M.Sc. Engg. Thesis

VEHICLE ROUTING PROBLEMS WITH SOFT TIME WINDOWS

by
Sumaiya Iqbal

Submitted to

Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000, Bangladesh

December, 2012

The thesis titled "**VEHICLE ROUTING PROBLEMS WITH SOFT TIME WINDOWS**," submitted by Sumaiya Iqbal, Roll No. 1009052017P, Session October 2009, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination was held on December 5, 2012.

Board of Examiners

Candidate's Declaration

This is to certify that the work entitled 'VEHICLE ROUTING PROBLEMS WITH SOFT TIME WINDOWS' is the outcome of the research carried out by me under the supervision of Dr. M. Sohel Rahman in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Sumaiya Iqbal
Candidate

Contents

Board of Examiners **i**

Candidate's Declaration **ii**

Acknowledgements **xii**

Abstract **xiii**

1 Introduction **1**

1.1 Vehicle Routing Problem 1

1.2 Application 3

1.3 Scope of this Thesis 4

1.3.1 Hardness of the Problem 4

1.3.2 Scope 5

1.4 Motivation 7

1.5 Contribution of this Thesis 8

1.6 Outline of the Thesis 10

2 Literature Review **11**

2.1 VRP and VRPTW 11

2.1.1 Exact and Heuristic Approaches 11

2.1.2 Metaheuristic Approaches 12

2.2 VRPSTW Problem 13

2.2.1 Exact and Heuristic Approaches 13

2.2.2 Metaheuristic Approaches 14

2.3 Previous Works on Artificial Bee Colony (ABC) Metaheuristic 14

3 Preliminaries **17**

3.1 VRPSTW 17

3.2 Artificial Bee Colony (ABC) Algorithm 20

3.2.1 Behavior of Foraging Honey Bee Swarm 21

3.2.2 The ABC Algorithm 23

4 ABC Algorithm for VRPSTW *(ABC_VRPSTW)* **28**

4.1 Heuristic vs. Metaheuristic Approach 28

4.2 ABC_VRPSTW 35

4.2.1 Generating Initial Solution (Food Source) 36

4.2.2 Determining Neighborhood Solutions (Food Sources) 37

4.2.3 Fitness Evaluation of Food Source 39

4.2.4 Assigning Food Sources to Employed Bees and Onlooker Bees . . . 40

4.2.5 Generating Scout Bee and Assigning Food Sources to Scout Bees . 41

4.2.6 Cost Evaluation 42

4.2.7 Overall Algorithm 43

4.3 Time Complexity Analysis of ABC_VRPSTW 45

4.4 ABC_VRPSTW and Some Other Variants of VRP and VRPTW 48

5 Experimental Results 52

5.1 Experimental Platform 52

5.2 Problem Instances 52

5.3 Experimental Plans 53

5.4 Results 53

6 Comparison with Previous Results 72

6.1 Discussion 79

7 Conclusion 88

7.1 Contribution 88

7.2 Future Work 90

7.2.1 Different VRP Variants 90

7.2.2 Future Scopes of ABC Algorithm 91

8 Appendix-I 93

9 Appendix-II 102

List of Figures

1.1 A Typical Scenario for the Vehicle Routing Problem 2

3.1 Typical Input for the Vehicle Routing Problem 18

3.2 An Output for the Instance in Figure 3.1 19

3.3 A Typical Problem Scenario 20

3.4 Behavior of Foraging Honey Bee in Nature 23

3.5 Flow Chart of ABC Algorithm 27

4.1 Sample Problem Scenario - I 30

4.2 Distances among the Customers and the Depot of Figure 4.1 31

4.3 Sample Solution following the Heuristic Strategy of Figure 4.1 32

4.4 Sample Solution using Metaheuristic of Figure 4.1 32

4.5 Sample Problem Scenario - II 33

4.6 Distances among the Customers and the Depot of Figure 4.5 33

4.7 Sample Solution following the Heuristic Strategy of Figure 4.5 34

4.8 Typical Problem Scenario - III 35

4.9 Distances among the Customers and the Depot of Figure 4.8 36

4.10 Sample Solution following the Heuristic Strategy of Figure 4.8 37

4.11 Sample Solution using metaheuristic of Figure 4.8 38

4.12 Problem Scenario 39

4.13 Distances among the Customers and the Depot of Figure 4.12 40

4.14 An Example Initial Solution - 1 for the Problem Scenario of Figure 4.12 . . 41

4.15 An Example Initial Solution - 2 for the Problem Scenario of Figure 4.12 . . 41

4.16 Initial Solution after the First Modification Step 42

4.17 Initial Solution after the Second Modification Step 43

4.18 Neighborhood Solution of the Initial Solution in Figure 4.14 44

5.1 Improvement of Traveling Cost with Iteration (R1 and R2 Problem Instance) 65

5.2 Improvement of Traveling Cost with Iteration (RC1 and RC2 Problem Instance) 66

5.3 Improvement of Traveling Cost with Iteration (C1 and C2 Problem Instance) 67

5.4 Variation of Results with Different Experimental Plans 68

5.5 Percentage of Improvements of Results with Different Experimental Plans . 69

5.6 Computational Time for Different Experimental Plans 70

5.7 Percentage of Increase of Computation Time among Different Plans 71

List of Tables

5.1 Parameters of Experimentation Plans 53

5.2 Best Result of ABC_VRPSTW for Problem Instances R1 **(Number of Customer = 200)** 54

5.3 Best Result of ABC_VRPSTW for Problem Instances R2 **(Number of Customer = 1000)** 55

5.4 Best Result of ABC_VRPSTW for Problem Instances RC1 **(Number of Customer = 200)** 56

5.5 Best Result of ABC_VRPSTW for Problem Instances RC2 **(Number of Customer = 1000)** 57

5.6 Best Result of ABC_VRPSTW for Problem Instances C1 **(Number of Customer = 200)** 58

5.7 Best Result of ABC_VRPSTW for Problem Instances C2 **(Number of Customer = 700)** 59

5.8 Improvement Sequence of Traveling Cost for R1 Problem Instance 60

5.9 Improvement Sequence of Traveling Cost for R2 Problem Instance 60

5.10 Improvement Sequence of Traveling Cost for RC1 Problem Instance 61

5.11 Improvement Sequence of Traveling Cost for RC2 Problem Instance 61

5.12 Improvement Sequence of Traveling Cost for C1 Problem Instance 62

5.13 Improvement Sequence of Traveling Cost for C2 Problem Instance 62

5.14 Best Result of ABC_VRPSTW for All Problem Instances **(in terms of Traveling cost)** 63

5.15 Computational time (sec) of ABC_VRPSTW for All Problem Instances . . 64

6.1 VRPSTW Results for R1 Problems, $W_{max} = 10\%$ and $P_{max} = 5\%$ 74

6.2 Results for RC1 Problems, $W_{max} = 10\%$ and $P_{max} = 5\%$ 75

6.3 VRPSTW Results for R1 Problems, $W_{max} = 10\%$ and $P_{max} = 10\%$ 76

6.4 Results for RC1 problems, $W_{max} = 10\%$ and $P_{max} = 10\%$ 77

6.5 Computational Time for Each Solution Method 78

6.6 Comparison for All R and RC Type Problem Instances **(Number of vehicles is same as in the benchmark data instances) - Part I** 80

6.7 Comparison for All R and RC Type Problem Instances **(Number of vehicles is same as in the benchmark data instances) - Part II** 81

6.8 Comparison for All R and RC Type Problem Instances **(Number of vehicles is equal to the best previous result) Part - I** 82

6.9 Comparison for All R and RC Type Problem Instances **(Number of vehicles is equal to the best previous result) Part - II** 83

6.10 Comparison for All R and RC Type Problem Instances **(Number of vehicles is improved (if possible) than the best previous result) Part I** 84

6.11 Comparison for All R and RC Type Problem Instances **(Number of vehicles is improved (if possible) than the best previous result) Part II** 85

6.12 Computational Time for Each Solution Method 85

6.13 Comparison for All R and RC Type Problem Instances **(with different number of vehicles)** Part - I 86

6.14 Comparison for All R and RC Type Problem Instances **(with different number of vehicles)** Part - II 87

8.1 Result of ABC_VRPSTW for Problem Instances R1-01 to R1-06 **(Number of Customer = 200)** 94

8.2 Result of ABC_VRPSTW for Problem Instances R1-07 to R1-12 **(Number of Customer = 200)** 95

8.3 Result of ABC_VRPSTW for Problem Instances R2-01 to R2-06 **(Number of Customer = 1000)** 96

8.4 Result of ABC_VRPSTW for Problem Instances R2-07 to R2-11 **(Number of Customer = 1000)** 97

8.5 Result of ABC_VRPSTW for Problem Instances RC1 **(Number of Customer = 200)** 98

8.6 Result of ABC_VRPSTW for Problem Instances RC2 **(Number of Customer = 1000)** 99

8.7 Result of ABC_VRPSTW for Problem Instances C1 **(Number of Customer = 200)** 100

8.8 Result of ABC_VRPSTW for Problem Instances C2 **(Number of Customer = 700)** 101

List of Algorithms

1 ABC Algorithm(pseudocode) 25

2 ABC_VRPSTW (Part - 1) 45

3 ABC_VRPSTW (Part - 2) 46

4 Generate-Random-Solution 103

5 Generate-Initial-Solution 104

6 Generate-Neighborhood-Solution($\mathcal{S}$) 104

7 Cost(S) 105

8 Fitness(S) 106

9 Onlooker-Search 106

Acknowledgments

All praises due to Allah, the most benevolent and merciful.

I express my heart-felt gratitude to my supervisor, Dr. M. Sohel Rahman for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I also want to show my gratitude to Dr. M. Kaykobad for his patience and valuable suggestions. It has been a great opportunity to work with him.

I also want to thank the other members of my thesis committee for their valuable suggestions. I thank Dr. Abu Sayed Md. Latiful Hoque, Dr. Mohammed Eunus Ali, and specially the external member Dr. Hasan Sarwar.

In this regard, I remain ever grateful to my beloved parents, who always exist as sources of inspiration behind every success I have ever made.

Abstract

The Vehicle Routing Problem (VRP) and its variants thereof are classical problems in computer science with diverse applications. The vehicle routing problem seeks to serve a number of customers with a fleet of vehicles. It can be described as the problem of designing optimal delivery or collection routes from one or several depots to a number of geographically scattered customers, subject to several constraints. It has been used to model various transportation and distribution related processes. Many other practical applications find the need for satisfying additional constraints and these necessities lead to many variants and extensions of VRP problems, which are complex to solve.

Due to the wide applicability and economic importance, the *Vehicle Routing Problem with Time Windows (VRPTW)* as well as its different variants has been extensively studied in computer science and transportation engineering literature. The vehicle routing problem with time windows consists of computing a minimum cost set of routes for a fleet of vehicles of limited capacity visiting a given set of customers with known demand, with the additional constraint that each customer must be visited within a specific time window.

Vehicle Routing Problem with Soft Windows (VRPSTW) is a relaxation of *Vehicle Routing Problem with Hard Time Windows (VRPHTW)*- in the former time window can be violated if a penalty is paid, whereas, in the latter, violations are not allowed. We consider the case in which time window constraints are relaxed into "soft" constraints, that is, penalty terms are added to the solution cost whenever a vehicle serves a customer outside the hard time window, but within the soft time window.

Vehicle routing problem is a combinatorial optimization problem and proved to be an NP-hard problem. VRPTW and its variants have been studied extensively in the literature. There have been both exact and heuristic approaches for solving VRPTW and its variants. The main contribution of this research is to focus on a new and efficient metaheuristic, *Artificial Bee Colony (ABC)*, inspired by the intelligent foraging behavior of the honey bee swarm and the application of ABC metaheuristic to solve the VRPTW problem with soft time windows. This research also focuses on how our approach can be easily modified to solve not only the VRPTW problem with soft time windows, but also many other variants of VRPTW problem. We have compared the performance of our ABC approach against the best approaches reported in the literature. Experimental results demonstrate the superiority of the new ABC approach over all the other approaches. The modular and intuitive approach obtains better quality solutions in reasonable amount of time.

Chapter 1

Introduction

1.1 Vehicle Routing Problem

About five decades ago Dantzig and Ramser [1] first introduced the Vehicle Routing Problem (VRP). It is a combinatorial optimization problem seeking to service a number of customers with a fleet of vehicles. Each vehicle has a certain capacity and each customer has a certain demand. Further on, there exists a depot and a distance (length / cost / time) matrix between the customers.

Figure 1.1 depicts a simple scenario of the problem. Here, three vehicles start their journey from the depot. After that, each vehicle starts serving customers as long as it has the capacity to serve a customer. When the capacity of a vehicle becomes less than the demand of a customer on its route, the vehicle comes back to the depot. The goal is to serve all the customers by the available vehicles. So, for each vehicle there is a single route containing the customers it serves.

The Vehicle Routing Problem with Time Windows (VRPTW) is one of the variants of VRP, which has been extensively studied in both Computer Science and Transportation Engineering literature [6, 13, 20, 21, 22]. In this variation, each customer has a constraint of being served within a specific time window. The Vehicle Routing Problem with Hard

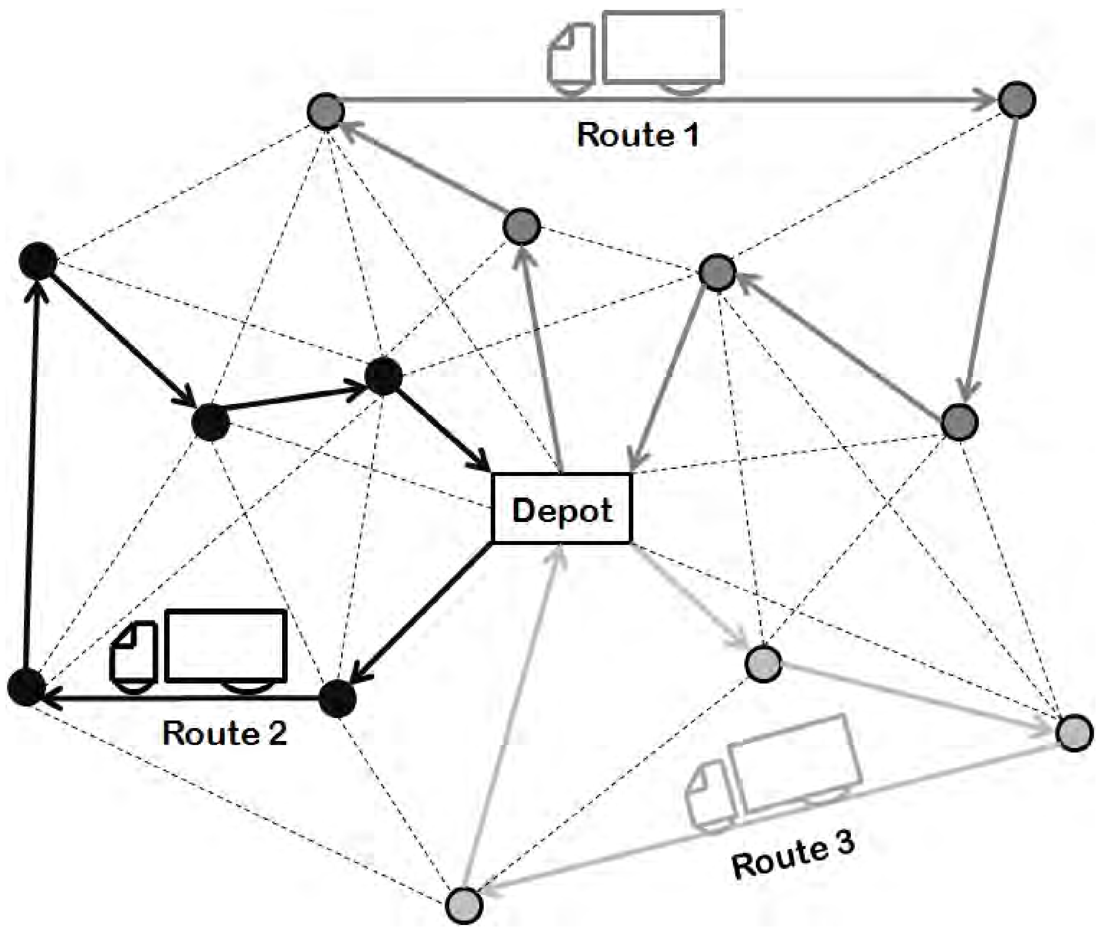


Figure 1.1: A Typical Scenario for the Vehicle Routing Problem

Time Windows (VRPHTW) is a well studied variant of VRPTW, where customers' time window cannot be violated. Another variation of VRPTW, Vehicle Routing Problem with Soft Time Windows (VRPSTW), is a relaxation of VRPHTW problem, where, the time windows can be violated by paying a penalty.

In the following sections, we focus on the real life applications of the Vehicle Routing Problem and its variants. we also discuss about the scope and motivation of our work and then the contribution of this thesis. Finally this chapter concludes with the overview of the rest of the chapters.

1.2 Application

Recent advances in technology have allowed the emergence of a wide new range of applications for vehicle routing. In particular, the last decade has seen the development of Intelligent Transport Systems (ITS), which are based on a combination of geolocation technologies, with precise geographic information systems, and increasingly efficient hardware and software for data processing and operations planning. Following are some applications of routing problem and their variants in case of providing service and transportation.

- A common application of routing can be found in maintenance operations. Maintenance companies are often committed by contract to their customers, which specify periodical or planned visits to perform preventive performance.
- Another application of routing arises in the context of organizations operates with a crew of service persons, who are called on duty via a call center coordinated with other emergency services.
- Application in city logistics is the courier service present in most urban areas. Couriers are dispatched to customer locations to collect packages and deliver them to their destination.
- The delivery of newspapers and magazines.
- Home delivery problem within a specific time window.

The variants of VRPTW are also important problems in the fields of transportation, distribution and logistics. Specifically, VRPSTW has many practical applications and seems to be practically more appealing because of the following reasons (borrowed from [2]):

- Relaxed time windows can result in lower total costs without disturbing customer satisfaction significantly.

- Many applications do not require hard time windows, *e.g.*, delivery of fuel/gas to service stations
- Travel times cannot be accurately known in many practical applications.
- The dispatcher may have qualitative information regarding the relative importance of service level across the customers.
- VRPSTW is a more general variant in the sense that the approaches to solve it can be used to solve some other variants of VRPTW problem (*e.g.*VRPHTW) by modifying the application of penalties appropriately.

In addition, VRPSTW solutions provide a workable plan of action when the problem with hard time windows is infeasible. Its increased practical visibility has evolved in parallel with the development of broader and deeper research directed at its solution.

1.3 Scope of this Thesis

1.3.1 Hardness of the Problem

It can be shown that how difficult the VRP problem is by the following way. Imagine a vehicle that has to deliver to 3 different locations A, B & C. The objective is to decide the order in which the vehicle should visit each location to minimize the overall travel distance.

There are 6 possible solutions:

- A-B-C
- A-C-B
- B-A-C
- B-C-A

- C-A-B
- C-B-A

So, the simplistic approach is to consider all 6 cases and work out the distance travelled for each one and finally choose the shortest. This simple problem would take a modern computer almost no time to solve. However, the difficulty increases surprisingly quickly as the number of deliveries increases:

- 4 locations have 24 possible solutions
- 5 locations have 120 possible solutions
- 6 locations have 720 possible solutions and so on.

N locations have $N \times (N-1) \times (N-2) \times \ldots 3 \times 2 \times 1$ solutions. This is known as experimental explosion. For a parcel delivery van which might make $80 - 100$ deliveries in a day, the number of possible routes/sequences is astronomical.

And, The VRP problem has already been proved as NP-hard [5]. Finding an exact solution for a real life optimization problem is sometimes less practical in comparison to using fast algorithm to compute near-optimal solutions. When the problems grow larger in size, obtaining the exact solutions can take excessive computational time and storage space. In such cases, the exact results obtained by a complex, time consuming method may turn out to be less attractive than near-optimal solutions. Furthermore, considering the imprecision of the real-world problem data, and the approximate nature of some formulations, obtaining a precise solution in reality may seem meaningless.

1.3.2 Scope

Three types of solution methods are typically employed to solve these types of problems (NP-hard problems):

- **Exact methods**. Exact methods guarantee that the optimal solution is found if the method is given sufficiently time and space. Exact methods must use clever techniques. But the worst case running time for NP-hard problems are going to be high though. We cannot expect to construct exact algorithms that solve NP-hard problems in polynomial time unless NP = P. For some classes of problems there are hope of finding algorithms that solve problem instances occurring in practice in reasonable time though.

- **Approximation algorithms**. Approximation algorithms are a special class of heuristic that provide a solution and an error guarantee. For example, one method could guarantee that the solution obtained is at most k times more costly than the best solution obtainable. Two classes of approximation algorithms called *polynomial time approximation scheme (PTAS)* and *fully polynomial time approximation scheme (FPTAS)* are of special interest as they can approximate the solution with any desired precision. That is for any instance I of the problem considered and any $\in > 0$ a PTAS or FPTAS can output a solution s such that $f(s) \leq (1 + \in)Opt$ (assuming that we are solving a minimization problem) where Opt is the optimal solution and $f(s)$ is objective of solution s. The difference between a PTAS and a FPTAS is that the PTAS is polynomial in the size of the instance I while the FPTAS is polynomial in the size of the instance I and $1/\in$. An FPTAS is therefore in a certain sense "stronger" than a PTAS. For some problems it is not possible to design a FPTAS, PTAS or even an polynomial time approximation algorithm with constant error guarantee unless $P = NP$ and approximation can be impractical: the error guarantee can be too poor or the running time of the algorithm can be too high.

- **Heuristics.** Heuristics are solution methods that typically relatively quickly can find a feasible solution with reasonable quality. There are no guarantees about the solution quality though, it can be arbitrarily bad. The heuristics are tested empirically and based on these experiments comments about the quality of the heuristic can be made. Heuristics are typically used for solving real life problems because of their speed and their ability to handle large instances.

A special class of heuristics that has received special attention in the last two decades is the metaheuristics. Metaheuristics provides general frameworks for heuristics that can be applied to many problem classes. High solution quality is often obtained using metaheuristics.

1.4 Motivation

The motivations of applying the Artificial Bee Colony (ABC) algorithm to solve VRPSTW problem is multi fold.

- Firstly, the ABC algorithm is a new swarm based metaheuristic approach for solving hard combinatorial optimization problems. It was first introduced by Karaboga [4] and inspired by the intelligent foraging behavior of honey bees. Recent application of ABC on different hard combinatorial problems have resulted in good solutions within reasonable and allowable time limit (*e.g.* solutions to the Leaf-constrained minimum spanning tree problem [3], Generalized assignment problem [36], In-Core Fuel Management Optimization problem [38], Traveling Salesperson Problem [39], Sudoku puzzles [37] etc.). This is specially true for offline problems where the results are not needed in real time and VRPSTW is one of such problems. Some surveys can also be found in the literature on the application of ABC algorithm [40, 41, 42].
- Secondly, in the literature most attempts to solve VRPSTW problem exploited heuristic approaches. In heuristic approaches, more often than not, there exists possibilities to exclude specific areas of the search space that are deemed uninteresting according to a particular heuristic. Metaheuristics in general, due to their stochastic nature, do not preclude such possibilities. In this case, we are motivated

to employ the ABC algorithm since it balances both exploration and exploitation to avoid local optima and reach the global optima. We will discuss this more elaborately later in Chapter 4.

- Thirdly, a metaheuristic is a general algorithmic framework for finding solutions to optimization problems. Within this framework, local heuristics are guided and adapted to effectively explore the solution space. VRPSTW is a discrete optimization problem (DOP) and metaheuristics are one popular algorithmic approach for generating solutions to DOPs.
- Fourthly, metaheuristics generally produce higher quality solutions but sometimes in expense of longer computational time. In our proposed ABC metaheuristics, effective and fast local search methods are applied on feasible solutions performing exchanges within a neighborhood maintaining the feasibility of the solutions. It helped us to avoid the local optima and get better solutions.
- Finally, different metaheuristics have already been applied to other variants of VRP and provided good result within reasonable amount of time [7, 8, 9, 10, 11, 12]. So, we are motivated to use it to solve VRPSTW problem.

1.5 Contribution of this Thesis

The study of VRPTW and its variants have given rise to major developments in the fields of exact algorithms and heuristics. Significant progress has been made both in the design of heuristics and metaheuristics approaches. Heuristic techniques have long been used to quickly solve many combinatorial optimization problems. Obtaining a near-optimal solution by heuristic/metahuristic techniques in a reasonable computational time may be beneficial and more practical. The real world need solution methods that are:

- Fast - the quicker the operator gets an answer back from the computer the better.

- Easy to apply to a variety of problem characteristics - when developing software for real life problems one wants to avoid reinventing the wheel every time a new client wants a software application for a new type of transportation problem.
- Precise - the better results a solution method returns the larger is the potential for savings.
- More robust - when solving real world problems it is often better to have a solution method that produces fairly good results for all problem instances.

The four characteristics listed above are to a certain extent in conflict with each other, so some sort of trade-off has to be achieved. Solution methods described in the literature are often evaluated in terms of speed, solution quality, and to a certain extent, robustness while the second characteristic listed above often receives less attention. In this thesis, a solution method that takes all four characteristics into account is presented.

The main contributions of this thesis are as follows.

- We focus on the VRPSTW problem and make an effort to solve the problem using Artificial Bee Colony metaheuristic.
- We have minimized the total traveling distance (cost) with penalties, the number of window violations and the number of routes and analyzed the performance of our algorithm both theoretically and experimentally.
- In particular, we have conducted exhaustive experiments using publicly available benchmark datasets to compare the performance of our algorithm with that of the state of the art algorithms in the literature. As will be reported later, ABC algorithm provides us with fast and high quality solutions to the VRPSTW problem.
- Additionally, we discuss how our algorithm can be adapted to handle other variants of the VRPTW problem.

1.6 Outline of the Thesis

The rest of the thesis is organized as follows. We describe related works in the literature in Chapter 2. In Chapter 3, we present the preliminary concepts necessary to comprehend the rest of the thesis. Here, we give a formal definition to VRPSTW and describe the ABC metaheuristic. We describe our algorithm to solve VRPSTW in Chapter 4. We present our experimental results and the comparison of our results with the previous ones in Chapter 5. Finally we briefly conclude in Chapter 7.

Chapter 2

Literature Review

In this chapter, we focus on the previous works carried out on both our problem, Vehicle Routing Problem and its variants and our metaheuristic approach. There is a vast amount of works in the literature on VRP and its variants. In this chapter, we follow the following strategy. We first discuss VRP and VRPTW in general. Then we shift our focus on to VRPSTW in particular. Also, within a particular section of our discussion, we first focus on exact and heuristic approaches followed by any metaheuristic works on the particular problem. After that, we discuss about the works done by the ABC metaheuristics.

2.1 VRP and VRPTW

Previous work on VRP, VRPTW and their variants include both exact, heuristic and metaheuristic approaches.

2.1.1 Exact and Heuristic Approaches

The complexity of a class of VRP is investigated by Lenstra and Rinnooy Kan [5] and shown as an NP-hard problem. The VRP with multiple use of vehicles is a variant of the classical vehicle routing problem. Azi *et al.* [6] introduce a branch-and-price approach to address this problem and lower bounds are computed using column generation method. The insertion-type heuristic was used by Solomon's [20] for vehicle routing and scheduling problems with time window constraints and Potvin and Rousseau [21] applied an insertion algorithm for the Vehicle Routing and Scheduling Problem with Time Windows (VRSPTW). A solution based on Atkinson's [45] Greedy Look-Ahead Heuristic for Combinatorial Optimization was given by loannou *et al.*'s [22] for solving VRPTW problem. A parallel route construction heuristic based on the insertion heuristic is applied by Hosny [43] for VRPTW problem. Recent publications include the work of Hashimoto and Yagiura [13] to solve VRPTW problem, which proposed a path relinking approach with an adaptive mechanism to control parameters. Here, the generated solutions using the path relinking approach are improved by a local search. Hsu *et al.* [14] proposed an algorithm tailored to the problem of perishable food distribution. In [15], the VRPHTW problem was tackled using a neighborhood-based heuristic, where a number of local search methods together with a diversification procedure were used. In [46], Tan *et al.*investigates various heuristic methods to solve the VRPTW to near optimal solutions. The heuristics explored by them are mainly third-generation artificial intelligent (AI) algorithms, namely simulated annealing (SA), Tabu search (TS) and genetic algorithm (GA). Based on the original SA theory proposed by Kirkpatrick [47] and the work by Thangiah *et al.* [48], we updated the cooling scheme and develop a fast and efficient SA heuristic. One of the variants of Glover's TS [49], strict Tabu, is evaluated and first used for VRPTW, with the help of both recency and frequency measures. Their GA implementation, unlike Thangiah's [50] genetic sectoring heuristic, used intuitive integer string representation and incorporates several new crossover operations and other advanced techniques such as hybrid hill-climbing and adaptive mutation scheme.

2.1.2 Metaheuristic Approaches

Researchers have also used a number of swarm based and metaheuristic approaches to solve VRPTW and its variants. Marinakis and Marinaki [7] introduced a new nature inspired approach based on Bumble Bees Mating Optimization for successfully solving the Vehicle Routing Problem. In [8], an improved ant colony optimization (IACO) is proposed to solve *Period Vehicle Routing Problem with Time Windows (PVRPTW).* A hybrid ant colony optimization (HACO) is used to solve VRPTW in [9] and a hybrid ant colony system (DSACA-VRPTW) is proposed in [10] to solve this problem. Here, each ants solution is improved by a dynamic sweep algorithm.

Häckel *et al.*presented a two-stage approach, ant algorithms and bee-inspired algorithms, that belongs to the class of metaheuristics to control a construction heuristic and has been applied successfully to Vehicle Routing Problem with Time Windows (VRPTW). A cooperative metaheuristic was used in [12] to solve VRPTW, which is based on the *solution warehouse strategy*, in which several search threads cooperate by asynchronously exchanging information on the best solutions identified. An improved ant colony optimization and a hybrid ant colony optimization were used to solve VRPHTW problem in [16] and [17], respectively. Tabu Perturbation Algorithm (TPA) integrated Tabu Search (TS) and Noising Method (NM) in [18] to develop a software for solving VRPHTW instances.

2.2 VRPSTW Problem

As our research is primarily focused on the VRPSTW problem, in this section, we will point out the works done in the literature on this specific variant of VRPTW problem. Here, we can also find exact, heuristic and metaheuristic based approaches.

2.2.1 Exact and Heuristic Approaches

Balakrishnan proposed three simple heuristics [19] for VRPSTW problem based on the nearest neighbor. In [23], the VRPSTW problem was heuristically decomposed into an assignment/clustering component and three simple heuristics were applied in [24] to obtain the results of VRPSTW problem instances. In [25], a nearest neighbor method was used which resulted in good solutions for the VRPSTW problem. Here, the nearest neighbor method was coupled with a problem generator that provides, in a structured manner, numerous instances that result from the manipulation of the level of time window violations and the population of customers that allow such violations. In [26], a goal programming approach was taken and Figliozzi in [27], proposed a new iterative route construction and improvement algorithm to solve VRPSTW.

2.2.2 Metaheuristic Approaches

To the best of our knowledge, there does not exist much metaheuristic approaches to solve VRPSTW in the literature. Genetic Algorithm was utilized in Bender's decomposition to solve VRPSTW problem [28]. Benders' decomposition or partitioning is widely-used technique for partitioning decision variables and assigning the task of selecting these variables to two interacting problems: the master problem which proposes values for the "hard" decisions and the subproblem which makes the "easy" decisions based upon those decisions proposed by the master problem (and in turn provides information in the form of dual variables to guide the master problem to making better decisions). In [29], tabu search is applied to a neighborhood of the current solution that swaps sequences of consecutive customers and also exploits an adaptive memory that contains the routes of the best previously visited solutions.

2.3 Previous Works on Artificial Bee Colony (ABC) Metaheuristic

The Artificial Bee Colony (ABC) algorithm is a swarm based meta-heuristic algorithm that was introduced by Karaboga [41] for optimizing numerical problems. The algorithm is specifically based on the model proposed by Tereshko and Loengarov [51] for the foraging behaviour of honey bee colonies. In this section, we focus on the application of ABC metaheuristics in the literature.

An enhanced Pareto-based artificial bee colony (EPABC) algorithm is applied by Wang *et al.*in [52] to solve the multi-objective flexible job-shop scheduling problem. Here, a pareto archive set is used to record the nondominated solutions that participate in crossover with a certain probability. In [53], a discrete artificial bee colony (DABC) algorithm is presented hybridized with an iterated greedy (IG) and iterated local search (ILS) algorithms. Here, the IG and ILS algorithms are embedded in a variable neighborhood search (VNS) procedure based on swap and insertion neighborhood structures to solve flow scheduling problem. Liu presented a hybrid discrete artificial bee colony (HDABC) algorithm in [54] to solve permutation flow shop scheduling problem. Here, Randomized Adaptive Search Procedure (GRASP) based on Nawaz Enscore Ham (NEH) heuristics is used. In [55], an ABC algorithm for solving generalized assignment problem is presented which is known to be an NP-hard problem. In [56], the application of the Chaotic Search ABC (CABC) algorithm in the PID (Proportional-Integral-Derivative) Control parameters tuning is simulated. Conventional PID control parameters tuning method uses the first proportional, second the integral, then the derivative three steps and changes the parameters until a satisfactory effect obtained. and [57], a discrete artificial bee colony (DABC) algorithm is proposed to solve the lot-streaming flow shop scheduling problem with the criterion of total weighted earliness and tardiness penalties under both the idling and no-idling cases. Here, an efficient initialization scheme, which is based on the earliest due date (EDD), the smallest slack time on the last machine (LSL) and the smallest overall slack time (OSL) rules, is presented to construct the initial population with certain quality and diversity. Furthermore, a simple but effective local search approach is embedded in the proposed DABC algorithm to enhance the local intensification capability. The artificial bee colony (ABC) algorithm is proposed to solve the multi-objective flexible job-shop scheduling problem in [58]. Here, the effective decoding scheme, hybrid initialization strategy and the exploration and exploitation abilities of ABC algorithm are used. Resource constrained project scheduling (RCPSP) is one of the most crucial problems in project problem. The aim of RCPSP, which is NP-hard, is to minimize the project duration. In [59], the artificial bee colony (ABC) algorithm is adopted to solve stochastic RCPSP and investigate its performance on the stochastic RCPSP. Various algorithms such as genetic algorithm and GRASP have been applied on stochastic RCPSP. Given an undirected, connected, weighted graph, the leaf-constrained minimum spanning tree (LCMST) problem seeks on this graph a spanning tree of minimum weight among all the spanning trees of the graph that have at least ℓ leaves. In [60], an artificial bee colony (ABC) algorithm is proposed for the LCMST problem.

Chapter 3

Preliminaries

This chapter presents the ideas necessary to comprehend the topics covered in this thesis. At first, we discuss the VRPSTW problem. After that, we present the behavior of foraging honey bee in the nature. This chapter then continues with the ABC algorithm based on this behavior of honey bees.

3.1 VRPSTW

The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem. The Vehicle Routing Problem (VRP) is a generic name given to a whole class of problems in which a set of routes for a fleet of vehicles based at one or several depots must be determined for a number of geographically dispersed cities or customers. The objective of the VRP is to deliver a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. In Figure 3.1 and Figure 3.2, we can see a picture of a typical input for a VRP problem and one of its possible outputs:

One of the most important extensions to the VRP is the VRPTW. This variant introduces additional constraints to the original definition that each costumer must be served within a specific time window (i.e., for each customer there is both an earliest and a latest time allowed for delivery). VRPSTW is a variant of Vehicle Routing Problem with Time

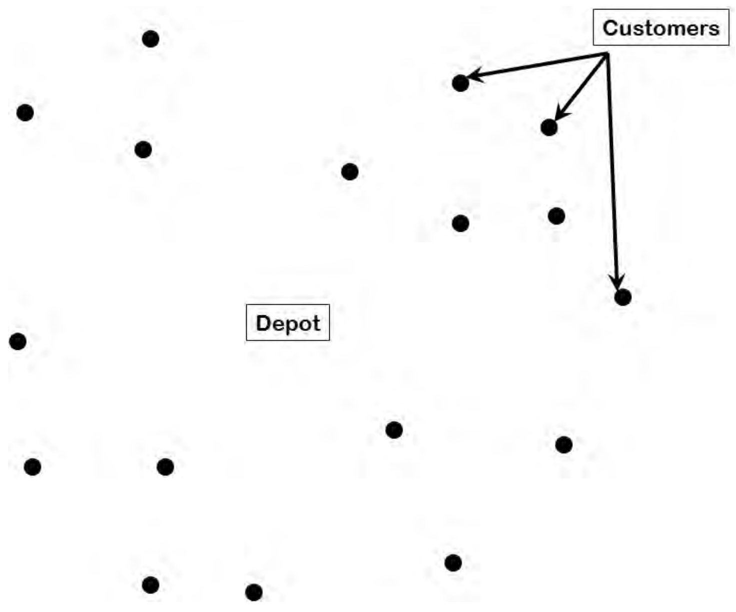


Figure 3.1: Typical Input for the Vehicle Routing Problem

Window where the window is maintained in a relaxed way. In other words, in VRPSTW the deliveries can be done outside the time window at the cost of some sort of penalties.

The VRPSTW can be modeled by a directed graph, $G = (V, E)$. Here, $V = \{v_0, v_1, v_2, \ldots, v_n\}$ is a set of $n+1$ vertices where each vertex v_i, $1 \leq i \leq n$ represents a customer and v_0 represents both the starting and ending depot. Now, E is a set of edges representing the connections between the depot and customers and among the customers. Clearly, $|V| = |C| + 1$, where, $C = \{c_1, c_2, \ldots, c_n\}$ is the set of customers. We use T to denote the set of vehicles. Now, we define the following variables:

- t_i denotes a *vehicle*, $1 \leq i \leq |T|$.
- For each vehicle t_i, there will be a route R_i, $1 \leq i \leq |T| = |R|$.
- Each vehicle has a fixed *capacity*, T_q and has the same *speed*, T_s.
- The *cost* associated with each edge $e(i, j) \in E$ is denoted by $cost_{ij}$.
- s_i is the start of *service time* for customer c_i.
- Every customer, $c_i \in C$ has a fixed demand, d_i.

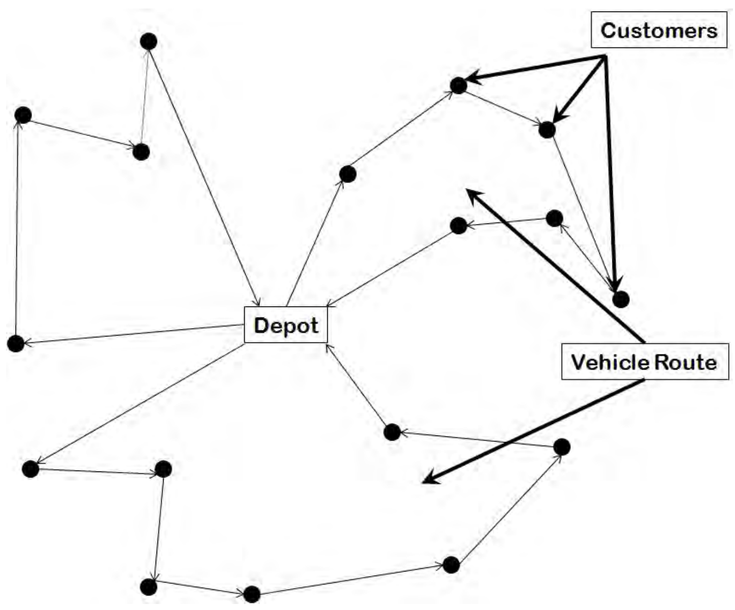


Figure 3.2: An Output for the Instance in Figure 3.1

- $[a_i, b_i]$, $1 \leq i \leq n$, is the *time window* associated to a customer c_i and $[a_0, b_0]$ is the *time window* associated to the depot.

$T_q, T_s, a_i, b_i, d_i, cost_{ij}, s_i$ are non-negative integers. There is an allowable violation of time windows denoted by $P_{max} \geq 0$. So for customer c_i, the relaxed time window is $[a_i - P_{max}, b_i + P_{max}] = [a_i^r, b_i^r]$. An early penalty $p_e \times (a_i - s_i)$ or a late penalty $p_\ell \times (s_i - b_i)$ is added to the service cost if any vehicle arrives earlier or late respectively. Here, both p_e and p_ℓ are penalty coefficients, $0 < p_e, p_\ell < 1$. The 'window break' for VRPSTW occurs when a customer c_i is not served within the un-relaxed time window $[a_i, b_i]$. If the customer, c_i is served within $[a_i^r, a_i]$ or $[b_i, b_i^r]$, then we have a 'window break' and appropriate penalty is added to the cost. The goal of the problem is to compute a set of routes, that minimizes total cost and number of window breaks satisfying the following constraints.

- Each customer is serviced exactly once and all the customers must be served.
- Every route originates and terminates at v_0, *i.e.*, the depot.
- The relaxed time windows of the customers and capacity constraints of the vehicles are obeyed.

- All the vehicles must start and come back to the depot with the time window of the depot, $[a_0, b_0]$.

Figure 3.3 illustrates an example of the problem scenario. Here, v_0 is the depot whose time window is $[a_0, b_0]$ according to VRPSTW problem. There are three vehicles and serves all the customers creating three routes and not violating any constraint.

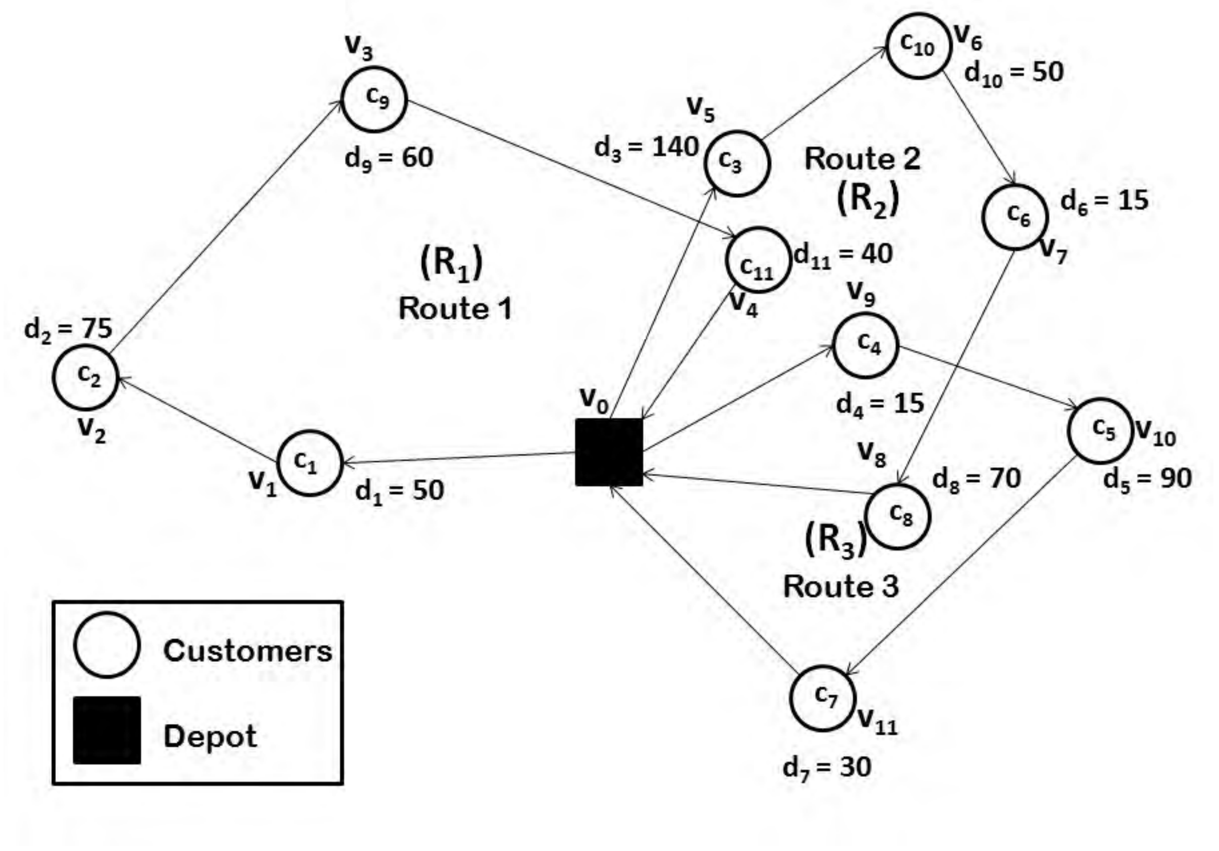


Figure 3.3: A Typical Problem Scenario

3.2 Artificial Bee Colony (ABC) Algorithm

The ABC algorithm is a new population-based metaheuristic proposed by Karaboga [4] and further developed by Karaboga and Basturk [44]. In this section, the foraging behavior of honey bees is described first and then the ABC algorithm is reviewed which is motivated by that behavior.

3.2.1 Behavior of Foraging Honey Bee Swarm

The term *swarm* in general refers to any restrained collection of interacting agents or individuals. Two fundamental concepts, self-organization and division of labor, are necessary and sufficient properties to obtain swarm intelligent behavior.

Self-organization: It can be defined as a set of dynamical mechanisms that establish basic rules for interactions between the components of the system. The rules ensure that the interactions are executed on the basis of purely local information without any relation to the global pattern. There are four basic properties on which self organization relies:

1. Positive feedback
2. Negative feedback
3. Fluctuations and
4. Multiple interactions

Division of Labor: In swarm behavior different tasks are performed simultaneously by specialized individuals. This is referred to as division of labor. It enables swarm to respond to changed conditions in the search space.

The model of forage selection that leads to the emergence of collective intelligence of honey bee swarms consists of three essential components as discussed below.

1. **Food Sources:** The value of a food source depends on many factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy. For the sake of simplicity, the *profitability* of a food source can be represented with a single quantity [30].

2. **Employed Foragers:** They are associated with a particular food source which they are currently exploiting or are employed at. They carry with them information about this particular source, its distance and direction from the nest, the profitability of the source and share this information with others with a certain probability.

3. **Unemployed Foragers:** They continuously look out for a food source to exploit. There are two types of unemployed foragers as follows.

   (a) **Scouts:** They search the environment surrounding the nest for new food sources. The mean number of scouts averaged over conditions is about 5-10% of the total bee population [30].

   (b) **Onlookers:** Each onlooker waits in the nest and establishes a food source through the information shared by employed foragers.

The two leading modes of behavior of a foraging bee is **the recruitment to a nectar source** and **the abandonment of a source**. The *exchange of information* among bees is the most important occurrence in the formation of the collective knowledge. The most important part of the hive with respect to exchanging information is the *Dancing Area*. Communication among bees related to the quality of food sources takes place in the dancing area. This dance is called a *Waggle Dance*.

Employed foragers share their information with a probability proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Since information about all the current rich sources is available to an onlooker on the dance floor, she can watch numerous dances and decides to employ herself at the most profitable source. There is a greater possibility of onlookers choosing more profitable sources since more information is circulated about the more profitable sources. Hence, the recruitment is proportional to the profitability of the food source [4]. Here, the basic self-organizing properties are as follows.

1. **Positive feedback:** As the nectar amount of food sources increases, the number of onlookers visiting them increases, too.

2. **Negative feedback**: The exploitation process of poor food sources is stopped by bees.

3. **Fluctuations:** The scouts carry out a random search process for discovering new food sources.

4. **Multiple interactions:** Bees share their information about food sources with their nest mates on the dance area.

Figure 3.4 illustrates the dynamic scenario of gathering information of bees from enviornment and adjusting the behavior of individual bees as described above.

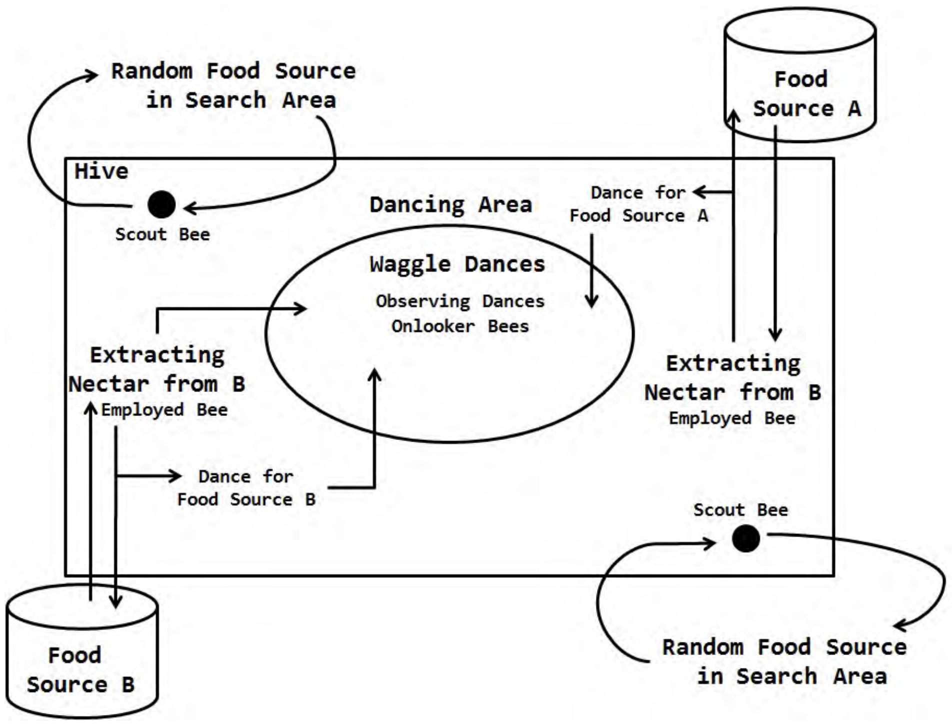


Figure 3.4: Behavior of Foraging Honey Bee in Nature

3.2.2 The ABC Algorithm

The ABC algorithm simulates behavior of real bees for solving multidimensional and multimodal optimization problems. In the ABC algorithm, each food source represents a possible solution to the problem under consideration and the nectar amount of a food source represents the quality of the solution. There are three types of bees - *Employed*, *Onlooker* and *Scout*. The ABC algorithm assumes that there is only one employed bee for every food source, *i.e.*, the number of food sources is the same as the number of employed bees. The employed bee of an abandoned food source becomes a scout and as soon as it finds a new food source it again becomes employed.

The ABC algorithm is an iterative algorithm. Each cycle of search consists of three steps:

1. Moving the employed and onlooker bees onto the food sources.
2. Calculating their nectar amounts.
3. Determining the scout bees and directing them onto possible food sources.

A food source is a possible solution to the problem to be optimized. The amount of nectar of a food source corresponds to the quality of the solution. All employed bees are first associated with randomly generated food sources (solution). During each iteration, every employed bee determines the best food source in the neighborhood of its current food source (*i.e.*among the neighborhood food sources) and evaluates its nectar amount (fitness). If its nectar amount is better than that of its currently associated food source then that employed bee moves to the new food source, otherwise it retains the current one.

When all employed bees have finished this process, they share the nectar information of the food sources with the onlookers. Onlookers are placed on the food sources by using a probability based selection process. As the nectar amount of a food source increases, the probability value with which the food source is preferred by onlookers increases, too. The probability p_i of selecting a food source i is: $p_i = \frac{f_i}{\sum_{j=1}^{m} f_j}$, where, f_i (f_j) is the fitness of the solution represented by the food source i (j) and m is the total number of food sources. After all onlookers have selected their food sources, each of them becomes employed and selects the best food source among the neighborhood food sources in the same way as described earlier.

If a solution represented by a particular food source does not improve for a predetermined number of iterations then that food source is abandoned by its associated employed bee and it becomes a scout, i.e., it will search for a new food source randomly. This tantamount to assigning a randomly generated food source (solution) to this scout and changing its status again from scout to employed. After that it selects the best neighborhood food source in the same way as described earlier. The whole process is repeated again and again till the termination condition is satisfied. Algorithm 1 formally presents the ABC algorithm.

Algorithm 1 ABC Algorithm(pseudocode)

```
Send the scouts on to the initial food sources.
while requirements are met do
    Send the employed bees onto the food sources and determine their nectar amounts.
    Determine neighborhood food sources of the initial one by the employed bee.
    Calculate the probability value of all the sources with which they are preferred by
    the onlooker bees.
    Send the onlooker bees onto the food sources and determine their nectar amounts.
    Stop the exploitation process of the sources exhausted by the bees.
    Send the scouts into the search are for discovering new food sources randomly.
    Memorize the best food source found so far.
end while
```

ABC algorithm in fact employs four different selection processes:

- A global selection process used by the artificial onlooker bees for discovering promising regions.
- A local selection process carried out in a region by the artificial employed bees and the onlookers depending on local information.
- A local selection process called greedy selection process carried out by all bees in that if the nectar amount of the neighborhood food source is better than that of the present one, the bee forgets the present one and memorizes the candidate source. Otherwise the bee keeps the present one in the memory.
- A random selection process carried out by scouts.

Here, onlookers and employed bees carry out the exploitation process in the search space. And scouts control the exploration process. So, we have a way to mix both the exploitative (usually local) and explorative (usually global) approach to search better quality solutions. That's why, by applying this algorithm, we can expect to have fast and higher quality solutions to VRPSTW instances. The whole process of ABC algorithm is explained with the help of a flow-chart shown in Figure 3.5.

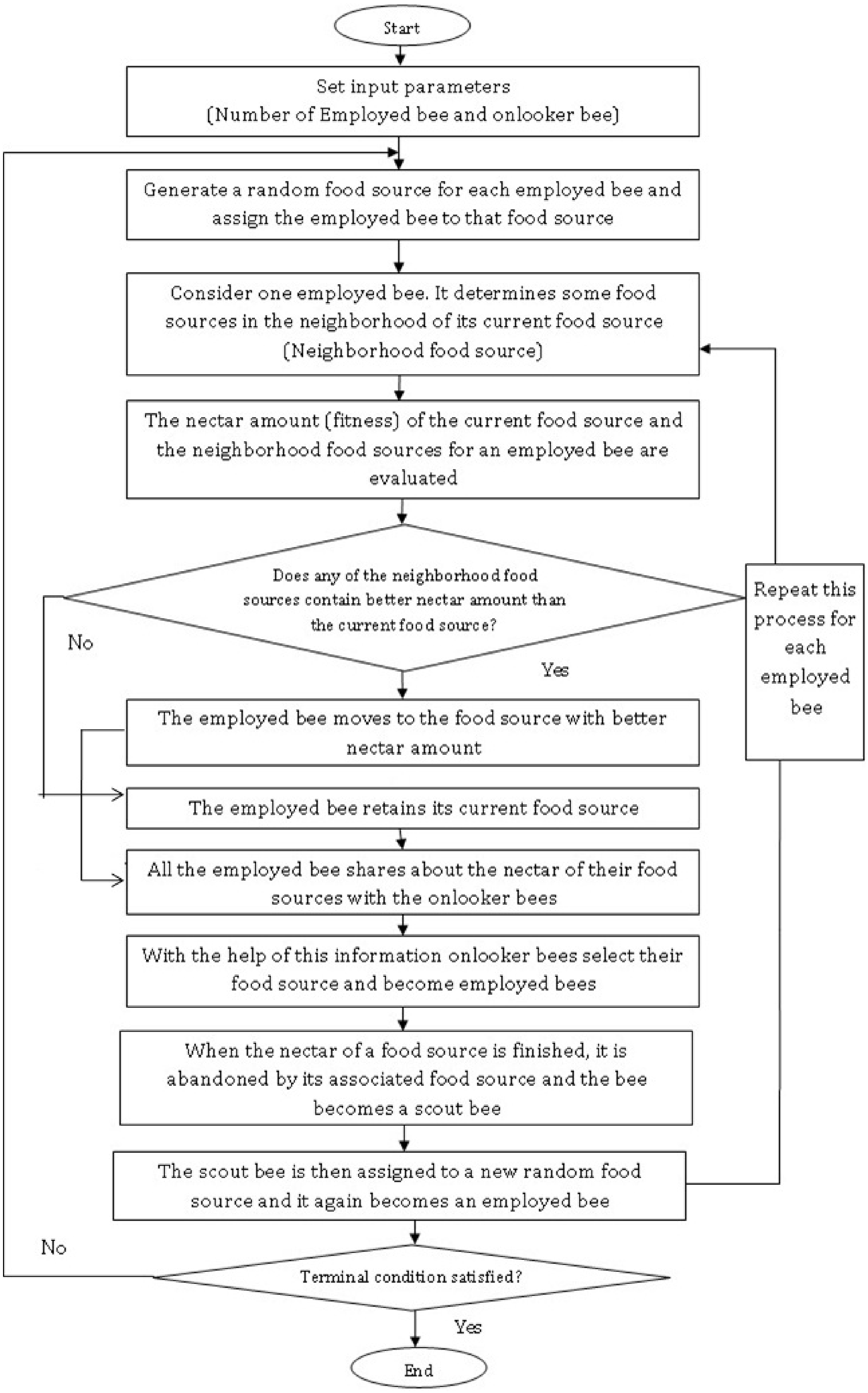

Start
Set input parameters (Number of Employed bee and onlooker bee)
Generate a random food source for each employed bee and assign the employed bee to that food source
Consider one employed bee. It determines some food sources in the neighborhood of its current food source (Neighborhood food source)
The nectar amount (fitness) of the current food source and the neighborhood food sources for an employed bee are evaluated
Does any of the neighborhood food sources contain better nectar amount than the current food source?
No
Yes
Repeat this process for each employed bee
The employed bee moves to the food source with better nectar amount
The employed bee retains its current food source
All the employed bee shares about the nectar of their food sources with the onlooker bees
With the help of this information onlooker bees select their food source and become employed bees
When the nectar of a food source is finished, it is abandoned by its associated food source and the bee becomes a scout bee
The scout bee is then assigned to a new random food source and it again becomes an employed bee
No
Terminal condition satisfied?
Yes
End

Figure 3.5: Flow Chart of ABC Algorithm

Chapter 4

ABC Algorithm for VRPSTW (*ABC_VRPSTW*)

In this chapter, we present a new metaheuristic algorithm based on Artificial Bee Colony (ABC) to solve VRPSTW. This algorithm contains novel ideas for both local and global search techniques. As will be shown in the next chapter, this algorithm produces comparable results with the current state-of-art algorithms. Also to the best of our knowledge, this work is the first attempt to solve VRPSTW using ABC. This chapter is organized as follows. At first, we present why we are motivated to apply the metaheuristic as a better choice than using a specific strategy of a heuristic for solving our problem. Then, we present the formation and mapping of the ABC_VRPSTW algorithm. This chapter concludes with the time complexity analysis of our algorithm.

4.1 Heuristic vs. Metaheuristic Approach

Heuristic refers to experience-based techniques for problem solving, learning, and discovery. Where an exhaustive search is impractical, heuristic methods are used to speed up the process of finding a satisfactory solution. And, metaheuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. We can say, a heuristic is a “good guess” function used as a building block of a larger (usually search) algorithm. A metaheuristic is sort of a “good guess” system in itself that keeps refining its guesses. We can think of a heuristic like an approximate (not approximation) solution to a problem. The difference between approximate and approximation is that the first is about getting a good guess of the solution of a problem, but that you don’t really know how good it is. The latter is about getting a solution for which you can prove how close it is to the optimal solution. Heuristics are often problem-dependent, that is, you define an heuristic for a given problem. Metaheuristics are problem-independent techniques that can be applied to a broad range of problems. Heuristic methods follow a specific strategy which can miss a better solution in a large search space. But the metaheuristic approach follows some kind of stochastic search method. It incorporates both local search and randomness by which the possibility of missing a better solution becomes low. Below we give an example demonstrating the limitations of a heuristic strategy which motivates us to apply a metaheuristic.

In our VRPSTW problem, the summation of the demand of the customers served by a vehicle must be less than the vehicle capacity. So, a simple heuristic strategy can be defined to add customers into a route of a vehicle, that is, *a vehicle will always serve a customer first that has lower demand among all the adjacent customers from its current position.* Assume that, three vehicles start from the depot at the same time. In our homogeneous system, all the vehicles have the same capacity. Assume the capacity of each vehicle is $T_q = 300$ objects. Now, let us consider the problem scenario of Figure 4.1. There are three vehicles serving thirteen customers. The costs (distances) among the adjacent customers and the depot is given in Figure 4.2 (the distances are assumed from Figure 4.1). Then a solution achieved through the above heuristic strategy is shown in Figure 4.3. Recall that, here, $t_1(R_1) = \{depot, c_{13}, c_{10}, c_{11}, c_6, c_{12}, depot\}$, $t_2(R_2) = \{depot, c_1, c_2, c_3, c_9, c_8, depot\}$ and $t_3(R_3) = \{depot, c_4, c_5, c_7, depot\}$ are the three routes, R_1, R_2, R_3, of the three vehicles t_1, t_2, t_3 respectively.

Now, let us calculate the cost of the solution which is the summation of distances

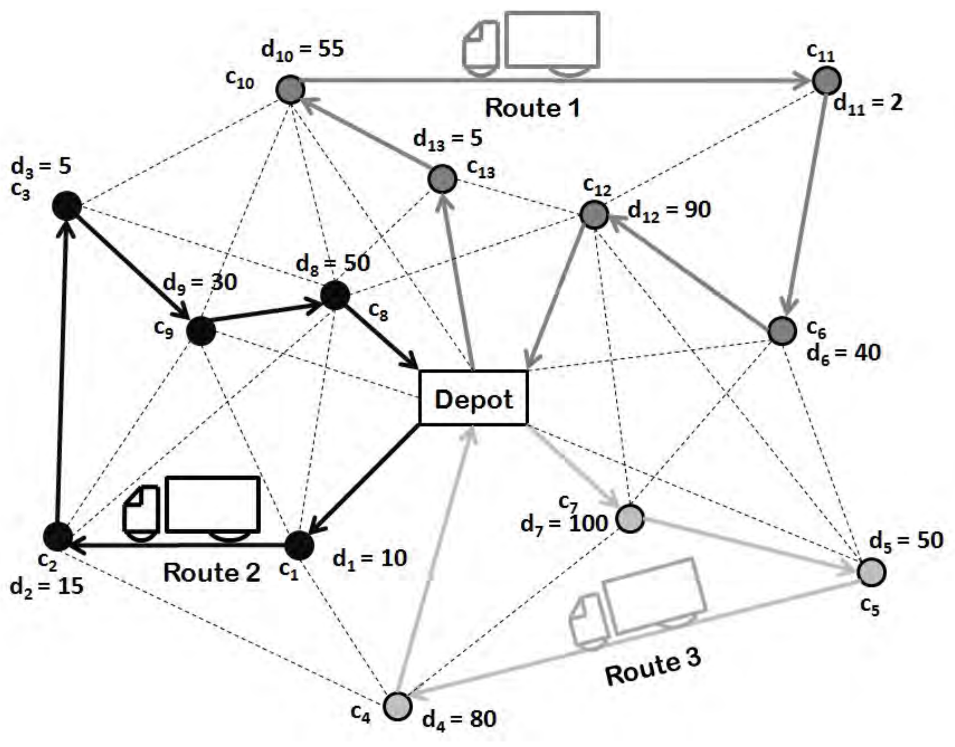


Figure 4.1: Sample Problem Scenario - I

between the customers and the depot. Cost of the solution shown in Figure 4.3 is:

$cost_{depot,13} + cost_{13,10} + cost_{10,11} + cost_{11,6} + cost_{6,12} + cost_{12,depot} +$

$cost_{depot,1} + cost_{1,2} + cost_{2,3} + cost_{3,9} + cost_{9,8} + cost_{8,depot} +$

$cost_{depot,4} + cost_{4,5} + cost_{5,7} + cost_{7,depot}$

$=2 + 1 + 7 + 3 + 3 + 2 +$

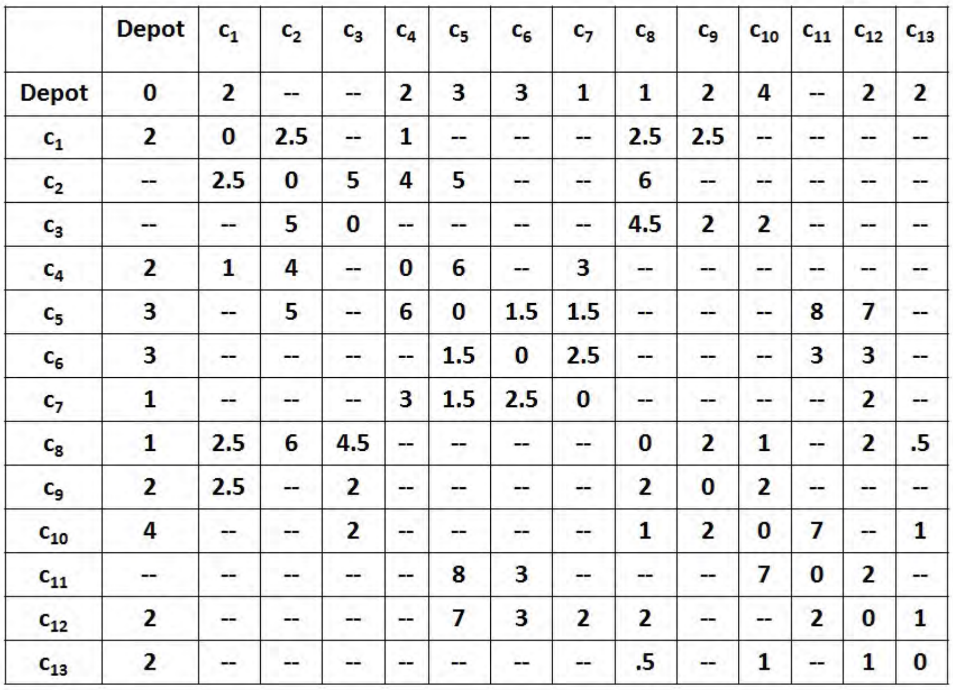
$2 + 2.5 + 5 + 2 + 2 + 1 +$

$2 + 6 + 1.5 + 1$

$= 41$ unit.

Note that, the solution given by the above heuristic strategy is a deterministic one. Now, if we do not follow a specific strategy and create a solution by local search and random restart and refine them in multiple iteration, then a solution can be found as shown in Figure 4.4.

Now, let us again calculate the cost of the solution in Figure 4.4.



	Depot	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}
Depot	0	2	--	--	2	3	3	1	1	2	4	--	2	2
c_1	2	0	2.5	--	1	--	--	--	2.5	2.5	--	--	--	--
c_2	--	2.5	0	5	4	5	--	--	6	--	--	--	--	--
c_3	--	--	5	0	--	--	--	--	4.5	2	2	--	--	--
c_4	2	1	4	--	0	6	--	3	--	--	--	--	--	--
c_5	3	--	5	--	6	0	1.5	1.5	--	--	--	8	7	--
c_6	3	--	--	--	--	1.5	0	2.5	--	--	--	3	3	--
c_7	1	--	--	--	3	1.5	2.5	0	--	--	--	--	2	--
c_8	1	2.5	6	4.5	--	--	--	--	0	2	1	--	2	.5
c_9	2	2.5	--	2	--	--	--	--	2	0	2	--	--	--
c_{10}	4	--	--	2	--	--	--	--	1	2	0	7	--	1
c_{11}	--	--	--	--	--	8	3	--	--	--	7	0	2	--
c_{12}	2	--	--	--	--	7	3	2	2	--	--	2	0	1
c_{13}	2	--	--	--	--	--	--	--	.5	--	1	--	1	0

Figure 4.2: Distances among the Customers and the Depot of Figure 4.1

$cost_{depot,12} + cost_{12,11} + cost_{11,6} + cost_{6,5} + cost_{5,7} + cost_{7,depot}$

$cost_{depot,1} + cost_{1,2} + cost_{2,4} + cost_{4,depot}$

$cost_{depot,13} + cost_{13,10} + cost_{10,3} + cost_{3,9} + cost_{9,8} + cost_{8,depot}$

$= 2 + 2 + 3 + 2 + 1.5 + 1 +$

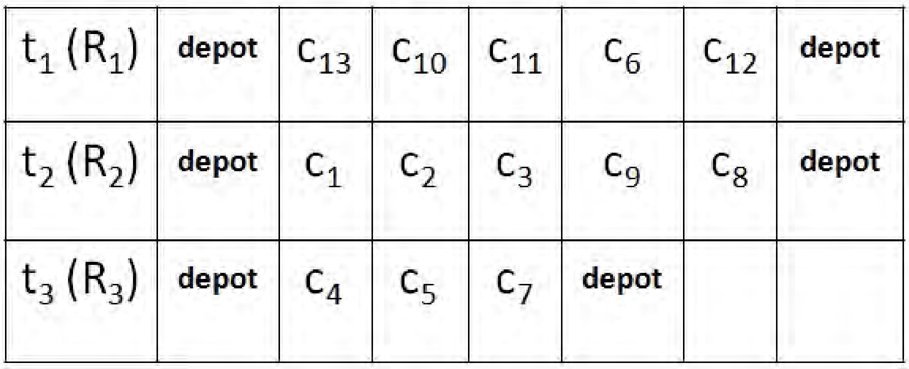
$2 + 2.5 + 4 + 3$

$2 + 1 + 2 + 2 + 2 + 1 +$

$= 33$ unit.

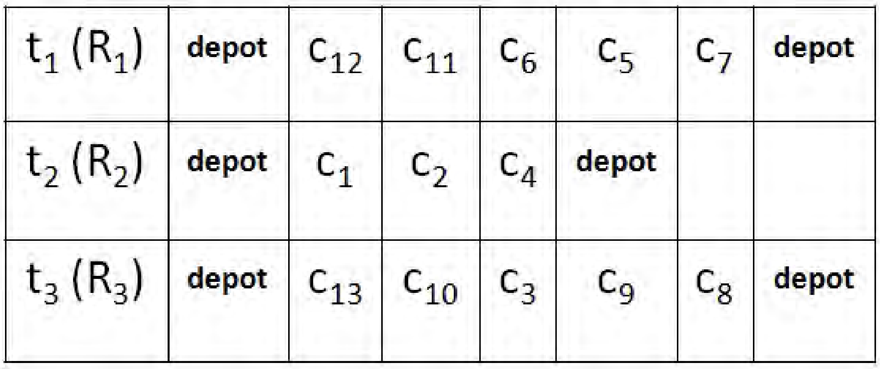
So, it can be said from the above example, that in a problem like vehicle routing, it is possible to miss a better just following a fixed strategy. It is better to search in a stochastic way and refine the solution in multiple cycles to have good solution.

Below we present another example that clarifies that how a heuristic strategy can miss a particular area of the search space completely. Assume another heuristic strategy, that is, *starting from depot a vehicle will always choose the next position which is the*



t_1 (R_1)	**depot**	C_{13}	C_{10}	C_{11}	C_6	C_{12}	**depot**
t_2 (R_2)	**depot**	C_1	C_2	C_3	C_9	C_8	**depot**
t_3 (R_3)	**depot**	C_4	C_5	C_7	**depot**		

Figure 4.3: Sample Solution following the Heuristic Strategy of Figure 4.1



t_1 (R_1)	**depot**	C_{12}	C_{11}	C_6	C_5	C_7	**depot**
t_2 (R_2)	**depot**	C_1	C_2	C_4	**depot**		
t_3 (R_3)	**depot**	C_{13}	C_{10}	C_3	C_9	C_8	**depot**

Figure 4.4: Sample Solution using Metaheuristic of Figure 4.1

nearest among all the customers and depot. Now, let us consider the problem scenario of Figure 4.5. Here, there is one vehicle. The costs (distances) among the adjacent customers and the depot is given in Figure 4.6 (the distances are assumed from Figure 4.5).

Then a solution achieved through the above heuristic strategy is shown in Figure 4.7. Note that, here due to the fixed strategy of the heuristic, a specific area of the search space remains completely unexplored and the vehicle will never serve the customers in the unexplored search space.

Here, again the stochastic search process of metaheuristic can help us to explore the whole search space and can give us a complete and better solution.

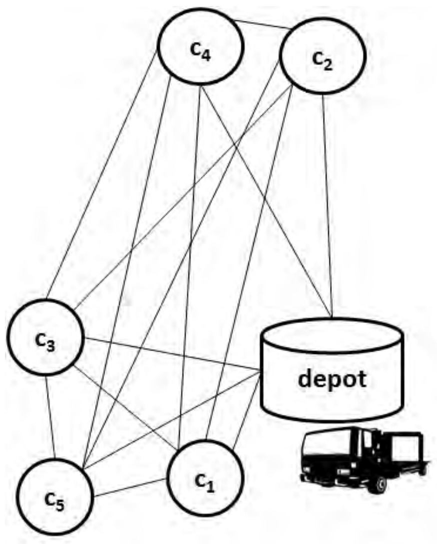


Figure 4.5: Sample Problem Scenario - II

	depot	c_1	c_2	c_3	c_4	c_5
depot	0	1	3	2	4	2.5
c_1	1	0	6	1.5	6	1
c_2	3	6	0	5	.5	8
c_3	2	1.5	5	0	5	1
c_4	4	6	.5	5	0	7
c_5	2.5	1	8	1	7	0

Figure 4.6: Distances among the Customers and the Depot of Figure 4.5

Now, we present another example, for which, the heuristic strategy is completely based on the time window. And, here we again show that, how a fixed strategy depending only on time window can make the solution deterministic and cam miss a better solution. Assume, the heuristic strategy is, *starting from depot, the vehicle will always serve the customer whose opening soft time is earlier among all the adjacent customers and depot.* Now, let us consider the problem scenario of Figure 4.8. Here, there is one vehicle. The costs (distances) among the adjacent customers and the depot is given in Figure 4.9 (the distances are assumed from Figure 4.8).

The solution found by following the heuristic strategy is shown in Figure 4.10. This

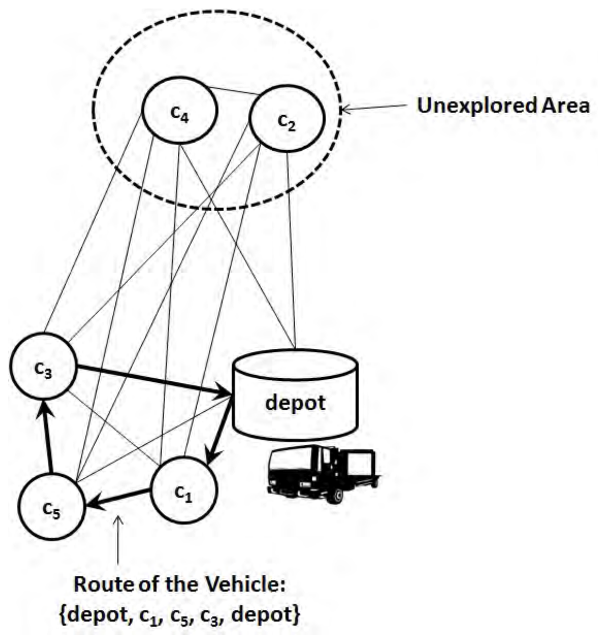


Figure 4.7: Sample Solution following the Heuristic Strategy of Figure 4.5

solution is a deterministic one.

Now, the cost of the solution given in Figure 4.10 is following.

$cost_{depot,4} + cost_{4,2} + cost_{2,3} + cost_{3,1} + cost_{1,depot}$
$= 3.5 + 2 + 2 + 4.5 + 4$
$= 16$ unit.

Now, following is another solution, shown in Figure 4.11, with does not follow the specific heuristic strategy, but the time window constraint is followed. This solution can be found by the stochastic search process of metaheuristic.

Now, the cost of the solution given in Figure 4.11 is following.

$cost_{depot,2} + cost_{2,4} + cost_{4,3} + cost_{3,1} + cost_{1,depot}$
$= 2 + 2 + 2 + 4.5 + 4$
$= 14.5$ unit.

So, it is clear from the above example, that in a problem like vehicle routing problem

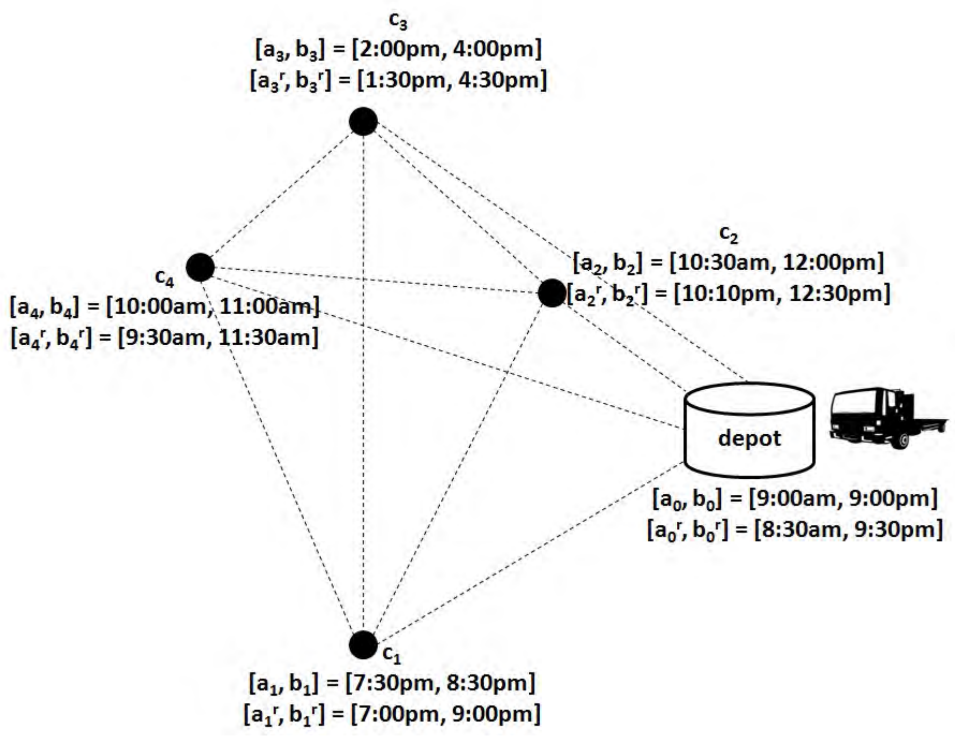
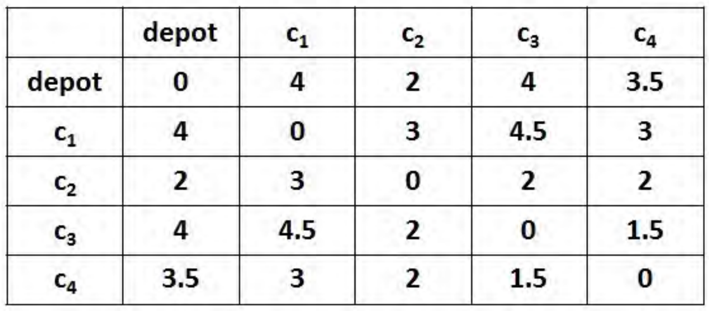

c3
[a3, b3] = [2:00pm, 4:00pm]
[a3r, b3r] = [1:30pm, 4:30pm]
c2
[a2, b2] = [10:30am, 12:00pm]
[a2r, b2r] = [10:10pm, 12:30pm]
c4
[a4, b4] = [10:00am, 11:00am]
[a4r, b4r] = [9:30am, 11:30am]
depot
[a0, b0] = [9:00am, 9:00pm]
[a0r, b0r] = [8:30am, 9:30pm]
c1
[a1, b1] = [7:30pm, 8:30pm]
[a1r, b1r] = [7:00pm, 9:00pm]

Figure 4.8: Typical Problem Scenario - III

with soft time window, it is possible to miss a better just following a fixed strategy. It is better to search in a stochastic way and refine the solution in multiple cycles to have good solution.

4.2 ABC_VRPSTW

In this section, we present the ABC_VRPSTW algorithm that applies the ABC metaheuristic to solve VRPSTW. Each solution of our problem can be seen as a collection of routes (of vehicles), where each route is presented by an integer-valued vector. Recall that, for each vehicle, t_i, there will be a route, R_i. If a vehicle, t_i serves customers in the order of $depot, c_1, c_2, \ldots, c_k, depot$, then the vector representing the route, R_i is $\{c_1, c_2, \ldots, c_k\}$. The depot is by default assumed to be in the first and last position of every route. Figure 4.12 presents the problem scenario. Here, there are three vehicles and eleven customers. The capacity of each of the vehicle is 300 objects. v_0 represents the



	depot	c_1	c_2	c_3	c_4
depot	0	4	2	4	3.5
c_1	4	0	3	4.5	3
c_2	2	3	0	2	2
c_3	4	4.5	2	0	1.5
c_4	3.5	3	2	1.5	0

Figure 4.9: Distances among the Customers and the Depot of Figure 4.8

depot and the demand of each customer is also shown in the figure. And the distances among customers and depot is given in Figure 4.13.

4.2.1 Generating Initial Solution (Food Source)

The following steps initializes a single solution:

- For each vehicle t_i, random customers are removed from the available customer pool and assigned to t_i without violating the capacity constraint.
- At first, each vehicle is assigned $\frac{|C|}{|V|}$ customers and then the assignment process for that vehicle is terminated.
- If there are still some unassigned customers, then they are assigned to the vehicles without violating their capacity constraints.
- If there are still unassigned customers after all the steps, discard the solution and commence the initialization step from the beginning again.

In order to ensure diversity, each initial solution is ensured to be unique. For the problem scenario shown in Figure 4.12, an initial solution can be as shown in Figure 4.14. Here, $R_1 = c_1, c_2, c_9, c_{11}$, $R_2 = c_3, c_{10}, c_6, c_8$ & $R_3 = c_4, c_5, c_7$ represents the three routes of the three vehicles. And, another initial solution can be as shown in Figure 4.15. Here, $R_1 = c_1, c_2, c_9$, $R_2 = c_3, c_{10}, c_6, c_5, c11$ & $R_3 = c_4, c_8, c_7$ represents the three routes of the three

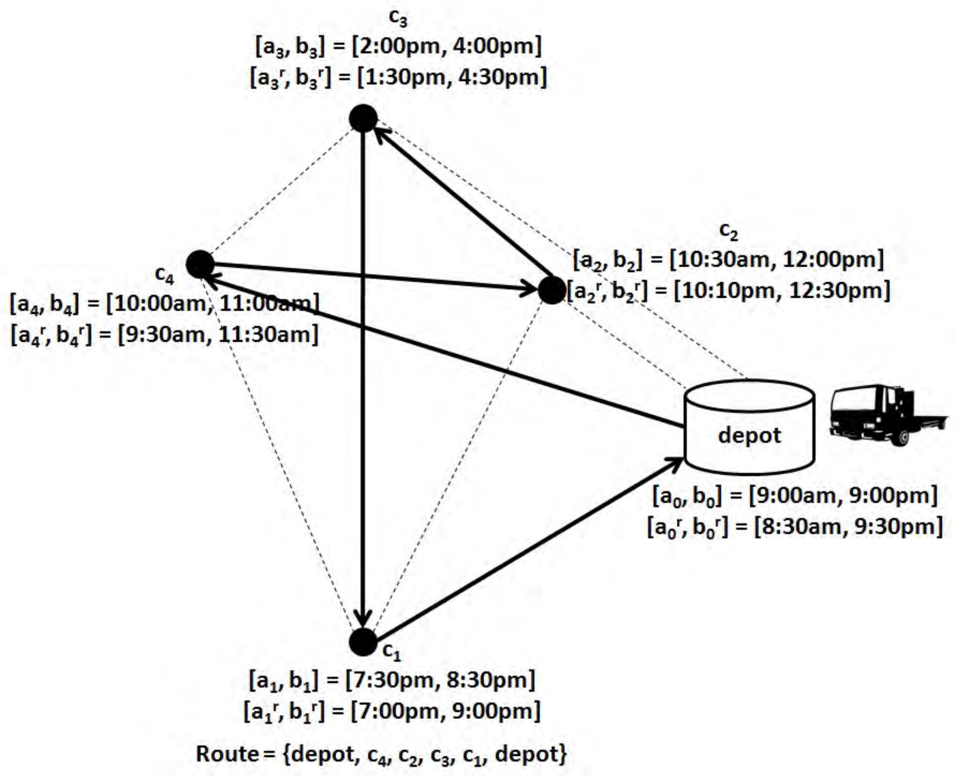

c3
[a3, b3] = [2:00pm, 4:00pm]
[a3r, b3r] = [1:30pm, 4:30pm]
c2
[a2, b2] = [10:30am, 12:00pm]
[a2r, b2r] = [10:10pm, 12:30pm]
c4
[a4, b4] = [10:00am, 11:00am]
[a4r, b4r] = [9:30am, 11:30am]
depot
[a0, b0] = [9:00am, 9:00pm]
[a0r, b0r] = [8:30am, 9:30pm]
c1
[a1, b1] = [7:30pm, 8:30pm]
[a1r, b1r] = [7:00pm, 9:00pm]
Route = {depot, c4, c2, c3, c1, depot}

Figure 4.10: Sample Solution following the Heuristic Strategy of Figure 4.8

vehicles. But, in the second solution the demand constraint is violated in the second route. So, that initial solution is discarded.

4.2.2 Determining Neighborhood Solutions (Food Sources)

In each iteration of the algorithm, each employed bee is first assigned to a random initial food source. After that, each employed bee explores the neighborhood area of the currently assigned food source for good neighborhood food source. It is desirable to utilize the information of the initial food source while generating a neighboring food source. The following two modifications are done on the currently assigned solution to produce neighborhood solutions from it:

1. At first two different routes R_i and R_j are chosen randomly from the solution. Then, two customers c_k from R_i and c_ℓ from R_j are selected, again randomly. Then c_ℓ and

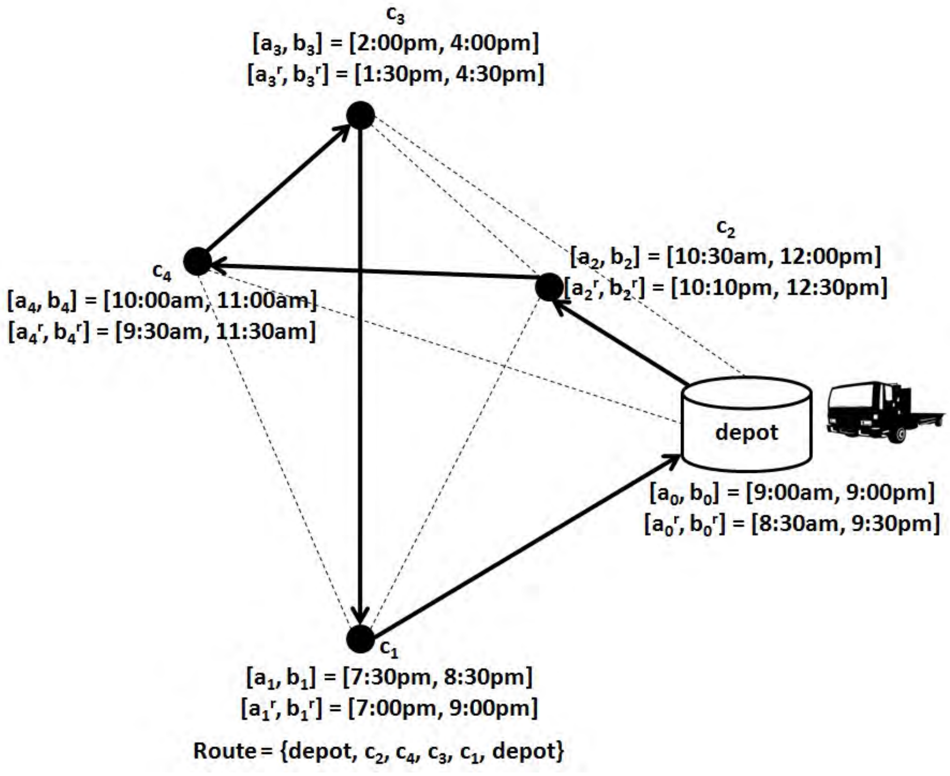

c3
[a3, b3] = [2:00pm, 4:00pm]
[a3r, b3r] = [1:30pm, 4:30pm]
c2
[a2, b2] = [10:30am, 12:00pm]
[a2r, b2r] = [10:10pm, 12:30pm]
c4
[a4, b4] = [10:00am, 11:00am]
[a4r, b4r] = [9:30am, 11:30am]
depot
[a0, b0] = [9:00am, 9:00pm]
[a0r, b0r] = [8:30am, 9:30pm]
c1
[a1, b1] = [7:30pm, 8:30pm]
[a1r, b1r] = [7:00pm, 9:00pm]
Route = {depot, c2, c4, c3, c1, depot}

Figure 4.11: Sample Solution using metaheuristic of Figure 4.8

c_k are swapped, *i.e.*, in the new solution R_i (R_j) now contains c_ℓ (c_k) instead of c_k (c_ℓ).

For the sample initial solution shown in Figure 4.14, let, $i = 2$, $j = 3$, $k = 3$ and $\ell = 7$ and c_k respectively. Then after this first modification step, the initial solution becomes as shown in Figure 4.16.

2. The order of customers being served is important. Varying this order can also vary the violation of time windows. In this step, we select a block of customers from a randomly chosen route of the solution and replace that block by a random permutation of itself. Here, the mixing and matching process of ***Genetic Algorithm (uniform crossover withing a single string)*** is used to tweak the initial solution with a goal to generate high quality neighborhood solution.

   From Figure 4.16, if randomly chosen route is R_2 and the block of customers is c_7, c_{10} and c_6, then after the second modification step, the solution is as shown in

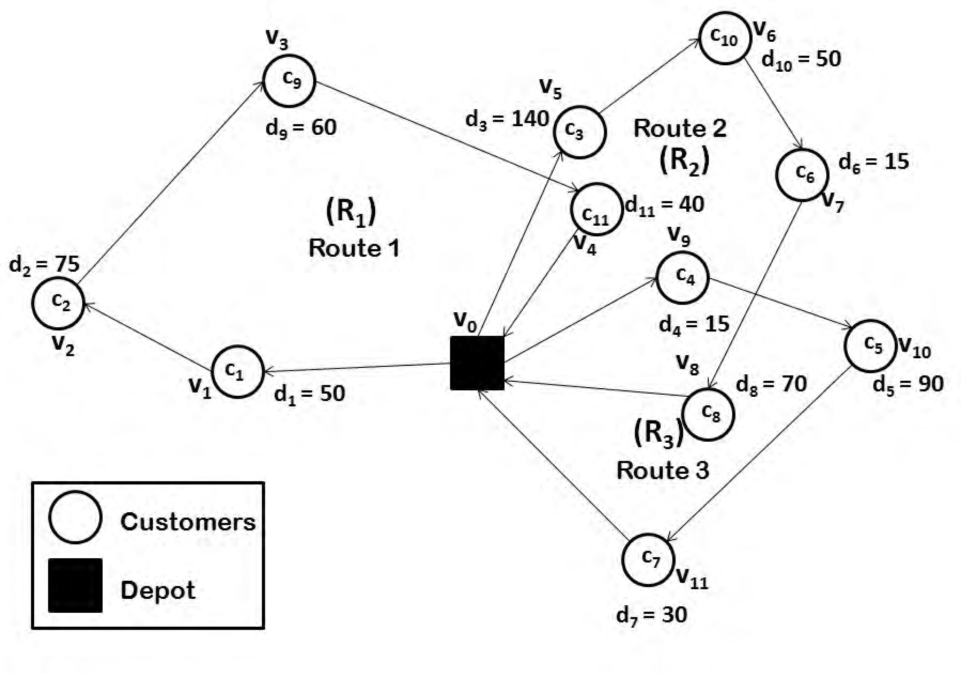


Figure 4.12: Problem Scenario

Figure 4.17.

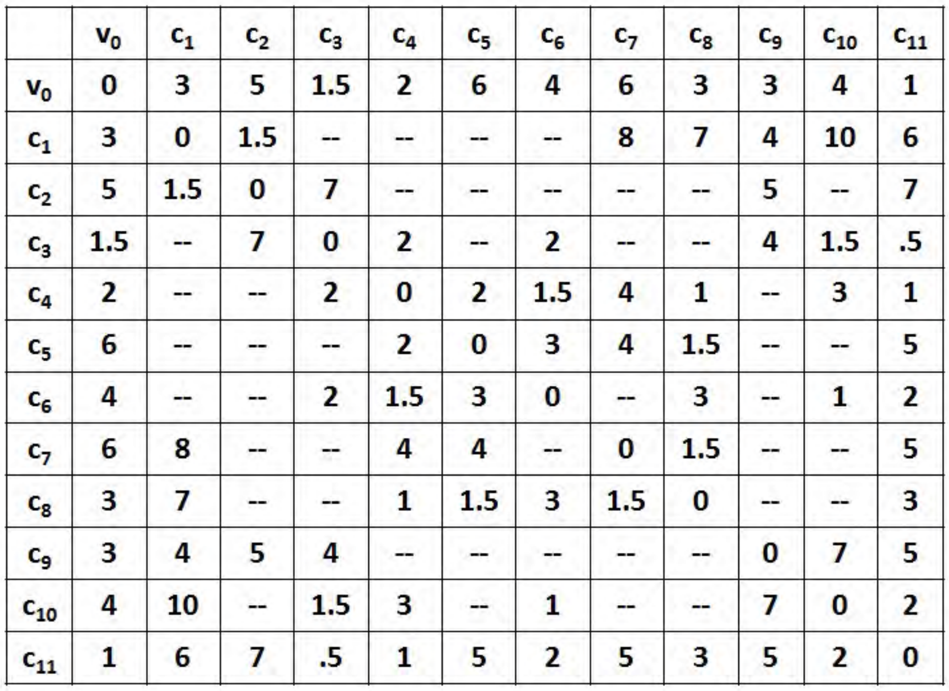
Altogether these two steps generate new neighborhood solutions which is shown in Figure 4.18.

4.2.3 Fitness Evaluation of Food Source

The fitness of a food source, i, is defined as the probability of selecting that food source. It is calculated by the following equation:

$$probability[\text{Select food source } i] = \frac{1/W_i}{\sum_{j=1}^{z} 1/W_j}.$$

Here, W_i (W_j) is the cost (traveling cost) of the food source (solution) i (j) and z is the number of total employed bees.



	v_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
v_0	0	3	5	1.5	2	6	4	6	3	3	4	1
c_1	3	0	1.5	--	--	--	--	8	7	4	10	6
c_2	5	1.5	0	7	--	--	--	--	--	5	--	7
c_3	1.5	--	7	0	2	--	2	--	--	4	1.5	.5
c_4	2	--	--	2	0	2	1.5	4	1	--	3	1
c_5	6	--	--	--	2	0	3	4	1.5	--	--	5
c_6	4	--	--	2	1.5	3	0	--	3	--	1	2
c_7	6	8	--	--	4	4	--	0	1.5	--	--	5
c_8	3	7	--	--	1	1.5	3	1.5	0	--	--	3
c_9	3	4	5	4	--	--	--	--	--	0	7	5
c_{10}	4	10	--	1.5	3	--	1	--	--	7	0	2
c_{11}	1	6	7	.5	1	5	2	5	3	5	2	0

Figure 4.13: Distances among the Customers and the Depot of Figure 4.12

4.2.4 Assigning Food Sources to Employed Bees and Onlooker Bees

At first, the randomly generated initial food sources are assigned to the employed bees. After that, every employed bee explores the neighborhood food sources of that initial solution. Then the best food source among them becomes the final destination of an employed bee.

The fitness information of food sources are made available to the onlooker bees by the employed bees. Depending on this fitness information, onlooker bees choose their food sources. In our *ABC_VRPSTW* algorithm, the number of onlooker bees assigned to a single food source is the fitness of that food source times the total number of onlooker bees in the hive. Clearly, more fitted food source will attract more onlooker bees. After this assignment, the onlooker bees become employed bees and then find their final destinations in the same process as described above.

t_1 (R_1)	c_1	c_2	c_9	c_{11}	**Σdemand = 50 + 75 + 60 + 40 = 225 < 300**	**Constraint Violation = NO**
t_2 (R_2)	c_3	c_{10}	c_6	c_8	**Σdemand = 140 + 50 + 15 + 70 = 275 < 300**	**Constraint Violation = NO**
t_3 (R_3)	c_4	c_5	c_7		**Σdemand = 15 + 90 + 30 = 135 < 300**	**Constraint Violation = NO**

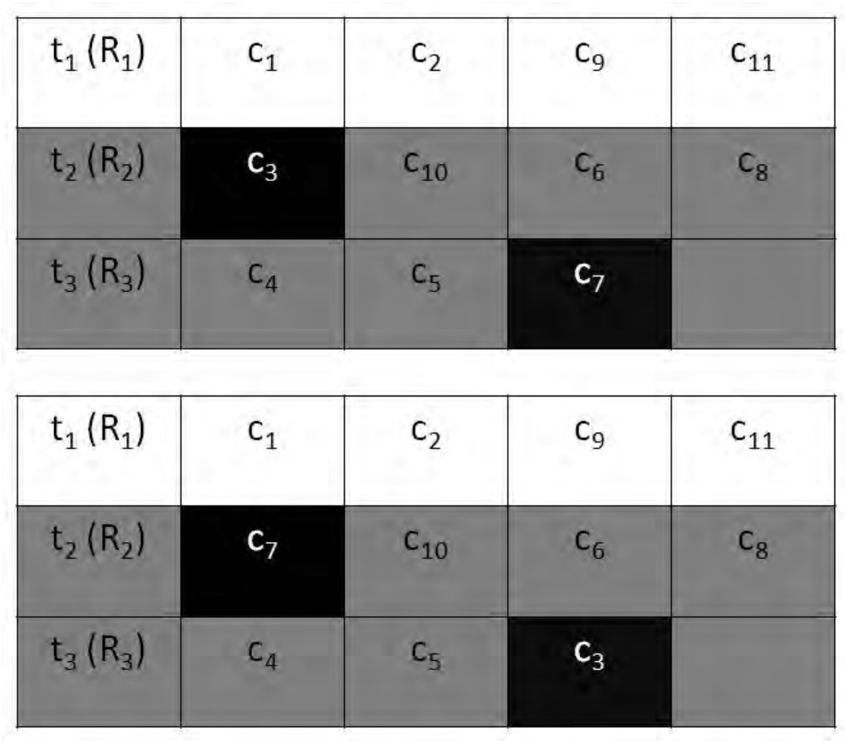
Figure 4.14: An Example Initial Solution - 1 for the Problem Scenario of Figure 4.12

t_1 (R_1)	c_1	c_2	c_9			**Σdemand = 50 + 75 + 60 = 185 < 300**	**Constraint Violation = NO**
t_2 (R_2)	c_3	c_{10}	c_6	c_5	c_{11}	**Σdemand = 140 + 50 + 15 + 90+ 40 = 335 > 300**	**Constraint Violation = YES**
t_3 (R_3)	c_4	c_8	c_7			**Σdemand = 15 + 70 + 30 = 115 < 300**	**Constraint Violation = NO**

Figure 4.15: An Example Initial Solution - 2 for the Problem Scenario of Figure 4.12

4.2.5 Generating Scout Bee and Assigning Food Sources to Scout Bees

In our ABC_VRPSTW algorithm, we try to improve the solution for a fixed number of iterations. But if a solution is not improved within 80% of the total number of iterations, that solution is discarded. Then the empoyed bee assigned to that solution becomes a scout bee. Then for that scout bee, another random initial solution is generated and that scout bee is assigned to that food source. Thus, the scout becomes an employed bee. Then it follows the same procedure described in the previous section.



t_1 (R_1)	c_1	c_2	c_9	c_{11}
t_2 (R_2)	c_3	c_{10}	c_6	c_8
t_3 (R_3)	c_4	c_5	c_7	

t_1 (R_1)	c_1	c_2	c_9	c_{11}
t_2 (R_2)	c_7	c_{10}	c_6	c_8
t_3 (R_3)	c_4	c_5	c_3	

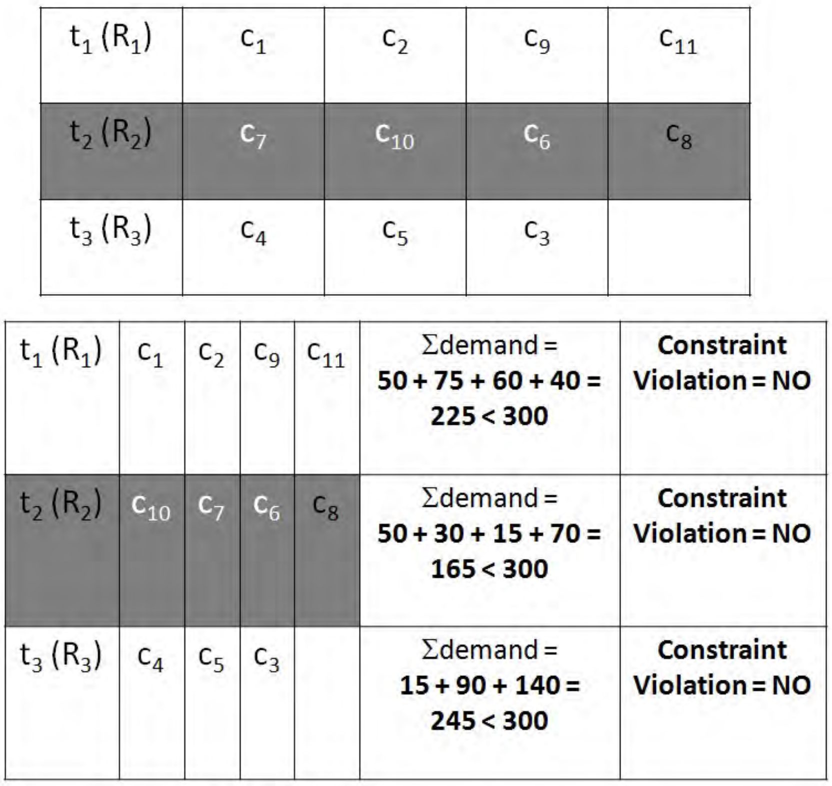
Figure 4.16: Initial Solution after the First Modification Step

4.2.6 Cost Evaluation

The traveling cost, *costij* between every pair of customers c_i and c_j, is the euclidean distance between their positions. So, the traveling cost of each solution is the summation of the cost of all the routes within it, including the penalties (if applicable). The total number of window breaks is the count of the number of customers, which are not served within the un-relaxed time window. While evaluating the cost and selecting/discarding a solution, following constraints are considered. If any of the following constraints are violated, that solution is discarded.

1. **Vehicle Capacity Constraint:** Suppose $c_k \in R_i$ denotes a customer in the route R_i. For any route, R_i, the summation of demands of the customers on that route $(\sum\limits_{c_k \in R_i} d_k)$ must be less than or equal to the capacity of the vehicle (T_q). Otherwise, that solution is then discarded.

2. **Time Window Constraint:** If a vehicle comes at any customer, c_i, within the



t_1 (R_1)	c_1	c_2	c_9	c_{11}
t_2 (R_2)	c_7	c_{10}	c_6	c_8
t_3 (R_3)	c_4	c_5	c_3	

t_1 (R_1)	c_1	c_2	c_9	c_{11}	Σdemand = **50 + 75 + 60 + 40 = 225 < 300**	**Constraint Violation = NO**
t_2 (R_2)	c_{10}	c_7	c_6	c_8	Σdemand = **50 + 30 + 15 + 70 = 165 < 300**	**Constraint Violation = NO**
t_3 (R_3)	c_4	c_5	c_3		Σdemand = **15 + 90 + 140 = 245 < 300**	**Constraint Violation = NO**

Figure 4.17: Initial Solution after the Second Modification Step

interval $[a_i^r \dots a_i]$ or $[b_i \dots b_i^r]$, the customer can be served by the vehicle in exchange of a penalty (due to the relaxation of time window). But if a vehicle comes before a_i^r or after b_i^r, the vehicle is not allowed to serve the customer and that solution is discarded.

4.2.7 Overall Algorithm

Following is a formal description of the ABC_VRPSTW Algorithm:

Step 1 : For each employed Bee,

(1.1) Generate an initial solution and assign the bee to that solution (food source).

(1.2) Generate some neighborhood solutions from the current solution by modifying that solution.

t_1 (R_1)	c_1	c_2	c_9	c_{11}
t_2 (R_2)	c_{10}	c_7	c_6	c_8
t_3 (R_3)	c_4	c_5	c_3	

Figure 4.18: Neighborhood Solution of the Initial Solution in Figure 4.14

(1.3) The cost of the current solution of the employed bee and the neighborhood solutions are evaluated and the best among those are determined.

(1.4) The Employed bee then moves to that best solution. By this way an improved solution is determined.

Step 2 : For the onlooker bees,

(2.1) According to the fitness information of the solution, each onlooker bee chooses a solution for itself.

(2.2) Repeat Steps *1.2* to *1.4*.

Step 3 : If any solution is not improved for a fixed number of iterations,

(4.1) That solution is discarded.

(4.2) The bee which is assigned to that solution becomes a scout bee.

Step 5 : For the scout bee,

(5.1) A new random solution is generated and that scout bee is assigned to it and it becomes an employed bee.

(5.2) Repeat Steps *1.2* to *1.4*.

Step 6 : Repeat Steps *1* to *5* for a predetermined number of iterations to get the best solution. Exit when the predetermined number of iterations are done.

Algorithm 2 and Algorithm 3 represent the overall ABC_VRPSTW algorithm. The algorithms for each step of the formation of ABC_VRPSTW in given in Appendix 9.

Algorithm 2 ABC_VRPSTW (Part - 1)

```
sort the customers in ascending order according to their window starting time (tie
breaks by sorting them in ascending order of their window closing time)
for each Employed Bee $eb$ do
    call $Generate - Random - Initial - Solution$ and assign the solution as
    $EmployedBee(eb)$'s food source
end for
$bestSolution \leftarrow \phi$
$bestCost \leftarrow \infty$
for each employed bee $eb$ do
    $S \leftarrow$ The solution associated with $eb$
    $cost \Leftarrow Cost(S)$
    if $cost < bestCost$ then
        $bestCost \leftarrow cost$
        $bestSolution \leftarrow S$
    end if
end for{to bo continued...}
```

4.3 Time Complexity Analysis of ABC_VRPSTW

Due to the simplicity and modularity of the ABC_VRPSTW algorithm, we can do a time complexity analysis of ABC_VRPSTW as follows. There are four major steps in the ABC_VRPSTW algorithm.

Algorithm 3 ABC_VRPSTW (Part - 2)

```
while true do
  for each employed bee $eb$ do
    $S \leftarrow$ The solution associated with $eb$
    call $Generate-Neighborhood-Solution(S)$ to generate a neighborhood solution
    $(NS)$
    if $Cost(NS) < Cost(S)$ then
      Assign $NS$ to $eb$
    else
      Assign $S$ to $eb$
    end if
  end for
  update $bestSolution$
  {Assign Onlooker Bee}
  for each employed bee $eb$ do
    $S \leftarrow$ Solution associated with $eb$
    $f \leftarrow$ Calculate the fitness of solution S
    Assign $f \times MAX\_ONLOOKER$ number of onlooker bee to $S$
  end for
  {Search with Onlooker Bee}
  if any solution was not updated for last $T$ iterations then
    call $Generate - Random - Initial - Solution$ and replace the old solution with
    the new one
  end if
  if $bestSolution$ is not updated for last $TIME$ iterations then
    break
  end if
end while
```

1. Solution Generation and Assignment of Bees (S_1)

2. Fitness Evaluation (S_2)

3. Modification of Solutions (Generation of Neighborhood Solutions) (S_3)

4. Cost Evaluation (S_4)

In the algorithm, a solution (food source) is generated for each employed bee. The onlookers and scouts also become employed bee when they are assigned to solutions. When a solution is generated, customers are randomly chosen and assigned to the vehicles. A single solution generation is done in such a way that all the customers are chosen and no customers is assigned to more than one vehicle. Assume that, the total number of customers is n. Now, we use a random number generator that produces random number uniformly in the range 0 to $n-1$. If the probability of choosing a customer is p and it takes m steps to choose a unique customer then then, we can say, the expected number of steps needed to a unique customer is $m = \frac{1}{p}$. Therefore the generation of a single solution consists of n customers takes an expected time of,
$O(m \times n) = O(\frac{n}{p}) = O(\frac{n}{n} + \frac{n}{n-1} \ldots + \frac{n}{1})$.
$= O(n(\frac{1}{n} + \frac{1}{n-1} \ldots + \frac{1}{1}))$
$= O(nH_n)$, where H_n is n^{th} harmonic number.
And, the assignment procedure takes $O(1)$ time. So,
$O(S_1) = (O(nH_n) + O(1)) \times (EB + OB + SB)$, where, EB = Number of Employed bee, OB = Number of Onlooker bee and SB = Number of Scout.

Fitness of a solution is evaluated by the equation given in Section 4.2.3. According to that equation,
$O(S_2) = \frac{Cost\_Evaluation\_of\_a\_Single\_Solution}{Cost\_Evaluation\_of\_a\_Single\_Solution \times (EB+OB+SB)}$
$= \frac{O(nH_n)}{O(nH_n) \times (EB+OB+SB)} \times (EB + OB + SB) = O(1)$.

In the Modification procedure given in Section 4.2.2 (while generating Neighborhood Solutions), randomly chosen solutions are modified in two steps. In first step, at first two routes are chosen randomly from the solution. Then, from each of randomly chosen route, one customer is chosen randomly. Since the number of route and customer to be chosen in this step is less than n, we can say the expected time needed for this step is less than $O(S_1)$. In second step of modification, a route is first chosen randomly from the solution to be modified and then a block of customers server in that route is selected randomly. Here, again the number of route and customer to be chosen is less than n. So, here the expected time needed for this step is less than $O(S_1)$. Since the exchange operation takes constant time, $O(S_3) = (O(S_1) + O(1)) \times (EB + OB + SB)$.

In the cost evaluation process a solution examines each customer in $O(1)$ time which yields a total of $O(n)$ running time. So, $O(S_4) = O(n) \times (EB + OB + SB)$.

So, the expected running time of the ABC_VRPSTW algorithm,
Total number of iteration $\times$ $(O(S_1)$ + $O(S_2)$ + $O(S_3)$ + $O(S_4))$
= Total number of iteration $\times(O(nH_n) + O(1)) \times (EB + OB + SB)$
= $O($Total number of iteration $\times$ $nH_n(EB + OB + SB))$.

4.4 ABC_VRPSTW and Some Other Variants of VRP and VRPTW

The flexibility and generality of our ABC_VRPSTW algorithm turn out to be very useful and important in real-world applications. Our ABC_VRPSTW algorithm can be easily changed to solve some other variants of VRP and VRPTW problems as briefly identified below.

Vehicle Routing Problem with Hard Time Window **(VRPHTW)** VRPHTW is a well studied variant of Vehicle Routing Problem with Time Window. Here, the customers to be served by the vehicle have fixed time windows and they must be served within that time windows. Here, window violation is not allowed in any condition. Our ABC_VRPSTW problem can be applied to solve VRPHTW problem. Two changes are to be done in our algorithm to apply it to solve VRPHTW problem.

At first, in the solution generation process, vehicles which come to a customer before and after the time window is not allowed to serve, so we cannot take that customer within the route of that vehicle. This change is to be made in Algorithm 4 given in appendix 9. And the second change is to be made in cost evaluation procedure. Here, no penalty term is to be introduced and added to the cost. This change is to be made in Algorithm 7 given in appendix 9.

Heterogeneous Fleet Vehicle Routing Problem **(HVRP)** HVRP is a variant Vehicle Routing Problem. Here, the customers have no time window and the vehicles of the fleet can have different capacity. This problem can also be solved by our ABC_VRPSTW Algorithm. For that, three changes are to be made, two are within our algorithm and other is within the input problem instances. While generating solution, for this problem, any vehicle can serve a customer at any time if the vehicle has enough capacity to serve that customer's demand. So, in solution generation process (Algorithm 4), no checking is to be done and in the cost evaluation process (Algorithm 7), only demand constraint is to be checked and no penalty term is needed. For our algorithm, all the vehicles have same capacity which is ensured by the input problem instances. But, here the input problem instances should contain different capacity for different vehicles.

Fleet Size and Mix Vehicle Routing Problem **(FSMVRP) [33]** Fleet Size and Mix Vehicle Routing Problem is a variation of the Vehicle Routing Problem (VRP). The VRP is characterized by a fixed or variable number of vehicles, common vehicles capacities, and minimization of the total distance traveled by all vehicles as the objective. The FSMVRP generalizes the VRP by including a vehicle fleet composition decision, using a number of heterogeneous vehicle types. Each vehicle type is characterized by its capacity, fixed cost and variable travel cost. Each route starts and finishes at the depot and the objective is to minimize the cost of servicing all customers. This cost is found from the sum of all fixed costs, plus the variable cost of the distance traveled by each vehicle. Our ABC_VRPSTW problem can be applied to this problem by modifying the cost evaluation process (Algorithm 7) in a way so that the cost is calculated taking the sum of all fixed costs and the variable cost of the distance traveled by each vehicle. And, also the time window constraint is to be removed from solution generation process (Algorithm 4).

Fleet Size and Mix Vehicle Routing Problem with Time Windows **(FSMVRPTW)** FSMVRPTW is a variation of FSMVRP where the time window constraint is added. Our algorithm already supports this time window constraint of the customers. So the only modification to be done in our algorithm for using it for FSMVRPTW is same as the previous FSMVRP problem.

Time Dependent Vehicle Routing Problem **(TDVRP)** [34] TDVRP is another variation of VRP that can be solved by our algorithm. In this variation, the customers are also moving and have a constant speed. The speed of the customers should be given in the input problem instances. Then while generating solution (Algorithm 4) and adding a customer into the route of a vehicle, the vehicle will need to identify the customers current location first and then will be able to serve.

Multi-Commodity VRP [35] In this variation of VRP, the customers have multiple demand instead of just one type of object. So, the vehicle must carry all the type of objects that a customer may want. Here, when a vehicle serves a customer, it first need to identify whether it has that specific types of objects that the customer needs. And then the capacity of the vehicle must be deducted carefully so that only the quantity of the objects that is given to a customer, should be deducted. This checking can be done while generating the solution (Algorithm 4) in our ABC_VRPSTW algorithm.

Capacitated Vehicle Routing Problem (with or without Time Windows) [73] Another variant of VRP and VRPTW is Capacitated Vehicle Routing Problem with or without Time Windows (CVRP or CVRPTW). Here, the constraint is no customer can have more demand than the capacity of a vehicle. Since the time window and demand constraints are already handled in our ABC_VRPSTW algorithm, it can be easily used to solve this variation of VRP and VRPTW.

Vehicle Routing Problem with Pickup and Delivery (VRPPD) [**74**] In this variation of VRP, objects need to be moved from certain pickup locations to other delivery locations. Customers are served from one depot, and on customer-side goods needs to be picked up and delivered. And, the goal is to find optimal routes for a fleet of vehicles to visit the pickup and drop-off locations. So here, at some locations capacity of the vehicle increases (in case of pick-up) and at some locations, capacity of the vehicle decreases (in case of delivery). This change can be done in Algorithm 4 in our ABC_VRPSTW procedure to solve this variation of VRP.

Vehicle Routing Problem with LIFO This is a variation of VRPPD problem with an additional constraint on the placement of loading of the vehicles, that is, at any delivery location, the item being delivered must be the item most recently picked up. This scheme reduces the loading and unloading times at delivery locations because there is no need to temporarily unload items other than the ones that should be dropped off. This additional restriction can also be added in Algorithm 4 in our ABC_VRPSTW procedure to solve this variation of VRPPD.

Chapter 5

Experimental Results

In this chapter, we assess the performance of our ABC_VRPSTW algorithm. The dataset we use are the benchmark dataset from [72]. Here, we also report the computational time need for our algorithm.

5.1 Experimental Platform

We have implemented our proposed algorithm in *C/C++*. We have used *Visual C++* compiler on *Windows 7 operating system* running on a machine having *2.0 GHz Intel Core 2 Duo* processor with 3GB RAM.

5.2 Problem Instances

Following the established practice in the literature, we have used instances from the well known Solomon's VRPTW benchmark problems [72]. The 56 Solomon benchmark problems are divided into six groups (C1, R1, RC1, C2, R2, RC2) of problem instances. They have been designed to put in evidence different factors which can make VRPTW instances more or less difficult. The customers' positions are generated at random in Groups R1 and R2, they are clustered in Groups C1 and C2, and they are hybrid in

Groups RC1 and RC2. Each data instance contains the number and capacity of vehicles and the position (coordinate), demand, time window (starting and ending time of the window), service time of all the customers.

5.3 Experimental Plans

We have set up different experimentation plans. These plans vary on a number of input parameters. These input parameters include, Number of Employed Bees, Number of Onlooker bees and the Number of iterations as the termination condition. The different parameters are spelled out in Table 5.1.

	Employed Bees	Onlooker Bees	Iterations
Plan 1	50	10	75
Plan 2	200	60	400
Plan 3	600	200	1000

Table 5.1: Parameters of Experimentation Plans

5.4 Results

For each of the 56 problem instances, we have run ABC_VRPSTW algorithm 30 times for all Plan 1, 2 and 3. We calculated the average (*avg*), minimum (*min/best result*), maximum (*max/worst result*) and median (*med/central tendency*) of the 30 runs for the three plans on each of the six types (R1, R2, RC1, RC2, C1 and C2) of problem instances. We report the best result found for 56 problem instances and three plans in Table 5.2 through Table 5.7. And the detail results including the best, worst, average and median is given in Table 8.1 through Table 8.8 in Appendix 8. In each table, we present the Total Traveling Cost (TTC), Percentage of Window Breaks (%WB), and Number of Vehicles (V).

Table 5.2: Best Result of ABC_VRPSTW for Problem Instances R1 **(Number of Customer = 200)**

Problem	**Plan: 1**		**Plan: 2**		**Plan: 3**	
Instances	TTC	%WB	TTC	%WB	TTC	%WB
R1_01 (V = 19)	644.338	25	643.844	26	636.789	25
R1_02 (V = 17)	703.958	26	643.844	26	694.13	26
R1_03 (V = 13)	708.006	17	699.975	14	691.056	14
R1_04 (V = 9)	674.069	13	661.496	15	647.448	11
R1_05 (V = 14)	694.533	28	678.493	30	657.779	29
R1_06 (V = 12)	696.176	25	677.948	20	668.798	23
R1_07 (V = 10)	671.102	19	660.319	16	651.059	17
R1_08 (V = 9)	670.514	13	652.587	14	638.095	11
R1_09 (V = 11)	687.587	25	670.828	26	659.515	26
R1_10 (V = 10)	708.006	17	699.975	14	691.056	14
R1_11 (V = 10)	672.439	22	658.587	21	637.658	21
R1_12 (V = 9)	708.006	17	649.683	13	644.144	12

In Figure 5.2, We reported the best result, Total Traveling Cost (TTC) and Percentage of Window Break (%WB), for R1 problem instances. In the R1 problem instances, the geographical data are randomly generated. Here, problem sets have a short scheduling horizon and only a few customers are allowed per route.

In Figure 5.3, We reported the best result, Total Traveling Cost (TTC) and Percentage of Window Break (%WB), for R2 problem instances. In the R2 problem instances, the

Table 5.3: Best Result of ABC_VRPSTW for Problem Instances R2 **(Number of Customer = 1000)**

Problem	**Plan: 1**		**Plan: 2**		**Plan: 3**	
Instances	TTC	%WB	TTC	%WB	TTC	%WB
R2_01 (V = 4)	910.716	6.9	914.716	6.9	886.698	7.0
R2_02 (V = 3)	842.053	5.0	861.790	5.8	823.261	5.7
R2_03 (V = 3)	777.352	4.0	769.420	4.2	828.751	5.0
R2_04 (V = 2)	759.558	3.7	773.662	3.4	773.662	3.4
R2_05 (V = 3)	889.370	6.3	916.573	7.3	941.128	7.1
R2_06 (V = 3)	782.620	4.0	821.217	5.4	826.995	5.2
R2_07 (V = 2)	822.401	4.8	822.401	4.8	822.401	4.8
R2_08 (V = 2)	722.845	2.7	743.445	2.8	743.445	2.8
R2_09 (V = 3)	806.453	5.1	808.712	5.1	811.718	5.7
R2_10 (V = 3)	792.310	5.8	798.339	4.7	798.339	4.7
R2_11 (V = 2)	671.890	7.8	671.890	7.8	671.890	7.8

geographical data are randomly generated. But, problem sets have a long scheduling horizon permitting many customers to be serviced by a same vehicle.

In Figure 5.4, We reported the best result, Total Traveling Cost (TTC) and Percentage of Window Break (%WB), for RC1 problem instances. In the RC1 problem instances, the geographical data are mix of random and clustered structures. Here, problem sets have a short scheduling horizon and only a few customers are allowed per route.

In Figure 5.5, We reported the best result, Total Traveling Cost (TTC) and Percentage of

Table 5.4: Best Result of ABC_VRPSTW for Problem Instances RC1 **(Number of Customer = 200)**

Problem Instances	**Plan: 1** TTC	%WB	**Plan: 2** TTC	%WB	**Plan: 3** TTC	%WB
RC1_01 (V = 14)	1171.831	25.5	1152.456	24	1120.158	22
RC1_02 (V = 12)	1238.334	18	1195.790	18	1163.283	18
RC1_03 (V = 11)	1174.398	15	1135.420	16	1109.849	16.5
RC1_04 (V = 10)	1219.007	9.5	1192.662	7.5	1061.688	6.5
RC1_05 (V = 13)	1296.57	20	1235.763	20.5	1197.953	21.5
RC1_06 (V = 11)	1130.176	21.5	1105.217	22	1082.742	22.5
RC1_07 (V = 10)	1238.584	15.5	1190.401	14	1117.909	11.5
RC1_08 (V = 10)	1178.61	8.5	1113.445	7.5	1070.015	7.5

Window Break (%WB), for RC2 problem instances. In the RC2 problem instances, the geographical data are mix of random and clustered structures. But, problem sets have a long scheduling horizon permitting many customers to be serviced by a same vehicle.

In Figure 5.6, We reported the best result, Total Traveling Cost (TTC) and Percentage of Window Break (%WB), for C1 problem instances. In the C1 problem instances, the geographical data are clustered. Here, problem sets have a short scheduling horizon and only a few customers are allowed per route.

In Figure 5.7, We reported the best result, Total Traveling Cost (TTC) and Percentage of Window Break (%WB), for C2 problem instances. In the C2 problem instances, the geographical data are clustered. But, problem sets have a long scheduling horizon permitting many customers to be serviced by a same vehicle.

Table 5.5: Best Result of ABC_VRPSTW for Problem Instances RC2 **(Number of Customer = 1000)**

Problem Instances	**Plan: 1** TTC	%WB	**Plan: 2** TTC	%WB	**Plan: 3** TTC	%WB
RC2_01 (V = 4)	912.361	5.4	889.566	5.2	900.566	5.9
RC2_02 (V = 3)	970.613	5.4	1067.804	7.1	1063.283	7.5
RC2_03 (V = 3)	956.547	4.2	833.043	6.8	833.043	6.8
RC2_04 (V = 4)	793.5	3.0	802.45	4.5	778.237	5.4
RC2_05 (V = 4)	829.757	4.9	858.511	6.0	807.953	5.7
RC2_06 (V = 3)	870.495	5.5	905.217	6.3	1008.137	6.5
RC2_07 (V = 3)	777.993	4.3	776.401	4.5	657.909	4.1
RC2_08 (V = 3)	664.573	1.8	772.208	5.5	690.015	4.5

Metaheuristic always designates a computational method that optimizes a problem by iteratively trying to improve a solution with regard to a given measure of quality. Artificial bee colony algorithm also refine the solution in multiple cycles. In each experimental plan defined in Table 5.1 for our ABC_VRPSTW algorithm, we have set multiple number of iterations and our algorithm also improves the traveling cost as the iterations progress. This flow of improving the traveling cost as the iteration progresses is shown for each plan and for the six type of benchmark data instances (R1, R2, RC1, RC2, C1, C2) in Table 5.8 through Table 5.13. This analysis is also depicted in Figure 5.1, Figure 5.2 and Figure 5.3.

We present the best results of ABC_VRPSTW algorithm *(in terms of Traveling Cost)* for all three plans in Table 5.14 in case of each of the six types of problem instances.

A quick analysis of Table 5.14 reveals how the input parameters of different plans in Table 5.1 affects the output of ABC_VRPSTW algorithm. Clearly with the increase of

Table 5.6: Best Result of ABC_VRPSTW for Problem Instances C1 **(Number of Customer = 200)**

Problem	**Plan: 1**		**Plan: 2**		**Plan: 3**	
Instances	TTC	%WB	TTC	%WB	TTC	%WB
C1_01 (V = 10)	869.803	33.5	830.298	32	791.508	32.5
C1_02 (V = 10)	800.949	24	767.804	23	736.759	25.5
C1_03 (V = 10)	612.522	17.5	599.18	17	589.187	18
C1_04 (V = 10)	850.532	8	702.45	7	628.310	6.5
C1_05 (V = 10)	909.222	33	858.511	32	835.119	32.5
C1_06 (V = 10)	733.590	28	705.217	28.5	689.941	31
C1_07 (V = 10)	905.613	27.5	876.401	28	844.600	27
C1_08 (V = 10)	876.692	23	772.208	23	779.385	24
C1_09 (V = 10)	670.832	16.5	690.386	18.5	773.385	24

Table 5.7: Best Result of ABC_VRPSTW for Problem Instances C2 **(Number of Customer = 700)**

Problem	**Plan: 1**		**Plan: 2**		**Plan: 3**	
Instances	TTC	%WB	TTC	%WB	TTC	%WB
C2_01 (V = 3)	554.971	11.7	547.566	11.7	541.204	12
C2_02 (V = 3)	541.204	9.7	541.204	9.7	541.204	9.7
C2_03 (V = 3)	549.445	5.4	542.201	6.2	541.204	6.2
C2_04 (V = 4)	551.445	3.4	541.204	3.4	541.204	3.4
C2_05 (V = 4)	547.374	10.8	541.204	11.4	541.204	11.4
C2_06 (V = 3)	556.673	10.7	541.204	11.1	541.204	11.1
C2_07 (V = 3)	543.275	10.5	541.204	9.4	541.204	9.4
C2_08 (V = 3)	559.006	9.7	559.006	9.7	559.006	9.7

number of bees and iterations, the results become better almost in all the cases. Figure 5.4 summarizes the result and Figure 5.5 shows the percentage of improvement.

Sine, the VRPSTW is an offline problem, computational time required by ABC_VRPSTW is not that much important provided that a reasonable limit is there. Here, we report the computational time required by ABC_VRPSTW algorithm to give the best result for the three plans in case of each of the six types of problem instances. We present this computational time in Table 5.15 and by Figure 5.6. Figure 5.7 shows the percentage of increase of time among the plans needed to give the best result. This analysis shows that as the parameters of the plan increases, the result improves (traveling cost decreases) but, naturally, the computation time increases. Notably for all the plans the best results have been achieved within a reasonable amount of time. For example, in worst case, the best result is achieved within less than 4/5 minutes which is perfectly reasonable for an offline problem like VRPSTW.

Table 5.8: Improvement Sequence of Traveling Cost for R1 Problem Instance

Plan: 1 Employed Bee: 50 Onlooker Bee: 10 Iterations: 75		Plan: 2 Employed Bee: 200 Onlooker Bee: 60 Iterations: 400		Plan: 3 Employed Bee: 600 Onlooker Bee: 200 Iterations: 1000	
Iteration Number	Traveling cost	Iteration Number	Traveling cost	Iteration Number	Traveling cost
0	776.623	0	776.623	0	776.623
1	719.231	1	719.231	1	719.231
2	712.826	2	712.826	2	712.826
4	672.846	4	672.846	4	672.846
38	641.388	38	641.388	38	641.388
		86	635.056	86	635.056
		103	634.750	399	611.831
		110	634.266	518	609.399
		196	630.439	644	606.350
		241	616.174	758	599.059
		278	612.499	824	586.161
		326	596.626	900	585.201
		361	594.390	972	573.365

Table 5.9: Improvement Sequence of Traveling Cost for R2 Problem Instance

Plan: 1 Employed Bee: 50 Onlooker Bee: 10 Iterations: 75		Plan: 2 Employed Bee: 200 Onlooker Bee: 60 Iterations: 400		Plan: 3 Employed Bee: 600 Onlooker Bee: 200 Iterations: 1000	
Iteration Number	Traveling cost	Iteration Number	Traveling cost	Iteration Number	Traveling cost
0	759.302	0	759.302	0	759.302
3	732.711	3	732.711	3	732.711
9	726.875	9	726.875	9	726.875
18	713.933	18	713.933	18	713.933
26	698.390	26	698.390	26	698.390
		163	692.139	163	692.139
		266	634.750	413	683.086
		317	634.266	587	682.829
		393	630.439	702	660.688

Table 5.10: Improvement Sequence of Traveling Cost for RC1 Problem Instance

Plan: 1		Plan: 2		Plan: 3	
Employed Bee: 50		Employed Bee: 200		Employed Bee: 600	
Onlooker Bee: 10		Onlooker Bee: 60		Onlooker Bee: 200	
Iterations: 75		Iterations: 400		Iterations: 1000	
Iteration Number	Traveling cost	Iteration Number	Traveling cost	Iteration Number	Traveling cost
3	1010.977	3	1010.977	3	1010.977
38	981.397	58	998.798	58	998.798
68	960.876	106	994.315	328	964.881
		212	985.399	491	954.419
		378	954.248	602	944.495
				756	692.139
				863	932.568
				905	985.404
				913	972.0o82

Table 5.11: Improvement Sequence of Traveling Cost for RC2 Problem Instance

Plan: 1		Plan: 2		Plan: 3	
Employed Bee: 50		Employed Bee: 200		Employed Bee: 600	
Onlooker Bee: 10		Onlooker Bee: 60		Onlooker Bee: 200	
Iterations: 75		Iterations: 400		Iterations: 1000	
Iteration Number	Traveling cost	Iteration Number	Traveling cost	Iteration Number	Traveling cost
0	785.314	0	785.314	0	785.314
1	765.808	1	765.808	1	765.808
5	726.477	5	726.477	5	726.477
18	721.990	18	721.990	18	721.990
31	696.034	31	696.034	31	696.034
51	685.734	103	683.544	103	683.544
63	655.891	170	656.427	170	656.427
		282	633.158	351	649.061
		374	629.228	378	625.553
				506	623.802
				626	617.342
				769	610.819
				984	609.454

Table 5.12: Improvement Sequence of Traveling Cost for C1 Problem Instance

Plan: 1 Employed Bee: 50 Onlooker Bee: 10 Iterations: 75		Plan: 2 Employed Bee: 200 Onlooker Bee: 60 Iterations: 400		Plan: 3 Employed Bee: 600 Onlooker Bee: 200 Iterations: 1000	
Iteration Number	Traveling cost	Iteration Number	Traveling cost	Iteration Number	Traveling cost
0	1512.428	0	1310.840	0	1590.457
22	1276.971	101	1043.404	25	1324.913
41	609.443	167	927.165	79	1269.401
		267	903.473	417	606.063
		311	592.004	688	567.047

Table 5.13: Improvement Sequence of Traveling Cost for C2 Problem Instance

Plan: 1 Employed Bee: 50 Onlooker Bee: 10 Iterations: 75		Plan: 2 Employed Bee: 200 Onlooker Bee: 60 Iterations: 400		Plan: 3 Employed Bee: 600 Onlooker Bee: 200 Iterations: 1000	
Iteration Number	Traveling cost	Iteration Number	Traveling cost	Iteration Number	Traveling cost
0	770.577	0	770.577	0	770.577
1	713.390	1	713.390	1	713.390
3	653.350	3	653.350	3	653.350
10	608.705	10	608.705	10	608.705
22	595.348	56	601.190	56	601.190
59	587.100	105	587.225	245	600.465
		158	585.694	316	591.126
		192	580.007	391	590.837
		224	577.782	424	579.487
		242	575.375	555	565.477
		317	573.493	618	565.230
		328	567.558	711	563.993
		342	561.495	838	563.277
				896	558.662
				912	557.559
				932	557.163
				944	550.723

Table 5.14: Best Result of ABC_VRPSTW for All Problem Instances **(in terms of Traveling cost)**

Problem Instances	**Plan: 1** Employed Bee: 50 Onlooker Bee: 10 Iterations: 75	**Plan: 2** Employed Bee: 200 Onlooker Bee: 60 Iterations: 400	**Plan: 3** Employed Bee: 600 Onlooker Bee: 200 Iterations: 1000
R1 (V = 19)	TTC = 641.388	TTC = 594.390	TTC = 573.365
R2 (V = 3)	TTC = 698.390	TTC = 671.850	TTC = 660.688
RC1 (V = 11)	TTC = 960.876	TTC = 954.248	TTC = 910.394
RC2 (V = 3)	TTC = 655.891	TTC = 629.288	TTC = 609.454
C1 (V = 10)	TTC = 609.443	TTC = 592.004	TTC = 567.047
C2 (V = 4)	TTC = 587.100	TTC = 561.495	TTC = 550.723

Table 5.15: Computational time (sec) of ABC_VRPSTW for All Problem Instances

Problem Instances	**Plan: 0** Employed Bee: 50 Onlooker Bee: 10 Iterations: 75	**Plan: 2** Employed Bee: 200 Onlooker Bee: 60 Iterations: 400	**Plan: 4** Employed Bee: 600 Onlooker Bee: 200 Iterations: 1000
R1 (V = 19)	time = 12.552	time = 49.978	time = 216.14
R2 (V = 3)	time = 11.395	time = 44.873	time = 205.452
RC1 (V = 11)	time = 16.414	time = 63.955	time = 131.144
RC2 (V = 3)	time = 12.796	time = 40.796	time = 153.399
C1 (V = 10)	time = 17.055	time = 27.279	time = 214.352
C2 (V = 4)	time = 10.406	time = 32.380	time = 144.842

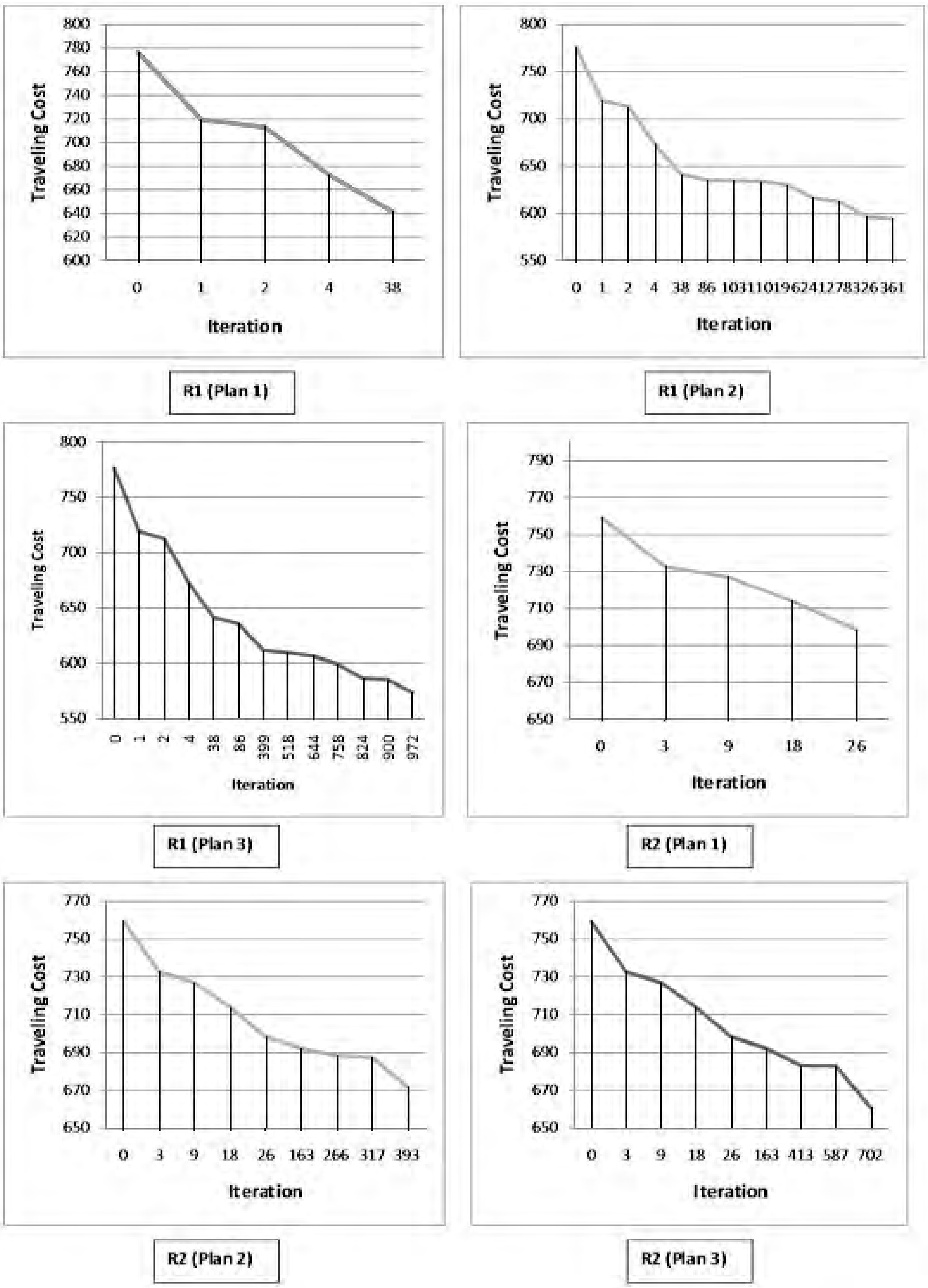


Figure 5.1: Improvement of Traveling Cost with Iteration (R1 and R2 Problem Instance)

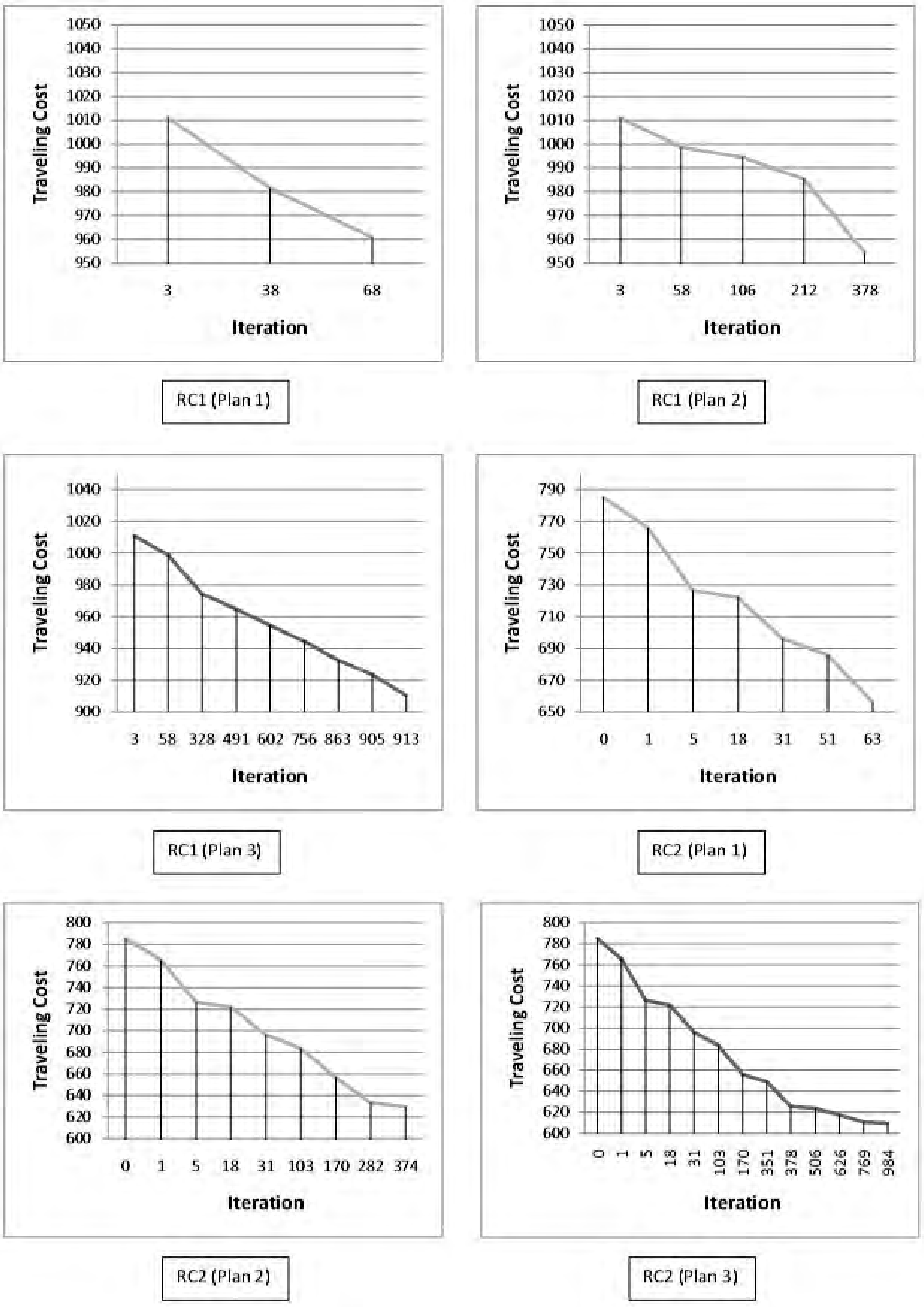


Figure 5.2: Improvement of Traveling Cost with Iteration (RC1 and RC2 Problem Instance)

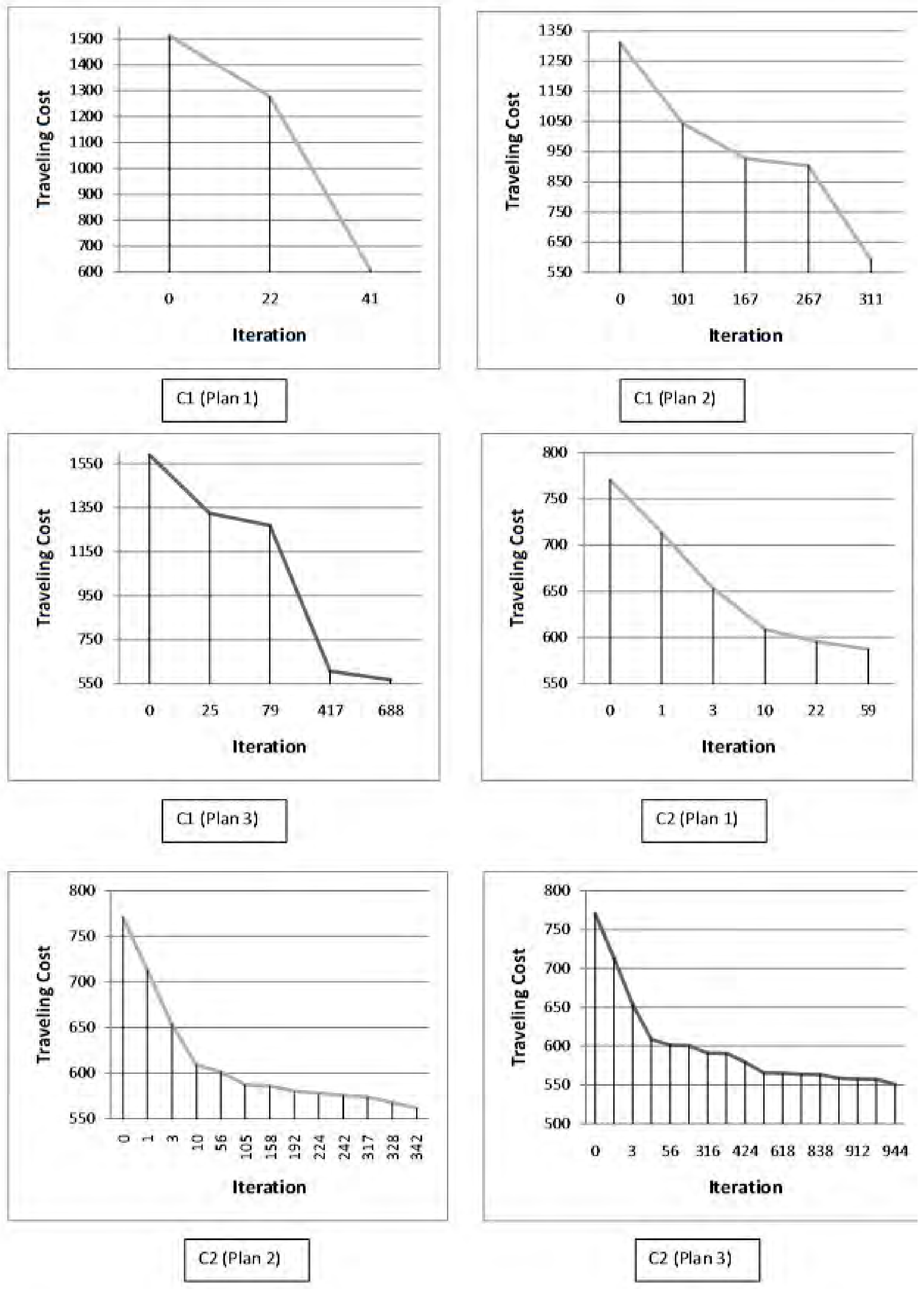


Figure 5.3: Improvement of Traveling Cost with Iteration (C1 and C2 Problem Instance)

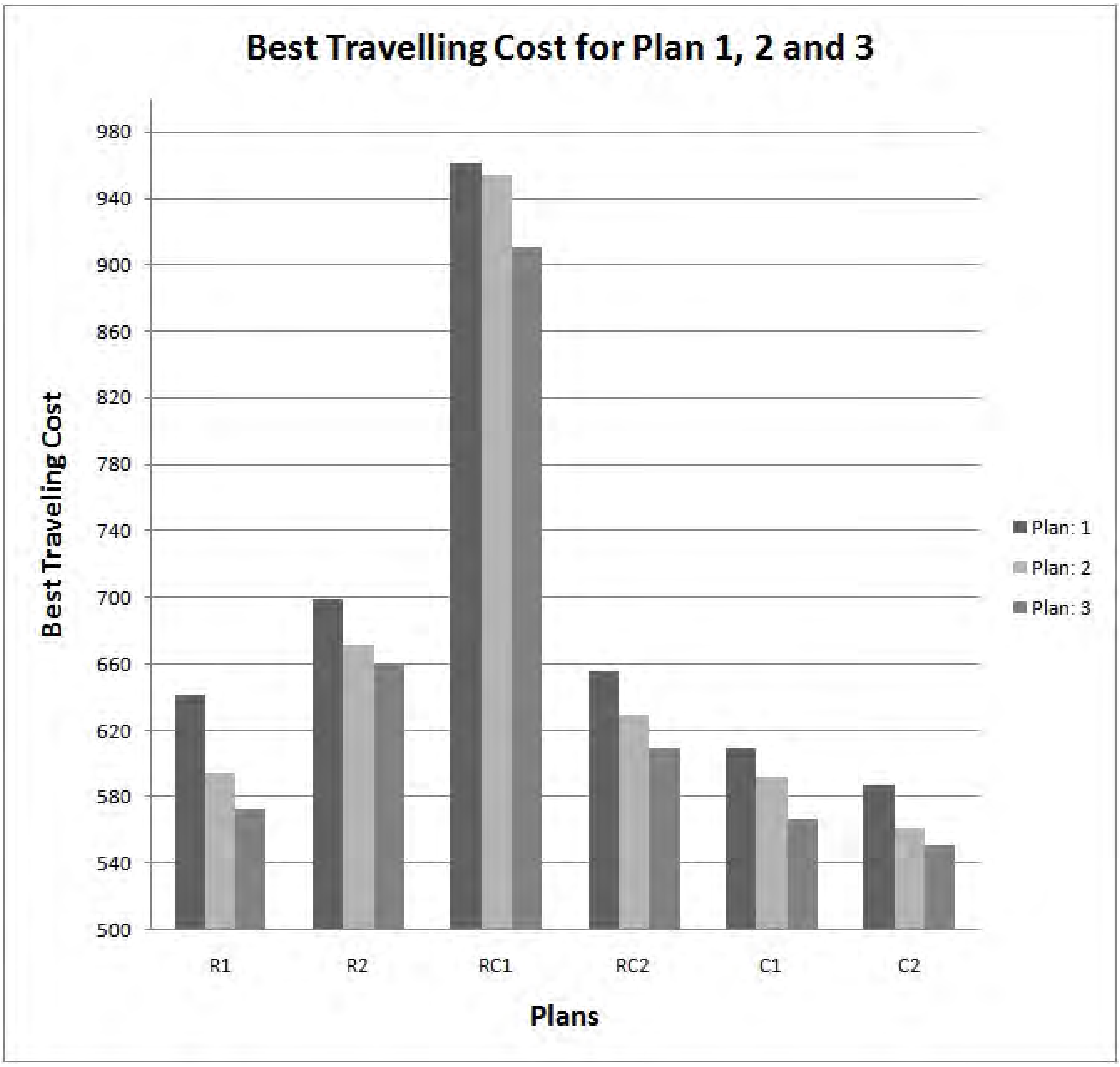
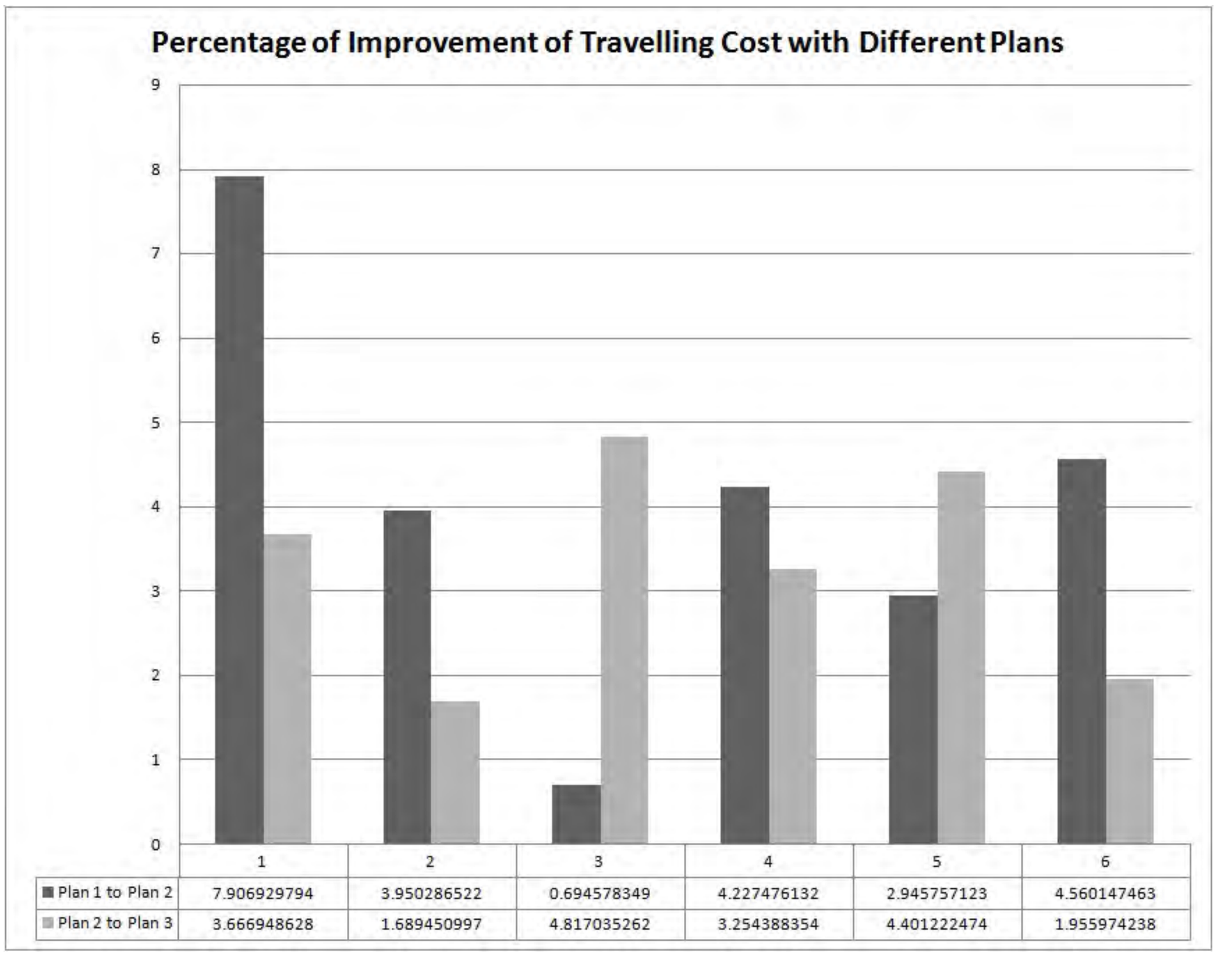
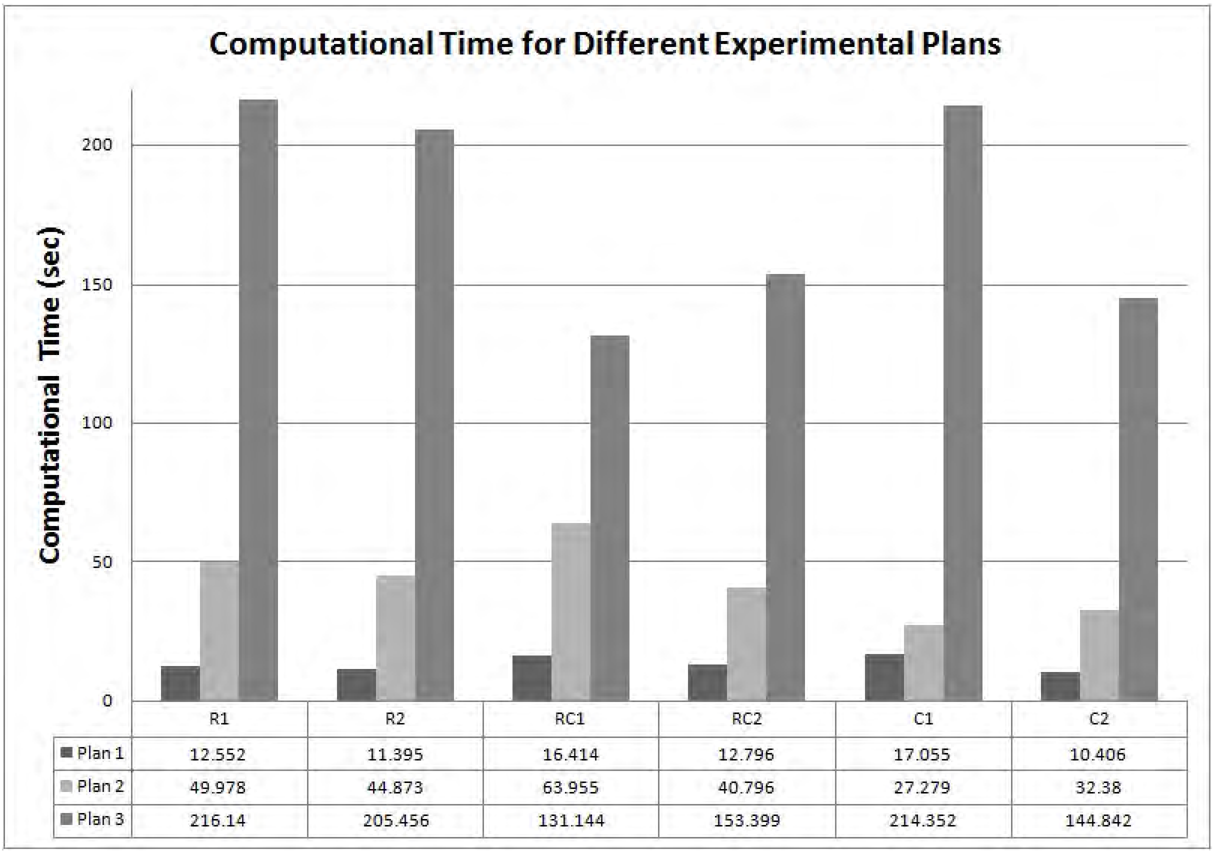


Figure 5.4: Variation of Results with Different Experimental Plans



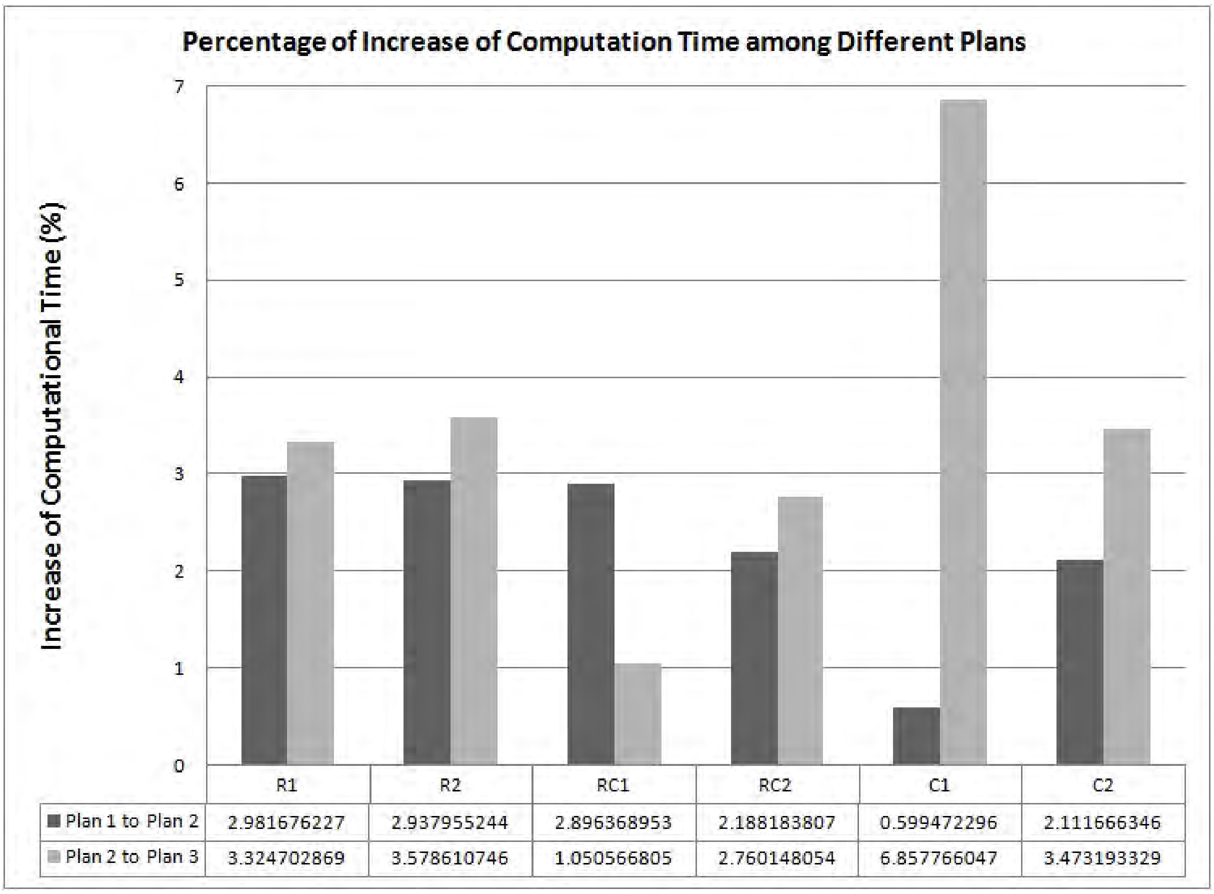
	1	2	3	4	5	6
■ Plan 1 to Plan 2	7.906929794	3.950286522	0.694578349	4.227476132	2.945757123	4.560147463
■ Plan 2 to Plan 3	3.666948628	1.689450997	4.817035262	3.254388354	4.401222474	1.955974238

Figure 5.5: Percentage of Improvements of Results with Different Experimental Plans



	R1	R2	RC1	RC2	C1	C2
■ Plan 1	12.552	11.395	16.414	12.796	17.055	10.406
■ Plan 2	49.978	44.873	63.955	40.796	27.279	32.38
■ Plan 3	216.14	205.456	131.144	153.399	214.352	144.842

Figure 5.6: Computational Time for Different Experimental Plans



	R1	R2	RC1	RC2	C1	C2
■ Plan 1 to Plan 2	2.981676227	2.937955244	2.896368953	2.188183807	0.599472296	2.111666346
■ Plan 2 to Plan 3	3.324702869	3.578610746	1.050566805	2.760148054	6.857766047	3.473193329

Figure 5.7: Percentage of Increase of Computation Time among Different Plans

Chapter 6

Comparison with Previous Results

This chapter presents the detail comparison of the results of the ABC_VRPSTW algorithm with other state of the art methods in the literature that report solution quality and computational time on benchmark problems for the VRPSTW. To the best of our knowledge, the only four publications that present results for VRPSTW benchmark instances are:

1. The work by Balakrishnan [24] (denoted by ***BAL***).
2. The two solution methods, Tabu Search and Advance Recovery, respectively, proposed by Chiang and Russell [2] (denoted by ***TS*** and ***AR***).
3. The Unified Tabu Search method proposed by Fu *et al.* [31] (denoted by ***UTS***) and
4. Two versions, depending on the number of iterations, of an iterative route construction solution proposed by M. A. Figliozzi [27] (denoted by ***IRCIs*** and ***IRCIe***).

Balakrishnan [24] worked on a subset of Solomon problems setting a P_{max} (the allowable limit for time window violation) that can be either 10%, 5%, or 0% of the total route duration $[a_0, b_0]$. Here, all the four works mentioned above also set a maximum vehicle waiting time limit, W_{max}. In particular, W_{max} limits the time that a vehicle can wait at a customer location before starting the service. So, a vehicle can arrive to Customer c_i only after $(a_i^r - W_{max})$ and waiting at a customer after the service is not allowed. It is argued by Figliozzi [27] that, since VRPSTW is a relaxation of VRPHTW, a new constraint limiting maximum waiting time is clearly opposite to the spirit of it. Despite the above argument a W_{max} of 10% constraint has been considered by Figliozzi [27] and also in our algorithm, ABC_VRPSTW, mainly to facilitate comparisons on a level playing field.

Table 6.1 and Table 6.2 present the comparisons for some R1 and RC1 instances (which are found in the literature) with $P_{max} = 5\%$ and $W_{max} = 10\%$ respectively. Following the strategy of [27], the penalty cofficients are set to 1, i.e., a unit of time of time window violation is assumed to be equivalent to a unit of distance traveled. ABC_VRPSTW_{best} indicates the best (optimized) result found from Plan 1, Plan 2 or Plan 3 of the ABC_VRPSTW algorithm. The asterisk "*" is used to indicate that our solution improves upon the other algorithms. We can see from the results that ABC_VRPSTW_{best} outperforms all the previous methods in case of traveling distance for three instances of R1 problem $(R1\_01, R1\_02, R1\_03)$ and three instances of RC1 problem $(RC1\_01, RC1\_02, RC1\_03)$.

Table 6.3 and Table 6.4 present the comparisons for some R1 and RC1 instances (which are found in the literature) with $P_{max} = 10\%$ and $W_{max} = 10\%$ respectively. For these benchmark problems, the ABC_VRPSTW_{best} has improved the traveling cost for one instance of R1 problem $(R1\_01)$ and three instances of RC1 problem $(RC1\_01, RC1\_02, RC1\_06)$.

Table 6.5 presents the details of the range of computational times for each of the algorithms compared in our experiments. The comparison of computational time between TS and AR algorithms, between IRCIs and IRCIe algorithms and among $ABC\_VRPSTW_{plan:1}$, $ABC\_VRPSTW_{plan:2}$ and $ABC\_VRPSTW_{plan:3}$ clearly show that as that our algorithm takes much reasonable time for an offline problem like VRPSTW. If we note the computational time needed among the three plans of our algorithm, we can see, as we move from Plan 1 to Plan 3, the solution quality improves (shown in Chapter 5) at the cost of higher computational time.

Table 6.1: VRPSTW Results for R1 Problems, $W_{max} = 10\%$ and $P_{max} = 5\%$

Method	BAL	TS	AR	UTS	IRCIs	IRCIe	ABC_VRPSTW_{best}	
R1-01								
#Vehicle	17	16	14	14	15	14	14	
%WB	72	65	24	45	71	68	68	
Distance(TTD)	1885	1491	1370	1438	1703	1633	1355.788	*
R1-02								
#Vehicle	15	13	12	12	13	12	12	
%WB	83	69	47	61	84	63	83	
Distance(TTD)	1636	1322	1265	1339	1629	1404	860.963	*
R1-03								
#Vehicle	13	11	11	11	11	11	11	
%WB	86	77	59	73	84	93	76	
Distance(TTD)	1452	1184	1066	1168	1357	1374	1046.573	*
R1-09								
#Vehicle	13	12	11	11	11	11	11	
%WB	95	84	60	75	85	93	72	
Distance(TTD)	1445	1154	1084	1168	1336	1393	1109.701	

Table 6.2: Results for RC1 Problems, $W_{max} = 10\%$ and $P_{max} = 5\%$

Method	BAL	TS	AR	UTS	IRCIs	IRCIe	ABC_VRPSTW_{best}	
RC1-01								
#Vehicle	14	14	13	13	14	13	13	
%WB	56	71	39	64	94	93	68	
Distance(TTD)	1839	1521	1424	1529	1776	1778	1165.142	*
RC1-02								
#Vehicle	13	13	11	12	13	12	12	
%WB	88	76	58	81	96	98	73	
Distance(TTD)	1850	1384	1375	1413	1653	1635	1256.77	*
RC1-03								
#Vehicle	12	11	10	11	11	10	10	
%WB	82	92	69	86	98	83	84	
Distance(TTD)	1496	1243	1183	1254	1456	1256	1148.507	*
RC1-06								
#Vehicle	12	12	11	11	12	11	11	
%WB	71	81	61	81	96	80	82	
Distance(TTD)	1496	1338	1223	1336	1507	1522	1303.646	

Table 6.3: VRPSTW Results for R1 Problems, $W_{max} = 10\%$ and $P_{max} = 10\%$

Method	BAL	TS	AR	UTS	IRCIs	IRCIe	ABC_VRPSTW_{best}	
R1-01								
#Vehicle	15	14	12	12	13	12	12	
%WB	62	49	8	31	43	25	44	
Distance(TTD)	1832	1388	1212	1376	1493	1314	1005.214	*
R1-02								
#Vehicle	14	13	10	11	12	10	10	
%WB	81	59	8	31	43	25	50	
Distance(TTD)	1569	1266	1173	1287	1463	1238	1240.905	
R1-03								
#Vehicle	13	11	10	10	11	10	10	
%WB	83	65	58	76	76	66	69	
Distance(TTD)	1657	1063	1013	1185	1274	1138	1228.452	
R1-09								
#Vehicle	12	11	10	11	11	10	10	
%WB	90	72	47	82	83	53	50	
Distance(TTD)	1431	1102	1005	1183	1280	1116	1221.943	

Table 6.4: Results for RC1 problems, $W_{max} = 10\%$ and $P_{max} = 10\%$

Method	BAL	TS	AR	UTS	IRCIs	IRCIe	ABC_VRPSTW_{best}	
RC1-01								
#Vehicle	14	15	11	12	14	11	11	
%WB	61	62	27	54	73	43	42	
Distance(TTD)	1795	1569	1275	1457	1839	1322	1206.937	*
RC1-02								
#Vehicle	13	12	11	11	13	11	11	
%WB	83	68	56	74	81	63	65	
Distance(TTD)	1719	1307	1222	1367	1632	1288	1156.38	*
RC1-03								
#Vehicle	12	10	10	11	11	10	10	
%WB	92	85	65	90	92	79	70	
Distance(TTD)	1530	1228	1119	1275	1400	1194	1161.612	
RC1-06								
#Vehicle	13	12	10	11	12	11	11	
%WB	97	77	49	81	92	66	64	
Distance(TTD)	1666	1342	1194	1359	1590	1253	1138.652	*

Table 6.5: Computational Time for Each Solution Method

Solution Method	CPU	Running time for each Algorithm (sec)
(1) BAL	25 MHz 80386	17 - 73
(2) TS	2.25 GHz Athlon	52 - 82
(3) AR	2.25 GHz Athlon	448 - 692
(4) UTS	600 MHz Pentium-II	193 - 1900
(5) $IRCIs$	Intel Pentium-M 1.6 MHz	4.5 - 4.9
(6) $IRCIe$	Intel Pentium-M 1.6 MHz	537 - 653
(7) $ABC\_VRPSTW_{plan:1}$	2.0 GHz Intel Core 2 Duo	5.3 - 13.55
(8) $ABC\_VRPSTW_{plan:2}$	2.0 GHz Intel Core 2 Duo	29.172 - 52.260
(9) $ABC\_VRPSTW_{plan:3}$	2.0 GHz Intel Core 2 Duo	61.065 - 137.229

However, comparisons regarding computational times should be performed with caution, because, in general, computational times are difficult to compare due to the differences in processing power and hardware. For a more complete and detailed comparison, all the algorithms need to be implemented on the same platform. However, since, VRPSTW is an offline problem, we feel that such a comparison is unnecessary. This is also supported by Figliozzi [27] and other authors [24, 2, 31] who had reported their results in the literature on this problem. The fact that ABC_VRPSTW can provide good quality solutions within reasonable amount of time guarantees the usability and applicability of it in practical settings. Readers still interested to compare the results from the point of view of computational time are referred to Dongarra's work [32] which includes the results of a set of standard programs that measure and compare processing power of different machines. Unfortunately, comparisons are usually not straightforward because not all processors are included in Dongarra's work [32]. In addition, it is difficult to account for potential differences in codes, compilers, and implementation computational efficiency.

Table 6.6 through Table 6.11 represent the comparison among two most recent results with the result of ABC_VRPSTW algorithm for all instances of type R1, R2, RC1 and RC2 problems. Results of problem types C1 and C2 are omitted here following the observation of [27] that in these instances the number of routes is bounded by the capacity constraints. Hence, even relaxing both early and late time windows does not reduce the number of routes [27]. In Table 6.6 and Table 6.7, the number of vehicles are kept same as in the standard benchmark data instances and in Table 6.8 and Table 6.9, the best previous result respectively. In Table 6.10 and Table 6.11, we present the result of ABC_VRPSTW in which we tried to reduce (improve) the number of vehicles (if possible). In all three cases, we have marked the improvement of traveling distance. The asterisk "*" beside the column labeled #vehicles and Distance under the result of ABC_VRPSTW in Table 6.6 through Table 6.11 indicates improvement.

Despite although, as has been argued above, the comparison og computational times is not very important for VRPSTW, we shall provide a comparison among ABC_VRPSTW, UTS and $IRCI_e$. Table 6.12 present the comparison of the average computational time for each of the problem type among two most recent results with result of ABC_VRPSTW algorithm.

6.1 Discussion

As reported in Table 8.1 through Table 8.8, we can see that the mean and median of the results for most of the problem instances have very small differences. Even, for some cases we have the same mean and median. This is an indication of achieving local or global optimal value for that dataset. Metaheuristics generally produce solutions of higher quality but this is usually at the expense of significantly longer computational times. Table 6.6 through Table 6.11 clearly show that, the number of routes (according to the definition of VRPSTW, which is equal to the number of vehicles) is negatively correlated with the traveling distance. Because, as the number of vehicles is reduced, percentage of window breaks remains equal/increases and so the distance also remains equal/increases. A better illustration of this analysis is given in Table 6.13 and Table 6.14.

Table 6.6: Comparison for All R and RC Type Problem Instances **(Number of vehicles is same as in the benchmark data instances) - Part I**

	UTS			IRCIe			ABC_VRPSTW			
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	
R101	14	75	1535.2	12	44	1128.7	19	25	636.789	*
R102	13	89	1416.8	11	54	1058.7	17	26	694.13	*
R103	11	96	1267.3	10	66	1027.4	13	14	691.056	*
R104	9	99	983.5	9	82	947.3	9	41	647.448	*
R105	13	98	1441.2	11	58	1073.5	14	29	657.779	*
R106	11	97	1355.3	10	67	1047.4	12	23	668.798	*
R107	10	100	1147.6	10	76	987.6	10	17	651.059	*
R108	9	100	987.7	9	86	947.2	9	51	638.095	*
R109	11	100	1264.2	10	72	1001.4	11	26	659.515	*
R110	11	100	1084.0	9	71	1013.4	10	14	691.056	*
R111	10	100	1138.5	10	74	983.3	10	21	637.658	*
R112	10	100	963.2	9	83	940.9	9	12	644.144	*
R201	3	89	1500.4	3	44	948.0	4	7.0	886.698	*
R202	3	100	1205.8	3	60	943.5	3	7.6	923.261	*
R203	3	100	950.4	2	70	901.8	3	5.0	828.751	*
R204	2	100	854.3	2	81	836.3	2	3.4	773.662	*
R205	3	100	1001.8	3	64	911.9	3	7.1	889.370	*
R206	3	100	917.9	2	75	956.9	3	5.2	826.995	*
R207	2	100	903.0	2	82	876.6	2	4.8	822.401	*
R208	2	100	738.3	2	89	833.4	2	2.8	743.445	*
R209	3	100	909.9	2	74	950.5	3	5.7	811.718	*
R210	3	100	948.2	2	71	963.8	3	4.7	798.339	*
R211	2	100	952.2	2	86	906.8	2	7.8	671.890	*

Table 6.7: Comparison for All R and RC Type Problem Instances **(Number of vehicles is same as in the benchmark data instances) - Part II**

	UTS			IRCIe			ABC_VRPSTW			
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	
RC101	13	92	1654.3	11	56	1255.3	14	22	1120.158	*
RC102	12	100	1593.7	10	68	1230.1	12	18	1163.283	*
RC103	11	100	1321.7	10	75	1154.6	11	16.5	1109.849	*
RC104	10	100	1175.2	10	88	1083.9	10	6.5	1061.688	*
RC105	12	92	1654.1	11	62	1219.7	13	21.5	1197.963	*
RC106	11	99	1422.7	10	73	1150.3	11	22.5	1082.742	*
RC107	11	100	1237.6	10	72	1123.0	10	11.5	1117.909	*
RC108	10	100	1184.6	10	90	1071.6	10	7.5	1070.015	*
RC201	4	100	1409.9	3	52	1147.4	4	5.9	900.566	*
RC202	3	100	1435.6	3	65	1073.5	3	7.5	1063.283	*
RC203	3	100	1062.4	3	71	906.3	3	6.8	833.043	*
RC204	3	100	800.0	2	86	850.7	4	5.4	778.237	*
RC205	3	93	1656.8	3	60	1158.4	4	5.7	807.953	*
RC206	3	100	1186.8	3	60	978.4	3	6.5	905.217	*
RC207	3	100	1127.8	3	67	986.4	3	4.1	657.909	*
RC208	3	100	846.1	2	79	885.5	3	4.5	690.015	*

Table 6.8: Comparison for All R and RC Type Problem Instances **(Number of vehicles is equal to the best previous result) Part - I**

	UTS			IRCIe			ABC_VRPSTW			
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	
R101	14	75	1535.2	12	44	1128.7	12	44	1005.214	*
R102	13	89	1416.8	11	54	1058.7	11	76	999.678	*
R103	11	96	1267.3	10	66	1027.4	10	50	1240.905	
R104	9	99	983.5	9	82	947.3	9	89	930.678	*
R105	13	98	1441.2	11	58	1073.5	11	75	928.793	*
R106	11	97	1355.3	10	67	1047.4	10	80	1012.653	*
R107	10	100	1147.6	10	76	987.6	10	83	934.298	*
R108	9	100	987.7	9	86	947.2	9	83	1001.328	
R109	11	100	1264.2	10	72	1001.4	10	77	995.908	*
R110	11	100	1084.0	9	71	1013.4	9	85	1001.45	*
R111	10	100	1138.5	10	74	983.3	10	61	1037.658	
R112	10	100	963.2	9	83	940.9	9	88	934.144	*
R201	3	89	1500.4	3	44	948.0	3	40	936.698	*
R202	3	100	1205.8	3	60	943.5	3	70	933.261	*
R203	3	100	950.4	2	70	901.8	2	96	885.007	*
R204	2	100	854.3	2	81	836.3	2	94	772.612	*
R205	3	100	1001.8	3	64	911.9	3	73	989.370	
R206	3	100	917.9	2	75	956.9	2	96	918.515	
R207	2	100	903.0	2	82	876.6	2	96	821.747	*
R208	2	100	738.3	2	89	833.4	2	98	813.369	
R209	3	100	909.9	2	74	950.5	2	95	922.727	
R210	3	100	948.2	2	71	963.8	2	96	940.150	*
R211	2	100	952.2	2	86	906.8	2	93	854.943	*

Table 6.9: Comparison for All R and RC Type Problem Instances **(Number of vehicles is equal to the best previous result) Part - II**

	UTS			IRCIe			ABC_VRPSTW			
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	
RC101	13	92	1654.3	11	56	1255.3	11	42	1206.937	*
RC102	12	100	1593.7	10	68	1230.1	10	55	1197.283	*
RC103	11	100	1321.7	10	75	1154.6	10	57	1130.479	*
RC104	10	100	1175.2	10	88	1083.9	10	87	1132.688	
RC105	12	92	1654.1	11	62	1219.7	11	63	1201.148	*
RC106	11	99	1422.7	10	73	1150.3	10	71	1112.742	*
RC107	11	100	1237.6	10	72	1123.0	10	69	1082.909	*
RC108	10	100	1184.6	10	90	1071.6	10	78	1110.015	
RC201	4	100	1409.9	3	52	1147.4	3	100	804.687	*
RC202	3	100	1435.6	3	65	1073.5	3	59	839.478	*
RC203	3	100	1062.4	3	71	906.3	3	62	833.043	*
RC204	3	100	800.0	2	86	850.7	2	93	677.343	*
RC205	3	93	1656.8	3	60	1158.4	3	63	813.741	*
RC206	3	100	1186.8	3	60	978.4	3	79	715.485	*
RC207	3	100	1127.8	3	67	986.4	3	76	646.888	*
RC208	3	100	846.1	2	79	885.5	2	84	729.712	*

Table 6.10: Comparison for All R and RC Type Problem Instances **(Number of vehicles is improved (if possible) than the best previous result) Part I**

	UTS			IRCIe			ABC_VRPSTW				
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles		WB(%)	Distance	
R101	14	75	1535.2	12	44	1128.7	10	*	81	1259.993	
R102	13	89	1416.8	11	54	1058.7	10	*	86	1047.908	*
R103	11	96	1267.3	10	66	1027.4	9	*	66	1240.993	
R104	9	99	983.5	9	82	947.3	9		89	930.678	*
R105	13	98	1441.2	11	58	1073.5	10	*	80	1242.694	
R106	11	97	1355.3	10	67	1047.4	9	*	85	1042.653	*
R107	10	100	1147.6	10	76	987.6	10		83	934.298	*
R108	9	100	987.7	9	86	947.2	8	*	92	1191.328	
R109	11	100	1264.2	10	72	1001.4	10		77	995.908	*
R110	11	100	1084.0	9	71	1013.4	8	*	90	1011.45	*
R111	10	100	1138.5	10	74	983.3	9	*	66	1188.568	
R112	10	100	963.2	9	83	940.9	8	*	92	1190.335	
R201	3	89	1500.4	3	44	948.0	2	*	55	1188.762	
R202	3	100	1205.8	3	60	943.5	2	*	100	937.967	*
R203	3	100	950.4	2	70	901.8	2		96	885.007	*
R204	2	100	854.3	2	81	836.3	2		94	772.612	*
R205	3	100	1001.8	3	64	911.9	2	*	92	1187.992	
R206	3	100	917.9	2	75	956.9	2		96	918.515	*
R207	2	100	903.0	2	82	876.6	2		96	821.747	*
R208	2	100	738.3	2	89	833.4	2		98	813.369	
R209	3	100	909.9	2	74	950.5	2		95	922.727	
R210	3	100	948.2	2	71	963.8	2		96	940.150	*
R211	2	100	952.2	2	86	906.8	2		93	854.943	*

Table 6.11: Comparison for All R and RC Type Problem Instances **(Number of vehicles is improved (if possible) than the best previous result) Part II**

	UTS			IRCIe			ABC_VRPSTW				
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles		WB(%)	Distance	
RC101	13	92	1654.3	11	56	1255.3	10	*	100	1223.016	*
RC102	12	100	1593.7	10	68	1230.1	9	*	59	1211.68	*
RC103	11	100	1321.7	10	75	1154.6	9	*	62	1145.479	*
RC104	10	100	1175.2	10	88	1083.9	9	*	93	1166.638	
RC105	12	92	1654.1	11	62	1219.7	9	*	83	1253.148	
RC106	11	99	1422.7	10	73	1150.3	9	*	79	1175.964	
RC107	11	100	1237.6	10	72	1123.0	9	*	76	1215.384	
RC108	10	100	1184.6	10	90	1071.6	9	*	84	1122.317	
RC201	4	100	1409.9	3	52	1147.4	2	*	100	914.687	*
RC202	3	100	1435.6	3	65	1073.5	2	*	79	999.478	*
RC203	3	100	1062.4	3	71	906.3	2	*	78	933.043	
RC204	3	100	800.0	2	86	850.7	2		93	677.343	*
RC205	3	93	1656.8	3	60	1158.4	2	*	74	1013.741	*
RC206	3	100	1186.8	3	60	978.4	2	*	89	915.485	*
RC207	3	100	1127.8	3	67	986.4	2	*	79	856.888	*
RC208	3	100	846.1	2	79	885.5	2		84	729.712	*

Table 6.12: Computational Time for Each Solution Method

	Average CPU time (sec)		
Problem Type	UTS (600 MHz Pentium-II)	IRCIe (1.6 MHz Pentium-M)	ABC_VRPSTW (2.0 GHz Core 2 Duo)
R1	786.3	724.1	366.796
R2	528.3	402.3	312.217
RC1	697.3	715.6	285.172
RC2	548.7	478.1	344.483

Table 6.13: Comparison for All R and RC Type Problem Instances **(with different number of vehicles)** Part - I

				ABC_VRPSTW					
	Table 6.6			Table 6.8			Table 6.10		
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance
R101	19	25	636.789	12	44	1005.214	10	81	1259.993
R102	17	26	694.13	11	76	999.678	10	86	1047.908
R103	13	14	691.056	10	50	1027.4	9	66	1240.993
R104	9	41	647.448	9	89	930.678	9	89	930.678
R105	14	29	657.779	11	75	928.793	10	80	1242.694
R106	12	23	668.798	10	80	1012.653	9	85	1042.653
R107	10	17	651.059	10	83	934.298	10	83	934.298
R108	9	51	638.095	9	83	1001.328	8	92	1191.328
R109	11	26	659.515	10	77	995.908	10	77	995.908
R110	10	14	691.056	9	85	1001.45	8	90	1011.45
R111	10	21	637.658	10	61	1037.658	9	66	1188.568
R112	9	12	644.144	9	88	934.144	8	92	1190.335
R201	4	7.0	886.698	3	40	936.698	2	55	1188.762
R202	3	7.6	923.261	3	70	933.261	2	100	937.967
R203	3	5.0	828.751	2	96	885.007	2	96	885.007
R204	2	3.4	773.662	2	94	772.612	2	94	772.612
R205	3	7.1	889.370	3	76	989.370	2	92	1187.992
R206	3	5.2	826.995	2	96	918.515	2	96	918.515
R207	2	100	903.0	2	96	821.747	2	96	821.747
R208	2	4.8	822.401	2	98	813.369	2	98	813.369
R209	3	5.7	811.718	2	95	922.727	2	95	922.727
R210	3	4.7	798.339	2	96	940.150	2	96	940.150
R211	2	7.8	671.890	2	93	854.943	2	93	854.943

Table 6.14: Comparison for All R and RC Type Problem Instances **(with different number of vehicles)** Part - II

			ABC_VRPSTW						
	Table 6.7			Table 6.9			Table 6.11		
Problem Type	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance	#Vehicles	WB(%)	Distance
RC101	14	22	1120.158	11	42	1206.937	10	100	1223.016
RC102	12	18	1163.283	10	55	1197.283	9	59	1211.68
RC103	11	16.5	1109.849	10	57	1130.479	9	62	1145.479
RC104	10	6.5	1061.688	10	87	1132.688	9	93	1166.638
RC105	13	21.5	1197.963	11	63	1201.148	9	83	1253.148
RC106	11	22.5	1082.742	10	71	1112.742	9	79	1175.964
RC107	10	11.5	1117.909	10	69	1082.909	9	76	1215.384
RC108	10	7.5	1070.015	10	78	1110.015	9	84	1122.317
RC201	4	5.9	900.566	3	100	804.687	2	100	914.687
RC202	3	7.5	1063.283	3	59	839.478	2	79	999.478
RC203	3	6.8	833.043	3	62	833.043	2	78	933.043
RC204	4	5.4	778.237	2	93	677.343	2	93	677.343
RC205	4	5.7	807.953	3	63	813.741	2	74	1013.741
RC206	3	6.5	905.217	3	79	715.485	2	89	915.485
RC207	3	4.1	657.909	3	76	646.888	2	79	856.888
RC208	3	4.5	690.015	2	84	729.712	2	84	729.712

Chapter 7

Conclusion

In this chapter, we draw the conclusion by highlighting the major contribution made by the research works associated with this thesis. We also provide some directions for further research.

7.1 Contribution

Vehicle Routing Problem with Time Windows has a lot of real life application including transportation system design, garbage collection fleet routing etc. Though it has been proved to be NP-Hard, the near to optimal results obtained by different heuristics and metaheuristics have aided much in its real life applications.

The real world need solution methods that are:

- Fast - the quicker the operator gets an answer back from the computer the better.
- Easy to apply to a variety of problem characteristics - when developing software for real life problems one wants to avoid reinventing the wheel every time a new client wants a software application for a new type of transportation problem.
- Precise - the better results a solution method returns the larger is the potential for savings.

- More robust - when solving real world problems it is often better to have a solution method that produces fairly good results for all problem instances.

The four characteristics listed above are to a certain extent in conflict with each other, so some sort of trade-off has to be achieved. Solution methods described in the literature are often evaluated in terms of speed, solution quality, and to a certain extent, robustness while the second characteristic listed above often receives less attention. In this thesis, a solution method that takes all four characteristics into account is presented.

The main contributions of this thesis are as follows.

1. The main contribution of this thesis is to apply a new, simple and swarm based metaheuristics, namely, Artificial Bee Colony algorithm, to solve the VRPSTW problem.
2. We develop an efficient and generalized algorithm ABC_VRPSTW for the problem based on a modular and flexible algorithmic approach which can be easily modified to solve many other variants of the VRP problem.
3. Our extensive experiments with this algorithm have shown excellent results.
4. We have minimized the total traveling distance (cost) with penalties, the number of window violations and the number of routes.
5. We analyzed the performance of our algorithm both theoretically and experimentally.
6. The proposed ABC_VRPSTW algorithm has provided high quality solutions within reasonable computational time.

VRPSTW solutions provide a workable plan of action when the problem with hard time window is infeasible. Its increased practical visibility has evolved in parallel with the development of broader and deeper research directed at its solution. We believe, our algorithm will be used in practical settings with excellent success.

7.2 Future Work

A number of future research directions and ideas have arisen out of our research work. Below we present some of the avenues which are worth investigation. In what follows, we first discuss some future research possibilities with respect to some other investigating variants of the VRP problem. Then we briefly some other future scopes related to Artificial Bee Colony algorithm in general.

7.2.1 Different VRP Variants

The flexibility and generality of our ABC_VRPSTW algorithm turn out to be very useful and important in real-world applications. Our ABC_VRPSTW algorithm can be easily changed to solve some other variants of VRP and VRPTW problems as briefly identified below.

***Vehicle Routing Problem with Hard Time Window* (VRPHTW)** VRPHTW can be solved by making the relaxed time window un-relaxed.

***Heterogeneous Fleet Vehicle Routing Problem* (HVRP)** HVRP can be solved by making the speed and capacity of the vehicles different.

***Fleet Size and Mix Vehicle Routing Problem* (FSMVRP) [33]** FSMVRP can be solved by using multiple fleet having different sizes.

***Fleet Size and Mix Vehicle Routing Problem with Time Windows* (FSMVRPTW)** FSMVRPTW can be solved in the same way as FSMVRP with additional constraint of time window.

***Time Dependent Vehicle Routing Problem* (TDVRP) [34]** Our algorithm can be easily adapted to solve TDVRP where customers are assigned time-dependent constant speeds

Multi-Commodity VRP [**35**] This variation can be solved by modifying our algorithm for each customer having multiple demand.

7.2.2 Future Scopes of ABC Algorithm

Artificial bee colony algorithm (ABC) is a recently proposed metaheuristic based on intelligent foraging behavior of honey bee swarm. Its gradually increasing interest of many researchers because of its simplicity, outstanding performance, wide applicability and fewer control parameters. A number of ABC variants have been proposed to achieve good results for many problems as follows.

- ***Bespoke Artificial Bee Colony Algorithm (BABC)*** is applied to determine the seismic location in the Earths crust and upper mantle. Here, onlooker phase of ABC is modified by embedding the concept of greedy bee inspired by [62].
- ***ABC algorithm with linear crossover operator*** is applied to solve enhanced form of real coded numerical optimization problem.
- ***Bee System (BS)*** *[63]* algorithm can be used in neural networks for pattern recognition, tuning a fuzzy logic controller for a robot gymnast, optimizing the design of mechanical components, data clustering etc.
- ***Bee Hive algorithm(BHA)*** *[64]* inspired by the communicative and evaluative method and procedures of bees is used to build an efficient fault-tolerant routing algorithm [65], to Optimize Multi Constrained Piecewise Non-Linear Economic Power Dispatch Problem [66] etc.
- ***Marriage Bee Optimization (MBO)*** *[69]*, inspired by mating behavior in honey bee is another important variation of bee based algorithm used to solve many problems.
- ***Virtual Bee Algorithm (VBA)*** *[67]* is used to solve function optimizations with the application in engineering problems. Here, a swarm of virtual bees are generated

and start to move randomly in the phase space.

- ***Queen Bee Algorithm*** based on the behavior of bees in nature where queen-bee plays a major role in reproduction process is used in wireless sensor networks [70], optimization problem of economic power dispatch [71] etc.

Chapter 8

Appendix-I

Experimental Results

Table 8.1: Result of ABC_VRPSTW for Problem Instances R1-01 to R1-06 **(Number of Customer = 200)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
R1_01 (V = 19)	*avg*	TTC = 681.124,%WB = 19	TTC = 669.418,%WB = 21	TTC = 652.823,%WB = 25
	min	TTC = 644.338,%WB = 25	TTC = 643.844,%WB = 26	TTC = 636.789,%WB = 25
	max	TTC = 702.832,%WB = 26	TTC = 693.547,%WB = 20	TTC = 685.425,%WB = 23
	med	TTC = 683.832,%WB = 20	TTC = 663.547,%WB = 20	TTC = 685.425,%WB = 27
R1_02 V = 17)	*avg*	TTC = 722.164,%WB = 30	TTC = 717.936,%WB = 29	TTC = 696.48,%WB = 28
	min	TTC = 703.958,%WB = 26	TTC = 643.844,%WB = 26	TTC = 694.13,%WB = 26
	max	TTC = 727.824,%WB = 31	TTC = 716.721,%WB = 27	TTC = 708.784,%WB = 30
	med	TTC = 725.824,%WB = 31	TTC = 716.721,%WB = 28	TTC = 690.784,%WB = 26
R1_03 (V = 13)	*avg*	TTC = 714.515,%WB = 15	TTC = 707.656,%WB = 18	TTC = 698.291,%WB = 19
	min	TTC = 708.006,%WB = 17	TTC = 699.975,%WB = 14	TTC = 691.056,%WB = 14
	max	TTC = 748.078,%WB = 22	TTC = 724.692,%WB = 18	TTC = 700.774,%WB = 17
	med	TTC = 711.854,%WB = 12	TTC = 710.7,%WB = 19	TTC = 697.487,%WB = 19
R1_04 (V = 9)	*avg*	TTC = 679.332,%WB = 16	TTC = 667.449,%WB = 16	TTC = 651.992,%WB = 12
	min	TTC = 674.069,%WB = 13	TTC = 661.496,%WB = 15	TTC = 647.448,%WB = 11
	max	TTC = 682.901,%WB = 17	TTC = 672.286,%WB = 16	TTC = 655.237,%WB = 13
	med	TTC = 679.824,%WB = 16	TTC = 665.721,%WB = 15.5	TTC = 654.784,%WB = 12.5
R1_05 (V = 14)	*avg*	TTC = 701.436,%WB = 28	TTC = 675.641,%WB = 27	TTC = 668.436,%WB = 29
	min	TTC = 694.533,%WB = 28	TTC = 678.493,%WB = 30	TTC = 657.779,%WB = 29
	max	TTC = 702.281,%WB = 27	TTC = 683.516,%WB = 27	TTC = 678.84,%WB = 29
	med	TTC = 705.428,%WB = 28.5	TTC = 671.127,%WB = 27	TTC = 665.784,%WB = 28.5
R1_06 (V = 12)	*avg*	TTC = 700.081,%WB = 24	TTC = 679.25,%WB = 21	TTC = 677.547,%WB = 25
	min	TTC = 696.176,%WB = 25	TTC = 677.948,%WB = 20	TTC = 668.798,%WB = 23
	max	TTC = 703.674,%WB = 21	TTC = 687.02,%WB = 23	TTC = 686.62,%WB = 26
	med	TTC = 701.824,%WB = 24	TTC = 670.127,%WB = 20	TTC = 680.784,%WB = 25

Table 8.2: Result of ABC_VRPSTW for Problem Instances R1-07 to R1-12 **(Number of Customer = 200)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
R1_07 (V = 10)	*avg*	TTC = 679.379,%WB = 20	TTC = 671.102,%WB = 19	TTC = 655.464,%WB = 19
	min	TTC = 671.102,%WB = 19	TTC = 660.319,%WB = 16	TTC = 651.059,%WB = 17
	max	TTC = 689.657,%WB = 22	TTC = 674.282,%WB = 17	TTC = 661.133,%WB = 19
	med	TTC = 680.542,%WB = 19.5	TTC = 675.09,%WB = 19.5	TTC = 660.784,%WB = 19
R1_08 (V = 9)	*avg*	TTC = 676.758,%WB = 14	TTC = 653.939,%WB = 11	TTC = 641.549,%WB = 12
	min	TTC = 670.514,%WB = 13	TTC = 652.587,%WB = 14	TTC = 638.095,%WB = 11
	max	TTC = 679.112,%WB = 14	TTC = 657.838,%WB = 15	TTC = 649.47,%WB = 12
	med	TTC = 677.564,%WB = 14	TTC = 645.70,%WB = 10	TTC = 645.04,%WB = 11.5
R1_09 (V = 11)	*avg*	TTC = 685.252,%WB = 23	TTC = 676.01,%WB = 25	TTC = 662.624,%WB = 27
	min	TTC = 687.587,%WB = 25	TTC = 670.828,%WB = 26	TTC = 659.515,%WB = 26
	max	TTC = 702.56,%WB = 30	TTC = 684.322,%WB = 27	TTC = 666.793,%WB = 26
	med	TTC = 690.004,%WB = 23.5	TTC = 667.721,%WB = 24.5	TTC = 666.784,%WB = 27
R1_10 (V = 10)	*avg*	TTC = 714.515,%WB = 15	TTC = 707.656,%WB = 18	TTC = 698.291,%WB = 19
	min	TTC = 708.006,%WB = 17	TTC = 699.975,%WB = 14	TTC = 691.056,%WB = 14
	max	TTC = 748.078,%WB = 22	TTC = 724.692,%WB = 18	TTC = 700.774,%WB = 17
	med	TTC = 715.4,%WB = 15	TTC = 706.001,%WB = 27	TTC = 690.784,%WB = 18
R1_11 (V = 10)	*avg*	TTC = 685.646,%WB = 25	TTC = 661.361,%WB = 23	TTC = 641.486,%WB = 20
	min	TTC = 672.439,%WB = 22	TTC = 658.587,%WB = 21	TTC = 637.658,%WB = 21
	max	TTC = 686.943,%WB = 24	TTC = 671.623,%WB = 25	TTC = 646.726,%WB = 20
	med	TTC = 690.248,%WB = 25.5	TTC = 666.721,%WB = 23.5	TTC = 640.784,%WB = 20
R1_12 (V = 9)	*avg*	TTC = 714.515,%WB = 15	TTC = 651.255,%WB = 12	TTC = 646.211,%WB = 14
	min	TTC = 708.006,%WB = 17	TTC = 649.683,%WB = 13	TTC = 644.144,%WB = 12
	max	TTC = 748.078,%WB = 22	TTC = 655.235,%WB = 15	TTC = 647.834,%WB = 17
	med	TTC = 727.824,%WB = 17	TTC = 656.721,%WB = 13	TTC = 640.784,%WB = 13.5

Table 8.3: Result of ABC_VRPSTW for Problem Instances R2-01 to R2-06 **(Number of Customer = 1000)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
	avg	TTC = 968.314,%WB = 7.4	TTC = 954.278,%WB = 7.8	TTC = 917.592,%WB = 7.3
R2_01	*min*	TTC = 910.716,%WB = 6.9	TTC = 914.716,%WB = 6.9	TTC = 886.698,%WB = 7.0
(V = 4)	*max*	TTC = 1031.827,%WB = 8.6	TTC = 989.310,%WB = 8.0	TTC = 1009.852,%WB = 8.3
	med	TTC = 967.824,%WB = 7.2	TTC = 956.721,%WB = 8.0	TTC = 914.784,%WB = 7.2
	avg	TTC = 947.211,%WB = 6.2	TTC = 943.222,%WB = 6.5	TTC = 931.070,%WB = 6.4
R2_02	*min*	TTC = 842.053,%WB = 5.0	TTC = 861.790,%WB = 5.8	TTC = 923.261,%WB = 6.7
(V = 3)	*max*	TTC = 1026.068,%WB = 7.6	TTC = 970.033,%WB = 7.5	TTC = 953.741,%WB = 6.5
	med	TTC = 947.824,%WB = 6.2	TTC = 956.721,%WB = 6.8	TTC = 940.784,%WB = 6.8
	avg	TTC = 856.707,%WB = 5.1	TTC = 834.405,%WB = 5.4	TTC = 834.405,%WB = 5.4
R2_03	*min*	TTC = 777.352,%WB = 4.0	TTC = 769.420,%WB = 4.2	TTC = 828.751,%WB = 5.0
(V = 3)	*max*	TTC = 905.421,%WB = 6.6	TTC = 888.498,%WB = 6.2	TTC = 847.512,%WB = 5.7
	med	TTC = 847.824,%WB = 5.0	TTC = 836.721,%WB = 5.5	TTC = 840.784,%WB = 5.6
	avg	TTC = 773.662,%WB = 3.4	TTC = 775.662,%WB = 3.4	TTC = 773.662,%WB = 3.4
R2_04	*min*	TTC = 759.558,%WB = 3.7	TTC = 773.662,%WB = 3.4	TTC = 773.662,%WB = 3.4
(V = 2)	*max*	TTC = 864.393,%WB = 5.2	TTC = 779.109,%WB = 3.4	TTC = 773.662,%WB = 3.4
	med	TTC = 777.638,%WB = 3.5	TTC = 780.276,%WB = 3.8	TTC = 775.209,%WB = 3.5
	avg	TTC = 927.829,%WB = 7.0	TTC = 967.613,%WB = 7.5	TTC = 945.430,%WB = 7.7
R2_05	*min*	TTC = 889.370,%WB = 6.3	TTC = 916.573,%WB = 7.3	TTC = 941.128,%WB = 7.1
(V = 3)	*max*	TTC = 1024.741,%WB = 8.0	TTC = 999.513,%WB = 7.8	TTC = 989.763,%WB = 7.7
	med	TTC = 827.256,%WB = 7.0	TTC = 956.7,%WB = 7.2	TTC = 940.784,%WB = 7.5
	avg	TTC = 847.121,%WB = 5.9	TTC = 877.384,%WB = 6.3	TTC = 865.719,%WB = 5.6
R2_06	*min*	TTC = 782.620,%WB = 4.0	TTC = 821.217,%WB = 5.4	TTC = 826.995,%WB = 5.2
(V = 3)	*max*	TTC = 963.197,%WB = 6.7	TTC = 909.899,%WB = 6.5	TTC = 879.956,%WB = 5.9
	med	TTC = 847.824,%WB = 6.0	TTC = 866.721,%WB = 6.0	TTC = 860.784,%WB = 5.2

Table 8.4: Result of ABC_VRPSTW for Problem Instances R2-07 to R2-11 **(Number of Customer = 1000)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
	avg	TTC = 832.099,%WB = 5.0	TTC = 827.401,%WB = 4.9	TTC = 822.401,%WB = 4.8
R2_07	*min*	TTC = 822.401,%WB = 4.8	TTC = 822.401,%WB = 4.8	TTC = 822.401,%WB = 4.8
(V = 2)	*max*	TTC = 953.703,%WB = 6.4	TTC = 832.099,%WB = 5.0	TTC = 822.401,%WB = 4.8
	med	TTC = 827.82,%WB = 4.8	TTC = 826.1,%WB = 5.0	TTC = 824.004,%WB = 5.0
	avg	TTC = 743.445,%WB = 3.6	TTC = 743.445,%WB = 2.8	TTC = 743.445,%WB = 2.8
R2_08	*min*	TTC = 722.845,%WB = 2.7	TTC = 743.445,%WB = 2.8	TTC = 743.445,%WB = 2.8
(V = 2)	*max*	TTC = 713.674,%WB = 4.2	TTC = 743.445,%WB = 2.8	TTC = 743.445,%WB = 2.8
	med	TTC = 743.445,%WB = 3.6	TTC = 743.445,%WB = 2.8	TTC = 743.445,%WB = 2.8
	avg	TTC = 862.452,%WB = 6.5	TTC = 831.036,%WB = 6.0	TTC = 837.869,%WB = 6.5
R2_09	*min*	TTC = 806.453,%WB = 5.1	TTC = 808.712,%WB = 5.1	TTC = 811.718,%WB = 5.7
(V = 3)	*max*	TTC = 906.269,%WB = 7.1	TTC = 875.979,%WB = 6.6	TTC = 870.849,%WB = 6.6
	med	TTC = 872.004,%WB = 6.8	TTC = 836.721,%WB = 6.2	TTC = 840.784,%WB = 6.7
	avg	TTC = 828.498,%WB = 5.8	TTC = 859.929,%WB = 6.4	TTC = 847.749,%WB = 6.2
R2_10	*min*	TTC = 792.310,%WB = 5.8	TTC = 798.339,%WB = 4.7	TTC = 798.339,%WB = 4.7
(V = 3)	*max*	TTC = 940.578,%WB = 7.3	TTC = 928.184,%WB = 7.1	TTC = 894.980,%WB = 6.2
	med	TTC = 827.824,%WB = 5.8	TTC = 856.721,%WB = 6.3	TTC = 840.784,%WB = 5.8
	avg	TTC = 671.890,%WB = 7.8	TTC = 671.890,%WB = 7.8	TTC = 671.890,%WB = 7.8
R2_11	*min*	TTC = 671.890,%WB = 7.8	TTC = 671.890,%WB = 7.8	TTC = 671.890,%WB = 7.8
(V = 2)	*max*	TTC = 671.890,%WB = 7.8	TTC = 671.890,%WB = 7.8	TTC = 671.890,%WB = 7.8
	med	TTC = 675.24,%WB = 8.0	TTC = 676.721,%WB = 8.0	TTC = 680.784,%WB = 8.1

Table 8.5: Result of ABC_VRPSTW for Problem Instances RC1 **(Number of Customer = 200)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
	avg	TTC = 1214.918,%WB = 24.0	TTC = 1178.173,%WB = 24.5	TTC = 1149.3895,%WB = 23
RC1_01	*min*	TTC = 1171.831,%WB = 25.5	TTC = 1152.456,%WB = 24	TTC = 1120.158,%WB = 22
(V = 14)	*max*	TTC = 1256.638,%WB = 25.5	TTC = 1203.890,%WB = 25	TTC = 1178.621,%WB = 24
	med	TTC = 1222.491,%WB = 23.0	TTC = 1175.456,%WB = 24.5	TTC = 1149.594,%WB = 23
	avg	TTC = 1267.32,%WB = 22	TTC = 1245.222,%WB = 20	TTC = 1193.9305,%WB = 19.25
RC1_02	*min*	TTC = 1238.334,%WB = 18	TTC = 1195.790,%WB = 18	TTC = 1163.283,%WB = 18
(V = 12)	*max*	TTC = 1308.372,%WB = 23	TTC = 1295.033,%WB = 22	TTC = 1224.578,%WB = 20.5
	med	TTC = 1265.139,%WB = 21	TTC = 1240,%WB = 19.5	TTC = 1194.372,%WB = 19.5
	avg	TTC = 1219.407,%WB = 16.5	TTC = 1182.959,%WB = 16.5	TTC = 1144.1565,%WB = 17.25
RC1_03	*min*	TTC = 1174.398,%WB = 15	TTC = 1135.420,%WB = 16	TTC = 1109.849,%WB = 16.5
(V = 11)	*max*	TTC = 1278.402,%WB = 17.5	TTC = 1230.498,%WB = 17	TTC = 1178.464,%WB = 18
	med	TTC = 1219.139,%WB = 16	TTC = 1172.547,%WB = 16	TTC = 1149.422,%WB = 17.5
	avg	TTC = 1288.926,%WB = 12	TTC = 1236.385,%WB = 9.75	TTC = 1198.0855,%WB = 9
RC1_04	*min*	TTC = 1219.007,%WB = 9.5	TTC = 1192.662,%WB = 7.5	TTC = 1061.688,%WB = 6.5
(V = 10)	*max*	TTC = 1351.943,%WB = 14	TTC = 1280.109,%WB = 12	TTC = 1234.483,%WB = 11.5
	med	TTC = 1289.177,%WB = 13	TTC = 1210.25,%WB = 8	TTC = 1188.263,%WB = 8.5
	avg	TTC = 1376.693,%WB = 21	TTC = 1278.126,%WB = 20.75	TTC = 1249.258,%WB = 22
RC1_05	*min*	TTC = 1296.57,%WB = 20	TTC = 1235.763,%WB = 20.5	TTC = 1197.953,%WB = 21.5
(V = 13)	*max*	TTC = 1394.202,%WB = 21.5	TTC = 1320.49,%WB = 21	TTC = 1300.563,%WB = 22.5
	med	TTC = 1336.645,%WB = 20	TTC = 1280.547,%WB = 21	TTC = 1276.728,%WB = 22
	avg	TTC = 1207.524,%WB = 24	TTC = 1150.558,%WB = 23	TTC = 1124.1785,%WB = 22
RC1_06	*min*	TTC = 1130.176,%WB = 21.5	TTC = 1105.217,%WB = 22	TTC = 1082.742,%WB = 22.5
(V = 11)	*max*	TTC = 1254.792,%WB = 26	TTC = 1195.899,%WB = 24	TTC = 1165.615,%WB = 20.5
	med	TTC = 1212.741,%WB = 25	TTC = 1161.89,%WB = 22.5	TTC = 1276.728,%WB = 22.5
	avg	TTC = 1280.581,%WB = 16.5	TTC = 1183.25,%WB = 15.5	TTC = 1196.226,%WB = 14.5
RC1_07	*min*	TTC = 1238.584,%WB = 15.5	TTC = 1190.401,%WB = 14	TTC = 1117.909,%WB = 11.5
(V = 10)	*max*	TTC = 1328.239,%WB = 17	TTC = 1176.099,%WB = 17	TTC = 1234.543,%WB = 18
	med	TTC = 1278.52,%WB = 17	TTC = 1197.39,%WB = 16.5	TTC = 1193.21,%WB = 14
	avg	TTC = 1231.835,%WB = 9.5	TTC = 1163.02,%WB = 9.25	TTC = 1128.743,%WB = 10
RC1_08	*min*	TTC = 1178.61,%WB = 8.5	TTC = 1113.445,%WB = 7.5	TTC = 1070.015,%WB = 7.5
(V = 10)	*max*	TTC = 1284.592,%WB = 11.5	TTC = 1212.6,%WB = 11	TTC = 1167.471,%WB = 11
	med	TTC = 1231.251,%WB = 9.5	TTC = 1160.07,%WB = 9	TTC = 1149.218,%WB = 12

Table 8.6: Result of ABC_VRPSTW for Problem Instances RC2 **(Number of Customer = 1000)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
	avg	TTC = 1167.42,%WB = 6.2	TTC = 1117.394,%WB = 6.6	TTC = 1104.347,%WB = 7.2
RC2_01	*min*	TTC = 912.361,%WB = 5.4	TTC = 889.566,%WB = 5.2	TTC = 900.566,%WB = 5.9
(V = 4)	*max*	TTC = 1374.361,%WB = 7.0	TTC = 1345.222,%WB = 8.0	TTC = 1319.613,%WB = 8.1
	med	TTC = 1222.491,%WB = 6.7	TTC = 1125.456,%WB = 7.0	TTC = 1215.945,%WB = 7.5
	avg	TTC = 1191.385,%WB = 7.1	TTC = 1145.222,%WB = 7.95	TTC = 1193.9305,%WB = 8.25
RC2_02	*min*	TTC = 970.613,%WB = 5.4	TTC = 1067.804,%WB = 7.1	TTC = 1063.283,%WB = 7.5
(V = 3)	*max*	TTC = 1456.777,%WB = 8.8	TTC = 1472.777,%WB = 8.8	TTC = 1489.578,%WB = 9.0
	med	TTC = 1264.266,%WB = 7.5	TTC = 1140.003,%WB = 7.90	TTC = 1194.372,%WB = 8.5
	avg	TTC = 1075.929,%WB = 6.25	TTC = 1069.262,%WB = 7.45	TTC = 1118.675,%WB = 7.45
RC2_03	*min*	TTC = 956.547,%WB = 4.2	TTC = 833.043,%WB = 6.8	TTC = 833.043,%WB = 6.8
(V = 3)	*max*	TTC = 1196.663,%WB = 8.3	TTC = 1088.324,%WB = 8.1	TTC = 1187.15,%WB = 8.1
	med	TTC = 1090.009,%WB = 7.0	TTC = 1110.02,%WB = 7.9	TTC = 1017.422,%WB = 7.1
	avg	TTC = 870.3193,%WB = 4.85	TTC = 936.813,%WB = 5.6	TTC = 1198.0855,%WB = 6.1
RC2_04	*min*	TTC = 793.5,%WB = 3.0	TTC = 802.45,%WB = 4.5	TTC = 778.237,%WB = 5.4
(V = 4)	*max*	TTC = 1024.207,%WB = 6.7	TTC = 997.465,%WB = 6.5	TTC = 995.389,%WB = 6.8
	med	TTC = 866.2625,%WB = 4.5	TTC = 912.456,%WB = 5.2	TTC = 1188.263,%WB = 8.5
	avg	TTC = 1092.967,%WB = 6.55	TTC = 1058.278,%WB = 5.75	TTC = 1049.258,%WB = 5.7
RC2_05	*min*	TTC = 829.757,%WB = 4.9	TTC = 858.511,%WB = 6.0	TTC = 807.953,%WB = 5.7
(V = 4)	*max*	TTC = 1395.636,%WB = 8.2	TTC = 1258.046,%WB = 7.2	TTC = 1250.563,%WB = 7.0
	med	TTC = 1105.963,%WB = 7.5	TTC = 1080.547,%WB = 7.8	TTC = 1116.728,%WB = 6.2
	avg	TTC = 1069.16,%WB = 6.6	TTC = 1130.194,%WB = 7.25	TTC = 1116.876,%WB = 6.8
RC2_06	*min*	TTC = 870.495,%WB = 5.5	TTC = 905.217,%WB = 6.3	TTC = 1008.137,%WB = 6.5
(V = 3)	*max*	TTC = 1252.488,%WB = 7.7	TTC = 1355.171,%WB = 8.2	TTC = 1225.615,%WB = 7.1
	med	TTC = 1070.579,%WB = 6.8	TTC = 1061.89,%WB = 6.5	TTC = 1011.728,%WB = 5.5
	avg	TTC = 913.0991,%WB = 5.95	TTC = 976.25,%WB = 6.5	TTC = 946.226,%WB = 6.1
RC2_07	*min*	TTC = 777.993,%WB = 4.3	TTC = 776.401,%WB = 4.5	TTC = 657.909,%WB = 4.1
(V = 3)	*max*	TTC = 1197.205,%WB = 7.6	TTC = 1176.099,%WB = 17	TTC = 1234.543,%WB = 18
	med	TTC = 937.539,%WB = 6.3	TTC = 938.778,%WB = 6.3	TTC = 1001.21,%WB = 7.3
	avg	TTC = 733.1407,%WB = 3.4	TTC = 809.404,%WB = 6.15	TTC = 1128.743,%WB = 4.5
RC2_08	*min*	TTC = 664.573,%WB = 1.8	TTC = 772.208,%WB = 5.5	TTC = 690.015,%WB = 4.5
(V = 3)	*max*	TTC = 811.641,%WB = 5.0	TTC = 846.6,%WB = 6.8	TTC = 767.471,%WB = 4.5
	med	TTC = 734.073,%WB = 3.8	TTC = 660.07,%WB = 6.2	TTC = 1119.218,%WB = 7.5

Table 8.7: Result of ABC_VRPSTW for Problem Instances C1 **(Number of Customer = 200)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
	avg	TTC = 938.263,%WB = 34.25	TTC = 900.622,%WB = 33.5	TTC = 853.677,%WB = 33.75
C1_01	*min*	TTC = 869.803,%WB = 33.5	TTC = 830.298,%WB = 32	TTC = 791.508 ,%WB = 32.5
(V = 10)	*max*	TTC = 1006.723,%WB = 35	TTC = 970.946,%WB = 35	TTC = 915.846 ,%WB = 35
	med	TTC = 930.698,%WB = 33.5	TTC = 925.456,%WB = 34	TTC = 850.945,%WB = 33
	avg	TTC = 859.787,%WB = 25	TTC = 820.290,%WB = 24	TTC = 803.645,%WB = 26.5
C1_02	*min*	TTC = 800.949,%WB = 24	TTC = 767.804,%WB = 23	TTC = 736.759 ,%WB = 25.5
(V = 10)	*max*	TTC = 918.625,%WB = 26	TTC = 872.777,%WB = 25	TTC = 870.531 ,%WB = 27.5
	med	TTC = 860.266,%WB = 25	TTC = 812.735,%WB = 23.5	TTC = 814.372,%WB = 27
	avg	TTC = 693.572,%WB = 18.5	TTC = 654.304,%WB = 17.5	TTC = 599.358,%WB = 18
C1_03	*min*	TTC = 612.522 ,%WB = 17.5	TTC = 599.18 ,%WB = 17	TTC = 589.187,%WB = 18
(V = 10)	*max*	TTC = 774.622 ,%WB = 19.5	TTC = 709.429 ,%WB = 18	TTC = 609.529,%WB = 20
	med	TTC = 689.009,%WB = 18.0	TTC = 650.02,%WB = 18	TTC = 589.422,%WB = 17.5
	avg	TTC = 936.9975,%WB = 9	TTC = 849.957,%WB = 8.25	TTC = 699.089,%WB = 7.75
C1_04	*min*	TTC = 850.532,%WB = 8	TTC = 702.45,%WB = 7	TTC = 628.310 ,%WB = 6.5
(V = 10)	*max*	TTC = 1023.463,%WB = 10	TTC = 997.465,%WB = 9.5	TTC = 769.868 ,%WB = 9
	med	TTC = 936.262,%WB = 9	TTC = 862.456,%WB = 8.5	TTC = 688.263,%WB = 7.5
	avg	TTC = 957.8165,%WB = 32.25	TTC = 958.278,%WB = 30.5	TTC = 894.087,%WB = 31
C1_05	*min*	TTC = 909.222,%WB = 33	TTC = 858.511,%WB = 32	TTC = 835.119 ,%WB = 32.5
(V = 10)	*max*	TTC = 1006.411,%WB = 31.5	TTC = 958.046,%WB = 32.5	TTC = 953.055 ,%WB = 29.5
	med	TTC = 965.003,%WB = 32.5	TTC = 980.547,%WB = 31.5	TTC = 896.728,%WB = 31
	avg	TTC = 807.75,%WB = 29.25	TTC = 730.194,%WB = 28.5	TTC = 703.426,%WB = 32
C1_06	*min*	TTC = 733.590,%WB = 28	TTC = 705.217,%WB = 28.5	TTC = 689.941 ,%WB = 31
(V = 10)	*max*	TTC = 881.910,%WB = 30.5	TTC = 855.189,%WB = 29	TTC = 716.910 ,%WB = 33
	med	TTC = 812.579,%WB = 29.5	TTC = 761.89,%WB = 28.5	TTC = 711.728,%WB = 32.5
	avg	TTC = 1149.255,%WB = 28.75	TTC = 976.25,%WB = 29	TTC = 923.840,%WB = 28.5
C1_07	*min*	TTC = 905.613,%WB = 27.5	TTC = 876.401,%WB = 28	TTC = 844.600 ,%WB = 27
(V = 10)	*max*	TTC = 1392.897,%WB = 30	TTC = 1176.099,%WB = 30	TTC = 1003.081 ,%WB = 30
	med	TTC = 1112.006,%WB = 26	TTC = 978.778,%WB = 29	TTC = 920.385,%WB = 28
	avg	TTC = 923.033,%WB = 25.5	TTC = 879.404,%WB = 24	TTC = 874.782,%WB = 26
C1_08	*min*	TTC = 876.692,%WB = 23	TTC = 772.208,%WB = 23	TTC = 779.385,%WB = 24
(V = 10)	*max*	TTC = 969.375,%WB = 28	TTC = 886.006,%WB = 25	TTC = 942.180,%WB = 28
	med	TTC = 934.073,%WB = 27	TTC = 876.907,%WB = 24	TTC = 869.218,%WB = 27
	avg	TTC = 747.097,%WB = 18.25	TTC = 746.590,%WB = 19.5	TTC = 862.782,%WB = 26
C1_09	*min*	TTC = 670.832,%WB = 16.5	TTC = 690.386,%WB = 18.5	TTC = 773.385 ,%WB = 24
(V = 10)	*max*	TTC = 823.362,%WB = 20	TTC = 802.795,%WB = 20.5	TTC = 952.180, %WB = 28
	med	TTC = 745.2,%WB = 18	TTC = 740.07,%WB = 19	TTC = 869.218,%WB = 27

Table 8.8: Result of ABC_VRPSTW for Problem Instances C2 **(Number of Customer = 700)**

Problem Instances	**Result Type**	**Plan: 1**	**Plan: 2**	**Plan: 3**
	avg	TTC = 562.73,%WB = 11.85	TTC = 553.5,%WB = 11.8	TTC = 544.424,%WB = 12.05
C2_01	*min*	TTC = 554.971,%WB = 11.7	TTC = 547.566,%WB = 11.7	TTC = 541.204,%WB = 12
(V = 3)	*max*	TTC = 570.489,%WB = 12	TTC = 560.222,%WB = 12.0	TTC = 547.644,%WB = 12.1
	med	TTC = 560.43,%WB = 11.8	TTC = 552.225,%WB = 11.8	TTC = 544.945,%WB = 12.0
	avg	TTC = 549.719,%WB = 10.1	TTC = 541.204,%WB = 9.7	TTC = 543.23,%WB = 9.7
C2_02	*min*	TTC = 541.204,%WB = 9.7	TTC = 541.204,%WB = 9.7	TTC = 541.204,%WB = 9.7
(V = 3)	*max*	TTC = 558.235,%WB = 10.5	TTC = 541.204,%WB = 9.7	TTC = 545.275,%WB = 9.7
	med	TTC = 544.266,%WB = 10.0	TTC = 541.204,%WB = 9.7	TTC = 543.23,%WB = 9.7
	avg	TTC = 556.4615,%WB = 6.1	TTC = 548.762,%WB = 6.2	TTC = 545.630,%WB = 6.2
C2_03	*min*	TTC = 549.445 ,%WB = 5.4	TTC = 542.201,%WB = 6.2	TTC = 541.204 ,%WB = 6.2
(V = 3)	*max*	TTC = 563.478 ,%WB = 6.8	TTC = 555.324,%WB = 6.2	TTC = 550.057 ,%WB = 6.2
	med	TTC = 560.009,%WB = 6.3	TTC = 547.02,%WB = 6.2	TTC = 545.0,%WB = 6.2
	avg	TTC = 560.938,%WB = 3.6	TTC = 542.239,%WB = 3.6	TTC = 542.239,%WB = 3.6
C2_04	*min*	TTC = 551.445 ,%WB = 3.4	TTC = 541.204,%WB = 3.4	TTC = 541.204,%WB = 3.4
(V = 4)	*max*	TTC = 570.432 ,%WB = 3.8	TTC = 543.275,%WB = 3.8	TTC = 543.275,%WB = 3.8
	med	TTC = 560.456,%WB = 3.6	TTC = 540.0,%WB = 3.5	TTC = 540.0,%WB = 3.5
	avg	TTC = 556.712,%WB = 11.25	TTC = 542.239,%WB = 11.4	TTC = 542.239,%WB = 11.4
C2_05	*min*	TTC = 547.374,%WB = 10.8	TTC = 541.204,%WB = 11.4	TTC = 541.204,%WB = 11.4
(V = 4)	*max*	TTC = 566.050,%WB = 11.7	TTC = 543.275,%WB = 11.4	TTC = 543.275,%WB = 11.4
	med	TTC = 555,%WB = 11	TTC = 542,%WB = 11.4	TTC = 542,%WB = 11.4
	avg	TTC = 558.409,%WB = 11.05	TTC = 544.947,%WB = 11.1	TTC = 544.947,%WB = 11.1
C2_06	*min*	TTC = 556.673,%WB = 10.7	TTC = 541.204,%WB = 11.1	TTC = 541.204,%WB = 11.1
(V = 3)	*max*	TTC = 560.146,%WB = 11.4	TTC = 548.690,%WB = 11.1	TTC = 548.690,%WB = 11.1
	med	TTC = 559,%WB = 11	TTC = 545,%WB = 11.1	TTC = 545,%WB = 11.1
	avg	TTC = 549.377,%WB = 10.85	TTC = 546.226,%WB = 10.25	TTC = 946.226,%WB = 10.25
C2_07	*min*	TTC = 543.275,%WB = 10.5	TTC = 541.204,%WB = 9.4	TTC = 541.204,%WB = 9.4
(V = 3)	*max*	TTC = 555.480,%WB = 11.2	TTC = 548.690,%WB = 11.1	TTC = 548.690,%WB = 11.1
	med	TTC = 550.555,%WB = 10.5	TTC = 547.854,%WB = 10.5	TTC = 1001.21,%WB = 7.3
	avg	TTC = 561.761,%WB = 10.25	TTC = 561.761,%WB = 6.15	TTC = 561.761,%WB = 10.25
C2_08	*min*	TTC = 559.006,%WB = 9.7	TTC = 559.006,%WB = 9.7	TTC = 559.006,%WB = 9.7
(V = 3)	*max*	TTC = 564.516,%WB = 10.8	TTC = 564.516,%WB = 10.8	TTC = 564.516,%WB = 10.8
	med	TTC = 562.245,%WB = 10.5	TTC = 562.245,%WB = 10.5	TTC = 562.245,%WB = 10.5

Chapter 9

Appendix-II

Algorithms of each step of ABC_VRPSTW

Algorithm 4 Generate-Random-Solution

```
1: for each vehicle $T$ in set of all vehicles do
2:     pick a random customer to be served served by $T$
3: end for
4: for each vehicle $T$ in set of all vehicles do
5:     $size_T \leftarrow |customer|/|vehicle|$ {$size_T$ is the average number of customers served by each vehicle}
6:     $done \leftarrow$ false
7:     while (!done) do
8:         pick a customer not previously chosen by any other vehicle and nearest to immediately previously chosen customer for $T$
9:         If the chosen customer's demand can be satisfied by the vehicle's capacity then pick the customer to be served by $T$
10:        if All the customer's are taken then
11:            $done \leftarrow$ true
12:        end if
13:        if $T$ has $size_T$ customers then
14:            $done \leftarrow$ true
15:        end if
16:    end while
17: end for
18: if any customer is not chosen by any vehicle then
19:     for all unchosen customers $c$ do
20:         Find a vehicle $T$ such that it has enough capacity to serve the demand of $c$ and assign $c$ to $T$
21:     end for
22: end if
23: return The assignment of customers to vehicles
```

Algorithm 5 Generate-Initial-Solution

```
1: for each Employed Bee $eb$ do
2:     call $Generate - Random - Solution$ and assign the solution as $eb$'s food source
3: end for
```

Algorithm 6 Generate-Neighborhood-Solution($\mathcal{S}$)

```
1: find out the customer($c$) from $\mathcal{S}$ which yields maximum penalty
2: swap $c$ with a random customer assigned to a randomly chosen vehicle from $\mathcal{S}$
3: for each vehicle $T$ in $\mathcal{S}$ do
4:     $x \leftarrow$ A RANDOM NUMBER
5:     select $x$ sequential customers assigned to $T$ such that they yield maximum penalty
       of all $x$ sized blocks of customers from $T$
6:     make a random permutation of the chosen block of customers
7: end for
8: return The $\mathcal{S}$ with the new configuration of customers
```

Algorithm 7 Cost(S)

```
1: for each vehicle $T \in S$ do
2:   $D \leftarrow$ Sum of demands of all customers assigned to $T$
3:   if $D > T_q$ then
4:     $S$ is abandoned.
5:     The Employed Bee associated with $S$ becomes **Scout**.
6:     call $Generate - Random - Solution$ for that Scout bee.
7:     Assign this randomly generated solution to that Scout bee and make it Employed
       bee.
8:     return $\infty$
9:   end if
10: end for
11: $total \leftarrow 0$
12: $totalPenalty \leftarrow 0$
13: $time \leftarrow 0$
14: for each vehicle $T \in S$ do
15:   $dist \leftarrow$ sum of successive customer distance required to travel from first to last
      customer
16:   During each successive movement between customers update time with the distance
      between the customers.
17:   if reached at any customer after its closing time window then
18:     add the delay from closing time window to $totalPenalty$
19:   end if
20:   if reached at any customer before its opening time window then
21:     set time to the opening time window of the customer
22:   end if
23:   $total \leftarrow total + dist$
24: end for
25: return $total$
```

Algorithm 8 Fitness(S)

```
1: $t \leftarrow 0$
2: for each employed bee $eb$ do
3:     $s \leftarrow$ The solution associated with $eb$
4:     $t \leftarrow t + \frac{1}{Cost(s)}$
5: end for
6: $p \leftarrow \frac{1}{Cost(s)}$
7: $fitness \leftarrow \frac{p}{t}$
8: return $fitness$
```

Algorithm 9 Onlooker-Search

```
1: {Assign Onlooker Bee}
2: for each employed bee $eb$ do
3:     $S \leftarrow$ Solution associated with $eb$
4:     $f \leftarrow$ Calculate the fitness of solution S
5:     Assign $f \times MAX\_ONLOOKER$ number of onlooker bee to $S$
6: end for
7: {Search with Onlooker Bee}
8: for each employed bee $eb$ do
9:     $S \leftarrow$ Solution associated with $eb$
10:    $n \leftarrow$ Number of Onlooker assigned with $S$
11:    call $Generate - Neighborhood - Solution(S)$ $n$ times and choose the solution with
       minimum $Cost$ of these $n$ solutions and $S$
12:    Assign the best solution to $eb$
13: end for
```

Bibliography

[1] G. B. Dantzig and J. H. Ramser. *The Truck Dispatching Problem*. MANAGEMENT SCIENCE, 6 : 80 - 91, 1959.

[2] W. C. Chiang and R. A. Russell. *A metaheuristic for the vehicle-routening problem with soft time windows*. Journal of the Operational Research Society, 55 : 1298 - 1310, 2004.

[3] A. Singh. *An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem*. Applied Soft Computing, 9(2): 625 - 631, 2009.

[4] D. Karaboga. *An idea based on honey bee swarm for numerical optimization*.TR06, October, 2005.

[5] J. K. Lenstra and A. H. G. Rinnooy Kan. *Complexity of vehicle routing and scheduling problems*. Networks, 11 : 221 - 227, 1981.

[6] N. Azi, M. Gendreau and J.-Y. Potvin. *An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles*. European Journal Of Operational Research, 202 : 756 - 763, 2010.

[7] Y. Marinakis and M. Marinaki. *Bumble bees mating optimization algorithm for the vehicle routing problem*. Handbook of Swarm Intelligence, Adaptation, Learning and Optimizatoin, 8 : 347 - 369, 2010.

[8] B. Yu and Z.-Z. Yang. *An ant colony optimization model: The period vehicle routing problem with time windows*. Transportation Research Part E: Logistics and Transportation Review, 47 : 166 - 181, 2010.

[9] X. Hu, Q. Ding and Y. Wang. *A hybrid ant colony optimization and its application to vehicle routing problem with time windows*. Life System Modeling and Intelligent Computing, Communications in Computer and Information Science, 97 : 70 - 76, 2010.

[10] T. Zhen, Q. Zhang, W. Zhang and Z. Ma. *Hybrid Ant Colony Algorithm for the Vehicle Routing with Time Windows* . Computing, Communication, Control and management, 1 : 8 - 12, 2008.

[11] S. Häckel and P. Dippold. *The bee colony-inspired algorithm (bcia): a two-stage approach for solving the vehicle routing problem with time windows*. In 11th Annual conference on Genetic and evolutionary computation, 2009.

[12] A. Le Bouthillier, and T. G. Crainic. *A cooperative metaheuristic for the vehicle routing problem with time windows*. Computers & Operations Research, 32, 2005.

[13] H. hashimoto, and M. Yagiura. *A path relinking approach with an adaptive mechanism to control parameters for the vehicle routing problem with time windows*.Lecture Notes in Computer Science 4972, 254, 2008.

[14] C. I. Hsu, S. F. Hung, and H. C. Li. *Vehicle routing problem with time-windows for perishable food delivery*. Journal of Food Engineering, 80 : 465 - 475, 2007.

[15] A. Imran, S. Salhi and N. A. Wassan. *A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem*. European Journal of Operational Research, 197 : 509 - 518, July, 2008.

[16] T. D. Seeley. *The Wisdom of the Hive*. Harvard University Press, Cambridge, MA, 1995.

[17] M.-C. Lai. *A hybrid genetic/benders' algorithm with applications to vehicle routing and scheduling problems.* Doctoral Dissertation, 209, 2004.

[18] Chen J. C. ,Chang P. F. ,Chen B. B. ,Chen C. S. ,Chen C. W. ,Huang S.. *Application of vehicle routing problem with hard time window constraints.* International conference on high performance scientific computing: Modelling, simulation and optimization of complex processes, 28, 2003.

[19] N. Balakrishnan. *Simple heuristics for the vehicle routing problem with soft time windows.* The journal of Operational Research Society, 44 : 279 - 287, 1993.

[20] M. M. Solomon. *Algorithms for vehicle-routing and scheduling problems with time window constraints.* Operations Research, 35 : 254 - 265, 1987.

[21] J. Potvin and J. Rousseau. *A parallel route building algorithm for the vehicle routing and scheduling problem with time windows.* European Journal of Operational Research, 66 : 331 - 340, 1993.

[22] G. Loannou, M. Kritikos and G. Prastacos. *A greedy look-ahead heuristic for the vehicle routing problem with time windows.* Journal of the Operational Reasearch Society, 52 : 523 - 537, 2001.

[23] Y. A. Koskosidis, W. B. Powell and M. M. Solomon. *An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints.* Transportation Science, 26 : 69 - 85, May, 1992.

[24] N. Balakrishnan. *Simple Heuristics for the Vehicle Routeing Problem with Soft Time Windows.* Journal of the Operational Research Society, 44 : 279 - 287, 1993.

[25] G. Ioannou, M. Kritikos and G. Prastacos. *A problem generator-solver heuristic for vehicle routing with soft time windows.* Omega : 41 - 53, 2003.

[26] H. I. Calvete, C. Galé, M.-J. Oliveros and B. Sánchez-Valverde. *A goal programming approach to vehicle routing problems with soft time windows.* European Journal of Operational Research, 177 : 1720 - 1733, November, 2005.

[27] M. A. Figliozzi. *An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows*. Transportation Research Part C, 18 : 668 - 679, 2010.

[28] M.-C. Lai. *A hybrid genetic/benders' algorithm with applications to vehicle routing and scheduling problems*. Doctoral Dissertation, 209, 2004.

[29] P. Badeau, M. Gendreau, F. Geurtin, J.-Y. Potvin and É. Taillard. *A tabu search heuristic for the vehicle routing problem with soft time windows*. Transportation Science, 31 : 170 - 186, May, 1997.

[30] T. D. Seeley. *The Wisdom of the Hive*. Harvard University Press, Cambridge, MA, 1995.

[31] Z. Fu, R. Eglese and L. Y. O. Li. *A unified tabu search algorithm for vehicle routing problems with soft time windows*. Journal of Operational Research Society, 59 : 663 - 673, 2008.

[32] J. J. Dongarra. *Performance of Various Computers Using Standard Linear Equations Software*. Technical Report CS-89-85, University of Tennessee, November 20, 2007.

[33] M. Gendreau, G. Laporte, C. Musaraganyi and É. D. Taillard. *A tabu search heuristic for the heterogeneous fleet vehicle routing problem*. Computers & Operations Research, 26 : 1153 - 1173, 1999.

[34] M. Figliozzi. *A route improvement algorithm for the vehicle routing problem time dependent travel times*. In Proceeding of the 88th Transportation Research Board Annual Meeting CD ROM, January, 2009.

[35] P. P. Repoussis, C. D. Tarantilis and G. Ioannou. *A hybrid metaheuristic for a real life vehicle routing problem*. Numerical Methods and Applications, 4310 : 247 - 254, 2007.

[36] A. Baykasoğlu, L. Özbakir and P. Tapkan. *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*. Swarm Intelligence, Focus on Ant and Particle Swarm Optimization.

[37] J. A. Pacurib. *Solving Sudoku Puzzles Using Improved Artificial Bee Colony Algorithm*. Innovative Computing, Information and Control (ICICIC), 885 - 888, December, 2009.

[38] I. M. S. de Oliveira and R. Schirru. *Swarm intelligence of artificialbees applied to In-Core Fuel Management Optimization*. Annals of Nuclear Energy, 38(5) : 1039 - 1045, May, 2011.

[39] P. Agrawal, H. Kaur and D. Bhardwaj. *ENHANCED BEE COLONY ALGORITHM FOR SOLVING TRAVELLING SALESPERSON PROBLEM*. International Journal of Control Theory and Computer Modelling (IJCTCM) : 2(4), July 2012.

[40] D. Teodorović, T. Davidović and M. *Š*elmić. *Bee Colony Optimization: The Applications Survey*.

[41] D. Karaboga and B. Akay. *A survey: algorithms simulating bee swarm intelligence*. Artificial Intelligence Review, 31 : 61 - 85, 2009.

[42] D. Karaboga, B. Gorkemli, C. Ozturk and N. Karaboga. *A comprehensive survey: artificial bee colony (ABC) algorithm and applications*. Artificial Intelligence Review, 2012.

[43] M. Hosny. *Heuristic Techniques for Solving the Vehicle Routing Problem with Time Windows*. International Conference on Future Information Technology (IPCSIT) : 13, 2011.

[44] D. Karaboga and B. Basturk. *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*. Journal of Global Optimization, 39 : 459 - 471, 2007.

[45] J B Atkinson. *A Greedy Look-Ahead Heuristic for Combinatorial Optimisation: An Application to Vehicle Scheduling with Time Windows.*Journal of the Operational Research Society, 1994.

[46] K. C. Tan , L. H. Lee , Q. L. Zhu , K. Ou . *Heuristic methods for vehicle routing problem with time windows.* 2000.

[47] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. *Optimization by Simulated Annealing.* Science, 220 : 671 - 680, 1983.

[48] Sam R. Thangiah , Ibrahim H. Osman , Tong Sun. *Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows.* 1993.

[49] Fred Glover. *TABU SEARCH FUNDAMENTALS AND USES.* June, 1995.

[50] Sam R. Thangiah, Ibrahim H. Osman and Tong Sun. *Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows.*

[51] http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm.Tereshko05

[52] Ling Wang, Gang Zhou, Ye Xu and Min Liu . *An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling.* The International Journal of Advanced Manufacturing Technology, 60 : 1111 - 1123, June, 2012.

[53] Tasgetiren, M.F., Quan-Ke Pan, Suganthan, P.N., Chen, A.H.-L. *A discrete artificial bee colony algorithm for the permutation flow shop scheduling problem with total flowtime criterion* . Evolutionary Computation (ECE), 1 - 8, July, 2010.

[54] Yan-Feng Liu and San-Yang Liu. *A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem.* Applied Soft Computing, November, 2011.

[55] Adil Baykasoğlu, Lale Özbakir and Pinar Tapkan. *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*. Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, 1 December, 2007.

[56] Gaowei YAN and Chuangqin LI. *An Effective Refinement Artificial Bee Colony Optimization Algorithm Based On Chaotic Search and Application for PID Control Tuning*. Journal of Computational Information Systems, 7(9) : 3309 - 3316, 2011.

[57] Quan-Ke Pan, M. Fatih Tasgetiren, P. N. Suganthan and T. J. Chua. *A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem*. Information Sciences: an International Journal, 181(12) : 2455 - 2468, June, 2011.

[58] Gang Zhou, Ling Wang, Ye Xu and Shengyao Wang. *An effective artificial bee colony algorithm for multi-objective flexible job-shop scheduling problem*. ICIC'11 Proceedings of the 7th international conference on Advanced Intelligent Computing Theories and Applications: with aspects of artificial intelligence , 2011.

[59] Amin Tahooneh and Koorush Ziarati. *Using Artificial Bee Colony to Solve Stochastic Resource Constrained Project Scheduling Problem*. Advances in Swarm Intelligence, 6728 : 293 - 302, 2011.

[60] Alok Singh. *An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem*. Applied Soft Computing, 9(2) : 625 - 631, 2009.

[61] http://neo.lcc.uma.es/radi-aeb/WebVRP/

[62] T.K. Sharma, M. Pant. *Enhancing different phases of artificial bee colony for continuous global optimization problems*. International Conference on Soft Computing for Problem Solving, SocProS, 130 : 715 - 724, 2011.

[63] P. Lučić, D. Teodorović. *Bee system: modelling combinatorial optimization transportation engineering problems by swarm intelligence*. Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, 2001.

[64] H.F. Wedde, M. Farooq, Y. Zhang. *BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behaviour*. Optimization and Swarm Intelligence, 2004.

[65] Horst F. Wedde, Muddassar Farooq, and Yue Zhang. textslBeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. 2004.

[66] B. Padmanabhan, J. Jasper M. E. and Siva Kumar R. S. *Bee Hive Algorithm to Optimize Multi Constrained Piecewise Non-Linear Economic Power Dispatch Problem in Thermal Units*. International Journal on Electrical Engineering and Informatics, November, 2001.

[67] X.S. Yang. *Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms.* IWINAC, 2005.

[68] C.S.Chong, M.Y.H.Low, A.I. Sivakumar, K.L. Gay. *A Bee Colony Optimization Algorithm to Job Shop Scheduling*. Proceedings of the 37th Winter Simulation, 2006.

[69] H.A. Abbass. *MBO: Marriage in Honey Bees Optimization A Haplometrosis Polygynous Swarming Approach*. CEC2001 Proceedings of the Congress on Evolutionary Computation, 2001.

[70] Z. Pooranian, A. Barati and A. Movaghar. *Queen-bee Algorithm for Energy Efficient Clusters in Wireless Sensor Networks*. World Academy of Science, 2011.

[71] Qin, L.D., Jiang, Q.Y., Zou, Z.Y. and Cao, Y.J. *A queen-bee evolution based on genetic algorithm for economic power dispatch*. Universities Power Engineering Conference, 2004.

[72] http://w.cba.neu.edu/msolomon/problems.htm

[73] Ted Ralphs, Joe Hartman and Matt Galati. *Capacitated Vehicle Routing and Some Related Problems.*

[74] Sophie N. Parragh, Karl F. Doerner, Richard F. Hartl. *A survey on pickup and delivery problems Part II: Transportation between pickup and delivery locations.* 2008.