A Comparative Analysis of ACO and BCO for Solving the TSP

Dissertation

Submitted To

Central Department of Computer Science & Information Technology
Institute of Science and Technology

Tribhuvan University, Kathmandu, Nepal

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science & Information Technology

Submitted By

Niranjan Kathayat
Roll No. : 20 (2011-2013)
June 05, 2014

Under Supervision of:

Asst. Prof. Nawaraj Paudel

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Date: 05-06-2014

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

… … … … … … …

Niranjan Kathayat
M.Sc. CSIT (2011-2013)
Central Department of Computer Science
and Information Technology,
Institute of Science and Technology,
Kirtipur, Kathmandu, Nepal

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Date: 05-06-2014

Supervisor's Recommendation

I hereby recommend that the dissertation prepared under my supervision by **Mr. Niranjan Kathayat** entitled "**A Comparative Analysis of ACO and BCO for Solving the TSP**" be accepted as in fulfilling partial requirement for the completion of Masters Degree of Science in Computer Science & Information Technology. In my best knowledge this is an original work in computer science.

Asst. Prof. Nawaraj Paudel

Head,
Central Department of Computer Science
and Information Technology,
Institute of Science and Technology,
Kirtipur, Kathmandu, Nepal

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Date: 05-06-2014

LETTER OF APPROVAL

We certify that we have read this dissertation work and in our opinion it is appreciable for the scope and quality as a dissertation in the partial fulfillment of the requirements of Masters Degree of Science in Computer Science & Information Technology.

Evaluation Committee

Asst. Prof. Nawaraj Paudel
Head of Department
Central Department of Computer Science
& Information Technology
Tribhuvan University
Kirtipur

Asst. Prof. Nawaraj Paudel
Head of Department
Central Department of Computer Science
and Information Technology (T.U)
(Supervisor)

(External Examiner)

(Internal Examiner)

Acknowledgements

First and foremost, I would like to express my sincere gratitude to all the people who provide their immense help, support, guidance, stimulating suggestions and encouragement all the time during the preparation of this dissertation entitled **"A Comparative Analysis of ACO and BCO for Solving the TSP".** This research work has been performed under the Central Department of Computer Science and Information Technology (**Tribhuvan University),** Kirtipur. I am very grateful to my department for giving me an enthusiastic support.

I would like to express my sincere gratitude to my supervisor **Asst. Prof. Nawaraj Poudel,** Head of the Department of Central Department of Computer Science and Information Technology. This work would have not been possible without his encouragement. He always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under his supervision. This research would not have been possible without his advices and patience.

Most importantly, I would like to thank to respected Lecturers Prof. Dr. Sashidhar Ram Joshi, Prof. Sudarsan Karanjit, Prof. Dr. Subarna Sakya, Mr. Min Bahadur Khati, Mr. Dheeraj Kedar Pandey, Mr. Sarbin Sayami, Mrs. Lalita Sthapit, Mr. Jagdish Bhatta, Mr. Arjun Singh Saud, Mr. Bishnu Gautam of CDCSIT, TU, for providing me such a broad knowledge and inspirations.

My greatest thanks are to my parents who bestowed ability and strength in me to complete this work. I am deeply indebted to my parents and dear friends for their inspiration and ever encouraging moral support, which enabled me to pursue my studies. Special thanks to members of educational organizations that I have been working, for their endless motivation, constant mental support and love which have been influential in whatever I have achieved so far.

I wish to thank to all my colleagues and friends especially Mr. Prakash Datt Bhatt, Mr. **Chhetra Bahadur Chhetri**, Mr. Dabal Singh Mahara, Mr. Keshav Bahadur Dhami, Mr. Ashok Pandey, Mr. Ganesh Bahadur Ayer & Mr. Bhupendra Saud for supporting me directly and indirectly in this research work.

I have done my best to complete this research work. Suggestions from the readers are always welcomed, which will improve this work.

Last but not the least; I would like to thank almighty God for not letting me down at the time of crisis and showing the silver lining in the dark clouds.

Abstract

Combinatorial optimization problems are very difficult to solve, since these problems span a large space of NP-hard problems. Various solution models are introduced and used to solve these problems. The Travelling Salesman Problem (TSP) is an intractable NP-hard combinatorial optimization problem studied in operations research and theoretical computer science. Exact methods, heuristics, evolutionary approaches, and meta-heuristic models are widely used to solve TSP like problems.

Ant Colony Optimization (ACO) is a modern and very popular Swarm Intelligence optimization paradigm inspired by the ability of ant colonies to find shortest paths between their nest and a food source. ACO is one of the high performance computing methods for Travelling Salesman Problem (TSP). It still has some drawbacks such as stagnation behavior, long computational time, and premature convergence problem on TSP. Those problems will be more obvious when the considered problem size increases.

Bee Colony Optimization (BCO) meta-heuristic belongs to the group of Swarm Intelligence techniques. BCO is the name given to the collective food foraging behavior of honey bee. The bee system is a standard example of organized team work, well coordinated interaction, coordination, labor division, simultaneous task performance, specialized individuals, and well-knit communication. The BCO uses a similarity among the way in which bees in nature look for a food, and the way in which optimization algorithms search for an optimum in combinatorial optimization problems.

The Swarm-based Algorithm is a search algorithm capable of locating good solutions efficiently. The algorithm could be considered as belonging to the category of “Intelligent Optimization Tools”. General ACO and BCO are implemented and tested on benchmark problems from TSPLIB and evaluations are carried out. Evaluation results have shown that BCO has improved **13.35%** optimal path solutions on average and computational time is improved by 1**3.41%** on average.

Keywords: *Combinatorial Optimization, BCO, ACO, TSP, Swarm Intelligence*

List of Abbreviations

CO	**Combinatorial Optimization Problem**
ACO	**Ant Colony Optimization**
TSP	**Travelling Salesman Problem**
DNA	**Deoxyribonucleic Acid**
AAC	**Artificial Ant Colony**
GA	**Genetic Algorithm**
SA	**Simulated Annealing**
SI	**Swarm Intelligence**
BCO	**Bee colony optimization**
AI	**Artificial intelligence**
ABC	**Artificial Bee Colony**

List of Figures

Figure 1.1: Foraging process of ants…..……..……...…2

Figure 1.2: Typical behavior of honey bee foraging…….……4

Figure 1.3: Symmetric TSP Example……..…….7

Figure 3.1: ACO State Transition Rule…….……..14

Figure 3.2: Flow chart of ACO for solving TSP…….17

Figure4.1: Waggle dance of honey bees ……..……20

Figure.4.2: Flowchart of BCO algorithm for TSP…….…….22

Figure 5.3: Optimal path solution comparison…….33

Figure 5.4: Processing time Comparison…….……...……....33

List of Tables

Table5.1. Parameter Values……..……..……..…25

Table5.2. Simulation Results……..……..……..……26-27

Table5.3. Simulation Results……...…….…..……28-29

Table5.4. Comparison Results.....…….……..……30

Table5.5. Comparison Results…...…….……..……..……31

Table5.6. Comparison Results…...…….……..……..……32

Table5.7. Conclusion Results…...…..……..……..……34

Table of Contents

Acknowledgements i

Abstract iii

List of Abbreviations iv

List of Figures v

List of Tables vi

CHAPTER 1

1. Introduction

1.1. Background……..……..…1-4

1.1.1. Ant colony optimization ……...……...1-3

1.1.1.1. Foraging process of ants……....1-3

1.1.2 Bee colony optimization……...3-4

1.1.2.1 Foraging process of bees……...…3-4

1.2. Research Gap and Motivation……..…..5

1.3. Problem Definition…….…….……5-7

1.3.1. TSP Modeling……...……...…….……6-7

1.3.1.1. Symmetric and Asymmetric TSP...…….……...……..….7

1.4. Objective of the Work…….……7

1.5. Structure of the Thesis……..…..7-8

CHAPTER 2

2. Literature Survey

2.1. Exact Methods……..……...9

2.2. Genetic Algorithms…….…….9

2.3. Simulated Annealing Approach……..…….....9-10

2.4. Tabu Search Methods……...10

2.5. Swarm Intelligence…….…….....10-11

2.6. Recent Trend in ACO,BCO and TSP……..……11-13

CHAPTER 3

3. Ant Colony Optimization Algorithm

3.1. Tour Construction…… ……...…….……15

3.2. Pheromone update…… ……..…….…….…….…..15-16

3.3. General ACO Algorithm & Complexity ……..……..…….…....16-17

3.4. Flow chart of ACO for solving TSP…….……17

CHAPTER 4

4. Bee colony optimization Algorithm

4.1. General Description …….…….18

4.2. Formal Definition……...…..19-21

4.2.1. Waggle dance of honey bees…….20-21

4.3. Frame of the BCO Algorithm…….……..21

4.4. Methodology of BCO algorithm…….….21-22

4.4.1. Flowchart of BCO algorithm for Solving the TSP……..…...22

4.5. Applying BCO for solving the TSP……..…….23

CHAPTER 5

5. Implementation & Analysis

5.1 Computational Results……..…….24-29

5.1.1 Parameter Setting……..…..……24

5.1.2 Computational Results for ACO……...25-27

5.1.3 Computational Results for BCO……..……..27-29

5.2 Performance Evaluation of the Improved Algorithm…….....29-32

5.2.1 Results of Comparison……...…….31-32

5.3 Discussions of the Evaluation Results……...…….32-35

5.3.1 Verification of the Improved Algorithm…….……32-35

CHAPTER 6

6 Conclusion

6.1 Summary…….……..……...……..……36
6.2 Research Limitations…….…….…..…..37
6.3 Future Work……...…….……..…..37

References **38-41**

Appendices **42-58**

CHAPTER 1
INTRODUCTION

1.1 Background

Before the study of any research work it is necessary to have domain knowledge and to introduce the some basic terms and terminology related to the research. In this context basic terms and terminologies related to this work are outlined in the following sections;

1.1.1 Ant Colony Optimization

Ant colony optimization was first proposed in 1992 by Marco Dorigo, for solving TSP like problems. Interaction among ants and the environment is based on pheromones where ants can smell the pheromone and tend to choose, probabilistically, paths marked by strong pheromone concentrations. There is a population of ants, where each ant finds solution and then communicating with the other ants. These observations inspired a new type of algorithm called ant algorithm (or ant system). Ant algorithm is originally oriented for TSP like problems.

Ant Colony Optimization (ACO) is a swarm based meta-heuristic method that is inspired by the behavior of real ant colonies. It is one of the techniques for approximate optimization. More specifically, ACO is inspired by the ants' foraging behavior. At the core of this behavior is the indirect communication between the ants by means of chemical pheromone trails, which enables them to find short paths between their nest and food sources [15]. ACO belongs to the class of meta-heuristics, which are approximate algorithms used to obtain good enough solutions to CO problems in a reasonable amount of computation time. In recent years, many research works have been devoted to ACO techniques in different areas.

1.1.1.1 Foraging Process of Ants

A way ants exploit the pheromone to find the shortest path between two points is depicted in figure 1.1. To fix the ideas, suppose that the distances between D and H, between B and H, and between B and D via C are equal to 1, and let C be positioned half the way between D and B. Now let us consider what happens at regular discretized intervals of time: t=0, 1, 2 and so on. Suppose that 30 new ants come to B from A, and 30 to D from E in each time unit, then each ant walks at speed of 1 per time unit, and while walking the ant lays down at time

‘t’ pheromone trail of intensity 1, which, to make the example simpler, evaporates completely and instantaneously in the middle of the successive time interval (t+1, t+2).

At t=0 there is no trail yet, but 30 ants are in B and 30 in D. Their choice about which way to go is completely random. Therefore, on the average 15 ants from each node will go toward H and another 15 ants toward C as shown in figure 1.1(b).

At time instance t=1, 30 new ants that come from A to B find trail of intensity 15 on the path leads to H, laid by the 15 ants who followed the same way from B, and trail intensity of 30 on the path to C, obtained as the sum of the trail laid by the 15 ants that went that way from B and by the 15 ants that reached B coming from D via C as shown in figure 1.1(c). The probability of choosing path is therefore biased, so that the expected number of ants going toward C will be the double of those going toward H: 20 versus 10 respectively. The same is true for the new 30 ants in D which came from E.

This process continues until all of the ants will eventually choose the shortest path.

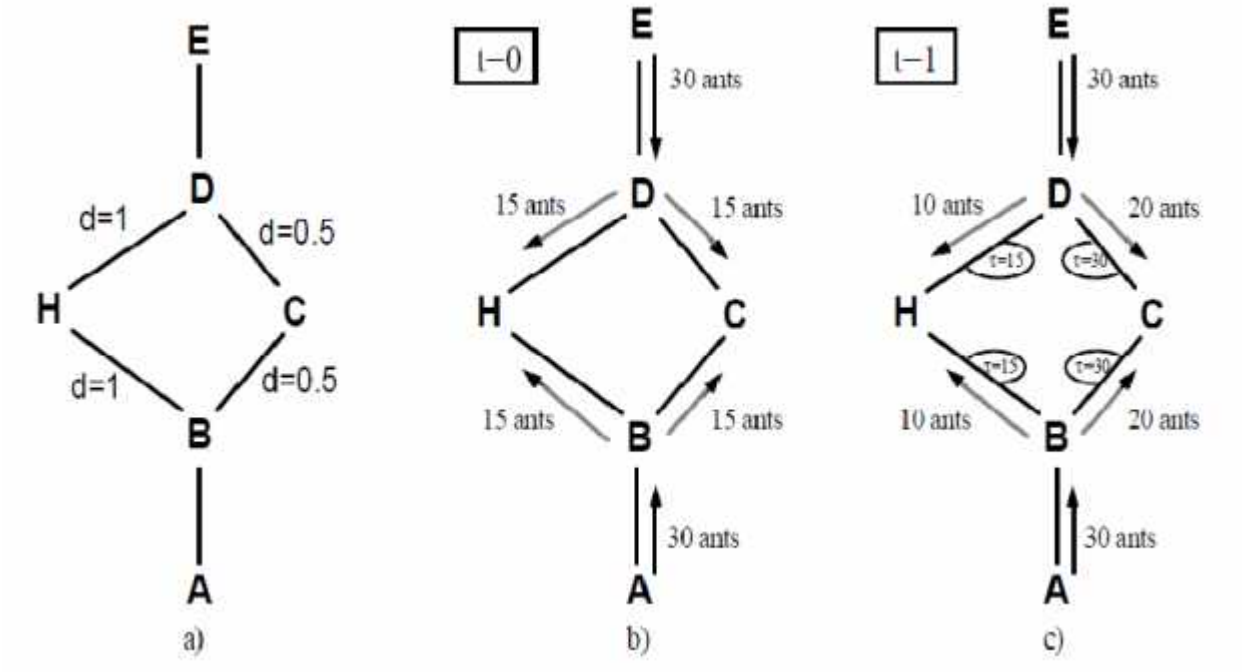


Figure 1.1: Foraging Process of Ants.

a) The initial graph with distances.

b) At time t=0 there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability.

c) At time t=1 trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

The idea is that if at a given point an ant has to choose among different paths, those which were heavily chosen by preceding ants (that is, those with a high trail level) are chosen with higher probability. Furthermore high trail levels are synonymous with short paths.

1.1.2 Bee colony optimization

In 2005, Karaboga proposed an Artificial Bee Colony (ABC) algorithm, which is based on a particular intelligent behavior of honeybee swarms. The Swarm-based Algorithm is a search algorithm capable of locating good solutions efficiently. The algorithm could be considered as belonging to the category of "Intelligent Optimization Tools". There was a great interest between researchers to generate search algorithms that find near-optimal solutions in reasonable running time. ABC is developed based on inspecting the behaviors of real bees on finding nectar and sharing the information of food sources to the bees in the hive.

1.1.2.1 Foraging process of bees

Bee Colony Optimization (BCO) meta-heuristic belongs to the group of Swarm Intelligence techniques. BCO is the name given to the collective food foraging behavior of honey bee. The bee system is a standard example of organized team work, well coordinated interaction, coordination, labor division, simultaneous task performance, specialized individuals, and well-knit communication. The BCO uses a similarity among the way in which bees in nature look for a food, and the way in which optimization algorithms search for an optimum in combinatorial optimization problems [15]. The three agents in Artificial Bee Colony are:

1. The Employed Bee
2. The Onlooker Bee
3. The Scout

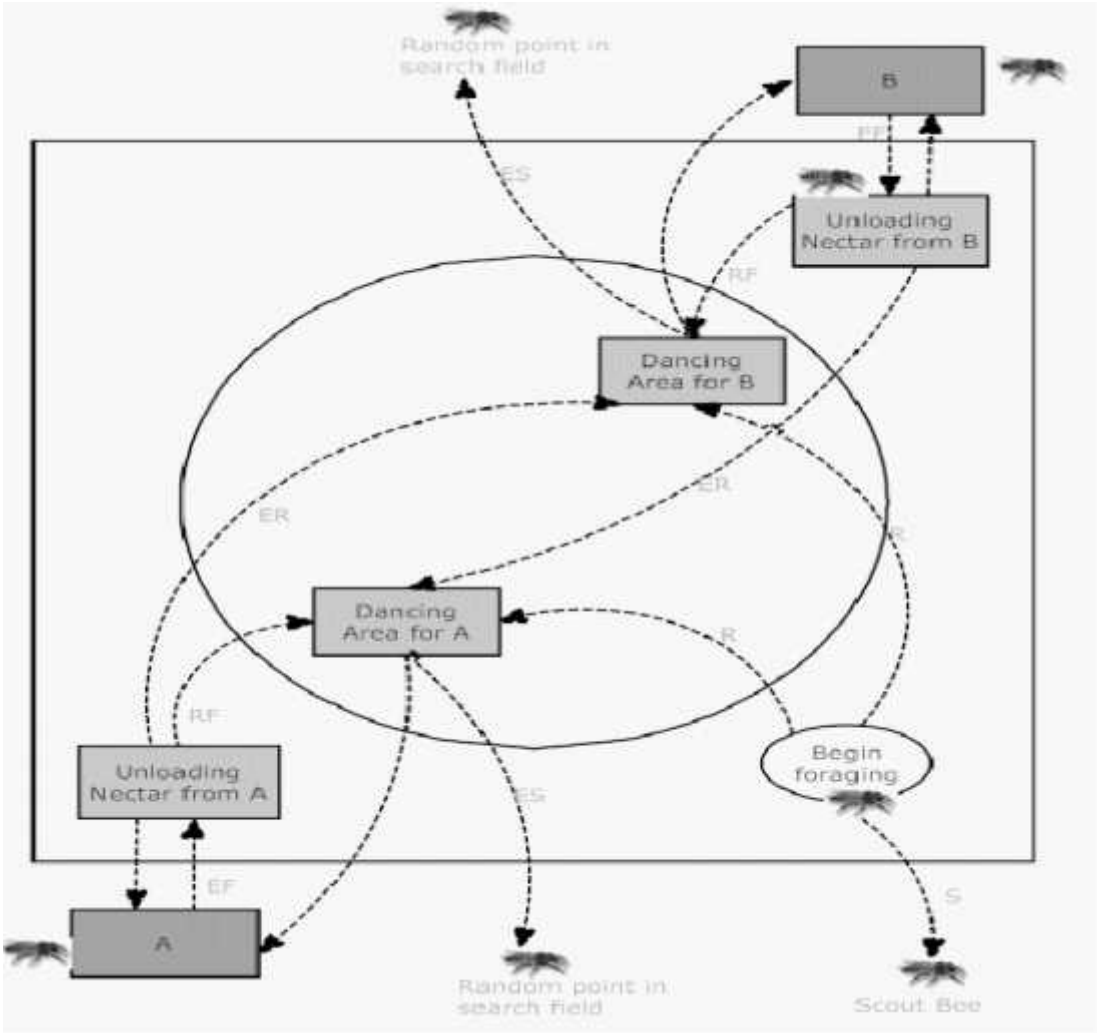


Figure 1.2: Typical Behavior of Honey Bee Foraging [4]

The employed bees are associated with the specific food sources, onlooker bees watching the dance of employed bees within the hive to choose the food source, and scout bees searching for food sources randomly. The onlooker bees and the scout bees are the unemployed bees. Initially, the scout bees discover the positions of all food sources, thereafter, the job of the employed bee starts. An artificial employed bee probabilistically obtains some modifications on the position in its memory to target new food source and find the nectar amount or the fitness value of the new source. Later, the onlooker bee evaluates the information taken from all artificial employed bees and then chooses the final food source with the highest probability related to its nectar number. If the fitness value of new one is higher than that of the previous one, the bee forgets the old one and memorizes the new position based on greedy selection. Then the employed bee whose food source has been exhausted becomes a scout bee to search for the further food sources once again [5].

1.2 Research Gap and Motivation

Although ACO has powerful capacity to find out optimal solutions to combinatorial optimization problems, it has the several problems regarding the stagnation, premature convergence, and slow processing speed. Those problems will be more obvious when the problem size increases. Therefore several algorithms and improvements of the general ACO algorithms were introduced over the years.

In this dissertation, the BCO is introduced to perform the best solution in compare to ACO.

Basic concept of BCO is listed below [15].

- BCO use Non-Stigmergic (Direct) Communication to exchange information between bees.
- A real bee uses waggle dance to recruit other nest mates to discover food source. Thus BCO technique uses the solution quality for recruitment procedure.
- In BCO, the experienced forage type bee maintains the solution quality.
- BCO technique avoids locally optimal solution. It searches for the best solution obtains by the entire bee colony. It is adaptive to changes in the environment.

1.3 Problem Definition

Traveling salesman problem (TSP) is one of the well-known and extensively studied problems in discrete or combinational optimization and asks for the shortest roundtrip of minimal total cost visiting each given city (node) exactly once. TSP is an NP-hard problem and it is so easy to describe and so difficult to solve. Graph theory defines the problem as finding the Hamiltonian cycle with the least weight for a given complete weighted graph. It is widespread in engineering applications and some industrial problems such as machine scheduling, cellular manufacturing and frequency assignment problems can be formulated as TSP [17].

A complete weighted graph G= (N, E) can be used to represent TSP, where N is the set of n cities and E is the set of edges (paths) fully connecting all cities. Each edge *(i,j)€E* is assigned cost *dij*, which is the distance between cities i and j. *dij* can be defined in the Euclidean distance and is given as follows:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

In General, the TSP states that for salesman who wants to visit n different cities, his objective would be to find tour plan that minimizes the cost and travel efforts by visiting each city

exactly once and finally returning back to the starting city. TSP is widely used various domain of research like network communications, transportation systems, manufacturing and resource planning, logistics, etc [33]. This dissertation work will mainly focus on comparative evaluation of these two algorithms: ACO and BCO in terms of time and path length.

The Travelling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pair wise distances, the task is to find a shortest possible tour that each city exactly once. The problem was first formulated as a mathematical problem in 1930, and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known. The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of the microchips. Slightly modified, it appears as sub-problem in many areas, such as DNA sequencing. In these applications, the concept city represents, for example, customers, soldering points, DNA fragments, and the concept distance represents travelling times or costs, or similarity measure between DNA fragments. The main reasons for the choice of TSP for ACO and BCO are as follows:

- TSP is an important NP-hard optimization problem that arises in several applications. It is a problem to which ACO and BCO algorithm can be easily applied.
- It is easily understandable, so that algorithm behavior is not obscured by too many technicalities.
- It is a standard test bed for new algorithmic ideas a good performance on the TSP is often taken as a proof of their usefulness.
- TSP is a typical combinatorial optimization problem. It is often used to validate a certain algorithm, making the comparison with other algorithms an easy work [9].

1.3.1 TSP Modeling

TSP can be modeled as an undirected weighted graph, such that cities act as vertices; paths as edges, and path distance is the edge's length. Often, the model is a complete graph. If no path exists between two cities, adding an arbitrary long edge will complete the graph without affecting the optimal tour.

1.3.1.1 Symmetric and Asymmetric TSP: In the symmetric TSP, the distance between two cities is the same in either direction, forming an undirected graph. The problem can be defined as follows: Let $G = (V,E)$ be a complete undirected graph with vertices W, $|W|=n$, where n is the number of cities, and edges E with edge length dij for (i,j). We focus on the symmetric TSP case in which di,j = dj,i , for all (i,j).

In the asymmetric TSP, paths may not exist in both directions, or the distance might be different, forming directed graph [31, 9]. In this work, TSP refers to the symmetric TSP. Since asymmetric TSP can be converted into symmetric TSP. It is considered as special case of symmetric TSP, because behavior of ACO algorithm is same for either case.

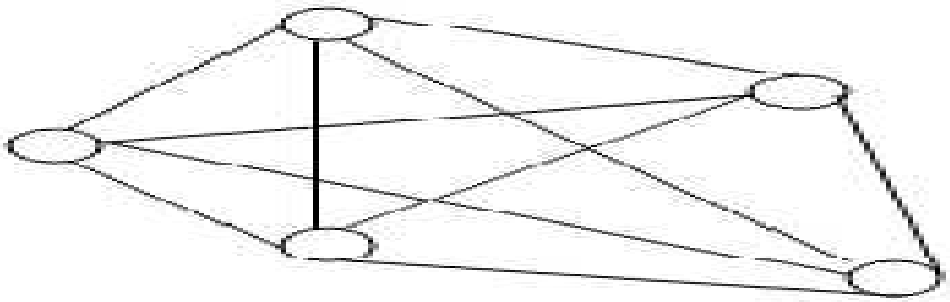


Figure 1.3: Symmetric TSP with 5 Nodes

1.4 Objectives:

The main objective of this dissertation work is to study ACO and BCO algorithms and analyze the performance of these algorithms for solving the TSP.

1.5 Structure of Thesis

Background study of this dissertation work focuses on basic concept of optimization algorithm and the related basic terms which are already mentioned above along with an introduction to ACO and BCO. Almost this chapter introduced the main aim of this dissertation as well as basic concept of optimization problem with ACO and BCO applied in travelling salesman problem. The rest of the material in this study is organized into subsequent four chapters.

Chapter 2 consists of literature review which briefly reviews the related study with this dissertation. Literature review includes summary of several traditional optimization

algorithms. This chapter mainly focuses on recent trend in the field of ACO, BCO and TSP. It also provides the basic framework of general ACO, BCO and its working strategy.

Chapter 3 describes the working of the ACO algorithm in detail. It consists of program development steps of our simulation.

Chapter 4 describes the working of the BCO algorithm in detail.
Chapter 5 contains the research methodology part which shows the flow of our research. This chapter also introduces the implementation detail with data collection and analysis part which shows trace driven input and output results with several analyzing graphs.

Finally, conclusion of this whole dissertation work and the future work which shows guidelines for further research are discussed in Chapter 6. It also briefly discusses the limitations of the work.

CHAPTER 2

LITERATURE SURVEY

This chapter classifies various methods to solve TSP and gives some general information about traditional methods. In this phase of study, recent trend in ACO, BCO and TSP is mainly emphasized. Some of the algorithms deployed for solving TSP problem are discussed below:

2.1 Exact Methods

In 1960, Land and Doig, proposed exact methods for linear programming which are most simple algorithms to solve combinatorial optimization problems. These methods guarantee the solution but computational time is unknown for larger instances. Even though number of exact methods were introduced for solving optimization problem such as Brute-force approach [3], Dynamic Programming methods [22], Branch and Bound [7], Linear Programming [26], but none of them were feasible for larger instances of TSP.

2.2 Genetic Algorithms

Genetic Algorithms were introduced by John H. Holland in 1975. The idea behind them is to imitate the biological principle of evolution by selection, recombination and mutation of a set of initial candidate solutions- the population, which is either created randomly or heuristically. The quality of these solutions is evaluated in every iteration by fitness function. Better solutions are more likely to advance to the next iteration and are also more likely to be recombined with other solutions to produce off springs. Genetic algorithms have been the most popular heuristic for global optimization today. They use population of potential candidate solutions, just like other variations of evolutionary algorithm. Selection, crossover and mutation approaches revolve the genetic algorithms. The algorithm terminates as soon as the desirable descendent is found or when the computation limit exhausts [23].

2.3 Simulated Annealing Approach

Another local search method that been applied to the problem of generating tree decompositions of small width is Simulated Annealing which was introduced by Kirkpatrick

in 1983. Simulated annealing is inspired by an analogy between the physical annealing of solids (crystals) and combinatorial optimization problems. In the physical annealing process, solid is first melted and then cooled very slowly, spending long time at low temperature to obtain perfect lattice structure corresponding to minimum energy state. SA transfers this process to local search algorithms for combinatorial optimization problems. It does so by associating the set of solutions of the problem attached with the states of the physical system. SA is local search strategy which tries to avoid local minima by accepting less bad solutions with some probability [2].

2.4 Tabu Search Methods

Tabu Search is a local search technique that was proposed by Glover in 1989. The Tabu search methods rely on the systematic use of memory to guide the search process. A local search algorithm tries to improve an initial solution (generated randomly or heuristically) by looking at neighborhood solutions. A neighborhood solution's is defined by some kind systematic modification of the solution. One such modification might be the swapping of two solution elements. The best solution in the neighborhood is selected and its neighborhood is evaluated next. What is special about Tabu search is that it remembers a certain number of previous moves and adds them to a so-called Tabu-list. For example, if an element has been swapped in the previous five moves, the element must not be swapped again. This will prevent the algorithm from moving in circles in the solution space [27].

2.5 Swarm Intelligence

Swarm Intelligence is a modern artificial intelligence discipline that is concerned with the design of multi agent systems with applications, e.g. robotics and optimization. It is fundamentally different design paradigm from traditional approaches to optimization. It's principle is not to focused on a single sophisticated controller that governs the global behavior, but on many unsophisticated entities that cooperate to exhibit desired behavior. The chief source of inspiration to swarm intelligence is the collective behavior of social insects such as ants, termites, bees, and wasps. However, behavior of other animal societies such as flocks of birds or schools of fish has also been taken as inspirational source. In such animal societies, the behavior of an individual does not mean much. On the contrary, collective behavior of each individual produces extraordinarily great results. Models based on such phenomenon have been found to be utilized by many scientists and mathematicians. Swarm Intelligence has been growing as one of the promising sectors of research.

Insects such as ants, termites and wasps build sophisticated nests collectively. When studied collectively, these insects show amazing ability –ability to work as a team. Moreover, it is even more interesting fact that these are not guided by a single mastermind or a master plan as to how to proceed in building nests. And they build their nests to their best fit. Such amazing phenomena have driven scientists and researchers to involve in their social behavior study. Swarm Intelligence is the term for the field of such research that is inspired by such natural collective culture. Although relatively new, two promising areas where swarm intelligence has been in use are optimization and swarm robotics. These fields make use of information exchange in collective behavior of entities. These have been found quite successful. Some of the other areas where swarm intelligence has been prominent are routing and load balancing in telecommunication networks.

Swarm Intelligence is a most recent concept for CO problems. It is nature inspired algorithm for solving TSP like problems. It is based on the collective behavior of the computational agents. SI algorithms are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local or to a certain degree random, interactions between such agents. ACO is a part of swarm intelligence method. SI method is intensively studied in [1].

2.6 Recent Trend in ACO, BCO and TSP

Ant colony optimization was first proposed in 1992 by Marco Dorigo, for solving TSP like problems and in the decade since its introduction, a growing number of researchers have been involved in further developing it. Literature on the topic of ACO is currently largely restricted to journals and research papers, as this is still an emerging algorithm that has not yet weaved its way into mainstream texts. The following paragraphs briefly outline some of the better papers that are relevant to this research.

The authors in [10], introduces distributed algorithm that is applied to the TSP. In ACS, set of cooperating agents called ants cooperate to find good solutions to TSP. However, according to [30], the authors give an overview on the available ACO algorithms for the TSP. In 2005, M. Dorigo, C. Blum [9] discussed the theoretical results of ACO algorithms. They have analyzed the convergence results, connection between ACO algorithm and random gradient ascent within the model based search. They also discussed the relation between ACO and other approximate methods for optimization. Thereafter in 2011, Zar Chi Su Su Hlaing, May

As mentioned in [17], ACO is taken as one of the high performance computing methods for TSP and ACO for TSP has been improved by incorporating local optimization heuristic.
In this way day by day research in TSP problem raise more ooptimized algorithm. In 2012, Krishna H. Hingrajiya, Ravindra Kumar Gupta, Gajendra Singh Chandel[16] published a paper on "An Ant Colony Optimization Algorithm for Solving Travelling Salesman Problem". In this paper, they investigate ACO algorithms with respect to their runtime behavior for the traveling salesperson (TSP) problem.
In 2013, Km. Shweta [28] published a paper on " An Experimental Study of Ant System for Solving Travelling Salesman Problem". In this paper, he present an implementation of Ant System, the very first algorithm of Ant Colony Optimization in MATLAB to solve Travelling Salesman Problem.

Artificial bee colony (ABC) Algorithm is an optimization algorithm based on the intelligent behavior of honey bee foraging. This model was introduced by Dervis Karaboga [19] in 2005, and the performance of BCO is analyzed in 2007 [20].
In 2009, Li-Pei Wong and Chin Soon Chong [32] pubhished a paper "An Efficient Bee Colony Optimization Algorithm for Traveling Salesman Problem using Frequency-based Pruning". This introduced a bees perform waggle dance in order to communicate the information of food source to their hive mates. This foraging behaviour has been adapted in a Bee Colony Optimization (BCO) algorithm together with 2-opt local search to solve the Traveling Salesman Problem.
In may 2012, Ashita S. Bhagade, Parag. V. Puranik[5] published a paper "Artificial Bee Colony (ABC) Algorithm for Vehicle Routing Optimization Problem". This journal introduced the biological phenomenon when applied to the process of path planning problems for the vehicles, it is found to be excelling in solution quality as well as in computation time. Simulations have been used to evaluate the fitness of paths found by ABC Optimization.
In 2012 july, Nishant Pathak, Sudhanshu Prakash Tiwari [25] published a journal an "Travelling Salesman Problem Using Bee Colony With SPV". In this journal they present a solution for TSP problem using ABC with SPV rule. In this method he extend Artificial Bee Colony algorithm using SPV rule.
In 2012 july , Anshul Singh, Devesh Narayan[29] published a journal on "Augmentation of Travelling Salesman Problem using Bee Colony Optimization". In this journal BCO and k-opt local search, the two heuristic techniques for optimization, are combined together to acquire sophisticated results. Comparisons of the proposed method with nearest

neighborhood approach is performed and shown with presented system proved to be superior to the rest.

In 2013 march, Shailesh Pandey_and Sandeep Kumar[24] published a paper on “Enhanced Artificial Bee Colony Algorithm and It’s Application to Travelling Salesman Problem”. This paper presents ABC with different types of real coded crossover operator and its application to Travelling Salesman Problem (TSP). Each crossover operator is applied to two randomly selected parents from current swarm. Two off-springs generated from crossover and worst parent is replaced by best offspring, other parent remains same. ABC with real coded crossover operator applied to travelling salesman problem.

In march 2013,Ginnu George and Dr. Kumudha Raimond[14] published a paper on “Solving Travelling Salesman Problem Using Variants of ABC Algorithm”. This paper mainly explains about the performance of variants of Artificial Bee Colony (ABC) algorithms in solving the Travelling Salesman Problem (TSP).

Since TSP is NP-hard problem, so research on TSP is mainly focused on finding the optimal solution for large instances. Particularly, swarm intelligence methods are known to be the most successful till date. Swarm intelligence methods can solve small and medium sized TSP in polynomial time.

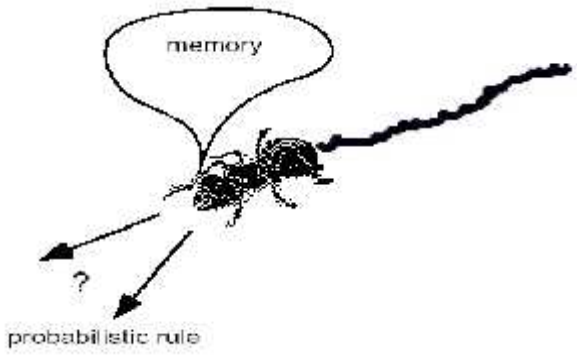
CHAPTER 3

ANT COLONY OPTIMIZATION ALGORITHM

In ACO algorithms ants are simple agents which, in the TSP case, construct tours by moving from city to city on the problem graph. The ants' solution construction is guided by (artificial) pheromone trails and an a priori available heuristic information[29].When applying ACO algorithm to the TSP, a pheromone strength Tij(t) is associated to each arc (i,j), where $Tij(t)$ is a numerical information which is modified during the run of the algorithm and t is the iteration counter.

Initially, each of the m ants is placed on a randomly chosen city and then iteratively applies at each city a state transition rule [30].

Fig.3.1: ACO State Transition Rule:



Next city is chosen between the not visited cities according to a *probabilistic.*

An ant constructs a tour as follows. At a city i, the ant chooses a still unvisited city j probabilistically, biased by the pheromone trail strength T$ij(t)$ on the arc between city i and city j and a locally available heuristic information, which is a function of the arc length. Ants probabilistically prefer cities which are close and are connected by arcs with a high pheromone trail strength. To construct a feasible solution each ant has a limited form of memory, called *tabu list*, in which the current partial tour is stored. The memory is used to determine at each construction step the set of cities which still has to be visited and to guarantee that a feasible solution is built. Additionally, it allows the ant to retrace its tour, once it is completed.

After all ants have constructed a tour, the pheromones are updated. This is typically done by first lowering the pheromone trail strengths by a constant factor and then the ants are allowed

to deposit pheromone on the arcs they have visited. The trail update is done in such a form that arcs contained in shorter tours and/or visited by many ants receive a higher amount of pheromone and are therefore chosen with a higher probability [30].

3.1 Tour construction

Initially, each ant is put on some randomly chosen city. At each construction step, ant k applies a probabilistic action choice rule. In particular, the probability with which ant k, currently at city *i*, chooses to go to city *j* at the *t* th iteration of the algorithm is:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \qquad \text{if } j \in \mathcal{N}_i^k$$

Where,

ηij= 1/*dij* is an a priori available heuristic value,

α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information,

and $\mathcal{N}_i^k$ is the feasible neighborhood of ant *k*, that is, the set of cities which ant *k* has not yet visited.

The role of the parameters α and β is the following.

If α = 0, the closest cities are more likely to be selected: with multiple starting points since ants are initially randomly distributed on the cities.

If β = 0, only pheromone amplification is at work: this method will lead to the rapid emergence of a *stagnation* situation with the corresponding generation of tours which, in general, are strongly suboptimal [11]. Search stagnation is defined in [12] as the situation where all the ants follow the same path and construct the same solution. Hence, a tradeoff between the influence of the heuristic information and the pheromone trails exists.

3.2 Pheromone update

After all ants have constructed their tours, the pheromone trails are updated. This is done by first lowering the pheromone strength on *all* arcs by a constant factor and then allowing each

ant to add pheromone on the arcs it has visited:

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

where $0 < \rho < 1$ is the pheromone trail evaporation.

P: is a parameter used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to "forget" previously done bad decisions. If an arc is not chosen by the ants, its associated pheromone strength decreases exponentially.

$\Delta\tau_{ij}^{k}(t)$ is the amount of pheromone ant *k* puts on the arcs it has visited; it is defined as follows:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) & \text{if arc } (i,j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases}$$

Where $L^{k}(t)$ is the length of the *k*th ant's tour. By above Equation, the better the ant's tour is, the more pheromone is received by arcs belonging to the tour. In general, arcs which are used by many ants and which are contained in shorter tours will receive more pheromone and therefore are also more likely to be chosen in future iterations of the algorithm.

3.3 ACO Algorithm and Complexity

The general ACO algorithm is given as follows [6].

Algorithm 3.1 General ACO

1. Initialize parameters.

2. FOR t=1 to number of pre-specified Iteration

    FOR k = 1 to m

        Until k-ant not yet visited all cities

    Choosing next city with pij

    Calculate total path-length Lk after completion of the tour

3. Update pheromone

4. Return the best tour.

The complexity of general ACO algorithm is O(NC.m.n^2), if algorithm iterates for NC number of iterations. In fact, path construction step i.e. for building partial solution it takes O(m.n^2). And pheromone update step, takes maximum of O(m. n^2). So, approximately general ACO algorithm is O(NC.n3). General ACO has the ability of finding the optimal solutions but these are strongly suboptimal [6,8].

3.4 Flow chart of ACO for solving TSP

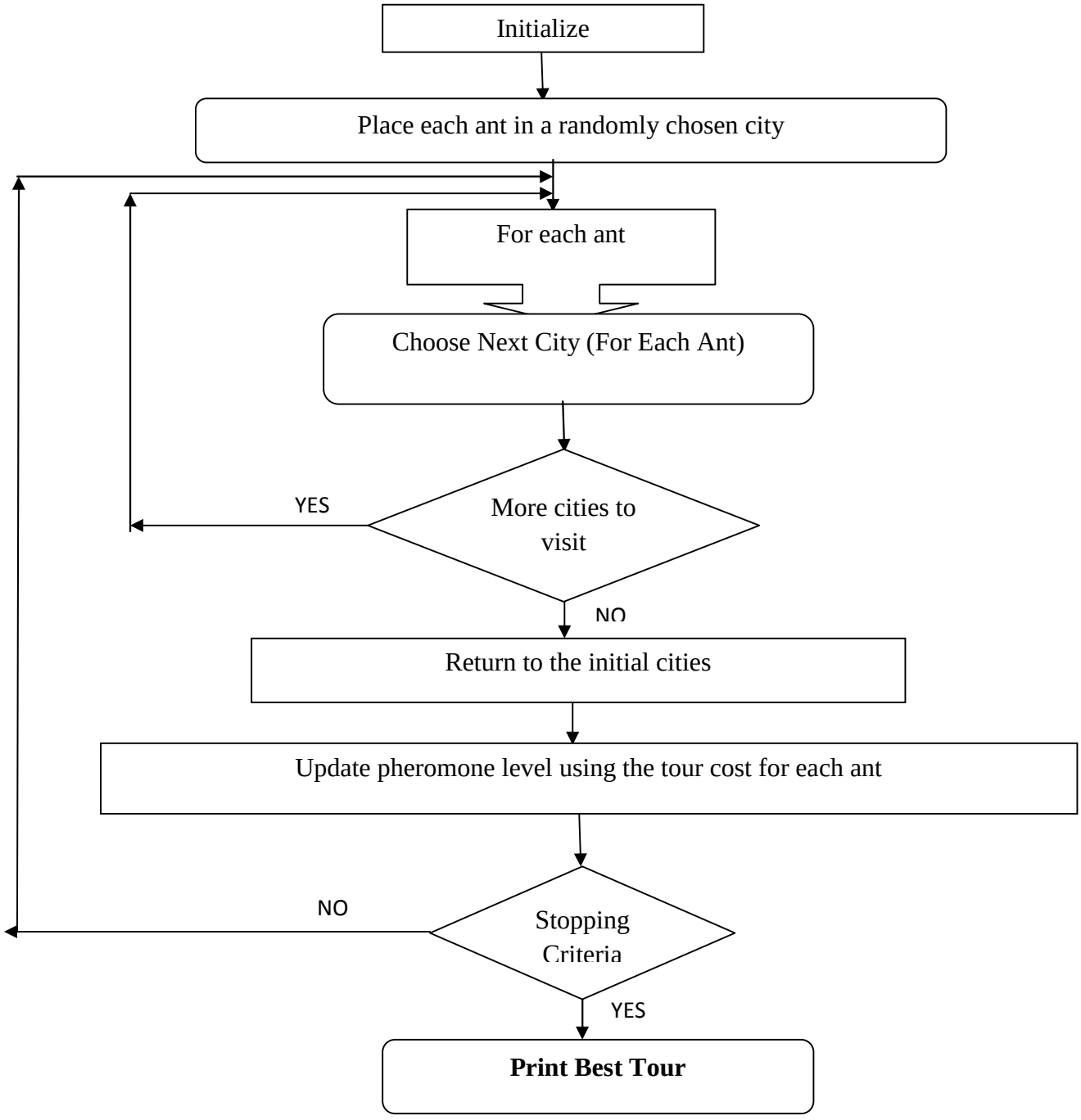


Fig.3.2: Flow chart of ACO for solving TSP [13]

CHAPTER 4

BEE COLONY OPTIMIZATION ALGORITHM

Artificial bee colony (ABC) Algorithm is an optimization algorithm based on the intelligent behavior of honey bee foraging. This model was introduced by Dervis Karaboga in 2005, and is based on inspecting the behaviors of real bees on finding nectar amounts and sharing the information of food sources to the other bees in the hive. These specialized bees try to maximize the nectar amount stored in the hive by performing efficient division of labour and self-organization [18]. The three agents in Artificial Bee Colony are:

- The Employed Bee
- The Onlooker Bee
- The Scout

4.1. General Description

The employed bees are associated with the specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly [29]. The onlooker bees and the scout bees are the unemployed bees. Initially, the scout bees discover the positions of all food sources, thereafter, the job of the employed bee starts. An artificial employed bee probabilistically obtains some modifications on the position in its memory to target a new food source and find the nectar amount or the fitness value of the new source. Later, the onlooker bee evaluates the information taken from all artificial employed bees and then chooses a final food source with the highest probability related to its nectar number. If the fitness value of new one is higher than that of the previous one, the bee forgets the old one and memorizes the new position [20]. This is called as greedy selection. Then the employed bee whose food source has been exhausted becomes a scout bee to search for the further food sources once again.

In ABC, the solutions represent the food sources and the nectar quantity of the food sources corresponds to the fitness of the associated solution. The number of the employed and the onlooker bees is same, and this number is equal to the number of food sources [9].

Employed bees whose solutions cannot be improved through a predetermined number of trials, specified by the user of the ABC algorithm and called limit‖, become scouts and their solutions are abandoned [29], [33].

4.2 Formal Definition:

In this algorithm, the employed bee produces a modification in the position (i.e. solution) in its memory and checks the nectar amount (fitness value) of that source (solution). The employed bee then evaluates this nectar information (fitness value) and then chooses the food source with the probability related to its fitness value [5].

Movement of onlookers:

- Probability of Selecting a nectar source:

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^{S} F(\theta_k)} \quad (1)$$

P_i : The probability of selecting the i^{th} employed bee

S : The number of employed bees

θ_i : The position of the i^{th} employed bee

$F(\theta_i)$: The fitness value

- **Calculation of the new position:** movement of onlookers

$$x_{ij}(t+1) = \theta_{ij}(t) + \phi(\theta_{ij}(t) - \theta_{kj}(t)) \quad (2)$$

  - x_i: The position of the onlooker bee.
  - t : The iteration number
  - θ_k : The randomly chosen employed bee.
  - j : The dimension of the solution
  - $\phi(\bullet)$: A series of random variable in the range [-1,1] .

Movement of scout bees

- The movement of the scout bees follows equation (3).

$$ij = j\min + r \cdot (j\max - j\min) \quad (3)$$

- r : A random number and $r \in [0,1]$

4.2.1 Waggle dance of honey bees

A branch of nature inspired algorithms which are called as swarm intelligence is focused on insect behaviour in order to develop some meta-heuristics which can mimic insect's problem solution abilities. Interaction between insects contributes to the collective intelligence of the social insect colonies. These communication systems between insects have been adapted to scientific problems for optimization. One of the examples of such interactive behaviour is the waggle dance of bees during the food procuring. By performing this dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates. So this is a successful mechanism which foragers can recruit other bees in their colony to productive locations to collect various resources. Bee colony can quickly and precisely adjust its searching pattern in time and space according to changing nectar sources. The information exchange among individual insects is the most important part of the collective knowledge. Communication among bees about the quality of food sources is being achieved in the dancing area by performing waggle dance [4].

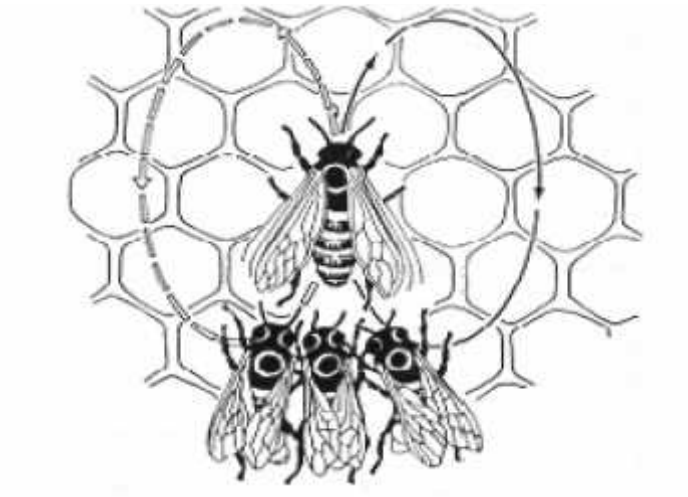


Figure 4.1: Waggle dance of honey bees [15]

The scout bees start from the hive in search of food source randomly. They keep on this exploration process until they are out of energy/tired and return back to the hive. When they return back to the hive, they share their experience and knowledge with the forager bees by

performing the mechanism called “waggle dance”. Waggling is a form of dance in circular direction in the shape of digit 8. It is repeated again and again by a bee. Its intensity and direction gives the idea of food source quality and food source location respectively to other bees. It is the means by which the bees communicate. It is used to convey the parameters like foods Source Quality, distance of food source from hive, Location of food source, to guide the path to the available forager bees [15]. These steps of the scout bees constitute the first step of the BCO process called the “Path Construction”.

Observations and studies on honey bee behaviors resulted in a new generation of Optimization algorithms.

4.3 Frame of the Bee colony optimization algorithm

This section presents the BCO algorithm [21].

Algorithm 4.1 Bee colony optimization

Send the scouts into the initial food source

REPEAT

Send the employed bees into the food source and determine their nectar amounts

Calculate the probability value of the sources with which they are preferred by the Onlooker bees

Send the onlooker bees into the food sources and determine their nectar amounts

Stop the exploitation process of the sources exhausted by the bees

Send the scouts into the search area for discovering new food sources, randomly

Memorize the best food source found so far

UNTIL (Requirements are met)

4.4 Methodology of BCO algorithm

The general scheme of the ABC algorithm is as follows: Bee Initialization Phase Set the Loop Employed Bee Phase Onlooker Bee Phase Scout Bee Phase Memorize the best solution found so far Until the loop is terminated. The essential control parameters in the Artificial bee colony algorithm are, the number of food sources which is equal to the number of employed/onlooker bees (CS-Colony Size), the working to onlooker bee rate, the value of the limit (L) and the number of cycles or the number of iterations (MCN) that are required to terminate the program. The implementation of Bee Colony optimization for solving the TSP problem is explained with the help of flow-chart shown below:

4.4.1. Flowchart of BCO algorithm for solving the TSP

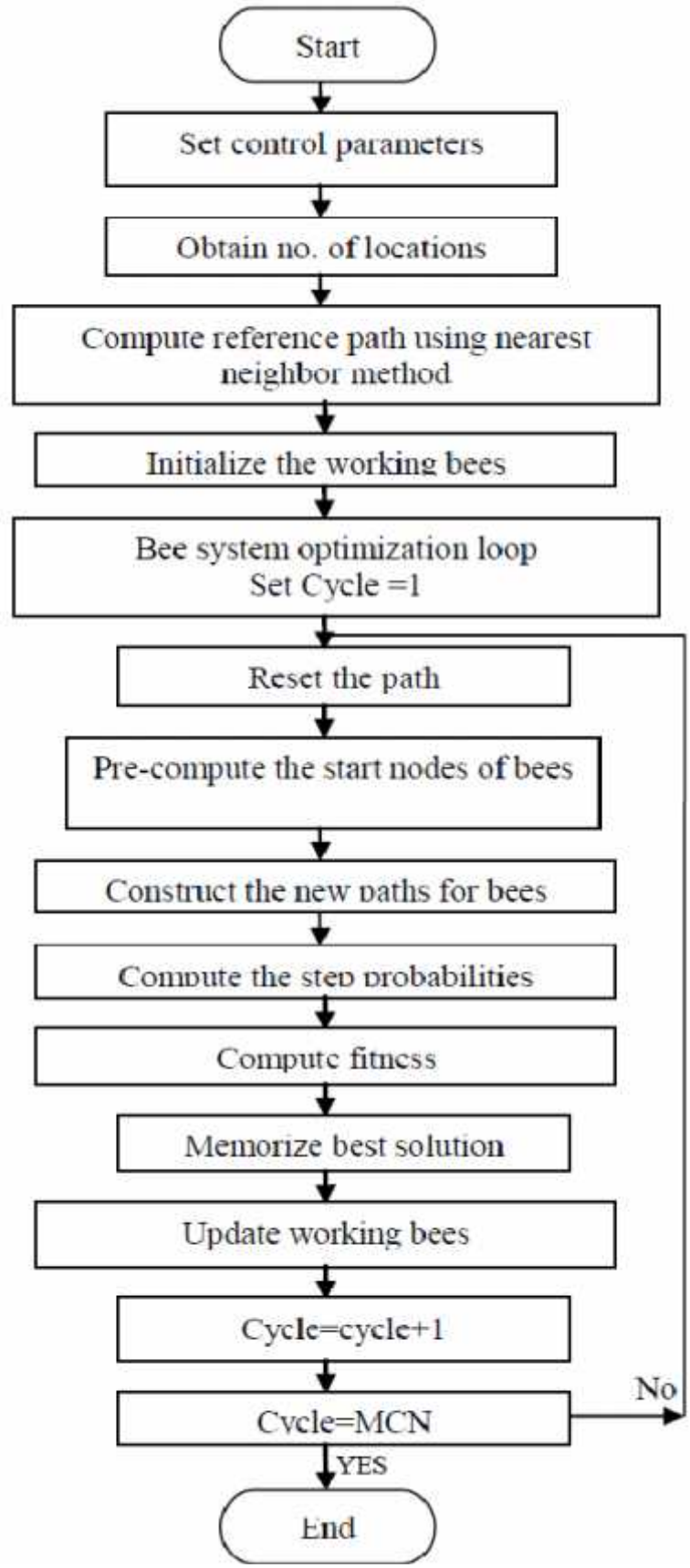


Fig.4.2: Flowchart of BCO algorithm for TSP [14]

4.5 Applying BCO for solving the TSP

In the initialization phase, the control parameters are set, such as colony size, iteration number (bee travel time), working to onlooker bee rate. In the next phase, the map is given as an input to the bee with the number of locations that are to be visited by the bee. Then a reference path is obtained by using nearest neighbor method. Further when the working bees are initialized, the bee optimization loop is set. Then the random node is assigned for the bee to start, then by computing the probabilities given by (1) the bees will work and draw the next node to obtain the path by using (2) and will memorize the best solution found so far using the greedy selection strategy. Finally the bees become scout bees and the number of working bees is updated, that is the employed bee which is exhausted becomes the scout bee again. The optimization loop is terminated when the numbers of iterations are completed and the best result is obtained. The scout bees then again start to search for the new path by (3) [5].

CHAPTER 5

IMPLEMENTATION AND ANALYSIS

ACO and BCO both the Algorithm has been implemented in java using NetBeans IDE 7.0.1 and executed on a PC with Windows7 professional-64 bit operating system, Intel(R)Core(tm)i5-2410M CPU(2.30 GHz), and 4GB of RAM. The complete graph with Euclidean distances for each pair of vertices is used. The inputs of the program are number of nodes and the costs of edges in the given graph. The classical TSP instances are obtained from TSPLIB library [31]. Sample Test Data and Sample test run are included in Appendix A and B respectively.

5.1 Computational Results

In this section computational results is present. With these results, the performance of the ACO & BCO algorithm is evaluated. Best solution is examined in terms of the number of iterations to produce the optimal solution and CPU time to produce such solution. The experiments show that the numbers of iterations are tremendously reduced by improved approach i.e. BCO. Different random and classical instances of TSP are tested using the general approach (ACO) and improved approach (BCO). Then computational results are compared with each other and then evaluation is carried out. The evaluation results clearly illustrate the main contributions of the thesis.

5.1.1 Parameter Setting

In all experiments of the following sections the numeric parameters, except when indicated differently, are set to the following tabulated values. These values are taken from [11, 8].

Parameters	Value
α	1
β	5
ρ	0.5
τ_0	0.8
Q	100
NC	500

Table 5.1: Parameter Values

5.1.2 Computational Results for ACO

In this dissertation, ACO is applied to instances of TSP with n≤ 200, and obtained results are presented in the table given below. It contains both random and classical instance of TSP. Instances named TSP10, TSP20, TSP30 are random and TSP40, TSP50, TSP60, TSP70, TSP80, TSP90, TSP100, TSP200 are classical instances of TSP. Data associated with each of these instances are listed in Appendix A. The number of Agents m is set to 50 for all the cases.

Case1: for instances of TSP

TSP Problem	Number of Nodes	Shortest path Length(km)			Elapsed Time(sec)		
TSP10	10	Run1	1378.48	AVG 1378.48	Run1	1529	AVG 1473
		Run2	1378.48		Run2	1378	
		Run3	1378.48		Run3	1513	
TSP20	20	Run1	75.89	75.89	Run1	3401	3432
		Run2	75.89		Run2	3385	
		Run3	75.89		Run3	3510	
TSP30	30	Run1	175.36	175.36	Run1	5912	5860
		Run2	175.36		Run2	5896	
		Run3	175.36		Run3	5772	
TSP40	40	Run1	6734.42	6758.44	Run1	9235	9307
		Run2	6827.16		Run2	9406	
		Run3	6713.74		Run3	9282	
TSP50	50	Run1	7854.38	7924.43	Run1	13572	13764
		Run2	7937.10		Run2	13572	
		Run3	7981.83		Run3	1419	
TSP60	60	Run1	433.04	432.02	Run1	18517	18735
		Run2	432.15		Run2	18782	
		Run3	430.88		Run3	18907	

TSP70	70	Run1	72172.71	72452.18	Run1	25334	25168
		Run2	72229.08		Run2	25054	
		Run3	72954.77		Run3	25116	
TSP80	80	Run1	1011.05	1019.58	Run1	32167	32552
		Run2	1022.88		Run2	32979	
		Run3	1024.91		Run3	32510	
TSP90	90	Run1	44110.84	44674.59	Run1	40498	40331
		Run2	45048.85		Run2	40263	
		Run3	44864.08		Run3	40233	
TSP100	100	Run1	25826.46	25655.02	Run1	49764	49842
		Run2	24898.75		Run2	49967	
		Run3	25239.85		Run3	499796	
TSP200	200	Run1	33841.44	34013.57	Run1	194657	196089
		Run2	33959.09		Run2	196430	
		Run3	34240.19		Run3	197180	

Table 5.2: Results for case1

5.1.3 Computational Results for BCO

BCO algorithm is applied exactly the same instances of TSP as used in the general ACO, and obtained results are presented in table below. Data associated with each of these instances are listed in Appendix A.

Case2: for instances of TSP

TSP Problem	Number of Nodes	Shortest path Length(km)			Elapsed Time(sec)		
TSP10	10	Run1	1375.74	AVG 1375.74	Run1	1113	AVG 1119
		Run2	1375.74		Run2	1143	
		Run3	1375.74		Run3	1101	
TSP20	20	Run1	63.24	63.24	Run1	3102	3329
		Run2	63.24		Run2	3270	
		Run3	63.24		Run3	3616	
TSP30	30	Run1	166.85	166.85	Run1	5844	5769
		Run2	166.85		Run2	5599	
		Run3	166.85		Run3	5866	
TSP40	40	Run1	6185.74	6185.74	Run1	7324	6925
		Run2	6185.74		Run2	6328	
		Run3	6185.74		Run3	6123	
TSP50	50	Run1	7074.46	7074.46	Run1	11743	11478
		Run2	7074.46		Run2	11304	
		Run3	7074.46		Run3	11388	
TSP60	60	Run1	363.49	353.32	Run1	15960	15663
		Run2	341.23		Run2	15288	
		Run3	355.25		Run3	15742	

TSP70	70	Run1	65609.87	65480.22	Run1	21194	21110
		Run2	65510.52		Run2	21900	
		Run3	65320.27		Run3	20238	
TSP80	80	Run1	882.27	870.7	Run1	31592	31273
		Run2	852.52		Run2	31544	
		Run3	877.60		Run3	30684	
TSP90	90	Run1	36429.34	36397.06	Run1	31142	27005
		Run2	36429.34		Run2	24876	
		Run3	36322.51		Run3	24998	
TSP100	100	Run1	20490.01	20490.01	Run1	47085	46704
		Run2	20490.01		Run2	45482	
		Run3	20490.01		Run3	47547	
TSP200	200	Run1	25625	25537.33	Run1	194720	196166
		Run2	26258		Run2	198860	
		Run3	24729		Run3	188920	

Table 5.3: Results for case2

5.2 Performance Evaluation of the Improved Algorithm

The computational results of the BCO are compared with those obtained in section 5.1.2. The following are the compared results of both the algorithms. Table 5.4, 5.5 present the comparisons of the execution time and the final path length from the two approaches. Table 5.6 present the comparison results.

Case1: for instances of TSP

TSP Problem	ACO(ms)	BCO (ms)
TSP10	1473	1119
TSP20	3432	3329
TSP30	5860	5769
TSP40	9307	6925
TSP50	13764	11478
TSP60	18735	15663
TSP70	25168	21110
TSP80	32552	31273
TSP90	40331	27005
TSP100	49842	46704
TSP200	196089	194166

Table 5.4: Comparison of execution time of getting the final path (ms)

Case2: for instances of TSP

TSP Problem	ACO	BCO
TSP10	1378.48	1375.74
TSP20	75.89	63.24
TSP30	175.36	166.85
TSP40	6758.44	6185.74
TSP50	7924.43	7074.46
TSP60	432.02	353.32
TSP70	72452.18	65480.22
TSP80	1019.58	870.7
TSP90	44674.59	36397.06
TSP100	25655.02	20490.01
TSP200	34013.57	25537.33

Table 5.5: Comparison of the length of the final path(km)

5.2.1 Results of Comparison

Comparison results are presented in table 5.6. Table 5.6 illustrates the comparison results for case1 and case2.

CALCULATION:-

For instance TSP10,

Path length improvement = {(1378.48-1375.74)/1378.48}×100 = 0.19%

& Execution time improvement = {(1473-1119)/1473}×100 = 24.03%

Similarly in all other instances.

TSP Problem	ACO		BCO		Improvement (%)	
	Path Length	Execution Time (ms)	Path Length	Execution Time (ms)	Path Length	Execution Time (ms)
TSP10	1378.48	1473	1375.74	1119	0.19	24.03
TSP20	75.89	3432	63.24	3329	16.66	3.00
TSP30	175.36	5860	166.85	5769	4.85	1.55
TSP40	6758.44	9307	6185.74	6925	8.47	25.59
TSP50	7924.43	13764	7074.46	11478	1072	16.60
TSP60	432.02	18735	353.32	15663	18.21	16.39
TSP70	72452.18	25168	65480.22	21110	9.62	16.12
TSP80	1019.58	32552	870.7	31273	14.60	3.92
TSP90	44674.59	40331	36397.06	27005	18.52	33.04
TSP100	25655.02	49842	20490.01	46704	20.132	6.29
TSP200	34013.57	196089	25537.33	194166	24.92	0.98

Table 5.6: Results of Comparison in case1 & case2

5.3 Discussions of the Evaluation Results

From the comparison results presented above in sections 5.2, it can be seen that the BCO algorithm can greatly enhance the performance rather than ACO for solving TSP. The evaluation results clearly show the main contributions of the thesis. Following sub-section show the verification of the BCO approach.

5.3.1 Verification of the Improved Algorithm

For instances of TSP when $n \leq 200$, BCO approach improves the processing time by 13.41% (on average) and optimal path length by 13.35% (on average) in comparison to the ACO approach. Figure 5.3, compares the optimized path length solution and Figure 5.4, compares the processing time of the two methods. Table 5.7 summarizes the BCO method improves the processing time and the optimal path length for solving the TSP. It can be seen that the BCO is superior to the ACO.

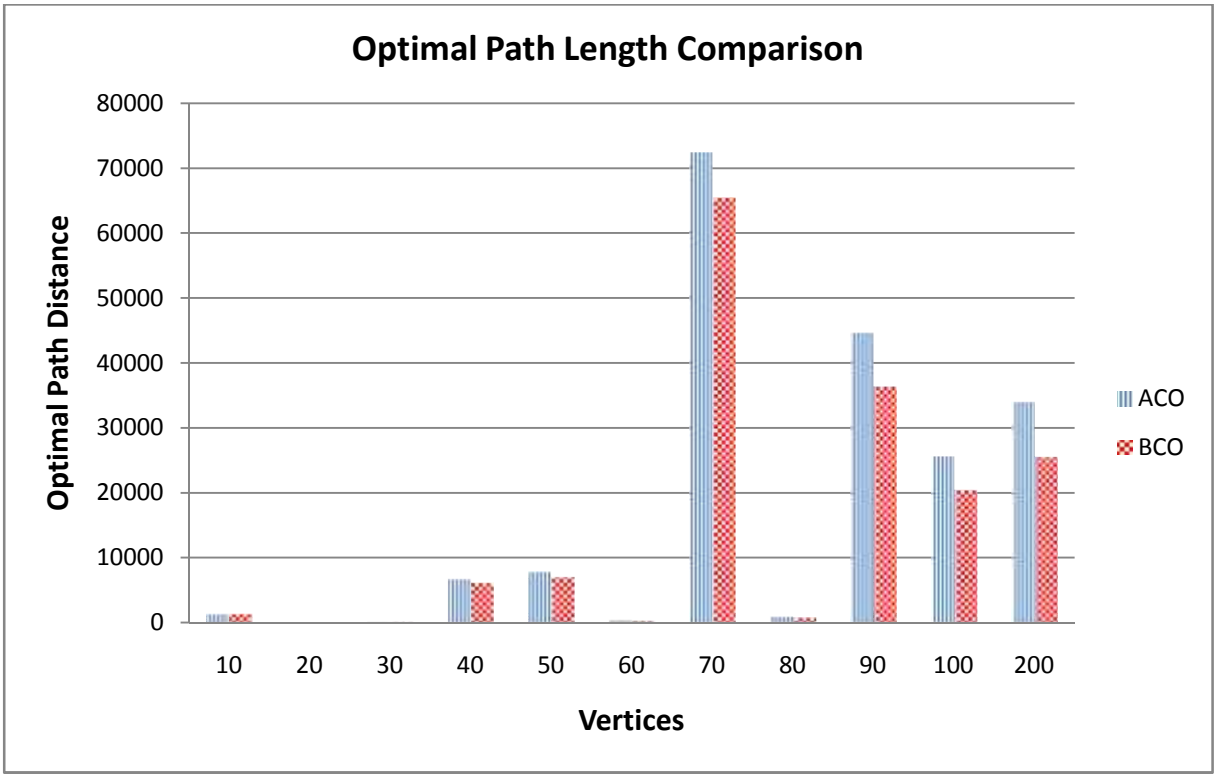


Figure 5.3: Optimal path solution comparison

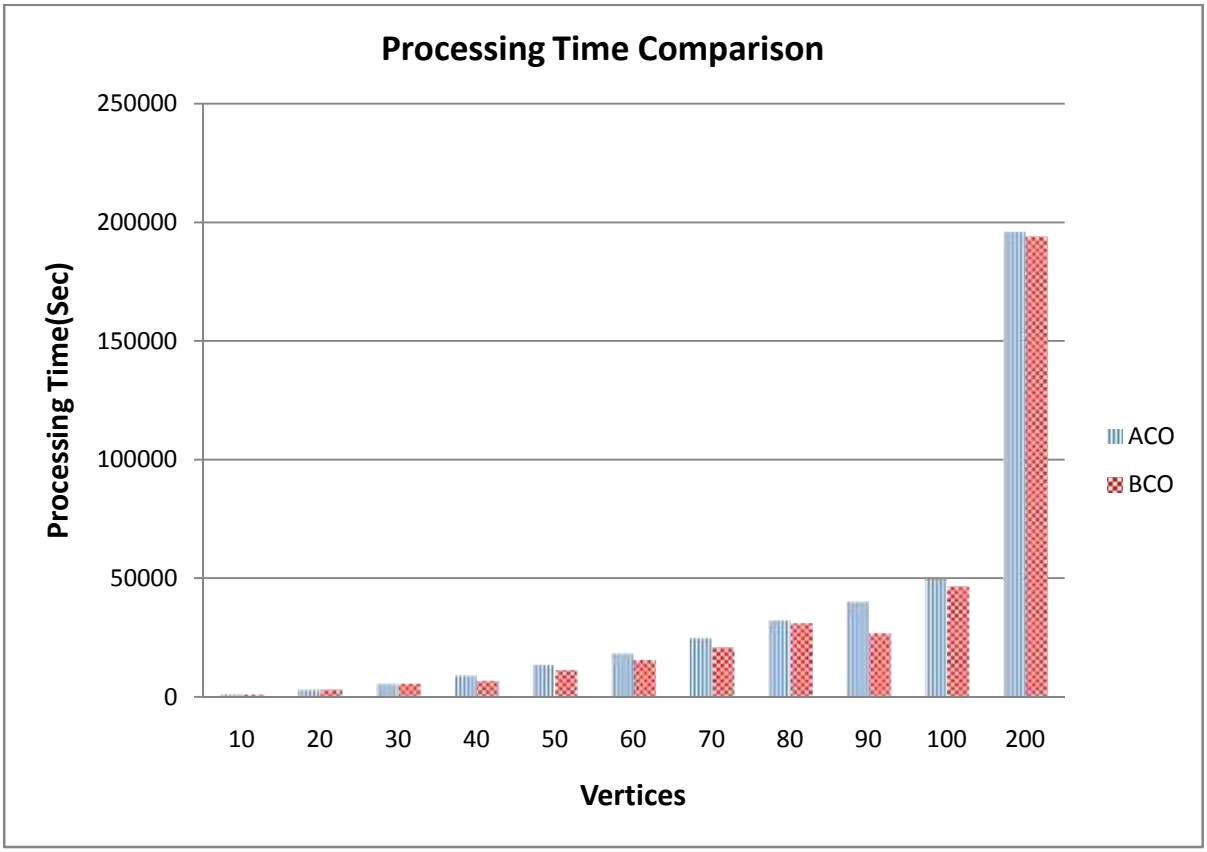


Figure 5.4: Processing time Comparison

TSP Problem	Improvement (%)	
	Optimal path length	Computation Time
TSP10	0.19	24.03
TSP20	16.66	3.00
TSP30	4.85	1.55
TSP40	8.47	25.59
TSP50	10.72	16.60
TSP60	18.21	16.39
TSP70	9.62	16.12
TSP80	14.60	3.92
TSP90	18.52	33.04
TSP100	20.13	6.29
TSP200	24.92	0.98

Table 5.7: Conclusion of the improvements

BCO Algorithm for solving the Travelling Salesman Problem is somehow similar to the ACO algorithm for solving the Travelling Salesman Problem. But the tour construction formula of both the algorithms is totally different. So the parameters of both the tour construction formula are also different. By using different tour construction formula and parameters, the result is also somehow different. Here, The BCO uses the concept of Non-Stigmergic (Direct) Communication to exchange information between bees. And the bee system is a standard example of organized team work, well coordinated interaction, coordination, labor division, simultaneous task performance, specialized individuals, and well-knit communication.

In this thesis, Artificial Bee Colony Optimization is presented by considering a new approach. The Artificial Bee Colony Optimization Algorithm can be used to solve several optimal problems. It is aimed to minimize the length of the tour and find the optimal path. To obtain performance comparisons with the other method, simulation framework is developed. The ACO and BCO approaches for solving the travelling salesman problem (TSP) is implemented and analyzed. The simulation outputs are shown in above tables and graphs.

The above graph in figure 5.3 clearly shows the optimal distance achieved by BCO Algorithm is smaller and error free as that of the ACO algorithm. The path length obtained by BCO is improved by 13.35% in comparison to ACO. From figure 5.4 it is clear that processing time of BCO algorithm is improved by 13.41 % on average in comparison to ACO algorithm. It is thus concluded that the Artificial Bee Colony Optimization can be efficiently used for solving the Travelling Salesman Problem in this thesis. Thus the Artificial Bee Colony optimization Algorithm is highly flexible and can be effectively used to find the shortest path by considering very few control parameters as compared with the other heuristic algorithms.

CHAPTER 6

CONCLUSION

The most common problems encountered by the general ACO algorithm are the premature deficiency and stagnation behavior. The main contribution of this thesis is a study of the avoidance of stagnation behavior and premature convergence. The computational results and performance comparison showed that the BCO algorithm reaches the better search performance over general ACO for solving the Travelling salesman Problem.

This chapter summarizes the thesis, and briefly discusses the limitations of the work. It also discusses the future directions on the research topic.

6.1 Summary

In this thesis, The ACO and BCO approaches for solving the travelling salesman problem (TSP) is implemented and analyzed. The BCO uses the concept of Non-Stigmergic (Direct) Communication to exchange information between bees. And the bee system is a standard example of organized team work, well coordinated interaction, coordination, labor division, simultaneous task performance, specialized individuals, and well-knit communication. The BCO uses a similarity among the way in which bees in nature look for a food, and the way in which optimization algorithms search for an optimum in combinatorial optimization problems. In this thesis, Artificial Bee Colony Optimization is presented by considering a new approach. The Artificial Bee Colony Optimization Algorithm can be used to solve several optimal problems. It is aimed to minimize the length of the tour and find the optimal path. To obtain performance comparisons with the other method, simulation framework is developed. The simulation outputs show that the optimal distance achieved by BCO Algorithm is smaller and error free as that of the ACO. Evaluation results have shown that BCO has improved by **13.35%** optimal path solutions on average and computational time is improved by 1**3.41%** on average. It is thus concluded that the Artificial Bee Colony Optimization can be efficiently used for solving the Travelling Salesman Problem in this thesis. Thus the Artificial Bee Colony optimization Algorithm is highly flexible and can be effectively used to find the shortest path by considering very few control parameters as compared with the other heuristic algorithms.

6.2 Research Limitations

The thesis successfully implements the ACO and BCO approach for solving the travelling salesman problem (TSP). However, there are still some research limitations of this thesis, such as, the search graph of the TSP instance is considered as a complete graph but this might not always be the case and BCO has slight complex modeling behavior than general ACO approach. For study, we considered the graphs with 10 to 200 nodes which is one of the limitations of this study and graphs with more number of nodes can be used to obtain more generalized result.

6.3 Future Work

The BCO approach can be applied to solve various combinatorial optimization problems in which there is an uncertainty of selection. Future work might incorporate the comparative study of Artificial Bee Colony Optimization algorithm with the other optimization algorithms.

References

[1] Afaq, H., Saini, S., *On the Solutions to the Travelling Salesman Problem using Nature Inspired Computing Techniques*, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 2, 2011.

[2] Alhamdy, S. A. S., Noudehi, A. N., Majdara, M., *Solving Traveling Salesman Problem (TSP) using Ants Colony (ACO) Algorithm and comparing with Tabu Search, Simulated Annealing and Genetic Algorithm*, Journal of Applied Sciences Research, 8(1): 434-440, 2012.

[3] Baldacci, R., Hadjiconstantinou, E., Mingozzi, A., *An Exact Algorithm for the Traveling Salesman Problem with Deliveries and Collections*, Department of Mathematics, University of Bologna, Via Sacchi 3, 47023 Cesena, Italy, 2011.

[4] Baykasoglu, A., Özbakır, L. and Tapkan, P., *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*, Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix Chan, T. S. and Tiwari, M. K., ISBN 978-3-902613-09-7, pp. 532, Itech Education and Publishing, Vienna, Austria, December 2007.

[5] Bhagade, A.S., Puranik, P. V., *Artificial Bee Colony (ABC) Algorithm for Vehicle Routing Optimization Problem*, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012.

[6] Brezina, I., Cickova, Z., *Solving the Travelling Salesman Problem Using the Ant Colony Optimization*, Management Information Systems, Vol.6 (2011), No.4, 2011.

[7] Buck, F., *Cooperative Problem Solving With a Distributed Agent System*, Swarm Intelligence, Dept. of Electrical and Computer Engineering, Utah State University, Logan, Utah 84322-4160, USA, 2005.

[8] Dhami K., *Comparative Study of General ACO and Improved ACO with Information Entropy for Solving TSP*, Master's Thesis, Tribhuvan University, Central Department of Computer Science and Information Technology, April 2013.

[9] Dorigo, M., Blum, C., *Ant colony optimization theory*: *A survey*, ELSEVIER, European Journal of Operational Research 344(2005), 243-278, 2005.

[10] Dorigo, M., Gambardella, L. M., Ant Colony System: *A Cooperative Learning Approach to the Travelling Salesman Problem*, Universite Libre de Bruxelles, TR/IRIDIA/1996-5, 1996.

[11] Dorigo, M., *Optimization, Learning, and Natural Algorithms*, PhD thesis, Department of Electronics, Milan, Italy, 1992.

[12] Dorigo, M., *The Ant System: Optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics–Part B, Vol.26, No.1, 1996, pp.1-13.

[13] Elshamli, A., Asmar, D., Elmasri, F., *Ant Colony Optimization*, 2010.

[14] George, G. and Raimond, K., *Solving Travelling Salesman Problem Using Variants of ABC Algorithm*, The International Journal of Computer Science & Applications (TIJCSA) Volume 2, No. 01, ISSN – 2278-1080, March 2013.

[15] Hashni, T., Amudha, Ms. T., *Relative Study of CGS with ACO and BCO Swarm Intelligence Techniques*, T Hashni et al ,Int.J.Computer Technology & Applications,Vol 3 (5), 1775-1781, Available online@www.ijcta.com, IJCTA | Sept-Oct 2012

[16] Hingrajiya, K. H., Gupta, R. K., Chandel, G. S., *An Ant Colony Optimization Algorithm for Solving Travelling Salesman Problem*, International Journal of Scientific and Research Publications, Volume 2, Issue 8, August 2012.

[17] Hlaing, Z. C. S. S., Khine, M. A., *An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem*, 2011 International Conference on Information Communication and Management IPCSIT vol.16 (2011) © (2011) IACSIT Press, Singapore, 2011.

[18] Karaboga, D., Akay, B., *Artificial bee colony (abc), harmony search and bees algorithms on Numerical optimization*, Erciyes University, the dept. Of computer engineering, 38039, melikgazi, kayseri, turkiye, 2010.

[19] Karaboga, D., *An Idea Based On Honey Bee Swarm For Numerical Optimization*, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005

[20] Karaboga, D., Basturk, B., *On the performance of artificial bee colony (ABC) algorithm*, Applied Soft Computing 8 (2008), pp.687-697, 2008.

[21] Larik, A. S., *Artificial Bee Colony Algorithm*, Journal of Applied Sciences Research, 8(1): 434-440, 2012.

[22] Montemanni, R., Barta, J., Gambardella, L. M., *An exact algorithm for the robust traveling salesman problem with interval data*, European Journal of Scientific Research, 2009.

[23] Moon, C., Kim, J., Choi, G., *An Efficient Genetic Algorithm for the Travelling Salesman Problem with Precedence Constraints*, ELSEVIER, European Journal of Operational Research 140(2002)606-617, 2002.

[24] Pandey, S. and Kumar, S., *Enhanced Artificial Bee Colony Algorithm and It's Application to Travelling Salesman Problem*, HCTL Open Int. J. of Technology Innovations and Research HCTL Open IJTIR, Volume 2, March 2013.

[25] Pathak, N., Tiwari, S. P., *Travelling Salesman Problem Using Bee Colony With SPV*, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-3, July 2012.

[26] Schrijver, A., *A Course in Combinatorial Optimization*, Department of Mathematics, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, Netherlands, 2013.

[27] Sha'ban, I. N. A. R. Z., *Tabu Search Method for Solving the Traveling salesman Problem*, Raf. J. of Comp. & Math's. , Vol. 5, No. 2, 2008.

[28] Shweta, K., *An Experimental Study of Ant System for Solving Travelling Salesman Problem*, International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 7, July 2013.

[29] Singh, A., Narayan, D., *Augmentation of Travelling Salesman Problem using Bee Colony Optimization*, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-1, Issue-2, July 2012.

[30] STÄUTZLEz, T. and DORIGO, M., *ACO Algorithms for the Traveling Salesman Problem*, 1999.

[31] TSPLIB Webpage (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp). Access Time: 20:48pm, 07-04-2014.

[32] Wong, L. P., Chong, C. S., *An Efficient Bee Colony Optimization Algorithm for Traveling Salesman Problem using Frequency-based Pruning*, 2009.

[33] Wong, L. P., HeaLow, M. Y., Chong, C. S., *A Bee Colony Optimization Algorithm for Travelling Salesman Problem*, Second Asia International Conference on Modellin & Simulation, pp. 818-823, 2008.

Appendix A: Sample Test Data

Problem Instance Name: TSP10, n= 10

Node number	X-coordinate	Y-coordinate
1	410.0	250.0
2	480.0	415.0
3	560.0	365.0
4	595.0	360.0
5	660.0	180.0
6	725.0	370.0
7	830.0	485.0
8	700.0	500.0
9	700.0	580.0
10	685.0	595.0

Problem Instance Name: TSP20, n= 20

Node number	X-coordinate	Y-coordinate
1	13.28	-16.39
2	11.51	-15.35
3	16.46	-3.01
4	12.39	-8.00
5	10.23	-9.18
6	9.31	-13.43
7	8.30	-13.15
8	6.18	-10.47
9	5.19	-4.02
10	6.41	-1.35
11	5.33	-0.13
12	6.08	1.13
13	6.29	2.37
14	12.22	-1.31
15	13.31	2.07
16	12.00	8.30
17	11.51	13.10
18	12.07	15.03
19	6.27	3.24
20	6.27	7.27

Problem Instance Name: TSP30, n= 30

Node number	X-coordinate	Y-coordinate
1	14.55	-23.31
2	28.06	-15.24
3	32.38	-16.54
4	31.38	-8.00
5	33.39	-7.35

6	**34.02**	**-6.51**
7	**34.05**	**-4.57**
8	**35.48**	**-5.45**
9	**35.43**	**-0.43**
10	**36.47**	**3.03**
11	**22.56**	**5.30**
12	**36.22**	**6.37**
13	**36.48**	**10.11**
14	**34.44**	**10.46**
15	**32.54**	**13.11**
16	**32.07**	**20.04**
17	**31.12**	**29.54**
18	**31.16**	**32.18**
19	**29.58**	**32.33**
20	**30.03**	**31.15**
21	**24.05**	**32.53**
22	**19.37**	**37.14**
23	**15.36**	**32.32**
24	**13.11**	**30.13**
25	**13.38**	**25.21**
26	**15.20**	**38.53**
27	**9.00**	**38.50**
28	**11.36**	**43.09**
29	**18.06**	**-15.57**
30	**14.40**	**-17.26**

Problem Instance Name: TSP40, n= 40

Node number	X-coordinate	Y-coordinate
1	**565.0**	**575.0**
2	**25.0**	**185.0**
3	**345.0**	**750.0**
4	**945.0**	**685.0**
5	**845.0**	**655.0**
6	**880.0**	**660.0**
7	**25.0**	**230.0**
8	**525.0**	**1000.0**
9	**580.0**	**1175.0**
10	**650.0**	**1130.0**
11	**1605.0**	**620.0**
12	**1220.0**	**580.0**
13	**1465.0**	**200.0**
14	**1530.0**	**5.0**
15	**845.0**	**680.0**
16	**725.0**	**370.0**
17	**145.0**	**665.0**
18	**415.0**	**635.0**
19	**510.0**	**875.0**
20	**560.0**	**365.0**
21	**300.0**	**465.0**

22	**520.0**	**585.0**
23	**480.0**	**415.0**
24	**835.0**	**625.0**
25	**975.0**	**580.0**
26	**1215.0**	**245.0**
27	**1320.0**	**315.0**
28	**1250.0**	**400.0**
29	**660.0**	**180.0**
30	**410.0**	**250.0**
31	**420.0**	**555.0**
32	**575.0**	**665.0**
33	**1150.0**	**1160.0**
34	**700.0**	**580.0**
35	**685.0**	**595.0**
36	**685.0**	**610.0**
37	**770.0**	**610.0**
38	**795.0**	**645.0**
39	**189.0**	**756.0**
40	**205.0**	**463.0**

Problem Instance Name: TSP50, n= 50

Node number	X-coordinate	Y-coordinate
1	**565.0**	**575.0**
2	**25.0**	**185.0**
3	**345.0**	**750.0**
4	**945.0**	**685.0**
5	**845.0**	**655.0**
6	**880.0**	**660.0**
7	**25.0**	**230.0**
8	**525.0**	**1000.0**
9	**580.0**	**1175.0**
10	**650.0**	**1130.0**
11	**1605.0**	**620.0**
12	**1220.0**	**580.0**
13	**1465.0**	**200.0**
14	**1530.0**	**5.0**
15	**845.0**	**680.0**
16	**725.0**	**370.0**
17	**145.0**	**665.0**
18	**415.0**	**635.0**
19	**510.0**	**875.0**
20	**560.0**	**365.0**
21	**300.0**	**465.0**
22	**520.0**	**585.0**
23	**480.0**	**415.0**
24	**835.0**	**625.0**
25	**975.0**	**580.0**
26	**1215.0**	**245.0**
27	**1320.0**	**315.0**

28	**1250.0**	**400.0**
29	**660.0**	**180.0**
30	**410.0**	**250.0**
31	**420.0**	**555.0**
32	**575.0**	**665.0**
33	**1150.0**	**1160.0**
34	**700.0**	**580.0**
35	**685.0**	**595.0**
36	**685.0**	**610.0**
37	**770.0**	**610.0**
38	**795.0**	**645.0**
39	**720.0**	**635.0**
40	**760.0**	**650.0**
41	**475.0**	**960.0**
42	**95.0**	**260.0**
43	**875.0**	**920.0**
44	**700.0**	**500.0**
45	**555.0**	**815.0**
46	**830.0**	**485.0**
47	**1170.0**	**65.0**
48	**830.0**	**610.0**
49	**605.0**	**625.0**
50	**595.0**	**360.0**

Problem Instance Name: TSP60, n= 60

Node number	X-coordinate	Y-coordinate
1	**-4.18**	**15.18**
2	**0.04**	**18.16**
3	**-5.54**	**22.25**
4	**0.30**	**25.12**
5	**-3.23**	**29.22**
6	**-1.57**	**30.04**
7	**0.19**	**32.25**
8	**-1.17**	**36.49**
9	**2.01**	**45.20**
10	**-4.03**	**39.40**
11	**-6.10**	**39.11**
12	**-6.48**	**39.17**
13	**-8.48**	**13.14**
14	**-12.44**	**15.47**
15	**-11.40**	**27.28**
16	**-12.49**	**28.13**
17	**-15.25**	**28.17**
18	**-20.09**	**28.36**
19	**-17.50**	**31.03**
20	**-15.47**	**35.00**
21	**-19.49**	**34.52**
22	**-25.58**	**32.35**
23	**-15.57**	**-5.42**

24	**-37.15 -**	**12.30**
25	**-22.59**	**14.31**
26	**-22.34**	**17.06**
27	**-26.38**	**15.10**
28	**-24.45**	**25.55**
29	**-25.45**	**28.10**
30	**-26.15**	**28.00**
31	**-29.12**	**26.07**
32	**-29.55**	**30.56**
33	**-33.00**	**27.55**
34	**-33.58**	**25.40**
35	**38.24**	**20.42**
36	**39.57**	**26.15**
37	**40.56**	**25.32**
38	**36.26**	**23.12**
39	**33.48**	**10.54**
40	**37.56**	**12.19**
41	**38.42**	**13.11**
42	**37.52**	**20.44**
43	**41.23**	**9.10**
44	**41.17**	**13.05**
45	**36.08**	**-5.21**
46	**38.47**	**15.13**
47	**38.15**	**15.35**
48	**37.51**	**15.17**
49	**35.49**	**14.32**
50	**39.36**	**19.56**
51	**38.09**	**24.36**
52	**36.09**	**23.00**
53	**40.44**	**13.57**
54	**40.33**	**14.15**
55	**40.37**	**14.23**
56	**37.57**	**22.56**
57	**40.0**	**21.0**
58	**24.0**	**25.0**
59	**12.0**	**63.0**
60	**45.0**	**51.0**

Problem Instance Name: TSP70, n= 70

Node number	X-coordinate	Y-coordinate
1	**9860**	**1415**
2	**9396**	**14616**
3	**11252**	**14848**
4	**11020**	**13456**
5	**9512**	**15776**
6	**10788**	**13804**
7	**10208**	**14384**
8	**11600**	**13456**
9	**11252**	**14036**

10	**10672**	**15080**
11	**11136**	**14152**
12	**9860**	**13108**
13	**10092**	**14964**
14	**9512**	**13340**
15	**10556**	**13688**
16	**9628**	**14036**
17	**10904**	**13108**
18	**11368**	**12644**
19	**11252**	**13340**
20	**10672**	**13340**
21	**11020**	**13108**
22	**11020**	**13340**
23	**11136**	**13572**
24	**11020**	**13688**
25	**8468**	**11136**
26	**8932**	**12064**
27	**9512**	**12412**
28	**7772**	**11020**
29	**8352**	**10672**
30	**9164**	**12876**
31	**9744**	**12528**
32	**8352**	**10324**
33	**8236**	**11020**
34	**8468**	**12876**
35	**8700**	**14036**
36	**8932**	**13688**
37	**9048**	**13804**
38	**8468**	**12296**
39	**8352**	**12644**
40	**8236**	**13572**
41	**9164**	**13340**
42	**8004**	**12760**
43	**8584**	**13108**
44	**7772**	**14732**
45	**7540**	**15080**
46	**7424**	**17516**
47	**8352**	**17052**
48	**7540**	**16820**
49	**7888**	**17168**
50	**9744**	**15196**
51	**9164**	**14964**
52	**9744**	**16240**
53	**7888**	**16936**
54	**8236**	**15428**
55	**9512**	**17400**
56	**9164**	**16008**
57	**8700**	**15312**
58	**11716**	**16008**
59	**12992**	**14964**

60	**12412**	**14964**
61	**12296**	**15312**
62	**12528**	**15196**
63	**15312**	**6612**
64	**11716**	**16124**
65	**11600**	**19720**
66	**10324**	**17516**
67	**12412**	**13340**
68	**12876**	**12180**
69	**13688**	**10904**
70	**13688**	**11716**

Problem Instance Name: TSP80, n= 80

Node number	X-coordinate	Y-coordinate
1	**288**	**149**
2	**288**	**129**
3	**270**	**133**
4	**256**	**141**
5	**256**	**157**
6	**246**	**157**
7	**236**	**169**
8	**228**	**169**
9	**228**	**161**
10	**220**	**169**
11	**212**	**169**
12	**204**	**169**
13	**196**	**169**
14	**188**	**169**
15	**196**	**161**
16	**188**	**145**
17	**172**	**145**
18	**164**	**145**
19	**156**	**145**
20	**148**	**145**
21	**140**	**145**
22	**148**	**169**
23	**164**	**169**
24	**172**	**169**
25	**156**	**169**
26	**140**	**169**
27	**132**	**169**
28	**124**	**169**
29	**116**	**161**
30	**104**	**153**
31	**104**	**161**
32	**104**	**169**
33	**90**	**165**
34	**80**	**157**
35	**64**	**157**

36	**64**	**165**
37	**56**	**169**
38	**56**	**161**
39	**56**	**153**
40	**56**	**145**
41	**56**	**137**
42	**56**	**129**
43	**56**	**121**
44	**40**	**121**
45	**40**	**129**
46	**40**	**137**
47	**40**	**145**
48	**40**	**153**
49	**40**	**161**
50	**40**	**169**
51	**32**	**169**
52	**32**	**161**
53	**32**	**153**
54	**32**	**145**
55	**32**	**137**
56	**32**	**129**
57	**32**	**121**
58	**32**	**113**
59	**40**	**113**
60	**56**	**113**
61	**56**	**105**
62	**48**	**99**
63	**40**	**99**
64	**32**	**97**
65	**32**	**89**
66	**24**	**89**
67	**16**	**97**
68	**16**	**109**
69	**8**	**109**
70	**8**	**97**
71	**8**	**89**
72	**8**	**81**
73	**8**	**73**
74	**8**	**65**
75	**8**	**57**
76	**16**	**57**
77	**8**	**49**
78	**8**	**41**
79	**24**	**45**
80	**32**	**41**

Problem Instance Name: TSP90, n= 90

Node number	X-coordinate	Y-coordinate
1	8375	4700
2	8775	4700
3	8375	4900
4	8175	4900
5	8775	4900
6	8575	4900
7	8775	5400
8	8375	5450
9	8775	5600
10	8575	5600
11	8375	5650
12	8175	5650
13	8375	6200
14	8775	6200
15	8375	6400
16	8175	6400
17	8775	6400
18	8575	6400
19	8375	7000
20	8775	7000
21	8375	7200
22	8175	7200
23	8775	7200
24	8575	7200
25	8375	7800
26	8775	7800
27	8375	8000
28	8175	8000
29	8775	8000
30	8575	8000
31	8375	8700
32	8775	8700
33	8375	8900
34	8175	8900
35	8775	8900
36	8575	8900
37	8375	9600
38	8775	9600
39	8375	9800
40	8175	9800
41	8775	9800
42	8575	9800
43	8375	10500
44	8775	10450
45	8375	10700
46	8175	10700
47	8775	10650
48	8575	10650
49	8375	11300
50	8775	11300

51	**8375**	**11500**
52	**8175**	**11500**
53	**8775**	**11500**
54	**8575**	**11500**
55	**15825**	**11500**
56	**15825**	**10700**
57	**15825**	**9800**
58	**15825**	**8900**
59	**15825**	**8000**
60	**15825**	**7200**
61	**15825**	**6400**
62	**15825**	**5650**
63	**15825**	**4900**
64	**16025**	**4700**
65	**16425**	**4700**
66	**16025**	**4900**
67	**16225**	**4900**
68	**16425**	**4900**
69	**16425**	**5400**
70	**16025**	**5450**
71	**16225**	**5600**
72	**16425**	**5600**
73	**16025**	**5650**
74	**16025**	**6200**
75	**16425**	**6200**
76	**16025**	**6400**
77	**16225**	**6400**
78	**16425**	**6400**
79	**16025**	**7000**
80	**16425**	**7000**
81	**16025**	**7200**
82	**16225**	**7200**
83	**16425**	**7200**
84	**16025**	**7800**
85	**16425**	**7800**
86	**16025**	**8000**
87	**16225**	**8000**
88	**16425**	**8000**
89	**16025**	**8700**
90	**16425**	**8700**

Problem Instance Name: TSP100, n= 100

Node number	X-coordinate	Y-coordinate
1	**1380**	**939**
2	**2848**	**96**
3	**3510**	**1671**
4	**457**	**334**
5	**3888**	**666**
6	**984**	**965**
7	**2721**	**1482**
8	**1286**	**525**
9	**2716**	**1432**

10	**738**	**1325**
11	**1251**	**1832**
12	**2728**	**1698**
13	**3815**	**169**
14	**3683**	**1533**
15	**1247**	**1945**
16	**123**	**862**
17	**1234**	**1946**
18	**252**	**1240**
19	**611**	**673**
20	**2576**	**1676**
21	**928**	**1700**
22	**53**	**857**
23	**1807**	**1711**
24	**274**	**1420**
25	**2574**	**946**
26	**178**	**24**
27	**2678**	**1825**
28	**1795**	**962**
29	**3384**	**1498**
30	**3520**	**1079**
31	**1256**	**61**
32	**1424**	**1728**
33	**3913**	**192**
34	**3085**	**1528**
35	**2573**	**1969**
36	**463**	**1670**
37	**3875**	**598**
38	**298**	**1513**
39	**3479**	**821**
40	**2542**	**236**
41	**3955**	**1743**
42	**1323**	**280**
43	**3447**	**1830**
44	**2936**	**337**
45	**1621**	**1830**
46	**3373**	**1646**
47	**1393**	**1368**
48	**3874**	**1318**
49	**938**	**955**
50	**3022**	**474**
51	**2482**	**1183**
52	**3854**	**923**
53	**376**	**825**
54	**2519**	**135**
55	**2945**	**1622**
56	**953**	**268**
57	**2628**	**1479**
58	**2097**	**981**
59	**890**	**1846**

60	**2139**	**1806**
61	**2421**	**1007**
62	**2290**	**1810**
63	**1115**	**1052**
64	**2588**	**302**
65	**327**	**265**
66	**241**	**341**
67	**1917**	**687**
68	**2991**	**792**
69	**2573**	**599**
70	**19**	**674**
71	**3911**	**1673**
72	**872**	**1559**
73	**2863**	**558**
74	**929**	**1766**
75	**839**	**620**
76	**3893**	**102**
77	**2178**	**1619**
78	**3822**	**899**
79	**378**	**1048**
80	**1178**	**100**
81	**2599**	**901**
82	**3416**	**143**
83	**2961**	**1605**
84	**611**	**1384**
85	**3113**	**885**
86	**2597**	**1830**
87	**2586**	**1286**
88	**161**	**906**
89	**1429**	**134**
90	**742**	**1025**
91	**1625**	**1651**
92	**1187**	**706**
93	**1787**	**1009**
94	**22**	**987**
95	**3640**	**43**
96	**3756**	**882**
97	**776**	**392**
98	**1724**	**1642**
99	**198**	**1810**
100	**3950**	**1558**

Appendix B: Sample Test Run

Test Run1: Result for TSP10 with n=10:

run:

Agent returned with new best distance of: 1502.4809533505818

Agent returned with new best distance of: 1451.130196364888

Agent returned with new best distance of: 1435.9178352630229

Agent returned with new best distance of: 1408.8000220502454

Agent returned with new best distance of: 1378.4822503133914

Waiting for 3 agents to finish their random walk!

Found best so far: 1378.4822503133914

[8, 9, 7, 3, 2, 1, 0, 4, 5, 6]

Took: 1341 ms!

Result was: 1378.4822503133914

BUILD SUCCESSFUL (total time: 2 seconds)

Test Run2: Result for TSP20, with n=20:

run:

Agent returned with new best distance of: 98.99766009720597

Agent returned with new best distance of: 94.70996124601996

Agent returned with new best distance of: 94.48523145298736

Agent returned with new best distance of: 91.4088120206939

Agent returned with new best distance of: 84.99916331316186

Agent returned with new best distance of: 84.1136201561342

Agent returned with new best distance of: 83.57894532276973

Agent returned with new best distance of: 81.15203497799894

Agent returned with new best distance of: 80.53943559328407

Agent returned with new best distance of: 79.26501683399569

Agent returned with new best distance of: 75.89029728138411

Agent returned with new best distance of: 75.8902972813841

Waiting for 3 agents to finish their random walk!

Found best so far: 75.8902972813841

[18, 12, 11, 10, 9, 8, 7, 6, 5, 1, 0, 4, 3, 2, 13, 14, 15, 16, 17, 19]

Took: 3495 ms!

Result was: 75.8902972813841

BUILD SUCCESSFUL (total time: 5 seconds)

Test Run3: Result for TSP50, with n=50:

run:

Agent returned with new best distance of: 9415.532456547142

Agent returned with new best distance of: 9112.971938529236

Agent returned with new best distance of: 8764.830455065669

Agent returned with new best distance of: 8678.514369794624

Agent returned with new best distance of: 8550.29877038806

Agent returned with new best distance of: 8384.431886401975

Agent returned with new best distance of: 8378.999905946697

Agent returned with new best distance of: 8106.0921308132

Agent returned with new best distance of: 8037.2467958138695

Agent returned with new best distance of: 7968.199017021965

Agent returned with new best distance of: 7927.871565545271

Agent returned with new best distance of: 7836.9997941294205

Waiting for 3 agents to finish their random walk!

Found best so far: 7836.9997941294205

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

Took: 13650 ms!

Result was: 7836.9997941294205

BUILD SUCCESSFUL (total time: 15 seconds)

Test Run4: Result for TSP100, with n=100:

run:

Agent returned with new best distance of: 33738.00090196493

Agent returned with new best distance of: 31321.835436725112

Agent returned with new best distance of: 30047.63831580783

Agent returned with new best distance of: 29842.45410018179

Agent returned with new best distance of: 28992.04310657391

Agent returned with new best distance of: 28703.86807539831

Agent returned with new best distance of: 28703.268635596607

Agent returned with new best distance of: 27059.83714663523

Agent returned with new best distance of: 26271.07446913352

Agent returned with new best distance of: 26071.069658684697

Agent returned with new best distance of: 24964.407660797002

Waiting for 3 agents to finish their random walk!

Found best so far: 24964.407660797002

[32, 75, 12, 94, 81, 49, 67, 84, 72, 43, 1, 63, 39, 53, 68, 80, 24, 60, 50, 86, 56, 6, 8, 82, 54, 11, 19, 85, 26, 34, 61, 59, 76, 22, 97, 90, 31, 44, 10, 14, 16, 58, 20, 73, 71, 9, 83, 35, 23, 37, 98, 17, 78, 52, 15, 87, 21, 93, 69, 65, 64, 3, 25, 96, 18, 48, 5, 62, 0, 91, 7, 88, 41, 30, 79, 55, 74, 89, 46, 92, 27, 57, 66, 33, 28, 45, 42, 2, 13, 99, 70, 40, 47, 29, 38, 77, 51, 95, 4, 36]

Took: 48968 ms!

Result was: 24964.407660797002

BUILD SUCCESSFUL (total time: 49 seconds)

Test Run1: Result for TSP10 with n=10:

run:

Number of cities = 10

Agent return with the best distance of: 3148.3454628669233

Agent return with the best distance of: 2883.260050712893

Agent return with the best distance of: 3420.390245231565

Agent return with the best distance of: 3244.5506167968433

Agent return with the best distance of: 2031.8676164524786

Agent return with the best distance of: 1792.5146452656227

Agent return with the best distance of: 1505.1616435251508

Agent return with the best distance of: 1375.7462810582965

[10, 9, 8, 7, 6, 4, 3, 2, 1, 5]

Best Solution : 1375.7462810582965

Time Required for Bee Colony Algorithm : 1197.0

BUILD SUCCESSFUL (total time: 10 seconds)

Test Run2: Result for TSP20 with n=20:

run:

Number of cities = 20

Agent return with the best distance of: 86.49753079312575

Agent return with the best distance of: 82.08651236156994

Agent return with the best distance of: 81.78479425339353

Agent return with the best distance of: 75.19271350330582

Agent return with the best distance of: 71.38795938773794

Agent return with the best distance of: 67.5771225153157

Agent return with the best distance of: 63.772677578441645

Agent return with the best distance of: 63.249377276373224

[19, 13, 12, 11, 10, 9, 8, 7, 6, 2, 1, 5, 4, 3, 14, 15, 16, 17, 18, 20]

Best Solution : 63.249377276373224

Time Required for Bee Colony Algorithm : 3644.0

BUILD SUCCESSFUL (total time: 8 seconds)

Test Run3: Result for TSP100 with n=100:

run:

Number of cities = 100

Agent return with the best distance of: 22264.378969951613

Agent return with the best distance of: 21725.04659357661

Agent return with the best distance of: 21707.6609354778

Agent return with the best distance of: 21699.593601902565

Agent return with the best distance of: 21366.145794445718

Agent return with the best distance of: 21049.28067966766

Agent return with the best distance of: 20991.624224159907

Agent return with the best distance of: 20857.682274596234

Agent return with the best distance of: 20792.507982646646

Agent return with the best distance of: 20612.163509024205

Agent return with the best distance of: 20490.014047501245

[44, 50, 73, 68, 85, 39, 30, 96, 78, 52, 5, 37, 33, 76, 13, 95, 82, 2, 54, 40, 64, 69, 81, 25, 61, 51, 87, 9, 7, 57, 20, 12, 27, 86, 35, 62, 60, 77, 23, 98, 91, 45, 32, 11, 15, 17, 74, 21, 59, 72, 10, 84, 36, 38, 24, 18, 79, 53, 88, 16, 22, 94, 70, 66, 65, 4, 97, 56, 80, 31, 89, 42, 8, 92, 1, 63, 6, 49, 90, 19, 75, 26, 99, 47, 93, 28, 67, 58, 55, 83, 34, 29, 46, 3, 43, 14, 71, 41, 100, 48]

Best Solution : 20490.014047501245

Time Required for Bee Colony Algorithm : 46091.5

BUILD SUCCESSFUL (total time: 27 seconds)