

# **Comparative Study of Optimisation Algorithms on Travelling Salesman Problem in Heterogeneous Environments**

**A**

## **Project Report**

*submitted in partial fulfillment of the  
requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE**

**Specialization in**

**Oil And Gas Informatics**

**By :**

<b>Name</b>	<b>Roll No</b>	<b>Branch</b>
Diwakar Kumar	R970216025	CSE OGI
Gaurav Mishra	R970216028	CSE OGI
Abhishek Suman	R970216004	CSE OGI

*Under the guidance of*

**Dr. Kingshuk Srivastava**  
**Assistant Professor (SG)**  
**Department of Informatics**



**Department of Informatics**  
**School of Computer Science**  
**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**  
**Bidholi, Via Prem Nagar, Dehradun, Uttarakhand**  
**2018-19**



## CANDIDATES DECLARATION

I/We hereby certify that the project work entitled **Comparative Study of Optimisation Algorithms on Travelling Salesman Problem in Heterogeneous Environments** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Oil and Gas Informatics and submitted to the Department of Informatics at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from **September, 2018** to **December, 2018** under the supervision of **Dr. Kingshuk Srivastava, Assistant Professor(SG), Department of Informatics**.

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

<b>Name of Student(s)</b>	Diwakar Kumar	Gaurav Mishra	Abhishek Suman
<b>Roll No.</b>	R970216025	R970216028	R970216004

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 17 December 2018)

**(Dr. Kingshuk Srivastava)**  
Project Guide

**Dr. Thipendra Pal Singh**

Head  
Department of Informatics  
School of Computer Science  
University of Petroleum And Energy Studies  
Dehradun - 248 001 (Uttarakhand)

## **ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our guide **Dr. Kingshuk Srivastava**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department, **Dr. Thipendra Pal Singh**, for his great support in doing our **Comparative Study of Optimisation Algorithms on Travelling Salesman Problem in Heterogeneous Environments** at SoCS.

We are also grateful to **Dr. Manish Prateek, Dean SoCS**, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

<b>Name of Student(s)</b>	Diwakar Kumar	Gaurav Mishra	Abhishek Suman
<b>Roll No.</b>	R970216025	R970216028	R970216004

# ABSTRACT

Every individual has an ability to interact mutually, which is one of the fundamental social behavior present from humans to insects. This social interaction increases the rate of adaptation and improve faster than biological evolution. The project implements these major driving concepts of optimization algorithm. For every problem there must be an optimal and feasible solution in order to be optimized. While dealing with number of algorithms per day, selecting an algorithm to implement in our problem domain is difficult. Therefore, this project depicts different optimization algorithms and their implementation in Travelling Salesman Problem. It will also depicts how different environment affects the efficiency of these optimization algorithms and their implementation in respective area of applications. This project includes a comparative study of different optimization algorithms (i.e. Ant Colony Optimization, Particle Swarm Optimization and Artificial Bee Colony) on Travelling Salesman Problem in heterogeneous environments.

**Keywords:** Social behaviour, Optimisation, Travelling Salesman Problem

# TABLE OF CONTENTS

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Particle Swarm Optimisation (PSO) . . . . .	1
1.2	Ant Colony Optimisation (ACO) . . . . .	2
1.3	Artificial Bee Colony (ABC) . . . . .	2
<b>2</b>	<b>Literature Review</b>	<b>2</b>
<b>3</b>	<b>Problem Statement</b>	<b>3</b>
<b>4</b>	<b>Objective</b>	<b>3</b>
<b>5</b>	<b>Design Methodology</b>	<b>3</b>
5.1	PSO . . . . .	3
5.2	ACO . . . . .	5
5.3	ABC . . . . .	7
5.3.1	Mathematical Expression . . . . .	7
<b>6</b>	<b>Implementation</b>	<b>9</b>
6.1	Pseudocode . . . . .	9
6.1.1	Particle Swarm Optimisation . . . . .	9
6.1.2	Ant Colony Optimisation . . . . .	9
6.1.3	Artificial Bee Colony . . . . .	10
6.2	Output Screen . . . . .	11
6.3	Result Analysis . . . . .	13
6.3.1	Execution Time of Program . . . . .	14
<b>7</b>	<b>Conclusion and Future Scope</b>	<b>18</b>
<b>A</b>	<b>APPENDIX I PROJECT CODE</b>	<b>20</b>

# LIST OF FIGURES

## List of Figures

1	<i>A pictorial comparison of classical and modern optimisation strategies . . . . .</i>	1
2	<i>PSO Search Domain . . . . .</i>	4
3	<i>Flowchart of PSO algorithm . . . . .</i>	5
4	<i>Flowchart of ACO algorithm . . . . .</i>	6
5	<i>Flowchart of ABC algorithm . . . . .</i>	8
6	<i>Output 1 of PSO Algorithm . . . . .</i>	11
7	<i>Output 2 of PSO Algorithm . . . . .</i>	11
8	<i>Output 1 of ABC Algorithm . . . . .</i>	12
9	<i>Output 2 of ABC Algorithm . . . . .</i>	12
10	<i>Systems Used . . . . .</i>	13
11	<i>Execution Time of Algorithms in LINUX . . . . .</i>	14
12	<i>Execution Time of Algorithms in LINUX . . . . .</i>	14
13	<i>Execution Time of Algorithms in Windows . . . . .</i>	15
14	<i>Execution Time of Algorithms in Windows . . . . .</i>	15
15	<i>Execution Time of Algorithms in MAC OS</i> 16	
16	<i>Comparison of PSO in Windows and LINUX . . . . .</i>	16
17	<i>Comparison of ABC in Windows and LINUX . . . . .</i>	17
18	<i>Comparison of PSO and ABC in LINUX and Windows . . . . .</i>	17

# 1 Introduction

Everything in the world is in search for the optimal state. The algorithm which selects the best feasible solution among the different set of possible solutions are termed as optimisation algorithm. For instance, suppose there are four ways to reach a final node from an initial node. The weight of first path is 45, for second path 78, for third path 28 and the last path has weight of 36. So, it is obvious that a user must go for the path which has least weight. Optimisation problems are mainly used for finding nearly optimal solution and are frequently encountered in various applications such as TSP, Container Loading problems (CL), Scheduling problems, engineering design etc. TSP is one of the NP hard optimisation problems. In TSP, the salesman travels all the cities at once and returns to the starting city with the possible shortest route within small duration[1]. Many heuristic optimisation methods are developed so far for searching nearly optimal solution in solving TSP such as Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Simulated Annealing (SA) and Artificial Bee Colony (ABC).

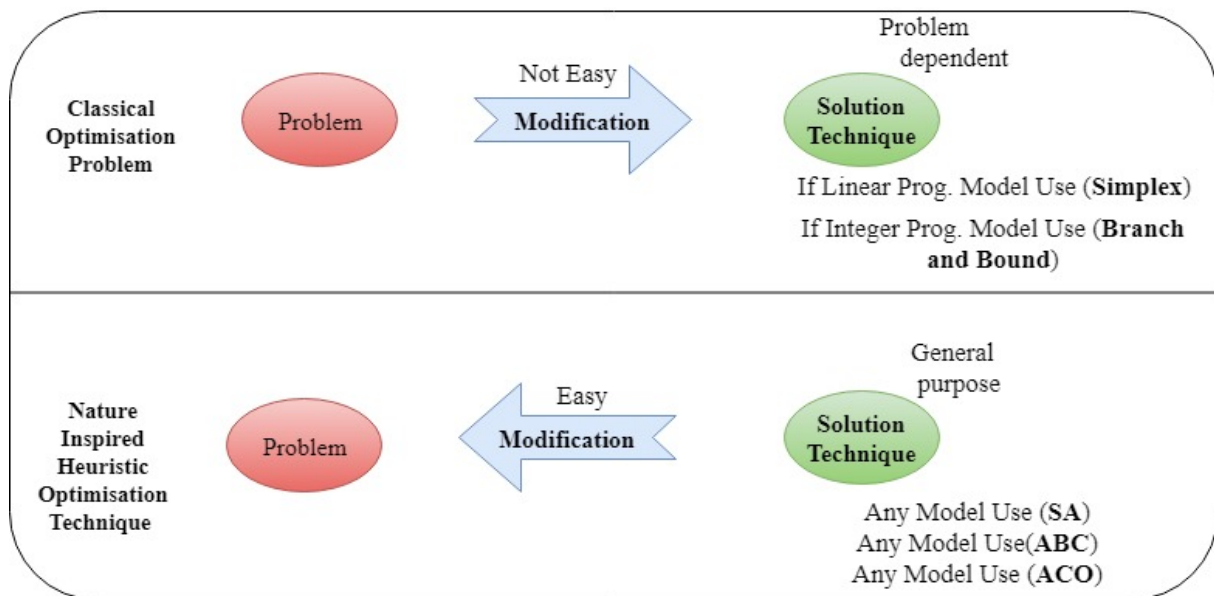


Figure 1: A pictorial comparison of classical and modern optimisation strategies

## 1.1 Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation was initially developed by James Kennedy and Russell C. Eberhart in 1995. After then, it was been improved and redefined by many other researchers. Particle Swarm Optimisation was developed based on the social behaviour of animals that is navigation and forwarding of flock of birds or school of fish in search of food and shelter. PSO depends upon the degree of randomness and freedom. We will study about the proper mathematical expressions and implementation of PSO. This whole optimisation algorithm was developed based on the communication between the particles and learning from each other.

PSO Search Strategy depend upon 3 main factors:

1. Current Position.
2. Personal best location.
3. Team's best location.

## 1.2 Ant Colony Optimisation (ACO)

An ACO is based on the metaphor ant seeking food. It is an algorithm developed by Stickland and Dorigo in 1991 for the problem which is based on finding optimal paths in graphs. It is a sign based stigmergy where optimisation problems are visualized as graphs (directed). As we know that ant follows the trails of other ants in process of seeking food, the pheromones released by the former ants helps the latter ants to detect the path trails. Higher the pheromone density larger the number of ants following that path. Ant colony optimisation can be applied to any discrete optimisation problem for which some solution construction mechanism can be conceived. For solution first we define a generic problem representation that the ants in ant colony optimisation may exploit to constructed solution afterwards we define the ACO metaheuristic (a method for solving very general classes of problems by trial and error).

## 1.3 Artificial Bee Colony (ABC)

The Artificial Bee Colony (ABC) algorithm is a recently introduced swarm intelligence based algorithm inspired by the intelligent food foraging behavior of the honey bees found in nature. ABC algorithm shows more effective when compared to the above optimisation algorithms since it is the latest optimisation technique which is developed by Dervis Karaboga in the year 2005. Since its advent, ABC and its variants have often successfully employed to wide and diverse range of problems, such as numeric optimisation, discrete optimisation, multi-objective optimisation, industrial process control, structural design, design of digital IIR filters, PID controller, machine learning and so on. In comparison to other greedy and local search based algorithms, ABC is more resilient against premature convergence and local optima, because the population of candidate solutions can maintain some amount of diversity that is necessary to continue search space explorations avoiding the locally optimal points.

# 2 Literature Review

ACO, PSO and ABC operation are the best to use in the machine learning as these are the more error less and optimised then the other. Here is the conclusion of some of the reference paper that we review to make our project better and to know more technologies that we can use in our system.

- In paper[2] by Saad Ghaleb Yaseen and Nada M.A.AL.-Slamy on Ant Colony Optimisation June 2008 AL-Zaytoonah University of Jordan, they have introduced about the ant colony optimisation algorithm and provides a basic approach of designing meta heuristic algorithms for combinatorial optimisation problems. This paper introduces ACO (Ant Colony Optimisation) as a distributed algorithm that is applied to solve Travelling Salesman Problem (TSP). They have also described how the food seeking behaviour of ants is used as finding optimal path in the Travelling Salesman Problem. This paper also includes how ants adaptively modify the way the problem is represented and perceived by other ants, but they are not adaptive themselves. ACO concepts are used for good propagation process, help to find a systematic, effective procedure to find good path for good propagation with respect to some predefined cost and constrains function.
- In paper[3] by James Kennedy' and Russell Eberhart<sup>2</sup> on Particle Swarm optimisation Washington, DC 2012 , <sup>2</sup>Purdue School of Engineering and Technology, they have discussed about methods for optimisation of continuous nonlinear functions. They have basically mentioned the origin of particle swarm optimisation. They have also shown a detailed study about social behaviour of animals (bird flocking and fish schooling) in



search of food. From the nature of animals, they have explained the five basic principles which the particles follow in order to find food- proximity principle, quality principle, diverse response, stability and adaptability. They have also shown the evolution and re-definition of Particle Swarm Optimisation algorithm in which the expression was change from pincement,gdecrement to pbest and gbest.

- In paper[4] by Sahil Sobti and Parikshit Singla on Artificial Bee Colony 6 June, 2013 Assistant Professor , DIET , Karnal, they have explained about the performance of ABC algorithm in solving Travelling Salesman Problem. They have also proposed on ABC based algorithm to enumerate rule based system.
- In paper[5] by Ginnu George and Dr.Kumudha Raimond on Artificial Bee Colony 1 March 2013, Computer Science and Engineering Karunya University, they have explained about the performance of variants of ABC in solving TSP. They have also investigated over variants of ABC like I-ABC(Improved) and PS-ABC(Prediction Selection) which are used to find optimal path for TSP. The variants of ABC are presented aiming at minimizing the distance of tour. The result of ABC are compared with I-ABC and PS-ABC and it shows that PS-ABC perform well in finding shortest distance within minimum span of time. Also the graphical representation of the comparison of performance of all 3 algorithms are shown.

### **3 Problem Statement**

To a given N numbers of stops, identifying the shortest or optimal path from start and back to the original location is a big problem for computation. With increasing number of stops, it is computationally difficult to solve as its complexity increases exponentially.

### **4 Objective**

To perform comparative study of optimisation algorithms for Travelling Salesman Problem in heterogeneous environments.

### **5 Design Methodology**

The main obejctive of doing comperative study is to find the best algorithm which has an optimal solution for TSP. In order to achieve it, implementation of their mathematical expression is to be performed.

#### **5.1 PSO**

PSO have many advantages such as comparative simplicity, rapid convergence and little parameters to be adjusted. It has been used in many field such as mechanical, chemical, aerospace design etc. The particle swarm algorithm is an optimisation technique inspired by the metaphor of social interaction observed among insects or animals. The kind of social interaction modelled within a PSO is used to guide a population of individuals (particles) moving toward the most promising area of the search space.[6] In a PSO algorithm, each particle is a candidate solution and each particle flies through the search space, depending on two important factors: the best position the current particle have found so far and the global best position identified from the entire population. The rate of change in positions of particle is given by its velocity.

Particles velocity and positions are updated related to the pbest and gbest values.[7] Lets derive the mathematical expression out of it.

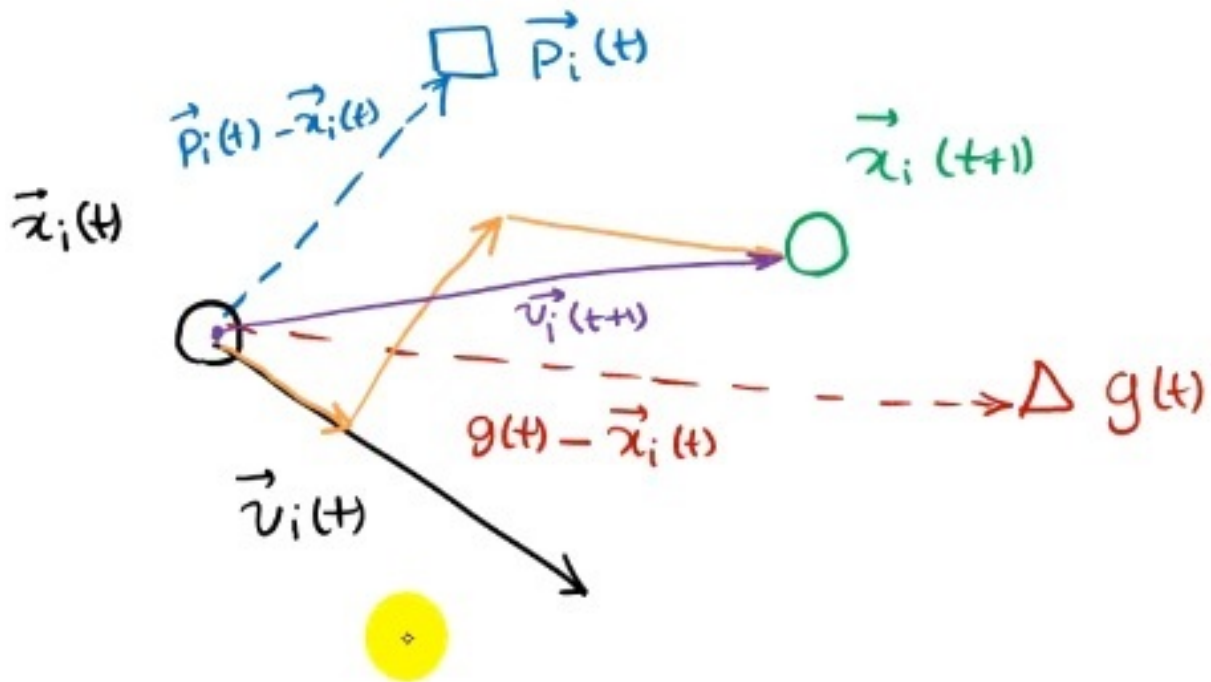


Figure 2: PSO Search Domain

Here,  $x_i(t)$  is the position of particle at time  $t$ ,  
 $v_i(t)$  is the velocity by which it is moving in current direction,  
 $P_i(t)$  is the personal best location of a particle, and  
 $g_i(t)$  is the global or teams best location.

So, the particle will first move into its current location and then towards its personal best location and then after its global best location. After all this movement finally the particle will rest in a better position than earlier. Likewise, the whole bunch of particles move together and until ones find the destination.

For updating the position,

$$X_i(t+1) = X_i(t) + V_i(t+1)$$

$$V_i(t+1) = wV_i(t) + c1(P_i(t)-X_i(t)) + c2(g(t)-X_i(t))$$

**Standard Scalar Particle Swarm Optimisation:**

$$V_{ij}(t+1) = wV_{ij}(t) + r1c1(P_{ij}(t)-X_{ij}(t)) + r2c2(g(t)-X_{ij}(t))$$

where  $r1$  and  $r2$  are random numbers range  $[0,1]$ ,  
 $c1$  and  $c2$  are acceleration coefficients,  
 $w$  is inertia coefficient,  
 $wV_{ij}(t)$  is inertia,  
 $r1c1(P_{ij}(t)-X_{ij}(t))$  is Cognitive Component, and  
 $r2c2(g(t)-X_{ij}(t))$  is Social Component

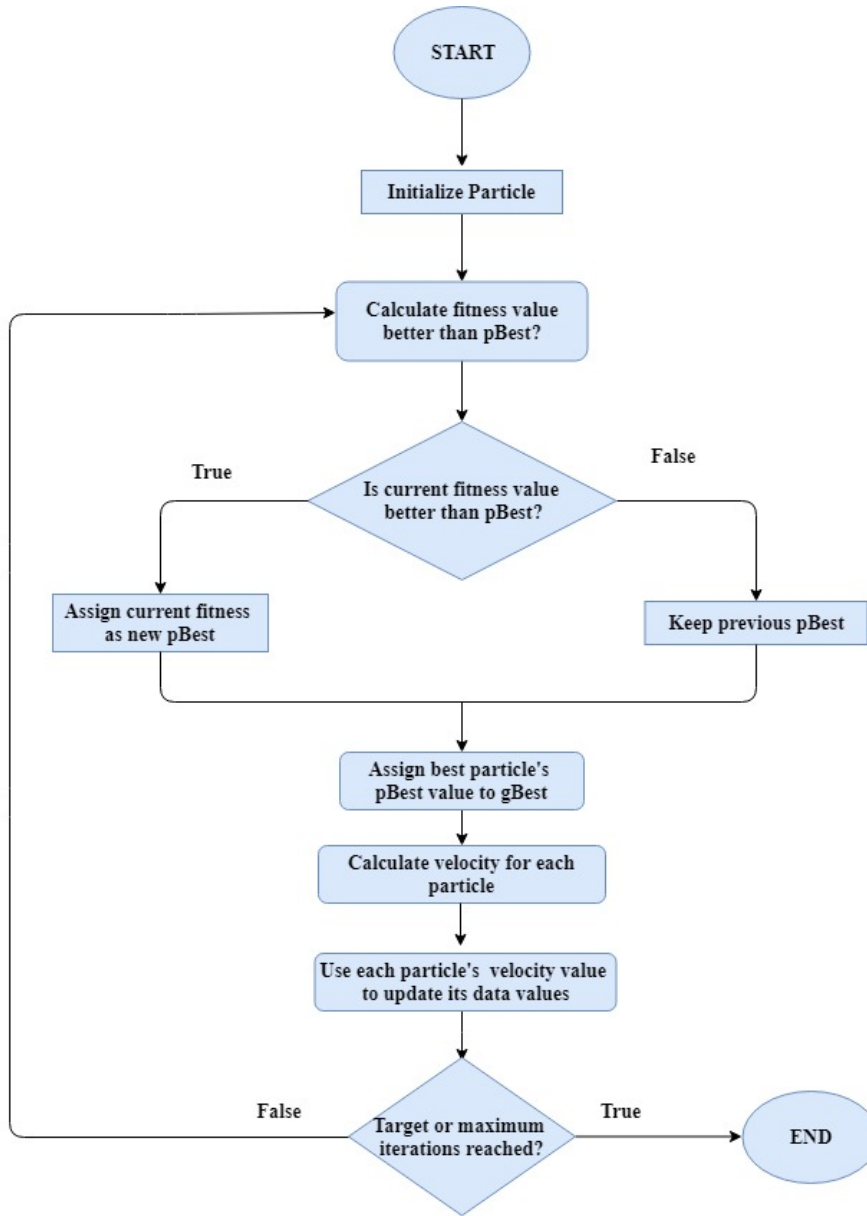


Figure 3: Flowchart of PSO algorithm

## 5.2 ACO

### Edge Selection

The probability ( $P_{ij}$ ) of an ant  $k$  to select a city  $j$ , standing at city  $i$  is given by the formula below:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k=1}^n \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}$$

where  $\tau_{ij}$  is intensity of pheromone trail between places  $i$  and  $j$ ,

$\alpha$  is parameter to regulate the influence of  $\tau_{ij}$ , usually it lies between 0 to 1,

$\eta_{ij}$  is the visibility of place  $j$  from place  $i$ ,

$\eta$  is  $1/d_{ij}$  where  $d_{ij}$  is distance between  $i$  and  $j$  (if distance is more then visibility is low and if distance is low then visibility is high),

$\beta$  is parameter to regulate the influence of visibility.[8]

## Pheromone Update

Amount of pheromone updated though edges during trail of ants is calculated by given equation:

$$\tau_{ij} = (1-\rho) \tau_{ij} + \Delta \tau_{ij}$$

where  $\tau_{ij}$  is amount of pheromone on a given edge i-j,

$\rho$  is rate of pheromone evaporation,

$\Delta \tau_{ij}$  is amount of pheromone deposited and it is given by,

$$\Delta \tau_{ij}^k = \begin{cases} \frac{1}{L_k} & \text{if ant k travels on edge i,j} \\ 0 & \text{otherwise} \end{cases}$$

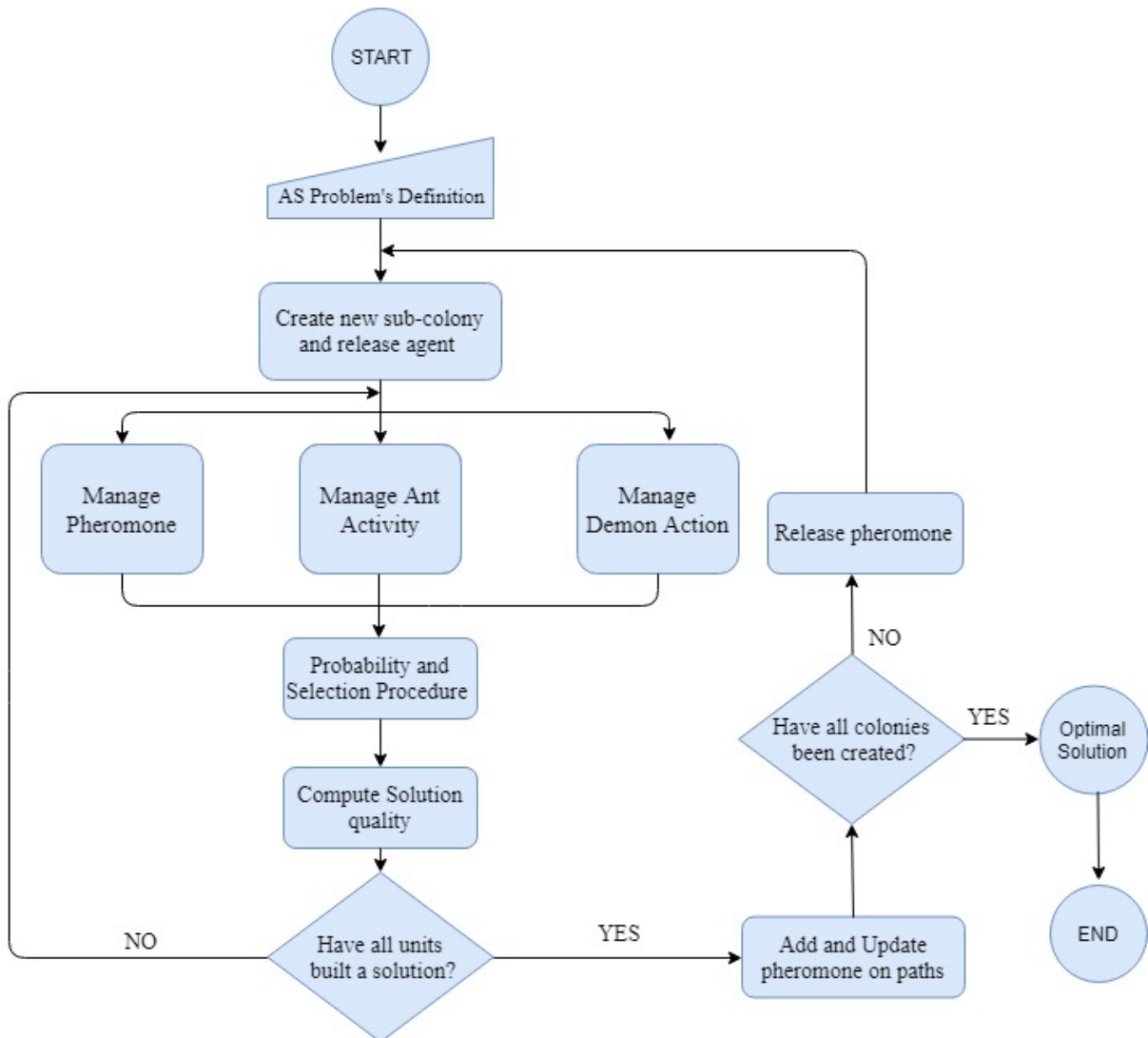


Figure 4: Flowchart of ACO algorithm

### 5.3 ABC

Artificial Bee Colony contains three types of colony:

1. Employee Bees
2. Onlookers
3. Scouts

**The main steps of algorithm are:**

1. Initial food sources are produced for employee bees
2. Repeat
  - (a) Each employee bee goes to a food source in her memory and determines a closest source, then evaluates its nectar amount and dance in hive.
  - (b) Each onlooker watches the dances of employee bee and choose one of their source depending on dance, and then goes to that source. After choosing a neighbor around that, she evaluates its nectar amount.
  - (c) Abandoned food sources are determined and are replaced with new food source discovered by scouts.[9]
  - (d) The best food source found so far is registered.
3. UNTIL (requirements are met)

#### 5.3.1 Mathematical Expression

Number of employee bees = Number of onlookers = Number of solution in swarm

**Step 1:** Artificial Bee Colony generates a randomly distributed initial population of solution (SN), where SN denotes the size of employee bee or onlookers.

Each solution  $x_i$  is a D-dimensional vector where  $i = 1, 2, 3, \dots, SN$  and D is number of optimisation parameters.

**Step 2:** Then the initial fitness of population is evaluated( **In ABC, the nectar amount of a food source correspond to quality(fitness) of associated solution**). The population of solution is then subjected to repeated cycles such as employee bees, onlookers and scouts.

**Step 3:** For each employee bee, a new solution( $V_{ij}$ ) is produced by using solution search equation:

$$V_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$$

where k is 1,2,...,SN and j is 1,2,...,D are randomly generated indexes,  
 $\phi$  is a random number between [-1,1] and  
 $x_{ij}$  is food position or solution.

**Step 4:** Calculate the probability  $P_{ij}$  for solution  $V_{ij}$  by following equation:

$$P_i = \frac{f_i t_i}{\sum_{j=1}^{SN} f_j t_j}$$

where  $f_i t_i$  denotes fitness value of  $i^{th}$  solution.

For each new solution, its fitness is evaluated and then it applies greedy mechanism that is fitness value of new one is better than the previous, then employee bee would memorize the new position and forgets previous one.

**Step 5:** If a position cannot be improved further through a predetermined cycles, the food source should be abandoned.

**Step 6:** Memorize the best solution that is obtained.

**Step 7:** Repeat the cycles until termination condition is satisfied.[10]

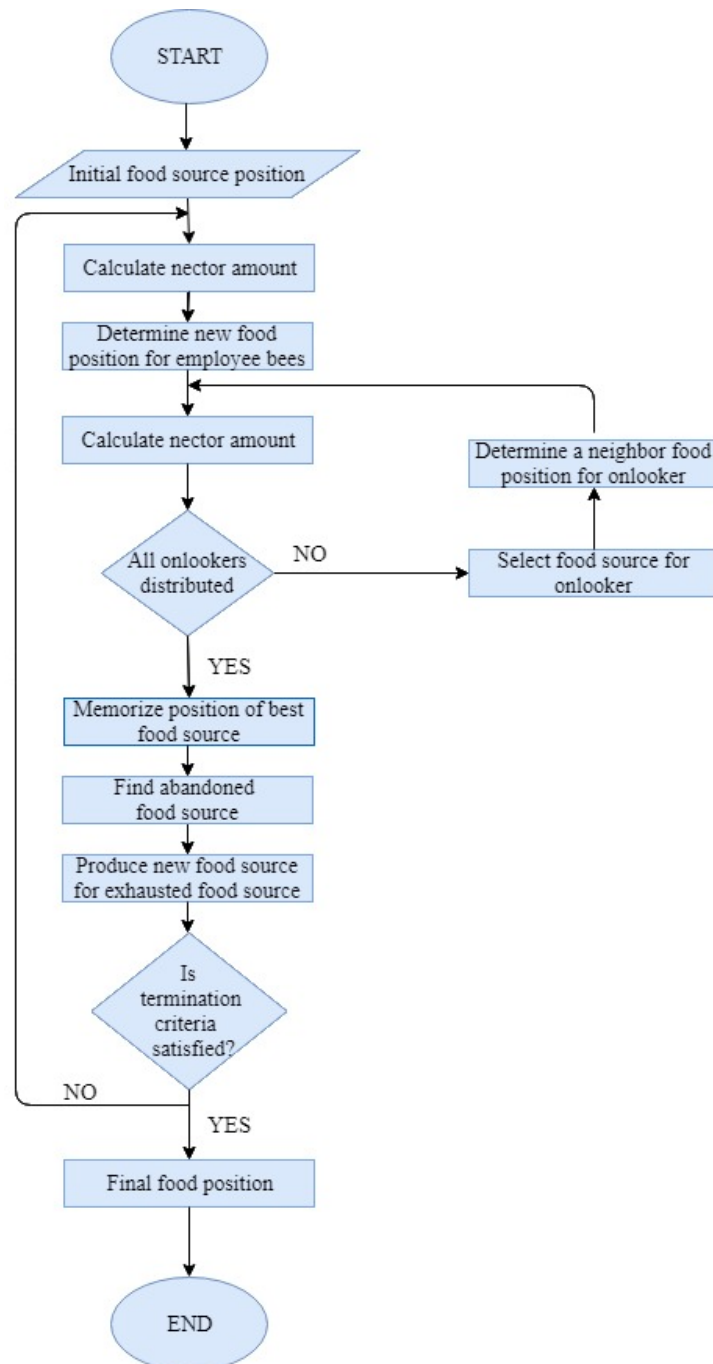


Figure 5: Flowchart of ABC algorithm

## 6 Implementation

In order to solve TSP, we have to implement each algorithm one by one to understand its working and how it is going to solve TSP. Here we are implementing the mathematical expression for each algorithm and the pseudo code for each of the algorithm is given below.

### 6.1 Pseudocode

#### 6.1.1 Particle Swarm Optimisation

---

**Algorithm 1** Particle Swarm Optimisation

---

```
1: for each particle  $i$  do
2:   Initialize particle(initial position,initial velocity,PSO parameters)
3: do until maximum iterations or minimum error criteria
4: for each particle  $i$  do
5:   Calculate fitness value
6:   if fitness value is better than pBest(history) then
7:     pBest= current fitness value
8:   if pBest is better than gBest then
9:     gBest = pBest
10: for each particle do
11:   Calculate particle velocity
12:   Use gBest and velocity to update particle data
```

---

#### 6.1.2 Ant Colony Optimisation

---

**Algorithm 2** Ant Colony Optimisation

---

```
1: Input- Instance  $x \in I$  of  $\Pi_{\text{opt}}$ 
2: Set algorithm parameters 0
3:  $i, j \leftarrow 0$ 
4: for  $j=1$  to colonies do
5:   Ant  $s_0 \leftarrow$  Create sub-colony and release agent
6:   while not-termination conditions on subcolony do
7:      $i = i + 1$ 
8:     Manage_antsactivity()
9:     Manage_Pheromone()
10:    Manage_Demon Action()
11:    Selection Procedure()
12:    Compute solution Quality()
13:   $j = j + 1$ 
14:   $S_{\text{best}} \leftarrow$  candidate to be optimal solution
15:  Update pheromone on arc()
16: Output:  $S_{\text{best}}$  "candidate" to be the best found solution  $x \in I$ 
```

---

### 6.1.3 Artificial Bee Colony

---

**Algorithm 3** Artificial Bee Colony

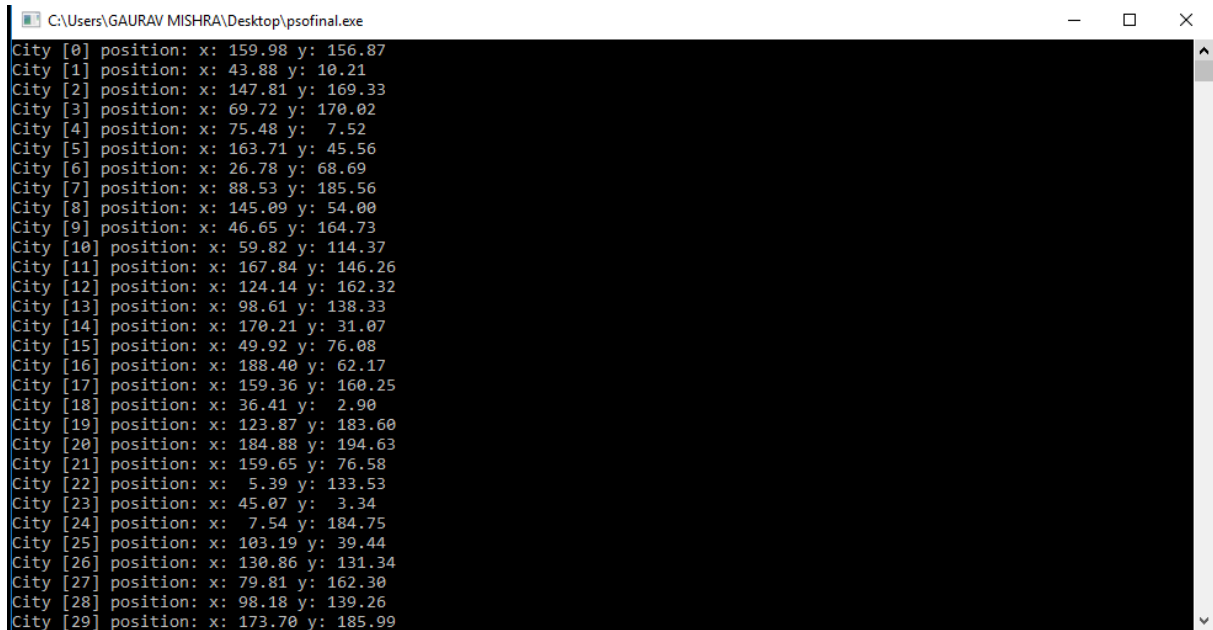
---

```
1: Load training samples
2: Generate the initial population  $z_i$ ,  $i=1, \dots, SN$ 
3: Evaluate the fitness ( $f_i$ ) of the population
4: set cycle to 1
5: repeat
6:   for each employed bees do
7:     Produce new solution  $v_i$ 
8:     Calculate the value  $f_i$ 
9:     Apply greedy selection process
10: Calculate the probability values  $p_i$  for the solutions  $z_i$ 
11: for each onlooker bee do
12:   Select a solution  $z_i$  depending on  $p_i$ 
13:   Produce new solution  $v_i$ 
14:   Calculate the value  $f_i$ 
15:   Apply greedy selection process
16: if there is an abandoned solution for the scout then
17:   replace it with a new solution which will be randomly produced
18: Memorize the best solution
19: cycle = cycle + 1
20: until cycle = MCN
```

---

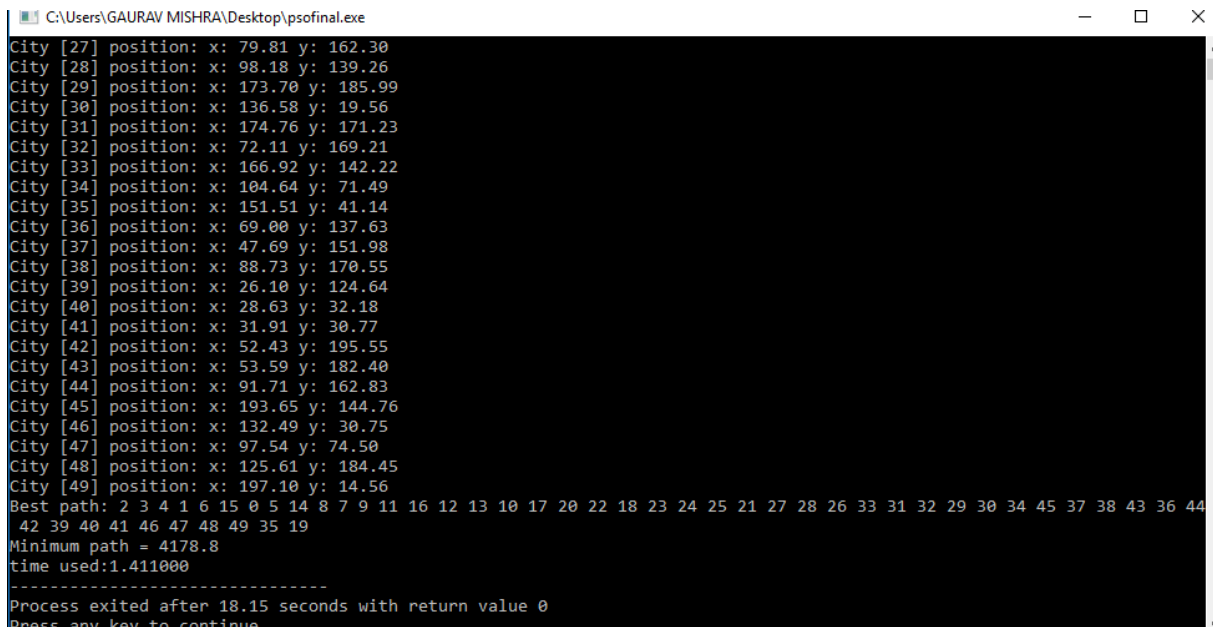


## 6.2 Output Screen



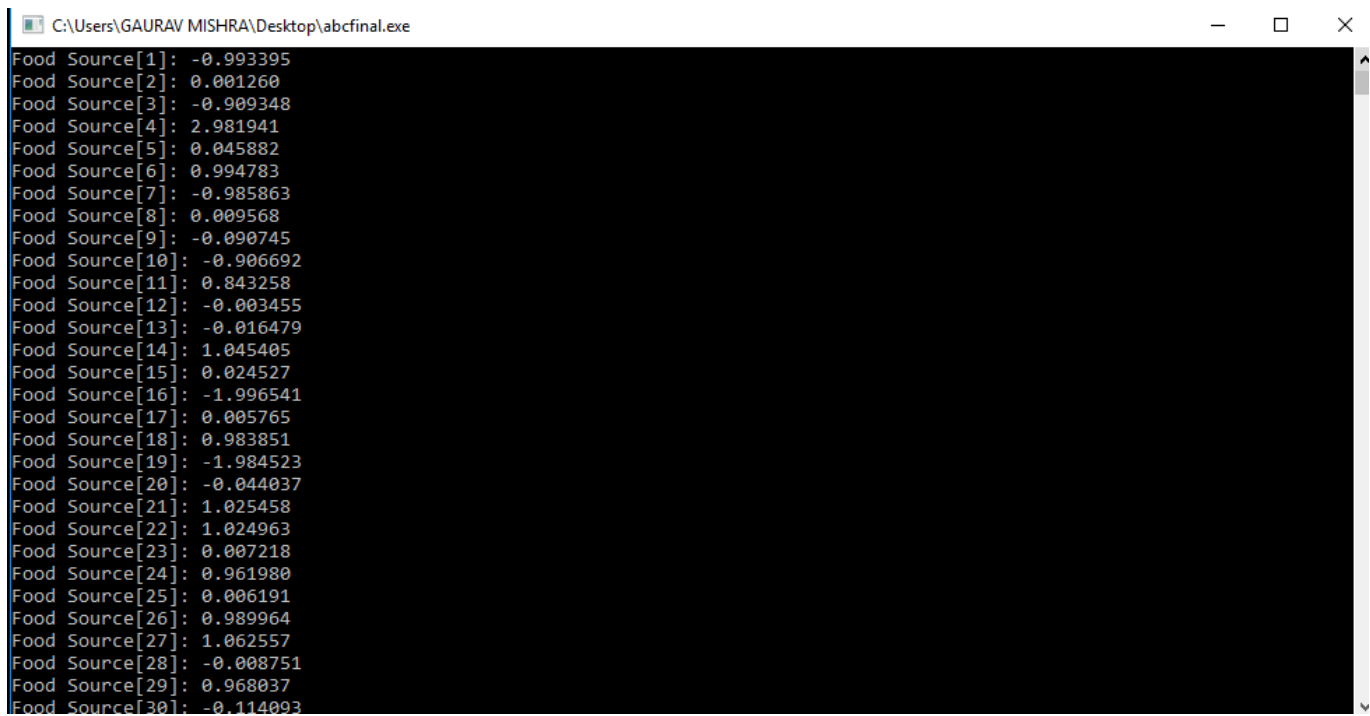
```
C:\Users\GAURAV MISHRA\Desktop\psofinal.exe
City [0] position: x: 159.98 y: 156.87
City [1] position: x: 43.88 y: 10.21
City [2] position: x: 147.81 y: 169.33
City [3] position: x: 69.72 y: 170.02
City [4] position: x: 75.48 y: 7.52
City [5] position: x: 163.71 y: 45.56
City [6] position: x: 26.78 y: 68.69
City [7] position: x: 88.53 y: 185.56
City [8] position: x: 145.09 y: 54.00
City [9] position: x: 46.65 y: 164.73
City [10] position: x: 59.82 y: 114.37
City [11] position: x: 167.84 y: 146.26
City [12] position: x: 124.14 y: 162.32
City [13] position: x: 98.61 y: 138.33
City [14] position: x: 170.21 y: 31.07
City [15] position: x: 49.92 y: 76.08
City [16] position: x: 188.40 y: 62.17
City [17] position: x: 159.36 y: 160.25
City [18] position: x: 36.41 y: 2.90
City [19] position: x: 123.87 y: 183.60
City [20] position: x: 184.88 y: 194.63
City [21] position: x: 159.65 y: 76.58
City [22] position: x: 5.39 y: 133.53
City [23] position: x: 45.07 y: 3.34
City [24] position: x: 7.54 y: 184.75
City [25] position: x: 103.19 y: 39.44
City [26] position: x: 130.86 y: 131.34
City [27] position: x: 79.81 y: 162.30
City [28] position: x: 98.18 y: 139.26
City [29] position: x: 173.70 y: 185.99
```

Figure 6: Output 1 of PSO Algorithm



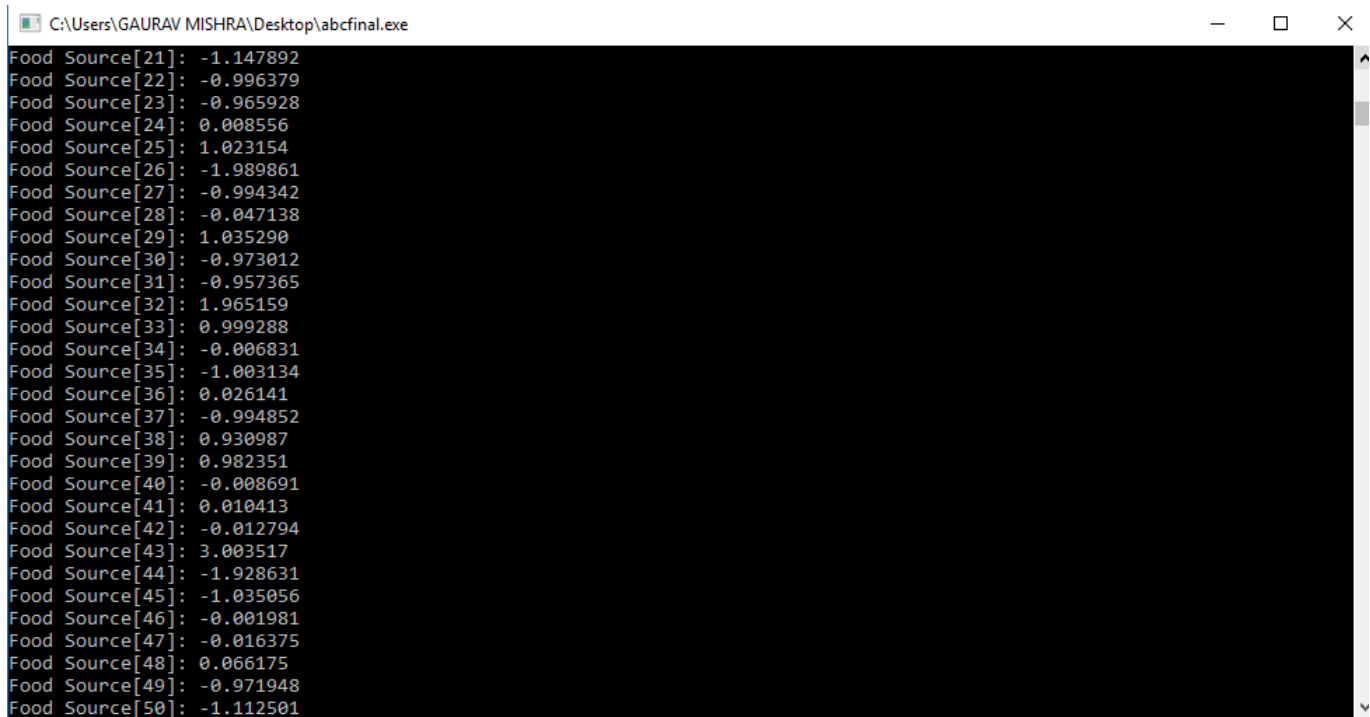
```
C:\Users\GAURAV MISHRA\Desktop\psofinal.exe
City [27] position: x: 79.81 y: 162.30
City [28] position: x: 98.18 y: 139.26
City [29] position: x: 173.70 y: 185.99
City [30] position: x: 136.58 y: 19.56
City [31] position: x: 174.76 y: 171.23
City [32] position: x: 72.11 y: 169.21
City [33] position: x: 166.92 y: 142.22
City [34] position: x: 104.64 y: 71.49
City [35] position: x: 151.51 y: 41.14
City [36] position: x: 69.00 y: 137.63
City [37] position: x: 47.69 y: 151.98
City [38] position: x: 88.73 y: 170.55
City [39] position: x: 26.10 y: 124.64
City [40] position: x: 28.63 y: 32.18
City [41] position: x: 31.91 y: 30.77
City [42] position: x: 52.43 y: 195.55
City [43] position: x: 53.59 y: 182.40
City [44] position: x: 91.71 y: 162.83
City [45] position: x: 193.65 y: 144.76
City [46] position: x: 132.49 y: 30.75
City [47] position: x: 97.54 y: 74.50
City [48] position: x: 125.61 y: 184.45
City [49] position: x: 197.10 y: 14.56
Best path: 2 3 4 1 6 15 0 5 14 8 7 9 11 16 12 13 10 17 20 22 18 23 24 25 21 27 28 26 33 31 32 29 30 34 45 37 38 43 36 44
42 39 40 41 46 47 48 49 35 19
Minimum path = 4178.8
time used:1.411000
-----
Process exited after 18.15 seconds with return value 0
Press any key to continue . . .
```

Figure 7: Output 2 of PSO Algorithm



```
C:\Users\GAURAV MISHRA\Desktop\abcfinal.exe
Food Source[1]: -0.993395
Food Source[2]: 0.001260
Food Source[3]: -0.909348
Food Source[4]: 2.981941
Food Source[5]: 0.045882
Food Source[6]: 0.994783
Food Source[7]: -0.985863
Food Source[8]: 0.009568
Food Source[9]: -0.090745
Food Source[10]: -0.906692
Food Source[11]: 0.843258
Food Source[12]: -0.003455
Food Source[13]: -0.016479
Food Source[14]: 1.045405
Food Source[15]: 0.024527
Food Source[16]: -1.996541
Food Source[17]: 0.005765
Food Source[18]: 0.983851
Food Source[19]: -1.984523
Food Source[20]: -0.044037
Food Source[21]: 1.025458
Food Source[22]: 1.024963
Food Source[23]: 0.007218
Food Source[24]: 0.961980
Food Source[25]: 0.006191
Food Source[26]: 0.989964
Food Source[27]: 1.062557
Food Source[28]: -0.008751
Food Source[29]: 0.968037
Food Source[30]: -0.114093
```

Figure 8: Output 1 of ABC Algorithm



```
C:\Users\GAURAV MISHRA\Desktop\abcfinal.exe
Food Source[21]: -1.147892
Food Source[22]: -0.996379
Food Source[23]: -0.965928
Food Source[24]: 0.008556
Food Source[25]: 1.023154
Food Source[26]: -1.989861
Food Source[27]: -0.994342
Food Source[28]: -0.047138
Food Source[29]: 1.035290
Food Source[30]: -0.973012
Food Source[31]: -0.957365
Food Source[32]: 1.965159
Food Source[33]: 0.999288
Food Source[34]: -0.006831
Food Source[35]: -1.003134
Food Source[36]: 0.026141
Food Source[37]: -0.994852
Food Source[38]: 0.930987
Food Source[39]: 0.982351
Food Source[40]: -0.008691
Food Source[41]: 0.010413
Food Source[42]: -0.012794
Food Source[43]: 3.003517
Food Source[44]: -1.928631
Food Source[45]: -1.035056
Food Source[46]: -0.001981
Food Source[47]: -0.016375
Food Source[48]: 0.066175
Food Source[49]: -0.971948
Food Source[50]: -1.112501
```

Figure 9: Output 2 of ABC Algorithm

### 6.3 Result Analysis

Three optimisation algorithms (ACO,PSO,ABC) are implemented to solve Traveling Salesman Problem for 50 cities.

But it was not possible to implement **Ant Colony Optimisation** to Solve TSP using **C** language.

#### ACO Failure Scenario:

In Ant Colony Optimization algorithm, ants are classified as objects so that each specific object performs two consecutive operations, first **edgeSelection()** and the second is **pheromoneUpdate()**. Once the edge is selected the respective pheromone density of selected edge needs to be changed. If thread implementation is taken into consideration then these two operations **edgeSelection()** and **pheromoneUpdate()** can overlap each others execution. Suppose t1 and t2 are two threads used to perform above two operations then the operation will execute in any order. The operation **pheromoneUpdate()** may execute before the **edgeSelection()** operation. Since, C is not an object oriented language so one was unable to classify ants as objects. This is the major drawback of **Ant Colony Optimization** algorithm implementation through C.

Algorithms are implemented on a total of seven systems-

System	RAM(GB)	Processor
LINUX	2	i3 5 <sup>th</sup> gen
LINUX	4	i3 5 <sup>th</sup> gen
LINUX	8	i3 5 <sup>th</sup> gen
WINDOWS	2	i3 5 <sup>th</sup> gen
WINDOWS	4	i3 5 <sup>th</sup> gen
WINDOWS	8	i3 5 <sup>th</sup> gen
MAC OS	8	i5 Mojave

Figure 10: *Systems Used*

### 6.3.1 Execution Time of Program

#### LINUX

RAM(GB)	PSO	ABC(runtime=1)	ABC (runtime=10)
2	0.65447	0.15214	1.04871
4	0.41227	0.46895	0.40810
8	0.35478	0.71245	0.68456

Figure 11: Execution Time of Algorithms in LINUX

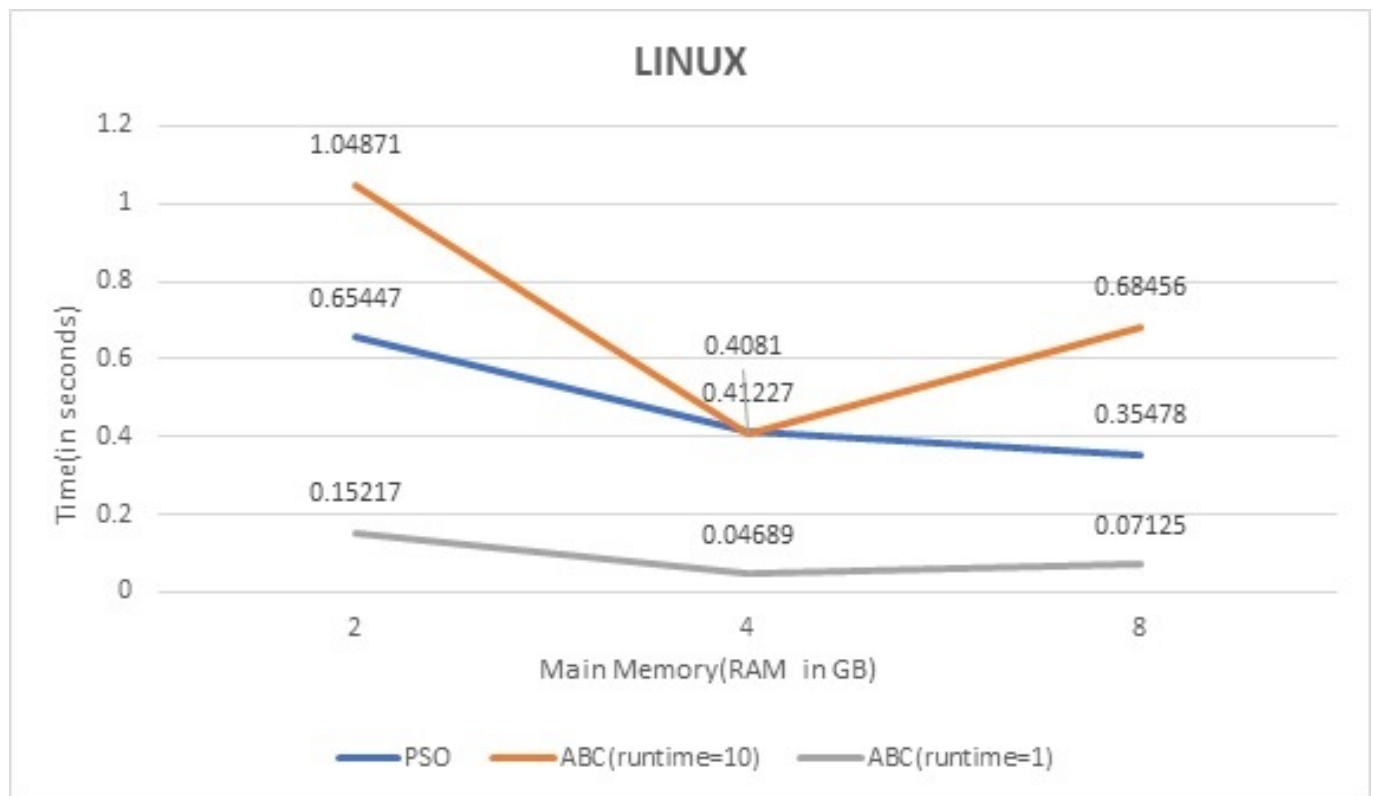


Figure 12: Execution Time of Algorithms in LINUX

## WINDOWS

RAM(GB)	PSO	ABC(runtime=1)	ABC(runtime=10)
2	3.22458	0.33330	3.14676
4	2.54000	0.25330	2.40666
8	0.57233	0.07170	0.72833

Figure 13: Execution Time of Algorithms in Windows

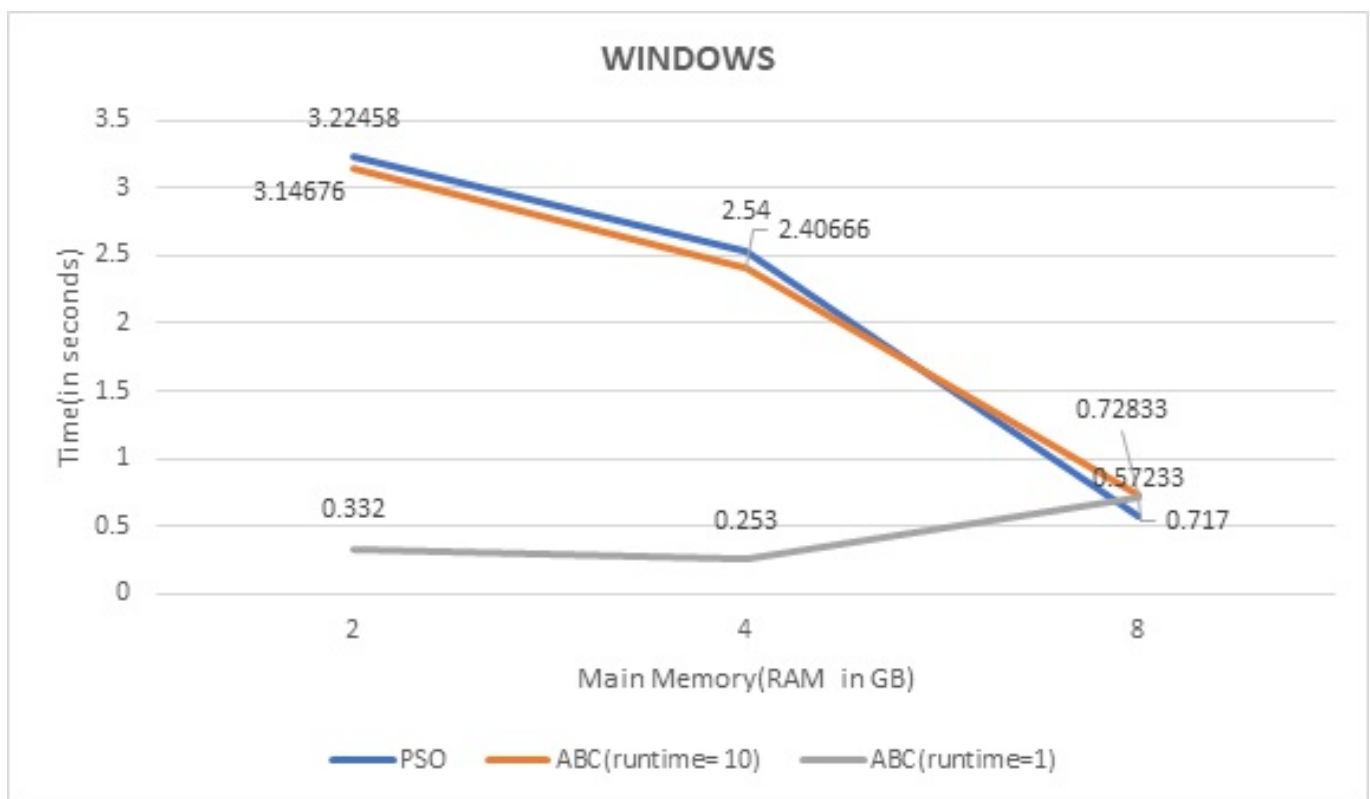


Figure 14: Execution Time of Algorithms in Windows

## MAC OS

RAM(GB)	PSO	ABC(runtime=1)	ABC(runtime=10)
8	0.592351	0.03391	0.34615

Figure 15: Execution Time of Algorithms in MAC OS

In LINUX, **Artificial Bee Colony(ABC)** took a bit more execution time than **Particle Swarm Optimisation(PSO)** when compared on a system of RAM 2gb and 8gb, whereas in case of 4gb both algorithms performs approximately same.

In Windows, **PSO** took slight a more execution time than **ABC** algorithm when implemented on a system of RAM 2gb and 4gb, whereas on increasing the RAM to 8gb **PSO** took less execution time than **ABC**.

In MAC OS, **ABC** works faster than **PSO** when implemented on a system of RAM 8gb.

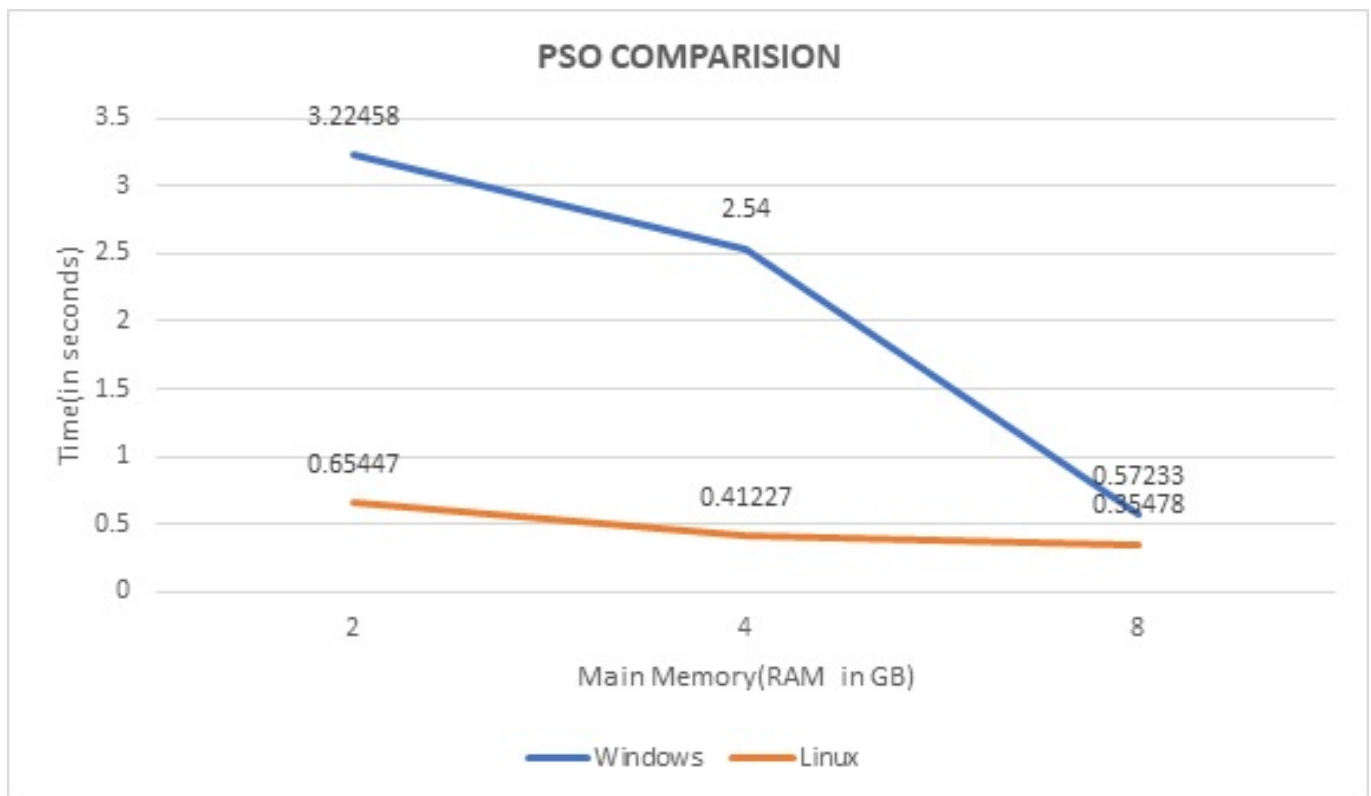


Figure 16: Comparison of PSO in Windows and LINUX

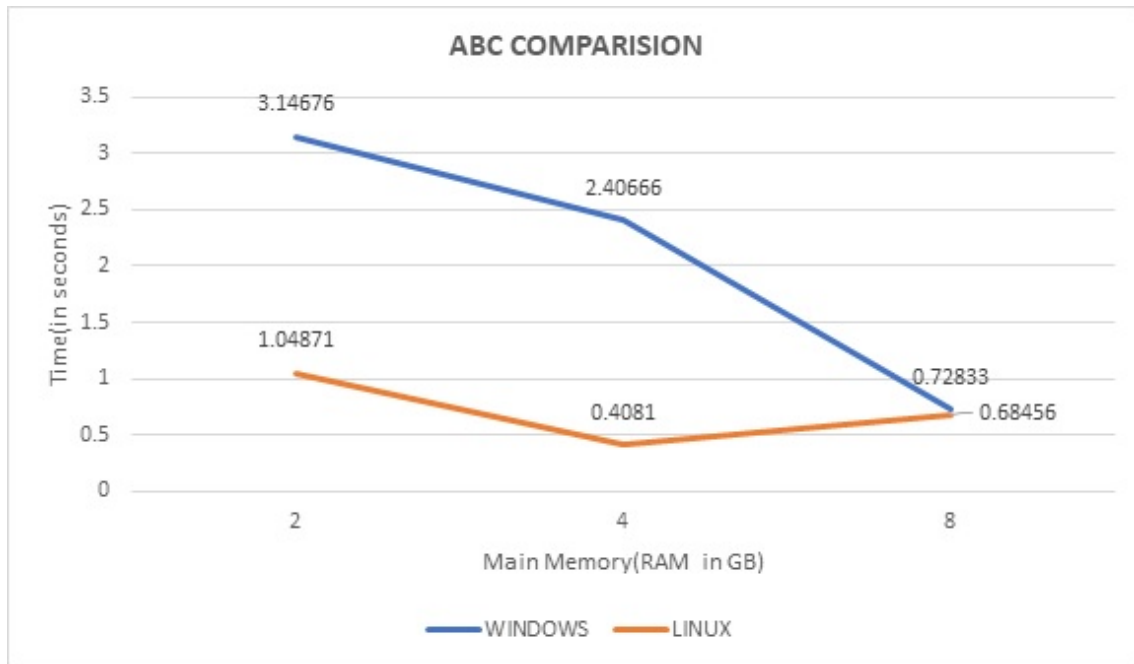


Figure 17: Comparison of ABC in Windows and LINUX

When we compare **PSO** and **ABC** on Windows and Linux, they work efficiently on Linux as compared to Windows with less execution time.

Based on the above comparison, we can say that **PSO** works better than **ABC** as **PSO** needs only one execution cycle to solve TSP whereas the execution cycle of **ABC** ranges from (10-30) which in turn will result in more execution time than **PSO** algorithm.

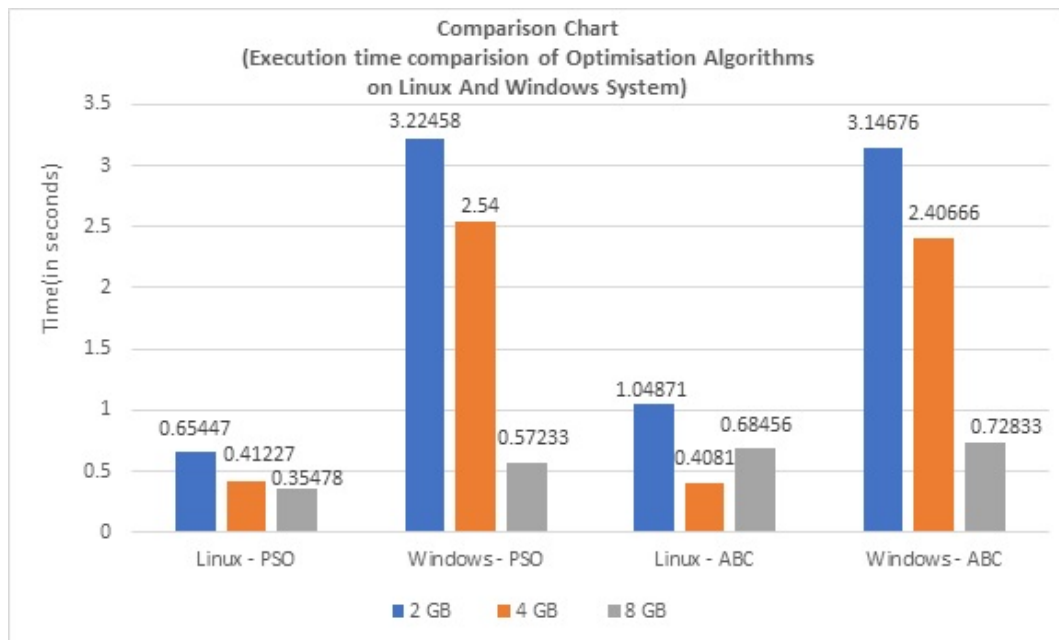


Figure 18: Comparison of PSO and ABC in LINUX and Windows

## 7 Conclusion and Future Scope

The traditional approach for solving Travelling Salesman Problem is computationally difficult for increasing number of stops whereas Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO) and Artificial Bee Colony (ABC) algorithm can easily solve Travelling Salesman Problem for a large number of stops. This proves that how swarm intelligence has acted as an inspiration paradigm for solving an optimisation problem.

Artificial Bee Colony works on the waggle dance to find food sources while Particle Swarm Optimisation works on global best position. Also, Artificial Bee Colony needs at least ten to thirty execution cycle to give better outcome while PSO does the work in one execution cycle. Location of cities is treated as food sources, where one by one each and every city are visited in the search domain of respective algorithm and avoiding repetition.

The code shows the best efficiency over LINUX platform as compared to Windows and MAC OS. Although, Particle Swarm Optimisation has worked faster than ABC a little change in a parameter can drastically affect the working of these algorithms. The best algorithm is yet to come as several variants of ABC, PSO and ABC are still under research. Proper parameters and topology selection are also one of the research areas. It has become a major concern for researchers as it provides a vast domain to work. Ant Colony Optimisation, Particle Swarm Optimisation and artificial bee colony algorithms are heuristic global optimisation which is also used in various real-life applications. Although ABC has great potential, it was clear to the scientific community that some modifications to the original structure are still necessary in order to significantly improve its performance. And also ABC can be used as an evolutionary framework into which different traditional or modern heuristic algorithmic components are integrated. ABC can be also applied for optimization in dynamic and uncertain environments.



## References

- [1] M. Alhanjouri, "International Journal of Advanced Research in Optimization Techniques for Solving Travelling Salesman Problem," no. November, 2017.
- [2] S. G. Yaseen, "Ant Colony Optimization," vol. 8, no. 6, pp. 351–357, 2008.
- [3] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," pp. 1942–1948, 1995.
- [4] S. Sobti and P. Singla, "Solving Travelling Salesman Problem Using Artificial Bee Colony Based Approach," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 6, pp. 186–189, 2013.
- [5] G. George, "Solving Travelling Salesman Problem Using Variants of ABC Algorithm," pp. 97–109, 2013.
- [6] J. V. Hook, "Application of particle swarm optimization for traveling salesman problem to lossless compression of color palette images," 2008.
- [7] V. A. Rane, "Particle Swarm Optimization ( PSO ) Algorithm : Parameters Effect And Analysis Abstract :," vol. 2, no. 7, pp. 8–16, 2013.
- [8] I. Brezina, "Solving the Travelling Salesman Problem Using the Ant Colony Optimization," no. December 2011, 2016.
- [9] D. Karaboga, B. Gorkemli, and N. Karaboga, "A comprehensive survey : Artificial bee colony ( ABC ) algorithm and applications A comprehensive survey : artificial bee colony ( ABC )," no. June, 2012.
- [10] L. Bao and J.-c. Zeng, "Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm," no. August, 2014.

## **A APPENDIX I PROJECT CODE**