10	20.50	0.948
5	100	1270.30	54.86	1255.02	58.67	1.218
6	100	642.55	54.24	634.91	57.69	1.203
Avg. sol.		755.38		747.68		

4.4 Numerical Study

The proposed algorithm is coded in Matlab 7.8.0 and executed on a 3.00 Ghz Pentium 4 computer. In order to test the performance of the algorithm for the VRPSPD, two sets of benchmark instances from the literature are used. The first data set consisting of 40 problems is created in Dethloff (2001) where the problem instances with 50 customers are classified according to two different geographical scenarios and labeled as either SCA or CON. In SCA instances, the coordinate values

of the customers are uniformly distributed within [0,100]. In CON data sets, half of the customers are scattered in a similar way with SCA while the coordinates of the other half are uniformly distributed in [100/3,200/3]. Thus, the scenario CON represents urban areas where big portion of the population are located in small regions. The delivery amount of customer i (d_i) is distributed in [0,100]. The pickup amount of each customer is determined by the equation $p_i=(0.5+r_i).d_i$, where r_i is a random number uniformly distributed over the interval [0,1]. The vehicle capacities are generated from the equation of $C=\sum_{s\in J} d_s/\mu$ where μ represents the minimum number of vehicles required. 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 (Dethloff, 2001).

The second data set is generated by Salhi & Nagy (1999). This data set consists of 14 instances involving from 50 to 199 customers. The instances are adapted from seven classical vehicle routing problems presented by Christofides, Mingozzi & Toth (1979). For each VRP instance, Salhi & Nagy (1999) generated a VRPSPD problem by splitting the original demand between delivery and pickup loads (Çatay, 2010). The instances obtained by this procedure are classified as X problems. The other seven instances which are called as Y problems are generated by exchanging the delivery and pickup load of every customer.

In addition to the benchmark instances of the VRPSPD, we test the performance of the algorithm on the VRPMPD instances generated by Wassan, Wassan & Nagy (2008). The data set consisting of 120 problems is based on the VRPSPD instances of Dethloff (2001). For each VRPSPD instance, every 10th, 4th and 2nd customer is set to be a pickup customer by discarding the delivery load of the selected customer and the pickup loads of the other ones. The new instances are labeled as T, Q and H, respectively. The euclidean distances between customers are used in all data sets to construct the cost network.

4.4.1 Computational Results

The proposed SALS-VND is run ten times for each benchmark problem. A statistical analysis for each data set is also conducted by applying a paired-t test at significance level of α=0.05 to reveal whether there exists significant differences between SALS-VND and other methods in terms of solution quality or not. Thus, we set up the null hypothesis as $H_0=\mu_{SALS-VND} - \mu_{CA}=0$ and a two-sided alternative hypothesis as $H_1=\mu_{SALS-VND} - \mu_{CA} \neq 0$. The symbols $\mu_{SALS-VND}$and μ_{CA} represent the population mean for SALS-VND and the compared algorithm, respectively.

We terminate the first stage of the algorithm after 1000 successive iterations without any improvement. The second stage is terminated after 15 consecutive VND implementations yielding no improved solution. We compare our algorithm with the following best-known algorithms available in the literature:

- WWN: Reactive tabu search (Wassan, Wassan & Nagy, 2008)
- WNA: Reactive tabu search (Wassan, Nagy & Ahmadi, 2008)
- R&P: Large neighborhood search (Ropke & Pissinger, 2006)
- A&K: Particle swarm optimization (Ai & Kachitvichyanukul, 2009)
- G&A: Ant colony system (Gajpal & Abad, 2009a)
- SDBOF: Parallel iterated local search (Subramanian et al., 2010)
- ZTK: Adaptive memory methodology (Zachariadis, Tarantilis & Kiranoudis, 2010)
- GKA: Hybrid discrete particle swarm optimization (Goksal, Karaoglan & Altiparmak, 2013)

Table 4.3 shows the computational results of the proposed algorithm for the instances proposed by Dethloff (2001). For SALS-VND, we report the best solution and the average solution observed from each instance. For other algorithms, only the best found solutions are reported because the average results are not reported in all studies. Additionally, the number of best-known solutions (BKS) found and the average deviations from BKS (Avg. dev.) are reported in last two rows. Bold numbers indicate that the algorithm has reached the best solution. As seen from the

table, SALS-VND, GKA, SDBOF and ZTK produce best solutions for all instances while G&A and R&P obtain 38 and 22 best solutions, respectively. Results of the paired-t test for the corresponding data set are presented in Table 4.4. As seen from the table, while SALS-VND is statistically significant different from R&P, there is no statistically significant difference between SALS-VND and G&A. The results of ZTK, SDBOF and GKA are not taken into account since they yield the same solutions with SALS-VND for all instances.

The computational results obtained from Salhi & Nagy's (1999) benchmark instances are reported in Table 4.5. Bold numbers for each algorithm signifies that the algorithm has reached the best-known solution as in Table 4.3. As seen from the table, SALS-VND comes as the second best algorithm together with GKA by achieving the six best solutions in the literature out of 14 problem instances. Whereas, WWN and ZTK reach the five, and G&A, R&P and A&K reach the two best solutions, respectively. Regarding the deviation of reached solution from the best one for each instance, the proposed SALS-VND produces high-quality solutions except for the instances CMT11X and CMT11Y. Without these instances, the average deviation of the results of SALS-VND from the best solutions is 0.0084. Although the deviation increases to 0.0145 with these instances, it is still smaller than that of R&P and A&K. Moreover, the performed paired-t test for the data set indicates that SALS is not statistically significant different from WWN, G&A, ZTK, GKA and SDBOF with the p-value of 0.906, 0.242, 0.150, 0.174 and 0.109, respectively (see Table 4.6). This outcome shows that our proposed algorithm works as good as the most sophisticated methods proposed for the VRPSPD.

Table 4.3 Computational results for the VRPSPD benchmark instances of Dethloff (2001)

Instance	n	R&P	G&A	ZTK	SDBOF	GKA	SALS-VND		
		Best sol.	Best sol.	Best sol.	Best sol.	Best sol.	Best sol.	Avg. sol.	Dev. of best sol.
SCA3-0	50	636.1	**635.62**	**635.62**	**635.62**	**635.62**	**635.62**	638.09	0.0000
SCA3-1	50	**697.8**	**697.84**	**697.84**	**697.84**	**697.84**	**697.84**	697.84	0.0000
SCA3-2	50	**659.3**	**659.34**	**659.34**	**659.34**	**659.34**	**659.34**	661.44	0.0000
SCA3-3	50	680.6	**680.04**	**680.04**	**680.04**	**680.04**	**680.04**	681.39	0.0000
SCA3-4	50	**690.5**	**690.50**	**690.50**	**690.50**	**690.50**	**690.50**	692.91	0.0000
SCA3-5	50	**659.9**	**659.90**	**659.90**	**659.90**	**659.90**	**659.90**	660.14	0.0000
SCA3-6	50	**651.1**	**651.09**	**651.09**	**651.09**	**651.09**	**651.09**	652.97	0.0000
SCA3-7	50	666.1	**659.17**	**659.17**	**659.17**	**659.17**	**659.17**	666.87	0.0000
SCA3-8	50	**719.5**	**719.47**	**719.47**	**719.47**	**719.47**	**719.47**	722.68	0.0000
SCA3-9	50	**681.0**	**681.00**	**681.00**	**681.00**	**681.00**	**681.00**	684.17	0.0000
SCA8-0	50	975.1	**961.50**	**961.50**	**961.50**	**961.50**	**961.50**	965.86	0.0000
SCA8-1	50	1052.4	**1049.65**	**1049.65**	**1049.65**	**1049.65**	**1049.65**	1050.74	0.0000
SCA8-2	50	1044.5	1042.69	**1039.64**	**1039.64**	**1039.64**	**1039.64**	1043.32	0.0000
SCA8-3	50	999.1	**983.34**	**983.34**	**983.34**	**983.34**	**983.34**	988.39	0.0000
SCA8-4	50	**1065.5**	**1065.49**	**1065.49**	**1065.49**	**1065.49**	**1065.49**	1069.80	0.0000
SCA8-5	50	**1027.1**	**1027.08**	**1027.08**	**1027.08**	**1027.08**	**1027.08**	1029.59	0.0000
SCA8-6	50	977.0	**971.82**	**971.82**	**971.82**	**971.82**	**971.82**	975.06	0.0000
SCA8-7	50	1061.0	1052.17	**1051.28**	**1051.28**	**1051.28**	**1051.28**	1056.10	0.0000
SCA8-8	50	**1071.2**	**1071.18**	**1071.18**	**1071.18**	**1071.18**	**1071.18**	1075.62	0.0000
SCA8-9	50	**1060.5**	**1060.50**	**1060.50**	**1060.50**	**1060.50**	**1060.50**	1062.92	0.0000
CON3-0	50	**616.5**	**616.52**	**616.52**	**616.52**	**616.52**	**616.52**	619.10	0.0000
CON3-1	50	**554.5**	**554.47**	**554.47**	**554.47**	**554.47**	**554.47**	556.11	0.0000
CON3-2	50	521.4	**518.00**	**518.00**	**518.00**	**518.00**	**518.00**	520.47	0.0000
CON3-3	50	**591.2**	**591.19**	**591.19**	**591.19**	**591.19**	**591.19**	591.54	0.0000
CON3-4	50	**588.8**	**588.79**	**588.79**	**588.79**	**588.79**	**588.79**	591.02	0.0000
CON3-5	50	**563.7**	**563.70**	**563.70**	**563.70**	**563.70**	**563.70**	566.01	0.0000
CON3-6	50	500.8	**499.05**	**499.05**	**499.05**	**499.05**	**499.05**	504.78	0.0000
CON3-7	50	**576.5**	**576.48**	**576.48**	**576.48**	**576.48**	**576.48**	578.37	0.0000
CON3-8	50	**523.1**	**523.05**	**523.05**	**523.05**	**523.05**	**523.05**	523.05	0.0000
CON3-9	50	586.4	**578.25**	**578.25**	**578.25**	**578.25**	**578.25**	579.52	0.0000
CON8-0	50	**857.2**	**857.17**	**857.17**	**857.17**	**857.17**	**857.17**	857.78	0.0000
CON8-1	50	**740.9**	**740.85**	**740.85**	**740.85**	**740.85**	**740.85**	744.27	0.0000
CON8-2	50	716.0	**712.89**	**712.89**	**712.89**	**712.89**	**712.89**	714.15	0.0000
CON8-3	50	**811.1**	**811.07**	**811.07**	**811.07**	**811.07**	**811.07**	811.91	0.0000
CON8-4	50	**772.3**	**772.25**	**772.25**	**772.25**	**772.25**	**772.25**	776.36	0.0000
CON8-5	50	755.7	**754.88**	**754.88**	**754.88**	**754.88**	**754.88**	755.29	0.0000
CON8-6	50	693.1	**678.92**	**678.92**	**678.92**	**678.92**	**678.92**	684.11	0.0000
CON8-7	50	814.8	**811.96**	**811.96**	**811.96**	**811.96**	**811.96**	812.46	0.0000
CON8-8	50	774.0	**767.53**	**767.53**	**767.53**	**767.53**	**767.53**	767.53	0.0000
CON8-9	50	809.3	**809.00**	**809.00**	**809.00**	**809.00**	**809.00**	809.44	0.0000
BKS f.*		22	38	40	40	40	40		
Avg. dev.		0.0032	0.0001	0.0000	0.0000	0.0000	0.0000		

* Number of best-known solutions found

Table 4.4 Results of paired-t test for the VRPSPD benchmark instances of Dethloff (2001)

Pairs (SALS-VND - Compared algorithm)	Mean Difference	p-value
SALS-VND – G&A	-0.099	0.219
SALS-VND – R&P	-2.528	0.001

Table 4.5 Computational results for the VRSPD benchmark instances of Salhi & Nagy (1999)

Instance	n	WWN	R&P	A&K	G&A	ZTK	SDBOF	GKA	SALS-VND		
		Best sol.	Best sol.	Best sol.	Best sol.	Best sol.	Best sol.	Best sol.	Best sol.	Avg. sol.	Dev. of best sol.
CMT1X	50	468.30	**467**	**467**	**466.77**	469.80	**466.77**	**466.77**	**466.77**	468.31	0.0000
CMT1Y	50	-	**467**	**467**	**466.77**	469.80	**466.77**	**466.77**	**466.77**	468.40	0.0000
CMT2X	75	**668.77**	704	710	684.21	684.21	684.21	684.21	684.89	686.31	0.0241
CMT2Y	75	**663.25**	685	710	684.94	684.21	684.21	684.21	684.89	686.24	0.0326
CMT3X	100	729.63	731	738	721.40	**721.27**	**721.27**	**721.27**	**721.27**	722.24	0.0000
CMT3Y	100	745.46	748	740	721.40	**721.27**	**721.27**	**721.27**	**721.27**	722.36	0.0000
CMT4X	150	876.50	879	912	854.12	**852.46**	**852.46**	852.83	**852.46**	854.68	0.0000
CMT4Y	150	870.44	876	913	855.76	**852.46**	**852.46**	**852.46**	**852.46**	856.73	0.0001
CMT5X	199	1044.51	1108	1167	1034.87	1030.55	**1029.25**	1033.50	1032.89	1038.69	0.0035
CMT5Y	199	1054.46	1146	1142	1037.34	1030.55	**1029.25**	1036.00	1032.89	1038.76	0.0035
CMT11X	120	861.97	837	895	839.66	**833.92**	**833.92**	**833.92**	874.85	877.15	0.0491
CMT11Y	120	**830.39**	920	900	840.19	833.92	833.92	833.92	874.85	878.02	0.0535
CMT12X	100	**644.70**	685	691	663.01	662.22	662.22	662.94	663.94	669.79	0.0298
CMT12Y	100	**659.52**	675	697	663.50	662.22	662.22	663.50	663.69	671.67	0.0067
BKS f.		5	2	2	2	5	9	6	6		
Avg. dev.		0.0131	0.0417	0.0609	0.0092	0.0076	0.0064	0.0072	0.0145		

Table 4.6 Results of paired-t test for the VRSPD benchmark instances of Salhi & Nagy (1999)

Pairs (SALS-VND - Compared algorithm)	Mean Difference	p-value
SALS-VND - GKA	5.737	0.174
SALS-VND - SDBOF	6.692	0.109
SALS-VND - ZTK	6.074	0.150
SALS-VND - G&A	4.282	0.242
SALS-VND - R&P	-23.865	0.028
SALS-VND - A&K	-39.651	0.002
SALS-VND - WWN	0.709	0.906

The computational results of VRPMPD benchmark instances are given in Table 4.7, 4.8 and 4.9. For the problems labeled as T, SALS-VND and GKA has reached 33 best solutions over 40 instances while WNA generates only 9 best solutions for the instances. The results of Q instances are showed in Table 4.8. As seen in the table, SALS-VND has obtained 36 best solutions out of 40 instances. GKA and WNA have reached 36 and 7 best solutions, respectively. The results of H instances are reported in Table 4.9. As seen from the table, SALS-VND is the winner algorithm in terms of the solution quality, producing 39 out of 40 best solutions while improving the best known solution of the instance SCA8-8H. Following, GKA and WNA obtains 38 and 2 best solutions out of 40 instances, respectively. The results of the conducted paired-t test for VRPMPD instances are showed in Table 4.10. As seen from the table, there is no statistically significant difference between SALS-VND and GKA since the p value of 0.319. However, the algorithm is statistically significant different from WNA with the mean difference of -11.983, indicating that SALS-VND outperforms WNA.

Table 4.7 Computational results for the VRPMPD instances labeled T of Dethloff (2001)

Instance	n	WNA	GKA	SALS-VND		
		Best sol.	Best sol.	Best sol.	Avg. sol.	Dev. of best sol.
SCA3-0T	50	627.92	**604.22**	**604.22**	611.62	0.0000
SCA3-1T	50	662.68	**657.99**	**657.99**	657.99	0.0000
SCA3-2T	50	636.57	**634.70**	**634.70**	635.41	0.0000
SCA3-3T	50	684.85	**651.86**	**651.86**	652.34	0.0000
SCA3-4T	50	666.18	**648.82**	**648.82**	649.61	0.0000
SCA3-5T	50	647.49	**619.31**	**619.31**	619.41	0.0000
SCA3-6T	50	634.83	**624.10**	**624.10**	624.68	0.0000
SCA3-7T	50	647.23	**631.41**	**631.41**	632.92	0.0000
SCA3-8T	50	700.52	**691.13**	**691.13**	691.77	0.0000
SCA3-9T	50	650.50	**642.86**	**642.86**	647.40	0.0000
SCA8-0T	50	911.61	**889.40**	**889.40**	889.55	0.0000
SCA8-1T	50	**947.00**	947.01	947.01	947.74	0.0000
SCA8-2T	50	**897.15**	897.16	897.16	897.29	0.0000
SCA8-3T	50	922.95	**922.96**	**922.96**	926.97	0.0000
SCA8-4T	50	991.49	**987.25**	**987.25**	987.27	0.0000
SCA8-5T	50	**957.44**	**957.44**	**957.44**	957.63	0.0000
SCA8-6T	50	**916.02**	916.03	916.03	917.09	0.0000
SCA8-7T	50	**931.48**	**931.48**	**931.48**	933.42	0.0000
SCA8-8T	50	1005.01	**1001.19**	**1001.19**	1002.15	0.0000
SCA8-9T	50	951.14	**938.75**	**938.75**	944.28	0.0000
CON3-0T	50	617.79	**601.85**	**601.85**	602.77	0.0000
CON3-1T	50	**538.66**	**538.66**	**538.66**	540.63	0.0000
CON3-2T	50	498.98	**498.15**	**498.15**	498.81	0.0000
CON3-3T	50	629.77	**569.07**	**569.07**	569.91	0.0000
CON3-4T	50	603.95	**572.52**	**572.52**	574.31	0.0000
CON3-5T	50	555.71	**549.49**	**549.49**	549.68	0.0000
CON3-6T	50	494.11	**489.54**	**489.54**	489.95	0.0000
CON3-7T	50	573.60	**553.95**	**553.95**	554.51	0.0000
CON3-8T	50	514.78	**509.16**	**509.16**	509.71	0.0000
CON3-9T	50	573.52	**570.38**	**570.38**	571.30	0.0000
CON8-0T	50	809.60	**802.83**	**802.83**	804.13	0.0000
CON8-1T	50	702.04	**682.83**	**682.83**	689.24	0.0000
CON8-2T	50	655.97	**653.29**	**653.29**	654.03	0.0000
CON8-3T	50	**740.61**	740.62	740.62	741.24	0.0000
CON8-4T	50	726.06	**720.59**	**720.59**	723.21	0.0000
CON8-5T	50	**717.21**	717.22	717.22	718.95	0.0000
CON8-6T	50	651.39	**644.75**	**644.75**	646.75	0.0000
CON8-7T	50	**755.97**	**755.97**	**755.97**	756.59	0.0000
CON8-8T	50	733.04	**714.17**	**714.17**	714.17	0.0000
CON8-9T	50	763.97	**760.96**	**760.96**	762.20	0.0000
BKS f.*		9	35	35		
Avg. dev.		0.0159	0.0000	0.0000		

* Number of best-known solutions found

Table 4.8 Computational results for the VRPMPD instances labeled Q of Dethloff (2001)

Instance	n	WNA	GKA	SALS-VND		
		Best sol.	Best sol.	Best sol.	Avg. sol.	Dev. of best sol.
SCA3-0Q	50	608.70	**591.84**	**591.84**	593.36	0.0000
SCA3-1Q	50	678.30	**652.32**	**652.32**	653.87	0.0000
SCA3-2Q	50	635.45	**622.10**	**622.10**	623.59	0.0000
SCA3-3Q	50	678.36	**644.85**	**644.85**	644.85	0.0000
SCA3-4Q	50	667.71	**642.04**	**642.04**	642.04	0.0000
SCA3-5Q	50	618.01	**617.42**	**617.42**	617.66	0.0000
SCA3-6Q	50	628.20	**601.28**	**601.28**	601.28	0.0000
SCA3-7Q	50	623.78	**616.84**	**616.84**	620.47	0.0000
SCA3-8Q	50	690.50	**676.85**	**676.85**	678.63	0.0000
SCA3-9Q	50	630.10	**628.31**	**628.31**	629.48	0.0000
SCA8-0Q	50	**804.92**	804.93	804.93	806.39	0.0000
SCA8-1Q	50	924.52	**922.00**	**922.00**	922.75	0.0000
SCA8-2Q	50	**830.43**	830.44	830.44	831.64	0.0000
SCA8-3Q	50	852.47	**849.00**	**849.00**	851.81	0.0000
SCA8-4Q	50	909.57	**906.29**	**906.29**	907.98	0.0000
SCA8-5Q	50	**880.58**	**880.58**	**880.58**	881.80	0.0000
SCA8-6Q	50	820.87	**804.21**	**804.21**	813.17	0.0000
SCA8-7Q	50	857.97	**855.24**	**855.24**	858.12	0.0000
SCA8-8Q	50	**927.09**	927.10	927.10	929.96	0.0000
SCA8-9Q	50	845.21	**841.97**	**841.97**	844.30	0.0000
CON3-0Q	50	622.66	**595.54**	**595.54**	595.72	0.0000
CON3-1Q	50	536.05	**529.05**	**529.05**	529.45	0.0000
CON3-2Q	50	499.40	**493.09**	**493.09**	497.59	0.0000
CON3-3Q	50	567.49	**557.18**	**557.18**	559.22	0.0000
CON3-4Q	50	580.89	**571.57**	**571.57**	571.67	0.0000
CON3-5Q	50	562.89	**545.90**	**545.90**	547.31	0.0000
CON3-6Q	50	486.89	**482.69**	**482.69**	483.38	0.0000
CON3-7Q	50	559.99	**541.61**	**541.61**	541.61	0.0000
CON3-8Q	50	508.61	**492.14**	**492.14**	493.79	0.0000
CON3-9Q	50	**563.88**	563.89	563.89	564.12	0.0000
CON8-0Q	50	755.44	**754.50**	**754.50**	756.02	0.0000
CON8-1Q	50	652.47	**648.30**	**648.30**	650.81	0.0000
CON8-2Q	50	596.33	**595.64**	**595.64**	596.57	0.0000
CON8-3Q	50	706.86	**685.99**	**685.99**	687.42	0.0000
CON8-4Q	50	**705.37**	**705.37**	**705.37**	705.95	0.0000
CON8-5Q	50	671.26	**666.17**	**666.17**	668.06	0.0000
CON8-6Q	50	585.43	**584.72**	**584.72**	589.03	0.0000
CON8-7Q	50	**724.57**	**724.57**	**724.57**	726.73	0.0000
CON8-8Q	50	652.76	**648.54**	**648.54**	648.54	0.0000
CON8-9Q	50	719.58	**714.64**	**714.64**	716.61	0.0000
BKS f.*		7	36	36		
Avg. dev.		0.0144	0.0000	0.0000		

* Number of best-known solutions found

Table 4.9 Computational results for the VRPMPD instances labeled H of Dethloff (2001)

Instance	n	WNA	GKA	SALS-VND		
		Best sol.	Best sol.	Best sol.	Avg. sol.	Dev. of best sol.
SCA3-0H	50	598.27	**567.20**	**567.20**	570.58	0.0000
SCA3-1H	50	624.86	**610.66**	**610.66**	611.52	0.0000
SCA3-2H	50	611.78	**584.52**	**584.52**	586.68	0.0000
SCA3-3H	50	650.03	**608.15**	**608.15**	612.70	0.0000
SCA3-4H	50	624.77	**597.35**	**597.35**	599.15	0.0000
SCA3-5H	50	602.57	**585.09**	**585.09**	589.22	0.0000
SCA3-6H	50	605.00	**574.25**	**574.25**	574.25	0.0000
SCA3-7H	50	579.64	**575.99**	**575.99**	576.44	0.0000
SCA3-8H	50	676.44	**635.84**	**635.84**	635.84	0.0000
SCA3-9H	50	645.69	**600.92**	**600.92**	602.49	0.0000
SCA8-0H	50	724.51	**714.86**	**714.86**	720.23	0.0000
SCA8-1H	50	787.66	**780.77**	**780.77**	781.76	0.0000
SCA8-2H	50	**739.93**	740.55	740.55	741.11	0.0008
SCA8-3H	50	764.52	**761.83**	**761.83**	762.97	0.0000
SCA8-4H	50	819.39	**794.69**	**794.69**	795.93	0.0000
SCA8-5H	50	788.42	**783.04**	**783.04**	785.37	0.0000
SCA8-6H	50	731.41	**723.21**	**723.21**	723.21	0.0000
SCA8-7H	50	735.98	**725.79**	**725.79**	725.79	0.0000
SCA8-8H	50	856.20	850.91	**849.20**	850.95	-0.0020
SCA8-9H	50	803.18	**771.96**	**771.96**	773.88	0.0000
CON3-0H	50	604.65	**579.67**	**579.67**	579.96	0.0000
CON3-1H	50	537.88	**513.46**	**513.46**	514.74	0.0000
CON3-2H	50	502.23	**482.74**	**482.74**	486.33	0.0000
CON3-3H	50	**549.68**	**549.68**	**549.68**	552.03	0.0000
CON3-4H	50	568.28	**551.92**	**551.92**	553.65	0.0000
CON3-5H	50	534.22	**529.93**	**529.93**	531.60	0.0000
CON3-6H	50	480.00	**466.61**	**466.61**	468.01	0.0000
CON3-7H	50	551.45	**527.98**	**527.98**	530.27	0.0000
CON3-8H	50	500.77	**475.77**	**475.77**	478.18	0.0000
CON3-9H	50	559.57	**536.89**	**536.89**	538.05	0.0000
CON8-0H	50	716.41	**697.34**	**697.34**	702.72	0.0000
CON8-1H	50	611.86	**610.23**	**610.23**	610.95	0.0000
CON8-2H	50	601.73	**590.43**	**590.43**	592.76	0.0000
CON8-3H	50	663.75	**653.81**	**653.81**	655.83	0.0000
CON8-4H	50	659.69	**641.59**	**641.59**	641.59	0.0000
CON8-5H	50	623.46	**608.59**	**608.59**	610.50	0.0000
CON8-6H	50	557.52	**546.06**	**546.06**	550.34	0.0000
CON8-7H	50	682.53	**652.70**	**652.70**	660.60	0.0000
CON8-8H	50	591.18	**595.60**	**595.60**	597.35	0.0000
CON8-9H	50	631.23	**624.03**	**624.03**	626.23	0.0000
BKS f.*		2	38	39		
Avg. dev.		0.0282	0.0001	0.0000		

* Number of best-known solutions found

Table 4.10 Results of paired-t test for the VRPMPD instances of Dethloff (2001)

Pairs (SALS-VND - Compared algorithm)	Mean Difference	p-value
SALS-VND - GKA	-0.014	0.319
SALS-VND - WNA	-11.983	0.000

4.4.2 An Illustrative Example

An illustrative example consisting of 50 customers is presented and solved in this section. These customers require simultaneous pickup and delivery services and served by a homogeneous fleet of vehicles located at a central depot. The vehicle capacity is 700 units. The coordinates of the nodes and the pickup (p_i) and delivery (d_i) demands of customers are given in Table 4.11 and Table 4.12. The node 0 represents the central depot.

The problem is solved ten times by using the proposed algorithm. The solution obtained from the first stage of the algorithm is illustrated in Figure 4.18. The total distance travelled by the vehicles is 1689.28. The vehicle routes of this solution are found as follows:

- Route 1: {0 14 38 4 49 45 33 26 12 8 15 28 6 0}
- Route 2: {0 11 10 7 0}
- Route 3: {0 27 46 40 19 34 5 25 22 29 47 21 0}
- Route 4: {0 43 48 39 50 42 44 36 41 30 18 20 0}
- Route 5: {0 3 1 31 35 16 24 23 17 32 37 13 9 2 0}

In addition to the solution obtained from the first stage of the algorithm, the solution obtained from the second stage is illustrated in Figure 4.19. The total distance travelled by the vehicles is 1679.44. Five vehicle routes which serve 50 customers are determined as follows:

- Route 1: {0 21 47 40 19 34 5 25 22 29 3 2 0}
- Route 2: {0 7 10 0}
- Route 3: {0 14 38 4 49 45 33 26 12 8 15 28 6 0}
- Route 4: {0 11 9 13 37 32 17 24 16 35 31 1 3 0}
- Route 5: {0 43 48 27 46 39 50 42 44 36 41 30 18 20 0}

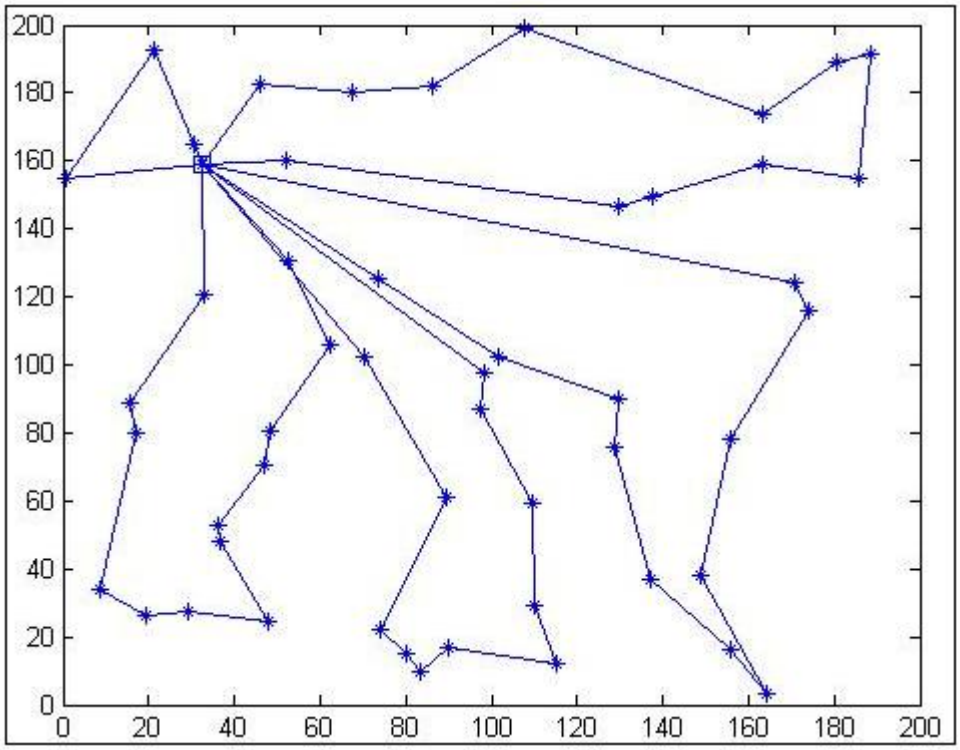


Figure 4.18 Illustration of the vehicle routes obtained by hybrid_SALS

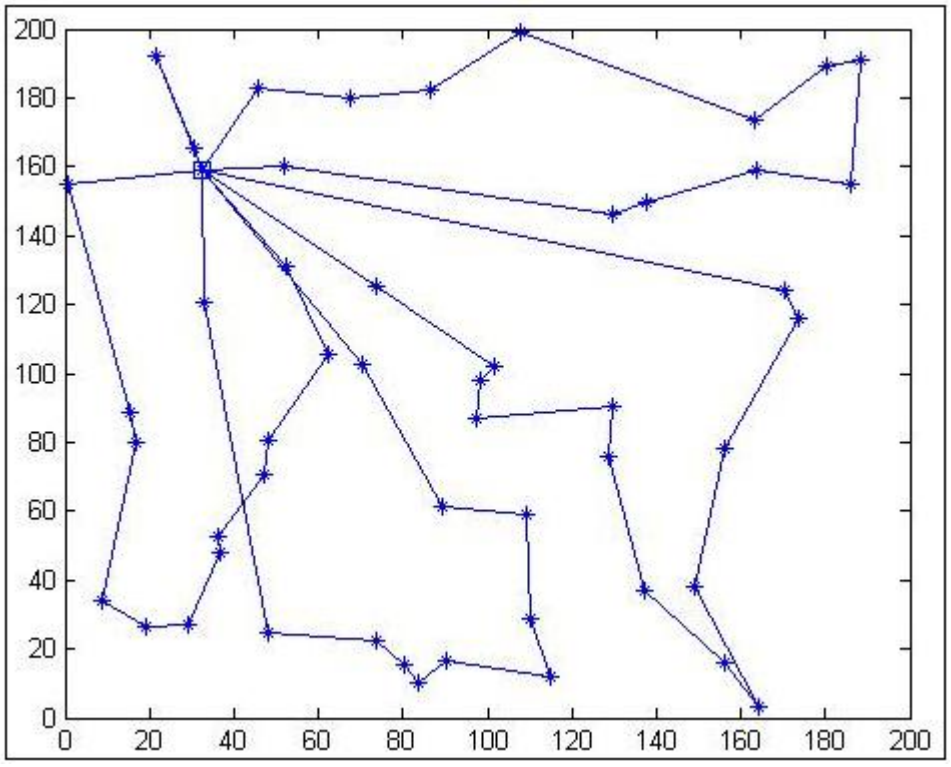


Figure 4.19 Illustration of the vehicle routes obtained by SALS-VND

Table 4.11 Coordinates, pickup and delivery demands from node 0 to node 25

node	d_i	p_i	x	y
0	0	0	32.4365	158.8569
1	81.4724	90.5792	62.243	105.7066
2	12.6987	91.3376	33.1297	120.3964
3	63.2359	9.754	52.5943	130.8158
4	27.8498	54.6882	137.8429	149.6303
5	95.7507	96.4889	90.1083	16.7643
6	15.7613	97.0593	45.7954	182.6675
7	95.7167	48.5376	30.4756	165.1634
8	80.028	14.1886	107.6685	199.2269
9	42.1761	91.5736	15.6351	88.5357
10	79.2207	95.9492	21.3306	192.3796
11	65.5741	3.5712	0.9268	154.9821
12	84.9129	93.3993	163.4606	173.7389
13	67.8735	75.774	16.8872	79.9565
14	74.3132	39.2227	51.9741	160.0137
15	65.5478	17.1187	86.2828	182.1295
16	70.6046	3.1833	36.3694	52.7606
17	27.6923	4.6171	29.1078	27.2137
18	9.7132	82.3458	173.8584	115.9409
19	69.4829	31.7099	109.972	28.991
20	95.0222	3.4446	170.6062	124.411
21	43.8744	38.1558	70.1905	102.6499
22	76.5517	79.52	80.3616	15.1933
23	18.6873	48.9764	47.9832	24.6638
24	44.5586	64.6313	36.7816	47.9905
25	70.9365	75.4687	83.4534	9.9309

Table 4.12 Coordinates , pickup and delivery demands from node 26 to node 50

node	d_i	p_i	x	y
26	27.6025	67.9703	180.5432	188.9574
27	65.5098	16.2612	98.1728	97.8505
28	11.8998	49.8364	67.5439	180.0108
29	95.9744	34.0386	73.8494	22.2406
30	58.5268	22.3812	156.0504	77.9478
31	75.1267	25.5095	48.3383	80.7824
32	50.5957	69.9077	19.2909	26.3947
33	89.0903	95.9291	188.4101	191.2269
34	54.7216	13.8624	115.0417	11.9559
35	14.9294	25.7508	46.956	70.6317
36	84.0717	25.4282	164.2388	3.0807
37	81.4285	24.3525	8.6048	33.798
38	92.9264	34.9984	129.8231	146.3445
39	19.6595	25.1084	129.5492	90.1847
40	61.6045	47.3289	109.4018	59.2642
41	35.166	83.0829	148.9386	37.791
42	58.5264	54.9724	137.3551	36.7022
43	91.7194	28.5839	73.6969	125.1237
44	75.72	75.3729	156.0455	16.2252
45	38.0446	56.7822	185.8772	155.1425
46	7.5854	5.395	97.3583	87.1717
47	53.0798	77.9167	89.3567	61.2699
48	93.4011	12.9906	101.7017	102.1543
49	56.8824	46.9391	163.5255	158.9663
50	1.1902	33.7123	128.8636	75.7219

CHAPTER FIVE
CONCLUSION

In this thesis, we deal with the two variants of the classical Vehicle Routing Problem (VRP) arising mainly in reverse logistic systems that include both distribution and collection operations. These problems are the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD), and its different version, the Vehicle Routing Problem with Mixed Pickup and Delivery (VRPMPD), where the customers have either pickup or delivery demand. The related problems have gained increased importance in recent years since the integration of pickup and delivery operations has become a vital task for companies to minimize costs.

In order to solve these problems, we have developed an adaptive local search algorithm (SALS-VND) that is constructed by hybridizing a SA-inspired algorithm with Variable Neighborhood Descent (VND). The applied SA-inspired algorithm, Self-Adaptive Local Search (SALS), is a parameter-free metaheuristic whose non-monotonic threshold function provides the required diversification to the search. On the other hand, the VND algorithm, which is a well-known local search method, has been implemented to intensify the search process. These procedures comprise the first stage of SALS-VND. The second stage of the algorithm consists of an extra optimization procedure that is performed by implementing the VND algorithm with a proposed perturbation mechanism. Since SALS-VND has been developed using parameter-free algorithms, it does not need any parameter setting, which consequently makes its implementation to the optimization problems much easier.

The performance of SALS-VND is tested by using well-known benchmark instances for the VRPSPD and VRPMPD. The computational results indicate that the SALS-VND generates high-quality solutions for the instances and perform as well as the most sophisticated methods proposed for the VRPSPD and the VRPMPD up to date. Moreover, considering its simplicity, SALS-VND has the advantage of being a parameter-free algorithm in comparison the existing algorithms.

Since the problems related with the operation of distribution and collection systems may occur in different settings, a variant number of constraints need to be included to these problems. For example, the customers may demand to be serviced in certain time periods. These requirements add time windows constraints to the original pickup and delivery problems. Another constraint may require the use of heterogeneous fleet. The problems we deal with have homogeneous fleets. However, in many transportation systems, generally different types of vehicles perform the same operations. These types of problems can be modeled as heterogeneous fleet vehicle routing problems with pickups and deliveries. Another constraint may be related with the number of depots the system involves. The transportation system may have multiple depots. Considering the related literature, there are only a few papers about the problems with these types of specifications. Our future research will be shaped based on these problems.

REFERENCES

Ai, T. J., & Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research, 36*(5), 1693-1702.

Alabas-Uslu, C., & Dengiz, B. (2011). A self-adaptive local search algorithm for the classical vehicle routing problem. *Expert Systems with Applications, 38*(7), 8990-8998.

Altiparmak, F., & Karaoglan, I. (2008). An adaptive tabu-simulated annealing for concave cost transportation problems. *Journal of the Operational Research Society, 59*(3), 331-341.

Anily, S. (1996). The vehicle-routing problem with delivery and back-haul options. *Naval Research Logistics, 43*(3), 415-434.

Azizi, N., & Zolfaghari, S. (2004). Adaptive temperature control for simulated annealing: a comparative study. *Computers & Operations Research, 31*(14), 2439-2451.

Baldacci, R., Mingozzi, A., & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research, 218*(1), 1-6.

Barnhart, C., & Laporte, G. (Eds.). (2007). *Handbooks in Operations Research & Management Science: Transportation: Transportation*, 14. Amsterdam: Elsevier.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top, 15*(1), 1-31.

Bianchessi, N., & Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, *34*(2), 578-594.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, *35*(3), 268-308.

Brandao, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, *173*(2), 540-555.

Casco, D. O., Golden, B. L., Wasil, E. A. (1988). Vehicle routing with backhauls: Models, algorithms, and case studies. In: B. L. Golden, & A. A. Assad (Eds.), *Vehicle Routing: Methods and studies*, 127-147. Amsterdam: Elsevier.

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41-51.

Chen, J. F., & Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, *57*(5), 579-587.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & L. Sandi (Eds.), *Combinatorial Optimization*, 315-338. Chichester: Wiley.

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In A. Langevin, & D. Riopel (Eds.), *Logistic Systems: Design and Optimization*, 279-297. US: Springer.

Crispim, J., & Brandão, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society*, *56*(11), 1296-1302.

Çatay, B. (2010). A new saving-based ant algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Expert Systems with Applications*, *37*(10), 6809-6817.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80-91.

Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, *40*(2), 235-247.

Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR-Spektrum*, *23*(1), 79-96.

Dethloff, J. (2002). Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *Journal of the Operational research society*, *53*(1), 115-118.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *26*(1), 29-41.

Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, *2*(3), 1461.

Gajpal, Y., & Abad, P. (2009a). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, *36*(12), 3215-3223.

Gajpal, Y., & Abad, P. L. (2009b). Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research, 196*(1), 102-117.

Gajpal, Y., & Abad, P. (2010). Saving-based algorithms for vehicle routing problem with simultaneous pickup and delivery. *Journal of the Operational Research Society*, *61*(10), 1498-1509.

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, *20*(4), 74-94.

Goetschalckx, M., & Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research, 42*(1), 39-51.

Goksal, F. P., Karaoglan, I., & Altiparmak, F. (2013). A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, *65*(1), 39-53.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95-99.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Michigan: The University of Michigan Press.

Hu, T. C., Kahng, A. B., & Tsao, C. W. A. (1995). Old bachelor acceptance: A new class of non-monotone threshold accepting methods. *ORSA Journal on Computing, 7*(4), 417-425.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. *In Proceedings of IEEE International Conference on Neural Networks*, New Jersey: Piscataway, 1942–1948.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simmulated annealing. *Science, 220*(4598), 671-680.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research, 34*(9), 2743-2757.

Li, F., Golden, B., & Wasil, E. (2005). Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research, 32*(5), 1165-1179.

Lin, C. K. Y., Haley, K. B., & Sparks, C. (1995). A comparative study of both standard and adaptive versions of threshold accepting and simulated annealing algorithms in three scheduling problems. *European Journal of Operational Research, 83*(2), 330-346.

Mester, D., & Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research, 34*(10), 2964-2975.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General, 23*(5), 377-386.

Mingozzi, A., Giorgi, S., & Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science, 33*(3), 315-329.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097-1100.

Montané, F. A. T., & Galvao, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research, 33*(3), 595-619.

Mosheiov, G. (1998). Vehicle routing with pick-up and delivery: tour-partitioning heuristics. *Computers & Industrial Engineering*, *34*(3), 669-684.

Nagy, G., & Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, *162*(1), 126-141.

Osman, I. H., & Wassan, N. A. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, *5*(4), 263-285.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, *58*(1), 21-51.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations research*, *34*(8), 2403-2435.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, *31*(12), 1985-2002.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-Ants: savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, *31*(4), 563-591.

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, *171*(3), 750-775.

Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 1034-1042.

Subramanian, A., Drummond, L. M. D. A., Bentes, C., Ochi, L. S., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, *37*(11), 1899-1911.

Subramanian, A., Uchoa, E., Pessoa, A. A., & Ochi, L. S. (2011). Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. *Operations Research Letters*, *39*(5), 338-341.

Talbi, E. G. (2009). *Metaheuristics: from design to implementation*, 74. New Jersey: John Wiley & Sons.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research*, *32*(9), 2309-2327.

Tasan, A. S., & Gen, M. (2012). A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, *62*(3), 755-761.

Toth, P., & Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, *31*(4), 372-385.

Toth, P., & Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, *113*(3), 528-543.

Toth, P., & Vigo, D. (Eds.). (2002). *The vehicle routing problem* , 9. Philedelphia: Siam.

Wassan, N. (2007). Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, *58*(12), 1630-1641.

Wassan, N. A., Wassan, A. H., & Nagy, G. (2008). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, *15*(4), 368-386.

Wassan, N. A., Nagy, G., & Ahmadi, S. (2008). A heuristic method for the vehicle routing problem with mixed deliveries and pickups. *Journal of Scheduling*, *11*(2), 149-161.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European Journal of Operational Research*.

Zachariadis, E. E., & Kiranoudis, C. T. (2011). A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications*, *38*(3), 2717-2726.

Zachariadis, E. E., & Kiranoudis, C. T. (2012). An effective local search approach for the Vehicle Routing Problem with Backhauls. *Expert Systems with Applications*, *39*(3), 3174-3184.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, *36*(2), 1070-1081.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, *202*(2), 401-411.

Zèapfel, G., Braune, R., & Bèogl, M. (2010). *Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics*. Springer-Verlag Berlin Heidelberg.