DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

META-HEURISTIC SOLUTION APPROACHES FOR TRAVELING SALESMAN AND TRAVELING REPAIRMAN PROBLEMS

by
Çağla CERGİBOZAN

January, 2013
İZMİR

META-HEURISTIC SOLUTION APPROACHES FOR TRAVELING SALESMAN AND TRAVELING REPAIRMAN PROBLEMS

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

by
Çağla CERGİBOZAN

January, 2013
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"META-HEURISTIC SOLUTION APPROACHES FOR TRAVELING SALESMAN AND TRAVELING REPAIRMAN PROBLEMS"** completed by **ÇAĞLA CERGİBOZAN** under supervision of **ASST. PROF. DR. A. SERDAR TAŞAN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Dr. A. Serdar TAŞAN

Supervisor

[illegible]

Asst. Prof. Dr. Derya EREN AKYOL

(Jury Member)

[illegible]

Prof. Dr. Okan TUNA

(Jury Member)

[illegible]

Prof. Dr. Mustafa SABUNCU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

During the progress of this thesis, I have received much support from some special people and I would like to thank all those people who have supported me incomparably.

Firstly, I wish to express my gratitude to my supervisor Asst. Prof. Dr. A. Serdar TAŞAN. He gave me encouragement, guidance and support not only for this thesis but also other fields in my life. I am also indebted to the authors who helped me to get the well-known data sets for the computations.

Besides, I thank all my friends and colleagues who have experienced this process with me. Their friendship and sincerity eased the process.

Finally, I am very grateful to my whole family for their endless love, support and tolerance; but my mother and my father deserve a separate thank. They have opened up my horizon with their valuable and creative ideas through the development of this thesis and my sister; she is always by my side whenever I need.

Çağla CERGİBOZAN

META-HEURISTIC SOLUTION APPROACHES FOR TRAVELING SALESMAN AND TRAVELING REPAIRMAN PROBLEMS

ABSTRACT

The traveling salesman problem (TSP) is a combinatorial optimization problem which has been extensively studied for years. TSP is the problem of creating a Hamiltonian cycle in which each node is visited only once to minimize total distance travelled. The ant colony optimization (ACO) is a meta-heuristic approach for solving optimization problems. In the study, an ACO based algorithm which utilizes local search heuristics is proposed. Proposed algorithm is applied to well-known TSP datasets and then the performance of the approach is discussed according to the results obtained from computations.

The travelling repairman problem (TRP) is the problem of finding a Hamiltonian path in which the objective is to minimize total waiting time of all customers that are situated at different locations. Genetic algorithms (GA) are meta-heuristic solution methods which are created by taking inspiration from the evolution process. As a second study, a hybrid algorithm which combines genetic algorithm with a local search heuristic is proposed to solve TRP. Proposed algorithm is applied to a set of instances that have been studied in the literature. Performance of the approach is evaluated according to the results of the computational study.

Aim of these studies is to develop efficient and effective algorithms that can be applicable to real life problems to solve large scale TSP and TRP problems.

As the third study, a case study about a snow disaster situation based on some assumptions is examined as TSP and TRP. Proposed algorithms are applied to the case and results are discussed.

Keywords: Traveling salesman problem, traveling repairman problem, genetic algorithms, ant colony optimization

GEZGİN SATICI VE GEZGİN TAMİRCİ PROBLEMLERİ İÇİN META-SEZGİSEL ÇÖZÜM YAKLAŞIMLARI

ÖZ

Gezgin satıcı problemi (GSP) uzun yıllardır yoğun bir şekilde çalışılan bir kombinatoryal optimizasyon problemidir. GSP, kat edilen toplam mesafeyi en aza indirmek için her noktaya sadece bir kez uğranılan bir Hamilton turu yaratma problemidir. Karınca kolonisi optimizasyonu (KKO), optimizasyon problemlerini çözmek için meta-sezgisel bir yaklaşımdır. Çalışmada, yerel arama sezgisellerinden yararlanan KKO tabanlı bir algoritma önerilmiştir. Önerilen algoritma iyi bilinen GSP veri setlerine uygulanmış ve sonrasında hesaplamalardan elde edilen sonuçlara göre algoritmanın performansı tartışılmıştır.

Gezgin tamirci problemi (GTP) farklı konumlarda bulunan müşterilerin bekleme sürelerinin toplamını en aza indirmenin amaçlandığı bir Hamilton turu bulma problemidir. Genetik algoritmalar (GA) evrim sürecinden ilham alınarak yaratılmış meta-sezgisel çözüm yöntemleridir. İkinci çalışmada GTP'yi çözmek için genetik algoritmayı yerel arama sezgiseli ile birleştiren bir hibrit algoritma önerilmiştir. Önerilen algoritma literatürde çalışılmış bir dizi örneğe uygulanmıştır. Algoritmanın performansı hesaplama çalışmasının sonucuna göre değerlendirilmiştir.

Bu çalışmaların amacı, büyük ölçekli GSP ve GTP problemlerini çözmek için gerçek hayat problemlerine uygulanabilen verimli ve etkili algoritmalar geliştirmektir.

Üçüncü çalışma olarak, varsayımları temel alan bir kar felaketi durumu hakkında bir vaka çalışması GSP ve GTP olarak çalışılmıştır. Önerilen algoritmalar vakaya uygulanmış ve sonuçları tartışılmıştır.

Anahtar sözcükler: Gezgin satıcı problemi, gezgin tamirci problemi, genetik algoritmalar, karınca kolonisi optimizasyonu

CONTENTS

Page

M.Sc THESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZ v

CHAPTER ONE – INTRODUCTION 1

CHAPTER TWO – META-HEURISTICS 4

2.1 Definition of the Term Meta-heuristic 4
2.2 Meta-heuristics in Combinatorial Optimization 4
2.2.1 Greedy Randomized Adaptive Search Procedure 6
2.2.2 Swarm Intelligence 7
2.2.2.1 Particle Swarm Optimization 7
2.2.2.2 Ant Colony Optimization 8
2.2.3 Tabu Search 10
2.2.4 Evolutionary Algorithms 10
2.2.5 Simulated Annealing 11

CHAPTER THREE– MAX-MIN ANT SYSTEM ALGORITHM COMBINED WITH LOCAL SEARCH HEURISTICS FOR TRAVELING SALESMAN PROBLEM 12

3.1 Introduction to Traveling Salesman Problem 12
3.2 Problem Definition 13
3.3 Literature Review 14
3.3.1 Solution Methods for Traveling Salesman Problem 15
3.3.1.1 Exact Methods 15

3.3.1.1.1 Integer Linear Programming Formulations ... 15
3.3.1.1.2 Branch and Bound Method ... 16
3.3.1.1.3 Dynamic Programming ... 16
3.3.1.1.4 Cutting Planes Algorithm ... 16
3.3.1.1.5 Branch and Cut Method ... 17
3.3.1.2 Heuristic Methods ... 17
3.3.1.2.1 2-opt Heuristic ... 17
3.3.1.2.2 3-opt Heuristic ... 18
3.3.1.2.3 Lin-Kernighan Type Exchange ... 19
3.4 Methodology of the Proposed Approach ... 19
3.4.1 Max-Min Ant System Algorithm ... 20
3.4.2 Details of Proposed Method ... 21
3.5 Computational Study ... 22
3.6 Results ... 25

CHAPTER FOUR – A HYBRID GENETIC ALGORITHM FOR TRAVELING REPAIRMAN PROBLEM ... 40

4.1 Introduction to Traveling Repairman Problem ... 40
4.2 Problem Definition ... 40
4.3 Literature Review ... 43
4.3.1 Solution Methods for Traveling Repairman Problem ... 43
4.3.1.1 Exact and Approximation Methods ... 43
4.3.1.2 Meta-heuristic Approaches ... 44
4.4 Methodology of the Proposed Approach ... 45
4.4.1 Genetic Algorithm ... 45
4.4.1.1 Solution Representation ... 46
4.4.1.2 Initialization ... 47
4.4.1.3 Evaluation ... 47
4.4.1.4 Selection ... 48
4.4.1.5 Recombination ... 48
4.4.1.6 Mutation ... 49

4.4.1.7 Replacement.. 49
4.4.2 Details of Proposed Method 50
4.5 Computational Study.. 51
4.5.1 Bounds for Traveling Repairman Problem... 51
4.6 Results 55

CHAPTER FIVE – A CASE STUDY FOR TRAVELING REPAIRMAN PROBLEM: ROUTING OF REPAIRMEN UNDER HEAVY SNOW CONDITIONS.. 72

5.1 Problem Definition 72
5.2 Literature Review... 73
5.3 A Case Study... 74
5.4 Computational Study.. 75
5.5 Results and Discussion.. 83

CHAPTER SIX – CONCLUSION 86

REFERENCES.. 89

CHAPTER ONE
INTRODUCTION

Reaching an objective by effectively use of limited resources on hand is a noticeable objective for not only humans but also several systems in our environment. One of these systems in the operations research context is the supply chain. Supply chain is the system of whole affected units during the emergence of a product/service and delivery to customer (*Glossary of Terms - Council of Supply Chain Management Professionals*, n.d.). Since the importance of the supply chain for all parts of the concept has been understood, it is continuously endeavoured to find a method to improve the performance of the linked channels.

One basic process in the supply chain is to transport goods to related customers and this process can be accomplished with logistics operations. The term logistics firstly emerged in military operations, but then used in business context. *Logistics (business) -- Britannica Online Encyclopedia* (n.d.) stated that the term logistics is defined by the Council of Logistics Management as "the process of planning, implementing, and controlling the efficient, effective flow and storage of goods, services, and related information from point of origin to point of consumption for the purpose of conforming to customer requirements". This definition puts forward that logistics operations not only consider transportation of the goods, but also provide storage of the related parts of the goods and consider managing activities. For every supply chain, logistic operations take an important place; because the transportation costs emerge in these operations highly affect the total cost of e.g. a firm. Therefore it is obviously needed to decrease the transportation costs for every part of a supply chain.

During the investigation process of the problems in transportation operations, the well-known traveling salesman problem (TSP) and traveling repairman problem (TRP) are encountered. TSP is the problem of finding a least cost tour of a salesman that starts with an initial point, delivers customer orders by visiting each customer exactly once, and returns to the initial point. It can be said that the basic concepts in this problem emerged with the studies of William Rowan Hamilton (Hamilton biography, 1998) and Thomas Penyngton Kirkman (Kirkman biography, 1996). TSP has been intensively studied for years. This problem may seem easy to compute; besides exact solution techniques for the problem have been proposed in the literature. Therefore, as the problem size (number of customers) grows, the computation time of an exact solution method increases greatly. The necessity of solving such a problem leads the notion of approximating to optimal solution of the problem. Thus, heuristic and meta-heuristic approaches emerged to be able to reach this aim.

The TRP is the problem of finding a tour in which the objective is to minimize total waiting times of all customers that are located at different places. TRP can be seen as a more recent problem in comparison to TSP. Exact and approximate methods have been developed in the TRP literature. The structure of TRP from the viewpoint of hardness is similar to TSP; therefore meta-heuristic approaches for TRP are improved in recent years.

The difference between TSP and TRP is that, in TSP, the objective is to minimize total cost of the salesman (product/service supplier); but in TRP the main objective is to minimize total waiting times of the customers. Consequently, it can be said in TRP a customer oriented view is applied while in TSP a supplier oriented view is considered. TSP and TRP are combinatorial optimization problems and there stands a wide field of research to find better solutions in accordance with the structures of the problems. In this thesis, TSP and TRP are examined with two meta-heuristic approaches.

The first thing that has done for solving these problems was to understand the foundations and the emergence forms of the problems. Afterwards, studies and methods that have been proposed for these problems are explored.

Firstly in Chapter 2, the term meta-heuristic will be defined and general meta-heuristics will be briefly described. The first meta-heuristic that will be examined in this thesis is ant colony optimization (ACO). ACO is a nature based meta-heuristic approach for solving optimization problems. Detailed information about ACO will be given in Chapter 2 and Chapter 3. The second meta-heuristic that will be examined is the genetic algorithm (GA). GA is the artificial form of the evolution process in combinatorial optimization. Detailed information on GA can be found in Chapter 4.

In Chapter 3, a max-min ant system (MMAS) algorithm based approach will be applied to TSP. The aim of the study is to develop an efficient and effective algorithm to solve large scale TSP instances. The algorithm will be applied to well-known TSP datasets and then the performance of the approach will be discussed according to the results.

A hybrid GA for TRP will be examined in Chapter 4. In the study, a hybrid algorithm which combines genetic algorithm with a local search heuristic will be proposed to solve TRP. Proposed algorithm will be applied to a set of instances that have been studied in the literature and then the performance of the approach will be evaluated according to the results of the computational study.

In Chapter 5, a case study for TRP will be examined. Severe winter conditions may cause some communication problems especially in areas have high altitudes. The case study is thought as a snow disaster situation. A district of Erzurum city from the East Anatolian Region in Turkey is accepted as the origin, and demand points are the villages of this district. This case will be studied as a conceptual example for TRP; in a real application there will be more constraints and special characteristics in the problem. Finally in Chapter 6, the study will be concluded.

CHAPTER TWO
META-HEURISTICS

2.1 Definition of the Term Meta-heuristic

The term *meta-heuristic* is a widely used concept in solution methods of the optimization problems. A meta-heuristic can be defined as an algorithmic structure which consists of concepts that guide heuristic methods to search for a good solution. Many definitions of meta-heuristics have been made in the literature, and two of them are mentioned here. Voß, Martello, Osman, & Roucairol (1999) define the term meta-heuristic as “an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions”. However Dorigo, Birattari, & Stützle (2006) pointed out problem independency of the term by definition of “a general-purpose algorithmic framework that can be applied to different optimization problems with relatively few modifications” (p.30).

Although there are differences between these definitions, common principles of meta-heuristics can be remarked as follows:

- Meta-heuristics are problem independent
- Meta-heuristics coordinate subordinate heuristics by utilizing their advantages and characteristics
- Meta-heuristics search the solution space of the problem to find a better solution
- Meta-heuristics effort to reach near optimal or optimal solutions as soon as possible

2.2 Meta-heuristics in Combinatorial Optimization

Meta-heuristic approaches have been widely used in the field of combinatorial optimization problems. Osman & Laporte (1996) define combinatorial optimization as “the mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects usually finite in numbers” (p.514). Meta-heuristic approaches have the advantage of reaching near optimal solutions in a reasonable time period in situations where exact solution approaches are not useful for implementation.

Figure 2.1 displays a general scheme for meta-heuristic solution approaches. Meta-heuristics can be examined under two distinct classes as construction based meta-heuristics and improvement based meta-heuristics from the viewpoint of solution structure.

Construction based meta-heuristics try to create a solution from starting with an empty solution on hand. A solution to the relevant problem is found by adding solution components to the initial element. This search can be said as a stepwise approach to reach a solution. As construction based meta-heuristic, greedy randomized adaptive search procedure and meta-heuristics arise from swarm intelligence approach can be mentioned here.

Improvement based meta-heuristics have a different solution search method. These meta-heuristics start with an initial solution and search the solution space to find a better solution. During this search, solution is modified and/or solution elements are recombined to improve the solution on hand. Meta-heuristics such as evolutionary algorithms, simulated annealing, local search based approaches and tabu search algorithm belong to this class of meta-heuristics.

Common meta-heuristics used to solve optimization problems are listed below and briefly described. Besides, the research on meta-heuristics is continuing and in recent years some meta-heuristics such as bat algorithm, firefly algorithm, bee-colony optimization and intelligent water drops algorithm are emerged by taking inspiration from the nature. This indication shows there is an open area to find new meta-heuristics that could yield good results to optimization problems.

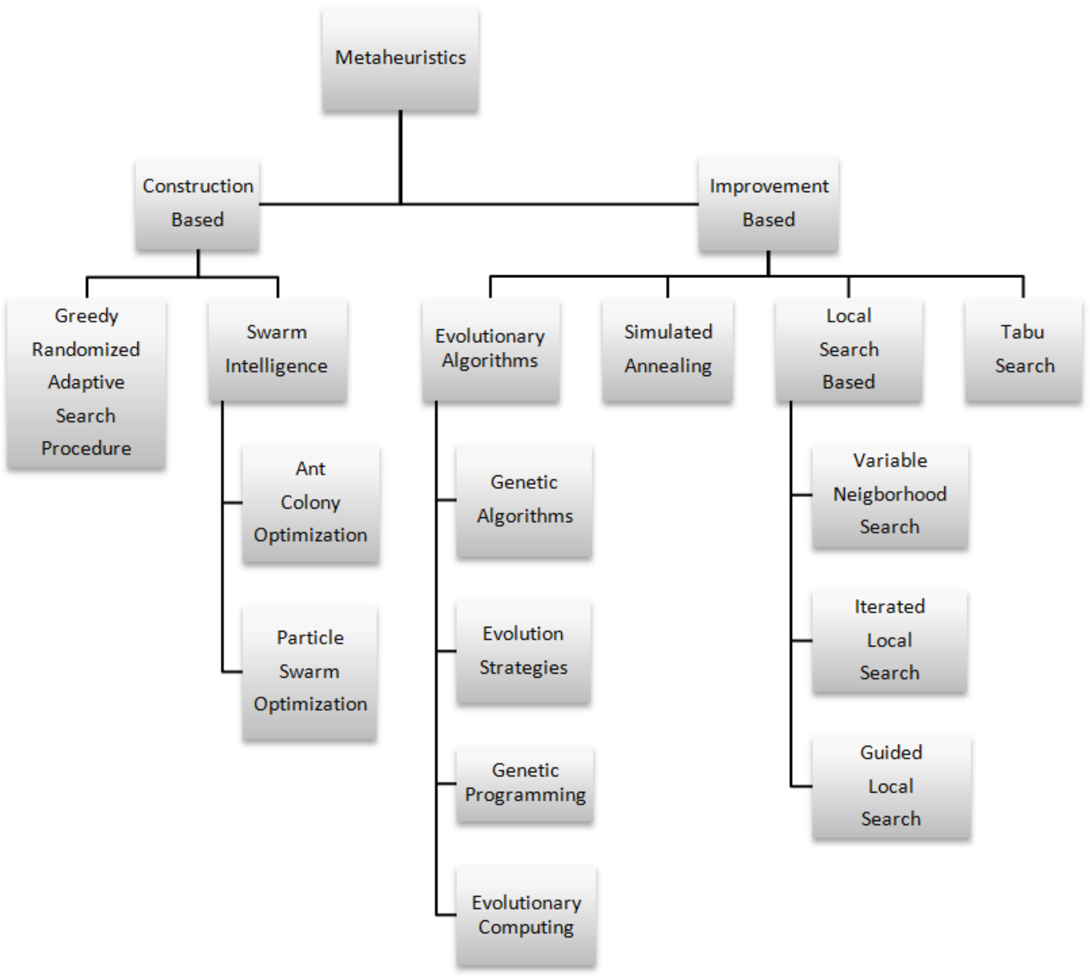

Figure 2.1 General structure of the meta-heuristics

2.2.1 Greedy Randomized Adaptive Search Procedure

Greedy randomized adaptive search procedure (GRASP) belongs to the class of constructive meta-heuristics. In GRASP, a solution is constructed from an empty solution by addition of the solution elements step by step. Insertion of an element to the solution is made according to predetermined performance criterion. Therefore, all of the candidates are evaluated and set up in an order of best meeting the performance criterion.

The method utilizes from a restricted candidate list in which best performing candidates are positioned. Restricted candidate list helps for selection of elements that will be added to the current solution and this selection is made randomly. The resulting solution is improved then by use of a local search procedure. Additional information about GRASP can be found in (Feo & Resende, 1995) and (Resende & Ribeiro, 2010).

2.2.2 Swarm Intelligence

While the investigations on solving combinatorial optimization problems were continuing, an approach is emerged with an inspiration from the natural behaviour of some species of living creatures. The communication between individuals in nature leads to improvement of the *swarm intelligence* concept (Bonabeau, Dorigo & Theraulaz, 1999). Individuals of living creature species can communicate with each other in different ways, for that reason; it can be said there is an open area for research in the swarm intelligence concept. The particle swarm optimization (PSO) and ant colony optimization (ACO) meta-heuristics can be examined under this concept.

2.2.2.1 Particle Swarm Optimization

One of the approaches in the swarm intelligence area is the PSO (Merkle & Middendorf, 2005). In PSO, it is aimed to reach an objective by ensuring exchange of information in the swarm.

Particles in the swarm can be thought as separate solutions. Each of these solutions has the knowledge of the global best solution in the swarm and continues to search in accordance with the result of evaluating some information. This information is the current objective function value of the solution, best objective function value of the solution, velocity of the solution and global best objective function value. All of the particles effort to improve the best objective function value with consideration of the information on hand.

2.2.2.2 Ant Colony Optimization

Definition of the ACO is made in (Dorigo & Stützle, 2010) as “a metaheuristic that is inspired by the pheromone trail laying and following behaviour of some ant species”. Ant colonies have the capability of finding a food source with their remarkable information sharing type (via *pheromone*). Figure 2.2 indicates behaviour of the ants. Ants leave pheromone substance on their ground while they are walking. Through the alternative paths between their nest and food, the path which has the most pheromone amount will be chosen by the ants. Since this behaviour is thought as a new solution approach, researchers’ investigation is increasingly proceeding on solving combinatorial optimization problems by use of ACO.

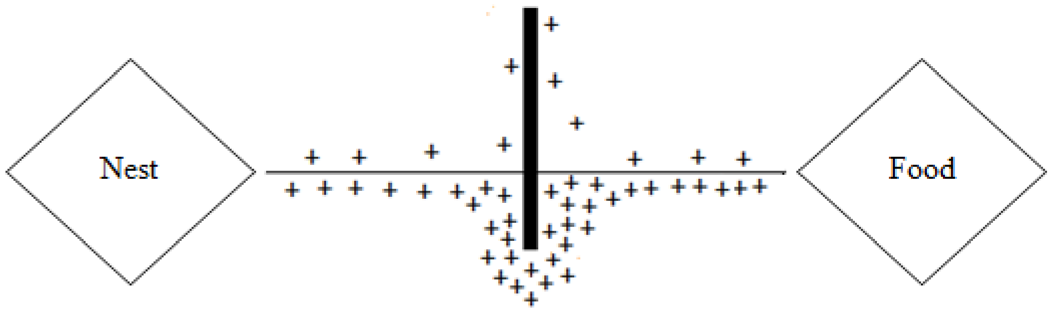

Figure 2.2 Behaviour of the ants. It can be seen that ants find their shortest way through their nest and food by the pheromone amount on that way (Zäpfel, Braune, & Bögl, 2010).

ACO algorithm is an artificial type of the natural behaviour of the ants. In Figure 2.3 ACO algorithm structure can be seen. Algorithm starts with creating paths for whole artificial ants and proceeds with calculating the lengths of these paths, updating the pheromone amount, keeping the shortest path in memory and finally ends when the capability criteria is reached. To achieve enhanced performance of the ant systems not only building the algorithm but it is also suggested to hybridize it with a local search component (Voß, 2001). Detailed information on ACO can be found in (Dorigo & Blum, 2005; Dorigo & Di Caro, 1999; Dorigo, Di Caro, & Gambardella, 1999; Dorigo & Stützle, 2004; Taillard, 1999).

```
Ant Colony Optimization Algorithm
    • Generate $m$ ants and assign each of them to one node
    • Determine initial parameters $\alpha, \beta, \rho,$ and pheromone levels ($\tau_{ij}$) on each edge
    for $t$ = 1: iteration no
        for $k$ = 1 : $m$
            • Compute the probability $p^k_{ij}$ of selecting next node
            • Determine length of the complete tour
        end
        for $i$ = 1 : $n$
            • Update pheromone amounts on all of the edges
        end
        Update the best solution
    end
```

Figure 2.3 The structure of the ACO meta-heuristic

ACO algorithms differ from each other with some aspects. In comparison, each of them has the basic ACO structure but some special characteristics. For instance, in the ant system (AS) algorithm, pheromone updating operation is made by all of the ants due to Eq. (2.1) and Eq. (2.2).

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{2.1}$$

$$\Delta\tau_{ij}^{\ k} = \begin{cases} Q/L_k & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour,} \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

Here τ_{ij} corresponds the pheromone amount on the edge *(i,j)*, ρ means pheromone evaporation rate, m is number of the ants, Q is a constant, L_k is the length of the route created by ant k and $\Delta\tau_{ij}^{k}$ is the pheromone trail information of the kth ant on the edge *(i,j)*. An ant that locates in a node searches for its next node by calculating the probability p^k_{ij} of selecting the alternative nodes which have not been visited yet.

This calculation can be seen in Eq. (2.3). Here α is a parameter shows importance of the pheromone and β parameter is the importance of the visibility factor. As it can be seen from Eq. (2.4), η_{ij} is the heuristic information associated with the distance between edge *(i,j)*, s^p is the partial solution that is constructed by the ant, $N\,(s^p)$ is the set of the candidate nodes for the corresponding ant, the solution component c_{ij} indicates that city *j* is the next city that will be visited after city *i* (Dorigo et al., 2006).

$$p_{ij}{}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_{ij} \in N(s^p), \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{2.4}$$

2.2.3 Tabu Search

Tabu search meta-heuristic is found by Glover (1986) and it utilizes from a basic local search principle. First of all, an initial solution is found. In each iteration, solutions that have been found in the previous iteration called as "*tabu*" and a set of these solutions, *tabu list,* is created. Whole neighbours of the initial solution are created; if current solution is improved in one of the neighbours and the neighbour is not in the tabu list, new solution will be this neighbour and the same search will proceed on this solution. Tabu list is updated in each iteration. A solution which declared as tabu can be removed from the tabu list after update operation. The main idea in this method is to make a search by avoiding repeated solutions with the help of the tabu list which works as a memory (Gendreau, 2003).

2.2.4 Evolutionary Algorithms

As it can be clearly understood from the name, evolutionary algorithms (EA) are the search strategies based on the evolution process. The evolution process is imitated and used for reaching the “best solution” among other solutions with these algorithms. Crossover, mutation, selection terms are adapted to the searching context in the combinatorial optimization problems. Genetic algorithms (GA), evolution strategies (ES), genetic programming (GP) and evolutionary programming (EP) can be mentioned under this class. GA is examined in this study and detailed information about GA is given in Chapter 4.

2.2.5 Simulated Annealing

Simulated annealing method belongs to the class of improvement meta-heuristics. The studies of Kirkpatrick, Gelatt & Vecchi (1983) and Cerny (1985) are the first studies on this method. The method is an optimization method inspired by the temperature changes of the materials. It starts with a temperature level and a solution. After that a neighbour of that solution is generated. Objective function value of the neighbour is evaluated according to the solution quality difference with the previous solution, and neighbour becomes new solution if it has better objective function value. If the neighbour has a worse objective function value than current solution, the neighbour becomes new solution with a probability in which quality of the solution and the temperature level is considered. Finally, the temperature level is updated and the algorithm is terminated when the capability criteria is reached.

CHAPTER THREE
MAX-MIN ANT SYSTEM ALGORITHM COMBINED WITH LOCAL SEARCH HEURISTICS FOR TRAVELING SALESMAN PROBLEM

3.1 Introduction to Traveling Salesman Problem

Assume that a salesman wants to make a tour in which he starts with an origin point, visits each customer only once and returns to the origin point with minimum cost. The traveling salesman problem (TSP) is the problem of finding the shortest route of a salesman who must visit each customer only once. As a mathematical expression, the term *Hamiltonian cycle* that is found by William Rowan Hamilton (Hamilton biography, 1998) can be used to define this problem. Hamiltonian cycle is a cycle in which each node is visited exactly once. Therefore, TSP is the problem of creating the shortest Hamiltonian cycle in a graph. TSP is one of the most studied combinatorial optimization problems. The problem can be differentiated with a few modifications and studied in different areas such as logistics, scheduling, manufacturing, mathematics, electronics, computer science etc. (Applegate, Bixby, Chvátal, & Cook, 2006). Over decades, TSP still draws researchers' attention and there stands a wide field of research for finding better solution methods in accordance with the structure of the problem (Traveling Salesman Problem, 2012).

TSP has a difficulty because of its computational complexity. There has not been found any polynomial time algorithm for TSP, so the complexity of the problem can be said as exponential. It is proven by Karp (1972) that TSP is an NP-complete (Non-deterministic Polynomial-time- Complete) problem.

Assume that n corresponds to the number of cities. Number of the possible Hamiltonian cycles in a complete graph will be $(n-1)!/2$ in the directed case, and $(n-1)!$ in the undirected case. This number means that in large sizes of instances, it is almost impossible to evaluate all of the tours and find a solution in a reasonable time period.

3.2 Problem Definition

The problem can be examined as *symmetric TSP* or *asymmetric TSP*. In symmetric TSP, distance between two locations is equal to distance in the reverse direction. In this situation, graph is an undirected graph. In asymmetric case, distance between two locations is not equal to distance in the reverse direction or the reverse way between these two locations doesn't exist, thus the graph becomes directed. TSP is examined as a complete graph which means all vertices have an edge with other vertices.

Assume that $G = (V, A)$ is a directed graph where $V = \{1,...,n\}$ is the vertex set and $A = \{(i, j) \mid i, j \in V, i \neq j\}$ is the arc set. c_{ij} is defined as the cost associated with the arc *(i, j)*. x_{ij} is a binary decision variable equal to 1 if arc *(i,j)* is included in the tour, 0 otherwise. The TSP can be formulated as follows with refer to (Gutin & Punnen, 2002):

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij}$

Subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1,...,n) \tag{3.1}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1,...,n) \tag{3.2}$$

$$\sum_{i \in S}\sum_{j \notin S} x_{ij} \geq 1 \qquad (S \subset V, |S| \geq 2) \tag{3.3}$$

$$x_{ij} \in \{0,1\} \qquad (i, j = 1,...,n) \tag{3.4}$$

In this formulation, the objective function is the total length of the tour. Constraints (3.1), (3.2) and (3.4) which are also called *assignment constraints* ensure that a city is visited only once. Constraint (3.3) is the *subtour elimination constraint* that assures at least one arc comes out from each subset $S = \{(i, j) \mid x_{ij} = 1\}$ in the tour

(Dantzig, Fulkerson, & Johnson, 1954). In other words, each subset is connected to another subset with this constraint (Gutin & Punnen, 2002).

3.3 Literature Review

In the 1800s, Irish physicist and mathematician William Rowan Hamilton (Hamilton biography, 1998; Wilkins, 2000) and British mathematician Thomas Penyngton Kirkman (Kirkman biography, 1996) studied some mathematical problems relevant to TSP. In the 1930s, mathematician Menger (1932) mentioned the problem in a colloquium held in Vienna and after a while Hassler Whitney (Whitney biography, 2005) and Merrill Flood (Merrill M. Flood, 2012) from Princeton University studied the problem and it seems that Whitney is the first person to call the problem as “Traveling Salesman Problem” (Lawler, Lenstra, Rinnooy Kan, & Shmoys, 1985). However it is stated in (Gutin & Punnen, 2002) that the report of Menger (1932) in which the TSP is examined as *Messenger Problem,* is the first published work on TSP. After that, studies of Dantzig et al. (1954), Flood (1956), Mahalanobis (1940) and Robinson (1949) can be seen as the earliest studies on TSP.

The first computational study of TSP is made by Dantzig et al. (1954) in which a 49 city problem is studied and the authors gave a solution with linear programming techniques. After that, a great number of problem instances have been created and used for the comparison of different solution algorithms. These well-known instances can be found in TSPLIB (2008).

An example of a Hamiltonian cycle is given in Figure 3.1. In the tour, it is obvious that each node is served only once and the tour is a closed tour which means the salesman finishes his tour in the initial point.

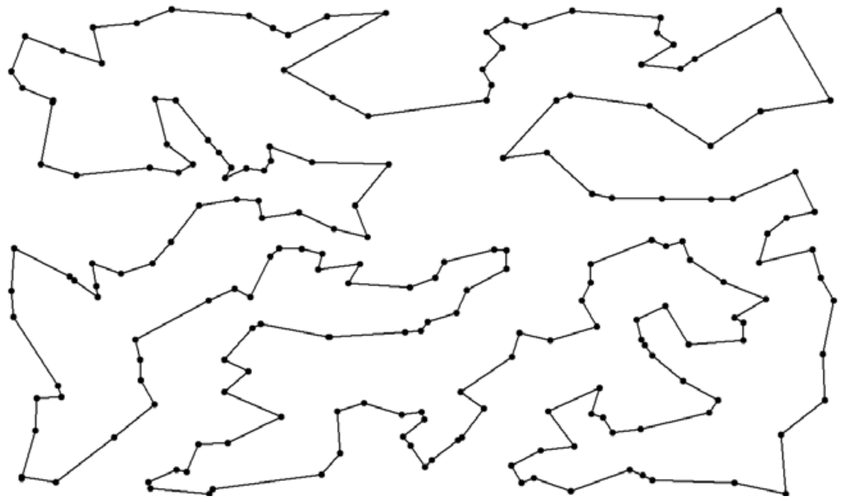
Figure 3.1 An example of a TSP solution with random points

3.3.1 Solution Methods for Traveling Salesman Problem

Since the emergence of the problem, TSP has been extensively studied and several solution techniques have been proposed in the literature. We can classify these methods as exact methods and heuristic methods.

3.3.1.1 Exact Methods

Exact solution methods for TSP are solution approaches that can find optimum solutions via enumerative search process. Exact methods for TSP are not seemed to be useful solution techniques for the large instances because of the required computational time. Integer linear programming formulations, branch and bound method, branch and cut method, dynamic programming and cutting planes algorithm can be mentioned as exact algorithms.

3.3.1.1.1 Integer Linear Programming Formulations. Integer linear programming (ILP) models ensure an exact solution by definition of decision variables, constraints and objective function. In these models, the structure of the constraints, variable types and objective function can be organized and modified according to special characteristics of the problem. ILP models find the optimal solution of the problem, but in some situations these models are not useful methods to examine. Consequently it can be said that it is better to use ILP models in situations of problems with small size, requirement for solution time is not very important and preparing/constructing the model is easy, etc.

3.3.1.1.2 Branch and Bound Method. Branch and bound (B&B) method that is proposed by Land & Doig (1960) is an exact algorithm for different optimization problems. The method searches for the optimal solution with a separating process named as “branching” and an investigation phase named as “bounding”. While branching operation divides the problem into sub problems, bounding operation decides which branch to be concluded.

In B&B method, the problem is examined with sub problems and bounds are determined for each sub problem not to proceed on a bad resulting solution. The main idea of the method is that bounds help to compare new solutions with the solutions previously found and avoid making a search around a solution that will clearly not yield an optimal solution (Lawler & Wood, 1966).

3.3.1.1.3 Dynamic Programming. In dynamic programming approach, the problem is examined under sequential steps. After completion of a step, following step starts from the solution of the previous step. After solving small sub problems, large sub problems are solved in accordance with solutions of the small sub problems.

3.3.1.1.4 Cutting Planes Algorithm. Cutting planes method tries to solve the problem with the assumption that is based on finding integer values for variables and objective function. In this method, firstly, LP relaxation of the problem is solved to obtain a lower bound. Then a source row of the solution from LP relaxation is chosen. A *cutting plane* which removes a set of non-integer solutions is determined according to the source row. By adding cutting plane to the simplex tableau, LP problem is solved. If all of the variables are integer, then the optimum solution is found (Winston, 2004).

3.3.1.1.5 Branch and Cut Method. The branch and cut method emerges from the mix of the B&B method and the cutting planes method. Bounds in the B&B method are determined according to the result of the cutting planes; thus it ensures searching for optimum solution in a smaller feasible solution space (Mitchell, 2002; Winston, 2004).

3.3.1.2 Heuristic Methods

In some situations where the exact solution techniques are impractical for application, heuristics for TSP try to reach near optimal solutions. TSP heuristics can be classified into two: *construction heuristics* and *improvement heuristics*.

Tour construction heuristics are heuristics that effort to form the tour by inserting nodes to the starting node step by step, as the name suggests. Some of the most used construction heuristics are nearest neighbour algorithm, insertion algorithms, heuristics based on spanning trees (e.g. Christofides algorithm) and savings methods.

In addition to construction phase, the solution found may not be satisfying and a better solution can be searched for. Therefore, improvement heuristics are used to enhance the quality of the solution on hand. For the improvement heuristics, common methods can be said as 2-opt and 3-opt heuristics with their variants and Lin-Kernighan type exchange (Johnson & McGeoch, 1997; Jünger, Reinelt & Rinaldi, 1995). As the problem size grows, it is getting difficult to solve the problem and at this point integration of some approaches becomes important.

3.3.1.2.1 2-opt Heuristic. 2-opt local search algorithm is proposed by Croes (1958) as a method for solving TSP. In the algorithm, the main idea is that two edges are removed and reconnected in a different way to obtain a new Hamiltonian cycle which could yield a better result. With this approach, different movements may result less costly solutions.

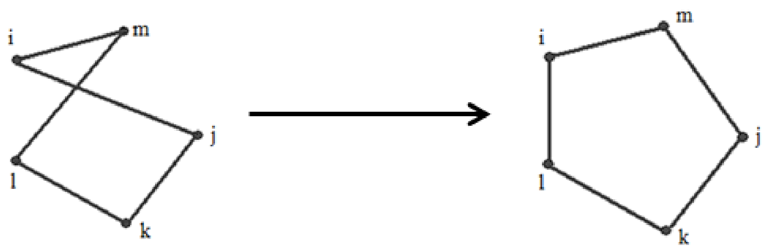

Figure 3.2 2-opt move

In Figure 3.2, 2-opt move can be seen. Here edge *(i,j)* and edge *(l,m)* are removed then resulting two paths are merged in a new way so that it would not generate sub tours. In this move, sequence in one of the two parts of the modified route is arranged in a reverse way. For example in Figure 3.2, initial tour is (*i, j, k, l, m*) and after 2-opt move the tour becomes (*i, l, k, j, m*). The algorithm continues to search until a tour which has less length than previous tour is found.

3.3.1.2.2 3-opt Heuristic. 3-opt algorithm that firstly introduced by Bock (1958) is a more comprehensive search algorithm than 2-opt move. In this algorithm three edges are deleted and re-joined the tour in a new way. In 2-opt move, there is only one combination of the edges that would yield a feasible solution after removal of the two edges. On the other hand in 3-opt move, it is possible to bring out seven different tours except the initial tour. It is stated in (Lin, 1965) that number of the new tours derived from exchanging three edges with each other in a tour with n vertices is $\binom{n}{3}$.

In Figure 3.3, 3-opt move can be seen. Number of the new tours that will emerge with the reconnection after removal of three edges in the tour is seven. In Figure 3.3 only two of these moves are displayed.

In comparison with the 2-opt heuristic, 3-opt heuristic searches more space and evaluates more neighbourhoods of the current solution. Therefore the computational time of the algorithm increases when the 3-opt heuristic is used.

A generalization of 2-opt and 3-opt heuristics can be said as *k-opt*, where *k* represents number of the removed edges. It is not generally recommended to use large values of *k* in *k*-opt heuristic; because as the value of *k* increases, the computational time also increases. Therefore, it can be said it is practical to use values of k<4 in the experiments (Jünger et al., 1995; Lin, 1965).

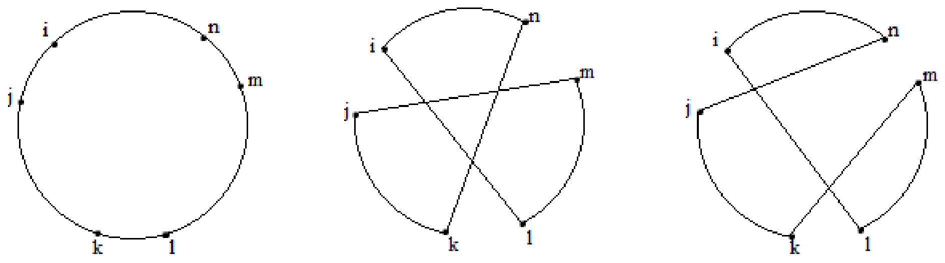

Figure 3.3 Two example of 3-opt moves. Assuming the left one is the initial tour and the following two are modified tours.

3.3.1.2.3 Lin-Kernighan Type Exchange. While in 2-opt and 3-opt heuristics number of the nodes that will exchange with each other is certain, in Lin Kernighan type exchange (Lin & Kernighan, 1973), this number is determined dynamically. The method is based on a search technique in which a number of different modifications are made. Some of the modifications applied may not necessarily lead better solutions. At each iteration, the best resulting solution is accepted as the new solution.

3.4 Methodology of the Proposed Approach

Since the introduction of ACO and its algorithms, continuous improvements about the algorithm have been made in the literature. TSP plays an important role in the ACO literature, because the first ACO algorithm, ant system (AS) (Dorigo, Maniezzo, & Colorni, 1991) is applied to this NP-hard problem. After emergence of the AS; elitist AS (Dorigo, Maniezzo, & Colorni, 1996), ant-q (Gambardella & Dorigo, 1995), ant colony system (Dorigo & Gambardella 1997; Gambardella & Dorigo, 1996), max-min ant system (Stützle & Hoos, 2000), rank-based AS (Bullnheimer, Hartl, & Strauss, 1997), best-worst AS (Cordón, Fernández de Viana, & Herrera, 2002), population-based ACO (Guntsch & Middendorf, 2002) and parallelized genetic ant colony system (Chen & Chien, 2011) algorithms that have been experimented for TSP are introduced.

3.4.1 Max-Min Ant System Algorithm

Max-min ant system (MMAS) algorithm is an ACO algorithm in which pheromone update rules are different from other ACO algorithms. In MMAS, pheromone updating is made by only the ant that has the best tour length, and this operation is implemented by use of Eq. (3.5), Eq. (3.6) and Eq. (3.7). Pheromone levels are restricted between maximum and minimum levels, $\tau_{max,}$ and τ_{min}, respectively (Dorigo et al., 2006; Stützle & Hoos, 2000).

$$\tau_{ij} \leftarrow \left[(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{best}\right]_{\tau_{\min}}^{\tau_{\max}} \tag{3.5}$$

$$[x]_b^a = \begin{cases} a & \text{if } x > a, \\ b & \text{if } x < b, \\ x & \text{otherwise}, \end{cases} \tag{3.6}$$

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/L_{best} & \text{if } (i,j) \text{ belongs to the best tour}, \\ 0 & \text{otherwise}, \end{cases} \tag{3.7}$$

L_{best} means global minimum length and while calculation of the bounds, minimum and maximum pheromone levels are found via Eq. (3.8) and Eq. (3.9). In Eq. (3.9), *avg* is accepted as *n/2.*

$$\tau_{\max} = \frac{1}{(1-\rho)\cdot L_{best}} \tag{3.8}$$

$$\tau_{\min} = \frac{\tau_{\max}\cdot(1-\sqrt[n]{0.05})}{(avg-1)\cdot\sqrt[n]{0.05}} \tag{3.9}$$

3.4.2 Details of Proposed Method

In this study, TSP is solved via MMAS algorithm, then result of the MMAS is accepted as an initial tour and tried to be improved by 2-opt and 3-opt local search heuristics. In Figure 3.4, the MMAS algorithm is shown.

```
Max-Min Ant System Algorithm
  • Generate $m$ ants and assign each of them to one node starting from node 1
  • Determine initial parameters $\alpha, \beta, \rho,$ and pheromone levels ($\tau_{ij}$) on each edge
  for $t$ = 1: iteration no
      for $k$ = 1 : $m$
          • Compute the probability $p^k_{ij}$ of selecting next node
          • Determine length of the complete tour
      end
      Choose the ant gives best tour length $L_{best}$
      for $i$ = 1 : $n$
          • Update pheromone amounts on all of the edges with consideration
            of $\tau_{max}$ and $\tau_{min}$ bound conditions
      end
  end
```
Figure 3.4 The structure of the MMAS algorithm

2-opt is a local search heuristic in which two nodes are exchanged to obtain better solutions. From the definition, algorithm may be misunderstood and be confused with two-exchange move. At this point, 2-opt differs from two-exchange move with the following: in 2-opt move, sequence in one of the two parts of the modified route is arranged in a reverse way.

3-opt algorithm is a more comprehensive search algorithm than 2-opt move. In 3-opt algorithm, number of the possible tours after removal of three edges is seven while in 2-opt move, there is only one other combination of the edges that would yield a feasible solution after removal of two edges. In these approaches, it is aimed to improve the quality of the solution at the end of the search process.

3.5 Computational Study

Initial parameter settings that are considered in MMAS are shown in Table 3.1. In the application of the algorithm, it is decided to make 10 iterations for the MMAS, because it is found in the experiments that, the solutions found in different iterations converge at that point. Besides, as the problem size grows, elapsed time is increasing explicitly.

As a start point, each ant is assigned to one node randomly, then for every ant, the route which has minimum length is found by use of Eq. (2.3) and Eq. (2.4). Pheromone update is implemented using Eq. (3.5), Eq. (3.6) and Eq. (3.7) with consideration of the parameters. This procedure is implemented at each iteration.

Table 3.1 Initial parameters for the algorithm

Parameter	**Value**
Impact of pheromones (α)	2
Impact of distance (β)	1
Pheromone evaporation rate (ρ)	0.2
Initial pheromone level (τ)	100
Number of ants (m)	Number of cities (n)

The first application is MMAS algorithm with 2-opt heuristic. With the best route taken from result of the MMAS, 2-opt algorithm is started. Nodes which will exchange with each other are determined randomly by generation of random numbers, then 2-opt procedure is applied to the solution. For 2-opt, iteration number is determined as *(n*4000)* to be able to make an extensive search.

The second application is MMAS algorithm with 3-opt heuristic. The initial route for the 3-opt is result of the MMAS. Generated random numbers are used for replacement of the nodes and then 3-opt procedure that searches seven different combination of the new tour is applied to the solution. For 3-opt, iteration number is determined as *(n* x *1500)* according to result of the experimental study. In the experiment, at first, the iteration number is determined as (*n* x *4000*) as in the 2-opt experiment. After an experiment with a small size of instance, it is decided to decrease the iteration number; because the computational effort is increased obviously. In the next step, the iteration number is set to *(n* x *3000)* and to make a decision, two instances are examined and their results are used for interpretation.

In Figure 3.4 and 3.5, the experimental study for eil51 and pr76 instances can be seen. On the vertical axis, length of the solution is displayed. The horizontal axis indicates the iteration number. For each iteration, the best solution is displayed. Since the convergence starts early points, the algorithm spends too much computational effort and it is seen that iteration number must be decreased.

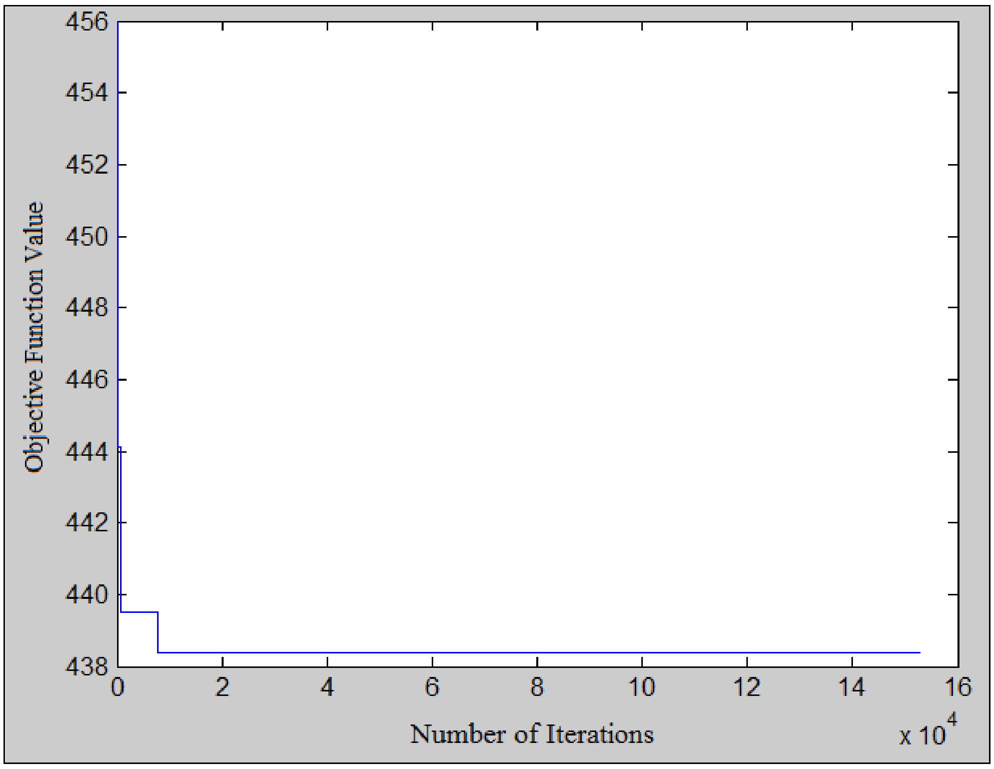

Figure 3.4 The result of eil51 experiment

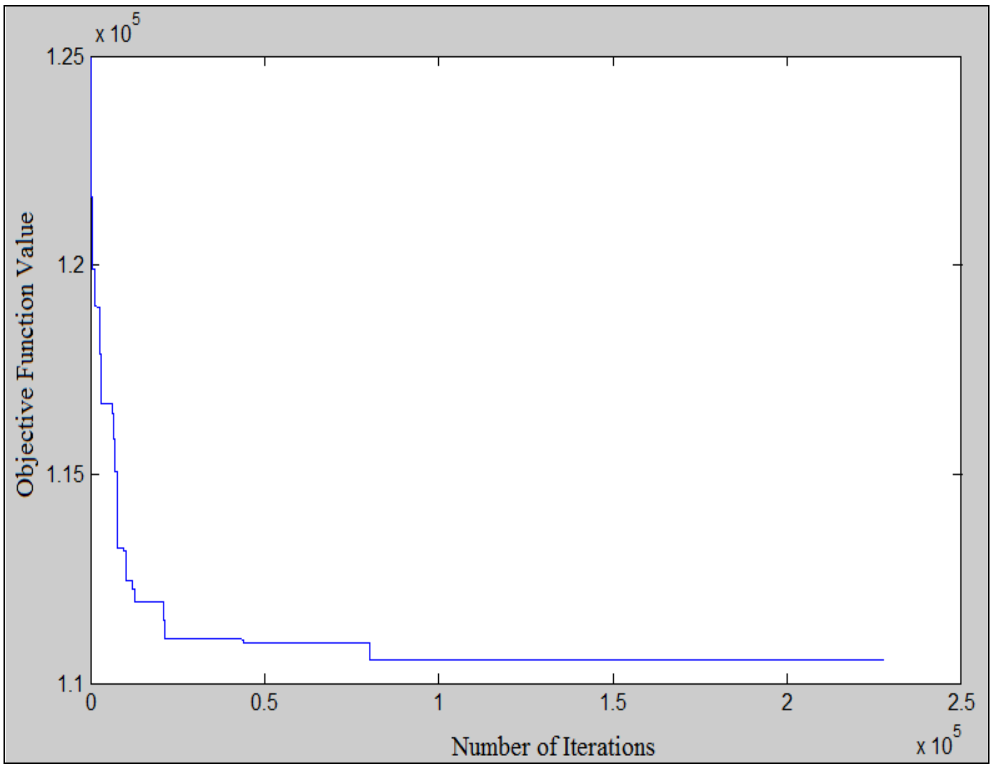

Figure 3.5 The result of pr76 experiment

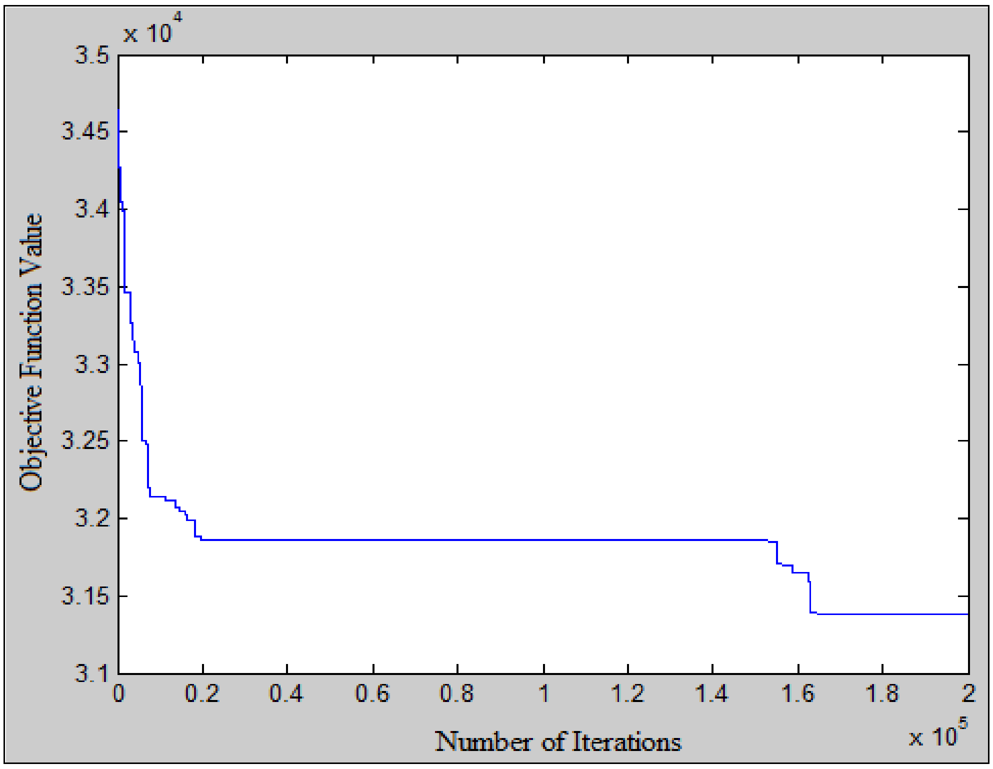

Figure 3.6 The result of kroB200 experiment

As the last experiment, kroB200 instance is examined. The iteration number is set to *(n* x *1000)* in this experiment. In Figure 3.6, result of this study can be seen. After evaluation of the experiment, it is decided to increase the iteration number; because the convergence is not seemed to be started. Finally, iteration number is set to *(n* x *1500).*

In application, algorithm is coded in MATLAB 7.7.0 software and tested on a computer with Pentium Dual-Core E2160, 1.80 GHz processor and 4 GB RAM. Twenty six symmetric TSP data sets with Euclidean distances are used in the computations and these data sets are taken from the TSPLIB (TSPLIB, 2008).

3.6 Results

As a start point to the proposed algorithm, result of the MMAS algorithm ensures an initial route and after that the result become more acceptable when 2-opt or 3-opt local search heuristic is implemented.

Table 3.2 shows the results of the first algorithm applied to the well-known data sets. Test of each data set is made for five times then best, worst and average lengths are stated in Table 3.2. In this table, optimal means the optimal solution for the data set. Percentage deviation of the best found solution to the optimal solution is calculated as Eq. (3.10). After the implementation of the proposed algorithm, in the first application, it is found that the percentage deviations of the best solution to the optimal solution are between 0 and 8.

$$\%Dev = \frac{Best - Optimal}{Optimal} \tag{3.10}$$

Figure 3.7 is the graphical interpretation of the solutions found in the first computational study. OPTIMAL is the optimal solution of the instance, BEST is the best found solution with MMAS+2-opt, AVG is the average solution and the WORST is the worst solution found.

Table 3.2 Results of MMAS+2-opt

TSP Instances	Optimal	Best	Worst	Average	Deviation of Best Sol.
wi29	27603.00	27603.00	28029.00	27885.60	0.0000
dj38	6656.00	6664.00	6664.00	6664.00	0.0010
eil51	426.00	434.75	439.52	437.80	0.0210
berlin52	7542.00	7598.40	7904.50	7682.54	0.0070
st70	675.00	686.11	693.65	689.75	0.0160
eil76	538.00	573.07	579.39	576.00	0.0650
pr76	108159.00	109150.00	116510.00	113826.00	0.0090
kroA100	21282.00	22006.00	22166.00	22084.00	0.0060
kroB100	22141.00	22923.00	23114.00	23035.00	0.0130
kroC100	20749.00	21242.00	21438.00	21324.80	0.0470
kroD100	21294.00	22187.00	22367.00	22311.60	0.0350
kroE100	22068.00	22523.00	22632.00	22580.00	0.0240
eil101	629.00	673.24	683.44	677.69	0.0700
lin105	14379.00	14634.00	15077.00	14792.80	0.0180
pr107	44303.00	44575.00	44659.00	44618.40	0.0590
pr124	59030.00	59799.00	60293.00	60028.00	0.0100
ch130	6110.00	6470.10	6560.20	6522.42	0.0470
pr136	96772.00	101360.00	102420.00	101942.00	0.0340
pr144	58537.00	60569.00	60569.00	60569.00	0.0400
ch150	6528.00	6593.90	6657.90	6616.08	0.0190
kroA150	26524.00	27593.00	28334.00	27944.00	0.0350
kroB150	26130.00	27002.00	27565.00	27272.40	0.0330
pr152	73682.00	75445.00	76395.00	75995.20	0.0800
kroA200	29368.00	29939.00	30199.00	30070.00	0.0240
kroB200	29437.00	31789.00	32096.00	31999.20	0.0420
lin318	42029.00	44024.00	47045.00	45307.00	0.0210

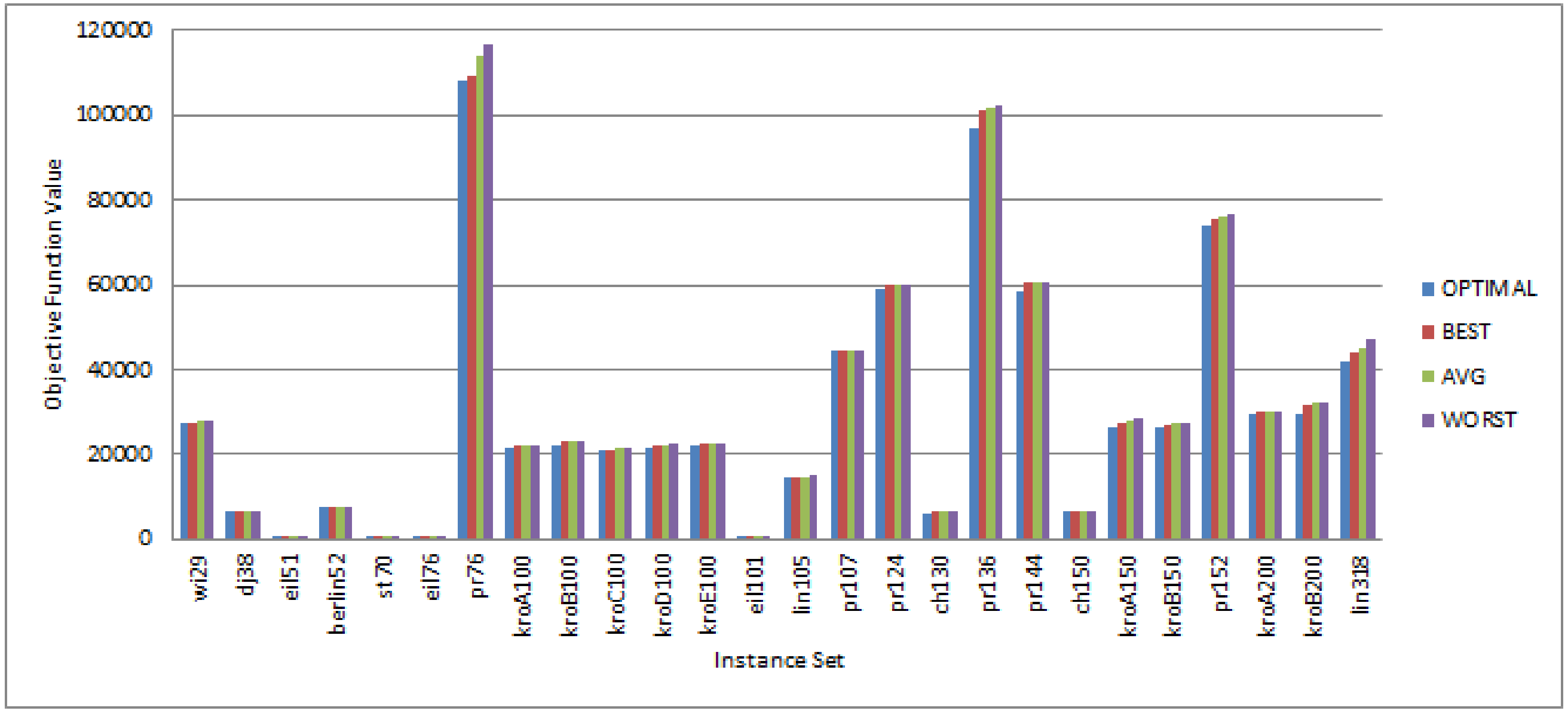

Figure 3.7 Results of the first computational study

As the second study, the MMAS+3-opt algorithm is applied to the data sets. Table 3.3 shows the results of the application to the TSP instances. In the second application, it is found that the percentage deviations of the best solution to the optimal solution are resulted between 0.05 and 6 and the deviations from the best solutions are decreased on average. Figure 3.8 is the results of the solutions found in the second computational study.

Figure 3.9 displays the percentage deviations of the first computational study. Percentage deviations of the BEST found solutions and AVG found solutions from the optimal solutions are shown as two distinct lines. From Figure 3.9, it can be seen that computations of wi29 and dj38 instances result with the lowest deviations and kroB200 instance set gives the highest deviations. Furthermore, it is noticeable that except a few instances, the percentage deviations are increasing as the instance size grows.

In Figure 3.10, percentage deviations in the second computational study can be seen. From Figure 3.10, it can be seen that computations of dj38 instance result with the lowest deviations and kroB200 instance set again gives the highest deviations. Furthermore, it can be said again that except a few instances, the percentage deviations are increasing as the instance size grows.

Figure 3.11 and Figure 3.12 display the comparison of the algorithms according to best found solutions and average found solutions. Except a few instances, 3-opt heuristic gives better results than the 2-opt heuristic. This can be explained as; searching area of the 3-opt move is wider than 2-opt move, so it can find better results with this searching process.

In Figure 3.13, resulting tour of the berlin52 instance after MMAS is given. The left side of the tour is improved after the 2-opt heuristic, and the new tour is given in Figure 3.14. At last, result of the MMAS+3-opt is shown in Figure 3.15. Following figures, Figure 3.16 to Figure 3.24, are some example tours of the solutions found for st70, eil101 and kroA200 instances.

Table 3.3 Results of MMAS+3-opt

TSP Instances	Optimal	Best	Worst	Average	Deviation of Best Sol.
wi29	27603.00	27872.36	28300.25	27987.45	0.0098
dj38	6656.00	6659.43	6659.43	6659.43	0.0005
eil51	426.00	437.56	438.59	437.93	0.0271
berlin52	7542.00	7598.44	7847.57	7696.30	0.0075
st70	675.00	684.14	690.43	687.33	0.0135
eil76	538.00	558.55	576.31	567.58	0.0382
pr76	108159.00	108881.96	110131.27	109691.20	0.0067
kroA100	21282.00	21365.54	22018.26	21670.63	0.0061
kroB100	22141.00	22362.35	22789.34	22619.24	0.0084
kroC100	20749.00	21204.52	21336.80	21267.37	0.0256
kroD100	21294.00	21566.69	22329.95	21910.73	0.0014
kroE100	22068.00	22261.69	22634.21	22385.21	0.0128
eil101	629.00	660.94	673.09	667.34	0.0508
lin105	14379.00	14795.47	14919.66	14851.07	0.0290
pr107	44303.00	44575.24	44840.79	44632.98	0.0391
pr124	59030.00	59523.59	60292.71	59900.41	0.0047
ch130	6110.00	6349.06	6525.69	6397.43	0.0374
pr136	96772.00	99244.57	101946.89	100904.45	0.0039
pr144	58537.00	58620.69	60501.81	59421.42	0.0385
ch150	6528.00	6558.66	6629.74	6589.01	0.0147
kroA150	26524.00	27545.14	27964.18	27778.45	0.0100
kroB150	26130.00	26825.27	27494.79	27081.68	0.0266
pr152	73682.00	74627.10	75966.30	75374.79	0.0619
kroA200	29368.00	29799.92	30206.90	29978.13	0.0220
kroB200	29437.00	31259.47	32061.15	31672.30	0.0128
lin318	42029.00	43598.92	44388.35	44154.55	0.0088

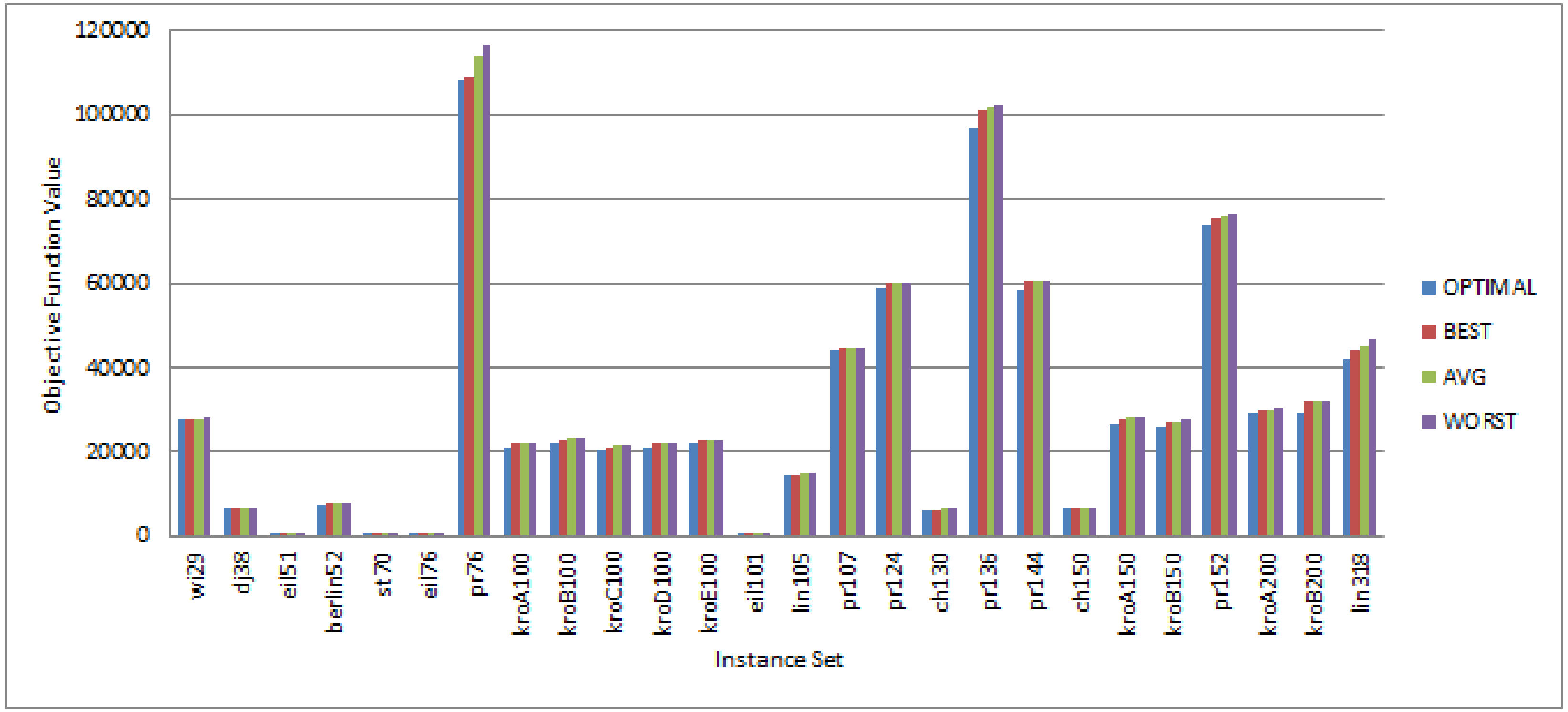

Figure 3.8 Results of the second computational study

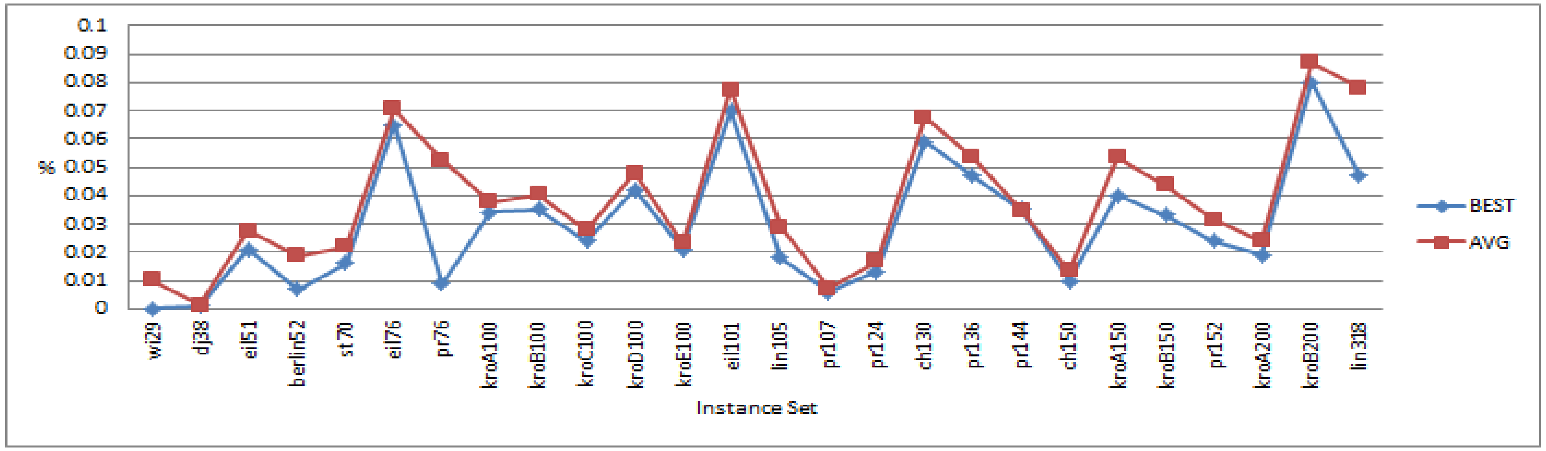

Figure 3.9 Percentage deviations in the first computational study

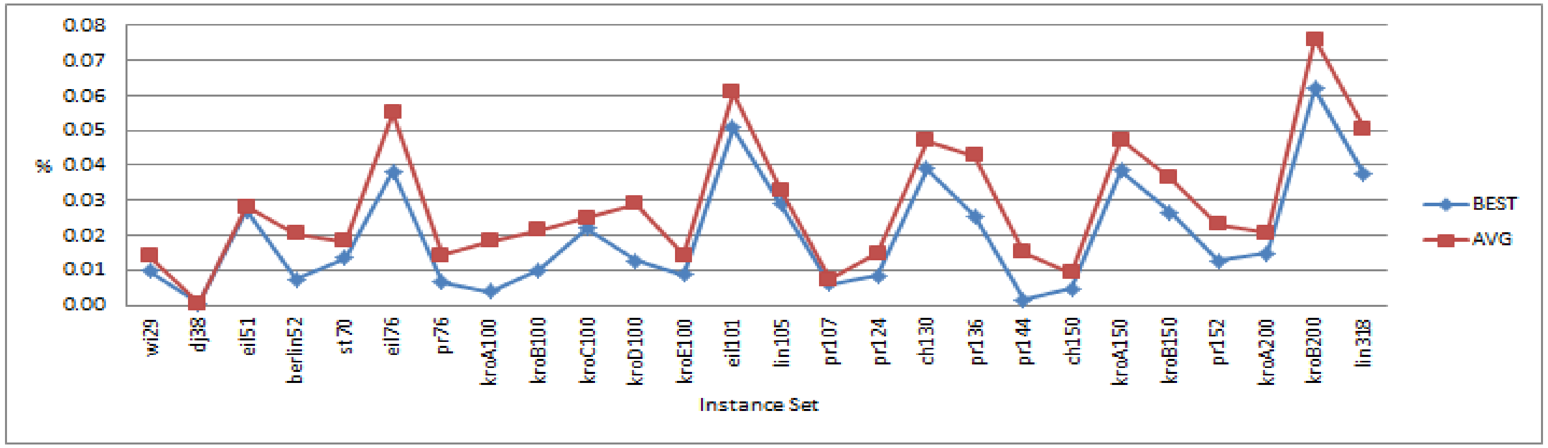

Figure 3.10 Percentage deviations in the second computational study

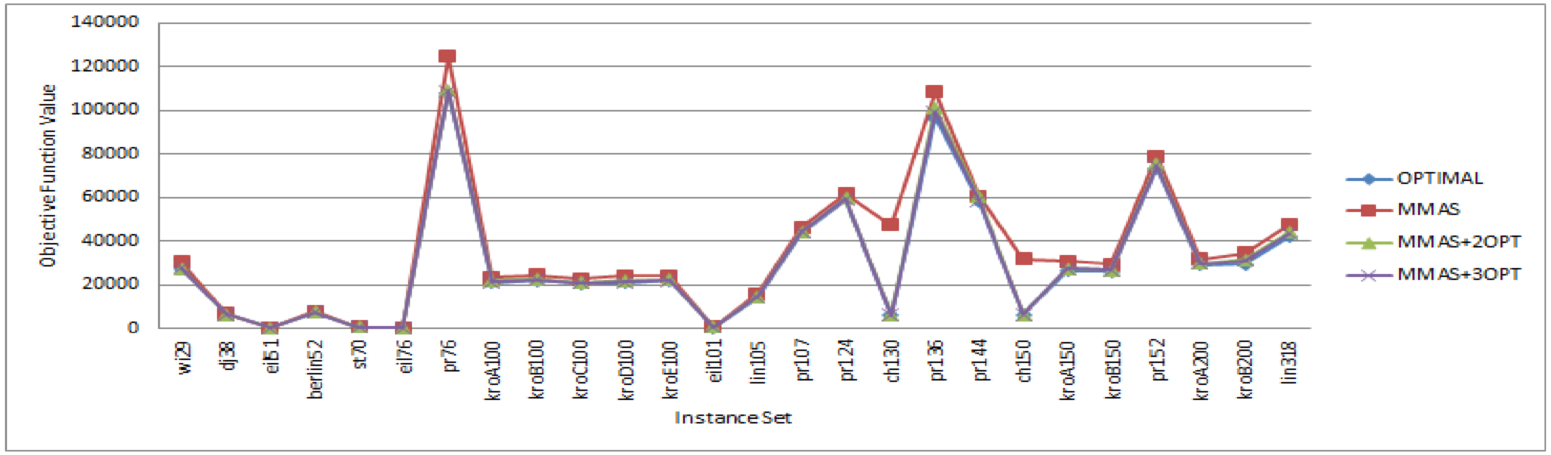

Figure 3.11 Comparison of the best found solutions

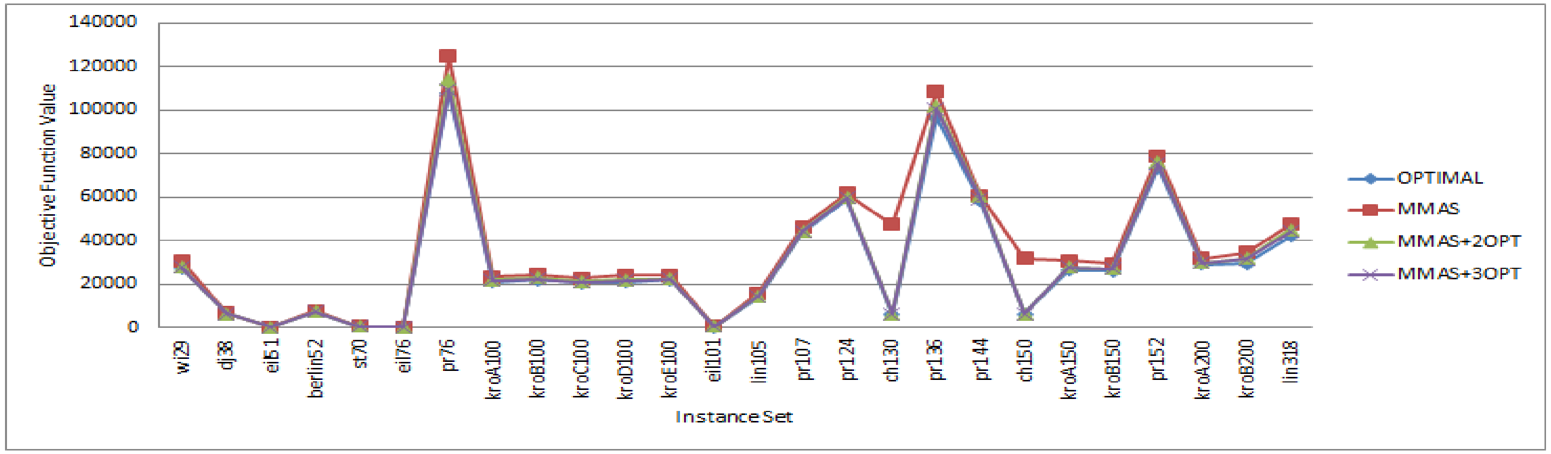

Figure 3.12 Comparison of the average found solutions

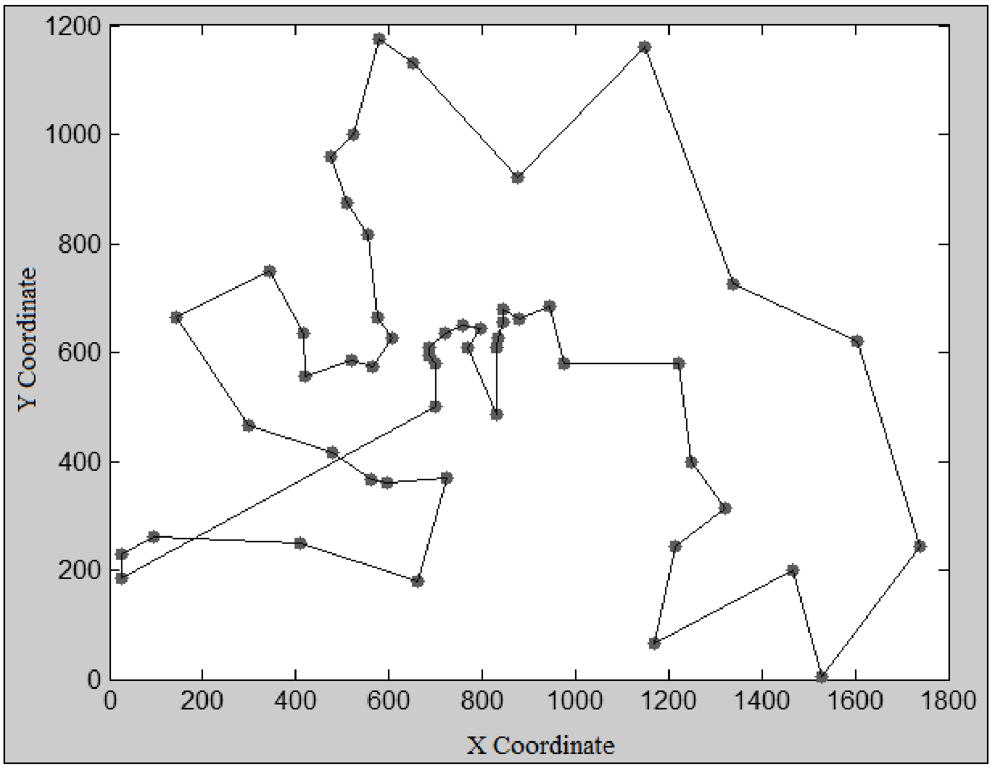

Figure 3.13 Solution of the berlin52 instance with MMAS. Tour length is 8051.9.

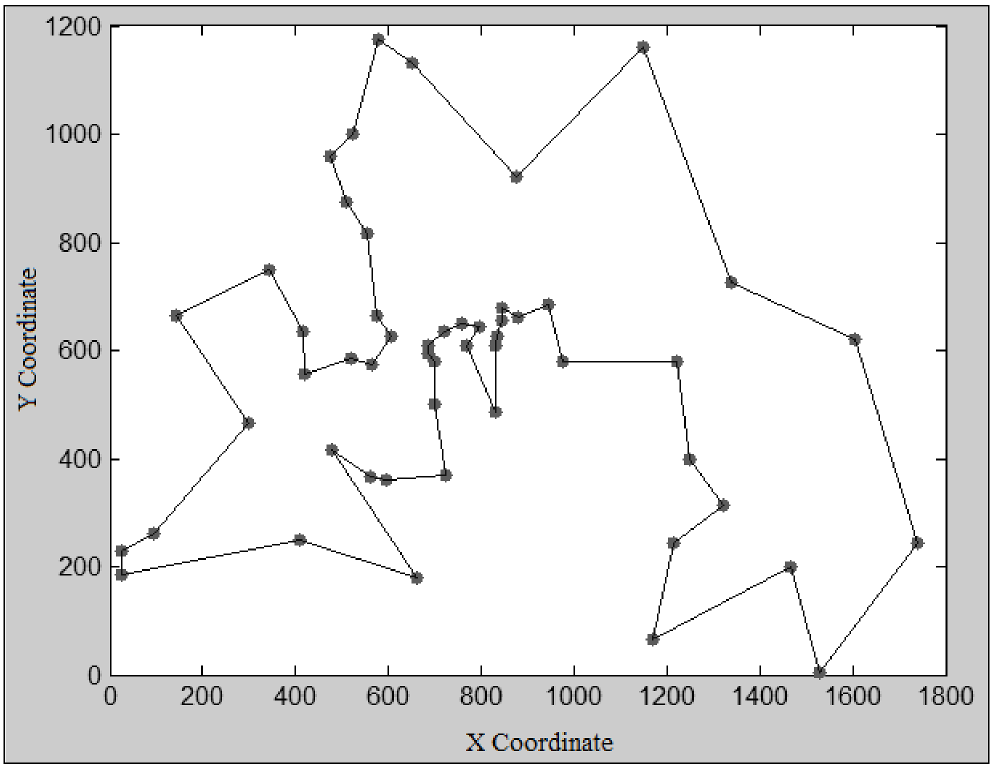

Figure 3.14 An example solution for berlin52 instance with MMAS+2opt. Tour length is 7713.

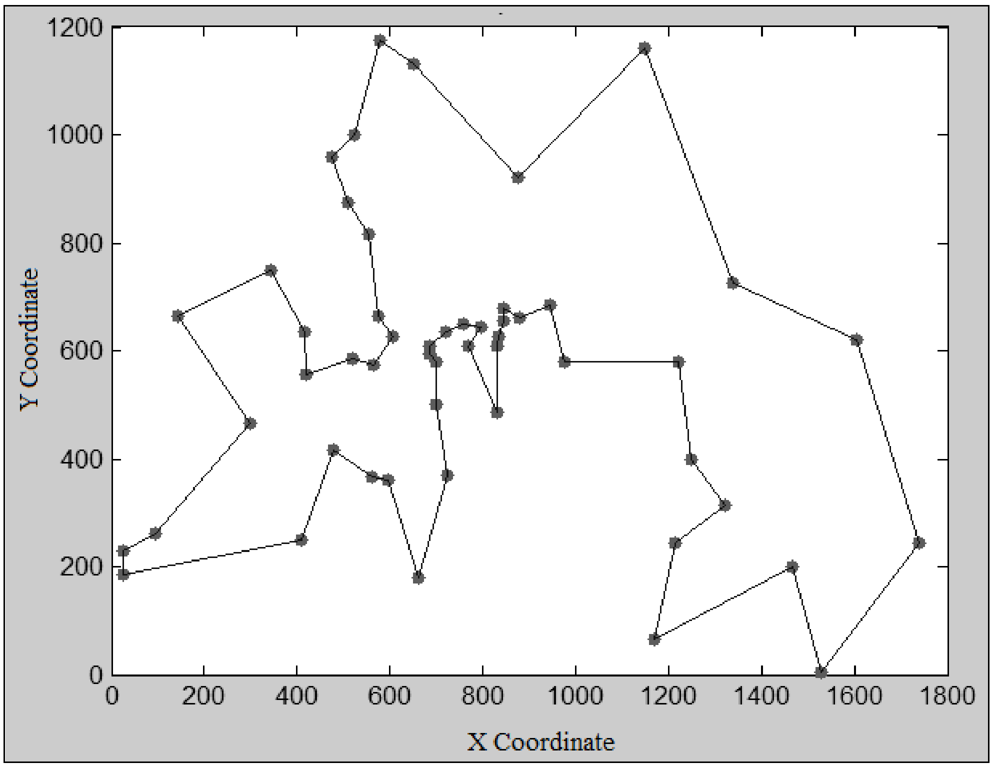

Figure 3.15 An example solution for berlin52 instance with MMAS+3opt. Tour length is 7598.4.

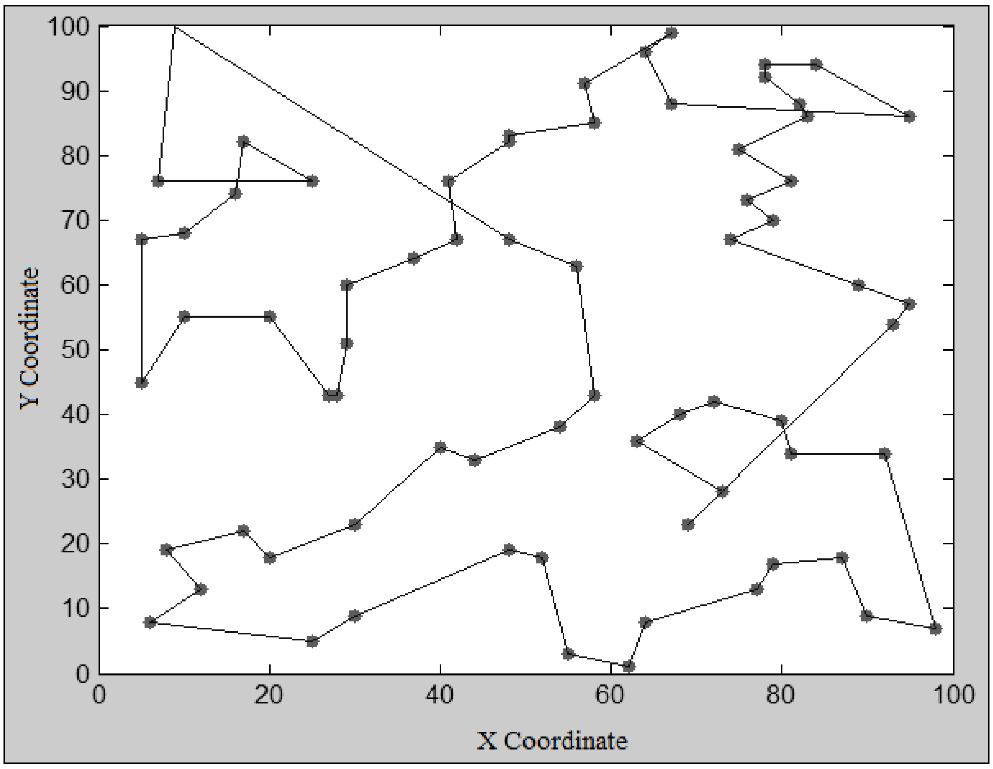

Figure 3.16 An example solution for st70 instance with MMAS. Tour length is 746.86.

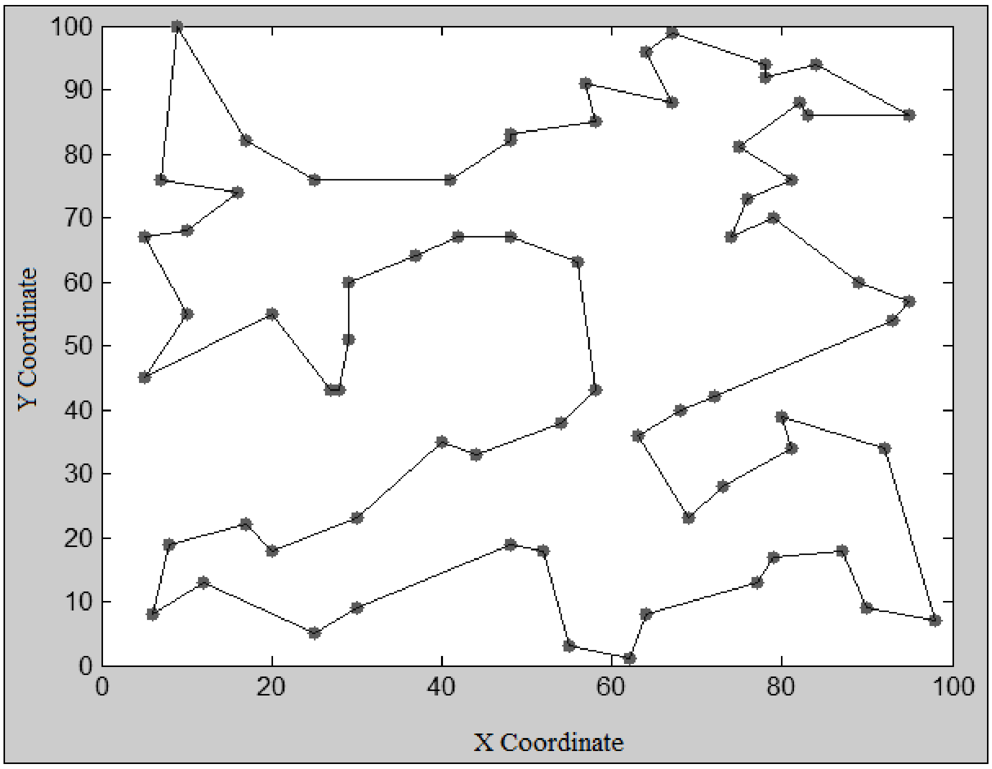

Figure 3.17 An example solution for st70 instance with MMAS+2opt. Tour length is 689.99.

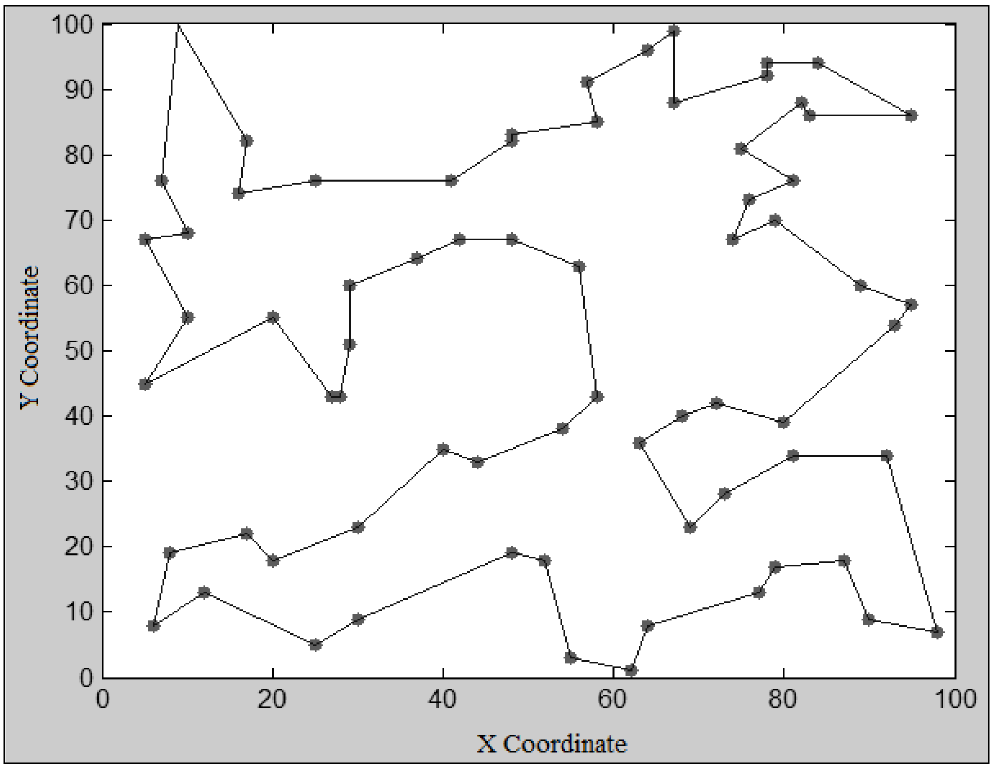

Figure 3.18 An example solution for st70 instance with MMAS+3opt. Tour length is 684.14.

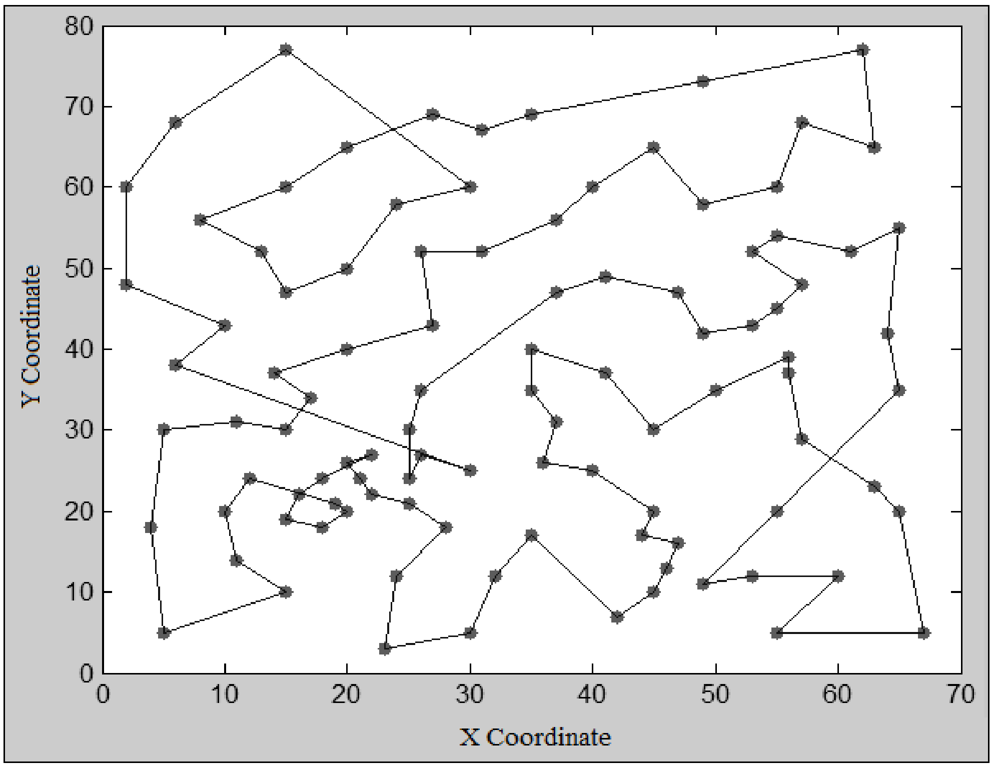

Figure 3.19 An example solution for eil101 instance with MMAS. Tour length is 713.33.

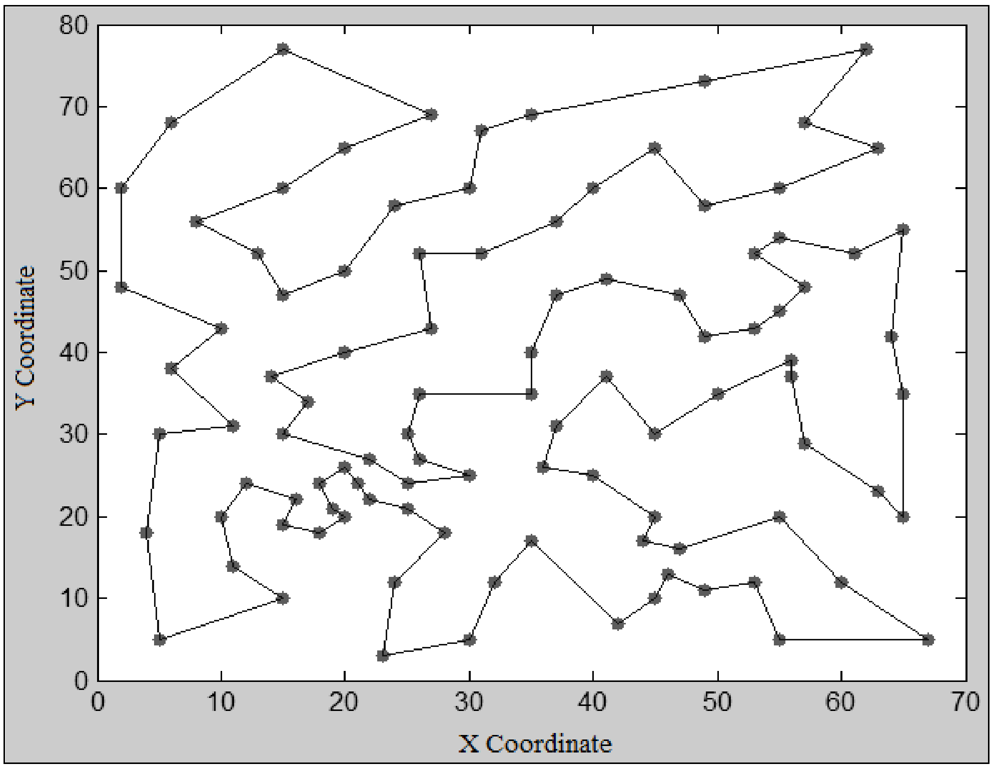

Figure 3.20 An example solution for eil101 instance with MMAS+2opt. Tour length is 675.49.

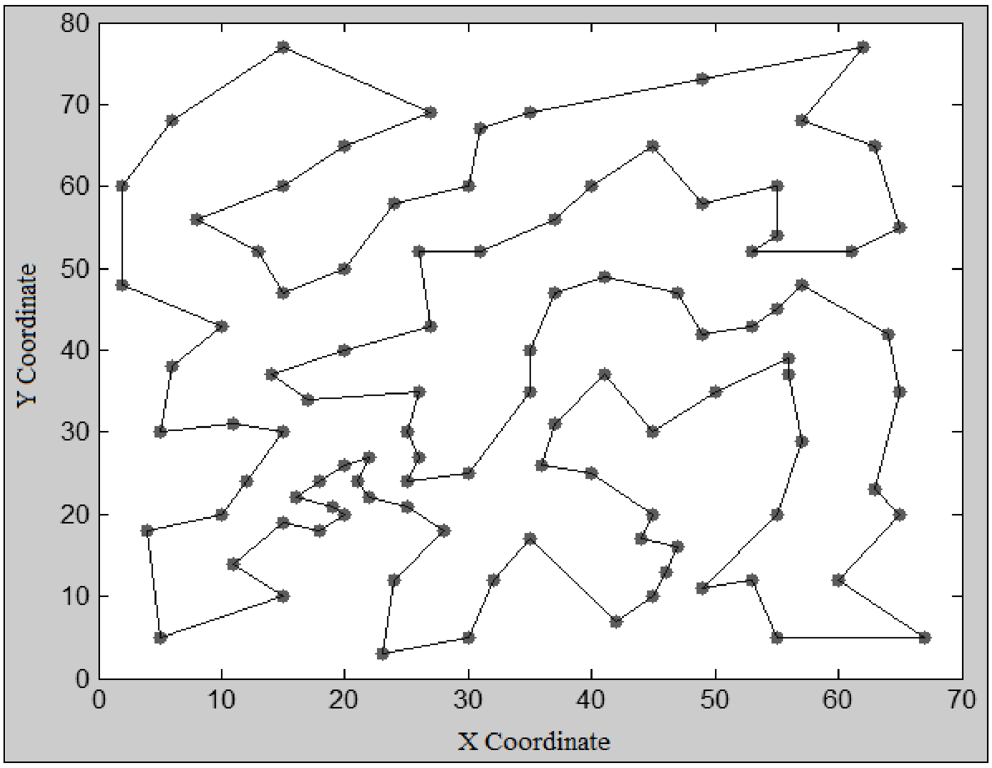

Figure 3.21 An example solution for eil101 instance with MMAS+3opt. Tour length is 670.63.

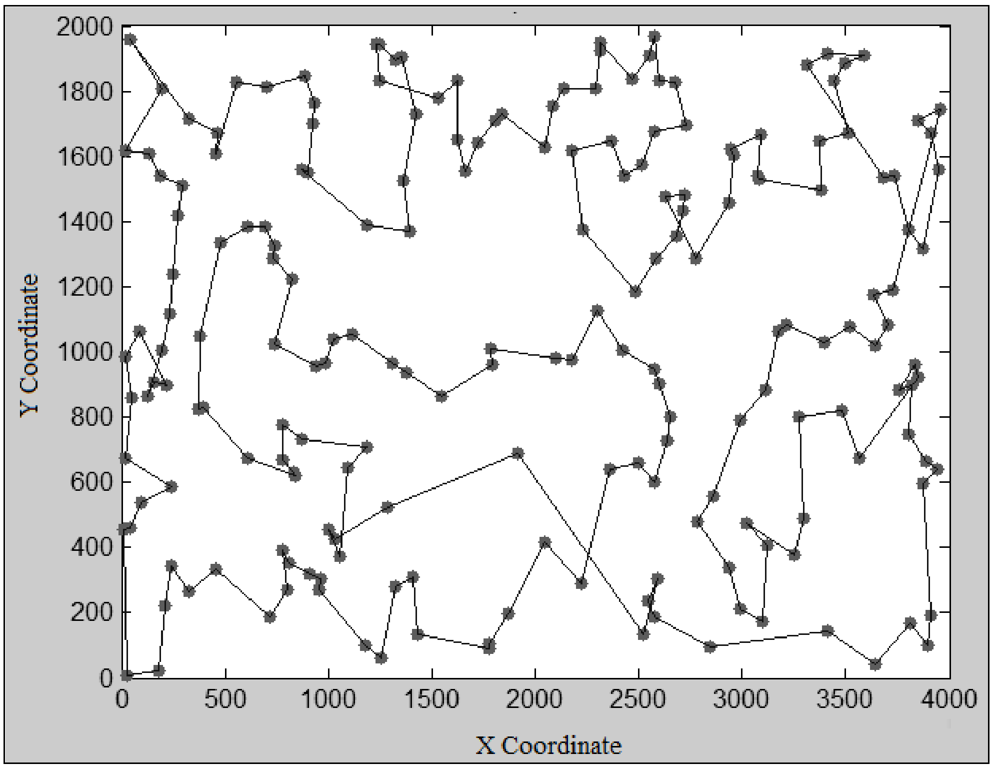

Figure 3.22 An example solution for kroA200 instance with MMAS. Tour length is 32055.

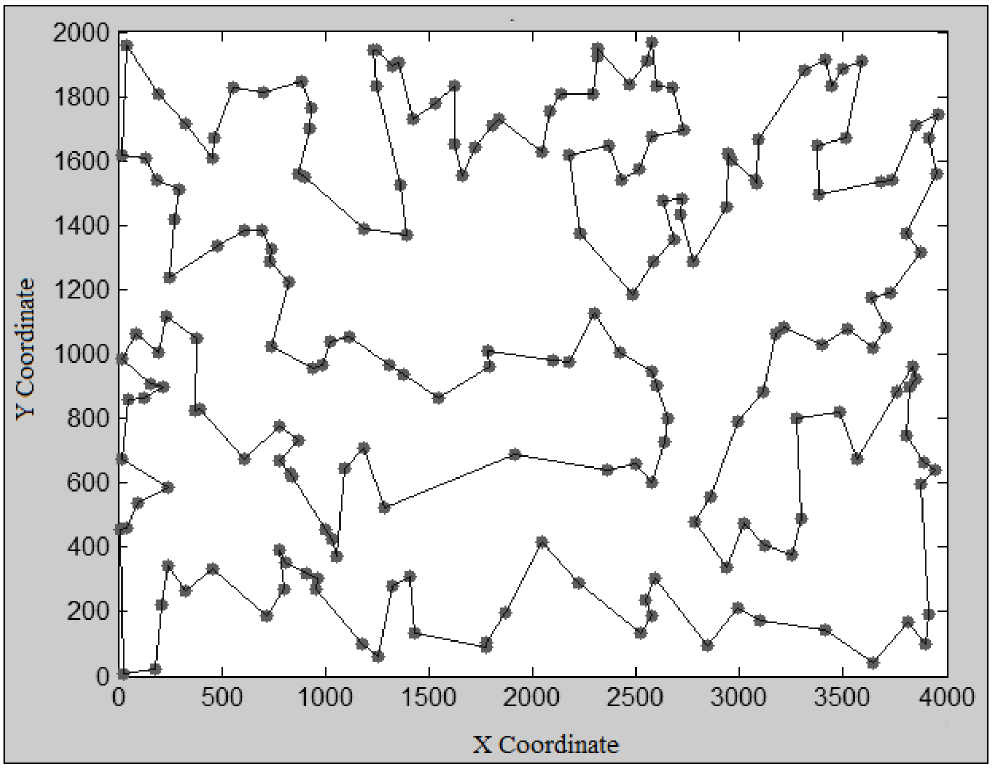

Figure 3.23 An example solution for kroA200 instance with MMAS+2opt. Tour length is 29843.

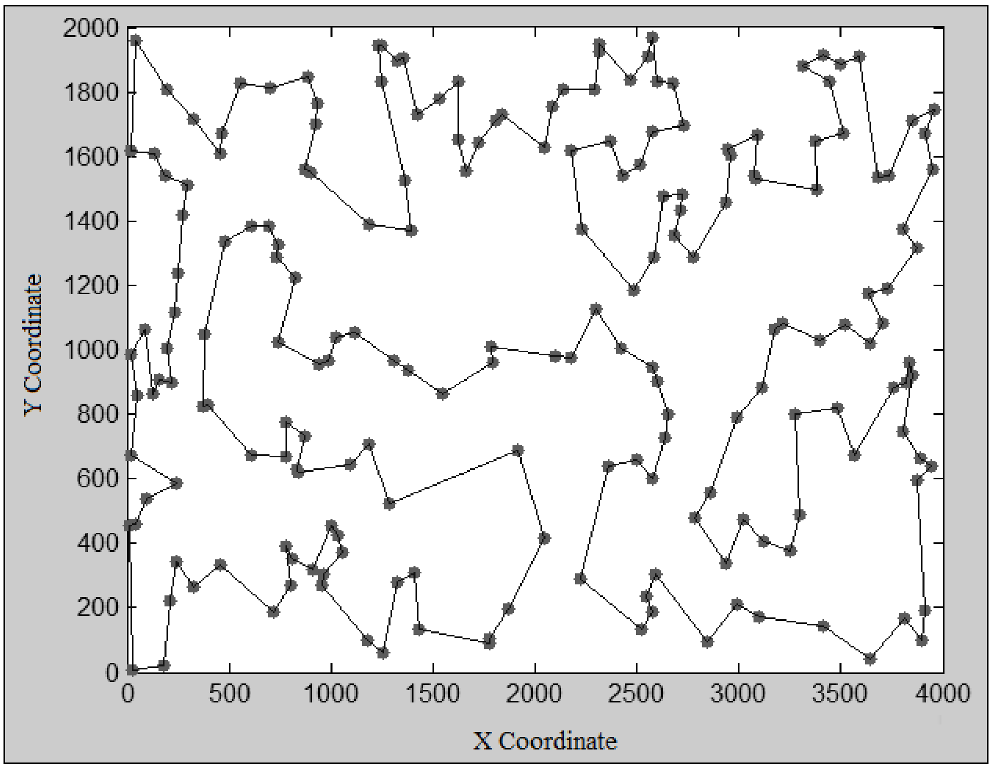

Figure 3.24 An example solution for kroA200 instance with MMAS+3opt. Tour length is 29982.

In the recent studies related with ACO, ant colony algorithms for the time dependent vehicle routing problem with time windows (Balseiro, Loiseau & Ramonet, 2011); parallelized genetic ant colony systems (Chen & Chien, 2011), elite-guided continuous ant colony optimization (Juang & Chang, 2011) and mutated ant colony optimization (Zhao, Wu, Zhao & Quan, 2010) algorithms have been considered. From these studies, it can be seen that future research on ACO focuses on optimization problems that include stochasticity, dynamic data modifications, and multiple objectives (Dorigo et al., 2006). In addition, applications of real projects, studies with large-scale instances stand as challenging areas. It is aimed to make dynamic parameter choices and decisions to obtain better solutions in the future studies.

CHAPTER FOUR
A HYBRID GENETIC ALGORITHM FOR TRAVELING REPAIRMAN PROBLEM

4.1 Introduction to Traveling Repairman Problem

Despite the fact that cost minimization is the common objective of product and service suppliers, customer oriented view is of vital importance in some cases. At this point, traveling repairman problem (TRP) is examined to ensure customer satisfaction. As the name suggests, in TRP, a repairman wants to serve all of the customers exactly once such that total waiting time of the customers is minimized. In other words, TRP is the problem of finding a Hamiltonian path in which the objective is to minimize total waiting time of all customers that are situated at different locations. Real life applications of TRP can be said as a delivery situation such as a pizza delivery problem, disk head scheduling, or the most important, a disaster situation. This problem could be viewed as a variation of the well-known TSP. In TRP, total waiting time of the customers denotes sum of the distances from an origin to every customer node (including origin) along the cycle (Fischetti, Laporte, & Martello, 1993).

In the literature, this problem is also known with other names such as the minimum latency problem (MLP) which uses latency expression that means the distance travelled before first visiting that customer (Blum et al., 1994) and the deliveryman problem (DMP). TRP is considered as cumulative travelling salesman problem (CTSP) in (Bianco, Mingozzi, & Ricciardelli, 1993), and Lucena (1990) examined the problem under the more general time dependent travelling salesman problem (TDTSP) concept.

4.2 Problem Definition

TRP is a NP-hard problem as TSP, so a meta-heuristic solution approach is used to be able to solve real life problems in a reasonable time period. This problem is based on customer needs; therefore the problem can be adapted into real life applications where customer satisfaction is important. An example of this can be a delivery problem in an emergency situation such as a disaster case.

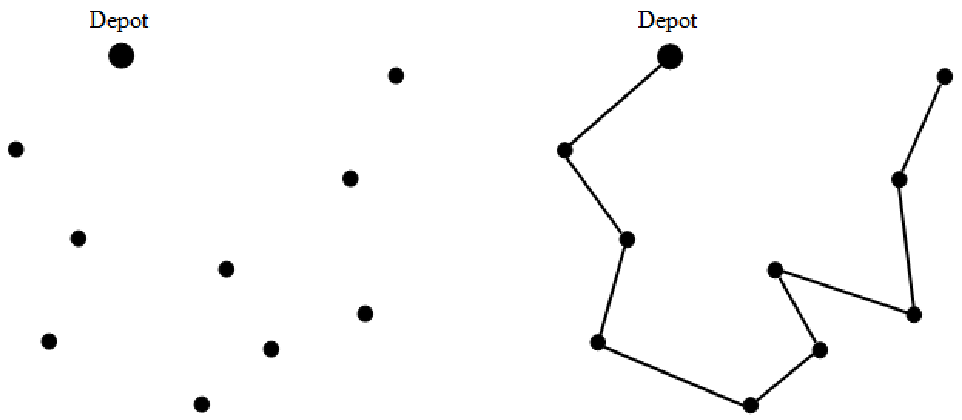

Figure 4.1 An illustration of TRP

As it is stated before, the objective is to minimize waiting times of all customers. Figure 4.1 displays graphical TRP. Each customer is situated at a separate node, so the number of the vertices equals to the number of customers. Let $G = (V, A)$ be a directed graph where $V = \{0,...,n\}$ is the vertex set where n corresponds to the number of customers and $A = \{(i, j) \mid i, j \in V, i \neq j\}$ is the arc set. Vertex 0 is the depot, d_{ij} denotes the distance between node i and node j. Waiting time of a customer depends on its position in the sequence of service. Here, waiting time or latency can be expressed as in Eq. (4.1).

$$\text{latency}\ (j) = \text{latency}\ (i) + \text{distance}\ (i,j) \tag{4.1}$$

where i precedes j. In TRP, the sum of distances from the depot to every customer node denotes total waiting time of the customers. Let w_i be the waiting time of ith customer. Thus, the total waiting time of the customers becomes $\sum_{i \in V \setminus \{0\}} w_i$. Calculation of the total latency is shown in Eq. (4.2) where d_{ij} denotes only *travelled* distance.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (n - k + 1) \cdot d_{ij} \tag{4.2}$$

Here, k is the position of the customer in the sequence. Accordingly, representation of the objective function will be *min* $\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k+1)\cdot d_{ij}$

The mathematical formulation of the problem can be seen below with reference to Fischetti et al. (1993). This formulation considers returning to depot in contrast to the open Hamiltonian cycle problem structure. Here, node 1 is accepted as the depot and *n-1* is the number of customers. c_{ij} is the cost or distance associated with arc (v_i,v_j). Variables x_{ij} take the value 0 if arc (v_i,v_j) is not used and the value *n-k+1* if it appears in position k on the Hamiltonian tour. Binary variables y_{ij} are equal to 1 if and only if arc (v_i,v_j) appears on the cycle. It is assumed $x_{ij} = y_{ij} = 0$ whenever $i = j$.

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\cdot x_{ij}$ (4.3)

Subject to

$$\sum_{j=1}^{n} y_{ij} = 1 \qquad (i=1,\dots,n) \tag{4.4}$$

$$\sum_{i=1}^{n} y_{ij} = 1 \qquad (j=1,\dots,n) \tag{4.5}$$

$$\sum_{i=2}^{n} x_{i1} = 1 \tag{4.6}$$

$$\sum_{i=1}^{n} x_{ik} - \sum_{j=1}^{n} x_{kj} = \begin{cases} 1-n & (k=1) \\ 1 & (k=2,\dots,n) \end{cases} \tag{4.7}$$

$$x_{ij} \le r_{ij}\cdot y_{ij} \qquad (i,j=1,\dots,n) \tag{4.8}$$

where

$$r_{ij} = \begin{cases} 1 & (j=1) \\ n & (i=1) \\ n-1 & otherwise \end{cases} \tag{4.9}$$

$$y_{ij} \in \{0,1\} \qquad (i,j=1,\dots,n) \tag{4.10}$$

$$x_{ij} \ge 0 \text{ and integer } (i,j=1,\dots,n) \tag{4.11}$$

In this formulation, objective function (4.3) gives the sum of distances from the depot to every customer node. Constraints (4.4), (4.5) and (4.10) are the assignment constraints that ensure each node is served only once. Constraints (4.6), (4.7) and (4.11) define a network flow problem. Constraints (4.8) and (4.9) ensure that x_{ij} can only take a positive value if vertex *j* follows vertex *i* on one of the subtours.

4.3 Literature Review

4.3.1 Solution Methods for Traveling Repairman Problem

The problem is examined with different solution methods in the literature. Three types of studies can be mentioned here. First one is the exact algorithm approaches. The second one is the approximation methods and many of the studies on this problem focus on these two solution methods. Several exact solution algorithms and approximation algorithms are examined in the literature (Archer & Williamson, 2003; Blum et al., 1994; Fischetti et al., 1993; Wu, Huang, & Zhan, 2004). The last one is the meta-heuristic approaches. There are a few studies that examine TRP with a meta-heuristic approach; therefore result of this study can only be compared with the results of these studies. Meta-heuristic approaches for TRP can also be found in (Ngueveu, Prins, & Calvo, 2010; Salehipour, Sörensen, Goos, & Bräysy, 2008, 2011; Silva, Subramanian, Vidal, & Ochi, 2012).

4.3.1.1 Exact and Approximation Methods

As exact and approximation algorithms, following studies can be referred to. In (Blum et al., 1994), TSP and TRP are compared with their characteristics, exact and approximation algorithms for TRP are proposed. Depth first search and dynamic programming are examined as the exact solutions; i- tree problem, constant factor approximation algorithm and positive-linear TDTSP approximation algorithm are examined as the approximation algorithms for the problem. The approximation factor is denoted as 144. In (Wu, Huang, & Zhan, 2004), exact algorithms are proposed to solve TRP. Developed algorithms are combinations of dynamic programming and branch and bound techniques. In (Fischetti et al., 1993), a linear integer programming formulation is proposed. TRP is considered as CTSP in (Bianco et al., 1993). Two exact algorithms that incorporate lower bounds provided by a Lagrangian relaxation of the problem are proposed. Lucena (1990) examines TRP under the more general TDTSP concept. A non-linear integer formulation, a branch and bound algorithm, is described. In (Archer et al., 2003), the authors give a 9.28 approximation algorithm for the problem. To the best of the knowledge, the smallest approximation factor for TRP is proposed in (Chaudhuri et al., 2003) with 3.59. The most recent study about TRP seems as the study of Angel-Bello, Alvarez, & García (2013). Two integer formulations for the TRP are proposed and compared with previous studies, and the asymmetric case for TRP is also experimented.

4.3.1.2 Meta-heuristic Approaches

The first meta-heuristic approach for the TRP is introduced in (Salehipour et al., 2008). In the study, GRASP is used as the construction phase and variable neighborhood descent (VND) is used as the improvement phase. Data sets are generated and algorithm is tested with these data sets. Results are obtained in a reasonable time period by comparison with the upper and lower bounds. In (Ngueveu et al., 2010), a memetic algorithm which is developed for cumulative capacitated vehicle routing problem (CCVRP) and its results are introduced. Although the algorithm is not specially designed for TRP, it is applied to TRP and it is stated that algorithm gives better results than the first meta-heuristic approach for the problem. Two different memetic algorithms are compared with the previous meta-heuristic and tested with the same data sets. Silva et al. (2012) proposed a meta-heuristic based on a greedy randomized approach for solution construction and iterated variable neighborhood descent with random neighborhood ordering for solution improvement. It is shown that algorithm gives better results than the previous meta-heuristic studies in terms of percentage deviations of the solutions from the upper bounds.

4.4 Methodology of the Proposed Approach

4.4.1 Genetic Algorithm

Genetic algorithm is a meta-heuristic approach which belongs to the class of evolutionary algorithms. Genetic algorithms can be seen as the reflectors of the evolution process and this point of view has made an encouragement for solving optimization problems (Gen & Cheng, 1997).

To explain the concept of the genetic algorithm, some special terms are mentioned here. In genetic algorithms, a solution element is named as *individual* or *chromosome*. An individual can be thought as the resulting tour in TRP. The group of solution elements constitutes a *population*. The solution quality of an individual is represented with the *fitness value* of that individual. A *parent* is the individual chosen from the population for recombination. The term *crossover* means recombination operation and a *child* emerges from recombination of the parents (Zäpfel et al., 2010).

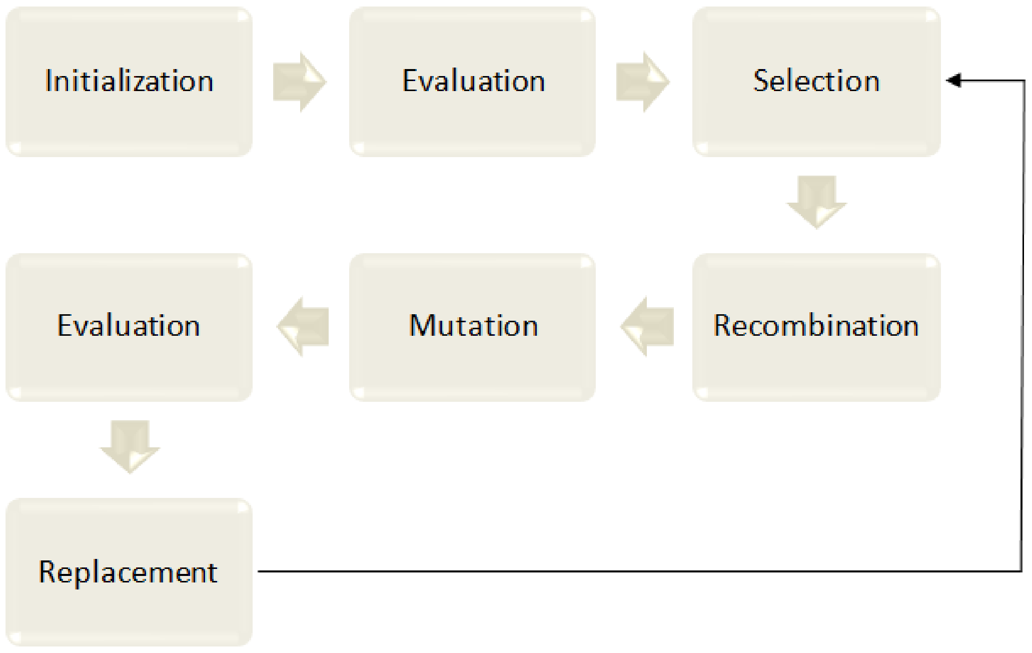

Figure 4.2 A general framework for genetic algorithms

In Figure 4.2, the genetic algorithm framework can be seen. Each of these phases can be constructed according to the features of the problem. Firstly, the algorithm starts with initialization phase in which the population is determined. Fitness values of the individuals are calculated in the evaluation phase. Following selection phase makes a selection among candidate individuals and after selection, a child is obtained from recombination of these parents. As a necessity of the evolution process, the child is mutated according to a probability and evaluated in conjunction with the parents on hand. If child gives a better solution than at least one of the parents, the child is added to the population and this process is named as *survival.* After replacement operation, the algorithm is terminated until a termination criterion is reached. This criterion can be a condition that is expected to be reached, or it can be a number of iterations. During this procedure, candidate solutions for recombination are chosen from the new population that is continuously updated. Detailed information on genetic algorithms can be found in (Goldberg, 1989; Mitchell, 1996; Reeves, 2003).

4.4.1.1 Solution Representation

In a genetic algorithm, it is often better to use problem dependent encoding type for the individual. For TRP, solution encoding is represented with a permutation vector of the vertices as in Figure 4.3.

Following phases are basic process steps of genetic algorithms. In these phases, several methods can be used and various assumptions can be made to search the solution space.

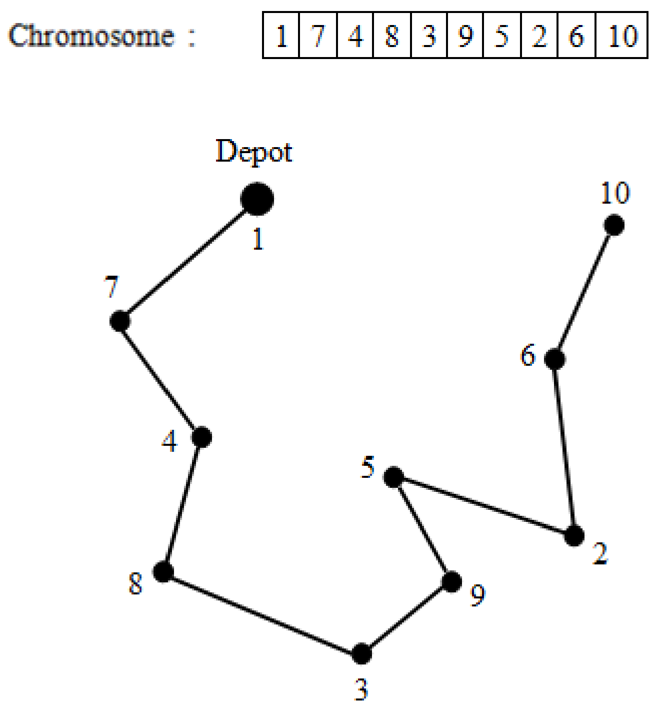

Figure 4.3 An example of permutation of a solution with 10 vertices

4.4.1.2 Initialization

Initialization phase is the starting point for the algorithm. The population that will be used in the algorithm is determined in this phase. It is remarkable that a population can be created randomly or it can be constructed with some heuristic approaches. In this study, a random generated population is used. Besides, not only the content but also size of the population is important. This size must be large enough to search the solution space and also small enough to reduce computational effort. Population size *p* used in this study varies in accordance with the size of the data sets.

4.4.1.3 Evaluation

In the evaluation phase, solution qualities of the individuals are determined. The fitness of a solution element depends on the structure of the problem. However, fitness value may be accepted as the objective function value. In TRP, fitness value of an individual can be thought as the length of its tour. In the algorithm, it is preferred to use a proportional fitness value as in Eq. (4.12). *f(i)* is the fitness of the *i*th chromosome and l_i is tour length of the *i*th chromosome.

$$f(i) = \frac{1}{\left(l_i / \sum_{i=1}^{p} l_i \right)} \tag{4.12}$$

4.4.1.4 Selection

After initial conditions are constituted, some of the individuals are chosen from the population according to a criterion. Selection procedures such as random selection, *Roulette wheel selection (RWS)*, tournament selection, linear ranking etc. can be used for selection and in this study RWS procedure is implemented to the individuals. The main idea in RWS is that the more individual fits to objective the more it is chosen. Here, fitness values of the individuals are designed so as the proportion of an individual will reflect its selection probability. Eq. (4.13) displays probability of selection where *p* is population size. Two individuals which have the highest probability of selection are selected for recombination operation. After selection, selected individuals are called as parents.

$$p(i) = \frac{f(i)}{\sum_{i=1}^{p} f(i)} \tag{4.13}$$

4.4.1.5 Recombination

To be able to make a search with the solutions on hand, some differences must be created between them. In this phase recombination of the parents is implemented and this operation is also named as crossover. Common crossover types can be said as m-point crossover, partial mapped crossover (PMX), order crossover (OX), cycle crossover (CX), and position based crossover etc. (Gen & Cheng, 1997). In the algorithm, order crossover is selected as the crossover operator. Figure 4.4 displays order crossover procedure. Firstly, a segment form one of the parents is determined randomly; this segment is copied to child's corresponding situation. From the second parent, nodes that are in this segment are removed, and empty places in child are filled with the remaining nodes in the second parent.

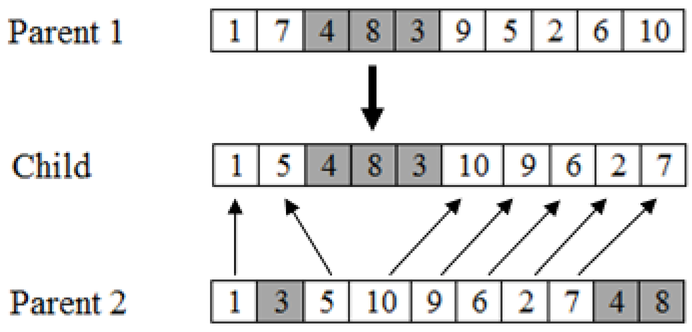

Figure 4.4 Order crossover operation

4.4.1.6 Mutation

In the mutation phase new solution element, the child, is mutated according to a mutation rate. In the study, mutation operator is determined as two-exchange (swap) operation. In this operation, two nodes are picked randomly and changed with each other. After mutation operation, to make a search around the new solution, 2-opt heuristic is applied to child, again with a 2-opt rate.

4.4.1.7 Replacement

In the replacement phase, parents and child are evaluated according to their solution quality. There are different types of replacement. One of them is adding new solution to the population without evaluation of how much it fits to objective. Another one is to select best two of them and while they are added to the population, worst one is removed from the population. This approach is known as *elitist* and in this study, the elitist replacement procedure is chosen.

4.4.2 Details of Proposed Method

In Figure 4.5 the proposed genetic algorithm structure can be seen, with refer to (Cergibozan & Tasan, 2012). Firstly, algorithm is initialized with a random generated population. In the evaluation phase, fitness values of candidate solutions are calculated considering the features of a minimization problem. After evaluation, candidates are selected for crossover operation. As the selection procedure, RWS is used in the algorithm. Order crossover is applied to selected parents as the crossover operator. After recombination of the parents, mutation operation is implemented according to mutation rate. Two-exchange operation is used as a mutation operator and new solution is kept in the population if there is an improvement. Result of the mutation operation is the start point of the exploration procedure. In the study, 2-opt local search heuristic is applied to the child if it will be added to the population. The algorithm is terminated when the capability criteria is reached.

```
Genetic Algorithm
  • Create initial population
  • Calculate fitness values of candidate solutions
    for t = 1: iteration no
        o Select parents via selection operator
        o Recombine parents with the crossover operator
        o Mutate child with the mutation operator
        o Evaluate solution of the child
          if fitness(child) better than or equal to fitness(worst parent)
              ▪ Implement 2-opt operation
              ▪ Replace child with the worst parent
          end
    end
```
Figure 4.5 Genetic algorithm structure of the proposed algorithm

4.5 Computational Study

To the best of the knowledge, there are three meta-heuristic approaches that have been studied in the TRP literature: Salehipour et al. (2008, 2011), Ngueveu et al. (2010) and Silva et al. (2012); thus the computational study is based on comparison with these studies. Data sets used in the computational study are obtained from the authors of Salehipour et al. (2008). There are six types of data sets as 10, 20, 50, 100, 200 and 500 customers and each of these sets has 20 instances. Salehipour et al. (2008) generated coordinates of these instances by using random numbers from a uniform distribution between 0 and 100 for instance sizes of 10, 20, 50, 100 and 200; and between 0 and 500 for instance size of 500.

4.5.1 Bounds for Traveling Repairman Problem

To be able to evaluate results of the study, two bounds are needed: a lower bound and an upper bound. In the previous studies, nearest neighbour algorithm (NNA) is used as an upper bound. The algorithm starts with an initial point (depot) and searches for the nearest node to this point. By adding nodes to the solution, same process is repeated for the next added node, until the final tour is obtained. Result of this algorithm is not as good as it sounds for TRP; because in TRP, closeness of the nodes to each other is less important than their distance to the starting node. Therefore, result of the NNA is examined in the computational study for the comparison with the previous studies.

In the computations, two types of lower bound are used. First lower bound which is shown in Eq. (4.14) is a multiplied form of the minimum spanning tree (MST), with reference to Salehipour et al. (2008, 2011). It is computed by sorting the edges of the MST of the graph in order of increasing weight -or distance- and multiplying each edge with the decreasing numbers starting with n.

$$LB = \sum_{i=1}^{n} (n - i + 1) \cdot w(t_i) \tag{4.14}$$

However Ngueveu et al. (2010) determined two lower bound procedures and used the maximum value of these two lower bound equations. The first lower bound in Eq. (4.15) is the case that each customer is connected directly to the depot. *R* is the number of vehicles, and its value is 1 in TRP.

$$LB_1 = \sum_{j \in V'} w_{0j} \tag{4.15}$$

Ngueveu et al. (2010) introduced a second lower bound procedure given in Eq. (4.16); but then improved it with a few modifications. Eq. (4.16) is equal to lower bound proposed in (Salehipour et al., 2008, 2011). In this equation, w_e is the weight of the *e*th shortest edge of graph *G*. Eq. (4.17) is the improved version of the lower bound given in Eq. (4.16). Here, variable w_e' is the cost of the *e*th shortest edge incident to the depot whereas w_e'' is the cost of the *e*th shortest edge between two customers.

$$LB_2 = \sum_{e=1}^{R} \left(\left\lceil \frac{R + n - e - (n \bmod R)}{R} \right\rceil \right) \cdot w_e \tag{4.16}$$

$$\begin{aligned} LB'_2 = {} & \sum_{e=1}^{R} \left(\left\lceil \frac{R + n - e - (n \bmod R)}{R} \right\rceil \right) \cdot w'_e \\ & + \sum_{e=1}^{n-R} \left(\left\lceil \frac{n - e - (n \bmod R)}{R} \right\rceil \right) \cdot w''_e \end{aligned} \tag{4.17}$$

After comparing two lower bound procedures in Eq. (4.15) and Eq. (4.17), Ngueveu et al. (2010) used the following Eq. (4.18) in computational study.

$$LB = \max\left(LB_1, LB'_2\right) \tag{4.18}$$

In the application, algorithm is coded again in MATLAB 7.7.0 software and tested on the computer with Pentium Dual-Core E2160, 1.80 GHz processor and 4 GB RAM. Best parameter combinations used in the computational study are determined by computational experiments. Because as the problem size grows, elapsed time for the computation is increasing obviously and a parameter study was implemented to reduce the computational time.

In the application of the algorithm, parameter sets are built according to the data sets used. It should be noted that parameters arranged similarly for each pair of data sets which are the data sets with sizes of 10 and 20 customers; data sets with sizes of 50 and 100 customers; and data sets with sizes of 200 and 500 customers. This classification is made according to sizes of the corresponding pairs. It is seen in the experiments that in small instances (instances with 10 and 20 customers), algorithm doesn't require much effort to find a good solution. In medium and large instances (instances with 50 and 100 customers, and instances with 200 and 500 customers, respectively), it is more difficult to find a good solution without getting caught in a local optimum. Therefore, the difference between parameter sets is based on intensifying the search process.

Table 4.1 Parameters for instance sizes of 10 and 20

Parameter	**TRP-10**	**TRP-20**
Population size *(p)*	300	300
Crossover rate	(*p**0.8)	(*p**0.8)
Number of GA iterations	250	250
Mutation rate	0.2	0.2
Number of two exchange iterations	10	10
2-opt rate	0.5	0.5
Number of 2-opt iterations	100	200

For the instance sizes of 10 and 20 customers, parameters are displayed in Table 4.1. Difference between two instance sets is only number of the 2-opt iterations.

Table 4.2 is for the instance sizes of 50 and 100. For these instances, population size is enhanced from 300 to 350 and number of GA iterations is increased from 250 to 500 to avoid convergence of the solutions.

Table 4.2 Parameters for instance sizes of 50 and 100

Parameter	**TRP-50**	**TRP-100**
Population size *(p)*	350	350
Crossover rate	(*p**0.8)	(*p**0.8)
Number of GA iterations	500	500
Mutation rate	0.2	0.2
Number of two exchange iterations	10	10
2-opt rate	0.5	0.5
Number of 2-opt iterations	100	200

In Table 4.3 parameter set for instance sizes of 200 and 500 can be seen. Population size and number of GA iterations are determined as 400 and 500, respectively.

Table 4.3 Parameters for instance sizes of 200 and 500

Parameter	**TRP-200**	**TRP-500**
Population size *(p)*	400	400
Crossover rate	(*p**0.8)	(*p**0.8)
Number of GA iterations	500	500
Mutation rate	0.2	0.2
Number of two exchange iterations	10	10
2-opt rate	0.5	0.5
Number of 2-opt iterations	100	200

Main objective in the selection of these parameter sets is to decrease computational effort while size of the instance set is increasing. It is found from the experiments that for the instance sizes of 10 and 20, a population size of 300 and GA iteration number of 250 is much enough to reach best solution. However for instance sizes of 50 and 100, population size is increased to 350 and number of GA iterations is determined as 500 to be able to search more space. For the instance sizes of 200 and 500, population size of 350 doesn’t yield a good result contrary to a population size of 400. For each pair of data sets, 2-opt iteration number is determined such that while enhancing 2-opt iteration number from 100 to 200, this number will not affect the computation time of the algorithm significantly.

4.6 Results

In Table 4.4 result of the computational study for instance sizes of 10 and 20 can be seen. Here, Instance displays the instance set with its size and number, Result shows best found solution with the proposed hybrid genetic algorithm in terms of tour length, Opt Sol means optimal solution for that instance set with refer to Salehipour et al. (2011).

As it is seen from Table 4.4, proposed algorithm can find optimum solutions in a reasonable time period; computation time of the algorithm is recorded as 26.8 seconds on average for the instance size of 10, and 53.6 seconds on average for the instance size of 20. It should be noted that the proposed hybrid genetic algorithm gives better results for instance sets TRP-20-17 and TRP-20-18 in comparison with Salehipour et al. (2011). While this research is on progress, Silva et al. (2012) found same results; therefore optimal solutions for corresponding instance sets are reached and proved to be optimal with refer to Silva et al. (2012).

Table 4.4 Results in terms of best tour lengths for instance sizes of 10 and 20

Instance	Result	Opt Sol*	Instance	Result	Opt Sol*
TRP-10-1	1303	1303	*TRP-20-1*	3175	3175
TRP-10-2	1517	1517	*TRP-20-2*	3248	3248
TRP-10-3	1233	1233	*TRP-20-3*	3570	3570
TRP-10-4	1386	1386	*TRP-20-4*	2983	2983
TRP-10-5	978	978	*TRP-20-5*	3248	3248
TRP-10-6	1477	1477	*TRP-20-6*	3328	3328
TRP-10-7	1163	1163	*TRP-20-7*	2809	2809
TRP-10-8	1234	1234	*TRP-20-8*	3461	3461
TRP-10-9	1402	1402	*TRP-20-9*	3475	3475
TRP-10-10	1388	1388	*TRP-20-10*	3359	3359
TRP-10-11	1405	1405	*TRP-20-11*	2916	2916
TRP-10-12	1150	1150	*TRP-20-12*	3314	3314
TRP-10-13	1531	1531	*TRP-20-13*	3412	3412
TRP-10-14	1219	1219	*TRP-20-14*	3297	3297
TRP-10-15	1087	1087	*TRP-20-15*	2862	2862
TRP-10-16	1264	1264	*TRP-20-16*	3433	3433
TRP-10-17	1058	1058	*TRP-20-17*	**2913**	2924
TRP-10-18	1083	1083	*TRP-20-18*	**3124**	3150
TRP-10-19	1394	1394	*TRP-20-19*	3299	3299
TRP-10-20	951	951	*TRP-20-20*	2796	2796

* (Salehipour et al., 2011)

Table 4.5 displays result of the computational study for instance size of 50. In Table 4.5, Best Result column indicates best found solutions for each instance with the proposed method. Average Result column displays average found solutions after 10 runs of the algorithm. Worst Result column displays worst found solutions. Computation time of the algorithm is recorded as 131.5 seconds on average for the instance size of 50.

Table 4.5 Results in terms of best tour lengths for instance size of 50

Instance	Best Result	Average Result	Worst Result
TRP-50-1	12241	13214.5	13839
TRP-50-2	11776	12688.3	13327
TRP-50-3	12424	12578	12641
TRP-50-4	13367	13646.4	13768
TRP-50-5	12713	12713	12713
TRP-50-6	12923	13278.8	13325
TRP-50-7	11645	11758.3	11966
TRP-50-8	13196	13196	13196
TRP-50-9	13413	13574	13954
TRP-50-10	13164	13464.3	13731
TRP-50-11	12442	12691.6	13574
TRP-50-12	10977	10977	10977
TRP-50-13	12621	12789.7	12883
TRP-50-14	13248	13324	13568
TRP-50-15	12348	12411.6	12455
TRP-50-16	12852	13082	13317
TRP-50-17	12981	13098.6	13126
TRP-50-18	13833	13929	13953
TRP-50-19	11430	11430	11430
TRP-50-20	12612	12612	12612

Table 4.6 displays result of the computational study for instance size of 100. Computation time of the algorithm is recorded as 150.2 seconds on average for the instance size of 100.

Table 4.6 Results in terms of best tour lengths for instance size of 100

Instance	Best Result	Average Result	Worst Result
TRP-100-1	34038	34185.6	34202
TRP-100-2	34029	34550.9	35249
TRP-100-3	33310	34219.7	34703
TRP-100-4	35829	36590.4	36959
TRP-100-5	35050	35054.6	35096
TRP-100-6	35539	35776.8	35805
TRP-100-7	35492	36332.5	37793
TRP-100-8	33052	33503.3	34820
TRP-100-9	35686	36520.3	38247
TRP-100-10	32451	33070.1	34673
TRP-100-11	36997	37068.7	37137
TRP-100-12	33589	33611.9	33652
TRP-100-13	34072	35525.7	36759
TRP-100-14	32544	32570.1	32615
TRP-100-15	35239	35412.9	35578
TRP-100-16	35571	37369.3	37764
TRP-100-17	35646	37237.6	39585
TRP-100-18	34982	35230.2	35459
TRP-100-19	37052	37062.6	37105
TRP-100-20	34425	35122.9	35872

Table 4.7 is result of the computational study for instance size of 200. Computation time of the algorithm is 198.4 seconds on average for the instance size of 200. After 10 runs; best, average and worst results are found for each instance and displayed in Table 4.7.

Table 4.7 Results in terms of best tour lengths for instance size of 200

Instance	Best Result	Average Result	Worst Result
TRP-200-1	96744	98593.4	99720
TRP-200-2	96761	98795.5	102800
TRP-200-3	103165	103763.6	104115
TRP-200-4	100134	101328	101782
TRP-200-5	93629	94332.9	95142
TRP-200-6	97092	99486.3	103158
TRP-200-7	96827	98040.8	98955
TRP-200-8	92734	95377.7	96427
TRP-200-9	96471	97085	97847
TRP-200-10	96813	97413.2	98848
TRP-200-11	97108	97855.7	98793
TRP-200-12	99376	99676	100280
TRP-200-13	92021	92433.3	92773
TRP-200-14	100941	101826	102546
TRP-200-15	97720	98392.3	98994
TRP-200-16	95769	96275.4	96588
TRP-200-17	91934	93140.7	93841
TRP-200-18	97370	98827.4	99927
TRP-200-19	101879	102306	102664
TRP-200-20	92702	93260.4	93460

In Table 4.8 result of the computational study for instance size of 500 can be seen. Computation time of the algorithm is 834.8 seconds on average for the instance size of 500. Best, average and worst found results for each instance are shown in Table 4.8.

Table 4.8 Results in terms of best tour lengths for instance size of 500

Instance	Best Result	Average Result	Worst Result
TRP-500-1	2061157	2082818	2121861
TRP-500-2	1997089	2024800	2042886
TRP-500-3	2032307	2051226	2069874
TRP-500-4	1999398	2008325	2015378
TRP-500-5	2059741	2070344	2082894
TRP-500-6	1978166	1981824	1988404
TRP-500-7	2036629	2047500	2063391
TRP-500-8	2015932	2027147	2040959
TRP-500-9	1880586	1893596	1905268
TRP-500-10	1894922	1911114	1936053
TRP-500-11	1987380	2002446	2021224
TRP-500-12	1947744	1954614	1965962
TRP-500-13	2046734	2057870	2072609
TRP-500-14	1965841	1976797	1996189
TRP-500-15	1926182	1933058	1937938
TRP-500-16	1977857	1987069	1992261
TRP-500-17	1981645	1993192	2001370
TRP-500-18	2019531	2047740	2060975
TRP-500-19	1951256	1956057	1969933
TRP-500-20	2016970	2023614	2032731

The comparison of the proposed hybrid GA with the previous studies is given in Table 4.9. *NI* indicates number of instances in each size of data set. *N* is the size of the data set. *%UB* and %LB columns display percentage deviations of the best solution from the upper and lower bounds and differ in accordance with the previous algorithms are deep or fast. *HGA* is the hybrid GA for the problem. *HGA %LB* column displays percentage deviation of the best solution from the first study's lower bound. In the same way, *HGA %LB'* displays percentage deviation of the best solution from the second study's lower bound. *HGA %UB* indicates percentage deviation of the best solution from the upper bound. In the last column, *T* indicates processing time of the hybrid GA in seconds. Percentage deviation calculations from lower bounds are different in the previous studies; percentage deviations remarked with (*) are found by use of Eq. (4.19), other percentage deviations are found by use of Eq. (3.10).

From Table 4.9, almost all of the percentage deviations of the solutions from the lower bounds are lower in comparison to Ngueveu et al. (2010) and for the half of the instances; algorithm gives better results from Salehipour et al. (2008). In addition, hybrid GA has the advantage in terms of computational time in the instances with large sizes. The result of the study yields lower values in the percentage deviations of the solutions from the upper bounds, the reason of that can be explained with a little difference between studies may has been occurred in the upper bound computations.

$$\%Deviation = \frac{(b-a)*100}{b} \qquad \text{where b>a} \qquad (4.19)$$

Figure 4.6 displays the percentage deviations from lower bounds for the instance size of 10. The following figures Figure 4.7, Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11 are percentage deviations from lower bounds for the instance sizes of 20, 50, 100, 200 and 500, respectively.

In the same way, in Figure 4.12, percentage deviations from the upper bound for instance size of 10 is shown. From Figure 4.13 to Figure 4.17 percentage deviations for instance sizes of 20, 50, 100, 200 and 500 can be seen.

From Figure 4.18 to Figure 4.23 pattern of the hybrid GA results are given in terms of tour lengths according to the instance number. The graph indicates both upper bounds and maximum of the lower bounds. It can be seen that hybrid GA results are between the lower bound and the upper bound.

Table 4.9 Results of the computational study according to instances and previous studies

NI	*N*	Salehipour et al. (2008)				*HGA %LB*	Ngueveu et al. (2010)				*HGA %LB'*	*HGA %UB*	*T*
		%UB *Deep*	%UB *Fast*	%LB *Deep*	%LB *Fast*		%UB *Deep*	%UB *Fast*	%LB *Deep*	%LB *Fast*			
20	10	2.44	2.14	33.04	33.48	33.04	2.43	2.43	26.68	26.68	40.55 (27.75)*	2.44	26.8
20	20	9.86	8.19	39.63	42.43	39.21	10.11	10.11	32.49	32.49	44.76 (30.49)*	10.14	53.6
20	50	9.67	6.85	45.41	50.07	48.19	9.36	9.32	37.58	37	49.06 (32.74)*	7.96	131.5
20	100	11.56	10.24	43.40	45.53	47.76	11.95	11.92	36.48	36.5	48.82 (32.71)*	8.89	150.2
20	200	11.33	10.69	42.32	43.37	47.71	12.81	12.7	34.57	34.69	48.22 (32.44)*	8.16	198.4
20	500	8.11	10.7	50.73	46.49	52.21	13.85	13.44	36.31	36.63	52.43 (34.38)*	7.35	834.8

* This value is found according to percentage deviation calculation of Ngueveu et al. (2010).

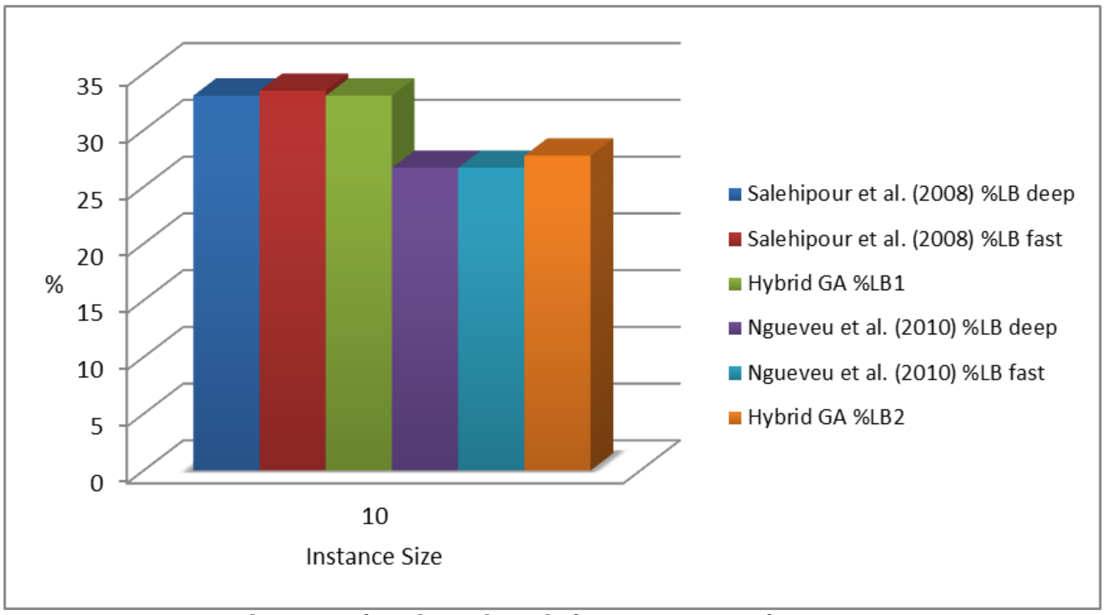

Figure 4.6 Percentage deviations from lower bounds for instance size of 10

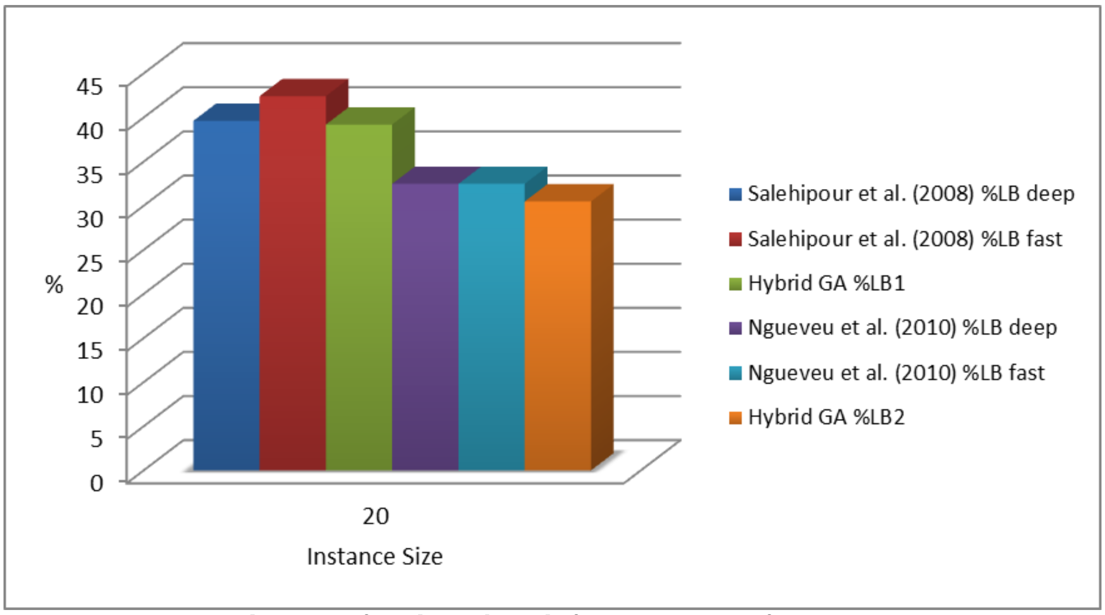

Figure 4.7 Percentage deviations from lower bounds for instance size of 20

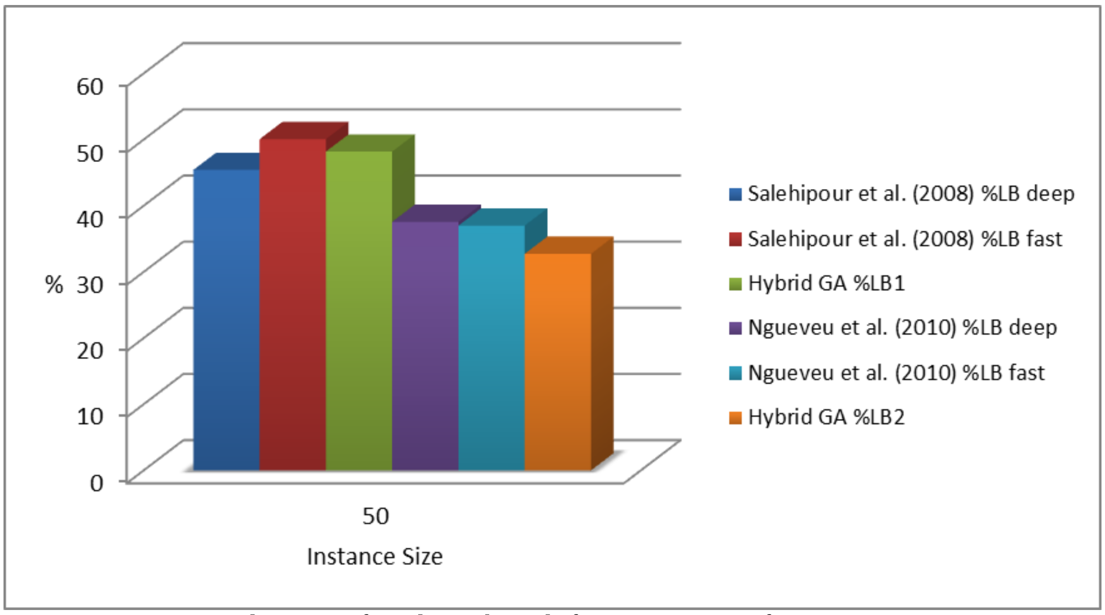

Figure 4.8 Percentage deviations from lower bounds for instance size of 50

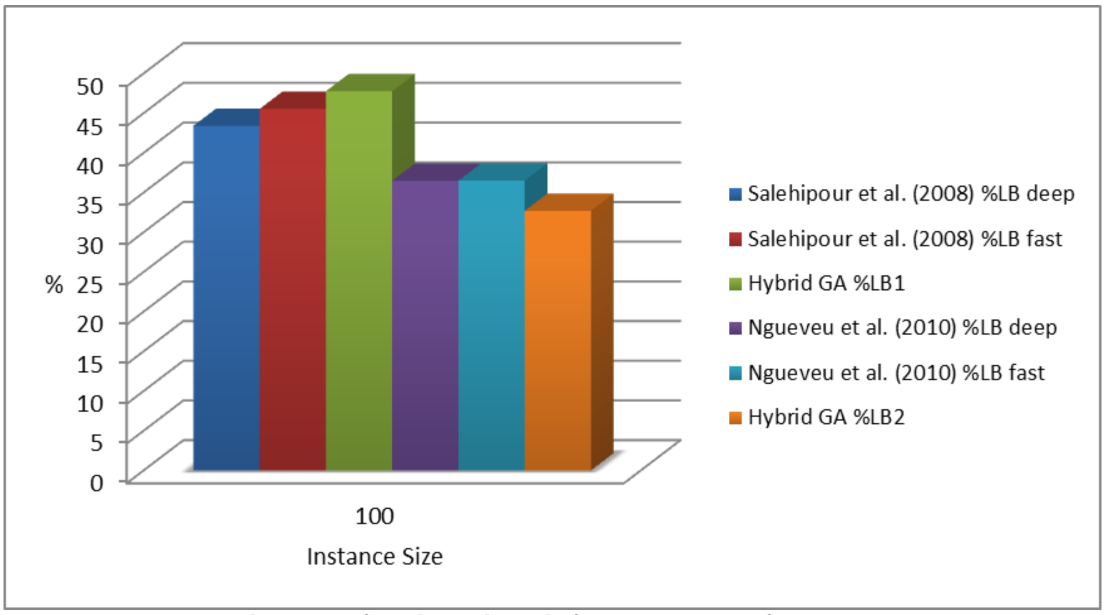

Figure 4.9 Percentage deviations from lower bounds for instance size of 100

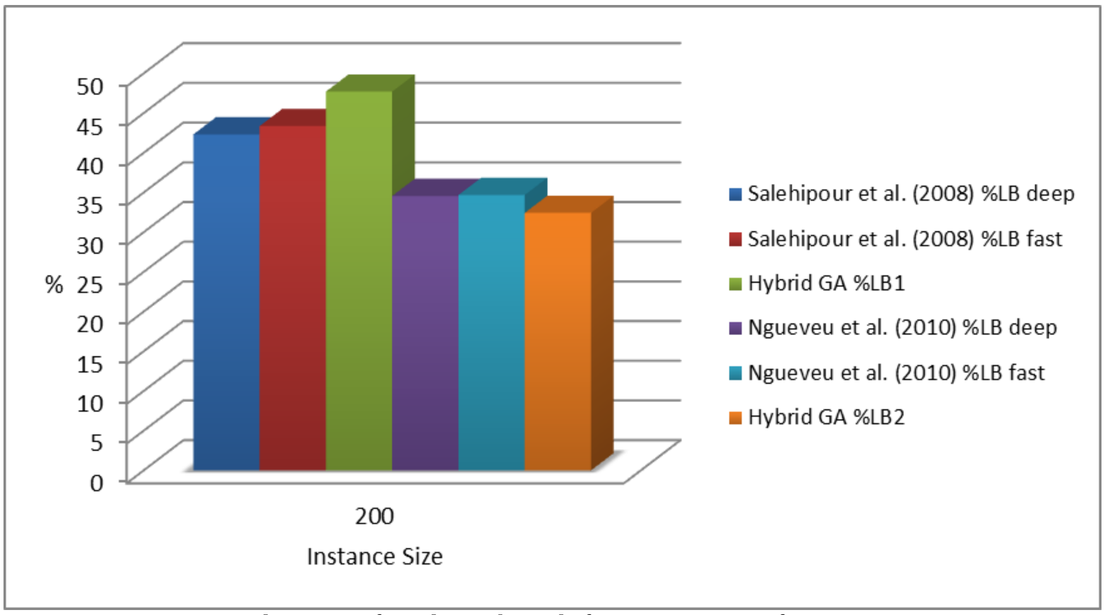

Figure 4.10 Percentage deviations from lower bounds for instance size of 200

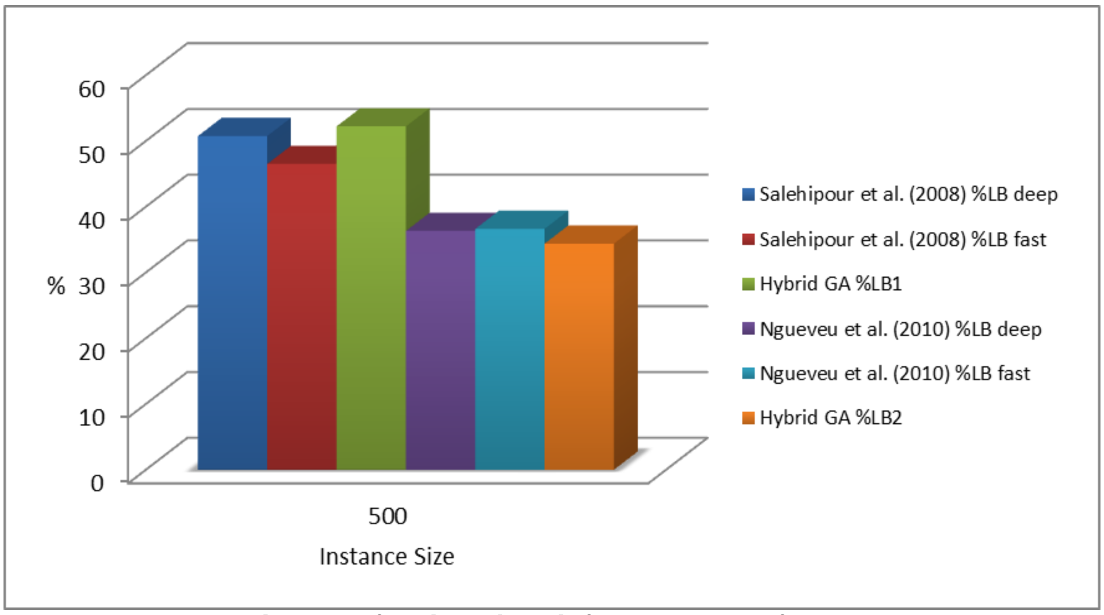

Figure 4.11 Percentage deviations from lower bounds for instance size of 500

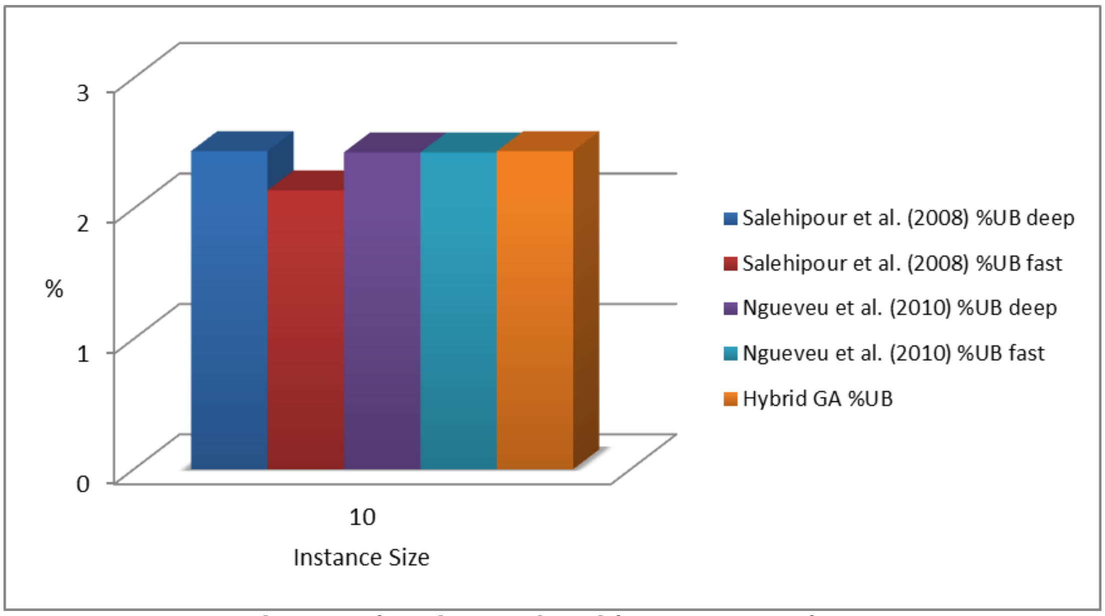

Figure 4.12 Percentage deviations from the upper bound for instance size of 10

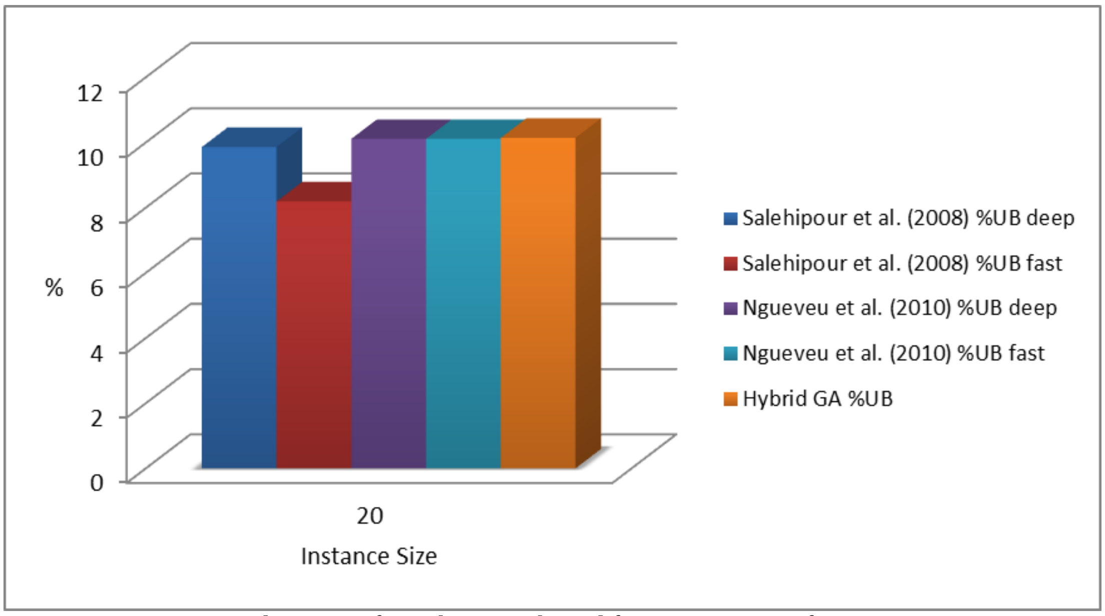

Figure 4.13 Percentage deviations from the upper bound for instance size of 20

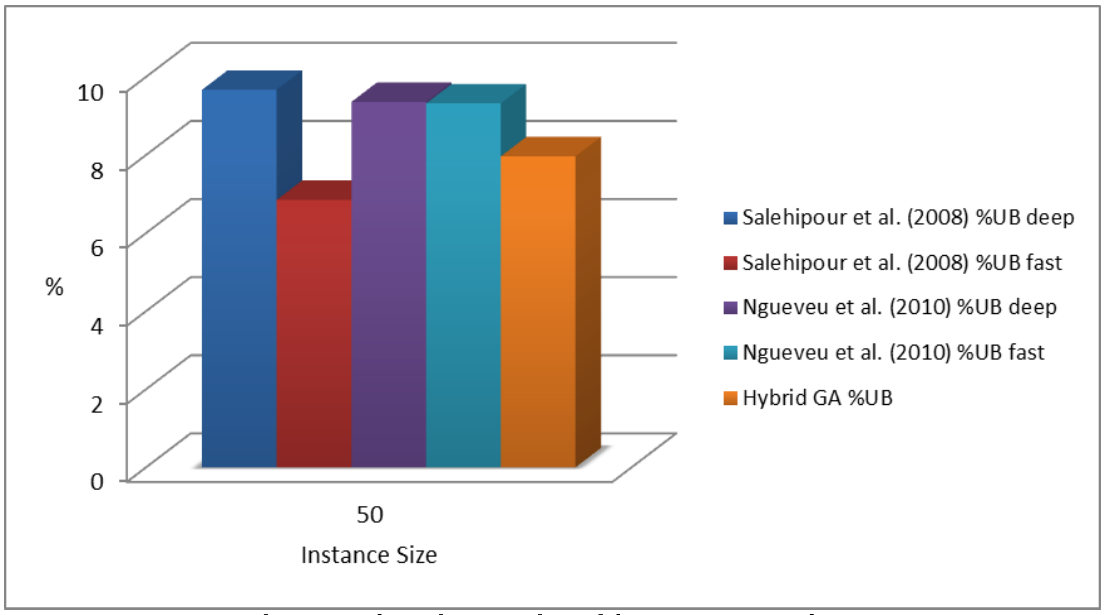

Figure 4.14 Percentage deviations from the upper bound for instance size of 50

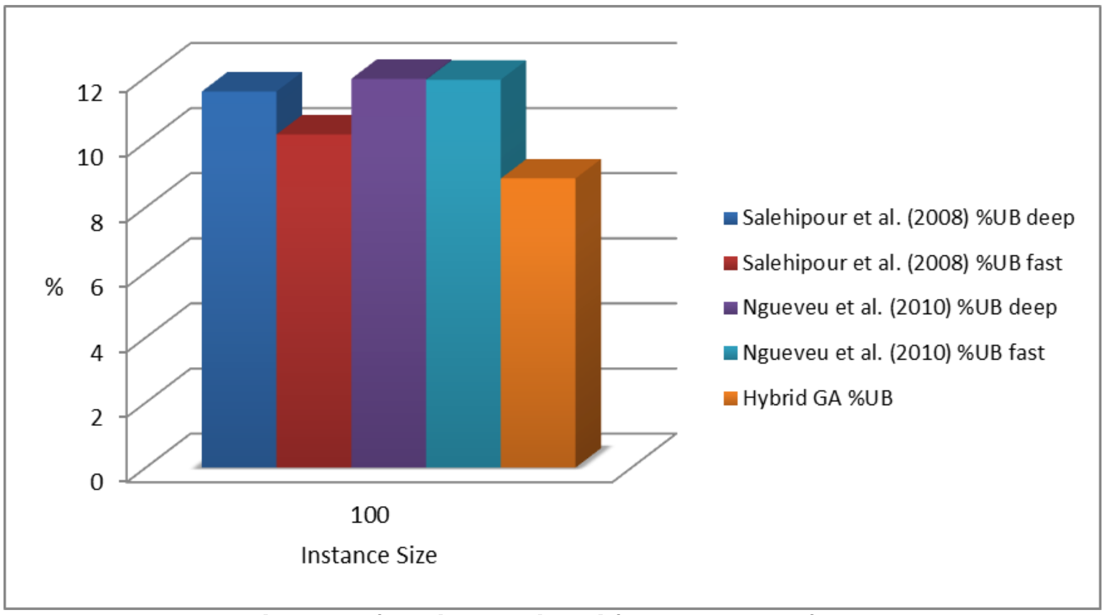

Figure 4.15 Percentage deviations from the upper bound for instance size of 100

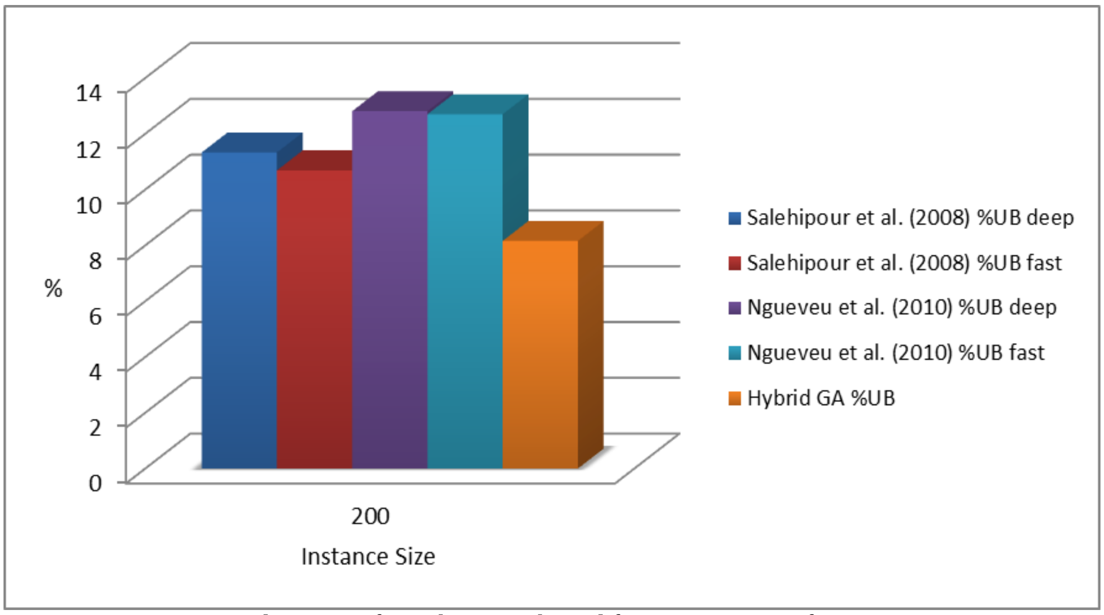

Figure 4.16 Percentage deviations from the upper bound for instance size of 200

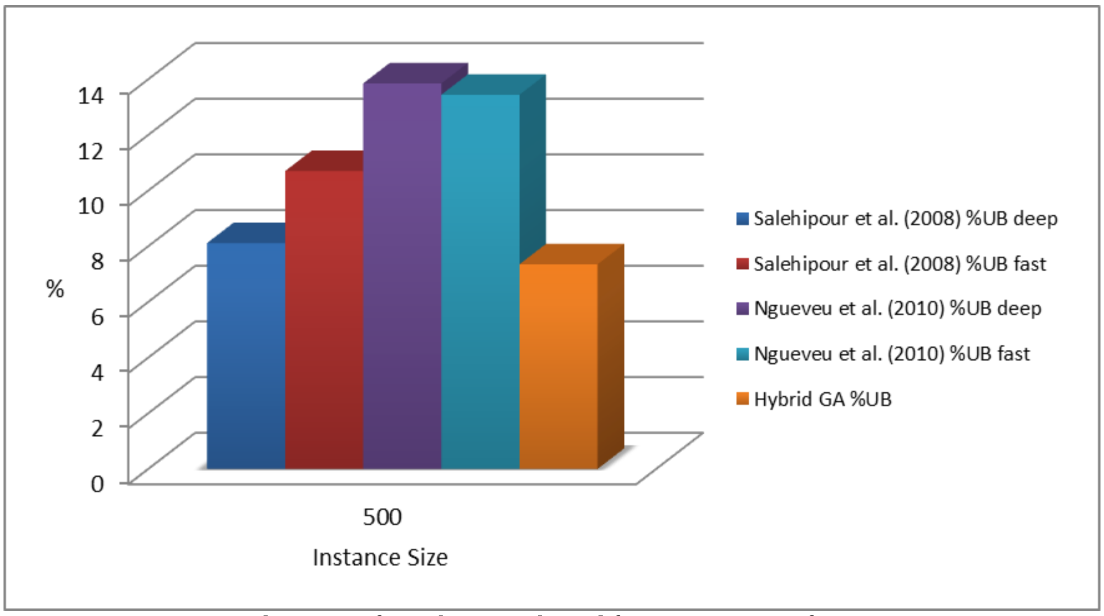

Figure 4.17 Percentage deviations from the upper bound for instance size of 500

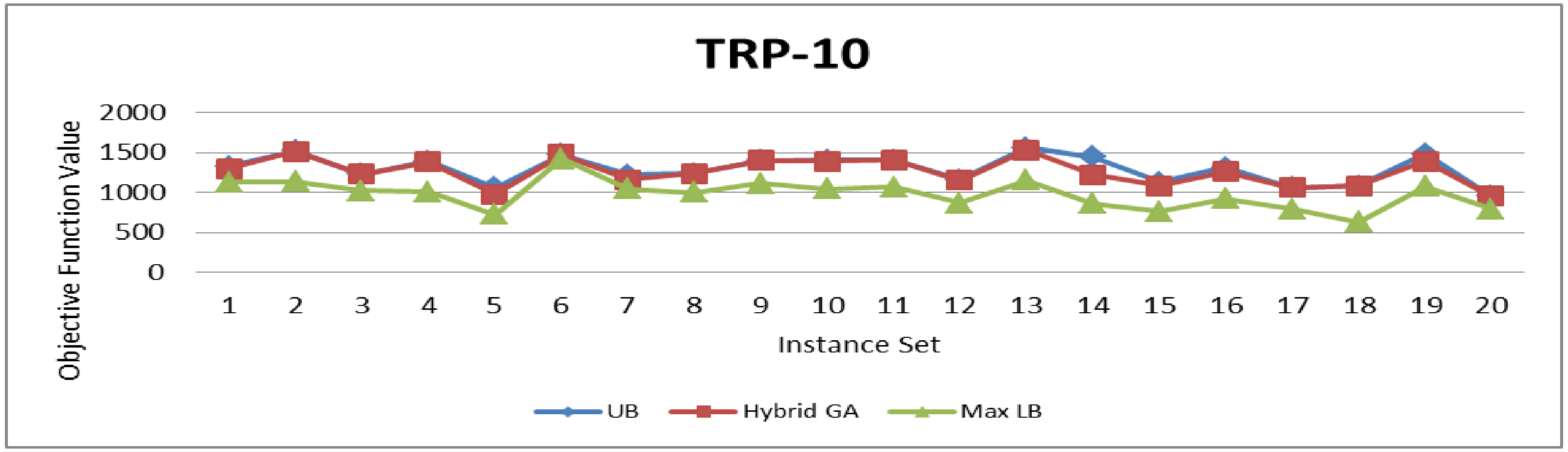

Figure 4.18 Pattern of the hybrid GA results for TRP-10 instances

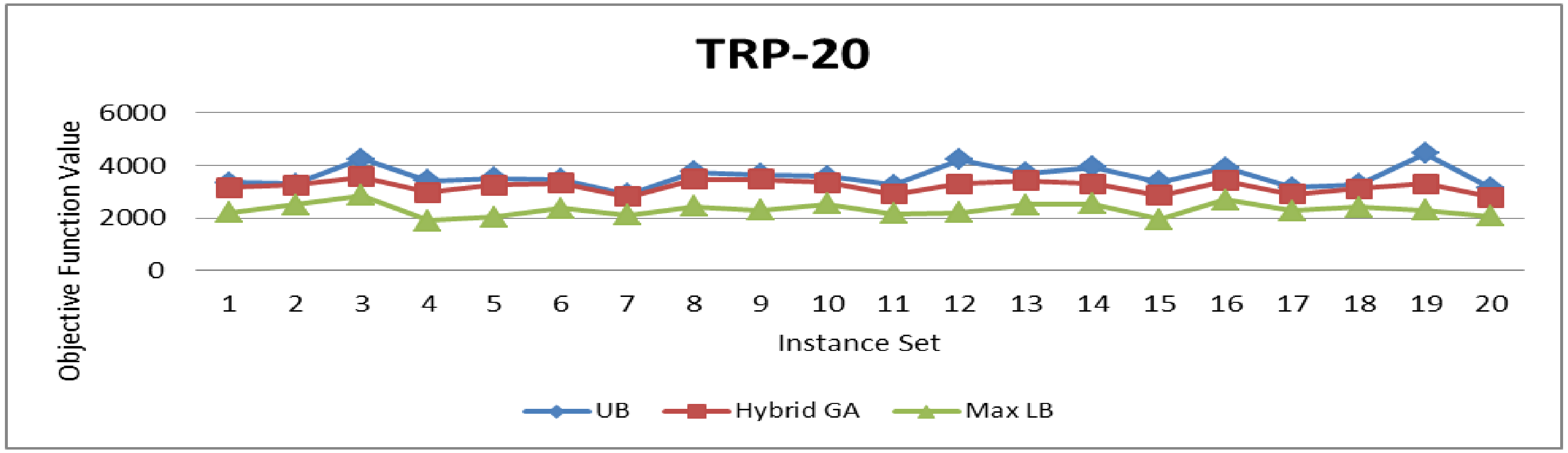

Figure 4.19 Pattern of the hybrid GA results for TRP-20 instances

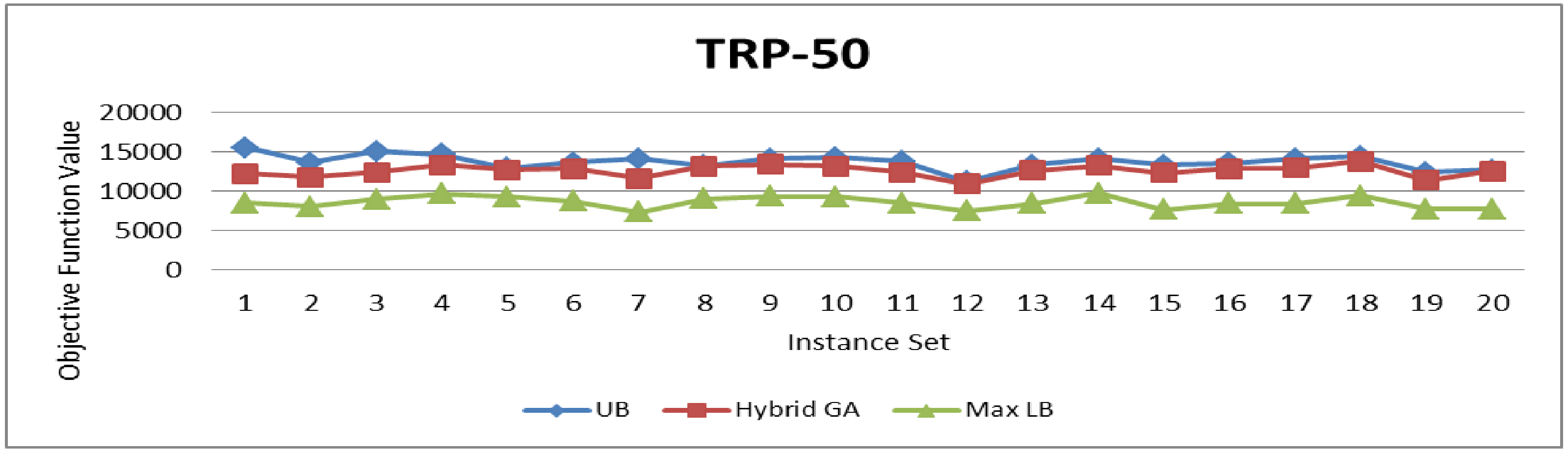

Figure 4.20 Pattern of the hybrid GA results for TRP-50 instances

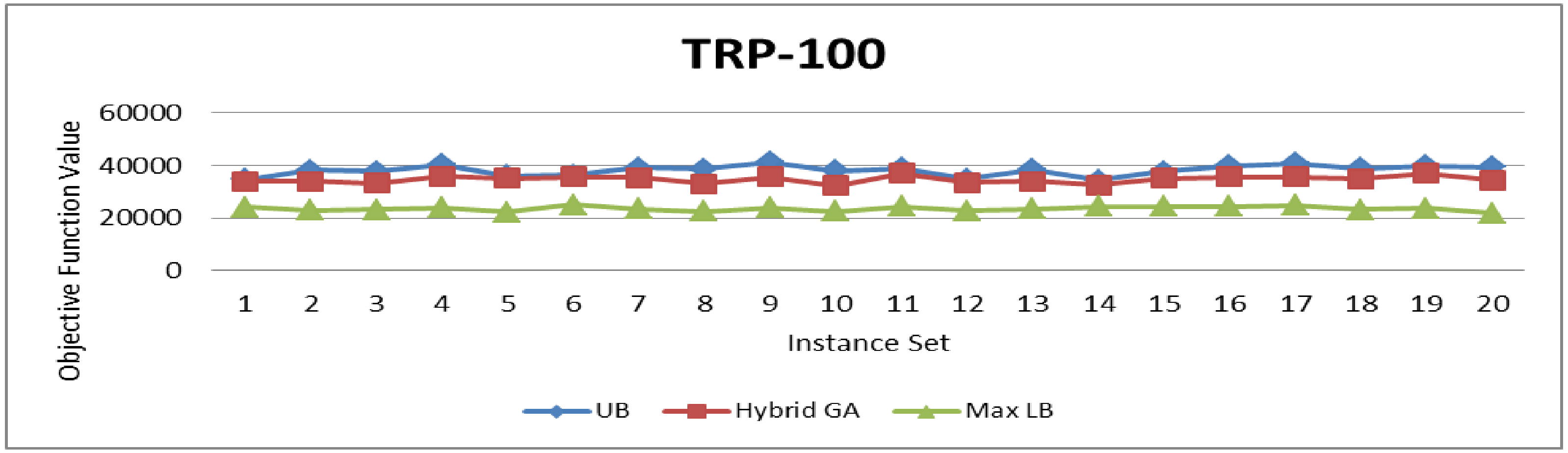

Figure 4.21 Pattern of the hybrid GA results for TRP-100 instances

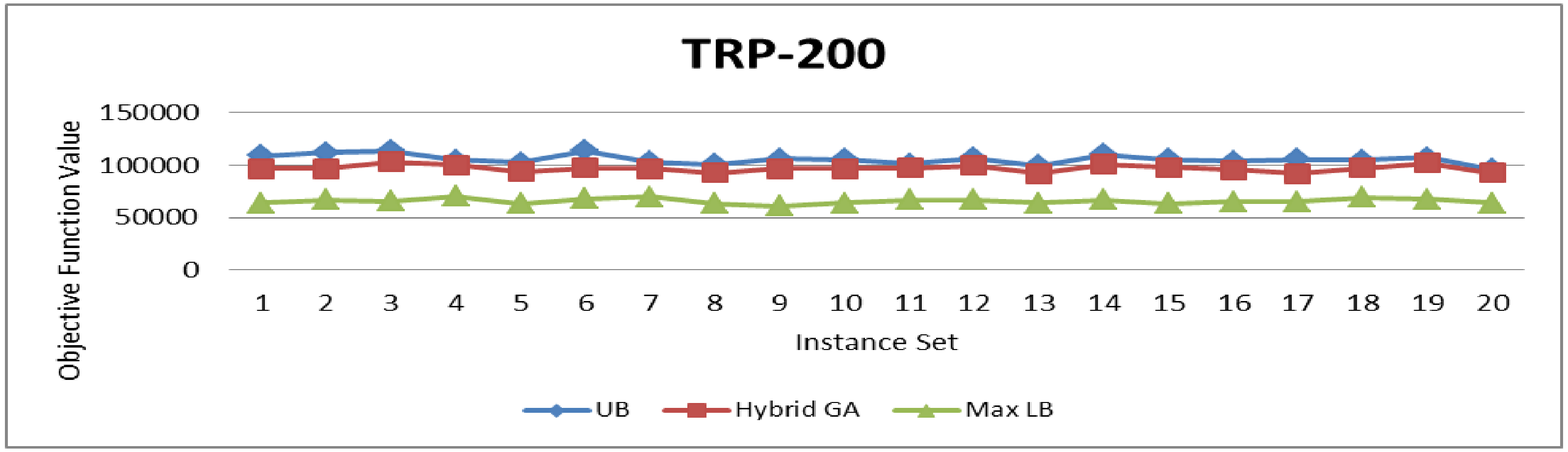

Figure 4.22 Pattern of the hybrid GA results for TRP-200 instances

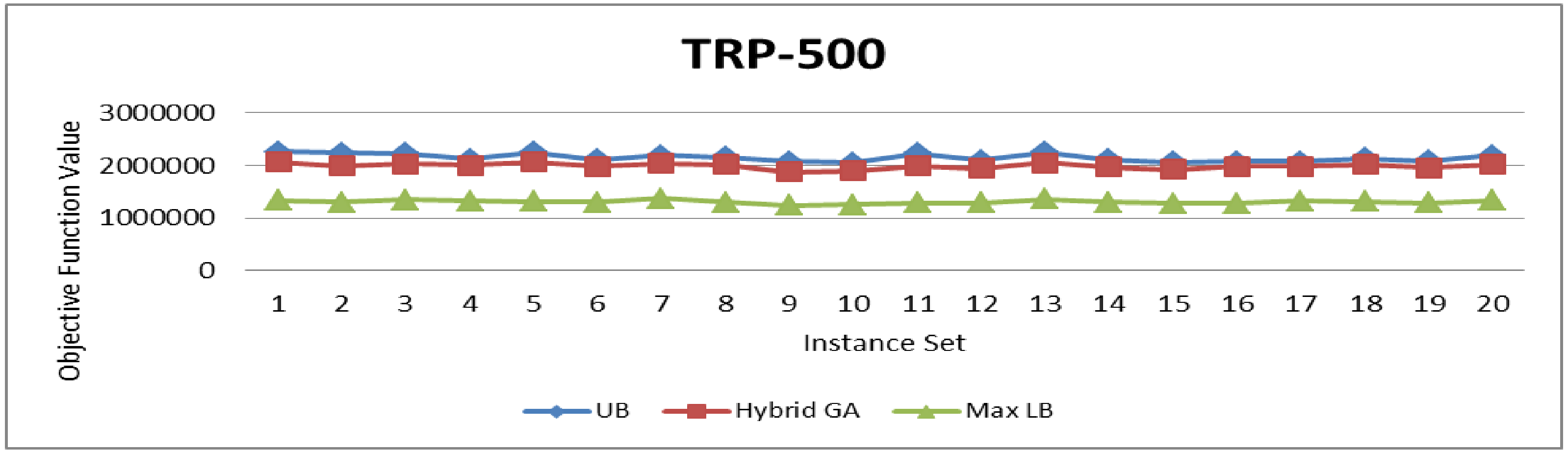

Figure 4.23 Pattern of the hybrid GA results for TRP-500 instances

CHAPTER FIVE
A CASE STUDY FOR TRAVELING REPAIRMAN PROBLEM: ROUTING OF REPAIRMEN UNDER HEAVY SNOW CONDITIONS

5.1 Problem Definition

Routing problems has been widely studied in the context of combinatorial optimization. Serving to demand points as quickly as possible with consideration of minimizing total transportation cost can be seen as main objective in most of the routing activities from the viewpoint of product/service supplier; but from the viewpoint of the customers, it is also expected from the suppliers to minimize waiting time of the customer. As a result, routing problems may have different objectives according to how the problem is considered.

In such a case that the priority of importance is given to customers, cost minimization stays behind this objective. In some situations, this is a necessity and none of the other objectives is considered except minimization of the total waiting time. The situation being talked of arises after a natural disaster where the health of a human is the main subject. A real life application of such a problem can be seen as routing of repairmen under heavy snow conditions.

In severe winter weather conditions, especially in the residential areas located in high altitudes, the problem of transporting essential requirements to the residents appears frequently. When the roads that link settlements to each other are blocked because of snow, it becomes almost impossible to communicate with the residents at the region. Sometimes the roads remain closed for a few days; but sometimes it lasts for weeks in accordance with the knowledge acquired from an authority.

The main objective in such a situation is to provide essential services such as health services as quickly as possible. Another major necessities emerges in this situation can be said as; fixing the power cut, the need for clothing, food and animal feeds, etc.

From one point of view, routing of repairmen under heavy snow conditions can be examined as a case study for the TRP; since the significance of such a situation is too high in respect to be able to meet human needs.

5.2 Literature Review

During the investigation process about routing of repairmen under heavy snow conditions, it is faced that the main problem which arises in heavy snow conditions mainly examined as an arc routing problem in spite of node routing problems such as TSP and TRP. Some of the studies related with heavy snow conditions are mentioned below.

Lemieux & Campagna (1984) examined the snow ploughing problem as a Chinese postman problem and proposed a heuristic algorithm. In the study, authors accepted that there are two types of streets: main and secondary streets. Therefore, priorities of these streets are considered and an itinerary for the snow ploughing truck is determined.

In Ghiani & Improta (2000), an exact algorithm for the hierarchical Chinese postman problem in which precedence relation is linear and each cluster is connected, is proposed. In the study, it is proved that algorithm solves mentioned problem in polynomial time.

Korteweg & Volgenant (2006) studied Hierarchical Chinese Postman Problem, a Chinese Postman Problem in which the arcs has precedence relation. It is stated in that snow ploughing, where streets are divided according to importance and some streets must be serviced quickly than others, can be seen as real life application of the problem. It is also emphasized that the snow ploughing problem belongs to HCPP with traversal costs, because service is required to make the street available for traversal.

Perrier, Langevin & Campbell (2007) proposed a survey of optimization models and solution methodologies for the routing of vehicles for ploughing and snow disposal operations. Characteristics of vehicle routing problems for snow ploughing operations are classified and it is investigated that in which situations these characteristics should be considered.

5.3 A Case Study

The routing of a repairman under heavy snow conditions is examined as a case study for the TRP. In the case, Aşkale district of Erzurum province which is located in the East Anatolian Region of Turkey is examined. The average height above sea level of Aşkale district is about 1650 meters. This value is an expected high as the other residential areas in that region and Aşkale has 66 villages which have high altitudes geographically (Aşkale Kaymakamlığı, n.d.).

Latitudes and longitudes of the settlements are derived from Yerel Yönetimler Portalı (n.d.), Türkiye Haritası (n.d.) and Google Earth software. Table 5.1 is a part of the distance matrix of the corresponding nodes with the Aşkale district which is thought as a depot.

Due to the terrestrial climate, winter conditions are very troublesome especially for residents in the East Anatolian Region. When it snows, it causes closing of the roads between settlements and this situation frequently emerges in that region. In the case of routing a repairman, minimizing total waiting times of all residents is important. In Figure 5.1, the map representation taken from Google Maps (n.d.) of the settlements from Aşkale district can be seen.

In the application, it should be noted that, some characteristics related with the routing of a repairman under heavy snow conditions are omitted and some important assumptions are made. These assumptions are listed below.

- *Depot.* The repairman starts from the Aşkale origin to its tour.

- *Customers.* In the study, the settlements of Aşkale district are accepted as customers; so the repairman will operate to arrive at these settlements as quickly as possible.

- *Number of repairmen*. In the application, it is decided to use only one repairman to study the problem as TRP.

- *Characteristic of the graph.* In such a real situation, some of the roads that link settlements with each other may not exist. Therefore in this study, the graph is considered as complete, means that for each node it is possible to reach all other nodes. Accordingly, the distances between locations are computed with the latitude and longitude values of the locations to indicate direct connections.

- *Velocity of the vehicle on the cleaned road*. In a real case, the vehicle of the repairman moves forward fast in the cleaned road in comparison with the unclean road. This feature affects service time of the vehicle to customers; nevertheless it is not considered in the application and it is accepted that the vehicle drives with the same velocity on both of these roads.

5.4 Computational Study

In the computational study, three different experiments are made. With the help of these experiments, it is tried to see the difference of the TSP and TRP. In the experiments, latitudes and longitudes of the settlements are used for distance computation.

In the computations, Matlab Mapping Toolbox (*MathWorks - MATLAB and Simulink for Technical Computing*, n.d.) is utilized for spherical distance computation according to latitudes and longitudes by assuming the Earth as a sphere.

A symmetric distance matrix indicating distances between locations is computed. Since the size of the distance matrix is (*67* x *67*), only a small part of it is shown in Table 5.1 as an example. In Table 5.1, settlements used in the case are shown with the numbers in rows and columns. Here number one represents the origin point and Aşkale district is the origin point of this case. Other numbers represents the villages of the Aşkale district, and these villages are sequenced alphabetically.

In the first experiment, the study is examined as a TSP and MMAS algorithm which is proposed in Chapter 3 is used as the solution approach. It is found that when only MMAS algorithm is applied, MMAS produces a solution with length of 296.49 kilometres and the tour of this solution is shown in Figure 5.2. The axes in Figure 5.2 display latitudes and longitudes of the settlements.

As the second experiment, MMAS+2-opt algorithm is used in the computational study. The algorithm is run for five times and for each run, length of the solution and the time for computation are displayed in Table 5.2. The algorithm gives a result in 56.98 seconds on average. The best found solution in the computation has length of 267.37 kilometres and Figure 5.3 shows the resulting tour of the solution.

Table 5.1 Part of the computed distance matrix of the settlements

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	. . .
1	1E+15	9.170465	19.27137	25.9176	19.41504	11.64471	13.95534	9.707654	12.98305	11.39988	25.30122	20.12766	4.567811	14.96137	14.92531	13.46594	13.54386	7.783018	2.118878	13.85449	. . .
2	9.170465	1E+15	21.17767	34.76927	11.03919	16.03319	23.09541	9.497674	19.04588	7.47864	18.21158	28.86603	12.42971	9.538655	6.283831	16.80149	10.61576	2.944841	10.77909	22.65917	. . .
3	19.27137	21.17767	1E+15	27.56481	22.50339	30.82771	25.15915	28.05365	10.04458	27.96758	21.80331	22.57245	23.15742	15.27099	21.44065	5.976283	30.92741	18.30826	17.87578	19.56487	. . .
4	25.9176	34.76927	27.56481	1E+15	43.23929	29.77305	14.27114	33.69904	19.05818	36.66856	46.45321	5.970495	24.83769	36.55619	39.55447	24.43315	37.92526	32.5819	24.05141	12.11199	. . .
5	19.41504	11.03919	22.50339	43.23929	1E+15	26.89731	33.20058	19.82932	25.12722	16.61999	8.117916	37.27614	23.265	7.413543	4.762253	20.81873	19.32268	11.63203	20.48065	31.45129	. . .
6	11.64471	16.03319	30.82771	29.77305	26.89731	1E+15	15.50216	7.483336	23.31693	11.38691	34.2388	25.13305	7.80951	24.85352	22.16204	24.9247	10.51601	16.75135	12.96763	19.70052	. . .
7	13.95534	23.09541	25.15915	14.27114	33.20058	15.50216	1E+15	19.94328	15.17634	23.28596	38.32032	10.02663	11.39696	27.81526	28.83599	19.74331	24.11628	21.63465	12.72102	6.469427	. . .
8	9.707654	9.497674	28.05365	33.69904	19.82932	7.483336	19.94328	1E+15	22.68686	3.939266	27.50016	28.33313	8.883273	18.94581	15.19539	22.60632	4.228027	11.11046	11.7624	22.22461	. . .
9	12.98305	19.04588	10.04458	19.05818	25.12722	23.31693	15.17634	22.68686	1E+15	23.9529	27.47856	13.43274	15.55057	17.98567	22.19541	5.434341	26.40434	16.23914	10.98362	9.558004	. . .
10	11.39988	7.47864	27.96758	36.66856	16.61999	11.38691	23.28596	3.939266	23.9529	1E+15	24.54088	31.10299	11.93484	16.92586	12.21059	22.96829	3.137412	9.896766	13.51466	24.87555	. . .
11	25.30122	18.21158	21.80331	46.45321	8.117916	34.2388	38.32032	27.50016	27.47856	24.54088	1E+15	40.63664	29.59775	10.50916	12.34811	22.32436	27.34796	17.87429	25.86163	35.39616	. . .
12	20.12766	28.86603	22.57245	5.970495	37.27614	25.13305	10.02663	28.33313	13.43274	31.10299	40.63664	1E+15	19.45372	30.64292	33.58652	18.86357	32.55585	26.63374	18.20383	6.274085	. . .
13	4.567811	12.42971	23.15742	24.83769	23.265	7.80951	11.39696	8.883273	15.55057	11.93484	29.59775	19.45372	1E+15	19.39319	18.59989	17.21079	13.11007	11.78622	5.311786	13.39038	. . .
14	14.96137	9.538655	15.27099	36.55619	7.413543	24.85352	27.81526	18.94581	17.98567	16.92586	10.50916	30.64292	19.39319	1E+15	6.563523	13.4456	20.03969	8.103688	15.38205	25.1291	. . .
15	14.92531	6.283831	21.44065	39.55447	4.762253	22.16204	28.83599	15.19539	22.19541	12.21059	12.34811	33.58652	18.59989	6.563523	1E+15	18.57153	15.09473	7.211575	16.17356	27.57644	. . .
16	13.46594	16.80149	5.976283	24.43315	20.81873	24.9247	19.74331	22.60632	5.434341	22.96829	22.32436	18.86357	17.21079	13.4456	18.57153	1E+15	25.78475	13.85671	11.95727	14.72116	. . .
17	13.54386	10.61576	30.92741	37.92526	19.32268	10.51601	24.11628	4.228027	26.40434	3.137412	27.34796	32.55585	13.11007	20.03969	15.09473	25.78475	1E+15	12.99593	15.65024	26.43084	. . .
18	7.783018	2.944841	18.30826	32.5819	11.63203	16.75135	21.63465	11.11046	16.23914	9.896766	17.87429	26.63374	11.78622	8.103688	7.211575	13.85671	12.99593	1E+15	8.962681	20.52094	. . .
19	2.118878	10.77909	17.87578	24.05141	20.48065	12.96763	12.72102	11.7624	10.98362	13.51466	25.86163	18.20383	5.311786	15.38205	16.17356	11.95727	15.65024	8.962681	1E+15	11.94731	. . .
20	13.85449	22.65917	19.56487	12.11199	31.45129	19.70052	6.469427	22.22461	9.558004	24.87555	35.39616	6.274085	13.39038	25.1291	27.57644	14.72116	26.43084	20.52094	11.94731	1E+15	. . .
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

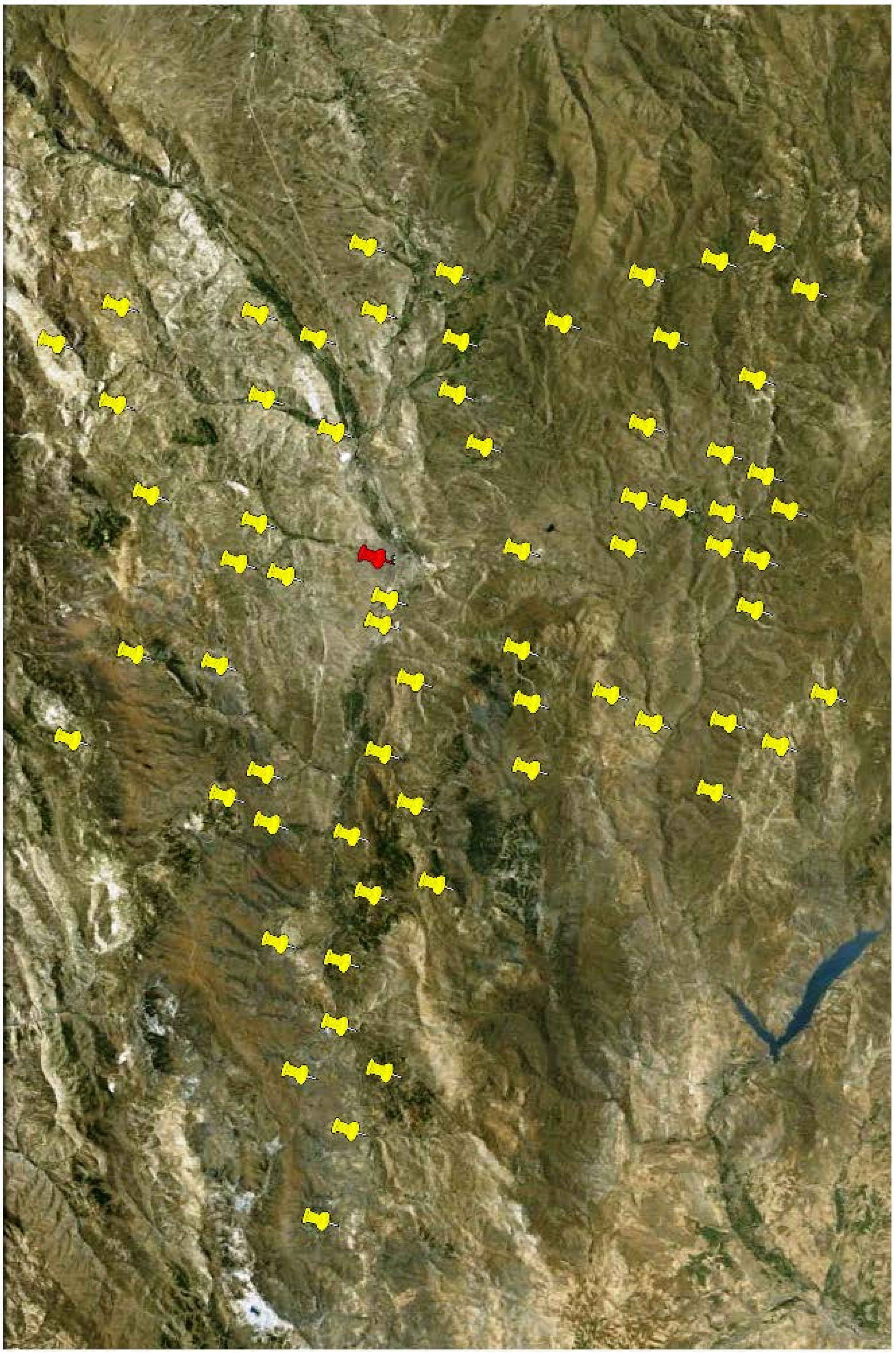
Figure 5.1 Map representation of the villages in Askale district

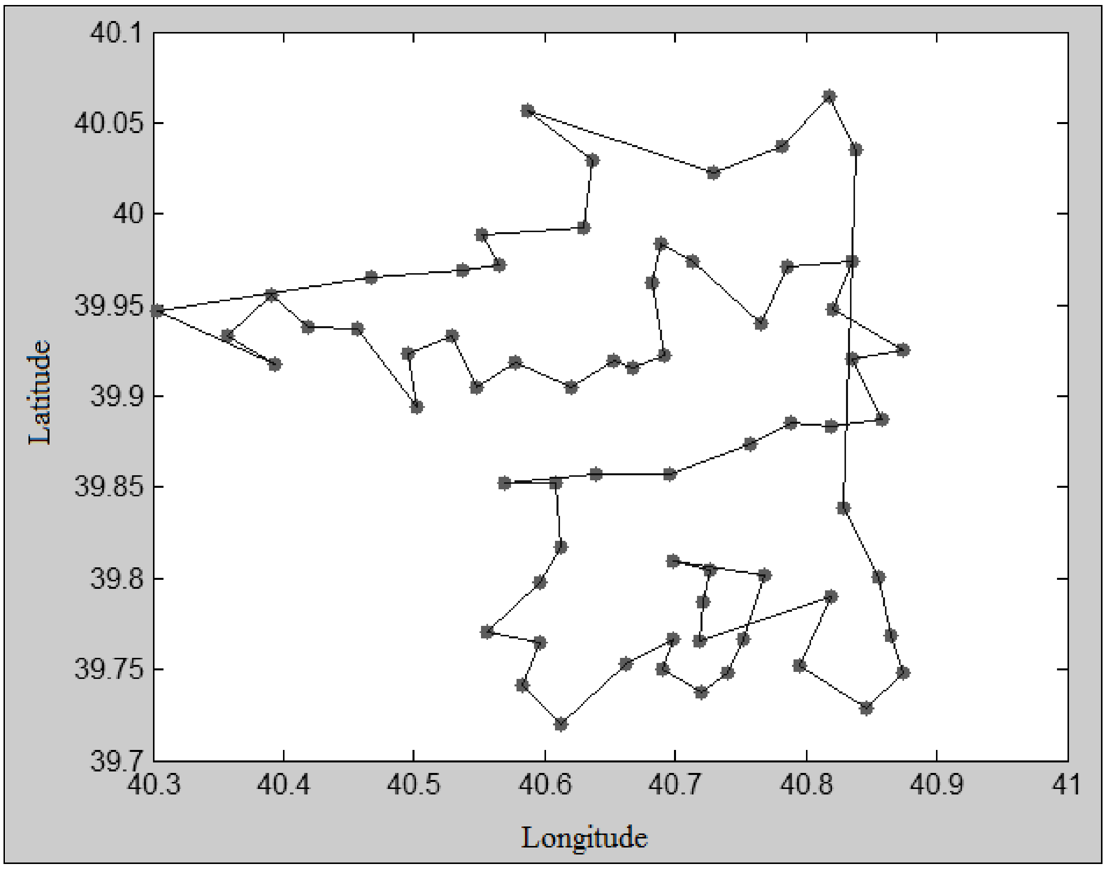

Figure 5.2 Best tour found with MMAS experiment

Table 5.2 Results of the MMAS+2-opt experiment

Number	**Result in Kilometres**	**Time in Seconds**
1	272.1	55.75
2	**267.37**	**56.83**
3	272.47	58.97
4	272.02	56.03
5	275.09	57.33
Average	**271.81**	**56.98**

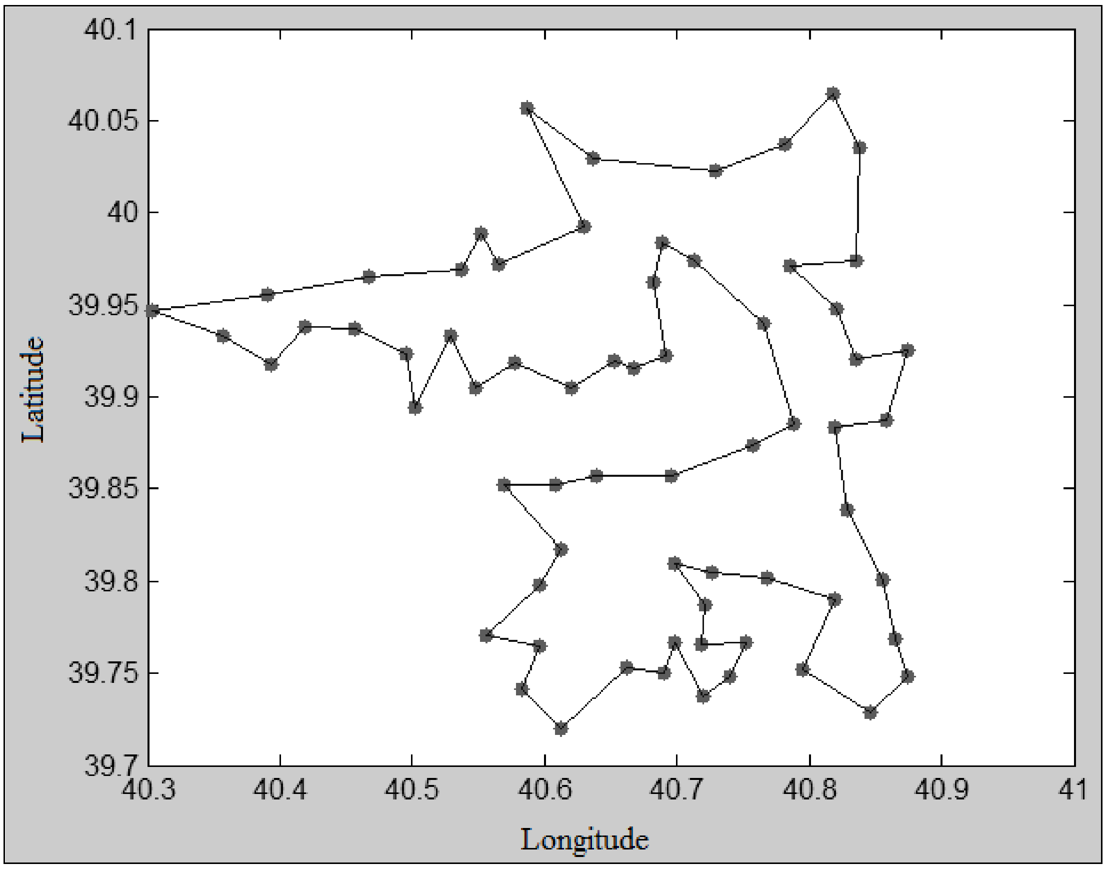

Figure 5.3 Best tour found in the MMAS+2-opt experiment

The third experiment is the MMAS+3-opt algorithm applied to the case. The algorithm is run for five times again and for each run, resulting tour length and the time for computation is displayed in Table 5.3. The algorithm results with 74.13 seconds on average. Best found solution in the computation has length of 265.73 kilometres and Figure 5.4 shows the resulting tour of the solution.

Table 5.3 Results of the MMAS+3-opt experiment

Number	**Result in Kilometres**	**Time in Seconds**
1	268.87	73.03
2	268.11	73.53
3	**265.73**	**76.62**
4	269.94	73.84
5	267.32	73.62
Average	**267.99**	**74.13**

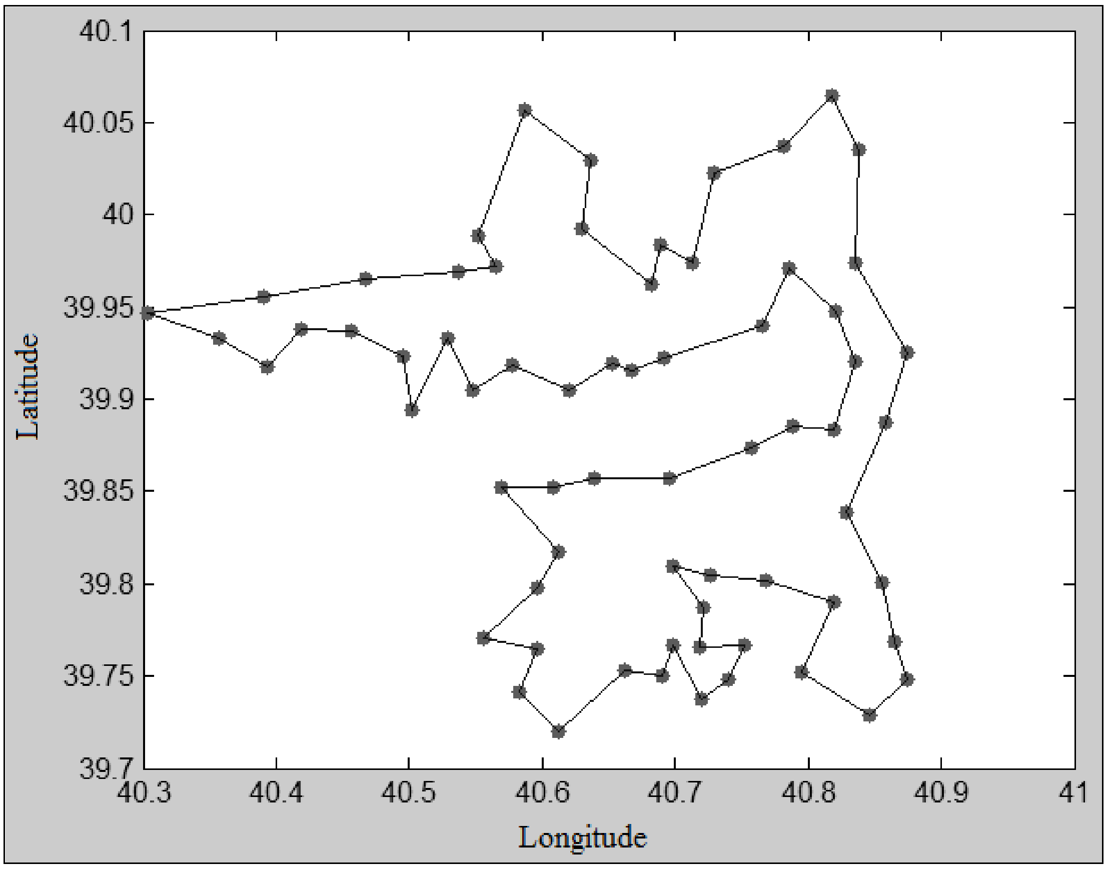

Figure 5.4 Best tour found in the MMAS+3-opt experiment

After TSP experiments, the best solution found in these MMAS+2-opt and MMAS+3-opt algorithms is examined as if the problem is TRP. In other words, since the objective of TSP is different than objective of TRP, the objective function of the best tour found in TSP is converted into the objective of minimizing total waiting times of the customers situated at different locations. Thus, the best found solution which gives a length of 265.73 kilometres in the MMAS+3-opt experiment is examined for this purpose. The objective function of the tour becomes the total latency of all settlements. The result of the converted objective value of this solution is found as 8910.9 kilometres.

In the fourth experiment, the case study is studied as a TRP. Graph of the paths between locations is accepted as complete. Therefore it results with the expression that there is a path for each village to all of the villages. As the solution approach, hybrid GA which is proposed for TRP in the previous chapter is used for the case of routing a repairman under heavy snow conditions. Parameters that are used in the computational study are shown in Table 5.4. For this case, parameter values are determined as the same with the parameter values for TRP-100 instance set.

Table 5.4 Parameters used in hybrid GA algorithm for the case

Parameter	**Value**
Population size *(p)*	350
Crossover rate	(*p**0.8)
Number of GA iterations	500
Mutation rate	0.2
Number of two exchange iterations	10
2-opt rate	0.5
Number of 2-opt iterations	200

After the computational study, resulting tour of the TRP study is given in Figure 5.5. After five run, objective function values of the resulting tours and computation times are displayed in Table 5.5. The algorithm gives a result in 126.2 seconds on average. The best objective function value is found as 8071.1 kilometres.

Table 5.5 Results of the hybrid GA experiment

Number	**Result in Kilometres**	**Time in Seconds**
1	8389.1	122
2	**8071.1**	**136**
3	8127.5	140
4	8127.5	122
5	8402.9	111
Average	**8223.6**	**126.2**

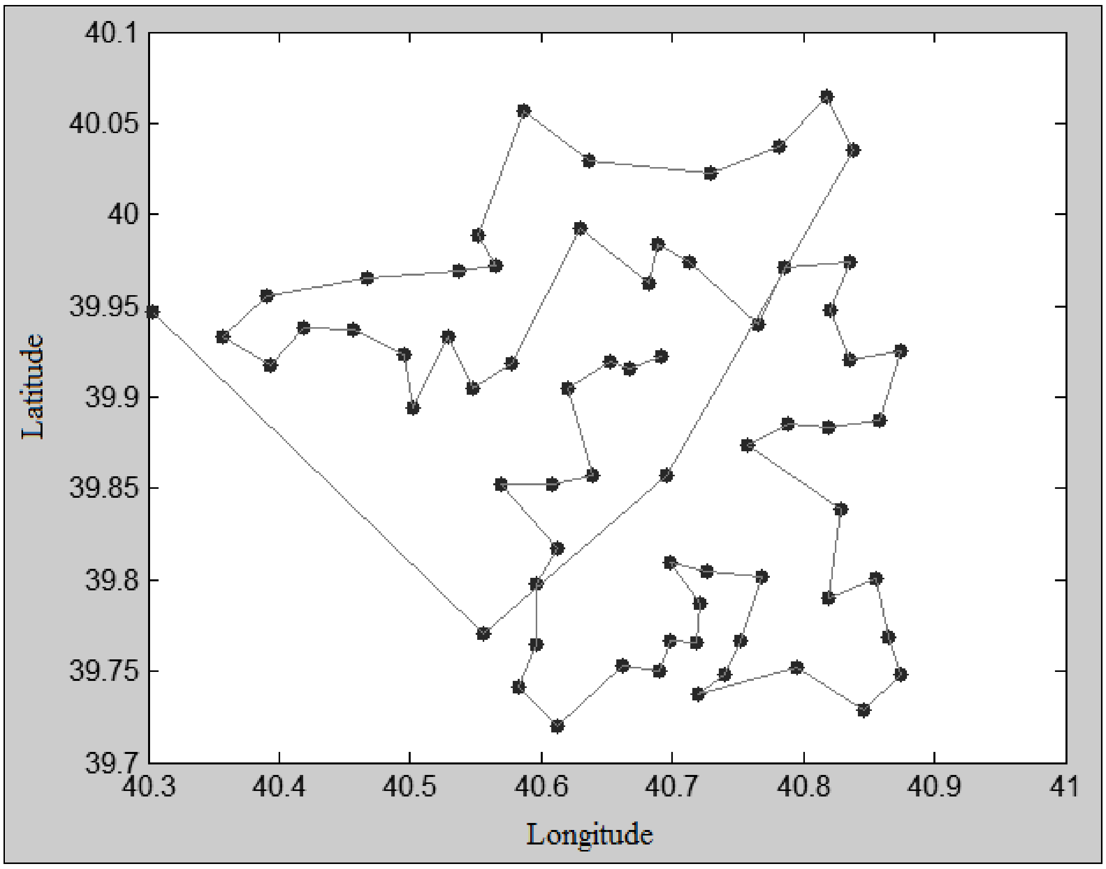

Figure 5.5 Best tour found in the hybrid GA experiment

5.5 Results and Discussion

In the case, at first, the problem is examined as a TSP and MMAS algorithm is applied to the case. After that, to make a local search around the solution, MMAS+2-opt algorithm is applied to the problem. The length of the solution is decreased when MMAS+2-opt algorithm is used. MMAS+3-opt algorithm is then applied to the problem to be able to search more space in the solution area. It is seen that MMAS+3-opt algorithm gives better results on average in comparison with the MMAS+2-opt algorithm. However in MMAS+3-opt experiment, the average computational time is higher than MMAS+2-opt experiment.

After these experiments, it is investigated that if the best found solution in the TSP experiments will also give a good result in case of the problem is examined as TRP. The objective function of the problem is modified to minimize total latency of all customers and after this modification; objective function value of the best found solution in TSP experiments is increased as expected.

As a second approach, the case is studied as a TRP and the hybrid GA algorithm is applied to the problem. After application, it is found that this approach yields better result than the previous assumption of modifying the objective function.

At the end of experimental studies, it can be said that the last experiment in which the problem is examined as TRP and hybrid GA is used to solve this problem, is the most suitable approach to serve the settlements quickly.

The result means that, if the case was examined as TSP and the solution was used for the aim of minimizing total latency of the settlements; it would not be a good decision. Because, when the problem is examined as TRP, it gives a solution with 8071.1 kilometres. Therefore it gives a solution with 8910.9 kilometres in the TSP case with modified objective function.

According to the results of these experiments, the assumption of the graph is complete will be a useful and faster method to find a solution. In such a situation, the only data needed will be the coordinates of the settlements.

On the other hand, in a real case, number of paths that connect settlements with each other is not as many as in the complete graph. There may not be a road between pair of the villages. Thus, in a real situation, some clustering methods will be useful for finding a more practical solution.

The routing operations under heavy snow conditions are common problems for many areas in the world. In the literature, proposed approaches for solving a problem about such a snow disaster situation have focused on assuming it as an arc routing problem - especially as a Chinese postman problem. According to a knowledge acquired from an authority, in a real snow removal operation, the main and primary objective is to clean the highway and the main transportation road; so it will be easier and faster for each of the villages to reach to the district, or the opposite. However, the main objective of this case study is to find out that if TRP is an appropriate problem for representing such a situation and if it can be used in other disaster circumstances. It is understood that such a case can be seen as a TRP with a few modifications and future research will be in this direction. In the future, assumptions made in the case will be decreased and the case will be studied according to its special constraints.

CHAPTER SIX
CONCLUSION

In general, the main objective for every part of a supply chain is to implement its activities from the viewpoint of cost minimization. The cost term used here can be thought as the amount of time, money, resource etc. While the aim of a supplier is to minimize its total cost that arises from operations of making and delivering the product/service to customers; a customer wants to have the corresponding product/service as soon as possible, with a low price and high quality. These aims may change according to special situations. Although the main objective varies from the point of view, in general, it can be said that objectives are in the same direction for both of the sides of a supply chain.

To reach the objective of transportation costs minimization, two types of transportation problems are examined in this thesis. The first one is the well-known traveling salesman problem (TSP). From the research on the TSP literature, it was decided to implement an ant colony optimization based algorithm to TSP; because the method is relatively new and successful in the context of combinatorial optimization. In this study, a MMAS algorithm combined with two local search heuristics is proposed. It is seen that while MMAS ensures an initial solution; 2-opt and 3-opt local search heuristics effectively improve the solution obtained from result of MMAS. Proposed algorithm is applied to well-known TSP instances and the performance of the approach is found to be near optimal; percentage deviations from optimal solution vary between %0 and %8 according to the results. Therefore, it is possible to apply different search algorithms to the problem; better search procedures will be also investigated in the future studies.

The second problem examined is the traveling repairman problem (TRP). An evolution based meta-heuristic, genetic algorithm (GA), is applied as the solution approach for the problem. In the study, a hybrid algorithm which comprises from a GA and a 2-opt local search heuristic is proposed to solve TRP. With this study, it is aimed to develop an efficient and effective algorithm that can be applicable to real life problems. Proposed algorithm is applied to a set of instances that have been studied previously. In general, the performance of the approach can be said to be competitive with the previous studies in the literature, from the viewpoint of percentage deviations from the lower bounds. Computation time of the algorithm is also seems as reasonable according to previous studies.

Since TRP is an adaptable problem to real life examples, a case study for TRP is created and examined as the last study. The case study is thought as a snow disaster situation. Aşkale district of Erzurum city from the East Anatolian Region in Turkey is accepted as the origin point for a repairman and demand points are the villages of this district. In this conceptual application, the repairman tries to reach all of the villages as soon as possible to meet the residents' requirements. Such a snow disaster situation is mainly studied as a snow removing problem and this problem is examined then as an arc routing problem in the literature. Nevertheless in this study, the objective is considered as meeting human needs quickly.

In the application, some assumptions are made; therefore TRP is examined with the aim of minimizing total latency of the customers located in different settlements. At first, the problem is examined as TSP then MMAS algorithm is applied to the case. After that, MMAS+2-opt and MMAS+3-opt algorithms are applied. It is found that MMAS+3-opt algorithm yields best results on average. The question of "What would be the result when the objective function of the best found tour in TSP experiments is converted to latency minimization?" is answered then by modifying the objective function of the resulting tour. As the last study, the problem is examined as TRP and hybrid GA is applied to the case. After evaluation of the results, it is seen that the TRP approach is the most suitable approach for the case and it yields better results than the converted solution found in the TSP experiment.

In conclusion, this thesis aims to focus on two fundamental problems related to transportation that could arise in every stage of supply chain systems. These problems are the well-known TSP and a more recent problem, TRP, compared to TSP. In the investigation process of these problems, it is realized that these problems are the main transportation problems only differ in the point of view. While TSP considers a supplier oriented view; other one TRP considers a customer oriented view. Both of two problems have importance in the transportation activities. Therefore, in this thesis, solution approaches are proposed for these problems.

REFERENCES

Angel-Bello, F., Alvarez, A., & García, I. (2013). Two improved formulations for the minimum latency problem. *Applied Mathematical Modelling*, *37,* 2257-2266.

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. Princeton, NJ: Princeton University Press.

Archer, A., & Williamson, D. P. (2003). Faster approximation algorithms for the minimum latency problem. *In Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms,* Baltimore, MD, 88-96.

Aşkale Kaymakamlığı (n.d.). Retrieved November 29, 2012, from http://www.askale.gov.tr/

Balseiro, S. R., Loiseau, I., & Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research*, *38*, 954-966.

Bianco, L., Mingozzi, A., & Ricciardelli, S. (1993). The traveling salesman problem with cumulative costs. *Networks*, *23*, 81-91.

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., & Sudan, M. (1994). The minimum latency problem. *In STOC '94 Proceedings of the twenty-sixth annual ACM symposium on Theory of computing,* Montreal, Canada, 163-171.

Bock, F. (1958). An algorithm for solving traveling-salesman and related network optimization problems. Presented at the Operations Research Society of America 14th National Meeting, St. Louis.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1997). A new rank-based version of the ant system: A computational study. *Technical Report April 97*. Vienna, Austria: University of Vienna, Institute of Management Science.

Cergibozan, C., & Tasan, A. S. (2012). A hybrid genetic algorithm for the travelling repairman problem. *In Proceedings of 54th conference of the OR Society (OR54)*, Edinburgh, Scotland, 82.

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

Chaudhuri, K., Godfrey, B., Rao, S., & Talwar, K. (2003). Paths, trees, and minimum latency tours. *In Proceedings of 44th symposium on foundations of computer science (FOCS)*, 36-45.

Chen, S. -M., & Chien, C. -Y. (2011). Parallelized genetic ant colony systems for solving the traveling salesman problem. *Expert Systems with Applications*, *38*, 3873-3883.

Cordón, O., Fernández de Viana, I., & Herrera, F. (2002). Analysis of the best-worst ant system and its variants on the TSP. *Mathware & Soft Computing*, *9*, 177-192.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, *6*, 791-812.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, *2*, 393-410.

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, *1*(4), 28-39.

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, *344*, 243-278.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, & F. Glover, (Eds.), *New Ideas in Optimization* (11-32). London: McGraw Hill.

Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life, 5,* 137-172.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1,* 53-66.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). Positive feedback as a search strategy. *Technical Report 91-016*. Italy: Politecnico di Milano.

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

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Massachusetts: MIT Press.

Dorigo, M., & Stützle, T. (2010). Ant colony optimization: Overview and recent advances. In M. Gendreau, & J. -Y. Potvin, (Eds.), *Handbook of metaheuristics, International Series in Operations Research & Management Science, 146*, (227-263). Springer Science+Business Media, LLC.

Feo, T. A., Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*, 109-133.

Fischetti, M., Laporte, G., & Martello, S. (1993). The delivery man problem and cumulative matroids. *Operations Research*, *41,* 1055-1064.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research, 4,* 61-75.

Gambardella, L. M., & Dorigo, M. (1995). Ant-q: A reinforcement learning approach to the traveling salesman problem. In A. Prieditis & S. Russell, (Eds.), *Proceedings of the twelfth international conference on machine learning (ML-95)* (252-260). California: Morgan Kaufmann Publishers.

Gambardella, L. M., & Dorigo, M. (1996). Solving symmetric and asymmetric TSPs by ant colonies. *In Proceedings of the 1996 IEEE international conference on evolutionary computation (ICEC'96)*, New Jersey: IEEE Press, 622-627.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: John Wiley & Sons.

Gendreau, M. (2003). An introduction to tabu search. In F. Glover, & G. Kochenberger, (Eds.), *Handbook of Metaheuristics*, (37-54). Kluwer Academic Publishers.

Ghiani, G., & Improta, G. (2000). An algorithm for the hierarchical Chinese postman problem. *Operations Research Letters, 26*, 27-32.

Glossary of Terms - Council of Supply Chain Management Professionals (n.d.). Retrieved December 26, 2012, from http://cscmp.org/resources-research/glossary-terms

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13,* 533-549.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison Wesley Longman.

Google Maps (n.d.). Retrieved November 29, 2012, from http://maps.google.com/

Guntsch, M., & Middendorf, M. (2002). A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, & G. R. Raidl (Eds.), *Applications* *of Evolutionary Computing-Lecture Notes in Computer Science*, *2279*, (72-81). Berlin: Springer-Verlag Berlin Heidelberg.

Gutin, G., & Punnen, A. P. (Eds.). (2002). *The traveling salesman problem and its variations*. Netherlands:Kluwer Academic Publishers.

Hamilton biography (June, 1998). Retrieved December, 2012, from http://www-history.mcs.st-andrews.ac.uk/Biographies/Hamilton.html

Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts & J. K. Lenstra, (Eds.), *Local Search in Combinatorial Optimization* (215-310). London:John Wiley & Sons.

Juang, C. -F., & Chang P. -H. (2011). Recurrent fuzzy system design using elite-guided continuous ant colony optimization. *Applied Soft Computing*, *11*, 2687-2697.

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. *Handbooks in Operations Research and Management Science*, *7*, 225-330.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller, & J. W. Thatcher, (Eds.), *Complexity of Computer Computations* (85-103). New York: Plenum Press.

Kirkman biography (December, 1996). Retrieved December, 2012, from http://www-history.mcs.st-andrews.ac.uk/Biographies/Kirkman.html

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

Korteweg, P., & Volgenant, T. (2006). On the hierarchical Chinese postman problem with linear ordered classes. *European Journal of Operational Research*, *169*, 41-52.

Land, A. H., & Doig, A.G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, *28*, 497-520.

Lawler, E. L., & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research, 14,* 699-719.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (Eds.). (1985). *The travelling salesman problem*: A guided tour of combinatorial optimization. Chichester: John Wiley & Sons.

Lemieux, P. F., & Campagna, L. (1984). The snow ploughing problem solved by a graph theory algorithm. *Civil Engineering Systems, 1,* 337-341.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J., 44*, 2245-2269.

Lin, S., & Kernighan, B.W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research, 21*, 498-516.

Logistics (business) -- Britannica Online Encyclopedia (n.d.). Retrieved December 26, 2012, from
http://www.britannica.com/EBchecked/topic/346422/logistics

Lucena, A. (1990). Time-dependent traveling salesman problem-the deliveryman case. *Networks, 20,* 753-763.

Mahalanobis, P. C. (1940). A sample survey of the acreage under jute in Bengal. *Sankhyā: The Indian Journal of Statistics*, *4*(4), 511-530.

MathWorks - MATLAB and Simulink for Technical Computing (n.d.). Retrieved July 07, 2011, from
http://www.mathworks.com/

Menger, K. (1932). Das botenproblem. *Ergebnisse Eines Mathematischen Kolloquiums*, *2,* 11-12.

Merkle, D., & Middendorf, M. (2005). Swarm intelligence. In E. K. Burke, & G. Kendall, (Eds.), *Search Methodologies - Introductory tutorials in optimization and decision support techniques,* (401-435). Springer Science+Business Media, Inc.

Merrill M. Flood (September, 2012). Retrieved December, 2012, from http://en.wikipedia.org/wiki/Merrill_M._Flood

Mitchell, M. (1996). *An introduction to genetic algorithms*. The MIT Press.

Mitchell, J. E. (2002). Branch-and-cut algorithms for combinatorial optimization problems. In P. M. Pardalos, & M. G. C. Resende, (Eds.). *Handbook of Applied Optimization* (65-77). Oxford: Oxford University Press.

Ngueveu, S. U., Prins, C., & Calvo, R. W. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 37,* 1877-1885.

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research, 63,* 513-623.

Perrier, N., Langevin, A., & Campbell, J. F. (2007). A survey of models and algorithms for winter road maintenance part IV: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research, 34,* 258-294.

Reeves, C. (2003). Genetic algorithms. In F. Glover, & G. Kochenberger, (Eds.), *Handbook of Metaheuristics*, (55-82). Kluwer Academic Publishers.

Resende, M. G. C., & Ribeiro, C. C. (2010). Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications. In M. Gendreau, & J. - Y. Potvin, (Eds.), *Handbook of metaheuristics, International Series in Operations Research & Management Science, 146*, (283-319). Springer Science+Business Media, LLC.

Robinson, J. B. (1949). On the hamiltonian game: A traveling salesman problem. *Technical report, RAND Research Memorandum RM-303*.

Salehipour, A., Sörensen, K., Goos, P., & Bräysy, O. (2008). An efficient GRASP + VND metaheuristic for the traveling repairman problem. *Working paper*.

University of Antwerp, Faculty of Applied Economics.

Salehipour, A., Sörensen, K., Goos, P., & Bräysy, O. (2011). Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem. *4OR: A Quarterly Journal of Operations Research 9*, 189-209.

Silva, M. M., Subramanian, A., Vidal, T., & Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research*, *221*, 513-520.

Stützle, T., & Hoos, H. H. (2000). Max–min ant system. *Future Generation Computer Systems*, *16*, 889-914.

Taillard, E. D. (1999). Ant systems. *Technical Report IDSIA-05-99*. Lugano: IDSIA.

Traveling Salesman Problem (August, 2012). Retrieved August 10, 2011, from http://www.tsp.gatech.edu/

TSPLIB (August, 2008). Retrieved August 10, 2011, from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Türkiye Haritası (n.d.). Retrieved November 29, 2012, from http://www.haritatr.com/

Voß, S. (2001). Meta-heuristics: The state of the art. In A. Nareyek, (Ed.), *Local Search for Planning and Scheduling, Lecture Notes in Computer Science, 2148,* (1-23). Springer-Verlag Berlin Heidelberg.

Voß, S., Martello, S., Osman, I. H., & Roucairol, C. (Eds.). (1999). *Meta-heuristics-Advances and trends in local search paradigms for optimization*. Netherlands: Kluwer Academic Publishers.

Whitney biography (August, 2005). Retrieved December, 2012, from http://www-history.mcs.st-and.ac.uk/Biographies/Whitney.html

Wilkins, D. R. (June 26, 2000). *Sir William Rowan Hamilton (1805-1865):*

Mathematical Papers. Retrieved December, 2012, from http://www.maths.soton.ac.uk/EMIS/classics/Hamilton/

Winston, W. L. (2004). *Operations research: Applications and algorithms* (4th ed.). Canada: Thomson Brooks/Cole.

Wu, B. Y., Huang, Z. -N., & Zhan, F. -J. (2004). Exact algorithms for the minimum latency problem. *Information Processing Letters*, *92*, 303-309.

Yerel Yönetimler Portalı (n.d.). Retrieved November 29, 2012, from http://www.yerelnet.org.tr/

Zäpfel, G., Braune, R., & Bögl M. (2010). *Metaheuristic search concepts: A tutorial with applications to production and logistics*. Springer-Verlag Berlin Heidelberg.

Zhao, N., Wu, Z., Zhao, Y., & Quan, T. (2010). Ant colony optimization algorithm with mutation mechanism and its applications. *Expert Systems with Applications*, *37*, 4805-4810.