International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET218135

Experimental Comparison between Genetic Algorithm and Ant Colony Optimization on Traveling Salesman Problem

Muhammed Yaseen Morshed Adib[*1], Jannatun Razia[2], Md. Toufiqur Rahman[3]

[*1] Computer Science and Engineering, Stamford University Bangladesh, Dhaka-1217, Dhaka, Bangladesh

[2]Computer Science and Engineering, Ahsanullah University o Science and Technology, Dhaka-1208, Dhaka, Bangladesh

[3]Computer Science and Engineering, Ahsanullah University o Science and Technology, Dhaka-1208, Dhaka, Bangladesh

Article Info
Volume 8 Issue 1
Page Number: 155-162
Publication Issue :
January-February-2021

Article History
Accepted : 02 Feb 2021
Published : 11 Feb 2021

ABSTRACT

This paper is based on bio-inspired optimization algorithms. Optimization is the process of selecting the best element by following some rules and criteria from some set of available alternatives. In this paper, we have solved Traveling Salesman Problem (TSP) using Swarm Intelligence algorithms and we have compared them. First we have implemented the basic Genetic Algorithm (GA) on TSP. Then we have implemented Ant Colony Optimization (ACO) Algorithm on TSP. In optimization problem, Genetic Algorithm (GA) and Ant Colony Optimization (ACO) Algorithm have been known as good meta-heuristic techniques. GA is designed by adopting the natural law of evolution, while ACO is inspired by the foraging behavior of ant species. Balancing the exploitation-exploration tradeoff is required in ACO. In contrast with the GA implementation, ACO was much easier to control.

Keywords : Bio-inspired optimization, Traveling Salesman Problem, Swarm Intelligence, Genetic Algorithm, Ant Colony Optimization, Meta-heuristic

I. INTRODUCTION

Meta-heuristics has become a research interest to many scientists in recent years. Meta-heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed [1]. Optimization problems are often very challenging to solve in real world. Optimization tools must be used to solve such problems, though there is no guarantee that the optimal solution can be obtained. Efficient search or optimization algorithms are needed to solve the optimization problem. There are many optimization algorithms which can be classified in many ways, depending on the focus and characteristics.

Genetic algorithm (GA) is a search heuristic that mimics the process of natural selection in the field of artificial intelligence. Genetic algorithms belong to the larger class of evolutionary algorithms (EA).

Those generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection and crossover [2][3].

Ant colony optimization (ACO) is introduced by Dorigo in his doctoral dissertation. ACO is a class of optimization algorithms modeled on the actions of an ant colony. ACO is a probabilistic technique which is useful in problems dealing with finding better paths through graphs. Artificial 'ants' (simulation agents) locate optimal solutions by moving through a parameter space which represent all possible solutions. Natural ants lay down pheromones and direct each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions. Later in simulation iterations more ants locate better solutions [4].

Ant colony optimization (ACO) is based on the intelligent ant swarm behavior for finding an (or, a set of) optimal route(s) from their nest to some distant food source(s). ACO based algorithms are appropriate for the optimization problems that need to find routes from a source (or, initial state) to a destination (or, goal state). In ACO, artificially simulated ant agents search across a parameter space and it represents the search space of all possible solutions to find some optimal solution. The artificial ants put pheromones, like natural ants, along their way from their nest to the resources to find the solution. Their search behavior is also randomly affected by the pheromones which laid down by other ants. Better trails have smaller lengths as a result, they require less time to travel and hence have higher density of pheromones that in turn attracts more ants to deposit pheromones along them. Gradually simulated ants find an optimal or near-optimal trail with their distributed, self-organized behavior and interactions among them [4].

Traveling Salesman Problem (TSP) is one type of discrete problem. If a list of cities and distances between each pair of cities are given, TSP finds the shortest possible route that visits each city exactly once and returns to the origin city.

To minimize the total distance travelled during the tour, it is required to calculate the distance d_{ij} between each pair of nodes i and j. The total distance travelled is then the sum of the distances of the edges included in the tour distance travelled $= \sum d_{ij} \in Ex_{ij}$

The tour should only pass through each city once. Therefore, each node in the graph should have exactly one incoming edge and one outgoing edge. In other words, for every node i exactly two of the x_{ij} binary variables should be equal to 1. This constraint can be written as:

$$\sum x_{ij} \in V = 2 \forall_i \in V \quad (1)$$

In this paper, TSP has been solved using these two algorithms GA and ACO and the comparative properties of GA and ACO algorithms to solve tsp have been discussed.

II. GA IMPLEMENTATION ON TSP

Traveling Salesman Problem (TSP) using Genetic Algorithm (GA) (TSP_GA) finds (near) optimal solution to the TSP by setting up a GA to search for the shortest route (least distance for the salesman to travel to each city exactly once and return to the starting city.)

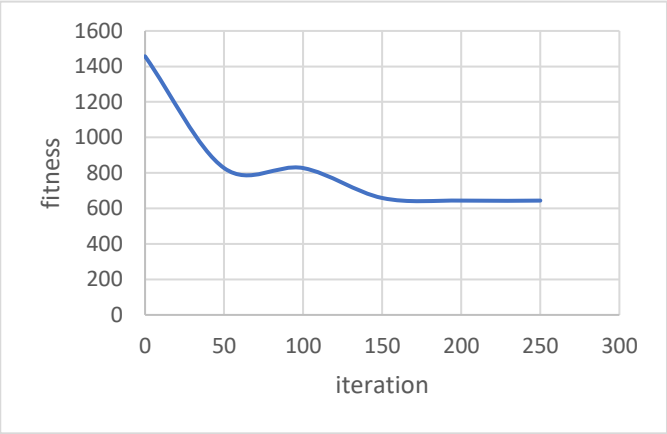


Figure 1: Traveling Salesman Problem using Genetic Algorithm

To find a solution to the traveling salesman problem, it requires setting up a genetic algorithm in a specialized way. For instance, a valid solution would need to represent a route which includes every location at least once and only once. If a route contains a single location more than once or missed a location out completely it would be invalid. Special types of mutation and crossover methods are needed to ensure that genetic algorithm meets these requirements [5].

Firstly, the mutation method should only be capable of shuffling the route. It should not ever add or remove a location from the route. Otherwise, it would risk creating an invalid solution. One type of mutation method which can be used is swap mutation. In swap mutation two locations in the route are selected at random then their positions are simply swapped. For example, if swap mutation is applied to the following list, [1, 2, 3, 4, 5] it might end up with, [1, 2, 5, 4, 3]. Here, positions 3 and 5 were switched to create a new list with the same values in just a different order. As swap mutation is only swapping pre-existing values, it will never create a list that has missing or duplicate values when compared to the original, and that is the requirement for the traveling salesman problem [5].

Now crossover method needs to select which can enforce the same constraint. One type of crossover method which can produce a valid route is ordered crossover. In this crossover method subset from the first parent is selected, then that subset has been added to the offspring. Any missing values are then added to the offspring from the second parent in a same order they are found [5]. For example:

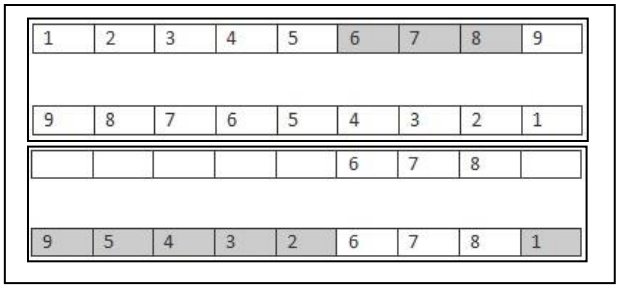


Figure 2: Parent's String and Offspring [5]

Here (6, 7 and 8) subset of the is taken from the first parent and added to the offspring's route. Next, the missing route locations are added from second parent serially from the start. 9 is the first location in the second parent's route, which is not in the offspring's route. So, it is added in the first available position. The next position in second parent's route is 8, which is in the offspring's route, so it is skipped. Until the offspring has no remaining empty values, this process continues. The result should be a route which contains all the positions its parents did with no positions missing or duplicated if correctly implemented [5].

A. **Performance**

Increasing diversity by genetic operators:

- Mutation
- Recombination

Decreasing diversity by selection:

- of parents
- of survivors

B. **Effects of GA Operators**

- Using selection alone will tend to fill the population with copies of the best individual from the initial population.
- Using selection and crossover will tend to cause the algorithm to converge on a good but sub-optimal solution.
- Using mutation alone considers a desultory walk through the search area.
- Utilizing cull and alteration engenders a parallel, noise-patient, hill climbing algorithm.

C. **Limitations**

There are constraints of the utilization of a genetic algorithm balanced to substitute optimization algorithms:

- Reiterated fitness function estimation for intricate obstacle are the most forbidden and restricting segment of artificial evolutionary algorithms [6]. Finding the optimal result to intricate high-spatial, multimodal obstacles repeatedly requires very extravagant fitness function estimations. [7]. In authentic world difficulties namely structural refinement obstacles, a single function estimation may need some hours to few days of consummate simulation. Average estimation techniques cannot be handled with such kinds of difficulties. In this instance, it may be compulsory to forgo an actual refinement and employ a close fitness that is well ordered. [8]. It is presumed that combination of close models may be one of the most auspicious proposals to employ GA to decode composite genuine-life obstacles [9].
- Genetic algorithms do not work vigilantly with complication. That is, where the quantity of components which are uncovered to mutation is immensely colossal there is often an epidemic rise in search space dimension. This shapes it extremely hard to utilize the proposals on obstacles namely designing an engine, a house or plane. To shape such difficulties manageable to evolutionary search, they must be crushed down into the easiest portrayal possible [10]. So, we generally observe evolutionary algorithms encrypting sketches for fan blades in lieu of engines, constructing shapes in lieu of detailed manufacture plans, and airfoils in lieu of whole aircraft sketches [11]. The next obstacle of complication is the matter of how to defend components that have advanced to constitute good findings from new disastrous mutation, categorically when their fitness analysis needs them to merge with other components [10].
- The "better" solution is only in comparison to other solutions. As a result, the stop scale is not understandable in all the difficulties.
- In numerous issues, GAs may tend to unite towards neighborhood optima or even discretionary focuses as opposed to the global optimum [10]. This implies that it does not "know how" to forfeit transient wellness to acquire longer-term wellness. The probability of this happening relies upon the state of the wellness scene: certain issues may give a simple climb towards a global optimum; others may make it simpler for the capacity to locate the neighborhood optima. This issue might be mitigated by utilizing an alternate fitness function, expanding the pace of change, or by utilizing selection procedures that keep an assorted populace of arrangements, albeit the No Free Lunch hypothesis demonstrates that there is no broad answer for this issue [12]. A typical strategy to keep up variety is to force a "niche penalty", wherein, any gathering of people of adequate similitude (niche radius) has a punishment added, which will decrease the portrayal of that bunch in ensuing ages, allowing other (less comparable) people to be kept up in the populace [12]. Another conceivable strategy is basically return part of the populace with haphazardly created people when most of the populace is excessively like one another. Variety is significant in genetic calculations (and genetic programming) since getting over a homogeneous populace does not yield new arrangements. In advancement methodologies and evolutionary programming, variety is not fundamental due to a more prominent dependence on change [10].
- Working on unique informational collections is troublesome, as genomes meet from the get-go towards arrangements which may at this point don't be substantial for later information. A few strategies have been proposed to cure this by expanding hereditary variety by one way or another and forestalling early assembly, either by expanding the likelihood of transformation when the arrangement quality drops (called set off hyper change), or by at times presenting

completely new, arbitrarily produced components into the genetic supply (called arbitrary outsiders). Once more, advancement techniques and evolutionary programming can be actualized with a purported "comma procedure" in which guardians are not kept up and unseasoned parents are chosen distinctly from posterity. This can be more powerful on unique issues.

- GAs cannot successfully take care of issues in which the solitary wellness measure is a solitary right/wrong measure (like choice issues), as it is absolutely impossible to join on the arrangement (no slope to climb) [13]. In these cases, an irregular inquiry may discover an answer as fast as a GA. In any case, if the circumstance permits the result preliminary to be continued giving (perhaps) various outcomes, at that point the proportion of accomplishments to disappointments gives a reasonable wellness measure [10].
- For explicit enhancement issues and issue examples, other streamlining calculations might be more proficient than hereditary calculations as far as speed of assembly. Option and corresponding calculations incorporate development procedures, evolutionary programming, simulated annealing, Gaussian variation, slope climbing, and multitude insight (e.g., ant colony optimization) and techniques dependent on number straight programming [10]. The reasonableness of hereditary calculations is subject to the measure of information on the issue; notable issues frequently have better, more specific methodologies.

III.ACO IMPLEMENTATION ON TSP

At the point when Ant System (AS) was first presented, it was applied to the TSP. At first, three distinct forms of AS were proposed; these were called ant-density, ant-quantity, and ant-cycle [14]. While in ant-density and ant-quantity the ants refreshed the pheromone straightforwardly after a move from a city to a contiguous one, in ant-cycle the pheromone update was just done after all the ants had built the visits and the measure of pheromone saved by every ant was set to be a component of the visit quality. Since ant cycle performed far superior to the next two variations, here we just present the ant cycle calculation, alluding to it as Ant System in the accompanying. In AS every one of m (counterfeit) ants constructs an answer (visit through) the TSP [15]. In AS no neighborhood search is applied. (It would not be easy to add nearby inquiry to AS).

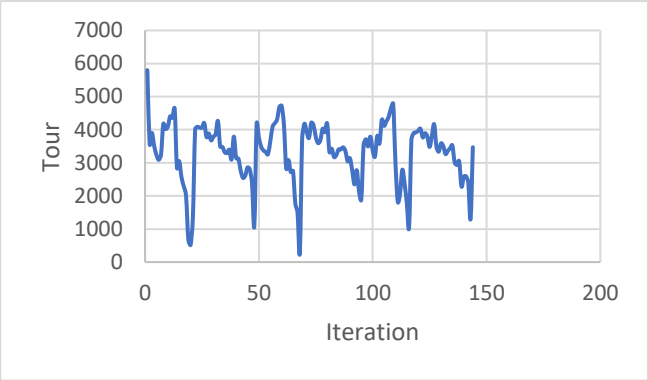


Figure 3: Traveling Salesman Problem using Ant Colony Optimization

This ACO implementation used the Ant-System (AS) variant where the movement from node i to node j is defined by:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum_{u \in \mathcal{N}_i^k(t)} \tau_{iu}^{\alpha}(t)\,\eta_{iu}^{\beta}(t)} & \text{if } j \in \mathcal{N}_i^k(t) \\ 0 & \text{if } j \notin \mathcal{N}_i^k(t) \end{cases} \quad (2)$$

In this ACO execution, showing up at the best arrangement requires adjusting the abuse investigation tradeoff. Setting the dissipation coefficient low makes the pheromones stay longer. Notwithstanding, this was adjusted by setting the way predisposition of the state exceptionally high, so they will investigate more choices close to the course

with the most noteworthy pheromone focus (rather than causally getting comfortable it).

A. Performance

Interestingly with the GA execution, ACO is a lot simpler to control. There are not many boundaries required and the investigation ability does not really run wild. It additionally helps that the ants are arbitrarily positioned in various regions of the guide and permitted to make a "guided" beginning visit. This makes the underlying qualities much lower as thought about on account of GA where beginning expenses are extremely high if an avaricious inquiry is not originally actualized to build the underlying visit.

IV.COMPARISON BETWEEN GA AND ACO ON TSP

We utilize three diverse informational collections to check the effectiveness of these calculations and analyze the outcome among GA and ACO.

A. Experimental Parameters

In GA and ACO the information comprises of 13, 25 and 35 urban communities individually [16]. TSP arrangements evaluated and analyzed by each computational time came about because of all strategies [16]. In this testing the estimations are in time unit and length unit [16].

With different suggested boundaries the recreation is performed multiple times [16]. The changed boundaries are the quantity of urban communities (n), the quantity of ants (m) or the populace size and number of cycles [16]. By utilizing the accompanying boundaries, the best trial result for every technique is acquired:

1) The crossover likelihood, Pc is 0.50 and mutation likelihood, Pm is 005 for essential GA. In addition, we utilized uniform irregular crossover as the crossover strategy [16].
2) For the essential Ant Colony Optimization, α is 1, β is 2, ρ is 0.5, τ is 0.1 and a steady of the path amount, Q is 1 [16].

For the 3 varieties of information, we set the quantity of ants or the populace size 10, 25, 50 respectively and the quantity of iterations 100, 250, 500 separately [16].

Our reproduction shows that GA is the quickest technique, yet the GA's answer is not superior to ACO. The rundown of the reenactment is introduced in Table 1, and the examination of the best recreation results is introduced in Figure 5. In each outline, by placing distinctive name in each line we separate the line [16].

TABLE 1: EXPERIMENTAL RESULT FROM GA AND ACO ON TSP

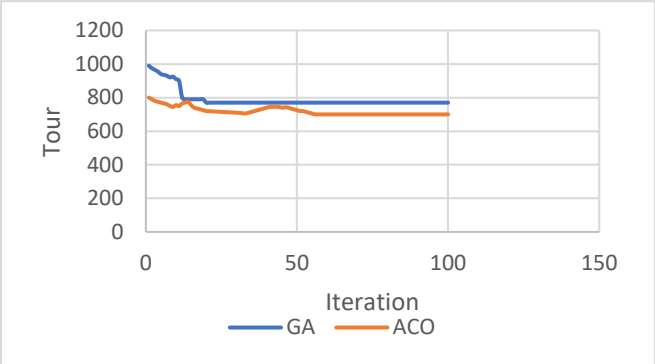
No.	Data	Number of Cities	GA		ACO	
			Time	Best Solution	Time	Best Solution
1	Random 13	13	1.0	77892	1.0	67950
2	Random 25	25	3.0	147178	3.0	80321
3	Random 35	35	8.0	127864	16.0	49998

B. Result Analysis

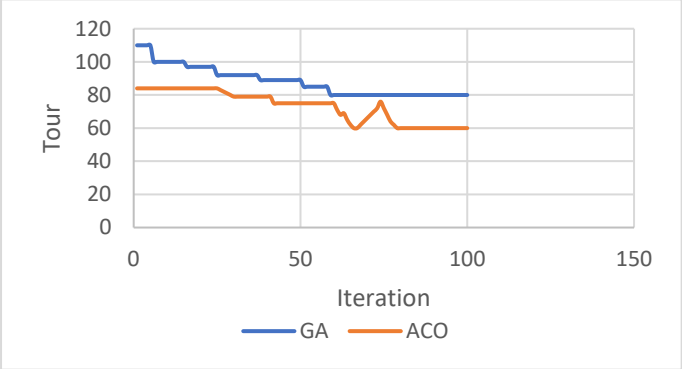
Thinking about the analyses, the GA is anything but a decent methodology by any means. It is sudden discovering on the grounds that GA can fall flat in any cases [16]. Furthermore, ACO gave better arrangement in modest quantity of information.

Here it is seen that for Random13 the best arrangement of GA is 77892 and the arrangement of ACO is 67950. For Random25 the thing that matters is expanding. Also, for Random35 there is an immense contrast. In the bigger measure of information, tests show that ACO can deliver

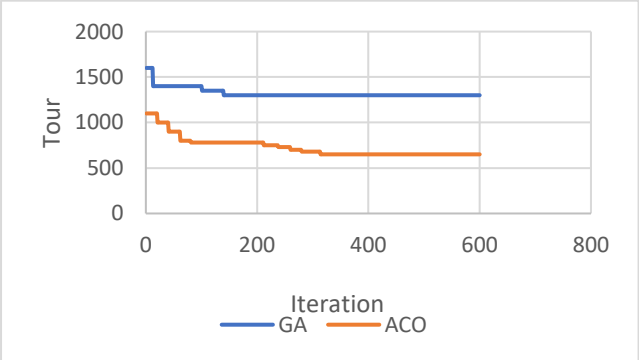
preferred arrangements over the GA technique reliably. In future, it is conceivable to mixture the two calculations and notice the exhibition of that half and half calculation.



a) Traveling Salesman Problem using Genetic Algorithm and Ant Colony Optimization with Random13 Cities



b) Traveling Salesman Problem using Genetic Algorithm and Ant Colony Optimization with Random25 Cities



c) Traveling Salesman Problem using Genetic Algorithm and Ant Colony Optimization with Random35 Cities

Figure 4: Comparison of the best solution in the simulation

Though the running time of ACO is longer than GA but it gave us better solution than GA for large dataset.

V. REFERENCES

[1] . Dr. M. S. Alam, Continuous Optimization with evolutionary and swarm intelligence algorithms, PhD Thesis, Bangladesh University of Engineering and Technology, September 2013.

[2] . Bäck, T., Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press,USA, 1996.

[3] . Mühlenbein, H., "The Breeder Genetic Algorithm – a provable optimal search algorithm and its application", IEEE Colloquium on Applications of Genetic Algorithms, Digest No. 94/067, London, March 15, 1994.

[4] . Dorigo, M. and Stützle, T., Ant Colony Optimization. MIT Press, Cambridge, MA, 2004.

[5] . Li, K., Kang, L., Zhang, W., Li, B., (2008), Comparative Analysis of Genetic Algorithm and Ant Colony Algorithm on Solving Traveling Salesman Problem, , in IEEE International Workshop. Semantic Computing and Systems.

[6] . J. Luo and D. El Baz, "A Survey on Parallel Genetic Algorithms for Shop Scheduling Problems," *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW),* Vancouver, BC, 2018.

[7] . Maumita Bhattacharya, Evolutionary Approaches to Expensive Optimisation, International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 3, 2013.

[8] . M.-R. Akbarzadeh-T, M. Davarynejad and N. Pariz, Adaptive fuzzy fitness granulation for evolutionary optimization, International Journal of Approximate Reasoning, June 2008.

[9] . M. Davarynejad, M. -. Akbarzadeh-T and C. A. Coello Coello, Auto-tuning fuzzy granulation for evolutionary optimization, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence),* Hong Kong, 2008

[10] . Nita H. Shah, Mandeep Mittal, Handbook of Research on Promoting Business Process Improvement Through inventory control techniques, IGI Global Publisher of Timely Knowledge, 2017

[11] . Roman V. Yampolskiy, Neuroevolution Methods Show Significant Success, Evoloution News and Science today, 2020

[12] . Oonsivilai, Anant & Oonsivilai, Ratchadaporn. (2009). A genetic algorithms programming application in natural cheese products. WSEAS TRANSACTIONS on SYSTEMS. 8. 44-54.

[13] . Shidhanta Poddar, Parallel Genetic Algorithm,

[14] . Valdez, Fevrier, Swarm Intelligence: A Review of Optimization Algorithms Based on Animal Behavior, Recent Advances of Hybrid Intelligent Systems Based on Soft Computing.

[15] . St, Thomas & Dorigo, Marco. (1999). ACO Algorithms for the Traveling Salesman Problem.

[16] . Zukhri, Zainudin & Paputungan, Irving. (2013). A Hybrid Optimization Algorithm based on Genetic Algorithm and Ant Colony Optimization. International Journal of Artificial Intelligence & Applications. 4. 63-75. 10.5121/ijaia.2013.4505.

Cite this article as :

Muhammed Yaseen Morshed Adib, Jannatun Razia, Md. Toufiqur Rahman, " Experimental Comparison between Genetic Algorithm and Ant Colony Optimization on Traveling Salesman Problem, International Journal of Scientific Research in Science, Engineering and Technology(IJSRSET), Print ISSN : 2395-1990, Online ISSN : 2394-4099, Volume 8, Issue 1, pp.155-162, January-February-2021. Available at

doi : https://doi.org/10.32628/IJSRSET218135

Journal URL : http://ijsrset.com/IJSRSET218135