TSP Problem Analysis Using GA and PSO Algorithms

Nivya Muchikel
Department of Mathematics
RV College of Engineering
Bangaluru, India
nivyamuchikel@rvce.edu.in

Rajat Raj
Department of Mathematics
RV College of Engineering
Bangaluru, India
rajatraj.ec19@rvce.edu.in

Aditya Singh
Department of Mathematics
RV College of Engineering
Bangaluru, India
adityasingh.ee19@rvce.edu.in

***Abstract*— The travelling salesman problem (TSP) is a well-known, popular and widely studied subject in the field of combinatorial optimization. The purpose of this article is to provide a collective intelligence strategy to aid solving optimization issues and apply it in particular to the Travelling Salesman Problem. The technique employed is the particle swarm optimization (PSO) whose fundamental premise is to replicate the collective behavior of a cloud. This article also compares the results achieved using PSO algorithm with those obtained by utilizing another popular metaheuristic wich is the Genetic Algorithm.**

Keywords—Travelling Salesman Problem, Genetic Algorithms, Particle Swarm Optimization

I. Introduction

Finding the best configurations from a number of discrete objects is a growing necessity in many real-world applications. This is a combinatorial optimization problem. One of the most well-known issues in combinatorial optimization is the travelling salesman problem (TSP), in which a salesperson must visit many locations precisely once, then return to the starting point while minimising the total distance travelled or the total cost of the trip.

The Traveling Salesman is a well-known NP-hard combinatorial problem that has served as a significant algorithmic testing ground. Finding the shortest total length Hamiltonian cycle of a given graph $G = (N,E)$, where $N = 1,...,n$ is the set of nodes and $E = 1,...,m$ is the set of edges, is the challenge. Costs, c_{ij}, are associated with each edge joining vertices I and j. By adding up the costs of the edges in a cycle, the length is determined. The situation is referred to as symmetric if the costs cij and cji are equal for all node pairs (i,j), else it is referred to as asymmetric. [1]

The TSP's significance is independent to salespeople's excessive efforts to cut down on travel time. It can be utilised for transportation, including school bus routes, service calls, and meal delivery Manufacturing: a machine that makes printed circuit boards with holes Layout of a VLSI (microchip) Planning a new telecommunications network: communication. Combinatorial optimization problems have been addressed using a variety of bio-inspired techniques, including Genetic Algorithms [2], Cultural Algorithms [3], Memetic Algorithms [4], and Ant Systems [5], Particle Swarm Optimization, etc.

Crossover, mutation, and selection operators are key components of the genetic algorithm. There are several representations, including binary, path, adjacency, ordinal, and matrix representations, that may be used to solve the travelling salesman problem using genetic algorithms. According to Holland [6], genetic algorithms (GAs) are stochastic methods without derivatives that are based on biological evolutionary processes. Natural selection favours the fittest individuals; therefore, the following generation should be healthier and more physically fit than the one before it. The population of chromosomes that GAs operate with is represented by a collection of underlying parameter codes.

II. GENETIC ALGORITHM (GA)

Computer engineers deploy the genetic algorithm (GA) as a computational intelligence methodology to identify approximations of solutions to combinatorial optimization problems. It would be more accurate to describe the genetic algorithms as an optimization method based on natural evolution.

They incorporate the algorithm for the "survival of the fittest" notion. Finding the person from the search space with the best "genet material" is the goal of employing a genetic algorithm. An evaluation function is used to gauge a person's quality. The population refers to the portion of the search space that needs to be looked at. A genetic algorithm functions roughly as follows: (see Figure 1):

```
Algorithm 2 GA Algorithm
 1: Set Parameters
 2: Choose encode method
 3: Generate the initial population
 4: while i < MaxIteration and Bestfitness < MaxFitness do
 5:     Fitness Calculation
 6:     Selection
 7:     Crossover
 8:     Mutation
 9: end while
10: Decode the individual with maximum fitness
return the best solution
```

Figure 1. The pseudo-code of the Genetic Algorithm (AGA).

It would be more accurate to describe the genetic algorithms as an optimization method based on natural evolution. They incorporate the algorithm for the "survival of the fittest" notion. In order to develop a new generation of solutions that should be superior to the prior generation, the concept is to first "guess" the solutions and then combine the best ones. In order to accommodate for possible mistakes, we additionally include a random mutation element.

The following are the steps in the genetic algorithm process:

1. Encoding: A appropriate encoding is discovered for the solution to our problem, resulting in a string-based encoding for each potential solution.
2. Evaluation follows the selection of the starting population, which is typically done at random but other methods utilising heuristics have also been suggested. The fitness of each population member is then calculated, i.e., how well the individual fits the problem and whether it is close to the optimum in comparison to the other population members.
3. Crossover: The fitness is used to determine a person's likelihood of crossing over. Crossover occurs when two people are united to produce new people who are then replicated into the new generation.
4. The subsequent mutation takes place. Randomly selected individuals are picked to be mutated, and a mutation point is then selected. The character in the corresponding position of the string is changed.
5. Decoding: Once this is done, a new generation has been formed and the process is repeated until some stopping criteria has been reached. At this point the individuals which is closest to the optimum is decoded and the process is complete.

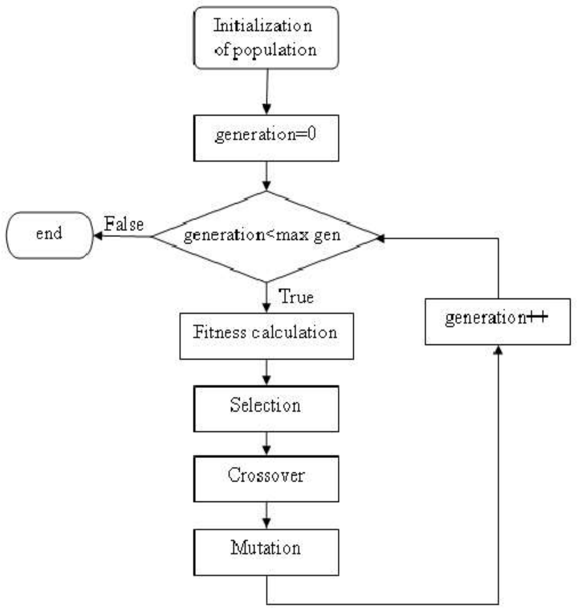


Fig 3: Flow chart of the GA algorithm

III. TSP USING GENETIC ALGORITHMTIC (GA)

In a lot less time, a genetic algorithm can find the answer. It can find a nearly perfect solution for 100 city tours in less than a minute, even though it might not find the best one. There are a better than either parent couple of fundamental stages to solve the TSP using a GA.

First, construct a collection of numerous random excursions in what is called a population. The greedy initial population used by this algorithm favours connecting cities that are close to one another. Select two of the population's better (shorter) parents, then combine them to create two new child tours. Hopefully, these young people will be. The chance that the infant is mutant is quite tiny. To avoid a uniform appearance among all tours in the population, this is done. Two of the lengthier tours are replaced by the new kid tours, which are introduced into the population. The population is still the same size. The creation of fresh kid-friendly tours continues until the intended outcome is achieved. Speed and population size are two variables that will affect how accurate the TSP solution is.

The optimal solution will be chosen after comparing these elements in each one, and it will provide the new shortest path in each iteration. With an increase in population size, the process of comparing and describing the result in each iteration becomes more difficult.

IV. PARTICLE SWARM OPTIMIZATION (PSO)

An elementary version of the PSO algorithm employs a population swarm) of potential solutions (particles). Throughout the search space, these particles are in motion. Their motions are dictated by both their individual best-known position inside the search-space and the best-known position of the entire swarm. The particles will start to take part in tracking the swarm's motions as they locate better spots. Repeating the procedure increases the likelihood—though not necessarily the certainty—that a workable solution will be found in the end. Formally, the cost function that needs to be minimised is denoted by f: Rn R. The function outputs a real number that represents the candidate solution and accepts a vector of real numbers as an input.

It draws its inspiration from the behaviour of fish schools, bird flocks, and swarming theory. Kennedy and Eberhart [7] were the first to present PSO techniques for continuous nonlinear function optimization. Their methodology's foundations are based on studies of computer simulations of social animal movements [8]

This particles move in the search space. Each one is considered as a solution of the problem, since they have a position Xid and a speed Vid. In addition, each particle has a memory about his best position visited Pid and the neighborhood's one Pgd.

$$Vid\ t{+}1 = \omega\ Vid\ t + C1R1\ (Pid - Xid) + \ldots..$$

$$\ldots.\ C2R2\ (Pgd - Xid) \qquad (1)$$

$$Xid\ t{+}1 = Xid\ t + Vid\ t{+}1 \qquad (2)$$

The coefficient of inertia is denoted as, the coefficients C1 and C2 are constants established empirically using the formula C1 + C2 4, and R1 and R2 are random positive values distributed uniformly between [0,1]. The following three factors affect a particle's displacement strategy, as shown in Fig. 2:

1. An element of inertia (Vid t): the particle prefers to continue travelling in the same direction;

2. The particle prefers to proceed in the direction of the best site for which it has already crossed over (C1R1 (Pid - Xid));

3. A social element (C2R2 (Pgd - Xid)): The particle has a propensity to depend on the knowledge of its congeners and, as a result, to move toward the optimal location previously attained by its neighbours.

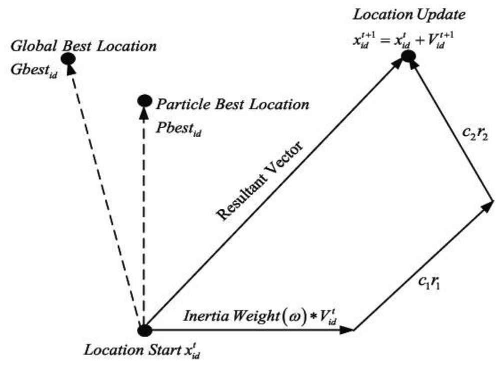


Fig 2: Particles movement

A genetic algorithm functions roughly as follows: (see Figure 3): The operations made in the PSO algorithm are explained in Fig 4.

Algorithm 1 PSO Algorithm

```
1:  Initialize the population position vector $xi^d$ and velocity
    vector $vi^d$ randomly at $j = 1$, N is the population size
2:  while (iteration limit is reached) do
3:      $j = j + 1$
4:      for loop over all $d^{th}$ dimensional N particles do
5:          Update $vi^d$ using Equation (1)
6:          Update $xi^d$ using Equation (2)
7:          Adjust the controller FIS
8:          Evaluate the objective function
9:          Determine the current best for each particle $pi^d$
10:     end for
11:     Determine the current global best $g^d$
12: end while
13: Output the final $gi^d$ and $pi^d$
```

Figure 1. The pseudo-code of the PSO

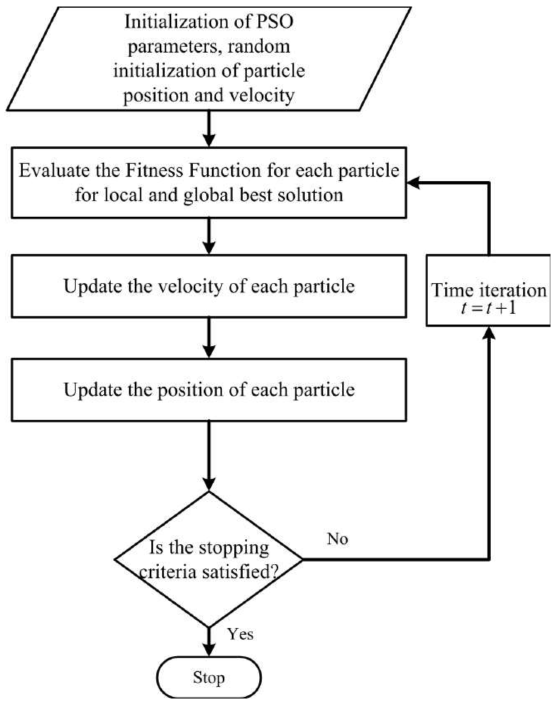

Initialization of PSO parameters, random initialization of particle position and velocity
Evaluate the Fitness Function for each particle for local and global best solution
Update the velocity of each particle
Update the position of each particle
Time iteration $t = t + 1$
Is the stopping criteria satisfied?
No
Yes
Stop

Fig 3: Flow chart of the PSO algorithm

For particle id, its velocity and position renewal, the mathematical algorithm is as follows:

In the classical PSO algorithm, each particle.

• has a position and a velocity
• knows its own position and the value associated with it
• knows the best position it has ever achieved, and the value associated with it
• knows its neighbours, their best positions and their values

The movement of a particle is a composed of three possible choices

• To follow its own way
• To go back to its best previous position
• To go towards its best neighbour's previous or present position.

V. TSP USING PSO

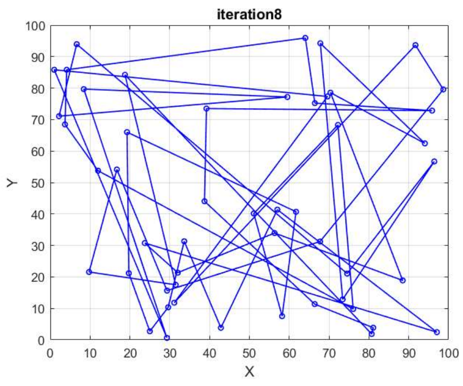
The first component of equation (1) denotes the particle's previous velocity. It gives the particle a growing tendency in the search space, which increases the capability of the algorithm for global searching; the second part is called cognition part. It symbolises the process of acquiring individual experiential information on the part of the the process of learning from the experiences of other particles is represented by the third component, which is referred to as the social aspect on the part of a specific particle, as well as the social collaboration and information sharing amongst particles. The Following is a succinct description of the PSO flow:

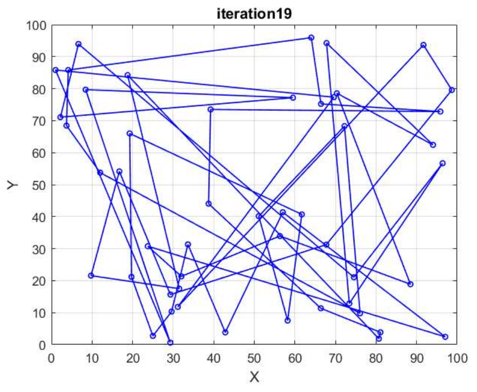
A set of particles should first be initialised, for example, by randomly assigning each particle a starting position (Xi) and starting speed (Vi), and then the fitness value (f) should be determined. the velocity and locations of regenerated particles in equation (1) were examined to determine a particle's fitness value for each repetition (2). It is also referred to as the cognition component in equation (1) when a particle finds a position that is better than its previous one and marks this coordinate into vector P1. The vector difference between P1 and the particle's current position will then be added at random to the next velocity vector.

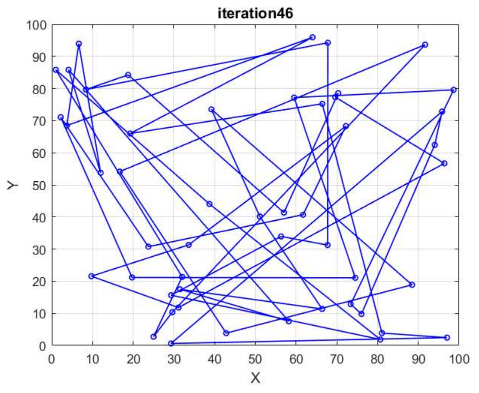
For the purpose of altering the following population velocity, the weight difference between the particle swarm's current position and its optimal position, or Pgd, will also be added to the velocity vector. This is often referred to as the social component in equation (1). Particles will be able to search around two bests thanks to these two modifications. The most evident benefit of PSO is the swarm's rapid convergence rate; academics have provided evidence of this convergence. to confirm the PSO algorithm's speed of convergence.

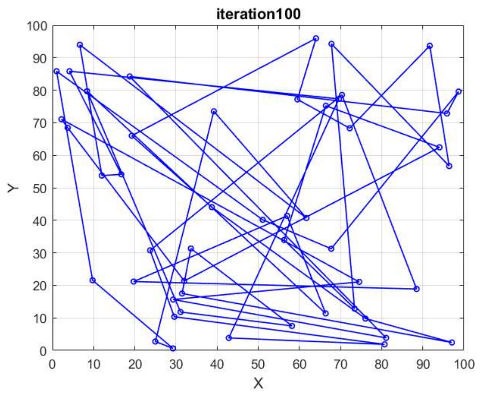
VI. RESULTS

All the simulations were completed on a Windows 64 bits personal computer with an i5-4710 processor clocked at 2.50GHz, and 8 GB of Ram. The Genetic algorithm and Particle Swarm Optimization was developed in MATLAB.









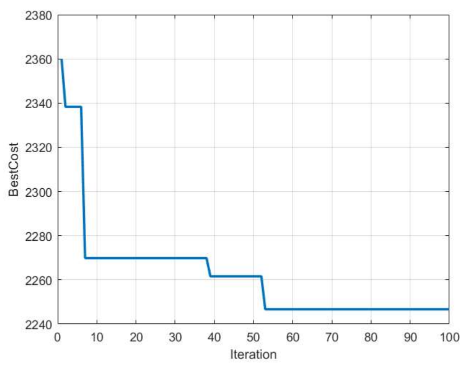


Fig: TSP Problem Solution with PSO at different Iterations

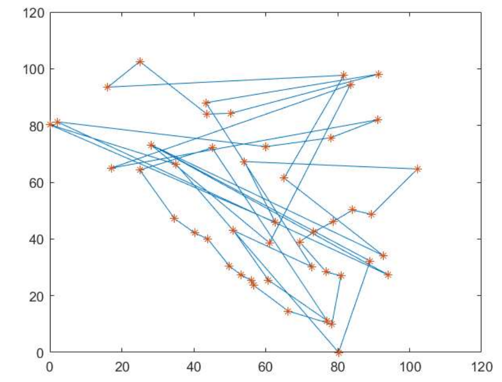


Fig: TSP Problem Solution with GA

VII. COMPARISION OF RESULTS

These two algorithmic techniques were compared qualitatively and quantitatively. This NP-hard optimization problem can be solved using any approach, and the results are satisfactory. Even when the search space is big and complicated, they are both effective in attacking every occurrence of the TSP.

VIII. CONCLUSION

The performance of particle swarm optimization and Genetic Algorithms are evaluated using MATLAB Simulator. In Optimization techniques cost and time are essential parameters. This project has presented the analysis of existing optimization techniques and their critical study. We have selected certain parameters associated with cost and time. Optimization techniques such as Particle swarm optimization (PSO) and Genetic Algorithm (GA) according to those parameters. In all scenarios we have observed that PSO gives better result than GA with respect to cost and time. In addition the increase in number of iterations and cities increases the execution time of both algorithms.

REFERENCES

[1] Gutin, G., Punnen, A. P. (Ed.): Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers (2002)

[2] Holland, J. H.: Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI (1975)

[3] Reynolds, R. G.: An Introduction to Cultural Algorithms. In: Proceedings of Evolutionary Programming, EP94, World Scientific,

River Edge, NJ (1994) 131-139K. Elissa, “Title of paper if known,” unpublished.

[4] Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Caltech Concurrent Computation Program, C3P Report 826, 1989.

[5] . Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation vol. 1, N. 1 (1997) 53-66.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Oxford, UK, 1975.

[7] Cheeseman, P., Kenefsky, B., Taylor, W.: Where the really hard problems are. In: Proceedings of the ijcai’91. (1991) 331–3370.

[8] B¨ack, T., Fogel, D., Michalewicz, Z., eds.: Handbook of Evolutionary Computation. Institute of Physics Publishing Ltd, Oxford University Press (1997)