T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM TEKNOLOJİLERİ MÜHENDİSLİĞİ ANABİLİM DALI

SINGLE OBJECTIVE HEURISTIC OPTIMIZATION TECHNIQUES

SOLVING TRAVELLING SALESMAN PROBLEM USING ARTIFICIAL BEE COLONY OPTIMIZATION ALGORITHM

MD AL AMIN HOSSAIN

218264001009

JANUARY-2023

KONYA

ABSTRACT

It is an NP-hard issue to estimate the shortest path between the starting point and each member of a collection of places. This is also called the Traveling Salesman Problem (TSP), a particular multiobjective problem researched in computational science and mathematics applications. It has been demonstrated that evolutionary algorithms outperform conventional approaches like avoidance trapping in local minimum areas for solving NP-complete problems like TSP. The Artificial Bee Colony (ABC) is a brand-new swarm-based optimization method that draws inspiration from honey bees' grazing activities. The traveling salesman problems (TSP) are solved using an artificial bee colony (ABC) algorithm in this work. The algorithms provide precise solutions, and the problem is also put through a simulation process. The outcomes demonstrate that the algorithm is capable of finding ideal or imperfect solutions with speed and efficiency.

CONTENTS

1. INTRODUCTION

An optimization problem is one in which the best element is chosen from a variety of possible options. In many of these circumstances, a thorough search is not possible [1]. It has significant uses in a variety of disciplines, including software engineering, mathematics, intelligent systems, and machine learning. The majority of these issues are regarded as NP-hard, meaning that they cannot be resolved efficiently in a polynomial amount of time [1]. The traveling salesman problem (TSP), the least spanning tree problem, etc. are a few examples of typical optimization issues [1].

The most important issue in the modern era is optimization, and a lot of research has been conducted to find a solution. Many studies have already been performed on GA, ABC, and the blending of different evolutionary algorithms. There aren't many pieces of literature that contrast performance assessment and recommend the optimal approach for particular issues [2]. TSP claims that for one salesman who wants to travel to various cities and is informed of their distances. All of them must be visited by the traveling salesman, who does not travel far. The goal is to choose a series of cities that will reduce the distance traveled [3 ,4].

There have been numerous strategies for solving TSP in recent years, which can be classified as precise and approximative strategies [5,6]. Researchers must use approximate methods that locate solutions that are close to optimal in a reasonable amount of time rather than exact methods that promise to locate the optimal solution in an exponential amount of time for real-world problems because exact methods can only be used for very small instances [7].

In this study, we apply the actual ABC method to the traveling salesman problem (TSP). This paper details our efforts to solve the "Traveling Salesman Problem (TSP)" using the Bee Colony Optimization (BCO) model, which is based on nectar gathering in bee colonies. The use of a novel bee colony algorithm for the dynamic allocation of Internet servers [8] is the inspiration for this study.

The paper is structured as follows: A detailed rundown of the Artificial Bee Colony (ABC) algorithm is provided in Section 2. The TSP is introduced in Section 3. The proposed methodology, known as the ABC approach for TSP, is described in Section 4. The experimental results are described in Section 5, and the conclusion is stated in Section 6.

2. ARTIFICIAL BEE COLONY(ABC) ALGORITHM

2.1. Overview

The artificial bee colony optimization algorithm is a population-based swarm intelligence metaheuristic algorithm. The ABC algorithm was inspired by the social interactions of honey populations. Dervis Karaboga introduced the artificial bee colony in 2005 [9]. The ABC optimization technique is used in both local and global searches. When looking for rich artificial food sources in ABC, bees, or agents, look for materials that describe good solutions. Certain features of ABC are:

- Nectar exploration
- Waggle dance
- Food foraging
- Division of bees
- Mating during flight.

A bee colony is based on three different models:

i. Food foraging
ii. Nest Site Search
iii. Marriage in the Bee Colony

Natural bees are excellent at looking for food sources. Any bee that discovers food will dance to let the other bees know. This informs the surrounding bees of the size and position of the food supply. This assists in pointing other bees in their quest for food in the right direction. These bees have the power to draw in a lot of other bees and continue to find a food source [7].

Unlike other optimization algorithms like the genetic algorithm (GA) and evolutionary algorithms, ABC has minimally adjusted parameters. Consequently, it is a quick and effective optimization procedure [10]. The effectiveness of ABC has also been demonstrated to be supreme to that of other well-known optimization algorithms, including the genetic algorithm (GA), differential evolution (DE), evolutionary strategies (ES), and particle swarm optimization (PSO) methods [10]. However, it has been discovered that ABC has a few drawbacks, including a poor accuracy rate [11]and rate of convergence [12]. Comparing the ABC optimization approach to the PSO algorithm, better outcomes are obtained.

2.2. Types of Bees

A swarm of honey bees is a group that works together to complete tasks successfully [13]. In the ABC algorithm, there are three types of bees:

- Employed bees: The employed bees explore for food nearby the food source in their memories while also informing the onlooker (observer) bees about these sources of food.
- Onlooker bees: From the food sources discovered by the employed bees, the onlooker bees frequently choose the best ones. The chance that the observer bees will choose the food source with greater quality (fitness) is much higher than the chance that they will choose the one with lower quality.
- Scout bees: The scout bees are derived from a small number of employed bees that leave their current food sources and look for new ones.

2.3. Purpose Of ABC

ABC optimization algorithm to solve different engineering optimization problems, numerical problems. Such as,

- Bee System: Genetics Problems.
- Bee Hive: Routing Protocols.
- Honey Bee Marriage: Cluster Analysis.
- Bee Colony Optimization: Travelling Salesman Problems (TSP), Vehicle Routing Problems, Ride Matching Problems, and Job Scheduling Problems.
- Artificial Bee Colony Optimization: Engineering Problems, Numerical Optimization.

The objectives of ABC algorithm are:

- To solve an optimization problem
- To find the best food source
- Pretty good at finding the optimum solutions

2.4. How Do Bees Communicate

Bees communicate through waggle dance, where waggle dance describes the fitness probability of food sources. If food is directly in line with the sun, the bee will

dance straight upwards. On the other hand, if food is at an angle to the sun, the dancing bee changes direction accordingly. Remarks of this process:

i. The direction of flower patches- angle between the sun and the patch
ii. The distance from the hive- duration of dance
iii. The quality rating(fitness)- frequency of the dance

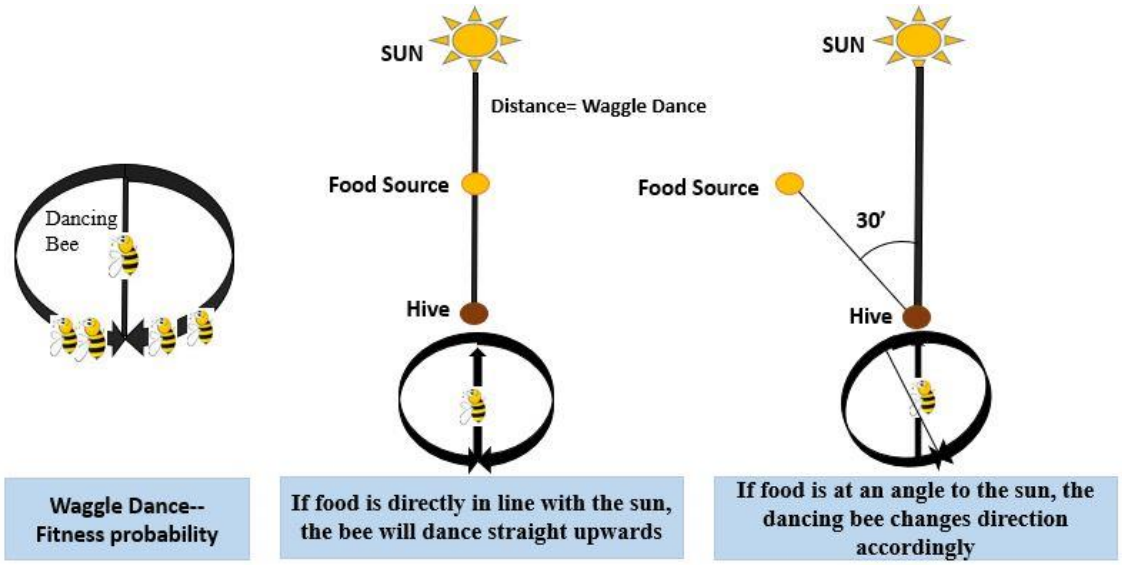

Figure 2.1: Bees communication process

2.5. How Bees Optimize Food Search

The location of a food source symbolizes a potential optimization solution in ABC, a community algorithm, and the nectar quantity of a food source refers to the quality (fitness) of the corresponding solution [14]. Bees optimize food search approach stepwise depicted in below:

- Bees from better-quality food sources perform the better dance. Onlooker bees choose to follow better waggle dancer bee routes.
- In this way, better food sources and routes get reinforced with positive findings.
- Refined new sources.
- Some scouts return to inform other bees of better food sources via waggle dance.
- In this way, more bees go to the next best food source.
- This mechanism helps them escape the local best value of 2 and move to the global best value of 7.

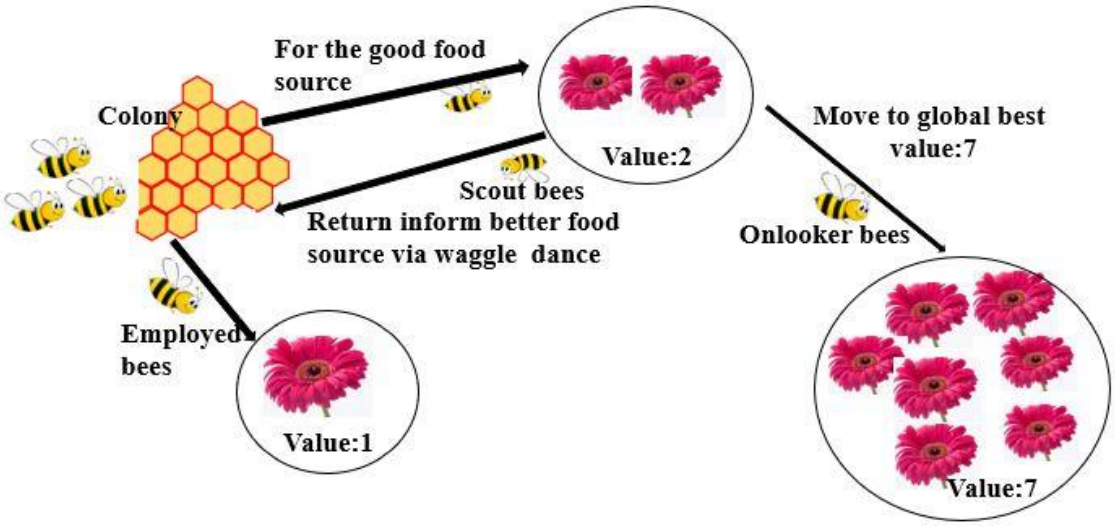

Figure 2.2: Bees food search optimization techniques

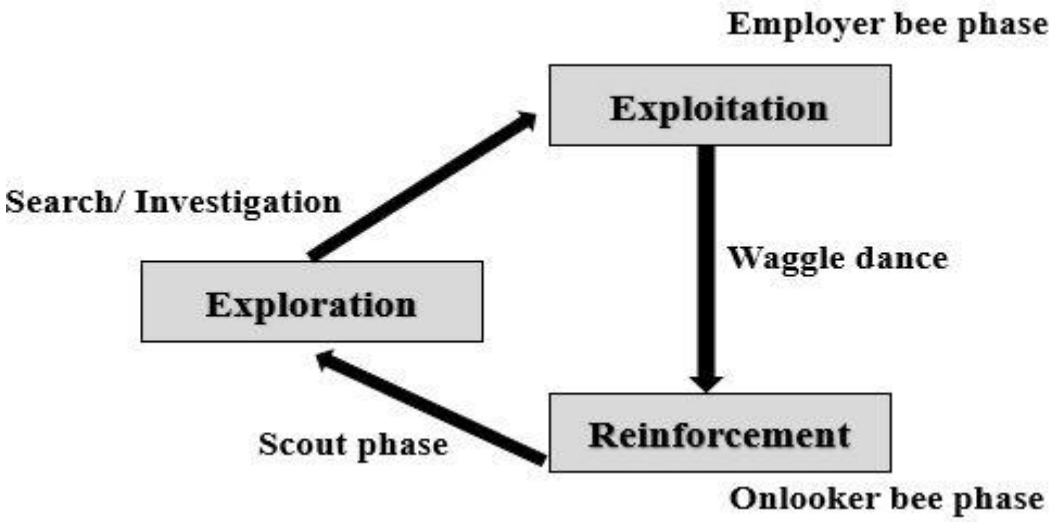

Figure 2.3: Bee foraging(search) cycle

2.6. Illustration Of ABC Algorithm

2.6.1. Phases of ABC Approach

The initialization, employed bees, onlooker bees, scout bees, and termination phases, which together comprise a total of fourteen steps or tasks, are the five key phases that compensate the ABC operating principle. Employed bees, onlooker bees, and scout

bee phases are the three performance-determining phases of ABC, whereas the other two are assisting phases. The next paragraphs cover the stages' specifics.

1. Initialization Phase: Food sources are a representation of a population's potential response to an issue in the ABC algorithm. They are generated at random. The consumer population size values form the basis for the population's initialization. The employed bees are subsequently given access to these food sources. Next, using the equation presented in, the nectar amounts that represent each food source's fitness value are determined [15].

$$\boldsymbol{fit_i} = \begin{cases} \frac{1}{1+f_i} & \boldsymbol{if} \quad \boldsymbol{f_i \geq 0}, \\ \boldsymbol{1 + f_{abs}(f_i)} & \boldsymbol{if} \quad \boldsymbol{f_i \leq 0}, \end{cases} \tag{1}$$

Where f_i is objective function value of *i*th food source.

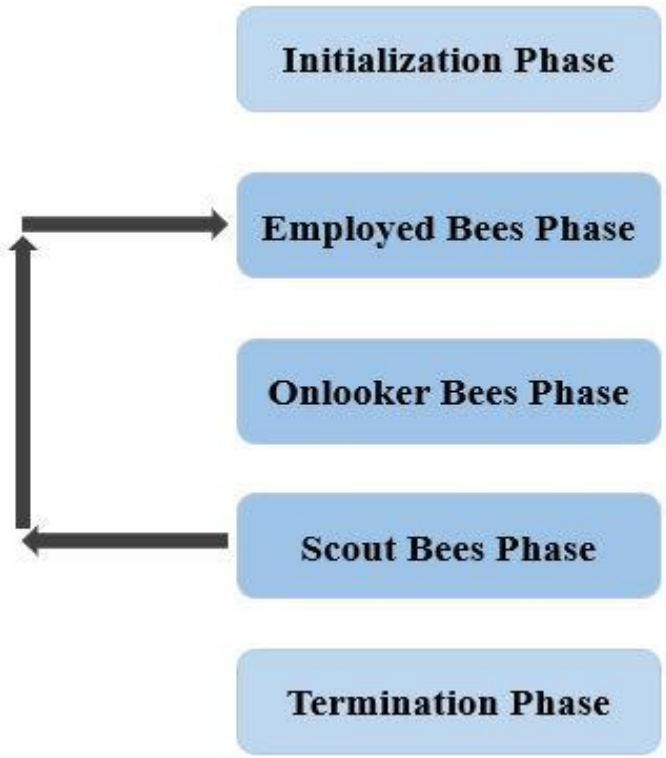

Figure 2.4: Phases of ABC

2. Employed Bees Phase: Employed bees investigate the area around the food sources that have been allocated to them during this phase, and they upgrade the sources of food utilizing the mutation formula provided by

$$\boldsymbol{v_{ij} = x_{ij} - \phi_{ij}\,[x_{ij} - x_{kj}]}, \tag{2}$$

where $k \in [1, 2,..., SN]$ is the number of food sources, v_{ij} is the candidate solution of the food sources, x_{ij} is the *j*th dimension of the ith food sources, and x_{kj} is the *k*th food sources that are randomly selected from a neighborhood of *i*th food sources. Food sources with the subscripts *k* and i are reciprocally unique. For the equation, $j \in [1,2,...,D]$, where *D* is the dimension of the search space and ϕ_{ij} is the control parameter that represents a random number from [-1, 1], inclusive. *K* and *j* are both chosen at random.

3. Onlooker Bees Phase: The spectator bees do not upgrade all of the possible healthier food sources that the employed bees have shared with them during this stage. Among all of the food sources exchanged with them, they use a fitness-proportion screening system to select a few that are more fit. The algorithm has converged quickly as a result of observer bees taking advantage of the food sources. The probability value, Pi, provided by the fitness-proportion selection method determines how,

$$\boldsymbol{P_i} = \frac{\boldsymbol{f_{it_i}}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{SN}} \boldsymbol{fit_j}} \quad \textbf{(3)}$$

where SN is the number of food sources accessible, f_{it_i} is the fitness value of the *i*th food source, and P_i is the probability of the *i*th food source.

The neighborhood of the chosen, better food sources was then examined by onlooker bees, who updated the food sources utilizing equation 2. The previous food supply and the new candidate solution are then contrasted using the greedy-selection method. Prior to beginning the scout-bee phase, the generation next memorizes its best food source to date.

4. Scout Bees Phase: When a food supply reaches its peak and no longer improves, it is discarded during the scout-bee phase [11]. The threshold serves as a control parameter to denote a depleted food source [11]. When a worker bee's food supply is exhausted, it transforms into a scout bee. In subsequent flights, the scout-bee will randomly explore the search area in seeking out new food sources by using

$$\boldsymbol{x_i^j} = \boldsymbol{x_{min}^j} + \boldsymbol{rand}(\mathbf{0}, \mathbf{1})\left(\boldsymbol{x_{max}^j} - \boldsymbol{x_{min}^j}\right) \quad (4)$$

where, respectively, x_{max}^j and x_{min}^j represent the lower and upper bounds of the search space. The function *rand (0, 1)* generates random numbers between [0, 1]. The scout-bee must take this step in order to swap the eroded food source with a fresh one and restore population balance.

5. Termination Phase: The algorithm's termination criteria are based on the highest possible number of iterations or cycles (MCN) [15]. Before the ABC algorithm computation, the user sets this value.

2.6.2. Stepwise Presentation of ABC Algorithm

Artificial Bee Colony (ABC) Optimization Algorithm Pseudocode:

1. ***Initialization Phase***

REPEAT

2. ***Employee Bees Phase***
3. ***Onlooker Bees Phase***
4. ***Scout Bees Phase***

Memories are the best solution achieved.
UNTIL Stopping criteria are met.

Artificial Bee Colony Optimization Steps:

Step 01: Generate the initial population randomly (X_i), i = 1,2,3,4,….Population Size

Step 02: Calculate fitness values for each agent in the population.

Step 03: Memorize the best (X_{Best}) solution in the population.

Step 04: Set Current Iteration (t = 1)

Step 05: Generate new solutions for employee bee (v_i) from old solutions (X_i).

Step 06: Compute the fitness of all new solutions in the population.

Step 07: Keep the best solution between current and candidate solutions.

Step 08: Calculate the Probability (P_i) for the solution (X_i).

Step 09: Generate new solutions (v_i) for onlooker bees from the solution selection depending on its P_i.

Step 10: Calculate the fitness of all new solutions in the population.

Step 11: Determined the abandoned solution if exist, replace it with a new random solution X_i.

Step 12: Keep the best solution found in the population.

Step 13: Increment counter $t = t+1;$

Step 14: Repeat until $t<=MaxT$ stopping criteria met.

3. TRAVELLING SALESMAN PROBLEM (TSP)

The travelling salesman problem (TSP) is an operations research and theoretical computer science NP-hard combinatorial optimization issue. Planning, logistics, network connectivity, transportation, and the production of semiconductor chips are perhaps a few of the many uses for TSP. The TSP can be specifically known as follows [2]: On an undirected graph representing cities or nodes to be visited, the goal of this permutation problem is to determine the path with the minimal length (or the lowest cost) [14].

The travelling salesman begins at one node, travels to each subsequent node just once, and then circles back to the initial node. In other words, the goal is to choose σi so that the total of all Euclidean distances between each node and its successor is as minimal as possible given n cities, denoted *c1, c2,..., cn,* and permutations, σ1,..., σn! The first node in the permutation is the successor of the previous node. Equation (5) determines the Euclidean distance *d* between any two cities with coordinates *(x1, y1)* and *(x2, y2).*

$$\boldsymbol{d} = \sqrt{(|\boldsymbol{x_1} - \boldsymbol{x_2}|)^2 + (|\boldsymbol{y_1} - \boldsymbol{y_2}|)^2} \quad (5)$$

The TSP must consequently choose a Hamiltonian tour with the lowest possible cost. Typically, TSP can be represented as follows using graph notations:

- Let *G = (C, E)* be a graph.
- *C* is an n-city set, and *C = c1,..., cn.*
- *E* is a set of arcs or edges,

Generally , *E* is related to a distance (or cost) matrix *d.*

For example, consider the graph shown in the figure-3.1. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.

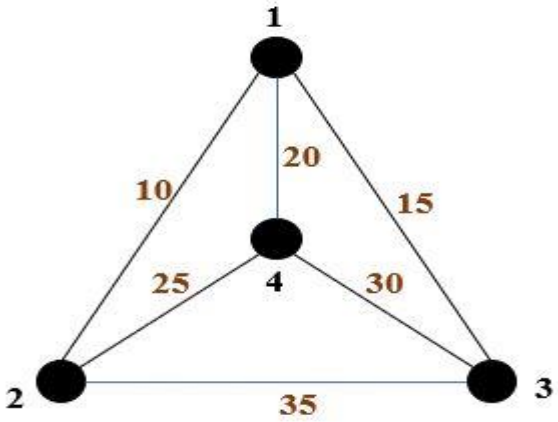

Figure 3.1: Graph representation of TSP

TSBLIB

TSPLIB is a text-based file format for storing graphs which is used in studying instances of the Traveling Salesman Problem(TSP) and related problems. The format consists of a sequence of node definitions, followed by a list of edges, which specify node pairs and an optional edge label. TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types. Instances of the following problem classes are available.

- Symmetric traveling salesman problem (TSP): Given a set of n nodes and distances for each pair of nodes, find a roundtrip of minimal total length visiting each node exactly once. The distance from node i to node j is the same as from node j to node i.
- Hamiltonian cycle problem (HCP): Given a graph, test if the graph contains a Hamiltonian cycle or not. Asymmetric traveling salesman problem (ATSP) Given a set of n nodes and distances for each pair of nodes, find a roundtrip of minimal total length visiting each node exactly once. In this case, the distance from node i to node j and the distance from node j to node i may be different.
- Sequential ordering problem (SOP): This problem is an asymmetric traveling salesman problem with additional constraints. Given a set of n nodes and distances for each pair of nodes, find a Hamiltonian path from node 1 to node n of minimal length which takes given precedence constraints into account. Each precedence constraint requires that some node i has to be visited before some other node j.
- Capacitated vehicle routing problem (CVRP): We are given n − 1 nodes, one depot and distances from the nodes to the depot, as well as between nodes. All nodes have demands which can be satisfied by the depot. For delivery to the nodes, trucks with identical capacities are available. The problem is to find tours for the trucks of minimal total length that satisfy the node demands without violating truck capacity constraint. The number of trucks is not specified. Each tour visits a subset of th nodes and starts and terminates at the depot. (Remark: In some data files a collection of alternate depots is given. A CVRP is then given by selecting one of these depots.) Except, for the Hamiltonian cycle problems, all problems are defined on a complete graph and, at present, all distances are integer numbers. There is a possibility to require that certain edges appear in the solution of a problem.

Reads and writes TSPLIB format files. TSPLIB files can be used by most TSP solvers. Sample instances for the TSP in TSPLIB format are available on the TSPLIB homepage. For example:

read_TSPLIB(file, precision = 0)
write_TSPLIB(x, file, precision = 6, inf = NULL, neg_inf = NULL)

S3 method for class 'TSP'
write_TSPLIB(x, file, precision = 6, inf = NULL, neg_inf = NULL)

S3 method for class 'ATSP'
write_TSPLIB(x, file, precision = 6, inf = NULL, neg_inf = NULL)

S3 method for class 'ETSP'
write_TSPLIB(x, file, precision = 6, inf = NULL, neg_inf = NULL)

tsplib95 is a Python library typically used in Utilities, Build Tool applications. tsplib95 has no vulnerabilities, it has build file available and it has low support. However tsplib95 has 5 bugs and it has a Non-SPDX License. tsplib95 has no vulnerabilities reported, and its dependent libraries have no vulnerabilities reported.tsplib95 code analysis shows 0 unresolved vulnerabilities.

4. ABC METHOD FOR TSP

A graph $G = (C, E)$ can be used to express TSP, where C = {1, 2,..., n} is an array of nodes, and $E = \{(i,j) \mid i,j \in C\}$ is a set of all connections between those nodes. Every city is represented by a node, and every edge denotes a potential route connecting two cities that are related. The distance d_{ij}, which is connected to edge (i,j), represents the Euclidean separation between cities i and j. Before running the main algorithm, the heuristic data is calculated. As a result, all edges' distances were estimated and recorded. The fitness value, which is derived from the sum of the Euclidean distances of the subsequent edges in the trip, is used to iteratively assess the quality of the assembled solutions after the solutions have been constructed. First, a starting population (food source placements) is produced randomly while taking population variety into account to solve TSP using the ABC algorithm.

Each algorithm has a set of control parameters that are necessary for it to operate effectively. Therefore, the artificial bee colony method also has some control parameters. To figure out the values of these control factors, we conducted our own tests and a thorough investigation of the literature. This led us to the conclusion that the parameters

we used in this experiment were both standard and appropriate for it. The Maximum Cycle Number (MCN) is the first control parameter. We utilized the outcomes for the 2500 MCN value, where MCN equals the maximum number of generation. The maximum population in our study is the next parameter, and we have set it to be 50. The number of attempts is another control parameter, and we've chosen 40 for our research. It should be emphasized that each run has a maximum cycle count, which in our experiment is 2500. Dimension, the fourth control parameter, is dependent on the number of cities.

```
procedure BCO
      Initialize_Population( )
         while stop criteria are not fulfilled do
             while all bees have not built a complete path do
                Observe_Dance( )
                 Forage_ByTransRule( )
                 Perform_Waggle_Dance ( )
             end while
         end while
end procedure BCO
```

Table 4.1: ABC approach for TSP

5. RESULTS ASSESSMENTS

Throughout this segment, we evaluated our strategy for solving the problem of the traveling salesman using 12 cities. The path map without optimization is shown in Figure 5.1, while the path diagram after optimization is shown in Figure 5.2. The path has been greatly optimized as a result of the artificial bee colony algorithm application. The distance can rapidly coincide with the minimal path length once the iteration is finished. We can also handle related optimization issues using our suggested algorithm ABC.

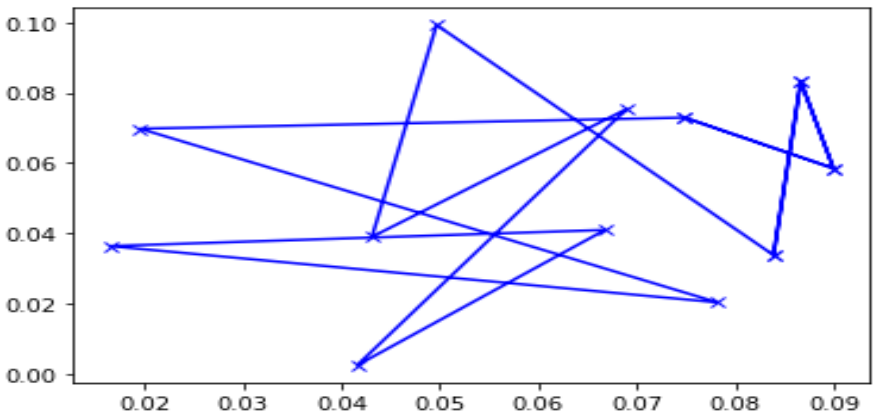

Figure 5.1: Path map for TSP without optimizatio

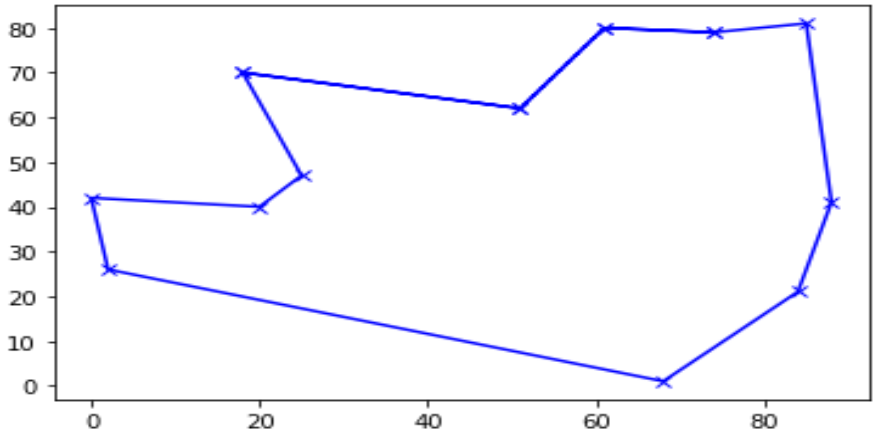

Figure 5.2: Path map for TSP with optimization

After implementing the ABC algorithm on TSP using the graph method for 12 cities, then I performed another implementation in Python using the TSPLIB library for cities of various sizes. I estimated the path distance and tour time for berlin52, att48, kroA100, burma14, ts225, att532, dj38, pr144. Then I listed these results in Table 5.1. After evaluating this, we can see that as the number of cities (nodes) increases, so does the tour time.

In jupyter environment, at first define libraries and load the TSPLIB file.

```
# define Libraries
import random
import math
import time

# Import the berlin52 TSPLIB dataset
cities = []
with open("berlin52.tsp", "r") as f:
    for line in f:
        if "NODE_COORD_SECTION" in line:
            break
    for line in f:
        if "EOF" in line:
            break
        parts = line.split()
        cities.append((float(parts[1]), float(parts[2])))
```

Define distance function using Euclidean distance formula.

```
# Define the distance function
def distance(path):
    dist = 0
    for i in range(len(path) - 1):
        x1, y1 = cities[path[i]]
        x2, y2 = cities[path[i+1]]
        dist += ((x1 - x2)**2 + (y1 - y2)**2)**0.5
    return dist
```

Next initialize the parameters for ABC algorithm

```
# Initialize the population of bees
num_bees = 50
num_sites = len(cities)
population = [[random.randint(0, num_sites-1) for _ in range(num_sites)] for _ in
range(num_bees)]

# Define the algorithm parameters
max_iterations = 1000
limit = 100
# Start the timer
start_time = time.time()
```

This section start the iteration, define fitness function, employed bee as well as onlooker bee phase.

```
# Iterate for a fixed number of iterations
for iteration in range(max_iterations):
    # Evaluate the fitness of each bee
    fitness = [distance(bee) for bee in population]
    # Select the employed bees
    employed = [population[i] for i in range(num_bees) if random.random() < 0.8]
    # Generate new solutions for the employed bees
    for bee in employed:
        i, j = random.randint(0, num_sites-1), random.randint(0, num_sites-1)
        while i == j:
            i, j = random.randint(0, num_sites-1), random.randint(0, num_sites-1)
        bee[i], bee[j] = bee[j], bee[i]
    # Evaluate the fitness of the newly generated solutions
    new_fitness = [distance(bee) for bee in employed]
    # Select the onlooker bees
    onlookers = []
    for i in range(num_bees - len(employed)):
        prob = [fitness[j]/sum(fitness) for j in range(num_bees)]
        r = random.random()
        for j in range(num_bees):
            r -= prob[j]
            if r <= 0:
                onlookers.append(population[j])
                break
```

Then define new solutions for onlooker bees and updated the new population by replacing worst solutions with scout bees.

```
# Generate new solutions for the onlooker bees
    for bee in onlookers:
        i, j = random.randint(0, num_sites-1), random.randint(0, num_sites-1)
        while i == j:
            i, j = random.randint(0, num_sites-1), random.randint(0, num_sites-1)
        bee[i], bee[j] = bee[j], bee[i]
    # Evaluate the fitness of the newly generated solutions
    new_onlooker_fitness = [distance(bee) for bee in onlookers]
    # Select the best solution from the employed and onlooker bees
    population = employed + onlookers
    fitness = new_fitness + new_onlooker_fitness
    best_bee = population[fitness.index(min(fitness))]
    # Update the population of bees by replacing the worst solutions with the scout bee
    for i in range(num_bees):
        if distance(population[i]) > min(fitness) + limit:
```

```
            population[i] = best_bee
    # Print the best distance in each iteration
    #print("Iteration:", iteration+1, "Best distance:", min(fitness))
```

Finally printed the output of best solution, best distance and total tour time for best distance.

```
#Stop the timer
end_time = time.time()

# Print the final best solution, distance, and time
print("Final best solution:", best_bee)
print("Final best distance:", min(fitness))
print("Time taken:", end_time - start_time, "seconds")
```

```
85 # Print the final best solution, distance, and time
86 print("Final best solution:", best_bee)
87 print("Final best distance:", min(fitness))
88 print("Time taken:", end_time - start_time, "seconds")
```

```
Final best solution: [15, 29, 41, 32, 36, 22, 2, 38, 6, 2, 36, 24, 42, 47, 42, 42, 31, 20, 4, 47, 43, 22, 49, 38, 49, 29, 44, 2
3, 23, 17, 18, 42, 29, 50, 5, 41, 47, 19, 37, 12, 44, 41, 42, 4, 35, 29, 30, 4, 29, 37, 45, 34]
Final best distance: 22454.543179385044
Time taken: 9.31383228302002 seconds
```

Figure 5.3: Final output for berlin52 TSPLIB library

Table-5.1: Final best path distance and time of 8 TSPLIB library for TSP using ABC

TSBLIB	NO. OF CITIES(NODES)	BEST PATH DISTANCE (km)	TAKEN TIME (seconds)
berlin52	52	22454	9.32
att48	48	133039	8.94
dj38	38	25269	7.89
pr144	144	728744	30.00
att532	532	1551449	99.05
ts225	225	1565709	39.18
kroA100	100	157092	18.54
burma14	14	71	4.55

Then printed the graphical representation of best optimal path for TSP using ABC.

```
import matplotlib.pyplot as plt

# Unpack the x and y coordinates of the cities
x = [city[0] for city in cities]
y = [city[1] for city in cities]
# Plot the cities
plt.scatter(x, y)

# Plot the best solution
best_x = [x[city] for city in best_bee]
best_y = [y[city] for city in best_bee]
best_x.append(x[best_bee[0]])
best_y.append(y[best_bee[0]])
plt.plot(best_x, best_y, 'b-')
```

```
# Add labels and show the plot
plt.xlabel("X Coordinate")
plt.ylabel("Y Coordinate")
plt.title("Best Solution for TSP using ABC Algorithm")
plt.show()
```

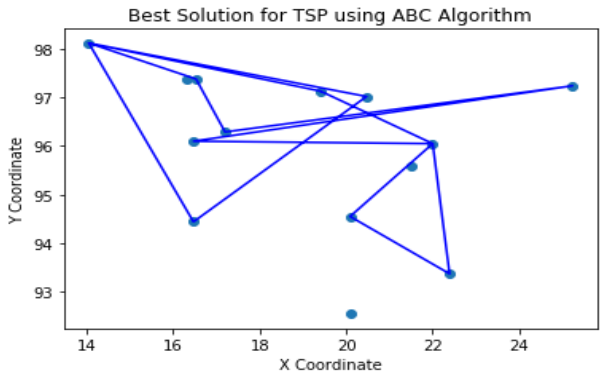

Figure 5.4 : Graphical view of best solution path for burma14 TSPLIB file

6. CONCLUDING REMARK

The ABC method for TSP was presented in this research to effectively reduce the complexity of evolutionary algorithms. The ABC algorithm can efficiently locate an ideal path since it has a powerful search capability in the graph space. Experimental findings demonstrate that this method also takes into account solution quality and execution duration. Future studies will involve combining the algorithm with other algorithms to provide better outcomes.

7. REFERENCES

[1] N. Pathak and S. Prakash Tiwari, "Travelling Salesman Problem Using Bee Colony With SPV," *Int. J. Soft Comput. Eng.*, vol. 2, no. 3, pp. 410–414, 2012.

[2] S. Pandey and S. Kumar, "Enhanced Artificial Bee Colony Algorithm and It's Application to Travelling Salesman Problem," *HCTL Open Int. J. Technol. Innov. Res.*, vol. 2, no. March, pp. 137–146, 2013.

[3] W. J. D. L.; Bixby, R. M.; Chvátal, V.; Cook, "The Traveling Salesman Problem," *Applegate*, no. ISBN 0691129932, 2006.

[4] G.A. Croes, "A method for solving traveling salesman problems".

[5] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[6] A. Ugur and D. Aydin, "An interactive simulation and analysis software for solving TSP using ant colony optimization algorithm," *Adv. Eng. Softw.*, vol. 40, no. 2, pp. 341–349, 2009.

[7] W. Li, W. Li, Y. Yang, H. Liao, J. Li, and X. Zheng, "Artificial bee colony algorithm for traveling salesman problem," *Adv. Mater. Res.*, vol. 314–316, no. Icmmcce, pp. 2191–2196, 2011, doi: 10.4028/www.scientific.net/AMR.314-316.2191.

[8] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in Internet hosting centers," *Adapt. Behav.*, vol. 12, no. 3–4, pp. 223–240, 2004.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Kaysari,Turkey, 2005.

[10] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput. J.*, vol. 8, no. 1, pp. 687–697, 2008.

[11] N. Sulaiman, J. Mohamad-Saleh, and A. G. Abro, "New Enhanced Artificial Bee Colony Algorithm for Reactive Power Optimization," *AIP Conf. Proc.*, vol. 1660, 2015, doi: 10.1063/1.4915670.

[12] A. G. Abro and J. Mohamad-Saleh, "Enhanced probabilityselection artificial bee colony algorithm for economic load dispatch: a comprehensive analysis .," *Eng. Optim.*, vol. 46, no. 10, pp. 1315–1330, 2014.

[13] Y. Xu, P. Fan, and L. Yuan, "A simple and efficient artificial bee colony algorithm," *Math. Probl. Eng.*, vol. 2013, 2013, doi: 10.1155/2013/526315.

[14] S. Sobti and P. Singla, "Solving Travelling Salesman Problem Using Artificial Bee Colony Based Approach," *Int. J. Eng. Res. Technol.*, vol. 2, no. 6, pp. 186–189, 2013, [Online]. Available: http://www.ijert.org/browse/volume-2-2013/june-2013-edition?download=3722:solving-travelling-salesman-problem-using-artificial-bee-colony-based-approach&start=20

[15] D. Karaboga and B. Akay, "A comparative study of artificial Bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132.