Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 19 Issue 3 Version 1.0 Year 2019
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Evaluation of Ant Colony Optimization Algorithm Compared to Genetic Algorithm, Dynamic Programming and Branch and Bound Algorithm Regarding Traveling Salesman Problem

By A.H.M. Saiful Islam, Mashrure Tanzim, Sadia Afreen & Gerald Rozario

*Notre Dame University*

*Abstract-* We use ant colony optimization (ACO) algorithm for solving combinatorial optimization problems such as the traveling salesman problem. Some applications of ACO are: vehicle routing, sequential ordering, graph coloring, routing in communications networks, etc. In this paper, we compare the performance of ACO to that of a few other state-of-the-art algorithms currently in use and thus measure the effectiveness of ACO as one of the major optimization algorithms in regard with a few more algorithms. The performance of the algorithms is measured by observing their capacity to solve a traveling salesman problem (TSP). This paper will help to find the proper algorithm to be used for routing problems in different real-life situations.

*Keywords: swarm intelligence, vehicle routing, ant colony optimization.*

*GJCST-D Classification: F.2.2*

# Evaluation of Ant Colony Optimization Algorithm Compared to Genetic Algorithm, Dynamic Programming and Branch and Bound Algorithm Regarding Traveling Salesman Problem

A.H.M. Saiful Islam$^{\alpha}$, Mashrure Tanzim$^{\sigma}$, Sadia Afreen$^{\rho}$ & Gerald Rozario$^{\omega}$

*Abstract-* We use ant colony optimization (ACO) algorithm for solving combinatorial optimization problems such as the traveling salesman problem. Some applications of ACO are: vehicle routing, sequential ordering, graph coloring, routing in communications networks, etc. In this paper, we compare the performance of ACO to that of a few other state-of-the-art algorithms currently in use and thus measure the effectiveness of ACO as one of the major optimization algorithms in regard with a few more algorithms. The performance of the algorithms is measured by observing their capacity to solve a traveling salesman problem (TSP). This paper will help to find the proper algorithm to be used for routing problems in different real-life situations.

*Keywords: swarm intelligence, vehicle routing, ant colony optimization.*

## I. Introduction

Ant colony optimization algorithm belongs to a special class of artificial intelligence called swarm intelligence. "Swarm intelligence is a relatively new approach to problem-solving that takes inspiration from the social behaviors of insects and of other animals. In particular, ants have inspired a number of methods and techniques among which the most studied and the most successful is the general-purpose optimization technique known as ant colony optimization (ACO)" (Dorigo, Birattari and Stutzle, 2006, p. 28). ACO has a powerful capacity to find out solutions to combinatorial optimization problems. But it has some issues like stagnation and premature convergence. The convergence speed of ACO is always slow.

(Raghavendra BV, 2015). These limitations become more noticeable when the problem size increases, and the number of nodes become more and more numerous. The aim of this paper is to compare the performance of Ant Colony Optimization with a few other algorithms when it comes to solving a particular problem: the traveling salesman problem. Davendra (Davendra D, 2010, Travelling Salesman Problem, Theory and Applications) defined TSP as, "Given a set of cities of different distances away from each other, and the objective of TSP is to find the shortest path for a salesperson to visit every city exactly once and return back [sic] to the origin city". "TSP is an important applied problem with many fascinating variants; like theoretical mathematics, computer science, NP-hard problem, combinatorial optimization, and operation research" (Abid and Muhammad, 2015, p. 1). There have been some works related to this field, attempting to address such a problem (Sudholt and Thyssen, 2012) (Wang Hui, 2012). But they are usually limited to a comparison between 2 algorithms and only a fixed situation, not for dynamically changing situations that might arise in a real-life problem. We analyze different aspects of their performance, like time complexity, space complexity, scalability, etc. and thus determine which algorithm is suitable for which situation. The algorithms we intend to compare with ACO are Genetic algorithm, Dynamic programming, Branch and bound algorithm. Some research papers comparing these algorithms with each other already exist (Wang Hui, 2012). But no work has been done to compare all of their performances at once and for different situations that might arise in a real -ife routing problem. In reality, a route might be blocked due to accidents, or the number of nodes might change due to unforeseen circumstances. In that situation, the performance of different algorithms will be different. We aim to find out which algorithm serves the best in what sort of situation faced in real life.

The second section of this paper shows the process of comparing the algorithms. The third section discusses the results we obtain and its implications, and finally, the conclusion is discussed in the fourth section.

## II. Comparing the Algorithms

To compare four different algorithms, we bring them in the same platform and use them to solve the same dataset. We use multiple datasets to ascertain their performance, to make sure that the algorithms are fairly adaptable to changing situations. We implemented the algorithms using Windows 10 as the operating

*Author α σ ρ Ѡ: Associate Professor, Department of Computer Science and Engineering, Notre Dame University Bangladesh, Dhaka, Bangladesh. e-mails: saiful@ndub.edu.bd, tanzimndub@gmail.com, sadiaafrinkhushbu111@gmail.com, nizelrozer@gmail.com*

system and Java as the programming language. We retrieve the program for ACO from the official ACO metaheuristic site. The software package ACOTSP-1.03 provides an implementation of various Ant Colony Optimization (ACO) algorithms applied to the symmetric Traveling Salesman Problem (TSP) (http://www.aco-metaheuristic.org/aco-code/public-software.html, January 15, 10.40 p.m).

To find the best results, the programs maintain the best universally accepted space and time complexity for their respective algorithms.

The main aspects of the performance comparison are:

- Time: The amount of time it takes to run the algorithm.
- Space: The amount of memory space required to solve an instance of the computational problem.
- Scalability: The ability of the algorithm to adapt to the increasing size of the problem.

In order to determine the various parameters for our comparison, we run the programs in a fixed platform and use a fixed dataset. We run these programs on a dataset bays29, which is a dataset of 29 cities in Bavaria with their street distances (https://github.com/pdrozdowski/TSPLib.Net/blob/master/TSPLIB95/tsp/bays29.tsp, March 3, 9.15 p.m)

To ensure that their performance is consistent, we also use a smaller dataset of 4 cities to test the algorithms. The data set has the following adjacency matrix:

$$\begin{matrix} 0 & 4 & 1 & 3 \\ 4 & 0 & 2 & 1 \\ 1 & 2 & 0 & 5 \\ 3 & 1 & 5 & 0 \end{matrix}$$

Thus, the programs will give a standardized output. We then use the obtained data to formulate graphs and a table to analyze the strengths and weaknesses of each algorithm regarding solving TSP.

We test the ACO algorithm first. The obtained result will set the standard for our research. The algorithm performs reasonably well in terms of time for both large (bays29.tsp) and small (mydataset.tsp) datasets. But it also consumes a considerable amount of memory.

Genetic algorithm is based on the property of reproductive cells. It assumes two separate data bits as chromosomes of two cells and creates a new chromosome from the parent chromosomes. The processes of creating new chromosomes vary. The algorithm does poorly in terms of time for both large and small datasets but performs better in terms of memory usage.

A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions using state space search. The set of candidate solutions is thought of as forming a rooted tree with the full set at the root. The algorithm explores branches of this tree, which represent subsets of the solution set. The algorithm performs admirably in terms of both time consumption and memory use for small datasets. But for large datasets, it enters into an infinite loop. Even after 30 minutes of running, it fails to produce any results. This limitation renders it unusable for large datasets.

Dynamic programming is both a mathematical optimization method and a computer programming method. After the initial emphasis on static problems, some of the focus is now shifting towards dynamic variants of combinatorial optimization problems. Recently some research is being done on TSP for dynamic problems. The program performs very well in terms of both time and memory use for small datasets. But for large datasets like bays29.tsp, it consumes a huge amount of memory. It exceeds the heap size even after setting the heap size at 3 GB. Thus, we can't use it for large datasets.

*Table 1:* Time and Memory use for large datasets (bays29.tsp)

| Algorithms | Time (seconds) | Memory usage (mbs) |
|---|---|---|
| ACO | 3.103 seconds | 55.158203125 mbs |
| Genetic algorithm | 5.50 seconds | 33.696289 mbs |
| Branch and bound | undefined | undefined |
| Dynamic Programming | undefined | undefined |

*Table 2:* Time and Memory use for small datasets (mydataset.tsp)

| Algorithms | Time (seconds) | Memory usage (mbs) |
|---|---|---|
| ACO | 1.6 seconds | 5.250947 mbs |
| Genetic algorithm | 2.30 seconds | 4.86230468 mbs |
| Branch and bound | 0.004 seconds | 1.30078125 mbs |
| Dynamic Programming | 0.002 seconds | 1.9501953125 mbs |

## III. Discussion and Analysis of Result

The results obtained from the codes show us the different aspects of the algorithms in different situations. As mentioned before, we shall compare the algorithms based on the amount of time it takes to run the algorithm, the amount of memory space required to solve an instance of the computational problem and the ability of the algorithm to adapt to the increasing size of the problem. From the obtained results, we find that:

For large datasets, the fastest way to solve the problem is the Ant colony optimization algorithm. It takes the least amount of time among the 4. But it will also consume the most memory of them all.

For large datasets, the cheapest in terms of memory, to solve the problem is the Genetic Algorithm. It takes somewhat longer than ant colony optimization to solve the problem. But performs better than dynamic programming or branch and bound algorithm, none of which can solve bigger datasets efficiently due to heavy memory usage or taking too much time. Thus, both have bad scalability. They can't adapt to problems with more nodes or higher complexity.

For small datasets, the fastest way to solve the problem is Dynamic Programming. It is the quickest method to solve small datasets. Branch and bound algorithm come to a close second. But it is the cheapest method. Both genetic algorithm and ant colony optimization are far behind them in terms of time and memory usage.

The results and their analysis helps us to draw the following conclusion. We arrange the algorithms in the descending order based on the time they take, the amount of memory they use, and how well they scale when faced with more complex problems.

For large datasets, the usefulness of the algorithms in descending order:

*Table 3:* Usefulness for large datasets (bays29.tsp)

| Serial No. | Time | Memory usage | Scalability |
|---|---|---|---|
| 1 | ACO | Genetic algorithm | Genetic algorithm |
| 2 | Genetic algorithm | ACO | ACO |
| 3 | Dynamic Programming | Branch and bound | Branch and bound |
| 4 | Branch and bound | Dynamic Programming | Dynamic Programming |

For small datasets, the usefulness of the algorithms in descending order:

*Table 4:* Usefulness for small datasets (mydataset.tsp)

| Serial No. | Time | Memory usage | Scalability |
|---|---|---|---|
| 1 | Dynamic Programming | Branch and bound | Genetic algorithm |
| 2 | Branch and bound | Dynamic Programming | ACO |
| 3 | ACO | Genetic algorithm | Branch and bound |
| 4 | Genetic algorithm | ACO | Dynamic Programming |

Thus, for solving sizeable problems with many nodes, it's best to use ACO for the fastest and Genetic algorithm for the cheapest results. But for smaller problems with fewer nodes, Dynamic programming is the best algorithm to solve it quickly. Branch and bound algorithm is the primary choice for a cheap solution. This comparison helps us to determine which algorithm performs best under which circumstance. If the routing problem involves many cities or many villages connected with roads, then we use the ant colony optimization to get the fastest result. However, if we are willing to sacrifice time for achieving a lower memory use, we should choose the Genetic algorithm. This method is more suitable when a large amount of data needs to be processed, and the technology available is limited. For a routing problem that works with few nodes, such as the route between divisions, or the interstate highways connecting states, Dynamic programming gives the best result. Since there are few destinations and fewer paths, time and memory consumption is low. But we should be aware that a system made for such a purpose will have bad scalability and will not work on more complex routing problems.

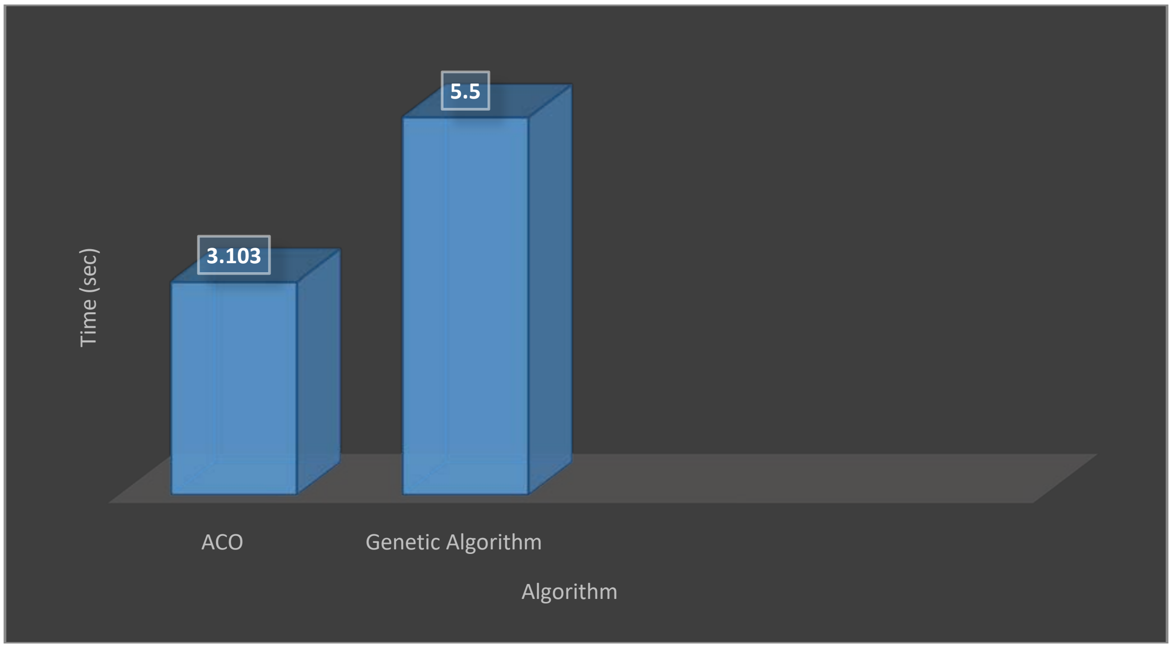

*Fig. 1:* Time comparison for large datasets (bays29.tsp)

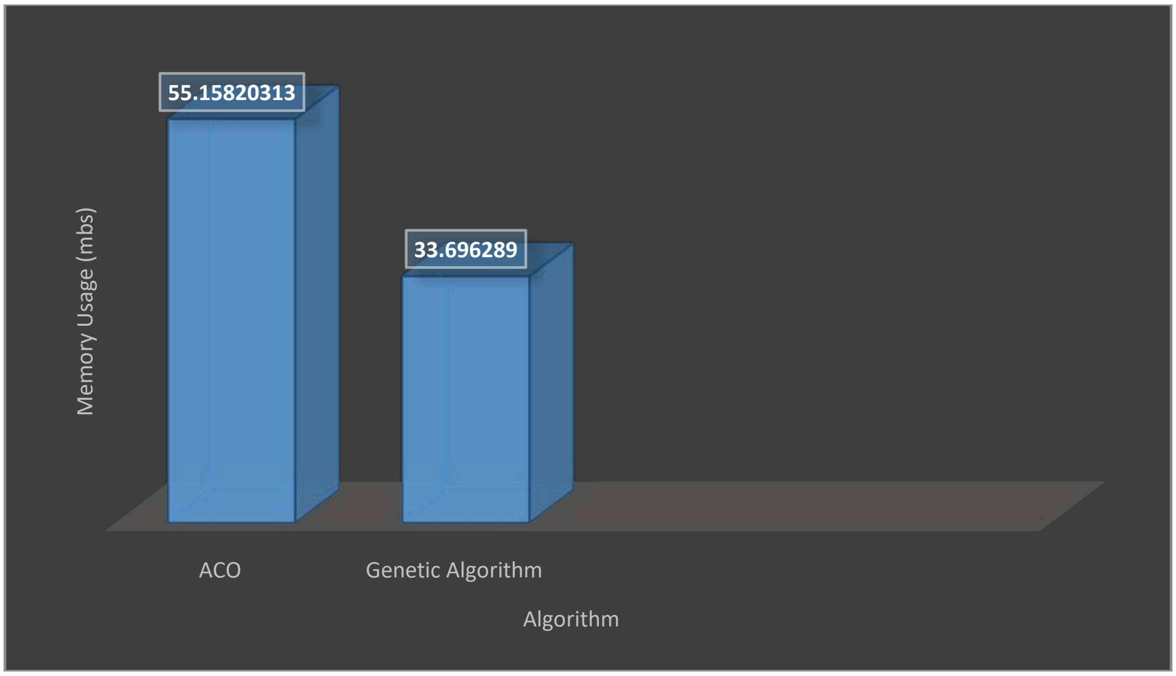

*Fig. 2:* Memory usage for large datasets (bays29.tsp)

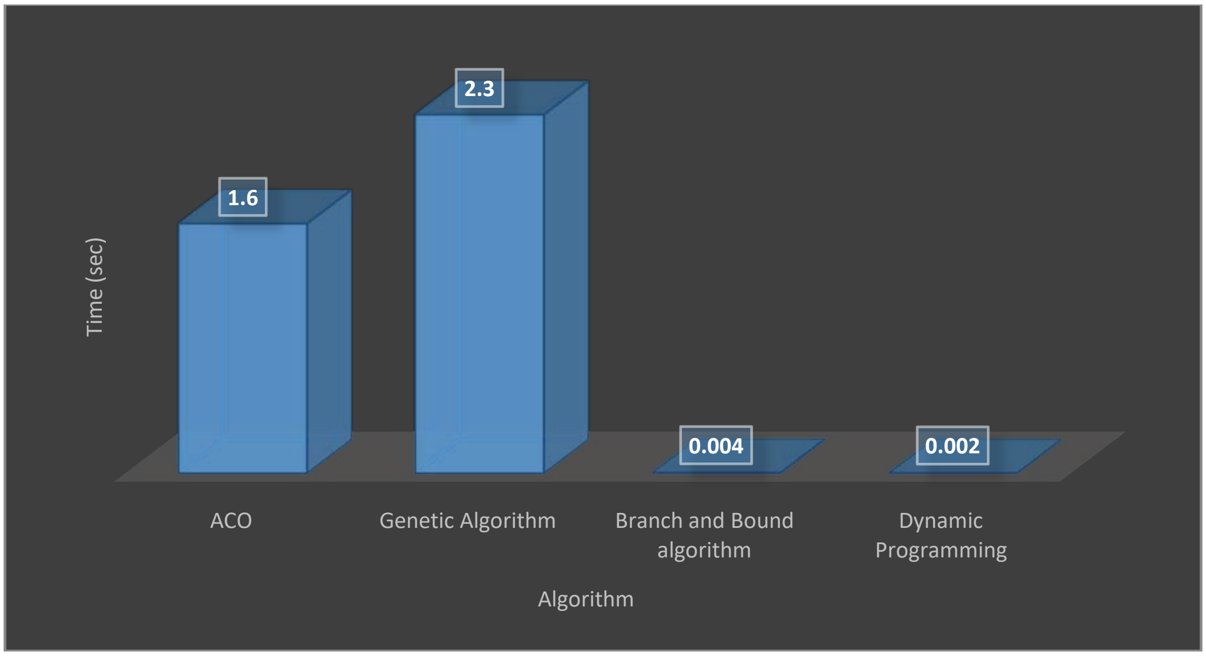

*Fig. 3:* Time comparison for small datasets (mydataset.tsp)

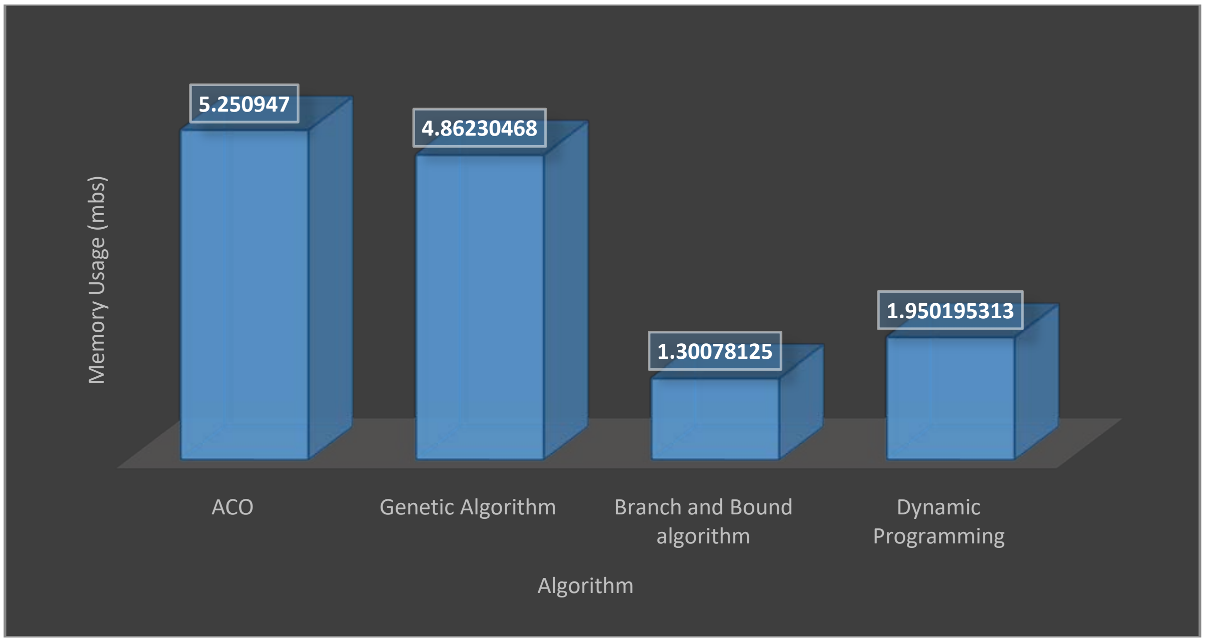

*Fig. 4:* Memory usage for small datasets (mydataset.tsp)

## IV. Conclusion

Traveling salesman problem is one of the most important problem faced by vehicle routing procedures. Choosing the appropriate algorithm for a situation is necessary. This paper presents four different algorithms that can solve the traveling salesman problem and compares their performance. This paper will help future engineers to device the proper algorithm for dynamic and changing situations in vehicle routing and logistics. In real life, the condition on the road can change at any moment due to unforeseen circumstances. In that case, the proper algorithm must be implemented to find the quickest route efficiently. This paper is a step forward in the effort to find the most practical algorithms for continually changing situations in real life.

## References Références Referencias

1. Abid M. and Muhammad I. (2015) Heuristic Approaches to Solve Traveling Salesman Problem. Indonesian Journal of Electrical Engineers and Computer Science, vol. 15, pp 390-396.
2. Davendra D. (2010) Traveling Salesman Problem, Theory and Applications. INTECH open access publishers.

3. Dirk Sudholt and Christian Thyssen. (2012) Running time analysis of Ant Colony Optimization for shortest path problems. Journal of Discrete Algorithms, Vol. 10, pp. 165-180
4. Dorigo M., Birattari M., and Stutzle T. (2006) Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique, IEEE Comput. Intell. Mag,vol. 1,pp. 28-39.
5. http://www.aco-metaheuristic.org/aco-code/public-software.html (January15, 10.40 p.m)
6. https://github.com/pdrozdowski/TSPLib.Net/blob/master/TSPLIB95/tsp/bays29.tsp (March 3, 9.15 p.m)
7. Raghavendra BV (2015) Solving Traveling Salesmen Problem using Ant Colony Optimization Algorithm. J Appl Computat Math 4:260. doi:10.4172/2168-9679.1000260
8. Wang Hui. (2012) Comparison of several intelligent algorithms for solving TSP problem in industrial engineering, The 2nd International Conference on Complexity Science & Information Engineering, pp 226-235.