American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-8, Issue-7, pp-148-153
www.ajer.org
Research Paper Open Access

Travelling Salesman Problem: Complexity Analysis of ACO with the Existing Algorithms

MashrureTanzim
(Notre Dame University Bangladesh,Dhaka,Bangladesh)
Corresponding Author: Sadia Afreen,Gerald Rozario,A.H.M. Saiful Islam(Notre Dame University Bangladesh, Dhaka, Bangladesh)

ABSTRACT :*The application of Ant colony optimization(ACO) for solving difficult combinatorial optimization problems such as the traveling salesman problem dates back to the nineties. Recent applications of ACO cover problems such as vehicle routing, sequential ordering, graph coloring, routing in communications networks etc. In this paper, the performance of ACO is compared to that of a few other algorithms currently in use and thus measure the effectiveness of ACO as one of the major optimization algorithms in regard with other algorithms. The performance of the algorithms are measured by observing their capacity to solve a specific NP-hard problem: the travelling salesman problem (TSP).*

KEYWORDS:*ant algorithms, ant colony optimization, swarm intelligence, metaheuristics.*

Date of Submission: 05-07-2019 Date of Acceptance: 21-07-2019

I. INTRODUCTION

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) [1].Although ACO has a powerful capacity to find out solutions to combinatorial optimizationproblems, it has the problems of stagnation, premature convergenceand the convergence speed of ACO is always slow. These problems become more obvious when the problem size increases [4].

We intend to compare the performance of Ant Colony Optimization with some other algorithms when it comes to solving a very specific problem: the travelling salesman problem.Davendra[5] 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 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 [6].

We analyze 3 aspects of the algorithms: time taken, memory usage, andscalability. Thus, wedetermine which algorithm is suitable for which circumstance. The algorithms we will compare with ACO are: Held-Karp algorithm, Genetic algorithm, Dynamic programming and Branch and bound algorithm. Some research papers comparing these algorithms individually has been published [2]. But no work has been done to compare all of their performances at once and for different situations that might arise in a real life routing problem. In different situations, the performance of different algorithms will be different. We intend to find out which algorithm serves the best in what sort of situation faced in real life. We will use computer programs written to implement these algorithms for solving a TSP and then run those programs on some datasets to get the results.

Comparison of the algorithms is shown in the second section of this paper. The third section discusses the results we obtain and its implications, and finally, the fourth section discusses the conclusion.

II. COMPARISON OF THE ALGORITHMS

The main parameters of the performance comparison are:

- Time cost.
- Memory usage.
- Scalability.

Time cost determines the time it takes to run a full tour with the algorithm. Memory usage denotes the memory cost of the algorithm. Scalability determines how well the algorithm can adapt when the problem size increases.

In order to determine the various parameters for our comparison, we run the programs in a fixed platform and use a fixed dataset. We apply these programs on a dataset bays29, which is a dataset of 29 cities in Bavaria with their street distances [3]. 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:

```
0 5 1 3
5 0 2 1
1 2 0 4
3 1 4 0
```

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

Held-Karp algorithm is a specialized heuristics for solving TSP. It was designed just for this purpose. Its performance for solving the larger dataset is almost comparable to ACO. But it takes an inconvenient amount of memory for implementing bigger datasets. For smaller datasets, it also perform better than ACO.

The ACO is a type of swarm intelligence algorithm. The algorithm performs reasonably well in terms of time for both large (bays29.tsp) and small (mydataset.tsp) datasets. But it consumes a considerable amount of memory

Genetic algorithm is modeled on the reproduction of human embryos. 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 by means of 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 very well in terms of both time and memory use for small datasets. But for larger datasets, it enters into an infinite loop. Even after half an hour of running the code, it fails to produce any results. This renders it unusable for larger 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 shiftingtowards dynamic variants of combinatorial optimization problems. Recentlysome 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. It can’t be used for large datasets.

Table I Time cost for large datasets (bays29.tsp)

Algorithms	Time (seconds)
ACO	3.103 seconds
Genetic algorithm	5.50 seconds
Branch and bound	undefined
Dynamic Programming	undefined
Held-Karp Algorithm	2.8 seconds

Table II Memory use for large datasets (bays29.tsp)

Algorithms	Memory usage (mbs)
ACO	55.158203125 mbs
Genetic algorithm	33.696289 mbs
Branch and bound	undefined
Dynamic Programming	undefined
Held-Karp Algorithm	73.8932 mbs

Table III Time cost for small datasets (mydataset.tsp)

Algorithms	Time (seconds)
ACO	1.6 seconds
Genetic algorithm	2.30 seconds
Branch and bound	0.003 seconds
Dynamic Programming	0.002 seconds
Held-Karp Algorithm	0.28 seconds

Table IV Memory use for small datasets (mydataset.tsp)

Algorithms	Memory usage (mbs)
ACO	5.250947 mbs
Genetic algorithm	4.86230468 mbs
Branch and bound	1.9501953125 mbs
Dynamic Programming	1.30078125 mbs
Held-Karp Algorithm	1.61592048 mbs

III. RESULTANALYSIS

After implementing the algorithms on the datasets, we find that:

Ant colony optimization algorithm isthe fastest way to solve the problem for large datasets. It takes the least amount of time among the 5. But it will also consume the most memory of them all, except Held-Karp algorithm.

Genetic algorithm can solve the problem with the lowest memory cost for large datasets. 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 larger datasets efficiently due to heavy memory usage or too long time. Thus, both have bad scalability. They can't adapt to larger problems.

Dynamic programming is the fastest method to solve small datasets. It is both the quickest and the cheapest method to solve small datasets. Branch and bound algorithmcomes to a close second. Both genetic algorithm and ant colony optimization are far behind them in terms of time and memory usage.

We draw the following conclusion from these findings. 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 larger problems.

Table V Algorithm usefulness for large datasets (bays29.tsp)

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

Table VI Algorithm usefulness for small datasets (mydataset.tsp)

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

We can see that Held- Karp algorithm is a nice compromise between adaptability and performance. It isn't the best at performing at both large and small datasets. But it is versatile and can give reasonable performance in both situations. Useful when a system needs to perform in a multitude of environments. It is especially useful for solving smaller datasets where it can function better than both ACO and genetic algorithm. For solving large problems with many nodes, it's the 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. Branch and bound algorithm is another option.

This analysis helps us to determine that which algorithm performs best under which situation. If the routing problem involves many cities or many villages connected with roads, then we use 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 Genetic algorithm. This 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: route between divisions, or the interstate highways connecting states, Dynamic programming gives the best result. Since there are few destinations and fewer routes, the 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.

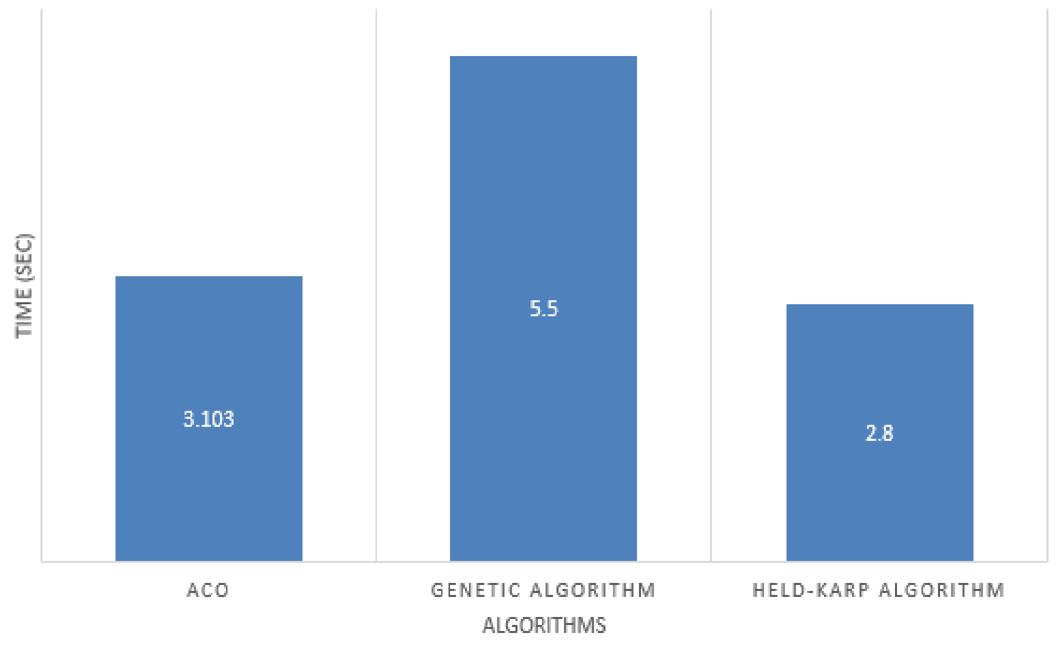

Fig. 1. Time comparison for large datasets (bays29.tsp)

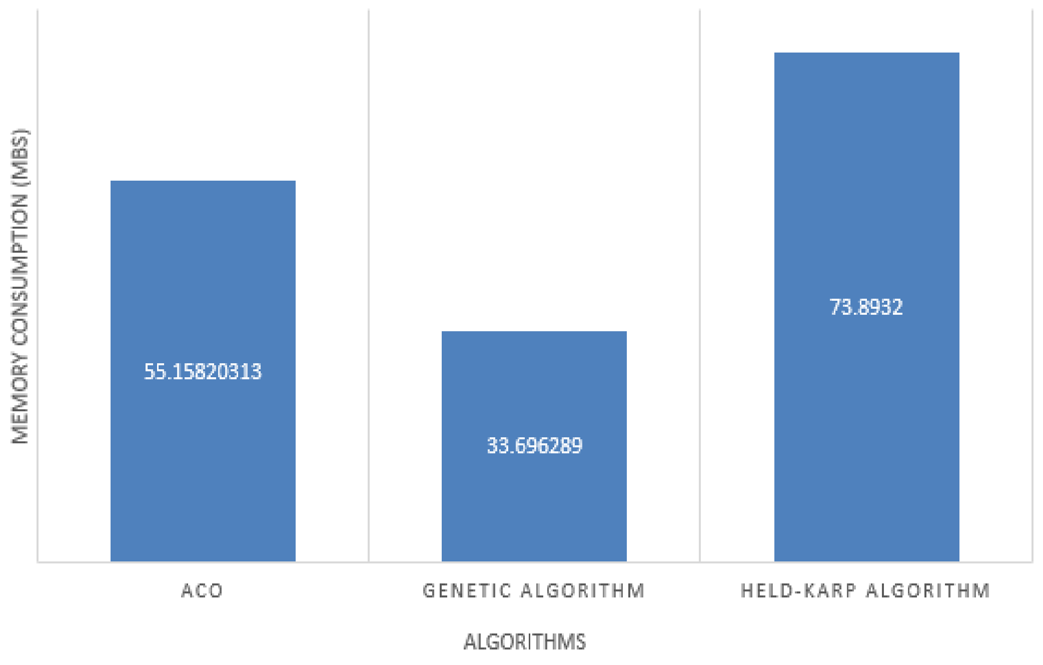

Fig. 2. Memory comparison for large datasets (bays29.tsp)

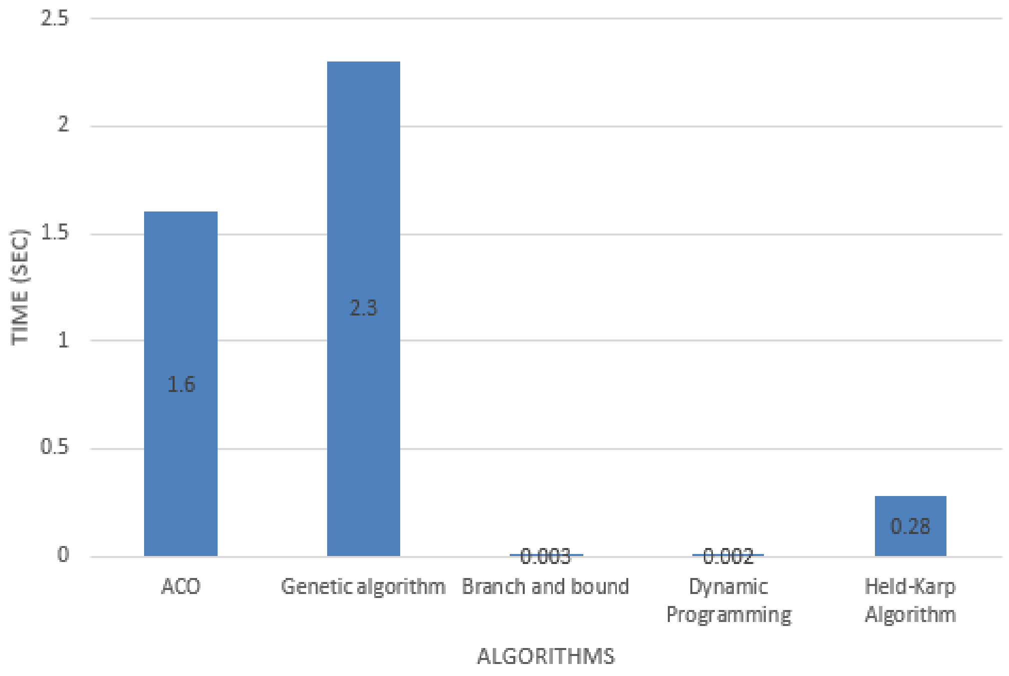

Fig. 3. Time comparison for small datasets (mydataset.tsp)

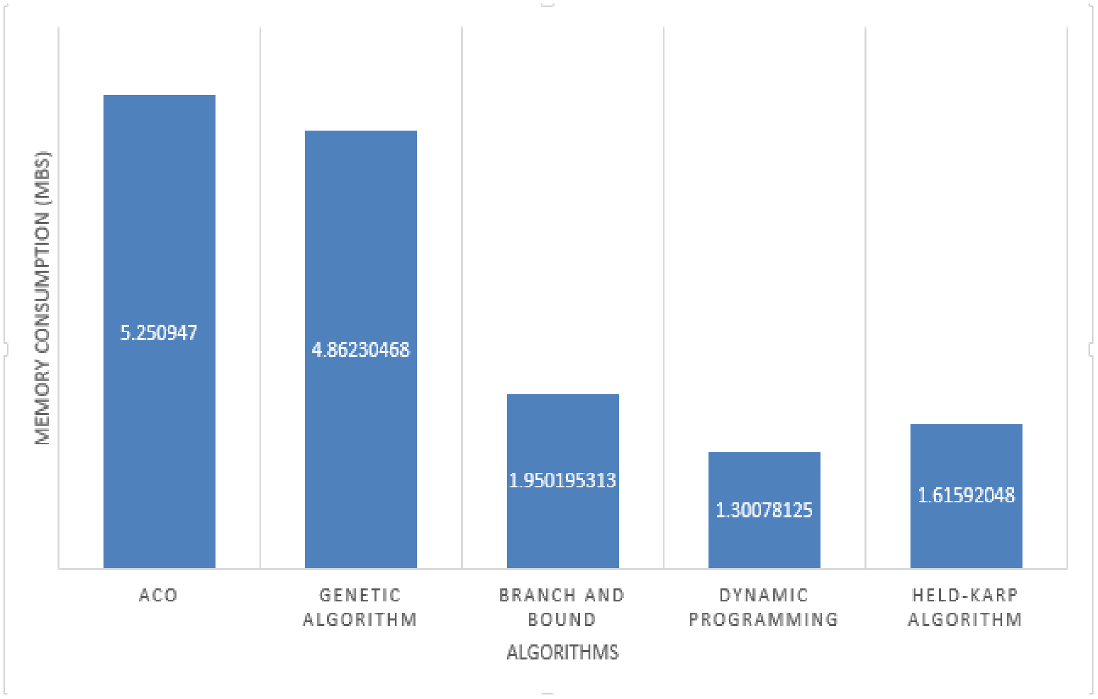

Fig. 4. Memory comparison for small datasets (mydataset.tsp)

IV. CONCLUSION

One of the first problems faced by vehicle routing procedures is the travelling salesman problem. It is important to choose the right algorithm in the right situation. In this paper, we presented five different algorithms that can solve the travelling salesman problem and compared their performance. This will help the future mathematicians and engineers to choose the proper algorithm for dynamic and constantly changing situations in vehicle routing and logistics. In real life, the situation in the field can change at any moment due to unforeseen circumstances and events. In such a case, the proper algorithm must be implemented to find the quickest or most efficient route. This paper is a step forward in the effort to find the most practical algorithms for real life problems.

REFERENCES

[1]. Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant ColonyOptimization,Artificial Ants as a Computational Intelligence Technique, IEEE Comput. Intell. Mag,vol. 1,pp. 28--39, 2006

[2]. Wang Hui.Comparison of several intelligent algorithms for solving TSP problem in industrial engineering, The 2nd International Conference on Complexity Science & Information Engineering, pp 226-235, 2012

[3]. https://github.com/pdrozdowski/TSPLib.Net/blob/master/TSPLIB95/tsp/bays29.tsp (March 3, 9.15 p.m)

[4]. Raghavendra BV (2015) Solving Traveling Salesmen Problem using Ant Colony Optimization Algorithm. J ApplComputat Math 4:260. doi:10.4172/2168-9679.1000260

[5]. Davendra D. Traveling Salesman Problem, Theory and Applications. INTECH open access publishers, 2010.

[6]. Malik MuneebAbid, Iqbal Muhammad. Heuristic Approaches to Solve Traveling Salesman Problem. Indonesian Journal of Electrical Engineers and Computer Science, vol. 15, pp 390-396, 2015.