TRAVELLING SALESMAN PROBLEM: GENETIC ALGORITHM AND ANT COLONY OPTIMIZATION

MSc. PROJECT

BALEMLAY YIZENGAW

June 2019

HARAMAYA UNIVERSITY, HARAMAYA

Travelling Salesman Problem: Genetic Algorithm and Ant Colony Optimization

A Project submitted to the Department of Mathematics,
Postgraduate Program Directorate
HARAMAYA UNIVERSITY

In Partial Fulfillment of the Requirements for the Degree of
MASTER OF SCIENCE IN MATHEMATICS
(OPTIMIZATION)

Balemlay Yizengaw

June 2019

Haramaya University, Haramaya

HARAMAYA UNIVERSITY

Postgraduate Program Directorate

We hereby certify that we have read and evaluated this project titled 'Travelling Salesman Problem: Genetic Algorithm and Ant Colony Optimization' prepared under my guidance by Balemlay Yizengaw. We recommend that it be submitted as fulfilling the project requirement.

Getinet Alemayehu (PhD)	_____	_____
Major Advisor	Signature	Date
Seleshi Demie (PhD)	_____	_____
Co Advisor	Signature	Date

As member of the board of Examiners of the MSc. Project Open Defense Examination, we certify that we have read and evaluated the project prepared by Balemlay Yizengaw and examined the candidate. We recommend that the project be accepted as fulfilling the project requirement for the degree of *Master of Science in Mathematics* (Optimization).

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

DEDICATION

To all my family members who have been a constant source of motivation, inspiration, and support.

STATEMENT OF THE AUTHOR

By my signature below, I declare that this project is my own work. I have followed all ethical and technical principles of scholarship in the preparation, and compilation of this project. Any scholarly matter that is included in the project has been given recognition through citation.

This project is submitted in partial fulfillment of the requirements for MSc. degree in Mathematics with specialization in optimization at Haramaya University. The project is deposited in the Haramaya University Library and is made available to borrowers under the rules of the library. I solemnly declare that this project has not been submitted to any other institution anywhere for the award of any academic degree, diploma or certificate.

Brief quotations from this project may be made without special permission provided that accurate and complete acknowledgement of the source is made. Requests for permission for extended quotations from or reproduction of this project in whole or in part may be granted by the Head of the Department when in his or her judgment the proper use of the material is in the interest of scholarship. In all other instances, however, permission must be obtained from the author of the project.

Name: Balemlay Yizengaw Tsegaw

Date _____

Department: Mathematics

Signature _____

ABBREVIATIONS AND ACRONYMS

ACO	Ant Colony Optimization
ATSP	Asymmetric Travelling Salesman Problem
Chr	Chromosome
GA	Genetic Algorithm
LOX	Linear Order Crossover
NP-hard	Non-Deterministic Polynomial hard Problem
OX	Order Crossover
PMX	Partially Mapped Crossover
STSP	Symmetric Travelling Salesman Problem
TSP	Travelling Salesman Problem
TSPLIB	A library of Travelling Salesman Problem

BIOGRAPHICAL SKETCH

The author was born in 1995 on June 10 in Amhara Regional State, East Gojam Zone, Machakele Woreda. She attended his primary education at Amanuel kuter hulet Primary school. Then after, she joined Amanuel Secondary and Preparatory School to attend her secondary education. Then she joined Wollo University in 2014 and received Bachelor of Science degree in Mathematics on July 4, 2016. She directly joined Postgraduate Program at Haramaya University, College of Natural and Computational Sciences, Department of Mathematics in 2017 to pursue a program of study for MSc. degree in Mathematics with specialization in Optimization.

ACKNOWLEDGEMENTS

First and foremost, I would like to be grateful for the loving and kindness of the Almighty God in bestowing my health, strength, patience and protection throughout my life.

I am deeply beholden to my advisors **Dr. Getinet Alemayehu** and **Dr. Seleshi Demie** for the profound commitment and encouragements they have shown throughout my study period. I pay respect and express indebtedness to them because of their inspiring guidance, consistent supervision and especially, their suggestions in selection of tools from the start of the proposal and at every phase of this project work.

I would like to extend my gratitude to Ministry of Education for the provision of the necessary support to complete this postgraduate study.

Table of contents

Table of contents(*continuied*)

Table of contents(*continuied*)

LIST OF TABLE

LIST OF FIGURE

TRAVELLING SALESMAN PROBLEM: GENETIC ALGORITHM AND ANT COLONY OPTIMIZATION

ABSTRACT

Travelling salesman problem being an optimization problem is used to find shortest path of a given set of cities, given a set of cities and the cost of travel (or distance) between each possible pairs, the Travelling salesman problem, is to find the best possible way of visiting all the cities and returning to the starting point that minimize the travel cost (or travel distance). The main objective of this study was to solve travelling salesman problem by using Genetic Algorithm and Ant Colony optimization Algorithms. In this project, we discussed genetic algorithm and ant colony optimization methods and solved travelling salesman problem using these two methods. We applied both techniques to solve the said problem by using the same dataset illustrative examples. MATLAB code was used to find the solution of travelling salesman problem. Comparison was made between the two methods result based on relative error to determine the better one for solving this problem with regards to known optimal value. The result of the comparisons using the illustrative examples considered in this study showed that the genetic algorithm was better method than ant colony optimization. The present investigation can be extended by using different algorithms such as particle swarm optimization and Simulated Annealing and then compared in order to classify which algorithm gives the best optimal results under a given set of conditions.

Keywords:, ant colony optimization, genetic algorithm, travelling salesman problem

1. INTRODUCTION

1.1 Background of the Study

Optimization is the process of making something is better. It involves finding the minimum/maximum of an objective function $\mathbf{f}(\mathbf{x})$ subject to some constraints $\mathbf{x} \in \mathbf{X}$. If there is no constraint for $\mathbf{x}$ to satisfy or, equivalently, $\mathbf{X}$ is the universe, then it is called an unconstrained optimization; otherwise, it is a constrained optimization. Many real-world decision problems can be modeled by an optimization framework (Sundaram, 1996).

Combinatorial optimization means searching for an optimal solution in a finite or countable infinite set of potential solutions. Optimality is defined with respect to some criterion function, which is to be minimized or maximized. Examples of minimization: cost, distance, length of a traversal, weight, processing time, material, energy consumption, number of objects and Maximization: profit, value, output, return, yield, utility, efficiency, capacity, number of objects. The solutions may be combinatorial structures like arrangements, sequences, combinations, choices of objects, sequences, subsets, sub-graphs, chains, routes in a network, assignments, schedules of jobs, packing schemes. Graph modeling can be used in optimization problems encountered e.g. in telecommunications, data transmission, transportation and logistics, services and distribution networks, wiring, neural networks (Biggs *et al*. 1986).

Given a collection of cities and the distance of travel between each pair of them, the travelling salesman problem is to find the shortest way of visiting all of the cities and returning to the starting point. Though the statement travelling salesman problem is simple to state but it is more difficult to solve. Travelling Salesman Problem is an optimization problem and has a vast search space and is said to be NP-hard, which means it cannot be solved by using exact methods within a given amount of time (Bagchi *et al*., 2015).

Let us consider an example describing the Travelling salesman problem. We have a set of four cities A, B, C and D. The distances between the cities are also given to us. Figure 1 illustrates the collection of the cities and their distances among each other. Let the tour with A→B→C→D→A will be the optimal route for given problem.

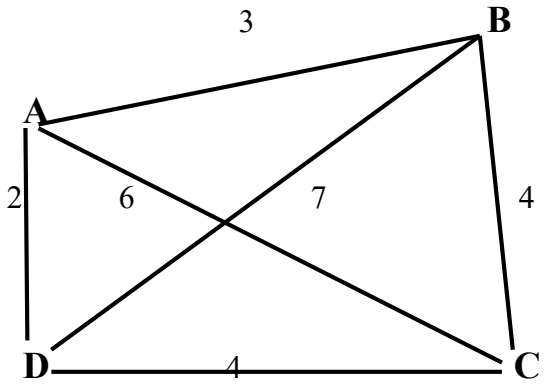


Figure 1 Travelling salesman problem.

Imagine a salesman who has the task of visiting a range of cities in order to sell his goods. He will probably start from a specific location to which he will also return after visiting each place just once. His route will be planned in a way that avoids detours and unnecessary effort; in short, the experienced travelling salesman will aim for the shortest possible route (Keller, 2004).

TSP being an optimization problem is used to find shortest path of a given set of cities, given a set of cities and the cost of travel (or distance) between each possible pairs, the TSP, is to find the best possible way of visiting all the cities and returning to the starting point that minimize the travel cost (or travel distance). The problem TSP is represented as an undirected graph, where cities are represented as vertices and paths are represented as edges in the graph. As a minimization problem it starts and finishes at a specified vertex by visiting each vertex at least once (Matai *et al*., 2010).

TSP is a classical model for various process sequencing and production scheduling problems. Many production scheduling problems can be reduced to a simple concept that there is a salesman who must travel from city to city, visiting each city exactly once and returning to the home city (Bagchi *et al*., 2006). It is possible for the salesman to select the orders of the cities visited so that the total distances travelled in his tour is as small as possible which will apparently save him time and money . The most popular practical application of TSP are regular distribution of goods or resources, finding of the shortest of costumer servicing route, planning bus lines etc.

Many optimization methods are used to find solution to the problem like greedy method, ant algorithms, simulated annealing, tabu search, genetic algorithms, branch and bound, dynamic programming and cutting plane method (Ahmed, 2010) but as the number of cities increases the computation to find the solution becomes difficult. Despite the computational difficulty, we can use methods like genetic algorithms, ant colony optimization, particle swarm optimization and tabu search which can give near to optimal solution for thousands of cities. In this project we can use genetic algorithms and ant colony optimization approaches used for solving travelling salesman problem.

Genetic algorithms are a type of optimization algorithm, meaning they are used to find approximate solution(s) to a given computational problem that maximizes or minimizes a particular function. Genetic algorithms represent one branch of the field of study called computation in that they imitate the biological processes of reproduction and natural selection to solve for the `fittest' solutions (Kinnear, 1994). Like in evolution, many of a genetic algorithms processes are random, however this optimization technique allows one to set the level of randomization and the level of control. Since genetic algorithms are designed to simulate a biological process, much of the relevant terminology is borrowed from biology. However, the entities that this terminology refers to in genetic algorithms are much simpler than their biological counterparts (Mitchell, 1995).

The fitness function is the function that the algorithm is trying to optimize (Mitchell, 1995). The word fitness is taken from evolutionary theory. It is used here because the fitness function tests and quantifies how fit each potential solution is. The term chromosome refers to a numerical value or values that represent a candidate solution to the problem that the genetic algorithm is trying to solve.

Ant colony optimization (ACO) was introduced by Dorigo and colleagues in ((Blum, 2005) as a novel nature inspired metaheuristic for the solution of hard combinatorial optimization problems. ACO belongs to the class of metaheuristics which are approximate algorithms used to obtain approximate solutions to hard combinatorial optimization problems in a reasonable amount of computation time. The inspiring source of ACO is the foraging behavior of real ants. As soon as an ant finds a food Source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the ant deposits a chemical

pheromone trail on the ground. The quantity of pheromone deposited will guide other ants to the food source. The ACO algorithm has been applied to the TSP which is the problem of finding a shortest closed tour which visits all the cities in a given set.

ACO has been widely applied to solving various combinatorial optimization problems such as Traveling Salesman Problem (TSP), Job-shop Scheduling Problem, Vehicle Routing Problem, Quadratic Assignment Problem etc. Although ACO has a powerful capacity to find out solutions to combinational optimization problems, it has the problems of stagnation and premature convergence and the convergence speed of ACO is very slow (Hlaing and Khine, 2011).

1.2 Statement of the Problem

The travelling salesman problem defined on a set of nodes and a set of edges which connect them. Each edge is associated with some cost. The problem is then to construct a solution in which each node is visited exactly once and one returns to the start node such that the sum of the arc distance is minimized. The travelling salesman problem (TSP) is an optimization problem which poses the question: "Given a set of cities and all the distances between them, what is the shortest route if each city is to be visited exactly once?".

The traveling salesman problem (TSP) is a typical example of a very hard combinatorial optimization problem. TSP is difficult to solve efficiently by conventional optimization techniques when its scale is very large. The earliest research on TSP problem was solved using exact methods such as branch-and-bound and dynamic programming. However the exact methods that guarantee to find the optimal solution of the problem but the major drawbacks to take very large computational time and only capable of handling small and medium size problem (Chong, 2001). Therefore it is necessary to use approximation methods for solving TSP problems such as genetic algorithm and ant colony optimization

This project tried to investigate good approximate solutions of travelling salesman problem by using GA & ACO algorithms.

This project was tried to answer the following questions:-

- How do Genetic algorithm (GA) and Ant Colony optimization (ACO) algorithm work to find the solution of travelling salesman problem?
- Which method is better (effective) from GA and ACO for solving the travelling salesman problem with regard to known optimal value?

1.3 Objective

The main objective of this project was to solve travelling salesman problem (TSP) by using Genetic Algorithm and Ant Colony optimization Algorithm.

The study intends to explore the following specific objectives:

- To describe genetic algorithm and ant colony optimization algorithm to solve travelling salesman problem.
- To make comparison between the results of two methods with regard to known optimal value.

2. LITERATURE REVIEW

2. Travelling Salesman Problem

The travelling salesman problem (TSP) concept is very easy stated, however it is hard to solve because classified as NP-hard problem. The main difficulty of this problem is because the immense number of possible tours: $(n-1)!/2$ for symmetric n cities tour and $(n-1)!$ for asymmetric. As the number of cities in the problem increases, the number of permutations of valid tours is also increasing e.g. for 5 cities is12, 7 cities is 360 and for 10 cities is 181,440 possible permutation. TSP can be described as a salesman who starts the journey from his home city, visiting each city exactly once and then return to his home city. It is possible for the salesman to select the order of his visits so that the total of the distances travelled in his tour is as small as possible. In this case, the problem is to minimize the total distance over the set of all tours or in terms of the graph theory, to find a Hamiltonian cycle of minimal length in a fully connected graph (Keller, 2004).

Reinelt (1995) made a collection of sample instances for travelling salesman problem and related problems from various sources and of various types with its known optimal solutions. Symmetric travelling salesman problem is one of the instances considered in this collection.

The TSP has received considerable attention over the last two decades and various approaches are proposed to solve the problem, such as cutting planes (Miliotis, 1978), 2-opt (Lin and Kernighan, 1973). Some of these methods are exact algorithms, while the others are near-optimal or approximate algorithms. The exact algorithms include the integer linear programming approaches with additional linear constraints to eliminate infeasible subtours. On the other hand, network models yield appropriate methods that are flexible enough to include the precedence constraints. More recently, GA approaches are successfully implemented to the TSP.

Huilian (2010) presented a novel hybrid discrete PSO algorithm by adding heuristic factor, crossover operator and adaptive disturbance factor into the approach.

Taiwo *et al*. (2013) studied an implementation method of solving TSP using Nearest Insertion and Nearest Neighbour Approaches. They provided a comparison between the two stating which algorithm gives the better result and what are the flaws in the other algorithm due to which it is not able to produce the desired result. The execution time of nearest insertion algorithm is slightly less than that of nearest neighbor algorithm. But it was observed that the solution could be found in very short computational time, which helped to reach the conclusion that the use of these two algorithms gives the acceptable results which may not be optimal but are close to the optimal result.

Particle swarm optimization-based algorithms are presented for TSP (Liang *et al*., 2007), where an uncertain searching strategy and a crossover eliminated technique is used to accelerate the convergence speed. A hybrid approach that joins PSO, Genetic Algorithms and Fast Local Search is presented by Machado & Lopes (2005) for the TSP. The positions of the particles represent TSP tours as permutations of cities. The value assigned to each particle (fitness) is the rate between a constant Dmin and the cost of the tour represented in the particle's position.

Goldbarg *et al*. (2006) presented a PSO algorithm for the TSP where the idea of distinct velocity operators is introduced. The velocity operators are defined according to the possible movements a particle is allowed to do. This algorithmic proposal obtained very promising results.

Gupta and Panwar (2013) solved the TSP using genetic algorithm. In this study the Euclidian distance between each city is given in the matrix format and the initial population was generated in random order. As a next step selection, crossover and mutation operators are applied. A Two point crossover is used where two positions in the chromosomes are chosen and then replaced the gene with each other in both chromosomes randomly. The process is continued until the termination condition is satisfied.

Ahmed (2010) studied various crossover methods. Since Crossover is the important step in the genetic algorithm, efficient crossover methods should be chosen. So a comparative study of various crossover methods such as Sequential Constructive Crossover, Generalized N-point Crossover and Edge Recombination Crossover are done here. The experimental result showed

that the sequential constructive crossover gives the high quality results for the travelling salesman problem.

Rani and Kumar (2014) investigated Roulette Wheel method for the Selection and also compared it with other selection method called Stochastic Universal Selection method where N equally spaced pointers are used to select the parents. The Stochastic Universal Selection method gives better result when the population size is small but when the problem size increases Roulette Wheel gives the better result. The crossover method used here is variation of Order Crossover (OX). In the new crossover method two cut points are selected. The nodes between the two cut points are copied and the rest of the nodes are selected from the second parent in relative order omitting the existing nodes and it is found that this method gives the better result than the existing crossover methods. The authors even mentioned that the Elitism method also gives the better result when the problem size is large.

Mohd Razali (2014) presented Genetic algorithm for process sequencing modelled as the travelling salesman problem with precedence constraints. This study was conducted under two main stages. The first stage is to study and examined the genetic algorithm procedure that is used to solve TSP and second stage is precedence constrained travelling salesman problem. At this stage, various genetic operators and parameters are explored.

Chudasama *et al*. (2011) solved travelling salesman problem by using genetic algorithm operators. And also included a comparative study on various parent selection methods such as Roulette Wheel, Elitism and Tournament selection methods for Travelling Salesman problem. The roulette wheel selection method selects the individual which is proportional to its fitness value, whereas in tournament selection every individual is paired with another individual in the population randomly. In the Elitism selection method the individuals are selected based on their fitness value. And finally they conclude that all the three selection methods give similar solution when the population size is small but when the population size is large Elitism method gives the better result.

Kumar (2012) described a survey on the travelling salesman problem using various genetic algorithm operators. The various methods for the genetic algorithm operators like selection methods, crossover methods and mutation methods are also mentioned.

Travelling salesman problem is solved using genetic algorithm operators. The genetic algorithm is used for the purpose of improving the solution space. The crossover is the important stage in the genetic algorithm. A new crossover method called sequential constructive crossover operator is used here. The sequential constructive crossover uses best edges of the parent's structure and produces the new offspring. It is compared against other existing crossover operators and it is proved that sequential constructive crossover results in a high quality solutions. And also includes a comparative study on Greedy Approach, Dynamic Programming and Genetic Algorithm for solving TSP (Dwivedi *et al*., 2012).

Aranganayaki (2011) investigated travelling problem is solved using genetic algorithm operators to reduce the total distance and time. This is achieved by generating the fittest criteria using selection, crossover and mutation operators. The main aim of this study was produce the high quality solutions in reasonable time. So a new crossover method, the Sequential Constructive Crossover method is used. This method was select the better edges from the parent chromosome and produce a new offspring which may have same edges as the parents or it may have new edges which is not present in the parent chromosomes.

Brezina and Čičková (2011) investigated solving the Travelling Salesman Problem Using the Ant Colony Optimization. The research is on TSP using ACO. Study the knock of some control parameters by actualize ACO algorithm. The individuality of the solution is analyzed with the optimal solution. It can be achieved that the affirmation of solutions depends on the number of ants. The lower number of ants allows the specific to change the path often faster. The greater number of ants in community causes the higher accession of pheromone on edges, and thus an invisible keeps the path with higher concentration of pheromone with a high rise probability.

Blum (2005) studied on Ant Colony Optimization Introduction and recent trends. The study showed that detailed description about the origin and the basic information of Ant colony optimization algorithm. They also gave the general description about the meta-heuristics of ACO.

Aggarwal and Saroj (2012) presented Compute travelling salesman problem using ant colony optimization. This study showed detailed description about the origin and the basic information of Ant colony optimization algorithm for TSP. They also gave the general

description about the parameters of ACO and the affection of parameters on the solution of TSP. The parameters are Number of ants, pheromone parameter (α). And also at the end the Performance of ACO for TSP has been analyzed under various parameters which shows that most optimal paths are obtained when the number of ants is equal to the no. of cities as well as the value of α should be close to 1.

Hlaing and Khine (2011) studied Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm. And also showed the changes in ant colony system, dynamic candidate list strategy, and proposed approach

a) Dynamic Candidate List Strategy

b) Heuristic parameter updating based on entropy and mergence of local search solution is proposed. An improved interpretation of ACO algorithm based on candidate list strategy and also used dynamic heuristic parameter updating establishes on degeneration and combine of local search solution is purposed. From the experimental results, the considered system is more capable than the ant colony system algorithm in terms of association speed and the ability to finding better solutions.

Wang *et al*. (2008) investigated a system in which the new ants memorize the best-so-far solution. The proposed model was called ants with memory or Mant. They did some important work in this project such as introducing the background knowledge of the TSP, ant system and ant colony system, provided definition of the parameters used and entire environment of the experiments, explains what ants with memory are and combined them in ant colony system, to amend the ants with memory, then the results were generated by amended ants and compare the performance of each algorithm, dedicated to discuss the main characters of ants with memory and suggesting directions for further research. In the end, it is showed through this project that the Mant is a very uncomplicated but interesting modernistic approach to the ACO system except the Average-Best in the table the statistical comparison told us that the probabilistic Mant has surpassed original ant in ant colony system at speed and quality while searching for the optimum solution. It has been showed to compare favorably with ant colony system algorithm, and its amelioration model probabilistic Mant got inspiring performance in ant colony system. However, competition on the TSP is very tough, and a utilization of best -

so-far tour information which converge these solutions to a near optimum seems to be a useful strategy.

Arora and Arora (2016) investigated Better Result for Solving TSP by using GA and ACO. This study also includes the comparative studies between GA and ACO. Finally the study showed that Genetic Algorithm gives better result in terms of distance travelled for less number of cities but as the author increases the complexity by increasing the number of cities, ACO proves to give better result than Genetic. While considering Execution time as the factor ACO is also proved to be better.

Bajpai and Yadav (2015) studied Ant Colony Optimization for the Traveling Salesman Problem (TSP) Using Partitioning. This work introduced the partitioning technique based on the divide and conquers strategy for the traveling salesman problem (which is one of the most important combinatorial problems) in which the original problem is partitioned into the group of sub problems. And then apply the ant colony algorithm using candidate list strategy for each smaller sub problems. Applying the local optimization and combining the sub problems to find the good solution for the original problem by improving the exploration efficiency of the ants. At the end of this work the two Authors' also be presented the comparison of result with the normal ant colony system for finding the optimal solution to the traveling salesman problem.

Angel *et al*. (1972) studied the problem of scheduling buses as a variation of the MTSP with some side constraints. The objective of the scheduling is to obtain a bus loading pattern such that the number of routes is minimized, the total distance travelled by all buses is kept at minimum, no bus is overloaded and the time required to traverse any route does not exceed a maximum allowed policy.

An application for deposit carrying between different branch banks is reported by (Svestka & Huckfeldt, 1973). Here, deposits need to be picked up at branch banks and returned to the central office by a crew of messengers. The problem is to determine the routes having a total minimum cost.

3. MATERIALS AND METHODS

This project work has been developed through:

- A number of reference books and materials which are available in the library of Haramaya universities and also collected some quite relevant information from the soft copies, internet, etc. Related journals were examined in detail and used to consolidate all the entire frameworks and skeleton of the project.
- Some necessary definitions, graphs and examples are contained in this project to illustrate the essential idea of genetic algorithm and ant colony optimization methods for solving travelling salesman problem.
- Genetic algorithm was used to solve travelling salesman problem.
- Ant colony optimization was used to solve travelling salesman problem.
- MATLAB programs were used to minimize the high consumption of time and energy that could happen during computations by hands and to accomplish the work easily.
- MATLAB code was used to plot its approximate route.

4. PRELIMINARIES

Definition 4.1. Complete graph is a graph with N vertices in which every pair of distinct vertices is connected by a unique edge.

4.1. Traveling Salesman Problem

Travelling salesman problem (TSP) consists of finding the shortest route in complete graph G with n nodes and $n(n-1)/2$ edges, so that the start node and the end node are identical and all other nodes in this tour are visited exactly once. There are several cases of TSP (not all are mentioned) (Matai *et al*., 2010).

- Symmetric travelling salesman problem (STSP): distance between nodes i and j is the same as distance between j and i.
- Asymmetric travelling salesman problem (ATSP): distance between nodes i and j is not the same as distance between j and i.
- Precedence constrained travelling salesman problem: nodes can be visited only in given order.
- Etc.

Further in this project is considered only a symmetric travelling salesman problem (STSP). Since TSP is NP-hard problem the number of possible solutions for graph with n nodes is $(n-1)!\,/\,2$ the exact algorithm can be used only for small number of nodes. Therefore only heuristics or metaheuristics are used for solving large TSP.

Definition 4.2. A Hamiltonian cycle is a cycle in a graph passing through all the vertices once.

Example:

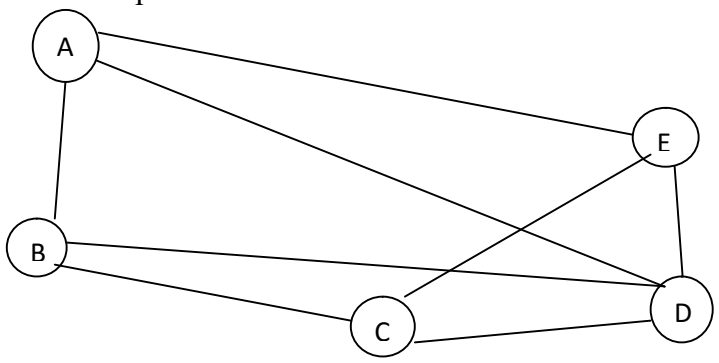


Figure 2 A graphs with various Hamiltonian paths.

P = {A, B, C, D, E} is a Hamiltonian cycle.

Definition 4.3. In graph theory travelling salesman problem defines as a task of finding of the shortest Hamiltonian cycle or path in complete graph of V nodes. A graph in which every pair of distinct vertices is adjacent is called a complete graph. Two vertices are adjacent if they are connected by an edge.

4.1.1. Mathematical problem formulation of TSP

Mathematically, the goal of this problem is to find a tour, among all the possible tours, that minimizes the total distance the salesman travels. The other criterion that the problem has to satisfy is that each city should be visited once and only once, except that he returns to the city from which the salesman starts. TSP can be formulated as an integer linear program. Let C_{ij} be the distance from city i to city j . Label the cities with the numbers $0, \dots, n$. For n cities to visit, let x_{ij} be the variable that has a value 1 if the salesman goes from city i to city j and a value 0 if the salesman does not go from city i to city j . Then, the mathematical formulation of TSP is as follows (Brezina and Čičková, 2011):-
Minimize the linear objective function:

$$Z = \text{minimize } \sum_{i=0}^{n} \sum_{j \neq i\, j=0}^{n} c_{ij}\, x_{ij} \qquad (4.1)$$

$$x_{ij} = \begin{cases} 1 \text{ if the salesman goes from city } i \text{ to } j \\ 0 \text{ if the salesman does not go from city } i \text{ to } j \end{cases}$$

Subject to;

$$\sum_{i=0}^{n} x_{ij=1} \qquad j = 0,1,2,3, \cdots \cdots \cdots \cdot, n; i \neq j \qquad (4.2)$$

$$\sum_{j=0}^{n} x_{ij=1} \qquad i = 0,1,2,3, \cdots \cdots \cdots \cdots, n; j \neq i \qquad (4.3)$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad i, j = 1,2\ 3 \cdots \cdots, n \qquad (4.4)$$

All $x_{ij} = 0$ or 1 and is a set of integers (4.1) is used to minimize the total cost or distance travelled in a tour. Constraint set (4.2) and (4.3) ensures, respectively that the salesman enter and leaves each city exactly once. Constraint (4.4) ensures excludes subtours

4.1.2. Methods to solve TSP

There are many algorithms to solve TSP. There are two possible approaches of optimization algorithm that can be classified as exact and heuristic (approximate) method. The exact methods will generate all possible solutions, while the heuristic methods only generate solutions according to evolution algorithm. It means that heuristic methods do not generate all solutions for the problem and do not guarantee the optimal solution (Chong, 2001).

Exact methods like branch-and-bound, linear programming and dynamic programming are seen to be an effective solution of combinatorial optimization problems and they have particular advantages and disadvantages. These methods always lead to the optimal solution. On the other hand there exists a limit on problem size for exact methods.

Heuristics or approximate methods were developed to find the near optimal solution for larger dimension problems within a reasonable time. The term heuristics derive from the Greek heuriskein meaning to find or discover. Heuristic algorithms have become a popular alternative to exact algorithms mainly because of their ability to handle more complex problems, larger size problems, and the numerous side constraints.

The most widely and successfully applied heuristic algorithms are local search algorithm. The general schemes to improve local search algorithms are called metaheuristics. The metaheuristics algorithm has often been inspired by analogies to naturally occurring phenomena like the physical annealing of metals or biological evolution. The most famous metaheuristics include genetic algorithm, simulated annealing, tabu search, particle swarm and ant colony.

4.2 Relative Error

A natural question arising for approximate or heuristic algorithms is how close, in the worst case, is the returned solution to the optimum. To indicate the quality of the returned solution, the relative error is defined as follows: The relative error of a feasible solution value with respect to optimal solution value reported in TSPLIB website, as given by the formula:

$$\text{Error} = \frac{\text{approximate solution value} - \text{optimal solution value}}{\text{optimal solution value}} \times 100 \qquad (4.1)$$

The relative error is close to 0 if the feasible solution is close to the optimum. Conversely, the relative error is close to 1 if the feasible solution is far from the optimal solution (Ahmed, Z.H. 2010)

5. TRAVELLING SALESMAN PROBLEM: GENETIC ALGORITHM AND ANT COLONY OPTIMIZATION

5.1 Genetic algorithm for TSP

5.1.1 Representation of tours

The first task to solve in every Genetic Algorithm is to represent individuals, i.e. possible solutions or tours in this case. A good representation supports GA operators to perform easily and fast on them. Many different representations have been proposed to the TSP using Genetic Algorithms. In this project we can use Path representation. Path representation also called permutation representation is the most natural representation of a tour. Again, a TSP tour is represented by a list of n cities. The tour: (2-3-6-4-5-1) is simply represented by: (236451).
Advantages of the path representation include simplicity of fitness evaluation and usefulness of final representation. The fitness of a given individual (TSP tour) is easy to evaluate, since it can be calculated by summing the costs of each pair of adjacent nodes.

5.1.2 Selection

A mechanism to select individual in population for reproduction to create new offspring or to transfer a part of the existing population to the next generation is needed. It is possible to perform the task of selection completely in a randomized fashion. This selection mechanism will eventually cause the algorithm to reach global minimum/maximum. The selections strategy addresses on which of the chromosomes in the current generation will be used to reproduce offspring in hopes that the next generation will have even higher fitness. A number of selection techniques exist including elitist, tournament, Roulette Wheel and rank-based Roulette Wheel. The differing selection techniques all develop solutions based on the principle of survival of the fittest. Fitter solutions are more likely to reproduce and pass on their genetic material to the next generation in the form of their offspring.

5.1.2.1 Elitist selection

Elitism is a general concept to favor the top individuals and to ignore the remaining ones. Individuals in the population are sorted according to their fitness values. The best n individuals are included in the selection process and the remaining individuals are discarded.

5.1.2.2 Tournament selection

In tournament selection technique, n individuals are selected from the larger population, and the selected individuals compete against each other. The individual with the highest fitness wins and will be included in the mating pool. The tournament selection also gives a chance for all individuals to be selected and thus it preserves diversity, although keeping diversity may degrade the convergence speed.

5.1.2.3 Roulette wheel selection

In keeping with the ideas of natural selection, it assumes that stronger individual, that is, those with higher fitness values, is more likely to mate than the weaker ones. One way to simulate this is to select parents with a probability that is directly proportional to their fitness values. This method is called the roulette wheel (or proportional Roulette Wheel) method. The idea behind the roulette wheel selection technique is that each individual is given a chance to become a parent in proportion to its fitness. The chances of selecting a parent can be seen as spinning a roulette wheel with the size of the slot for each parent being proportional to its fitness. Obviously, those with the largest fitness (slot sizes) have more chance of being chosen.

5.1.2.4 Rank-based Roulette Wheel selection

Rank-based Roulette Wheel selection in which the probability of a chromosome being selected is based on its fitness rank relative to the entire population. Rank-based selection schemes first sort individuals in the population according to their fitness and then computes selection probabilities according to their ranks rather than fitness values. Hence rank-based selection can maintain a constant pressure in the evolutionary search where it introduces a uniform scaling across the population and is not influenced by super-individuals or the spreading of fitness values at all as in proportional selection. Rank-based selection uses a function to map the indices of individuals in the sorted list to their selection probabilities.

5.1.3 Crossover

After the completion of the selection process, the chromosomes chosen to be parents for the next generation are recombined to form children that are new chromosomes. In nature, although it may be much more complicated, crossover basically occurs as follows: chromosomes of both parents are randomly divided from the same gene positions into a

number of segments and the corresponding segments are exchanged and copied to the chromosome of the newly created offspring. Therefore, the offspring inherit traits from the both parents. In the genetic algorithm, special techniques for permutation-based chromosomes are deployed, which ensure that, when applied on two permutation-based chromosomes, the chromosomes of the resulting offspring are also valid permutations. Some of the most popular generic permutation-based crossover techniques in genetic algorithm are partially mapped crossover (PMX), order crossover (OX), Linear order crossover (LOX) and cycle crossover (CX).

5.1.3.1 Partially mapped crossover (PMX)

The main purpose of a crossover operator is to create offspring that inherits traits from both parents. PMX passes on ordering and value information from the parent tours to the offspring tours. A portion of one parent's string is mapped onto a portion of the other parent's string and the remaining information is exchanged. For example, consider chromosomes which represent TSP are given as follows: $P1 = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$ and $P2 = (3\ 7\ 5\ 1\ 6\ 8\ 2\ 4)$. On this two parent chromosomes, a two cut points are randomly selected. Suppose that the first cut point is selected between the third and the fourth gene, and the second one between the sixth and seventh gene.

$$P1 = 1\ 2\ 3|4\ 5\ 6|7\ 8$$

$$P2 = 3\ 7\ 5|1\ 6\ 8|2\ 4$$

The substrings between the cut points are called the mapping sections. In this example, the mapping section is $4-1, 5-6$ and $6-8$. Now the segments (mapping section) between cut points are swapped. The mapping section of the first parent is copied into the second offspring$(O2)$, and the mapping section of the second parent is copied into the first offspring $(O1)$, become;

$$O1 = * * * |1\ 6\ 8| * *$$

$$O2 = * * * |4\ 5\ 6| * *$$

Then we can fill further tour cities (from the original parents), for which there is no conflict;

$$O1 = *\ 2\ 3\ 1\ 6\ 8\ 7\ *$$

$$O2 = 3\ 7 * 4\ 5\ 6\ 2 *$$

The first * in the offspring O1 would be 1 which the same as the first element is of parent P1. However there was a conflict (visiting city 1 twice) and hence is replaced by 4, because of the mapping $1 - 4$. The last *of offspring O1 would be an 8, which is already present. Because of the mappings $8 - 6,$ and $6 - 5$, it is chosen to be a 5. Hence;

$$O1 = 4\ 2\ 3\ 1\ 6\ 8\ 7\ 5$$

$$O2 = 3\ 7\ 8\ 4\ 5\ 6\ 2\ 1$$

5.1.3.2 Order crossover (OX)

Order crossover (OX) constructs an offspring by choosing a subtour of one parent and preserving the relative order of the cities of the other parent. For example, consider the following two parent tours;

$$P1 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$$

$$P2 = 2\ 4\ 6\ 8\ 7\ 5\ 3\ 1$$

Suppose that we select a first cut point between the second and the third gene and a second one between the fifth and the sixth gene. Hence;

$$P1 = 1\ 2|3\ 4\ 5|6\ 7\ 8$$

$$P2 = 2\ 4|6\ 8\ 7|5\ 3\ 1$$

The offspring is created in the following way. First, the tour segments between the cut points are copied into the offspring, which gives

$$O1 = * * |3\ 4\ 5| * * *$$

$$O2 = * * |6\ 8\ 7| * * *$$

Next, starting from the second cut point of one parent, the rest of the city are copied in the order in which they appear in the other parent, and omitting the city that are already present. Reaching the end of the parent string, we continue from its first position. Now the sequence of the cities in the second parent from the second cut point is 5 3 1 2 4 6 8 7. After removal of cities $3, 4$ and 5, which are already in the first offspring, we get 1 2 6 8 7. This sequence is placed in the first offspring, starting from the second cut point, which gives;

$O1 = 8\ 7|3\ 4\ 5|1\ 2\ 6$

The sequence of the cities in the first parent from the second cut point is 6 7 8 1 2 3 4 5. After removal of cities 6, 8 and 7, which are already in the second offspring, we get 1 2 3 4 5. This sequence is placed in the second offspring, starting from the second cut point, which gives;

$O2 = 4\ 5|6\ 8\ 7|1\ 2\ 3$

The OX crossover exploits a property of the path representation that the order of cities (not their positions) is important i.e., the two tours 6 7 8 1 2 3 4 5 and 12 3 4 5 6 7 8 are identical.

5.1.3.3 Linear order crossover (LOX)

Linear order crossover (LOX) is a modified version of the order crossover operator. Recall that the order crossover operator treats the chromosome as a circular string, in which it wraps around from the end of the chromosome back to the beginning. This circular assumption may not play a big role in the TSP. As such, the LOX operator treats the chromosome as a linear entity. For this operator, the swap occurs in the same fashion as it occurs in the OX operator, but when sliding the parent values around to fit in the remaining open slots of the child chromosome, they are allowed to slide to the left or right. This allows the chromosome to maintain its relative ordering and at the same time preserve the beginning and ending values. In the below example, after the values are swapped, there are two open spaces in the front of the chromosome and three open spaces at the end. The algorithm then goes through Parent 1 and finds the first two values that were not part of the swap, in this example they are 5 and 4. These values are shifted left to fill the first two chromosome locations. The final three locations are filled in a similar manner.

$P1 = 3\ 9|5\ 4\ 6\ 2|7\ 1\ 8$

$P2 = 7\ 4|3\ 8\ 9\ 2|1\ 5\ 6$

$O1 = * * 3\ 8\ 9\ 2\ * * *$

$O2 = * * 5\ 4\ 6\ 2\ * * *$

$O1 = 5\ 4\ 3\ 8\ 9\ 2\ 6\ 7\ 1$

$O2 = 7\ 3\ 5\ 4\ 6\ 2\ 8\ 9\ 1$

5.1.4 Mutation

The operation of mutation allow new individual to be created. It begins by selecting an individual from the population based on its fitness. A point along the string is selected at random and the character at that point is randomly changed, the alternate individual is then copied in to the next generation of the population. Mutation is performed after crossover by randomly choosing a chromosome in the new generation to mutate. We randomly choose a point to mutate and switch that point. Many types of mutation operators exist.

5.1.4.1 Inversion mutation

The inversion mutation operator randomly selects two cut points in the chromosome, and it reverses the subtour between these two cut points. Suppose that the first cut point is chosen between city 9 and city 5, and the second cut point between the sixth and seventh city. For example, consider the tour

Parent: 3 9|5 4 6 2|7 1 8

This result in

Offspring: 3 9 2 6 4 5 7 1 8

5.1.4.2 Insertion mutation

The insertion mutation operator selects a gene at random and then inserts it at a random position. Suppose that the insertion mutation operator selects city5, removes it, and randomly inserts it after city 7. For example, consider again the tour

Parent: 3 9 5 4 6 2 7 1 8

Hence, the resulting offspring is

Offspring: 3 9 4 6 2 7 5 1 8

5.1.4.3 Displacement mutation

The displacement mutation operator first selects a subtour at random. This subtour is removed from the tour and inserted in a random place. For example, consider the tour represented by

Parent: 3 9 5 4 6 2 7 1 8

Suppose that the tour (5 4 6) is selected. Hence, after the removal of the subtour we have(3 9 2 7 1 8), and suppose we randomly select city 7 to be the city after which the subtour is inserted. This result in

Offspring: 3 9 2 7 5 4 6 1 8

5.1.4.4 Exchange mutation

The exchange mutation operator, also known as reciprocal exchange mutation or swapping randomly selects two cities in the tour and exchanges them. For example, consider the tour represented as below and suppose that third and the eighth city are randomly selected.

Parent: 3 9 5 4 6 2 7 1 8

This result in

Offspring: 3 9 7 4 6 2 5 1 8

5.1.5 Genetic algorithm parameters

One of the main difficulties in building a practical GA is in choosing suitable values for parameters such as population size, crossover rate, and mutation rate De Jong's guidelines are still widely followed which is to start with a relatively high crossover probability (0.6-0.7), relatively low mutation probability (typically set to 1/l for chromosomes of length l), and a moderately sized population (50-500).

- **Population size:** - It determines how many chromosomes and thereafter, how much genetic material is available for use during the search. If there is too little, the search has no chance to adequately cover the space. If there is too much, the GA wastes time evaluating chromosomes.
- **Crossover probability**: - It specifies the probability of crossover occurring between two chromosomes.
- **Mutation probability**: - It specifies the probability of doing bit-wise mutation.
- **Termination criteria (**maximum number of generations): - It specifies when to terminate the genetic search.

5.1.6. Genetic algorithm procedure for TSP

The algorithm starts with supplying important information to the GA program such as the location of the cities, distance, time or cost incurred between the cities, and the GA Parameters such as maximum number of generations, population size, the probability of crossover and the probability of mutation. The algorithm will generate an initial random population of chromosomes in which path also called permutation is used as a chromosome representation.

Then the algorithm continues with evaluating the fitness of each chromosome according to the fitness function, and selecting the best chromosomes as parents for reproduction according to selection mechanism.

In every generation the offspring (i.e. new chromosomes) are produced by a combination of linear order crossover and inversion mutation mechanism with specified probability of crossover(P_c) and probability of mutation (P_m) . A new set of chromosomes where the size is equal to the initial population size (pop_size) is evolving. The procedure of the GA algorithms for solving TSP is described as follow:

Step 1: Initialization & representation

The purpose of initialization step is to establish an initial chromosome population of the given TSP problem. Each gene of a chromosome takes a label of node such that no node can appear twice in the same chromosome. The initial chromosome is very useful to create a new chromosome which is known as offspring in the next generation. The chromosome representation process is performed by applying Path representation.

Step 1.1: Set GA parameters and the problem information

The problem information such as the location of the cities or the distance matrix for a particular TSP is the most important input to the GA program. The distance or cost matrix also can be automatically generated by supplying the location of the cities in the GA program. The GA parameters such as population size, the termination criteria (i.e. maximum number of generations), the probability of crossover and probability of mutation must be set earlier in the program.

Step 1.2: Generate random initial population

In order to create an initial chromosome population, random permutation method is used. The random permutation method creates initial chromosome by generating numbers between 1 and the total of string, n in random sequence. Therefore, if the chromosome population size,pop_size $= 10$ it means there are 10 sets of chromosomes which consists of a number from 1 to n in random sequence. Here, the number of strings n represents the number of cities.

Step 2: Evaluation & selection

The evaluation of the chromosome is performed by measuring the fitness of each chromosome. Selection is then performed to choose the chromosome to be re-generated for the next generation**.**

Step 2.1: Evaluation

Every chromosome is evaluated by calculating the fitness value using a fitness function. If the distance between the cities is known for n cities location, we can calculate the total distance (fitness value) of each tour (chromosome) in the population.

Step 2.2: Selection

The selection process is performed by applying the proportional Roulette Wheel method. The probability of an individual to be selected is simply proportionate to its fitness value. The roulette wheel selection scheme can be implemented as follows:

1. Evaluate the fitness, f_i of each individual (chromosome) in the population.
2. Compute the probability p_i of selecting each member of the population:
$p_i = f_i / \sum_{J=1}^{n} f_J$, where n is the population size.
3. Calculate the cumulative probability q_i , for each individual: $q_i = \sum_{J=1}^{i} p_J$
4. Generate a random number $r \in (0,1]$
5. If $r < q_1$ then select the first chromosomex_1, else select the individual x_i such that $q_{i-1} < r \leq q_i$
6. Repeat steps 4–5 n times to create n candidates (chromosomes) in the mating pool.

To illustrate, consider a population with five individuals($n = 5$), with the fitness values as shown in the table below. The total fitness $f_J = 28 + 18 + 14 + 9 + 26 = 95$. The probability of selecting an individual and the corresponding cumulative probabilities are also shown in the table below.

Table 1 the probability of selecting an individual and the corresponding cumulative probabilities

Chromosome	Fitness, f	Probability p_i	Cumulative probability,q_i
1	28	28/95=0.295	0.295
2	18	0.189	0.484
3	14	0.147	0.631
4	9	0.095	0.726
5	26	0.274	1.000

Now if we generate a random number r say 0.585, then the third chromosome is selected as q_2=0.484<0.585≤q_3=0.631.

Step 3: Generation of offspring

The new chromosomes (i.e. offspring) are produced through two different mechanisms which is crossover and mutation. The two selected parent chromosomes are first mating by crossover techniques and produced two new offspring. These two new offspring is then mutated in order to further improve their genetic material.

Step 3.1: Crossover

In this step, firstly, a random number $P_c \in [0.1]$ is generated. This number is the percentage of the population on which the crossover is performed. The crossover operation required two parent chromosomes of the population to create two new chromosomes. Linear order crossover is used to create two new offspring. Chromosome number k will be selected for crossover if random generated value for Chromosome k belowP_c.

As an example, assume that the parent chromosomes are Pa = [3 9 5 4 6 2 7 1 8] and Pb = [7 4 3 8 9 2 1 5 6]. The crossover point is then selected randomly between second and third string and between sixth and seventh string. The selected sections are then swapped. Thus, the swap_sect are $O_1 =$ [3 8 9 2] and $O_2 =$ [5 4 6 2] . Then the remaining unselected string form Pa is filled in Offspring 1, which finally produce $O_1 =$ [5 4 3 8 9 2 6 7 1] . Similarly the procedure is applied to Offspring 2 by selecting the remaining string from Pb. Therefore,$O_2 =$ [7 3 5 4 6 2 8 9 1].

Step 3.2: Mutation

In this step, firstly, a random number $P_m \in [0,1]$ is generated. This number is the percentage of population on which the mutation is performed. Mutation operation is performed in a single chromosome to create a single new chromosome. Generate a random number between 0 and 1 for each gene if random numbers for all genes less than mutation rate P_m then this section of genes are mutated. Here, inversion mutation is applied to an individual after going through a crossover process. The inversion mutation procedure starts by selecting two cut points randomly in the chromosome. Then, the section of these genes is reversed (flip left to right) to create a new chromosome. For example the chromosome [7 3 | **5 4 6 2 8** | 9 1] is mutated becoming [7 3 8 2 6 4 5 9 1] .

Example 5.1

Suppose we have a TSP problem with 5 cities. The distance between cities is given by the following matrix.

Table 2 Distance (d) between each pair of cities

Cities/ Cities	1	2	3	4	5
1	0	10	12	11	14
2`	10	0	13	15	8
3	12	13	0	9	14
4	11	15	9	0	16
5	14	8	14	16	0

Determine the best route with minimum total distance by using the GA.

Step 1:

Parameter Setting:

Pop_size=3;

$P_m = 0.1$

$P_c = 0.6$

Random initial chromosomes are

$\text{Chr } 1 = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 1$

$\text{Chr } 2 = 5 \quad 3 \quad 1 \quad 2 \quad 4 \quad 5$

$\text{Chr } 3 = 3 \quad 4 \quad 5 \quad 2 \quad 1 \quad 3$

Step 2: Evaluation

Every chromosome is evaluated by calculating the fitness value (total distance) using a fitness function.

Total distance of Chr $1 = 10 + 13 + 9 + 16 + 14 = 62$

Total distance of Chr $2 = 14 + 12 + 10 + 15 + 16 = 67$

Total distance of Chr $3 = 9 + 16 + 8 + 10 + 12 = 55$

The fittest chromosomes have higher probability to be selected for next generation. To compute the fitness probability by using 1⁄(total distance of each chromosome)

Fitness of Chr $1 = {}^{1}/_{62} = 0.016$

Fitness of Chr $2 = {}^{1}/_{67} = 0.014$

Fitness of Chr $3 = {}^{1}/_{55} = 0.018$

Step 3: Selection

The selection strategy process is performed by using roulette wheel selection technique.

Step 1. Compute the probability p_i of selecting each member of the population:

$p_i = f_i / \sum_{J=1}^{n} f_J$, where n is the population size.

P (Chr i) = (Fitness (Chr i))⁄(total fitness of Chr)

Total fitness of Chr$= 0.016 + 0.014 + 0.018 = 0.048$

P(Chr 1) $= {}^{0.016}/_{0.048} = 0.333$

P(Chr 2) $= {}^{0.014}/_{0.048} = 0.291$

$$P(\text{Chr } 3) = {}^{0.018}/_{0.048} = 0.375$$

Step 2. Calculate the cumulative probability q_i , for each individual: $q_i = \sum_{j=1}^{i} p_j$

$$\boldsymbol{q_1} = \mathbf{q}\ (\text{Chr } 1) = 0.333$$

$$\boldsymbol{q_2} = \mathbf{q}(\text{Chr } 2) = 0.333 + 0.291 = 0.624$$

$$\boldsymbol{q_3} = \mathbf{q}\ (\text{Chr } 3) = 0.333 + 0.291 + 0.375 = 0.999$$

Step 3. . Generate a random number $r \in (0,1]$

$$R_1 = 0.291$$

$$R_2 = 0.492$$

$$R_3 = 0.408$$

Step 4. . If $R_1 < q_1$ then select the first chromosomex_1, else select the individual x_i such that $q_{i-1} < r \leq q_i$

If random number R_1 is less than $\boldsymbol{q_1}$ then select chr 1 as a chromosome in the new population for next generation:

new chr1 = chr 1

If random number R_2 is greater than $\boldsymbol{q_1}$ and smaller than $\boldsymbol{q_2}$ then select Chr2 as a chromosome in the new population for next generation by using the same process we get the following new chromosomes for next generation

new Chr1 =Chr 1

new Chr2 =Chr 2

new Chr 3 =Chr 2

Chromosomes in the population thus became:

new chr1 = 1 2 3 4 5 1

new chr2 = 5 3 1 2 4 5

new chr3 = 5 3 1 2 4 5

Step 4: Crossover

Parent chromosome which will mate is randomly selected and the number of mate Chromosomes is controlled using crossover rate (P_c) parameters. For the Crossover process we use LOX. Chromosome k will be selected as a parent if $R_k < P_c$ suppose we set that the crossover rate is 0.6 then Chromosome number k will be selected for crossover if random

generated value for Chromosome k below 0.6. The process is as follows: First we generate a random number R as the number of population.

$R_1 = 0.231$

$R_2 = 0.682$

$R_3 = 0.521$

For random number R above, parents are new Chr1 and new Chr3 will be selected for crossover.

Linear order crossover is used to create two new offspring:-

New chromosome (new offspring) = new chr1 ><new chr3

1 2 | 3 4 | 5 1 >< 5 3 | 1 2 | 4 5

new offspring1 =∗∗ 1 2 ⋆= 3 4 1 2 5 3

new offspring3 =∗∗ 3 4 ⋆= 5 1 3 4 2 5

Thus Chromosome population after experiencing a crossover process:

new chr1 = 3 4 1 2 5 3

new chr 2 = 5 3 1 2 4 5

new chr3 = 5 1 3 4 2 5

Step 5: Mutation

Number of chromosomes that have mutated in a population is determined by the mutation_rate parameter. Generate random number between 0 and 1 for all 30 genes.

Table 3 Generate random number between 0 and 1 for all 25 genes

Chr number	Random number				
1	0.25	0.54	0.01	0.020	0.39
2	0.238	0.92	0.56	0.53	0.90
3	0.52	0.80	0.48	0.26	0.58

If random numbers for all genes less than mutation rate $P_m = 0.1$ then this section of genes are flip left to right.

Thus Chromosome population after experiencing a mutation process

new chr1 = 3 4 2 1 5 3

new chr 2 = 5 3 1 2 4 5

new chr3 = 5 1 3 4 2 5

Finishing mutation process then we have one iteration or one generation of the genetic algorithm. These new Chromosomes will undergo the same process as the previous generation of Chromosomes such as evaluation, selection, crossover and mutation and at the end it produce new generation of Chromosome for the next iteration. This process will be repeated until a predetermined number of generations. For this example, by using maximum generations (2) best chromosomes (route) are obtained:

ch1=5 3 1 2 4 5 with total distance 67

ch2=5 3 2 1 4 5 with total distance 64

ch3=5 1 3 4 2 5 with total distance 58

The best tour with minimum distance is an optimal solution (tour):-

Chromosome (tour) = 5 1 3 4 2 5 with best total distance of route 58

Example 5.2

We have used GA on a well-known problem instance of classical city dataset from TSPLIB (Reinelt, 1995) in which the optimal solution is known. There are three symmetric TSP problems called burma14, bay29 and dantzig42. burma14, bay29 and dantzig42, have 14, 29 and 42 cities, respectively.

We set the GA parameters that are used in each problem according to many experiments. We have taken from (Mohd Razali, 2014) that means pop_size= 140 600 800,Pc = 0.9 0.9 0.95 , Pm = 0.01; ngener = 50,150,250. Note that different parameters setting are required for different problems. The larger the problem size, the larger population size and maximum number of generations used to ensure better search process. Figure 3 Through Figure 5 presents the optimal tour.

Table 4 GA parameters setting for TSP

Problem	pop_size	Pc	Pm	Ngener
burma14	140	0.9	0,01	50
bay29	600	0.9	0,01	150
dantzig42	800	0.95	0,01	250

Using the parameters given above in Table 4 and Run by using MATLAB we obtain the following solutions (results) and shortest path for each problem.

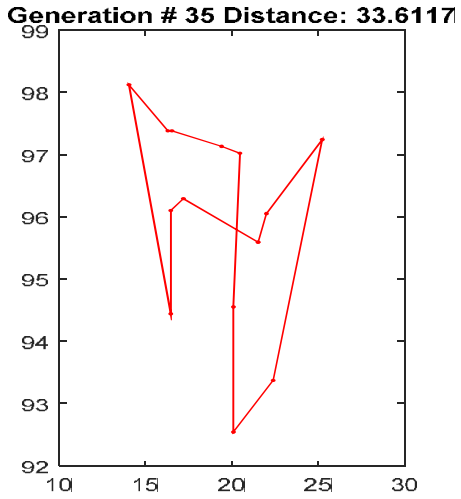


Figure 3 Optimal tours for burma14 with GA algorithm

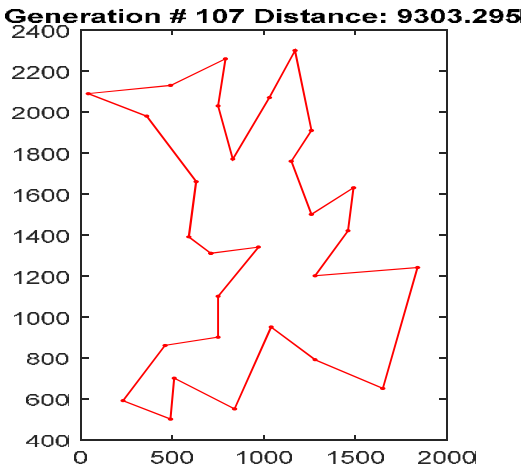


Figure 4 Optimal tours for bay29 with GA algorithm

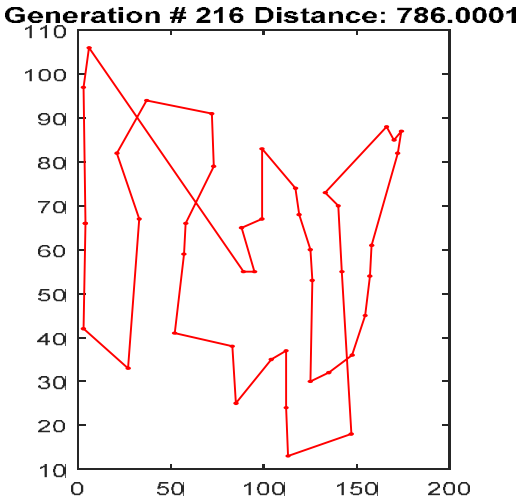


Figure 5 Optimal tours for dantzig42 with GA algorithm

Table 5 Quality of solution (Best solution) comparisons for TSPLIB problem

Problem	Known optimal value	Approximate solution by using GA
burma14	31	33.6117
bayg29	1610	9303.295
dantzig42	699	786.0001

The purpose of TSP experiments is to evaluate and to test the effectiveness of the GA procedure that uses different combination of parameters. To indicate the quality of the returned solution, the relative error is calculated based on equation (4.1). The percentage of relative error of the best solution obtained with respect to the known optimal value is presented in Table 6.

Table 6 Percentage of relative error for all TSP instances by using GA

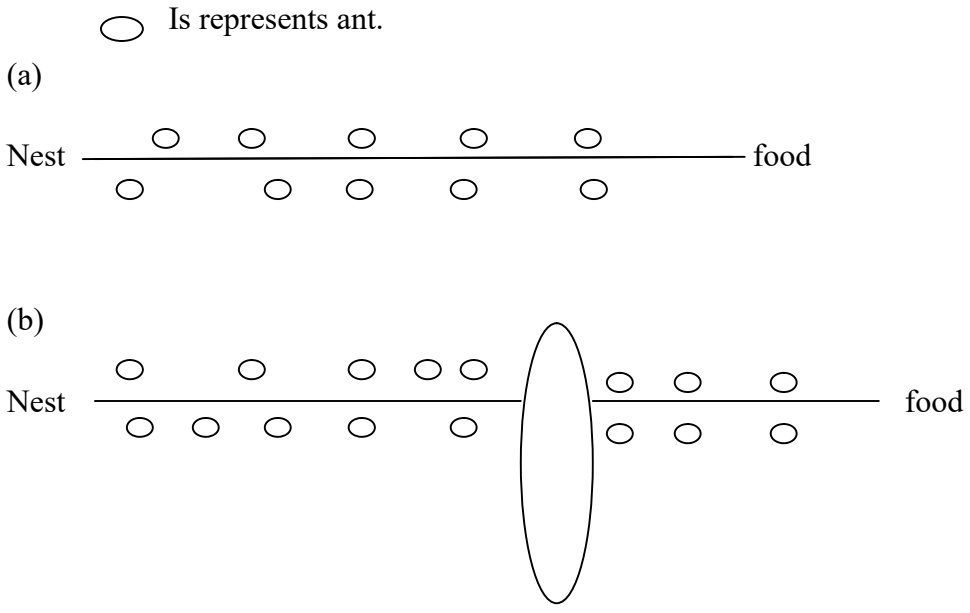
Problem	GA (%)
burma14	8.42
bay 29	477.84
dantzig42	12.44

5.2. Ant colony optimization for TSP

5.2.1 The concept of ACO

Real ants follow a path between nest and food source. An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. Ant leaves volatile hormone when it travels. Each ant can smell a special signal called pheromone. Ants utilize pheromone to communicate with each other and exchange information. Pheromone is deposited more quickly on the shorter path and all ants have chosen the shorter path. Figure 6 shows the behavior of real ants', in figure 6(a), the ants find food and move to the direction of the food. They leave pheromone along the path to which they move. As shown in figure 6 (b) and (c), a barrier blocks stop the ants' direct path to the food, so the ants pass the barrier by taking either the upper or the lower path around the barrier. The ants moving along the upper path get back quicker than those moving along the lower path because that the upper path is shorter than the lower one. Therefore, the amount of pheromone on the upper path is larger than the lower path in a period. According to the amount of pheromone, the following ants know which path is the shorter path, as shown in figure 6 (d).

Based on the above phenomenon, the ACO algorithm is constructed as follow:



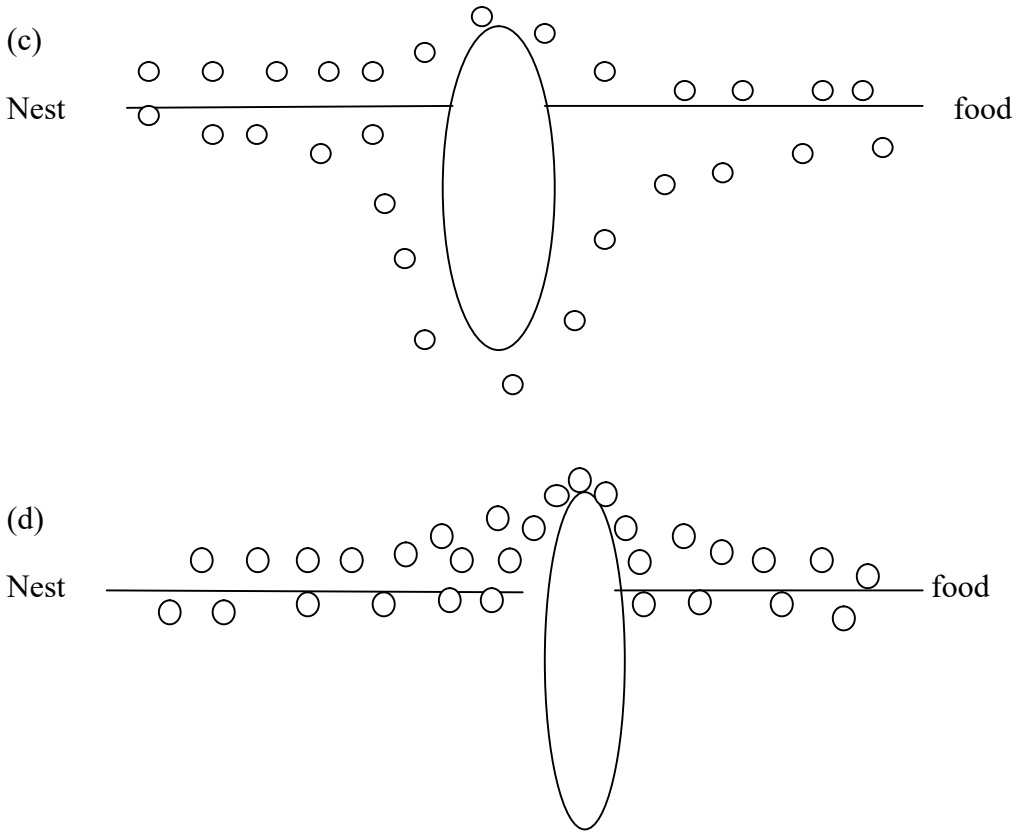


Figure 6 illustrating the behavior of real ant movements.

Figure 6(a) the ants find food and move to food. (b) A barrier prevents the ants from moving. (c) The ants try to move around all paths and leave pheromone. (d) The following ants know which path is shorter by sensing the amount pheromone on paths. Ants which find the shortest path always be the first one return to the nest, thus leaving more hormones on the path. As the shortest path has accumulated more hormones, more and more ants choose this path; in the end all the ants will tend to choose this shortest path.

Now we use figure 7 to descrip the theory of ants finding the shortest path:

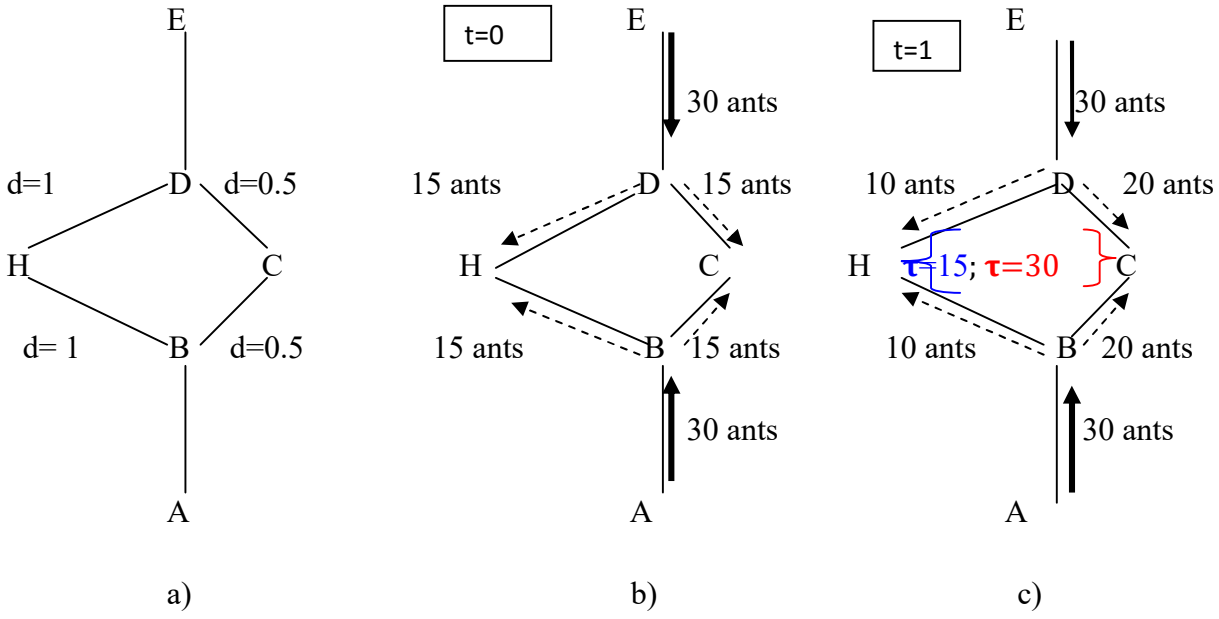


Figure 7 to descrip the theory of ants finding the shortest path.

Shown as Figure 7(a), A is the ants nest, E is the food source, HC is an obstacle, because of the obstacle, the ants can only reach E through H or C from A. Assume that the distance between D and H, H and B, B and D (through C) are 1. At a unit time, there are 30 ants reached B from A, and 30 ants reached D from E. The ant left 1 pheromone after it pass, the ant moves at the speed of 1 per unit time. At the moment t=0, because of there is no pheromone exists on the path: BD, BC, DH, DC, the ants at B and D can randomly select their path to walk. From the statistical point of view, they have the same probability to choose the path BH, BC, DH, DC. Therefore, in average, form the direction B and D, there are 15 ants travel to H, and 15 ants travel to C, as Figure 7(b).

At the moment t=1, there are 30 new ants arrived at B from A. They found that the pheromone concentration on the road leading to H is 15, which is the sum of the pheromone left by the 15 ants choose the direction H from B to D and the 15 ants choose the direction H form D to B. And the pheromone concentration on the road leading to C is 30, thus, the probability of selecting path is changed, as Figure 7(c).Thus, the ants to the C direction will be the twice of ants to the H direction, respectively 20 and 10.The ants starting to reach D from E, will do the same. This process continues, in the end all the ants will choose the shortest.

5.2.2 Applying ACO algorithms to the TSP

Social insects such as ants, termites, wasps, and bees live in almost every land habitat on earth. An ant colony optimization (ACO) technique is an optimization technique to solve combinatorial optimization problems. Swarm intelligence is a relatively new approach to Problem solving that takes inspiration from the social behaviors of insects and of other animals. ACO is one of the most successful techniques in the wider field of swarm intelligence. ACO are multi agent system in which the behavior of each single agent called ant, is inspired by the behavior of real ants. Ants use the environment as a medium of communication. They exchange information indirectly by deposits pheromones, all detailing the status of their work.

Ant colony optimization (ACO) algorithms are heuristic optimization methods that simulate the real ants. ACO is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The algorithm can be applied to solve combinatorial problems such as TSP. The ants release pheromone on the ground while walking from their nest to food and then go back to the nest. The ants move according to the amount of pheromones, the richer the pheromone trail on a path is, the more likely it would be followed by other ants.

ACO algorithms are constructive stochastic search procedures that make use of pheromone model and heuristic information on the problem being tackled in order to probabilistically construct solutions. A pheromone model is a set of so-called pheromone trail parameters. The numerical values of these pheromone trail parameter reflect the search experience of the algorithm. They are used to bias the solution construction over time towards the region of the search space containing high quality solutions.

Artificial ants imitate the behavior of real ants how they forage the food but can solve much more complicated problems than real ants can. Artificial ants probabilistically prefer cities that are connected by edges with a lot of pheromone trail and which are close-by. ACO is a metaheuristic in which a colony of artificial ants cooperates in finding good solutions of optimization problems. Cooperation is a key design component of ACO algorithms because

the choice is to allocate the computational resources to a set of relatively simple agents (artificial ants) that communicate indirectly by pheromone information.

In ACO algorithms ants are simple agents which, in the TSP case, construct tours by moving from city to city on the problem graph. It chooses the city to move to using a probabilistic function both of trail accumulated on edges and of a heuristic value, which was chosen here to be a function of the edges length. Artificial ants try to construct paths containing all nodes exactly once. Initially, m artificial ants are placed on randomly selected cities. When all the ants have completed a tour the ant that made the shortest tour modifies the edges belonging to its tour termed global trail updating by adding an amount of pheromone trail that is inversely proportional to the tour length.

Tour Construction:

In ACO each ant is initially put on a randomly chosen city and has a memory which stores the partial solution it has constructed so far (initially the memory contains only the start city). At each construction step, Ant k applies a probabilistic action choice rule called random proportional rule, to decide which city to visit next. In particular the probability with which ant k currently at city i chooses to go to city j is

$$P_k(i,j) = \begin{cases} \frac{[\tau_{i,j}]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{w \in J_{k(i)}}[\tau_{i,w}(t)]^{\alpha}[\eta(i,w)]^{\beta}} & , \text{if } w \in J_k(i) \\ 0 \qquad \text{otherwise} \end{cases} \tag{5.1}$$

$P_k(i,j)$ Means a probability that ant k moves from node i to node . $J_k(i)$ Is a set of neighboring nodes which ant k will visit; in another word, it is a set of edges (i,w) here the parameter w means the city not visited by ant k. $\tau_{i,j}(t)$ is the amount of pheromone on edge (i,j) and $\eta(i,j)$ weight (distance) of edge (i,j) that means $\eta(i,j) = {}^{1}/_{d_{ij}}$ is a heuristic value. α And β are parameters which are used to control the influence of the pheromone trial and the heuristic information. If the value of α zero, ant k will choose the shortest path according to the distance of all edges in the graph and the closed vertex i more likely to be selected. If the value of β is zero, ant k will choose the shortest path according to the amount of pheromone this will lead to the rapid emergence of a stagnation situation with the corresponding

generation of tours which, in general, are strongly suboptimal (Search stagnation is defined as the situation where all the ants follow the same path and construct the same solution). The setting of α and β will affect the correctness of the solution. Notice that the amount of pheromone corresponding to each edge in the graph is the same initially.

Pheromone Update:

After each ant completes its tour, the pheromone amount on each path will be adjusted according to equation $(1-\rho)$ is the pheromone decay parameter $(0 < \rho < 1)$ where it represents the trail evaporation when the ant chooses a city and decides to move. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails. Pheromone evaporation is given by:

$$\tau_{ij} = (1-\rho)\tau_{ij} \tag{5.2}$$

After all the ants have constructed their tours, the pheromone trails are updated which is done by first lowering the pheromone value on all arcs by a constant factor and then adding pheromones on the arcs they have crossed in their tours. After all the ants complete their tour the pheromone is required to be updated using

$$\tau_{ij}(t+1) = (1-\rho)\tau_{i,j}(t) + \sum_{k=1}^{m_{ij}} \Delta\tau_{i,j}^{k}(t) \tag{5.3}$$

In this formula, $\tau_{ij}(t+1)$ is the amount of pheromone on edge (i,j) after the update step, the parameter ρ is the rate of the pheromone evaporation, and the range of ρ is between 0 and 1. In equation (5.3) m_{ij} means the number of ants which have passed edge (i,j), and $\Delta\tau_{i,j}^{k}(t)$ is the amount pheromone which ant k left on edge (i,j) it has visited; it is defined as follows:

$$\Delta\tau_{i,j}^{k}(t) = \begin{cases} 1/L_{(t)}^{k} & \text{if arc } (i,j) \text{ is used by ant k} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

Where $L_{(t)}^{k}$ is the length of the k^{th} ant's tour. By Equation (5.4), the better the ant's tour is, the more pheromone is received by arcs belonging to the tour. The amount of pheromone on path is larger if the total length of the path is shorter than others. The parameter is defined based on the problem. The spirit residing in equation (5.3) is that the pheromone is increased in short paths and is decreased in the bad ones gradually. In general, arcs that are used by many ants

and which are part of short tours, receive more pheromone and are therefore more likely to be chosen by ants in future iterations of the algorithm.

Example 5.3

Suppose we have a TSP problem with 5 cities. The distance between cities is given by the following matrix.

Table 7 Distance (d) between each pair of cities

Cities/ Cities	1	2	3	4	5
1	0	10	12	11	14
2`	10	0	13	15	8
3	12	13	0	9	14
4	11	15	9	0	16
5	14	8	14	16	0

Determine the best route with minimum total distance by using the ACO.

Step 1:

Parameter Selection:

No of Ants=3

α =1, β=2, Q=1, Number of iteration=5 initially pheromone (τ) distribution for each edge=1
We determine the visibility h between the cities by taking the inverse of the distance $^{1}/_{d}$. We get

$$h = \begin{bmatrix} 0 & 0.1000 & 0.0833 & 0.0909 & 0.0714 \\ 0.1000 & 0 & 0.0769 & 0.0667 & 0.1250 \\ 0.0833 & 0.0769 & 0 & 0.1111 & 0.0714 \\ 0.0909 & 0.0667 & 0.1111 & 0 & 0.0625 \\ 0.0714 & 0.1250 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

Then for each segment between cities we give the initial pheromone value $\tau_0 = 1$, suppose this Value is the same for all segments, namely 1.

Step 2:

Determine the probability of moving from one city to another. In this case city 1 is designated as a departure city. Then from the last city visited we are going back to city 1. As city 1 is chosen as the beginning city 1 then city will be taboo to visit again, so that the level of visibility of the city 1 is made =0.

$$\mathrm{h} = \begin{bmatrix} 0 & 0.1000 & 0.0833 & 0.0909 & 0.0714 \\ 0 & 0 & 0.0769 & 0.0667 & 0.1250 \\ 0 & 0.0769 & 0 & 0.1111 & 0.0714 \\ 0 & 0.0667 & 0.1111 & 0 & 0.0625 \\ 0 & 0.1250 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

Then we calculate the possibility to visit other cities from city 1 using the formula in equation (5.1). Suppose ants 1 from city 1 will choose the next city, then we calculate:

$[\tau_{1,j}]^1[\eta(1,j)]^2$ (from 1 city to all cities)

$1 \times (0.1)^2 = 0.01$(from city 1 to city 2)

$1 \times (0.0833)^2 = 0.0069$ (from city 1 to city 3)

$1 \times (0.0909)^2 = 0083$ (from city 1 to city 4)

$1 \times (0.0714)^2 = 0.0051$ (from city 1 to city 5)

Total number of $[\tau_{1,j}]^1[\eta(1,j)]^2 = 0.0303$

So probability of ant 1 going from city 1 to all cities respectively calculated by the formula

$$\mathrm{P}_1(1,j) = \frac{[\tau_{1,j}]^1[\eta(1,j)]^2}{\sum[\tau_{1,j}]^1[\eta(1,j)]^2}$$

Obtained the probability to go from

City 1 to 2 = $0.01 / 0.0303 = 0.3299$

City 1 to 3 = $0.0069 / 0.0303 = 0.2291$

City 1 to 4 = $0.0083 / 0.0303 = 0.2727$

City 1 to 5 = $0.0051 / 0.0303 = 0.1683$

Step 3:

The selection of next city is based on the maximum probability within the set of possible selection of the cities for next move. Thus the city to visit is city 2.

Since city 2 has been selected then second column of the matrix, the visibility set to 0 so that

$$h = \begin{bmatrix} 0 & 0 & 0.0833 & 0.0909 & 0.0714 \\ 0 & 0 & 0.0769 & 0.0667 & 0.1250 \\ 0 & 0 & 0 & 0.1111 & 0.0714 \\ 0 & 0 & 0.1111 & 0 & 0.0625 \\ 0 & 0 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

From here we calculate again the value of

$[\tau_{2,j}]^1[\eta(2,j)]^2$ (from 2 city to all cities except city 1)

$1 \times (0)^2 = 0$ (from city 2 to city 2)

$1 \times (0.0769)^2 = 0.0059$ (from city 2 to city 3)

$1 \times (0.0667)^2 = 0.0044$ (from city 2 to city 4)

$1 \times (0.1250)^2 = 0.0156$ (from city 2 to city 5)

The total number of $[\tau_{2,j}]^1[\eta(2,j)]^2 = 0.0259$

So probability ant 1 going from city 2 to other cities respectively calculated by the formula

$$P_1(2,j) = \frac{[\tau_{2,j}]^1[\eta(2,j)]^2}{\sum[\tau_{2,j}]^1[\eta(2,j)]^2}$$

Obtained the probability to go from

City 2 to city 2 $0.0/0.0259 = 0.000$

City 2 to city 3 $0.0059/0.0259 = 0.2278$

City 2 to city 4 $0.004/0.0259 = 0.1699$

City 2 to city 5 $0.0156/0.0259 = 0.6023$

Maximum probability is 0.6023 thus city 5 be selected city, since city 5 has been selected then 5^{th} column of the matrix, the visibility set to 0 so that

$$h = \begin{bmatrix} 0 & 0 & 0.0833 & 0.0909 & 0 \\ 0 & 0 & 0.0769 & 0.0667 & 0 \\ 0 & 0 & 0 & 0.1111 & 0 \\ 0 & 0 & 0.1111 & 0 & 0 \\ 0 & 0 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

From here we calculate again the value of

$[\tau_{5,j}]^1[\eta(5,j)]^2$ (from 5 city to all cities except city 1 and2)

$1 \times (0.0714)^2 = 0.0051$ (from city 5 to city 3)

$1 \times (0.0625)^2 = 0.0039$ (from city 5 to city 4)

$1 \times (0)^2 = 0.00$ (from city 5 to city 5)

The total number of $[\tau_{2,j}]^1[\eta(2,j)]^2 = 0.009$

So probability ant 1 going from city 5 to other cities respectively calculated by the formula

$$P_1(2,j) = \frac{[\tau_{2,j}]^1[\eta(2,j)]^2}{\sum[\tau_{2,j}]^1[\eta(2,j)]^2}$$

Obtained the probability to go from

City 5 to city 3 $0.0051/0.009 = 0.5667$

City 5 to city 4 $0.0039/0.009 = 0.4333$

Maximum probability is 0.5667 thus city 3 be selected city and finally the reaming unvisited city is city 4 then final selected city is city 4 and back to the starting city 1.

Ant 1 path: 1-2-5 -3-4-1 total distance of route is 52

By using the same process we can find Ant 2 path. In this case city 2 is designated as a departure city. Then from the last city visited we are going back to city 2. As city 2 is chosen as the beginning city 2 then city will be taboo to visit again, so that the level of visibility of the city 2 is made =0.

$$h = \begin{bmatrix} 0 & 0 & 0.0833 & 0.0909 & 0.0714 \\ 0.1000 & 0 & 0.0769 & 0.0667 & 0.1250 \\ 0.0833 & 0 & 0 & 0.1111 & 0.0714 \\ 0.0909 & 0 & 0.1111 & 0 & 0.0625 \\ 0.0714 & 0 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

Then we calculate the possibility to visit other cities from city 2 using the formula in equation (3.1). Suppose ants 2 from city 2 will choose the next city, then we calculate:

$[\tau(2,j)]^1[\eta(2,j)]^2$ (from city 2 to all cities)

$1 \times (0.1)^2 = 0.01$ (from city 2 to city 1)

$1 \times (0.0769)^2 = 0.0059$ (from city 2 to city 3)

$1 \times (0.0667)^2 = 0.0044$ (from city 2 to city 4)

$1 \times (0.1250)^2 = 0.0156$ (from city 2 to city 5)

The total number of $[\tau_{2,j}]^1[\eta(2,j)]^2 = 0.0359$

So probability of ant 2 going from city 2 to all cities respectively calculated by the formula

$$P_2(2,j) = \frac{[\tau_{2,j}]^1[\eta(2,j)]^2}{\sum[\tau_{2,j}]^1[\eta(2,j)]^2}$$

Obtained the probability to go from

City 2 to 1 = 0.01 / 0.0359 = 0.2786

City 2 to 3 = 0.0059 / 0.0359 = 0.1643

City 2 to 4 = 0.0044 / 0.0359 = 0.1226

City 2 to 5 = 0.0156 / 0.0359 = 0.4345

The selection of next city is based on the maximum probability then city 5 is selected city as city 5 has been selected then 5^{th} column of the matrix of the visibility set to 0 so that

$$h = \begin{bmatrix} 0 & 0 & 0.0833 & 0.0909 & 0 \\ 0.1000 & 0 & 0.0769 & 0.0667 & 0 \\ 0.0833 & 0 & 0 & 0.1111 & 0 \\ 0.0909 & 0 & 0.1111 & 0 & 0 \\ 0.0714 & 0 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

From here we calculate again the value of τ

$[\tau_{5,j}]^1[\eta(5,j)]^2$ (from city 5 to all cities except city 2)

$1 \times (0.0714)^2 = 0.0051$ (from city 5 to city 1)

$1 \times (0.0714)^2 = 0.0051$ (from city 5 to city 3)

$1 \times (0.0625)^2 = 0.0039$ (from city 5 to city 4)

$1 \times (0.0)^2 = 0.0000$ (from city 5 to city 5)

The total number of $[\tau_{5,j}]^1[\eta(5,j)]^2 = 0.0141$

So probability of ant 2 going from city 2 to all cities respectively calculated by the formula

$$P_2(5,j) = \frac{[\tau_{5,j}]^1[\eta(5,j)]^2}{\sum[\tau_{5,j}]^1[\eta(5,j)]^2}$$

Obtained the probability to go from

City 5 to 1 = $0.0051 / 0.0141 = 0.3617$

City 5 to 3 = $0.0051 / 0.0141 = 0.3617$

City 5 to 4 = $0.0039 / 0.0141 = 0.2766$

City 5 to 5 = $0.0 / 0.0141 = 0.0000$

Selection of next city is based on the maximum probability thus city 1 is selected city then first column of visibility is made =0

$$h = \begin{bmatrix} 0 & 0 & 0.0833 & 0.0909 & 0 \\ 0 & 0 & 0.0769 & 0.0667 & 0 \\ 0 & 0 & 0 & 0.1111 & 0 \\ 0 & 0 & 0.1111 & 0 & 0 \\ 0 & 0 & 0.0714 & 0.0625 & 0 \end{bmatrix}$$

From here we calculate again the value of τ

$[\tau_{1,j}]^1[\eta(1,j)]^2$ (from city 1 to all cities except city 2 and city 5)

$1 \times (0.0833)^2 = 0.0069$ (from city 1 to city 3)

$1 \times (0.0909)^2 = 0.0083$ (from city 1 to city 4)

The total number of $[\tau_{1,j}]^1[\eta(1,j)]^2 = 0.0152$

So probability of ant 2 going from city 1 to all cities respectively calculated by the formula

$$P_2(1,j) = \frac{[\tau_{1,j}]^1[\eta(1,j)]^2}{\sum[\tau_{1,j}]^1[\eta(1,j)]^2}$$

Obtained the probability to go from

City 1 to 3 = $0.0069 / 0.0152 = 0.4539$

City 1 to 4 = $0.0083 / 0.0152 = 0.5461$

Based on maximum probability city 4 is selected and finally the only unvisited city is city 3 then city 3 selected and back to city 2.

Ant 2 path: 2-5-1-4-3-2 with total distance of route is 55

By using this process we obtained Ant 3 path: 3-4-5-2-1-3 with total distance of route is 55

The calculation we did in the first iteration for all 3 ants we obtained the following:

Ant 1 path: 1-2-5 -3-4-1

Ant 2 path: 2-5-1-4-3-2

Ant 3 path: 3-4-5-2-1-3

With a total distance of each route is 52, 55 and 55. The total distance is used to update the Pheromone level τ using equation (5.3)

Step 4:

All ants will provide additional pheromone to the pheromone matrix according to the route passed through. Ants will provide additional pheromone by 1/total distance i.e., $\frac{1}{52} = 0.0192$; $\frac{1}{55} = 0.0181$ and $\frac{1}{55} = 0.0181$. In addition, evaporation will occur at $(1 - 0.5)\,x1 = 0.5$, where ρ, evaporation coefficient $= 0.5$. So the total pheromone becomes 0.5192, 0.5181 and 0.5181. In other segments which are not passed by ants, only evaporation will occur without the addition of pheromone.

$$\tau = \begin{bmatrix} 0.5 & 0.5192 & 0.5181 & 0.5181 & 0.5 \\ 0.5181 & 0.5 & 0.5 & 0.5 & 0.5373 \\ 0.5 & 0.5181 & 0.5 & 0.5373 & 0.5 \\ 0.5192 & 0.5 & 0.5181 & 0.5 & 0.5181 \\ 0.5181 & 0.5181 & 0.5192 & 0.5 & 0.5 \end{bmatrix}$$

Then in the next iteration we repeat from step 2 until the maximum number of iterations (2) obtained the solution that is: Ant 1 path: 1-2-5 -3-4-1 with total distance 52

Ant 2 path: 2-5-1-4-3-2 with total distance 55

Ant 3 path: 3-4-5-2-1-3 with total distance 55

Best path: 1-2-5 -3-4-1 with total distance 52

Example 5.4

We can use ACO algorithm on solving a well-known problem instance of classical city dataset from TSPLIB (Reinelt, 1995), in which the optimal solution is known. There are three symmetric travelling salesman problems called burma14, bay29 and dantzig42. The burma14, bay29 and dantzig42 have 14 and 29 cities, respectively.

We set the ACO parameters used in each problem according to many experiments. We taken from (Aggarwal and Saroj, 2012) that means pheromone parameter (α) =1 number of ants m= number of city, heuristic factor β=2, rate of the pheromone evaporation (ρ) =0.65, the amount of pheromone Q =10, Maxiter=500

Table 8 ACO parameter setting for TSP

Problem	pheromone Parameter (α)	number of ants m	heuristic factor β	Rate of pheromone evaporation (ρ)	the amount of pheromone Q	Maxiter
burma14	1	14	2	0.65	10	500
bayg29	1	29	2	0.65	10	500
dantzig42	1	42	2	0.65	10	500

Using the parameters given above in table 8 and Run by using MATLAB we obtain the following solutions (results) and shortest path for each problem.

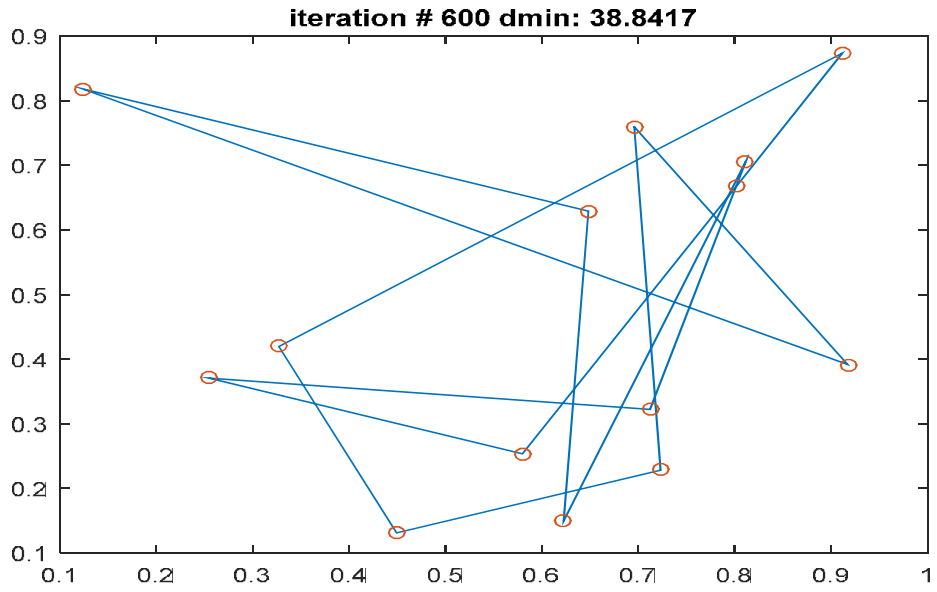


Figure 8 Optimal tours for burma14 with ACO algorithm

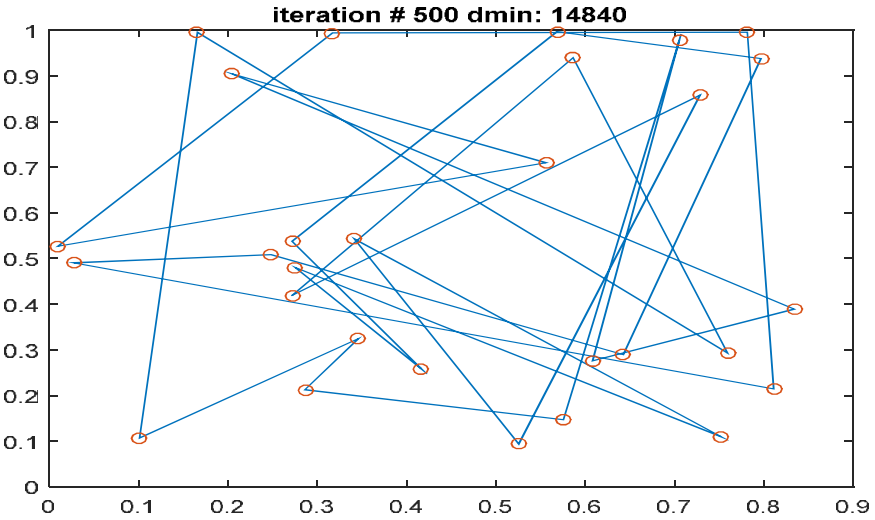


Figure 9 Optimal tours for bay29 with ACO algorithm

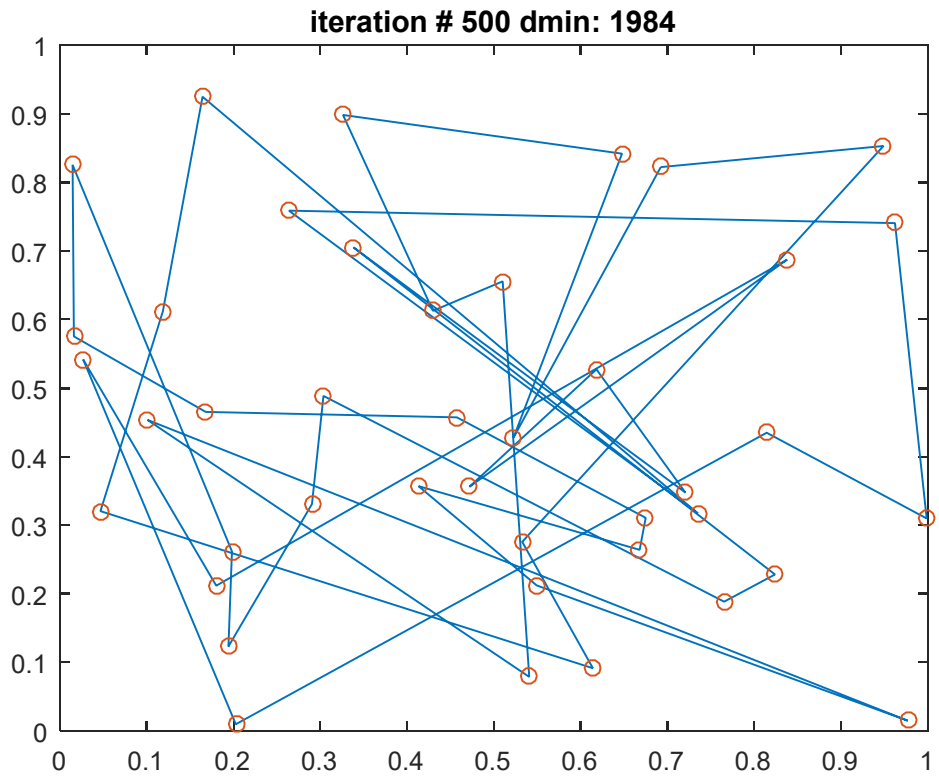


Figure 10 Optimal tours for dantzig42 with ACO algorithm

Table 9 Quality of solution (Best solution) comparisons for TSPLIB problem

Problem	Known optimal solution(value)	approximate solution by using ACO
burma14	31	38.8417
bayg29	1610	14840
dantzig42	699	1984

The purpose of TSP experiments is to evaluate and to test the effectiveness of the ACO algorithm. To indicate the quality of the returned solution, the relative error is calculated based on equation (4.1). The percentage of relative error of the best solution obtained with respect to the known optimal solution is presented in Table 10.

Table 10 Percentage of relative error for all TSP instances by using ACO

Problem	ACO (%)
burma14	25.29
bayg29	821.73
dantzig42	183.83

Genetic Algorithm and Ant Colony Optimization are used to solve travelling salesman problem with high acceptable performance therefore we compare between them and determine when we can use one as better than other. According to the results of table 6 and table 10 demonstrates that GA is better than ACO with reason of relative error, where the results of GA algorithm to give minimum error than ACO. Therefore we discussed that GA method is better methods for solving large problem size travelling salesman problem. The programming is performed using MATLAB version 8.7 R2015a on windows 10 to execute the program. The MATLAB code of GA was used linear order crossover and roulette wheel selection and ACO was used pheromone parameter $\alpha = 1$ and number of ants equals number of cities. Stopping criteria include the maximum number of iterations. After running the algorithms for the maximum number of generations, GA and ACO code returns route with minimum distance, is the final solution. The computational experiment of three problems, are solved, and the results show that the GA algorithm achieves best solution when problem size is large comparing to ACO.

6. SUMMARY, CONCLUSION AND RECOMMENDATION

6.1. Summary

Combinatorial optimization means searching for an optimal solution in a finite or countable infinite set of potential solutions. The traveling salesman problem (TSP) is a typical example of a very hard combinatorial optimization problem. TSP being an optimization problem is used to find shortest path of a given set of cities, given a set of cities and the cost of travel (or distance) between each possible pairs. TSP is difficult to solve efficiently by conventional exact optimization techniques when its scale is very large. Therefore it is necessary to used approximation methods for solving TSP problems such as genetic algorithm, ant colony optimization. The proposed methods that are genetic algorithm and ant colony optimization were suitable for solving travelling salesman problem. These two methods were solved travelling salesman problem by using MATLAB. Genetic algorithm and ACO are a type of heuristic optimization algorithms, meaning they are used to find the optimal solution(s) to a given computational problem.

The general aims of this project was to solve travelling salesman problem by using genetic algorithm and ant colony optimization and to compared the performance of the two methods one with the other based on the relative error with regards to known optimal value. The natural evolution process was always used by genetic Algorithm to solve the problems for generated successively shorter feasible tours by using selection, crossover and mutation operator of travelling salesman problem. The foraging behavior of real ants was used by ant colony optimization for generated successively shorter feasible tours of travelling salesman problem Some examples have been given to indicate the performance of the methods and from the comparison made among the given methods, genetic algorithm was better performing methods than ant colony optimization algorithm for the large size problem.

6.2. Conclusion

In this project, we have discussed genetic algorithm and ant colony optimization methods to solve travelling salesman problem by using MATLAB. Various techniques of genetic algorithm and ant colony optimization have been discussed in this paper to study travelling salesman problem. The methods are examined on different examples which were solved by

these two methods and we presented a comparative study among GA and ACO for some TSPLIB instances. We compared GA and ACO based on relative error for each three TSP instances such as burma14, bay29 and dantzig42. The aims of these comparisons are to determine which method takes minimum distance. According to table 6 and table 10 we described that GA was gives better result (minimum distance) for large problem size TSP than ACO based on relative error with regard to known optimal value.

6.3. Recommendation

Based on solving travelling salesman problem by using genetic algorithm and ant colony optimization methods the following basic recommendations are suggested.

- Individuals interested to work develop genetic algorithm and ant colony optimization for precedence constrained asymmetric travelling salesman problem and compare the two methods by using MATLAB
- Future work could be tailored towards simulation on system with higher memory. However, other optimization techniques such as Simulated Annealing, Particle Swarm Optimization could be used for evaluation and conventional programming languages such as C++, java, and Python could also be used as a platform to check results.

7. REFERENCES

Aggarwal, M. and Saroj, D. 2012. Compute travelling salesman problem using ant colony optimization. *International Journal of Computing and Business Research ISSN:* 2229-6166.

Ahmed, Z.H. 2010. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics & Bioinformatics*, 3(6):96.

Angel, R.D., Caudle, W.L., Noonan, R. and Whinston, A.N.D.A.1972. Computer-assisted school bus scheduling. *Management Science*, 18(6), pp.B-279.

Aranganayaki, A. 2011. Reduce total distance and time using genetic algorithm in Traveling Salesman Problem. Optimization, 6(4):4.

Arora, K. and Arora, M. 2016. Better result for solving TSP: GA versus ACO. Int. *J. Adv. Res. Comput. Sci. Softw. Eng*, 6:219-24

Bagchi, T.P., Gupta, J.N. and Sriskandarajah, C. 2006. A review of TSP based approaches for flowshop scheduling. *European Journal of Operational Research*, 169(3):816-854.

Bajpai, A. and Yadav, R. 2015. Ant colony optimization (ACO) for the traveling salesman problem (TSP) using partitioning. *Int. J. Sci. Technol. Res*, 4(09):376-381.

Biggs, N., Lloyd, E.K. and Wilson, R.J. 1986. *Graph Theory*, 1736-1936. Oxford University Press.

Blum, C. 2005. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353-373.

Brezina Jr, I. and Čičková, Z. 2011. Solving the travelling salesman problem using the ant colony optimization. *Management Information Systems*, 6(4):10-14.

Chong, Y. N. 2001. Heuristic algorithms for routing problems. PhD Thesis, Curtin Univ. of technology.

Chudasama, C., Shah, S.M. and Panchal, M. 2011. Comparison of parent's selection methods of genetic algorithm for TSP. *In International Conference on Computer Communication and Networks Proceedings*: 85-87.

Dwivedi, V., Chauhan, T., Saxena, S. and Agrawal, P. 2012. Travelling salesman problem using genetic algorithm. *IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues*, 1:25.

Goldbarg, E.F.G. Souza, G.R. & Goldbarg, M.C. 2006. Particle swarm for the traveling salesman problem. Lecture Notes in Computer Science, 3906:99-110.

Gupta, S. and Panwar, P. 2013. Solving travelling salesman problem using genetic algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(6):376-380.

Hingrajiya, K.H., Gupta, R.K. and Chandel, G.S. 2012. An ant colony optimization algorithm for solving travelling salesman problem. *International Journal of Scientific and Research Publications*, 2(8):1-6.

Hlaing, Z.C.S.S. and Khine, M.A. 2011. An ant colony optimization algorithm for solving traveling salesman problem. *In International Conference on Information Communication and Management*, 16: 54-59.

Huilian, F.A.N. 2010. Discrete particle swarm optimization for TSP based on neighborhood. *Journal of Computational Information Systems*, 6(10):3407-3414.

Keller, M. T. 2004. Knot theory: history and applications with a connection to graph theory, Thesis, Dept of Maths, North Dakota State University.

Kinnear, K. E. 1994. A Perspective on the Work in this Book. *Advances in Genetic Programming*, 3-17. Cambridge: MIT Press.

Kumar, N. 2012. A Genetic Algorithm Approach to study Travelling Salesman Problem. *Journal of Global Research in Computer Science*, 3(3):33-37.

Li, B., Wang, L. and Song, W. 2008. Ant colony optimization for the traveling salesman problem based on ants with memory. *In Fourth International Conference on Natural Computation*: 496-501.

Liang, Y.C., Shi, X.H., Lee, H.P., Lu, C. and Wang, Q.X. 2007. Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information processing letters*, 103(5), pp.169-176.

Lin S., Kernighan B.W.1973. An effective heuristic algorithm for travelling salesman problem. *Operations Research*, 498–516.

Machado, T.R. & Lopes, H.S. 2005. A hybrid particle swarm optimization model for the traveling salesman problem. In Natural Computing Algorithms, 255-258.

Matai, R., Singh, S. and Mittal, M.L. 2010. Traveling salesman problem: an overview of applications, formulations, and solution approaches. *In Traveling Salesman Problem, Theory and Applications*. InTech.

Miliotis, P. 1978. Using cutting planes to solve the symmetric travelling salesman problem. *Mathematical programming*, 15(1):177-188.

Mitchell, M. 1995. Genetic Algorithms: An Overview. Complexity, 1(1):31-39.

Mohd Razali, N. 2014. Genetic algorithm for process sequencing modelled as the travelling salesman problem with precedence constraints (Doctoral dissertation, Dublin City University).

Rani, K. and Kumar, V. 2014. Solving travelling salesman problem using genetic algorithm based on heuristic crossover and mutation operator. *International Journal of Research in Engineering & Technology*, 2(2):27-34.

Reinelt, G. 1995. Tsplib95. *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen* (*IWR*), *Heidelberg*, 338.

Sundaram, R.K. 1996. *A first course in optimization theory*. Cambridge university press.

Svestka, J.A. and Huckfeldt, V.E. 1973. Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, 19(7):790-799.

Taiwo, O.S., Josiah, O., Taiwo, A., Dkhrullahi,S. and Sade,O.K. 2013. Implementation of Heuristics for Solving Travelling Salesman Problem Using Nearest Neighbour and Nearest Insertion Approaches. *International Journal of Advance Research*, 1(3):139-155.

8. APPENDIX

A MATLAB code for Genetic Algorithm and Ant Colony Optimization for solving Travelling Salesman Problem.

A. MATLAB code for Genetic Algorithm for solving Travelling Salesman Problem .

```
function [opt_rte, ave, best] =gg(~,~,~,~,~)
pop_size;
Pc;
Pm;
ngener;
Ncity;
city_location;
N=size (city_location, 1);
a=meshgrid (1: N);
city_distance=reshape(sqrt(sum((city_location(a,:)-city_location(a',:)).^2,2)),N,N);
%generate initial random chromosome
ngenes=N; %number of genes in a chromosome
Chrom=zeros (pop_size, ngenes);
for k=1:pop_size
Chrom (k, :) =randperm (ngenes);
end
%fitness evaluation in the initial population
ObjV=zeros (1, pop_size);
for p=1:pop_size
d=city_distance (chrom (p, ngenes), chrom (p, 1)); % closed path-same %starting and ending point
for k=2: ngenes
d=d+city_distance (chrom (p, k-1), chrom (p, k));
end
ObjV (p) =d;
end
```

```
best=min (ObjV); %minimum distance in the initial population
ave=mean (ObjV); %average distance in the initial population
for i=1:ngener
%Parent selection-proportional roulette wheel proportional;
fitness= (1./ObjV)';
numsel=round (pop_size*0.9);
cumfit=cumsum (fitness);
chance=cumfit(pop_size).*rand(numsel,1);
Mf=cumfit(:,ones(1,numsel));
Mt=chance(:,ones(1,pop_size))';
[selind,idx]=find(Mt < Mf & [zeros(1,numsel); Mf(1:pop_size -1,:)] <= Mt);
newchrom=chrom(selind,:) ;
%Crossover mechanism-Linear order
%Mutation mechanism-Inversion (flip left to right)
for q=1:numsel
   if rand(1)<Pm
points=sort((round(rand(floor(numsel/2),1).*(ngenes-1))+1)');
newchrom(q,:)=[newchrom(q,1:points(1)),...
fliplr(newchrom(q,points(1)+1:points(2))),...
newchrom(q,points(2)+1:ngenes)];
end
end
if pop_size-numsel %preserving a part of the parent chromosome population
[answ,Index]=sort(fitness); %sort the fitness of parent chromosome & preserving the
% best nind-numsel chromosomes
chrom=[chrom(Index(numsel+1:pop_size),:);newchrom];
else %replacing the entire parent chromosome population with a new one
chrom=newchrom;
end
%Fitness Evaluation
Fitness Evaluation;
```

```
best=[best min(ObjV)]; %minimum distance in every generation
ave=[ave mean(ObjV)]; %average distance in every generation
[min_dist index]=min(ObjV);
opt_rte=chrom(index,:); %optimal route
[c d]=min(best)
subplot(1,2,1);
 rtes=opt_rte([1:ngenes 1])
 plot(city_location(rtes,1),city_location(rtes,2),'r.-')
 title(['Generation # ',num2str(d),' Distance: ',num2str(min_dist)] )
```

B. MATLAB code for Ant Colony Optimization for solving Travelling Salesman Problem.

```
Ncity; % number of cities on tour
Nants=Ncity; % number of ants=number of cities
maxit; % max number of iterations
α; β; ρ; Q;
city_location;
N=size (city_location, 1);
a=meshgrid(1:N);
%distance between cities
for ic=1:Ncity
for id=1:Ncity
city_distance=reshape(sqrt(sum((city_location(a,:)-city_location(a',:)).^2,2)),N,N);
end % id
end %ic
vis=1./city_distance; % visibility equals inverse  of distance
phmone=.1*ones(Ncity,Ncity);% initialized pheromones between cities
% initialize tours
 for ic=1:Nants
 tour(ic,:)=randperm(Ncity);
 end % ic
```

```
tour(:,Ncity+1)=tour(:,1); % tour ends on city it starts with
for it=1:maxit
% find the city tour for each ant
% st is the current city
% nxt contains the remaining cities to be visited
for ia=1:Nants
for iq=2:Ncity-1
[iq tour(ia,:)];
st=tour(ia,iq-1);
nxt=tour(ia,iq:Ncity);
prob=((phmone(st,nxt).^a).*(vis(st,nxt).^b))/ sum((phmone(st,nxt).^a).*(vis(st,nxt).^b));
rcity=rand;
for iz=1:length(prob)
if rcity<sum(prob(1:iz))
newcity=iq-1+iz; % next city to be visited
%end % if
%end % iz
temp=tour (ia,newcity); % puts the new city
% selected next in line
tour(ia,newcity)=tour(ia,iq);
tour(ia,iq)=temp;
end
end
end
end
% calculate the length of each tour and pheromone distribution
phtemp=zeros(Ncity,Ncity);
for ic=1:Nants
dist(ic,1)=0;
for id=1:Ncity
   dist(ic,1)=dist(ic)+city_distance(tour(ic,id),tour(ic,id+1));
```

```
phtemp(tour(ic,id),tour(ic,id+1))=Q/dist(ic,1);
end % id
end % ic
[dmin,ind]=min(dist);
if dmin<dbest
dbest=dmin;
end % if
% pheromone for elite path
ph1=zeros(Ncity,Ncity);
for id=1:Ncity
ph1(tour(ind,id),tour(ind,id+1))=Q/dbest;
end % id
% update pheromone trails
phmone=(1-ρ)*phmone+phtemp+e*ph1;
dd(it,:)=[dbest dmin];
[it dmin dbest];
[tour,dist];
figure(1)
```