MINOR PROJECT REPORT

CAPACITATED VEHICLE ROUTING PROBLEM WITH TIME WINDOWS(CVRPTW)

Under the guidance of

Ms. Parul Agrawal

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Group Members:

Sakshi gupta (17104039)

Manvi Chawla(17104041)

Adhar Agrawal(17104068)

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our mentor Ms. Parul Agrawal for teaching us, helping us with the Project as well as guiding us with the relevant resources, and ultimately providing us with an opportunity to make and present this report.

DECLARATION

We hereby declare that the project report is based on our own work carried out during the course of our study under the supervision of our mentor Ms. Parul Agrawal.

I. We assert the statements made and conclusions drawn are an outcome of our research work.

II. I further certify that the work contained in the report is original .The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Sakshi Gupta(17104039)

Manvi Chawla(17104041)

Adhar Agrawal(17104068)

TABLE OF CONTENTS

1. Introduction

We have picked Capacitated Vehicle routing problem with time windows(CVRPTW) as our project. CVRPTW is a complex combinatorial optimization problem that belongs to the NP-complete class. Hence, the use of exact optimization methods may be difficult to solve these problems in acceptable CPU times, when the problem involves real-world data sets that are very large. The vehicle routing problem comes under combinatorial problem. Hence, to get solutions in determining routes which are realistic and very close to the optimal solution, we use heuristics and meta-heuristics. In this paper we discuss the various exact methods and the heuristics and meta-heuristics used to solve the problem.

We were inspired by this problem because of the challenges we saw that the various delivery agents have to come across with. Customers nowadays prefer to order only from those websites where they will get their delivery on time. To survive in such a competitive market, a company must provide delivery as soon as possible in the least possible cost.

To save time and for good customer satisfaction, they need to identify an optimized route for delivery. Some algorithms consider either only cost or deliver goods according to the dates on which the customer ordered.

We will be using nature-inspired algorithms to find an optimized route for different vehicles with different capacities so that they can deliver goods on time with the least cost possible.

We will be hybridizing different nature inspired algorithms and will be comparing costs and computational time for all datasets.

Different applications involve:

1)Waste collection

2) School bus drop

3) Delivery system

2. PROBLEM STATEMENT

Our challenge is to find "What is the optimal set of routes for a delivery agent to traverse in order to deliver to a given set of customers?".The VRP concerns the service of a delivery company.

We are provided with latitude and longitudes of different customer's location. We have to conclude which algorithm works best for our problem.

While researching about our problem statement, we came across different variants of vehicle routing problem which are mentioned below:

1) Capacitated vehicle routing problem (CVRP)

2) Split delivery vehicle routing problem (SDVRP)

3) Stochastic vehicle routing problem (SVRP)

4) Vehicle routing problem with time windows (VRPTW)

In our project, we have combined all these aspects mentioned above in a single problem statement using different nature inspired algorithms-genetic algorithm(GA) ,particle swarm optimization(PSO), ant bee colony(ABC), hybridized algorithm of PSO and ABC and hybridized algorithm of GA with GA

We have also implemented a hybridized algorithm of Particle Swarm Optimization algorithm and Ant Bee Colony algorithm and also a hybridized algorithm using Genetic algorithm with Genetic algorithm itself.

At the end, we are comparing the algorithms by applying different instances on them and plotting a bar graph of cost vs algorithm for each instance.

2.1 Objectives

1)Minimize the transportation cost based on the distance

2)Improved customer satisfaction by delivering goods on time

2.2 Constraints

1)Different number of vehicles

2)Different number of customers

3)Different vehicle capacity

4)Each customer is ordering different amount of goods

5) Each Customer wants Delivery at different time

6) Each route is a tour which starts from a depot, visits a subset of the customers, and ends at the same depot.

4.1 CVRP USING GENETIC ALGORITHM

4.1.1 INTORDUCTION OF ALGORITHM

In computer science and operation research , a **genetic algorithm** (**GA**) is a meta-heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problem by relying on bio-inspired operators such as mutation crossover and selection.

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotype) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype which can be mutated and altered. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encoding are also possible.

A typical genetic algorithm requires:

1)A genetic of the solution domain
2)A fitness function to evaluate the solution domain.

Like in our algorithm,We are generating 50 new random chromosomes. We are creating a 2d array in which first column comprises of random numbers and second column comprises of customers. Then we are sorting our 2d array on the basis of first column which leaves us with a new set of chromosome everytime.

4.1.2 STEPS INVOLVED IN GENETIC ALGORITHM

Crossover

Among these 50 chromosomes,40 chromosomes undergo crossover. In crossover two parents are selected.

p1=[1,3,4,5,2,7,6]

p2=[2,3,4,1,5,7,6]

The starting position for crossover is index 2 and ending is 5. It selects the cities from 2 to 5 in p1 i.e [4,5,2,7]. It then delete these cities in p2 and p2 becomes [3,1,6]. Then these deleted elements are appended at the end of p2 which generates a new child.

Child=[3,1,6,4,5,2,7]

Above is just an example with small size of data set. For our algorithm, starting position is 2 and ending position is 18. The crossover factor here is 0.8.

Mutation

The 10 chromosomes undergo mutation process. In the mutation process a parent generates a new child.

p1=[1,3,2,4,6,5,7]

It starts with index 3 and deletes the object till index 5. The child becomes [1,3,2,7] and then the rest of the elements are appended at end.

Child=[1,3,2,7,4,6,5]

For our algorithm, starting position is 3 and ending position is 10.The mutation factor is 0.2.

Calculate cost

Then we calculate the cost of all these 100 chromosomes. The cost is calculated by distance between various customers with the help of latitude and longitudes.

Since,different customers order different amount and each vehicle has different capacity. We assign each vehicle a set of customers according to its delivering capacity. We then check in our data set whether the customer demanded the order in morning or evening and added penalties accordingly

Fitness function

The fitness function is calculated as below:

Alpha*timecost+ beta*distancecost ,where alpha and beta are two randomly generated values

0<alpha<1

0<beta<1

Less the fitness function, better the route is.

Then we select the best 50 chromosomes among these 100 chromosomes and best 50 undergoes the same process for next generation

Each generation prints the best route calculated, its cost and its fitness function value.

We continue the process for 50 generations

Selection of fittest chromosomes

We are using a combination of tournament and stochastic solution for selecting best parents among 50 chromosomes to undergo crossover and mutation.

In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent. We are using first 10-way tournament selection on our population

In a roulette wheel selection, the circular wheel is made. A fixed point is chosen on the wheel circumference and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated. We will be using stochastic selection which is an extended version of roulette wheel in which all the parents are chosen in one spin of the wheel.

Implementation wise, we used the following steps

- Calculate S = the sum of a finesses.
- Generate a random number between 0 and S.
- Fixedpoint = random.uniform(0,sum)
- Starting from the top of the population, keep adding the finesses to the partial sum P, till P<S.
- if(partialsum > fixedpoint):
- bestone[p]=array[i]
- count=count+1
- p=p+1
- The individual for which P exceeds S is the chosen individual

In this way, we are selecting the best chromosomes for our algorithm. The chromosomes with the best fitness function and least cost are selected. The same process is repeated for mutation also while selecting one parent.

4.1.3 FLOWCHART

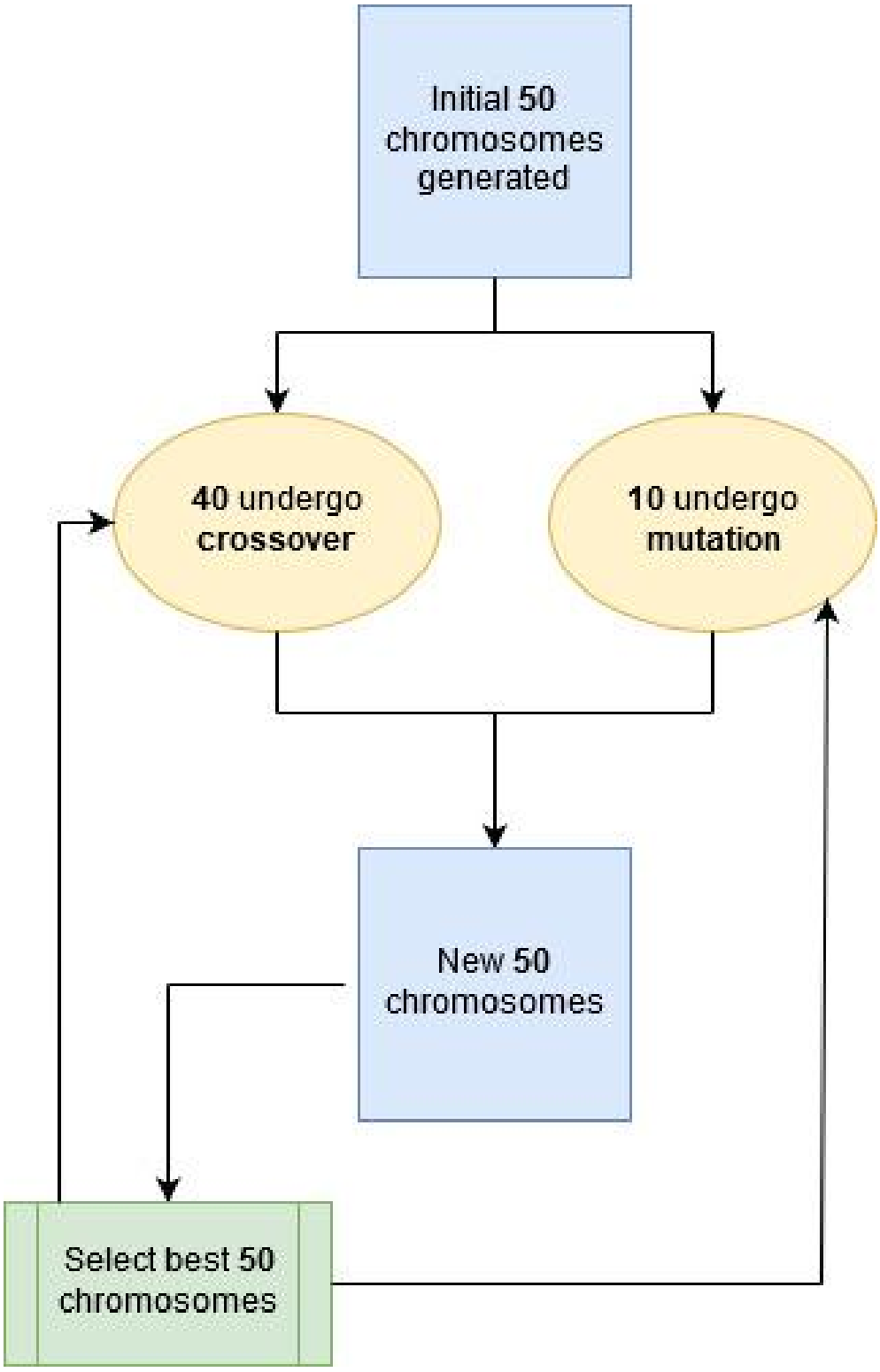

Fig 1. Flowchart of Genetic Algorithm

4.2 CVRP USING PARTICLE SWARM OPTIMIZATION ALGORITHM

4.2.1 INTRODUCTION OF ALGORITHM

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr.Eberhart and Dr. Kennedy in 1995, inspired by social behaviour of bird flocking or fish schooling.

PSO simulates the behaviours of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called P_{best}. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called G_{best}. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called l_{best}.

After finding the two best values, the particle updates its velocity and positions with following equations

$$v[t] = v[t-1] + c1 * rand() * (P_{best} - x[t-1]) + c2 * rand() * (G_{best} - x[t-1])$$

$$x[t] = x[t-1] + v[t]$$

4.2.2 STEPS INVOLVED IN PARTICLE SWARM OPTIMIZATION ALGORITHM

VELOCITY OF PARTICLE

The velocity of particle is represented as a tuple of 3 constraints i.e

V(swap operator ,c1 ,c2)

Where, c1 and c2 are the probability of finding a particle and values lie between 0 and 1.

POSITION UPDATE

x(t) = x(t-1) + v(t)
Example:

(3,1,2,4,5) = (1,3,2,4,5) + SO(1,2)

VELOCITY UPDATE

v[new] = v[old] + c1 * rand() * (P_{best} - x[t-1]) + c2 * rand() * (G_{best} - x[t-1])

i.e merging two swap sequences

the probability that all the swap operators in swap sequence (P_{best} –x(t-1)) are include in the updated velocity is c1.

the probability that all the swap operators in swap sequence (G_{best} –x(t-1)) are include in the updated velocity is c2.

PSEUDO CODE

-Random initialization of possible path.

-For each time step:

Update G_{best}

Update P_{best}

-For each city:

v[t] = v[t-1] + c1 * rand() * (P_{best} - x[t-1]) + c2 * rand() * (G_{best} - x[t-1])

x[t] = x[t-1] + v[t]

4.2.3 FLOWCHART

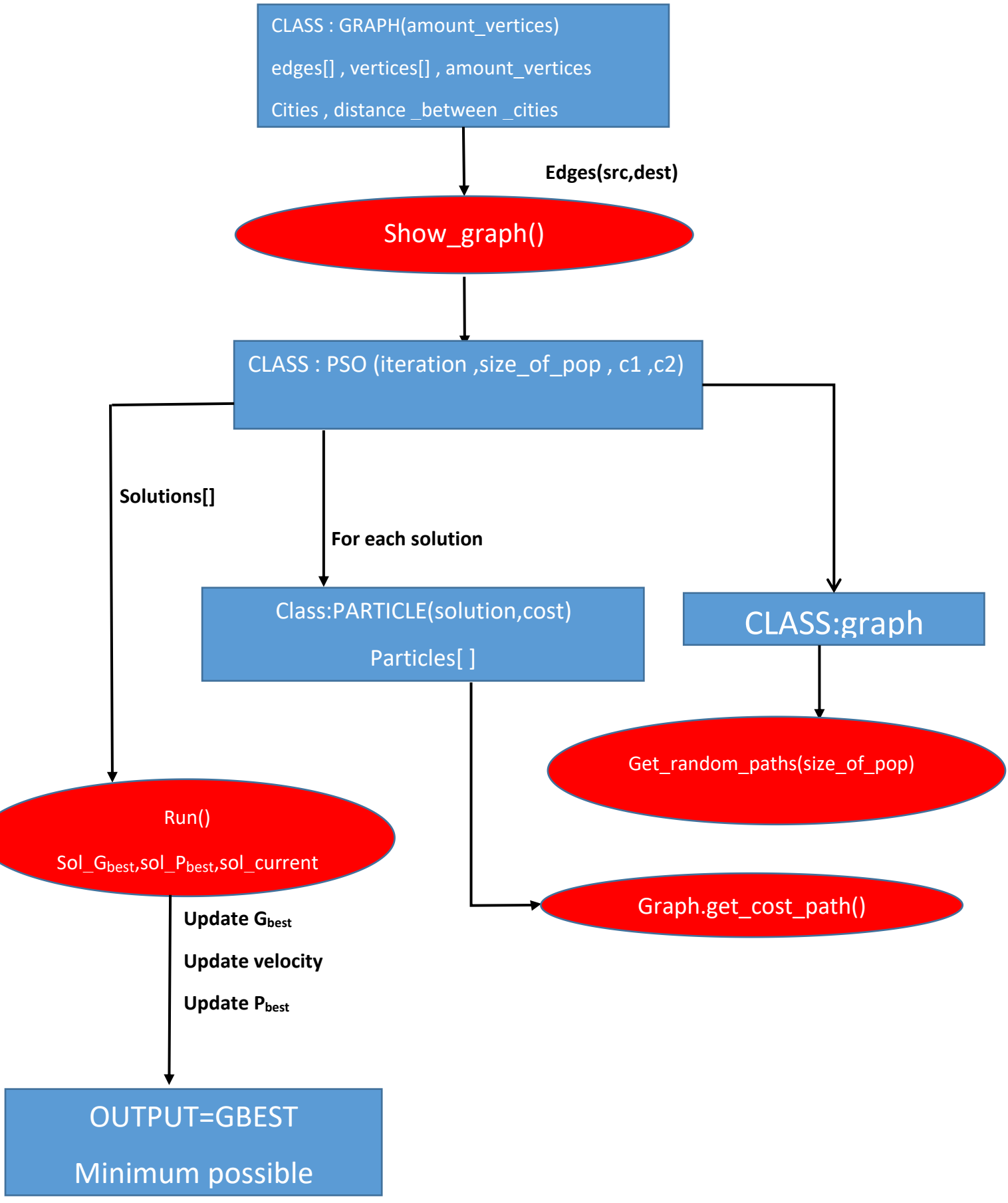

Fig 2. Flowchart of Particle Swarm Optimization Algorithm

4.3 CVRP USING ANT BEE COLONY ALGORITHM

4.3.1 INTRODUCTION OF ALGORITHM

Ant bee colony (ABC) is a new population-based stochastic algorithm that has shown good search abilities on many optimization problems. However, the original ABC shows slow convergence speed during the search process. In order to enhance the performance of ABC, this paper proposes a new artificial bee colony algorithm, which modifies the search pattern of both employed and onlooker bees. A solution pool is constructed by storing some best solutions of the current swarm. New candidate solutions are generated by searching the neighborhood of solutions randomly chosen from the solution pool.

ABC is a new swarm intelligence algorithm that is inspired by the behavior of honey bees. Since the development of ABC, it has been applied to solve different kinds of problems. Similar to other stochastic algorithms, ABC also faces up some challenging problems. For example, ABC shows slow convergence speed during the search process. Due to the special search pattern of bees, a new candidate solution is generated by updating a random dimension vector of its parent solution. Therefore, the offspring (new candidate solution) is similar to its parent, and the convergence speed becomes slow.

Moreover, ABC easily falls into local minima when handling complex multi nodal problems. The search pattern of bees is good at exploration but poor at exploitation. However, a good optimization algorithm should balance exploration and exploitation during the search process.

4.3.2 STEPS INVOLVED IN ANT BEE COLONY ALGORITHM

1) Each ant is assigned a random city to begin.

2) Each ant chooses next city on the basis of pheromone and visits that city and deposits pheromone there.

The pheromone to add:

pheromone_to_add = self.pheromone_deposit_weight / distance

3) Each ant updates global best if it has a better solution with it

4) This algorithm uses 2 parameters: Colony size and steps

5) The selection of next city is based on roulette wheel

Heuristic total= Adding the distances between all the edges

Roulette wheel= Pow((Pheromone from origin to unvisited node),alpha)
*Pow((Heuristic total/Distance between origin to unvisited node),beta)

4.3.3 FLOWCHART

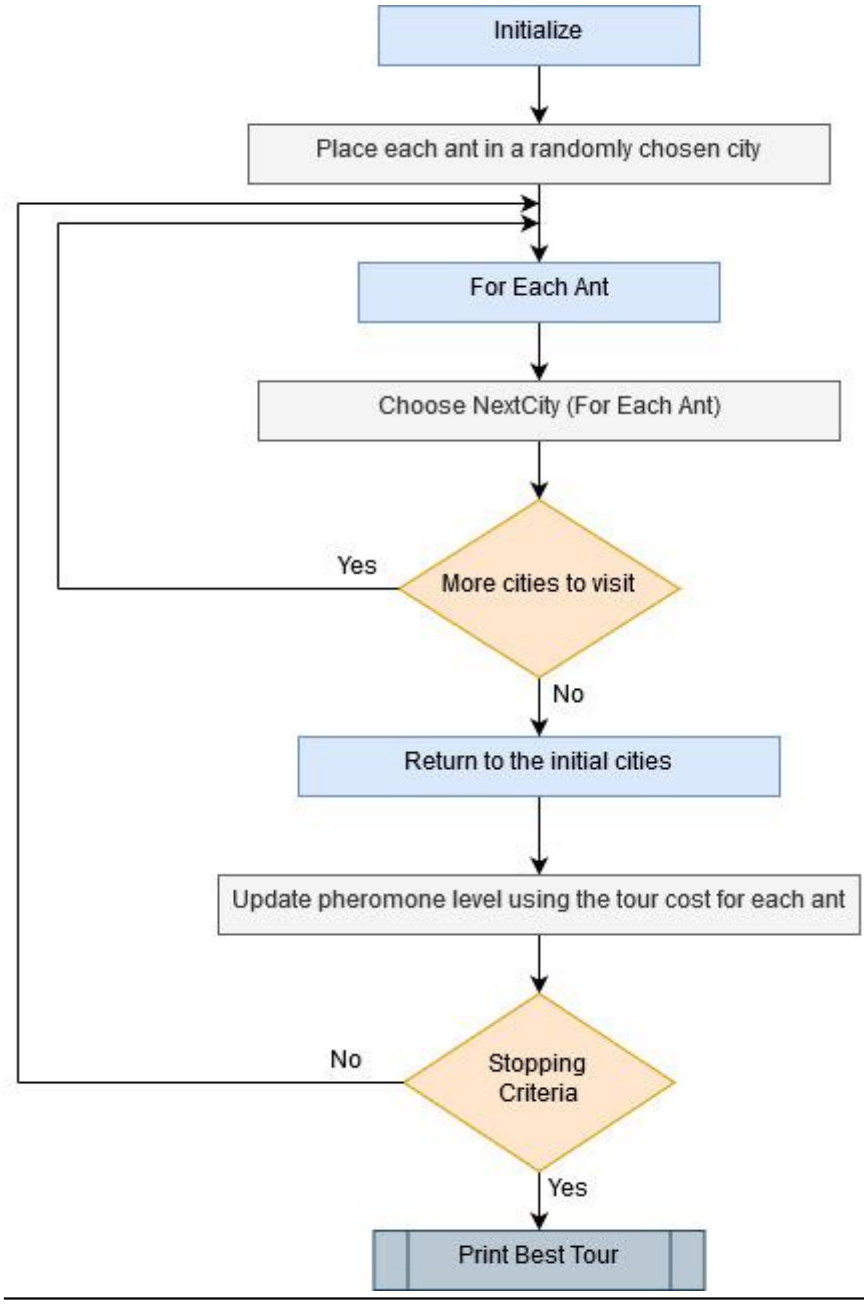

Fig 3. Flowchart of Ant Bee Colony Algorithm

4.4 PACO ALGORITHM

4.4.1 INTRODUCTION OF ALGORITHM

The PACO Algorithm is the name given to a hybridized algorithm using Particle Swarm Optimization algorithm and the Ant Bee Colony algorithm.In the proposed algorithm, each artificial ant, like a particle in PSO, is allowed to memorize the best solution ever found. After solution construction, only elite ants can update pheromone according to their own best-so-far solutions. Moreover, a pheromone disturbance method is embedded into the ACO framework to overcome the problem of pheromone stagnation. Two sets of benchmark problems were selected to test the performance of the proposed algorithm. The computational results show that the proposed algorithm performs well in comparison with existing swarm intelligence approaches. The PACO algorithm incorporates the merits of PSO into the ACO algorithm. One of the advantages of applying ACO to the CVRP is that ACO can cluster customers and build routes at the same time. However, laying pheromone (long-term memory) on trails as ant communication medium is time consuming. The merit of PSO is that it can speed convergence through memorizing personal and global best solutions to guide the search direction. Inspired by the merit of PSO, the PACO algorithm allows artificial ants to memorize their own best solution so far and to share the information of swarm best solution.

Reasons why we have hybridized PSO and ABC Algorithm-

1. Many other algorithms (other than the used ones)[1] perform the so-called single-starting-point search and thus their performance relies highly on a good initial solution. However, GA, PSO, and ABC are all population-based algorithms and can start the search from multiple points. Their initial solutions have little influence on their performance. Thus, we consider adopting population-based algorithms to solve CVRPs.
2. PSO and ABC have a memory that enables the algorithms to retain the knowledge of good solutions, while the genetic operators of GA may destroy previously learned knowledge when producing the offspring. In view of these two considerations, we select PSO and ABC as the solution for CVRP.
3. The PSO algorithm uses the least computational time and it could easily be combined with ABC algorithm in comparison to genetic algorithm which uses a high computational power.
4. Both algorithms hybridized are already providing us with good results.

4.4.2 STEPS INVOLVED IN PACO ALGORITHM

During the searching process, artificial ants construct solution routes, memorize the best solution ever found, and lay pheromone on the routes of swarm and personal best solutions. To prevent being trapped in local optima and to increase the probability of obtaining better solutions, PACO performs pheromone(long-term memory) disturbance and short-term memory resetting operations to adjust stagnated pheromone trails. Disturbed pheromone trails guide ants to find new P_{best} and G_{best} solutions.

The merits of PSO adopted in PACO can speed convergence during a run, even after pheromone (long-term memory) disturbance operations. Computational results show that the performance of PACO is competitive in terms of solution quality when compared with existing ACO- and PSO-based approaches. We have also modified the existing PACO Algorithm with time windows and having multiple depots. The results that we have achieved will be of great use in future. These results will bring a change in CVRP in the future.

STEPS

1) Initialization (Initialize all parameters).
2) Let m ants construct solution routes.
3) The top r best ants perform local search.
4) Update the G_{best} and P_{best} solutions of ants and select the r elite ants.
5) If G_{best} is not improved within successive iterations, go to Step 6; otherwise, go to Step 7.
6) Randomly disturb the pheromone matrix, reset the P_{best} solutions for some ants, and go to Step 8.
7) Update the pheromone matrix based on the P_{best} solutions of elite ants.
8) If iteration number reaches the maximum number of iterations (MaxIte), go to Step 9; otherwise, go to Step 2 for the next iteration.
9) Output G_{best}, the best solution ever found.

We have combined ant bee colony elitist mode and particle swarm to obtain good results. For comparison of results, refer to section 6.

4.4.3 FLOWCHART

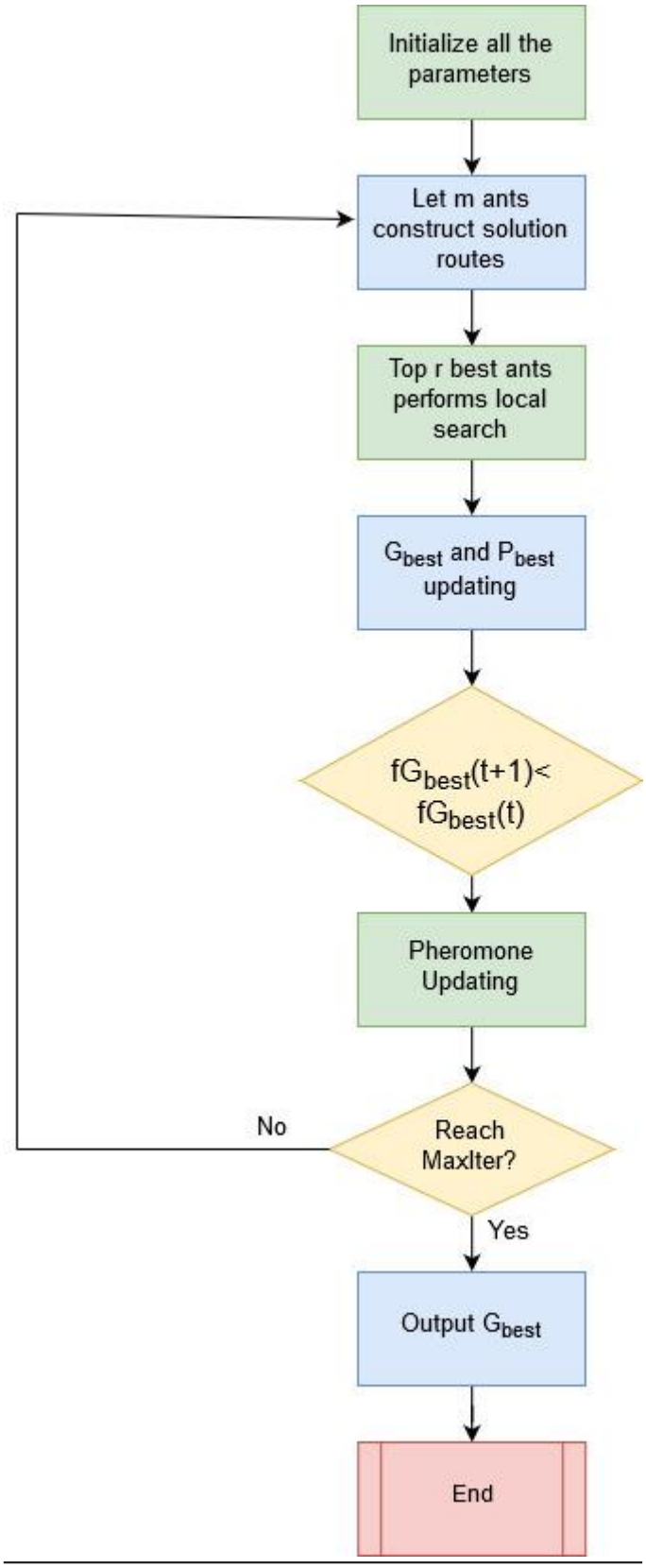

Fig 4. Flowchart of PACO Algorithm

4.5.1 Genetic hybrid algorithm

In this algorithm, we have optimized our genetic algorithm. The basic genetic algorithm is run for 50 generations. Then the best chromosome is stored for each generation. As a result, we get 50 best of best chromosomes.
Then these 50 chromosomes are passed in generation 51 of genetic algorithm and generates new 50 random chromosomes by mutation and crossover for 51st generation. Then 50 best chromosomes among these 100 are again passed into next generation. The algorithm runs for 49 more generations. And the final chromosome which we obtain is our final result.

4.5.2 FLOWCHART

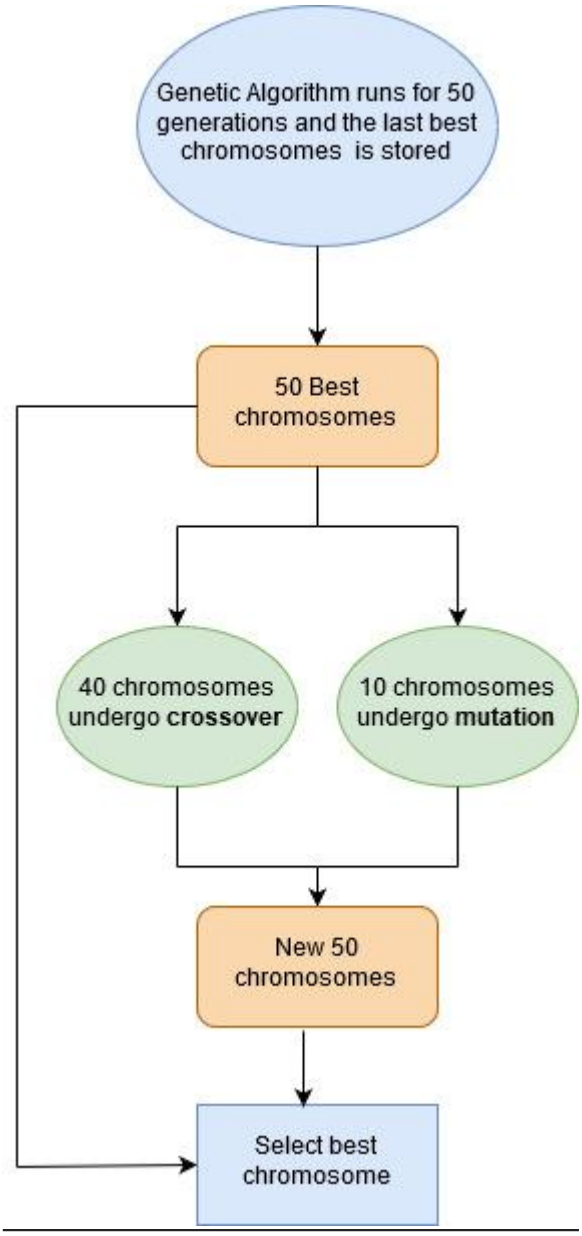

Fig 5. Flowchart of Hybridized Genetic Algorithm with Genetic Algorithm

5. EXPERIMENTAL STUDY

The specific requirements of the project are as follows-

1. HARDWARE INTERFACE- The hardware interface will include standard I/O hardware available in a desktop PC.

2. SOFTWARE INTERFACE- The software interface will include python version: 3.7 and python libraries such as-

a. Numpy

b. Matplotlib- For plotting latitudes and longitudes on graph

c. Geoplot- For plotting on real world map

d. Math- Computing squares and power

e. Time- For computing computational time for each algorithm

6.RESULTS AND ANALYSIS

INSTANCE	GENETIC ALGORITHM		PARTICLE SWARM OPTIMISATION		ANT COLONY OPTIMISATION			HYBRID(GA-GA)		HYBRID(PSO-ABC)	
	COST	TIME	COST	TIME	COST ACS	COST Ellitist	TIME	COST	TIME	COST	TIME
	40903.8	306.85	39398.2	6.18	32070.8	33152.8	447.09	40070.8	642.58	32676.5	158.68
	40298.1	322.28	40544.5	6.23	33754.9	32240.9	355.55	41754.9	675.78	31246.3	159.90
1	40766.4	340.09	41732.3	5.93	31563.3	34421.7	445.89	42563.3	634.56	31692.0	112.92
	40818.9	312.89	39696.6	5.40	30896.9	32199.7	460.9	40896.9	608.96	33469.2	133.97
	40777.9	350.78	41800.9	5.79	31555.7	32417.8	389.09	40555.7	635.45	35529.2	136.37
Avg	40766.4	322.23	38098.8	5.46	32456.7	33456.7	345.56	40987.6	657.86	32456.7	134.56
	46403.7	307.46	43099.1	6.75	36752.9	35407.1	430.99	43099.1	685.25	35603.6	138.96
2	42673.9	366.50	40633.8	5.75	34456.8	34567.8	406.33	44633.8	682.34	34526.7	132.45
	43411.0	389.00	42300.0	5.78	33456.0	35467.0	423.00	41300.0	676.54	32456.6	140.32
	47899.9	309.78	42300.0	6.31	36543.0	34652.0	427.22	42300.0	679.65	33542.8	134.45
	46401.2	366.50	41123.7	5.90	34526.7	32345.0	423.34	42400.0	623.65	33234.8	144.32
Avg	45345.6	350.43	40623.9	5.90	33452.6	33456.5	423.56	42435.7	634.56	32456.7	137.89

	40011.3	322.43	37152.7	6.31	30070.8	31370.8	422.43	31170.8	441.45	30811.5	156.29
3	38256.9	345.76	37156.9	5.90	31754.9	31754.9	445.76	31754.9	417.54	29925.3	154.89
	43411.0	340.09	39937.0	5.75	31563.3	32563.3	440.09	31563.3	415.63	31429.0	138.90
	47899.9	312.89	42336.9	5.78	30896.9	31896.9	412.89	30896.9	408.96	28343.9	166.79
	46401.2	378.7	46401.2	6.23	31555.7	32555.7	478.7	31555.7	415.55	33999.2	150.11
Avg	39876.5	350.67	38765.4	5.68	30896.9	31896.9	456.3	30896.9	408.96	32456.7	143.34
	32117.7	283.00	31752.7	6.31	22070.8	26070.8	381.00	31070.8	481.00	27505.9	166.79
	32867.6	276.71	27156.9	5.90	22754.9	25754.9	346.71	32754.9	446.71	29925.3	143.09
4	43411.0	389.00	30937.0	5.75	24563.3	26563.3	389.00	34563.3	489.00	24117.8	145.90
	47899.9	309.78	30336.9	5.78	23896.9	24896.9	359.78	33896.9	459.78	28999.0	159.09
	46401.2	299.9	26401.2	6.23	21555.7	21355.5	379.9	31555.7	479.9	32008.9	160.9
Avg	40401.2	289.9	24401.2	6.21	23896.9	25896.9	345.6	32896.9	445.6	25435.6	154.6
	47099.1	222.43	47152.7	6.31	35070.8	34070.8	383.00	45099.1	222.43	31070.8	183.00
	46633.8	345.76	45156.9	7.90	32754.9	33754.9	376.71	43633.8	345.76	31754.9	176.71
5	46300.0	349.09	40937.0	6.75	34563.3	35563.3	389.00	42300.0	349.09	32563.3	189.00
	44300.0	302.89	40336.9	6.09	33896.9	34896.9	309.78	41300.0	302.89	31896.9	109.78
	45123.7	278.7	49401.2	6.56	35555.7	36555.7	399.9	42123.7	278.73	30555.7	199.9
Avg	45634.7	354.67	45637.8	6.76	32896.9	33896.9	345.6	43453.6	287.6	30896.9	145.6

6.1.1 GENETIC ALGORITHM RESULT

```
best fitness is=>
14575.03236796322
cost of best element is
40814.57935264458
best route is=>
[  4  19  84  67  30  86  32  64  72  13  17  89   7  25  46  82   9  41
  33  69  76  92  36  91  29  78  10  14  42  49  59  58  93  70  47  95
   1  15  31  51  63  16  21  56  85  61  65  90  12  68  97  18  27  79
  94  80  73  81  75  24  22  34  43  54  37  48  53  87  74  77   5  62
   3  23  50 101   6  35 100  44   2  83  55  71  96  66  98  20  11  26
  57  99  28  38  39  88  45  60   8  40  52]
route
[[ 0  0  0 ...  0  0  0]
 [ 4 19 84 ...  0  0  0]
 [30 86 32 ...  0  0  0]
 ...
 [79 81 83 ...  0  0  0]
 [75 24  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]]
('AND THE FINAL COST FOR OUR SOLUTION IS', 40814.57935264458)
```

Fig 6. Genetic Algorithm

6.2.1 GRAPH OF GENETIC ALGORITHM

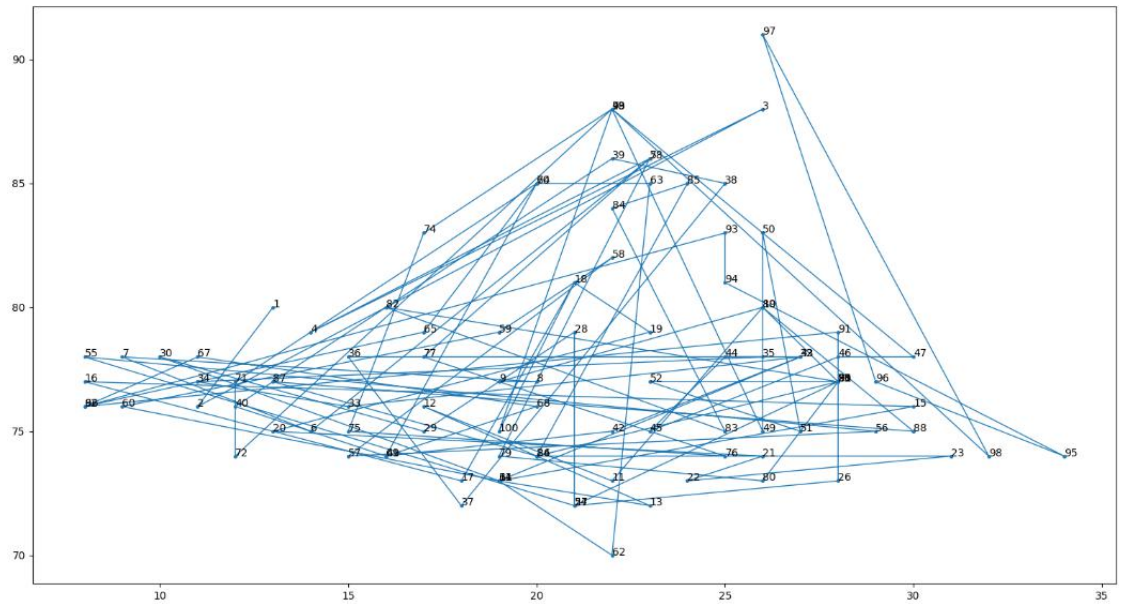

Fig 7. Graph of Genetic Algorithm

6.1.2 PARTICLE SWARM OPTIMIZATION ALGORITHM RESULT

```
route
[[ 0  0  0 ...  0  0  0]
 [93 78 53 ...  0  0  0]
 [82 30 32 ...  0  0  0]
 ...
 [33 40 23 ...  0  0  0]
 [98  5  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]]
AND THE FINAL COST FOR OUR SOLUTION IS
40296.83469439252
```

Fig 8. Particle Swarm Optimization Algorithm

6.2.2 GRAPH OF PARTICLE SWARM OPTIMIZATION ALGORITHM

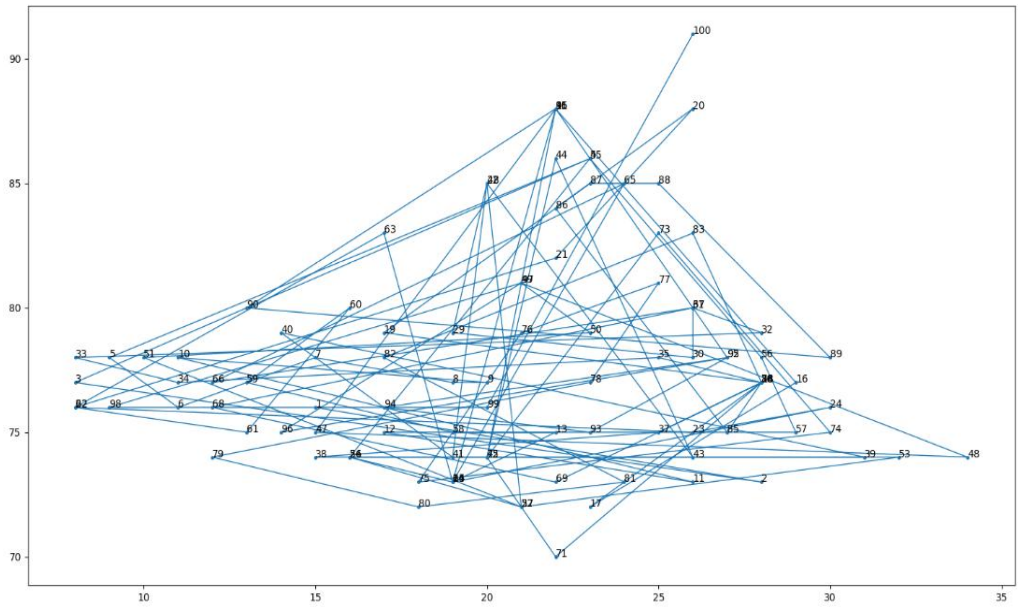

Fig 9. Graph of Particle Swarm Optimization Algorithm

6.1.3 ANT BEE COLONY ALGORITHM RESULT

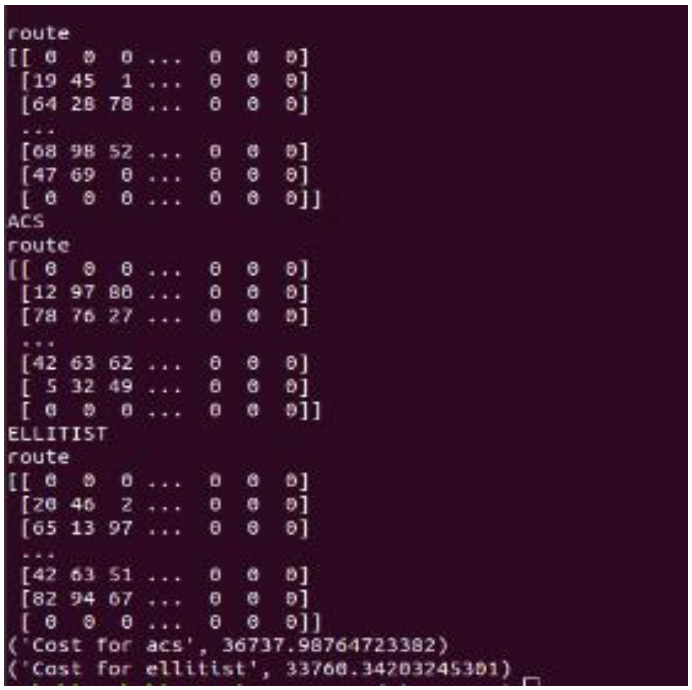

Fig 10. Ant Bee Colony Algorithm

6.2.3 GRAPH OF ANT BEE COLONY ALGORITHM

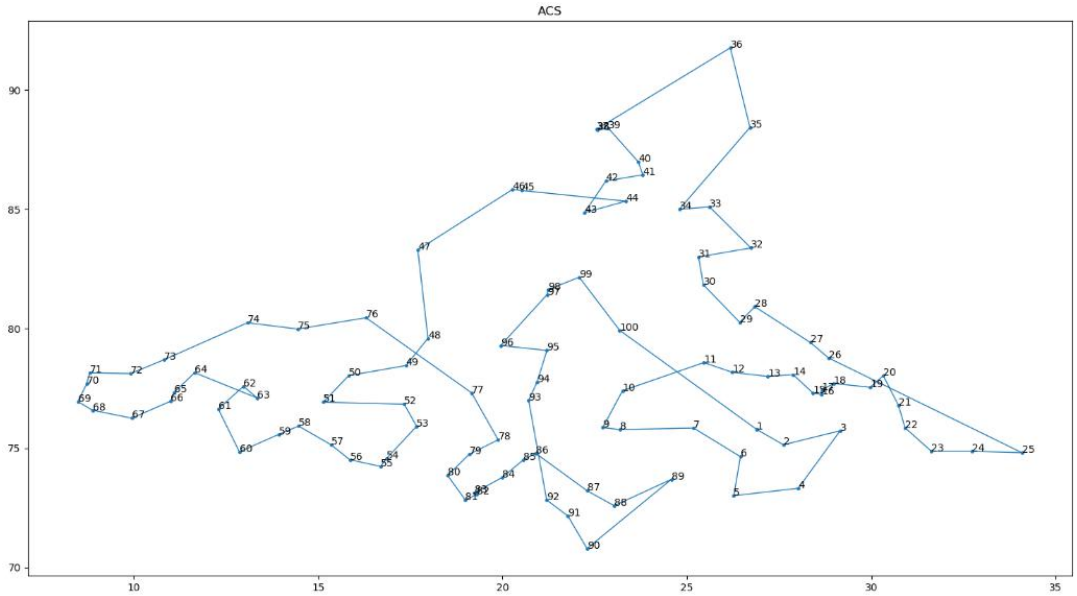

Fig 11. Graph of Ant Bee Colony Algorithm(ACS)

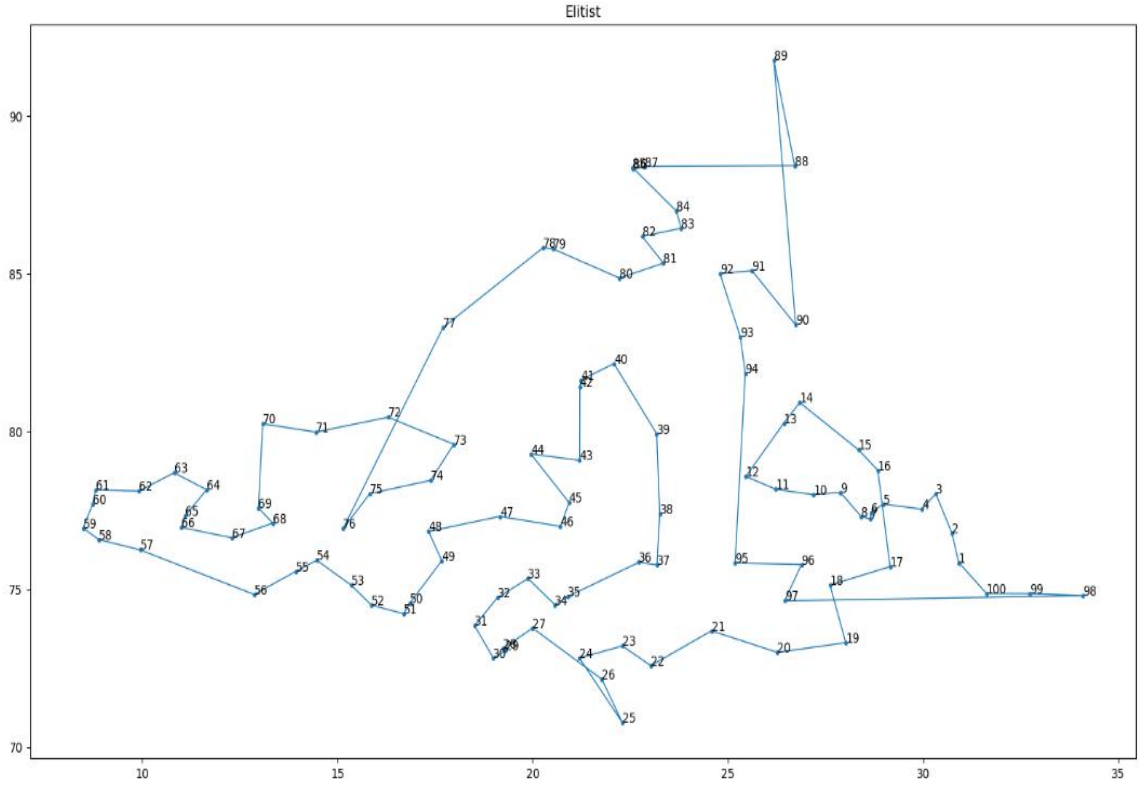

Fig 12. Graph of Ant Bee Colony Algorithm(Ellitist)

6.1.4 HYBRIDIZED PSO AND ABC ALGORITHM RESULT

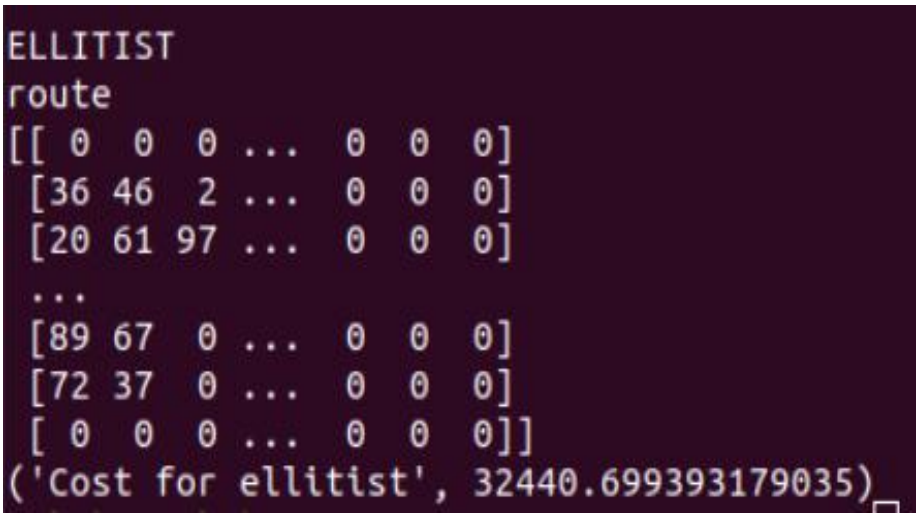

Fig 13. Hybridized PSO and ABC Algorithm

6.2.4 GRAPH OF HYBRIDIZED PSO AND ABC ALGORITHM

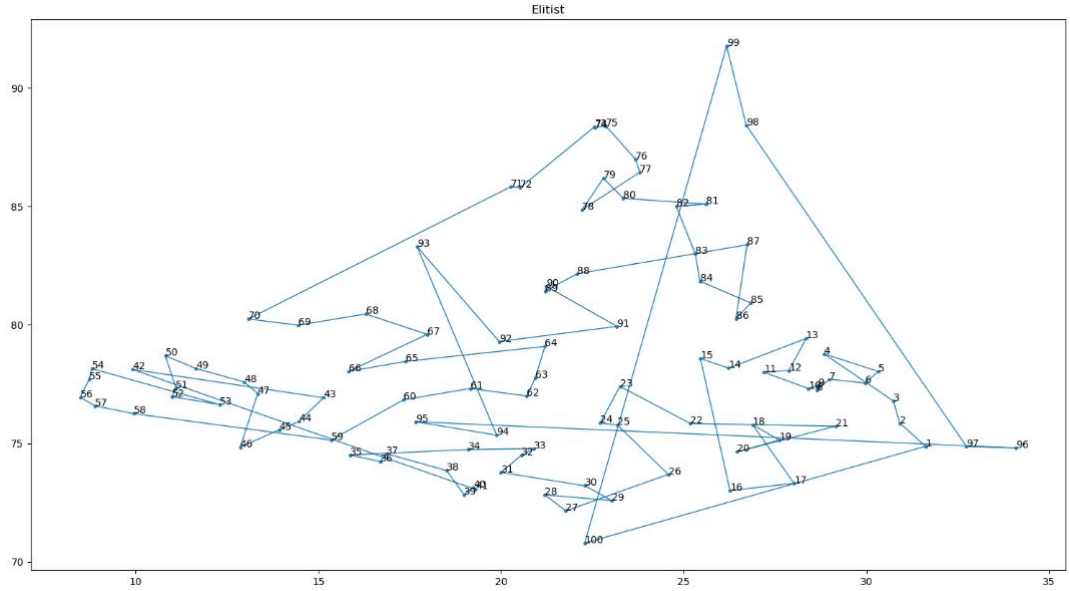

Fig 14. Graph of Hybridized PSO and ANT Algorithm

6.1.5 HYBRIDIZED GENETIC AND GENETIC ALGORITHM RESULT

```
route
[[  0   0   0 ...   0   0   0]
 [ 39  62   1 ...   0   0   0]
 [ 44  55   2 ...   0   0   0]
 ...
 [ 56  74 100 ...   0   0   0]
 [ 98  89   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
('AND THE FINAL COST FOR OUR SOLUTION IS', 39874.5408984615)
```

Fig 15. Hybridized Genetic and Genetic Algorithm

6.2.5 GRAPH OF GENETIC AND GENETIC ALGORITHM

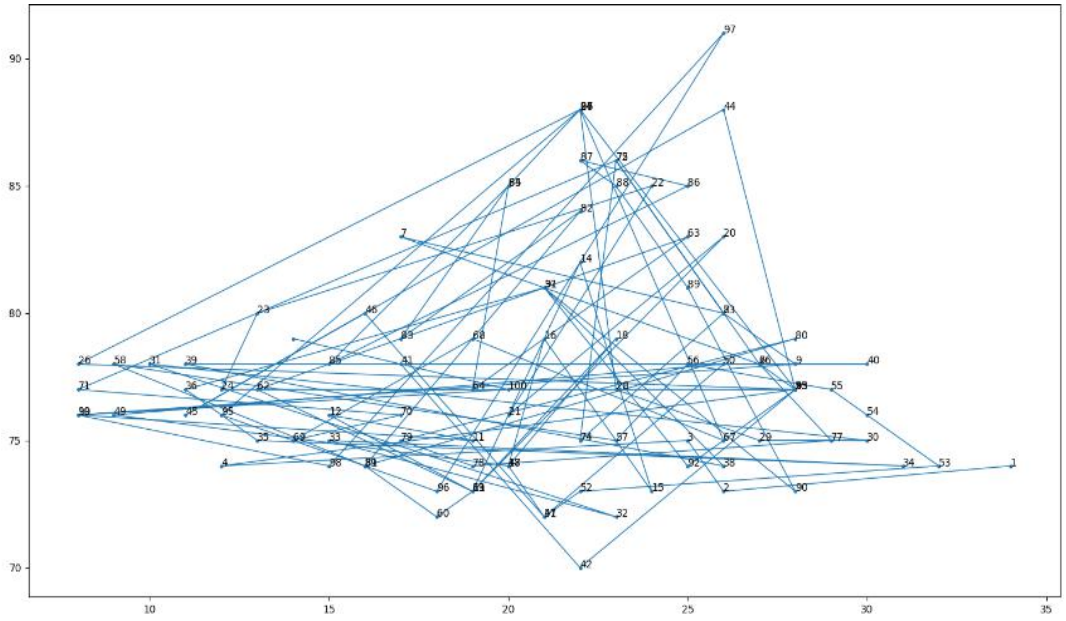

Fig 16. Graph of Hybridized Genetic with Genetic Algorithm

COMPARATIVE GRAPHS OF THE STUDIED ALGORITHMS

COST FOR INSTANCE I

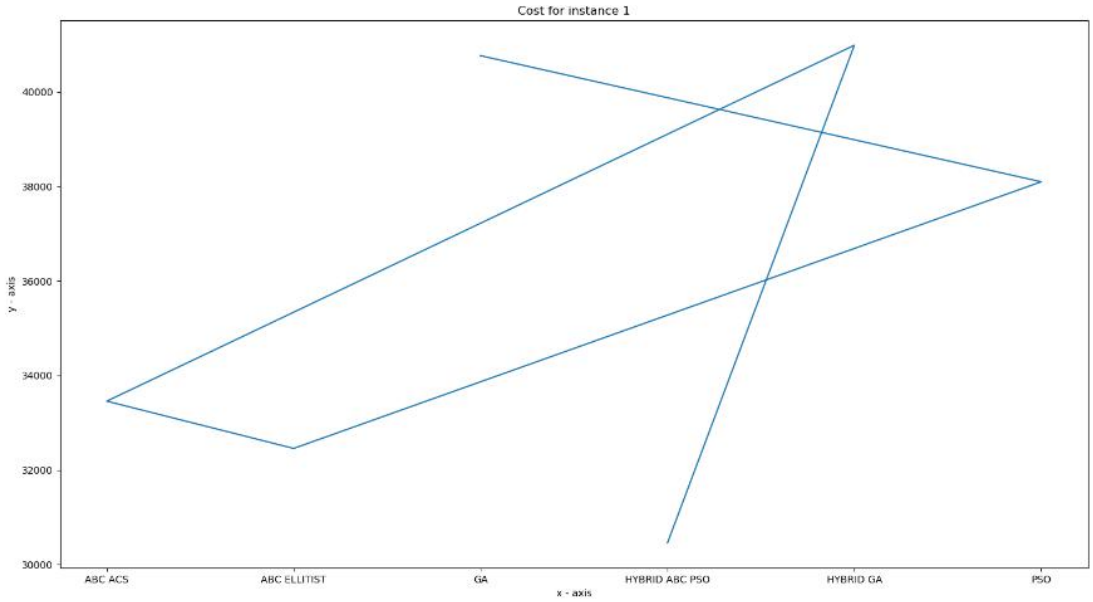

Fig 17. Graph for comparison of cost of instance I

COST FOR INSTANCE II

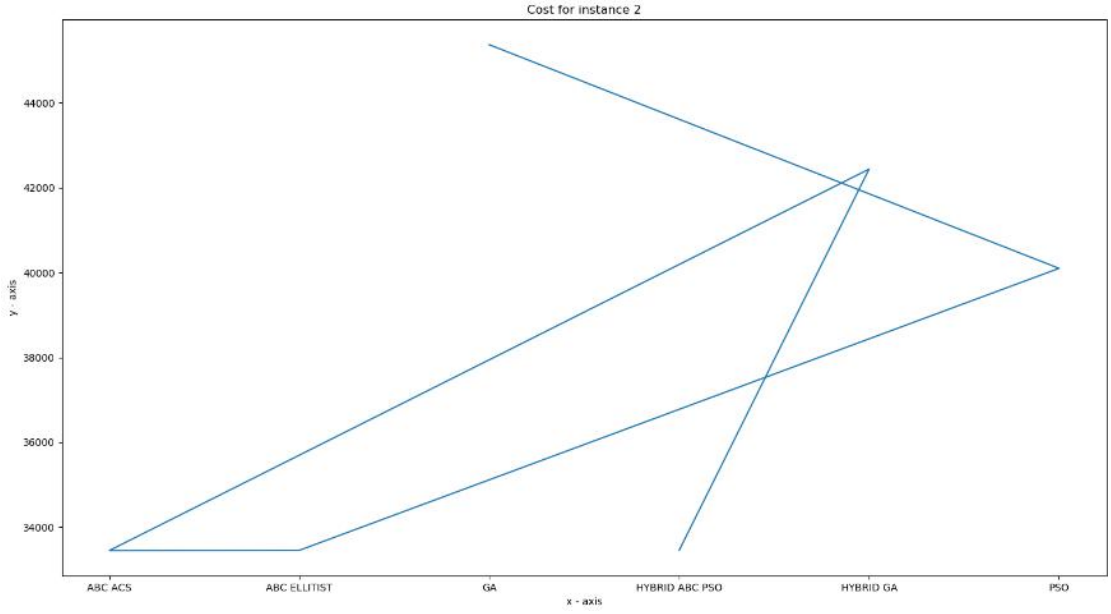

Fig 18. Graph for comparison of cost for instance II

COMPUTATIONAL TIME FOR INSTANCE I

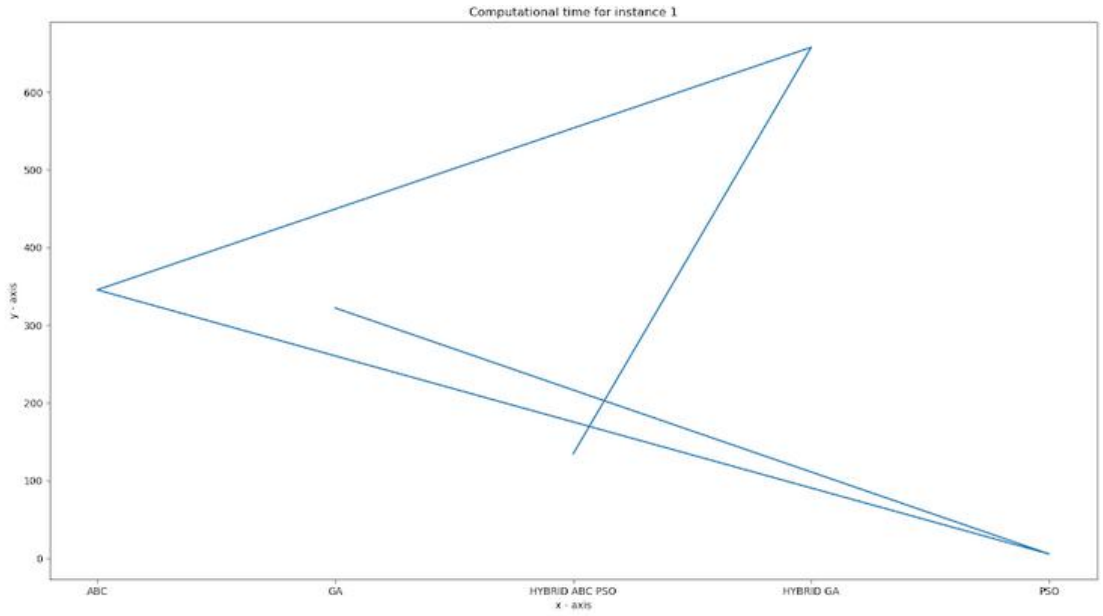

Fig 19. Graph for comparison of computational time for instance I

COMPUTATIONAL TIME FOR INSTANCE II

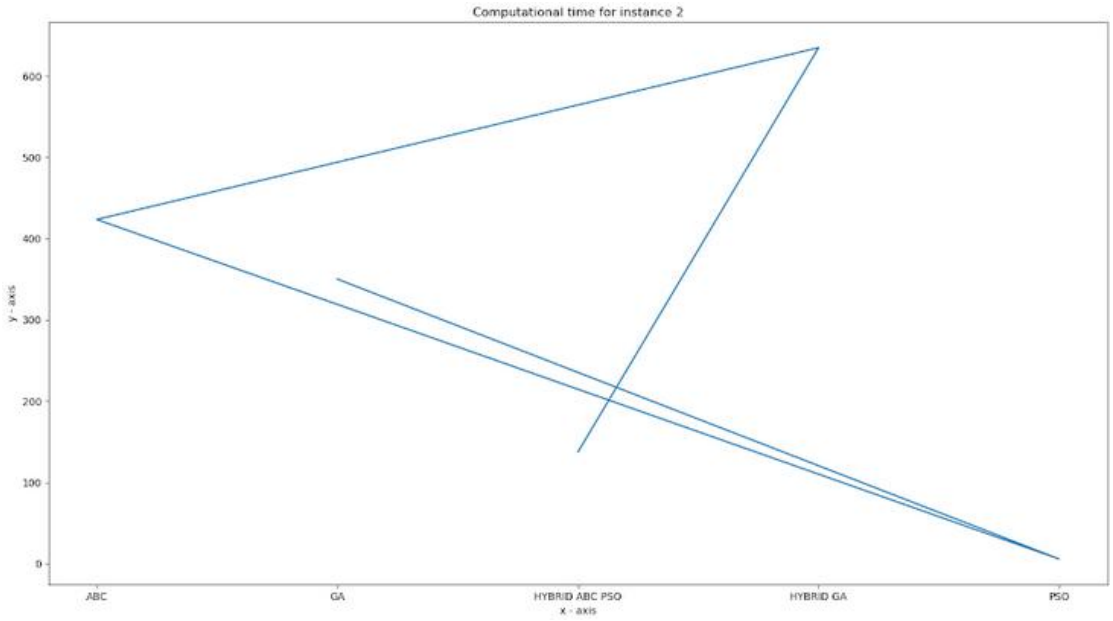

Fig 20. Graph for comparison of computational time for instance II

PARAMETER VARIATION FOR HYBRID ABC AND PSO

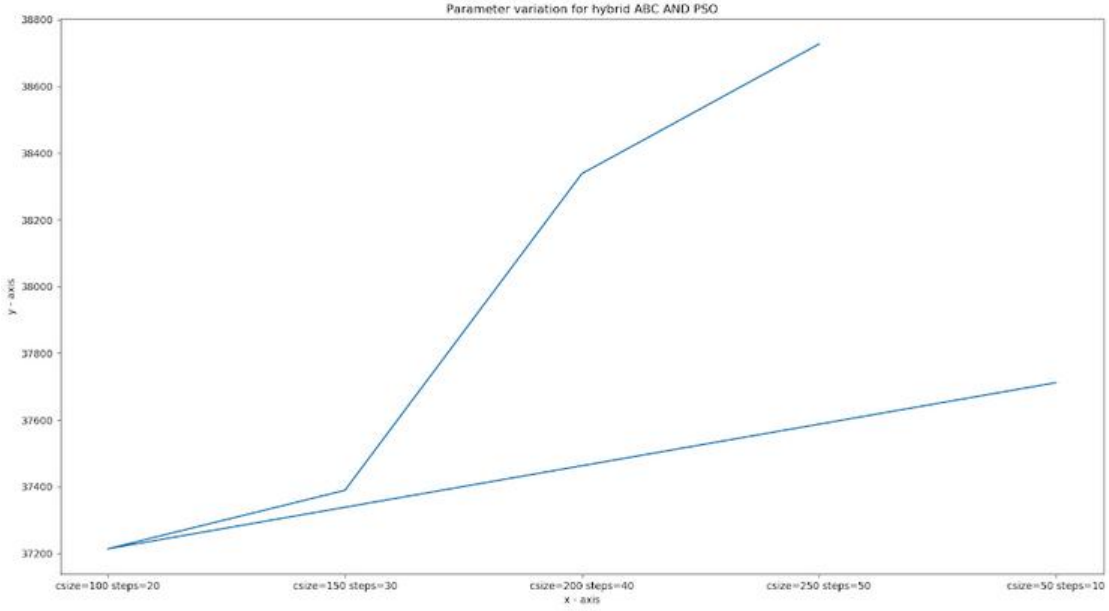

Fig 21. Graph for parameter variation of hybridized PSO and ABC algorithm

COST vs COMPUTATIONAL TIME FOR INSTANCE I

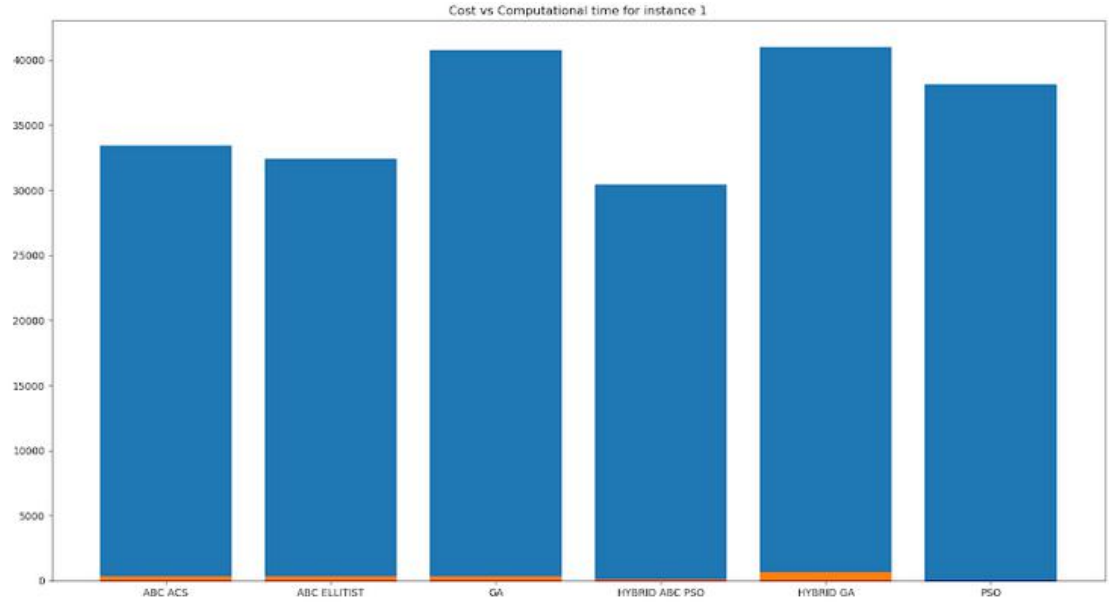

Fig 22. Graph for cost vs computational time for instance I

COST vs COMPUTATIONAL TIME FOR INSTANCE II

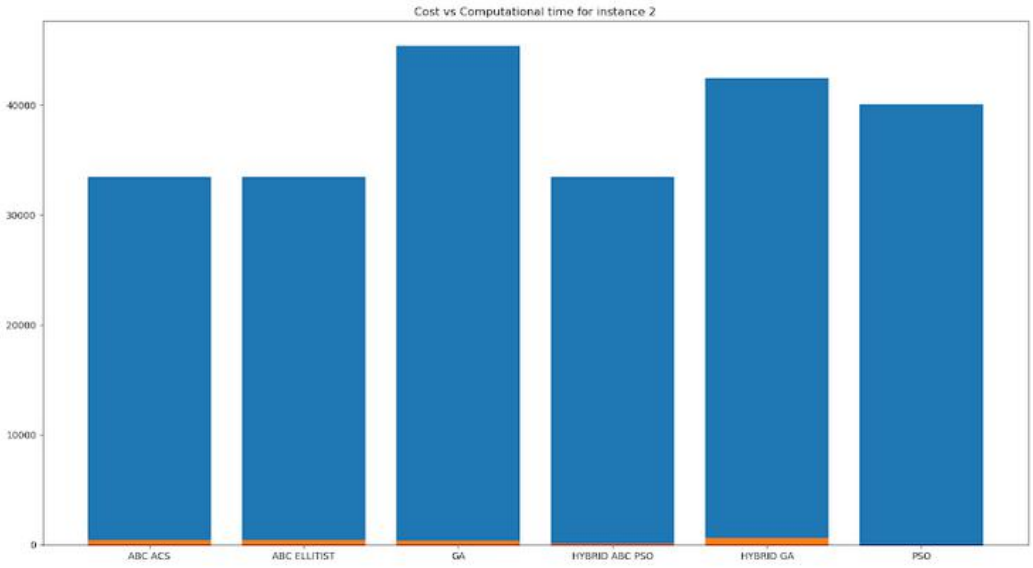

Fig 23. Graph for cost vs computational time for instance II

6.3 PICTORIAL DISPLAY OF MOST OPTIMIZED ROUTE

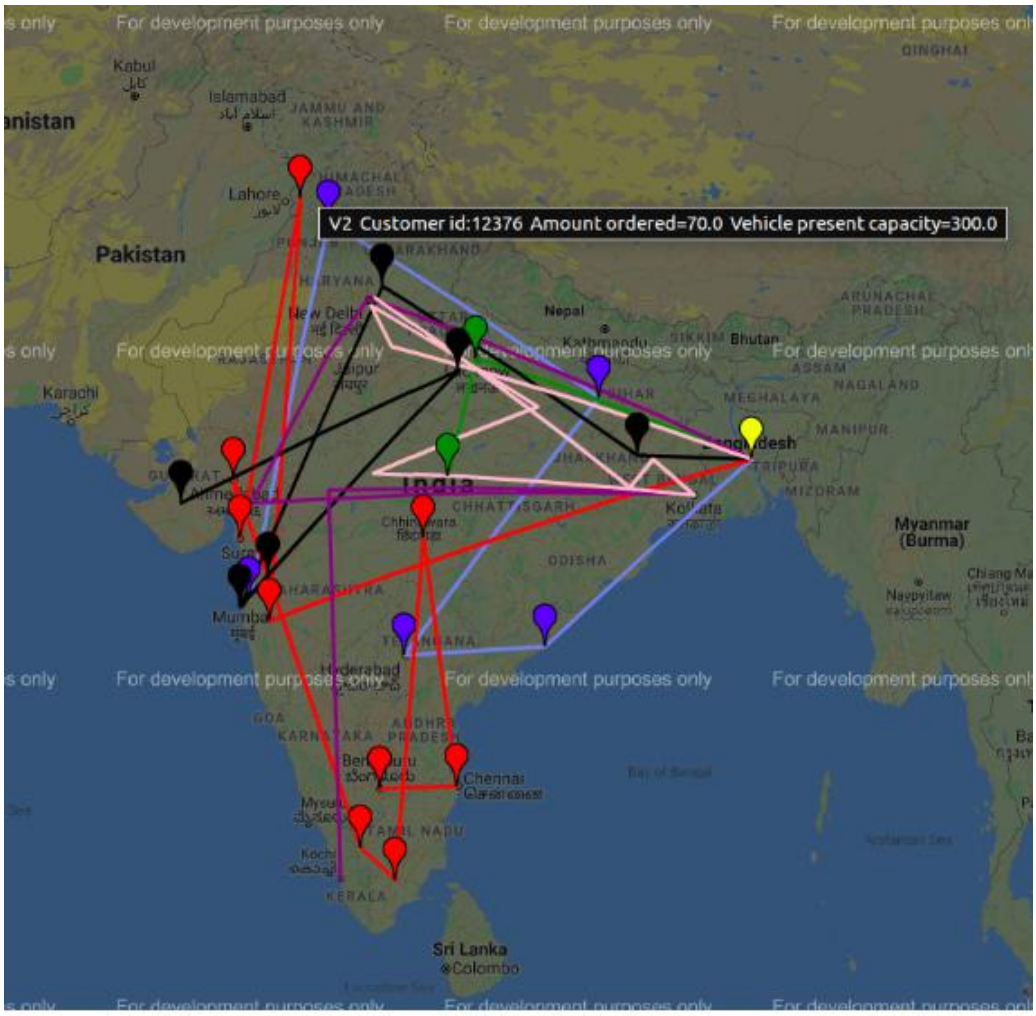

Fig 24. Pictorial Display of the most Optimized Route on a Map

(This map is plotted using gmplot in python)

Assumptions considered in map-

A) We have assumed our depot to be Bangladesh.

B) Different colors in the map shows different vehicles used for delivering the package.

7. FUTURE SCOPE

i. Vehicle routing problems can be of great use in the future. In the future, many new algorithms can be implemented. We can improve by implementing a multi-depot vehicle routing algorithm in which multiple depots to load and unload the packages would be available.

ii. A split delivery vehicle routing algorithm can also be implemented in which the package once loaded in a vehicle can be shifted to other vehicles delivering in the same location to save time but in compliance with the loading capacity of the vehicle only.

iii. A new algorithm can also be implemented in which the routing algorithm would be applicable to the pickup facility also. The vehicle which would be near to the depot would be allotted the package for delivering to save time.

8. CONCLUSIONS

This study proposed five meta heuristic methods named Genetic Algorithm (GA),Particle Swarm Optimization (PSO), Ant bee colony(ABC), hybridized genetic and PACO to solve the Capacitated Vehicle Routing Problem (CVRP). We tested the proposed algorithm on a data set from solomon benchmark problems to see the performance of all the algorithms. The results indicate that the performance of proposed algorithm is competitive.

The genetic algorithm gives us the solution with maximum cost and high computational time. The hybridized genetic algorithm provides us with less cost than genetic algorithm but requires more computational time. Though the gaps are relatively small but still if not concerned about the computational time hybridized genetic algorithm is better than genetic algorithm.

The PSO algorithm provides us with the better cost than genetic and even hybridized genetic algorithm and the computational time is almost negligible in comparison to genetic algorithm.Among these three, PSO turns out to be the best in both parameters, cost and computational time.

This research also compares all these algorithms on the real data set of India cities and the depot is located in Bangladesh and the performance is compared.

The ant colony optimization has two variants : Ant colony system and Ellitist.

The Ant colony system provides us with a better solution than Ellitist. These two variants were tested on the same values of initial pheromone, alpha, beta and rho.

Also we can conclude that ACS and Ellitist, both provides us with a better solution than Genetic and PSO algorithm, with ACS being the best.

In the desire for more better results, our implementation of hybridized ABC and PSO algorithm (PACO) turned out to be the best solution finally on all datasets. The PACO algorithm provides us with a route with the least cost. And moreover, to our surprise its computational time is also relatively less. Though PSO algorithm has less computational time than PACO but the cost difference is high in both the algorithms.

So our research concludes that PACO algorithm provides us with the best results in both parameters: Cost and computational time.

We would like to carry our research forward in this field and would compare the performances of more hybridized algorithms on the same dataset.

9. REFERENCES

[1] Yucheng Kao, Ming-Hsien Chen, and Yi-Ting Huang, "A Hybrid Algorithm Based on ACO and PSO for Capacitated Vehicle Routing Problems," Mathematical Problems in Engineering, vol. 2012, Article ID 726564, 17 pages, 2012. https://doi.org/10.1155/2012/726564.

[2] http://web.cba.neu.edu/~msolomon/problems.htm

[3] https://simplemaps.com/data/in-cities

[4] https://www.sciencedirect.com/topics/engineering/artificial-bee-colony

[5] https://www.google.com/amp/s/www.researchgate.net/figure/The-CVRP-with-Time-Windows-CVRPTW_fig3_319754352/amp

[6] https://pypi.org/project/gmplot/

[7] https://www.localsolver.com/docs/last/exampletour/vrptw.html

[8] P. L. N. U. Cooray and Thashika D. Rupasinghe, "Machine Learning-Based Parameter Tuned Genetic Algorithm for Energy Minimizing Vehicle Routing Problem," Journal of Industrial Engineering, vol. 2017, Article ID 3019523, 13 pages, 2017. https://doi.org/10.1155/2017/3019523.

[9] https://www.google.com/url?sa=t&source=web&rct=j&url=https://papers.nips.cc/paper/8190-reinforcement-learning-for-solving-the-vehicle-routing-problem.pdf&ved=2ahUKEwi4iu_mooLmAhVz8HMBHZo3DHUQFjADegQIAxAB&usg=AOvVaw0WhfHuvZ1fzvnSoXCfHwK0&cshid=1574578744580

[10] https://www.google.com/url?sa=t&source=web&rct=j&url=https://towardsdatascience.com/improving-operations-with-route-optimization-4b8a3701ca39&ved=2ahUKEwi4iu_mooLmAhVz8HMBHZo3DHUQFjAIegQIBBAB&usg=AOvVaw3oe7v7lcYRXdf7AZRoDiIZ&cshid=1574578744580

[11] https://www.google.com/url?sa=t&source=web&rct=j&url=http://users.jyu.fi/~miettine/kurssit/jatkoksem/jussi03032011.pptx&ved=2ahUKEwi4iu_mooLmAhVz8HMBHZo3DHUQFjAPegQICRAB&usg=AOvVaw2m965FKHRKA0dxLi0Jx8Vh&cshid=1574578744580

[12] https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.getfareye.com/ecommercelogistics/blog/machine-learning-based-vehicle-routing-software&ved=2ahUKEwi4iu_mooLmAhVz8HMBHZo3DHUQFjAQegQIBhAB&usg=AOvVaw1t2HTwwKzZUi6xF2TpPSi_&cshid=1574578744580