Comparison of Genetic Algorithm, 2-Opt Nearest Neighbour and Ant Colony Optimisation to Solve The Travelling Salesman Problem

Abdullah Marghoobul Haque
School of Computer Science
University of Nottingham Malaysia
hcyah2@nottingham.edu.my

Kanishka Jayakody
School of Computer Science
University of Nottingham Malaysia
hcykj1@nottingham.edu.my

Zabir Shahid
School of Computer Science
University of Nottingham Malaysia
hcyzs1@nottingham.edu.my

***Abstract*—This paper evaluates and compares the performance of three algorithms – Genetic Algorithm, 2-Opt Nearest Neighbour and Ant Colony Optimisation on an instance of the Travelling Salesman Problem containing 107 cities. These algorithms have been implemented in Java and their results and execution times are compared. It was found that the 2-Opt Nearest Neighbour algorithm produced the best results (~46k) as well as had the shortest execution time (<1s) whereas the Genetic Algorithm had the worst performance (>50k, >15s), and the Ant Colony Optimisation had the second-best performance in terms of shortest distance obtained (~47k) and execution time (~3s).**

Keywords—Travelling Salesman Problem, Ant Colony Optimisation, 2-Opt Optimisation Nearest Neighbour, Genetic Algorithm

I. Introduction

The Travelling Salesman Problem (TSP) is one of the most classical and popular optimisation problems. TSP is categorised as an NP-hard [1] problem because it is a large and complex problem that consists of (n-1)!/2 possible tours for symmetric TSP [2]. The objective of TSP is to find the shortest distance by visiting all cities and returning to the starting point. The practicality of TSP is quite important as some of its typical applications include vehicle routing, computer wiring and job sequencing.[3]

There have been multiple algorithms that have been explored and implemented to solve the TSP. In this paper, Ant-Colony Optimisation Algorithm (ACO), 2-Opt Nearest Neighbour (2-NN) and Genetic Algorithm (GA) will be implemented and their results compared. These algorithms have a lower computational speed than traditional exact algorithms in solving the TSP. [2]

II. Literature Review

There have been several algorithms implemented to get a quick and efficient solution to the TSP. However, this paper will only focus on three of them, namely Genetic Algorithm, 2-Opt Nearest Neighbour and Ant Colony Optimisation.

Genetic algorithm produces good results for various TSP instances, with run time increasing as the number of cities is increased [3]. A study by Haroun et al concluded that GA is fast, easy to implement and cost efficient in terms of computational resources [5]. Another study proved that compared to newer algorithms such as the Particle Swarm Optimization, the Genetic Algorithm provides better solutions [6].

In a study by Mukhairez et al, the Ant Colony Optimisation produced the best results when compared to GA and Simulated Annealing, however in terms of execution speed it was not the fastest [7]. Similarly, a thesis stated that ACO yields better results than GA without the consideration of execution times [5]. In the study by Haroun et al, it was concluded that ACO was best for large complex problems and provided high quality solutions in noticeable time [8].

According to two studies, the best results are obtained using hybrid approaches [14][15]. A study concluded that the Nearest Neighbour (NN) algorithm provided close to optimal solutions in a really low execution time [16]. In a study by Dian et al, it was proven that the hybrid approach of solving TSP using 2-opt and NN (2-NN) provided solutions very close to the optimal solution and outperformed both the individual algorithms [12].

III. Methdology

A. *Genetic Algorithm*

Genetic Algorithms are computation algorithms whose development for optimisation has been inspired by the study of Darwin on evolution and natural selection [9]. The main idea behind GAs is the application of selection, mutation, and recombination to a population of solutions. If given enough time, this algorithm can generate global optimum, making it a good option for a problem like TSP [10]. The basic steps of the algorithm are as follows:

Initialisation: Generate a random initial population

Evaluation: Evaluate each member of the population and calculate a 'fitness' value for that individual

Selection: Keep only the best individuals in the population

Crossover: Create new individuals by combining aspects of selected individuals. The motivation is to combine certain traits from two or more individuals to obtain an offspring that inherits the best traits from each of its parents.

Mutation: Maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next by making small changes at random to an individual's genome.

These steps are then repeated, from the second step, until a termination criterion has been reached [6].

B. *Ant Colony Optimisation*

Ant Colony Optimisation (ACO) is a population-based metaheuristic that is based on the behaviour of real ants [4]. Using data acquired from a pheromone trail left on the edges of the TSP graph, ants in the simulated colony can make successively shorter viable journeys [7]. It performs the following steps [11]:

1. Initialise parameters
2. Determine the probability of pheromone on each path
3. Apply local search and save best solution
4. Update pheromone on each path till max iteration is reached
5. Pick optimal solution and update generated solution as best solution if termination condition is satisfied, else go to step 3

C. *2-Opt Nearest Neighbour Optimisation*

Nearest Neighbour is a greedy algorithm that computes the best possibility (the shortest distance from a node) in each step. Given the coordinates of several points, the distance between two points is computed by the Euclidean distance [12]. The algorithm mainly consists of four following steps:

1. Select an arbitrary node as a starting point
2. Find the shortest unvisited city from the last visited city then go there
3. Is there still an unvisited city? If yes, back to step 2, if not go to step 4
4. Back to the starting point [12].

In optimization, the 2-opt algorithm is a simple local search algorithm. The idea of the 2-opt algorithm is to identify two inter-crossed routes and then rearrange them not to cross each other by removing two edges and adding two new edges.

The hybrid heuristic implementation is simply constructed by the following steps [13] :

1. Apply the nearest neighbour algorithm to construct an initial route by trying all possible starting points to obtain the best initial route.

2. Apply the 2-opt algorithm to improve the route generated by the nearest neighbour algorithm.

IV. Result and Discussion

The three algorithms were run on Java on an AMD Ryzen 5 4000 series processor to obtain fair results as different processors provide different execution times. The Euclidean method was used to compute the distances between two cities. Since the final output is not always the same, each algorithm is run atleast 10 times and the average distance and execution time is considered.

The GA contains the following parameters – population size, mutation rate, crossover rate, elitism count, tournament size, and number of generations. The parameters' functionalities are as follows:

Population Size – Number of candidate solutions which exist in a population at a given time.

Mutation Rate – Probability that parts of a chromosome will mutate.

Crossover Rate – Probability that offspring is made from parts of a parent's chromosome.

Elitism Count – Number of solutions passed to the next generation without any change.

Tournament Size – Number of solutions in the tournament selection process.

Number of Generations – Number of iterations the algorithm is run for.

The figures below show the effects on distance and execution time which was a result of experimentation using different parameters. It is to be noted that all other parameters have been kept constant when checking one parameter.

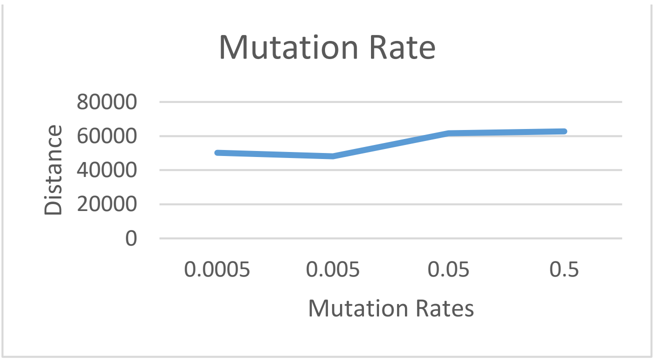

Fig. 1. Performance of GA with different mutation rates

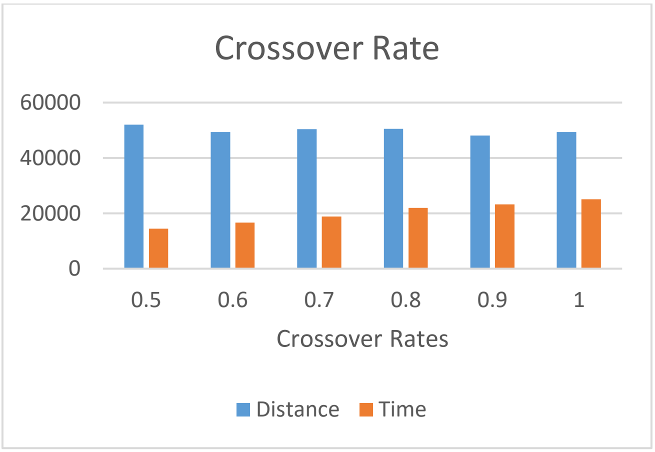

Fig. 2. Performance of GA with different crossover rates

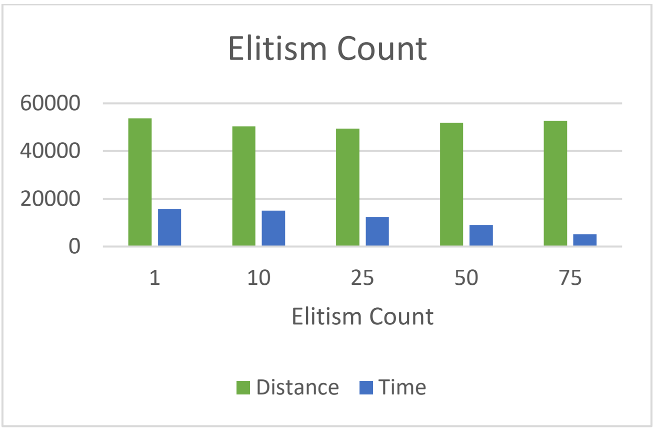

Fig. 3. Performance of GA with different elitism counts

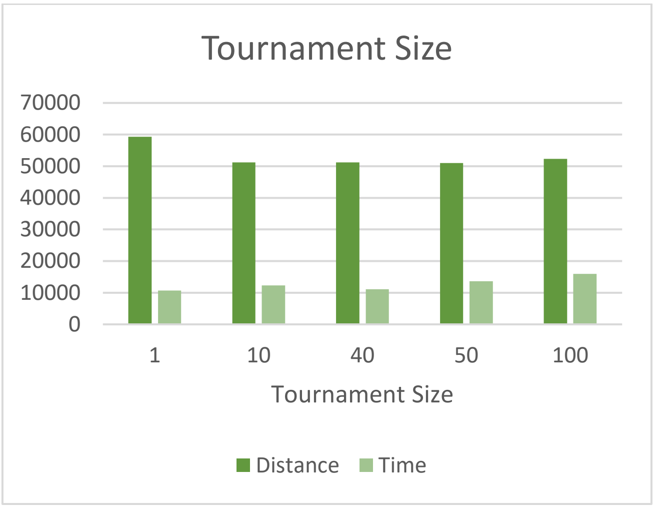

Fig. 4. Performance of GA with different tournament sizes

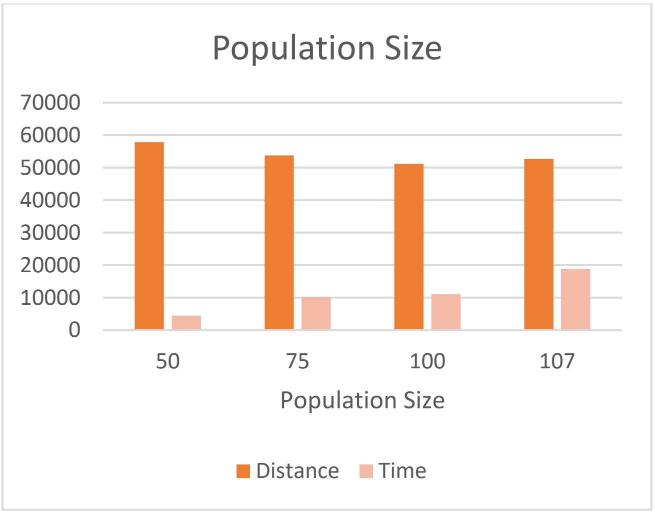

Fig. 5. Performance of GA with different population sizes

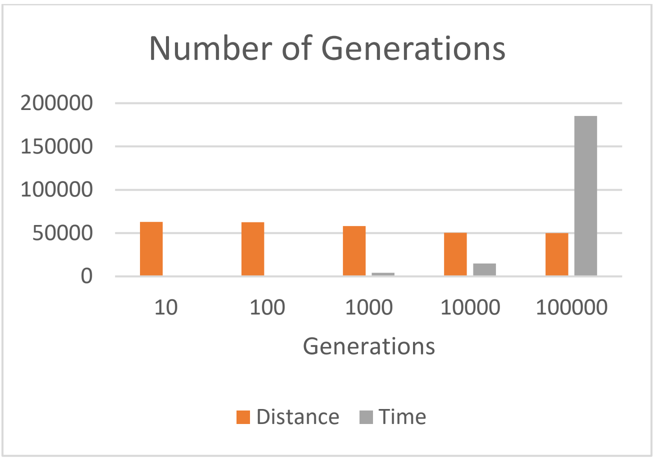

Fig. 6. Performance of GA with different number of generations

Changing some parameters have clear effects on the distance obtained and time taken for execution. From Fig 1. it can be noticed that the ideal mutation rate is 0.005 as it produces the most ideal distance as compared to other values. Also, the time taken for execution with the different mutation rates was all similar, so it has not been considered in this case. Similarly, from Fig 2. Changing cross-over rates did not have a huge difference on the distance however the execution time varied. A rate of 0.6 was determined to be ideal since the distance and time obtained with it was relatively low. In Fig 3., 4., and 5., it can be noticed that the parameter values that produced the best distances were selected. In Fig 6., the number of generations was set to 10,000 to obtain an optimal distance in a satisfactory time even though lesser generations were producing faster results at the cost of the distance obtained. Hence, the final parameters selected were as shown in Table 1.

TABLE I. GENETIC ALGORITHM PARAMETERS

Parameter	Value
Population Size	100
Mutation Rate	0.005
Crossover Rate	0.6
Elitism Count	25
Tournament Size	40
Number of Generations	10000

ACO contains the following parameters:

Trials – The number of attempts that the algorithm is run

Alpha and Beta – Factors that control the algorithm from exploiting the pheromone trails and exploring the search space

Evaporation Rate – Rate at which pheromone on a trail evaporates over time. Helps from getting stuck in suboptimal solutions

Pheromone Remaining – Determines the pheromones left on the trails explored by the ants

Ant Factor – Factor that controls the number of ants used in each iteration

Random Factor – Factor that controls the randomness of the ants' decision making.

The figures below show the effects on distance and execution time which was a result of experimentation using different parameters. It is to be noted that all other parameters have been kept constant when checking one parameter.

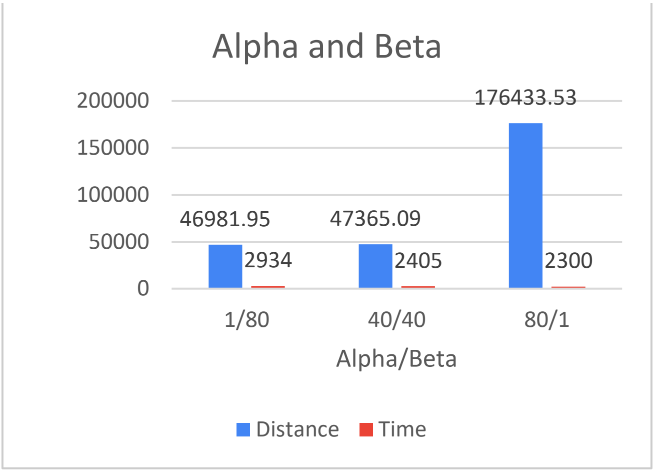

Fig. 7. Performance of ACO with different alpha and beta values

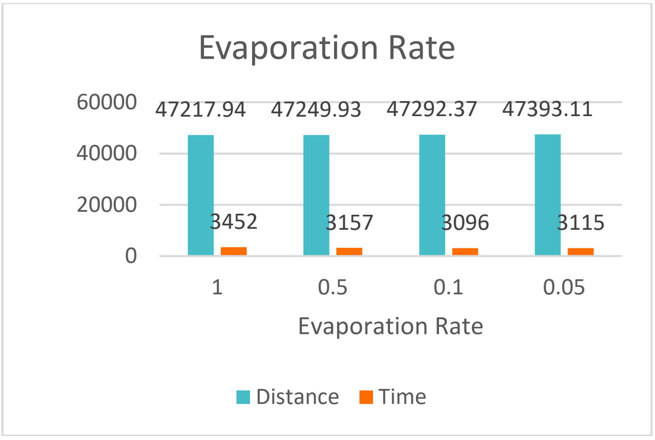

Fig. 8. Performance of ACO with different evaporation rates

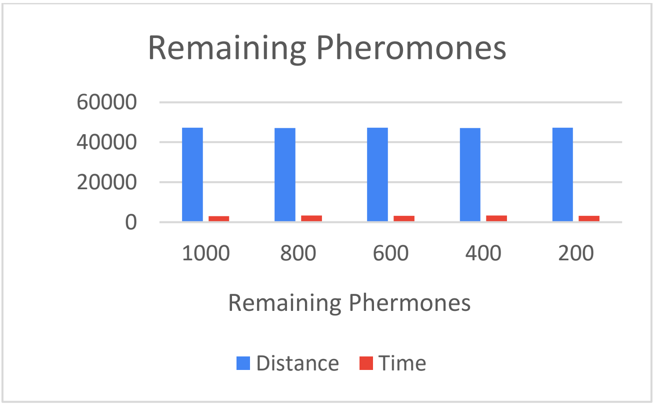

Fig. 9. Performance of ACO with different Remaining Pheromones

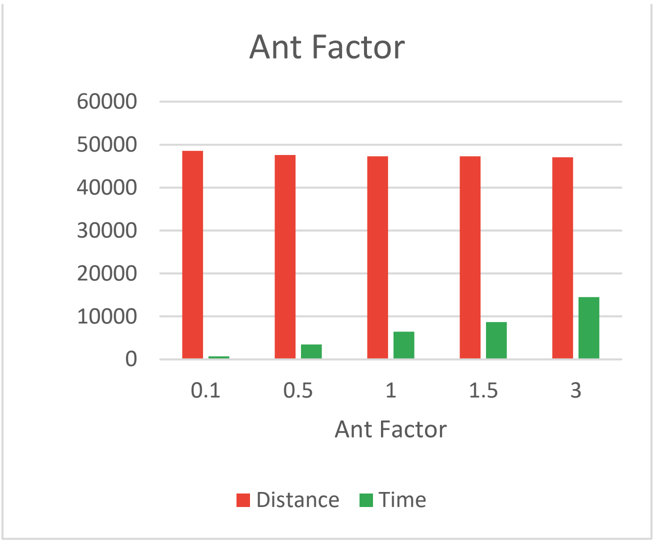

Fig. 10. Performance of ACO with different Ant Factors

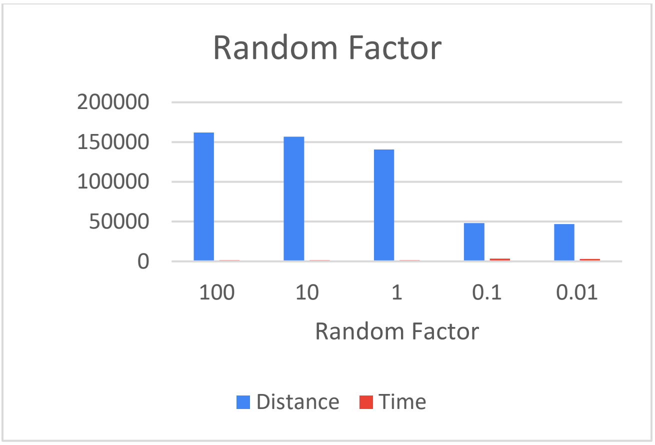

Fig. 11. Performance of ACO with different Random Factors

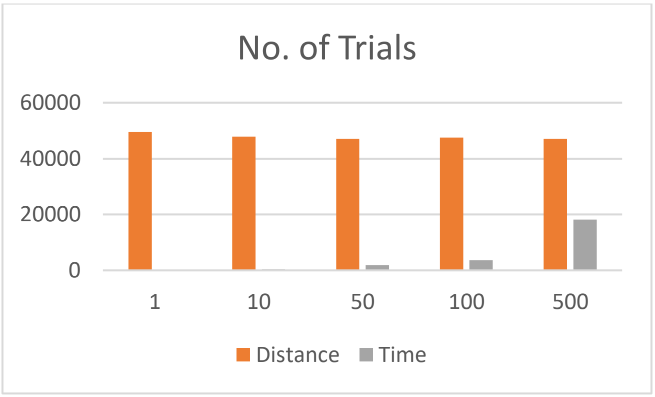

Fig. 12. Performance of ACO with different number of trials

From Fig 7., it can be noticed that increasing alpha to a value higher than beta causes the algorithm to produce an distance which is not optimal. However, if alpha is set to lesser than beta, the distance obtained is more optimal although the time taken on average is greater. In Fig 8., and 9., it can be concluded that tampering these two parameters has no drastic effect on the results produced by the algorithm. Based on Fig 10., and Fig 11., it can be concluded that the optimal values for ant factor and random factor are 0.5 and 0.001 respectively as these produce the optimal distance in a good enough execution time. From Fig 12., it can be said that increasing the number of trials has no significant improvement over the quality of distance, rather it only increases the execution time. Hence, the final parameters selected were as shown in Table 2.

TABLE II. ACO PARAMETERS

Parameter	Value
Trials	50
Alpha, Beta	1, 80
Evaporation Rate	0.5
Pheromone Remaining	800
Ant Factor	0.5
Random Factor	0.001

The 2-NN algorithm is automatic and does not have any parameters to be tuned according to the algorithm. However, it is to be noted that in each execution of the algorithm, the starting city is not always the same.

Since different executions provide different results, the algorithms were run ten times and their average results were calculated for comparison.

TABLE III. RESULTS OBTAINED

Parameter	Distance	Time(ms)
ACO	47252.55	3044.7
2-NN	46881.12	169
GA	51223.34	16240.2

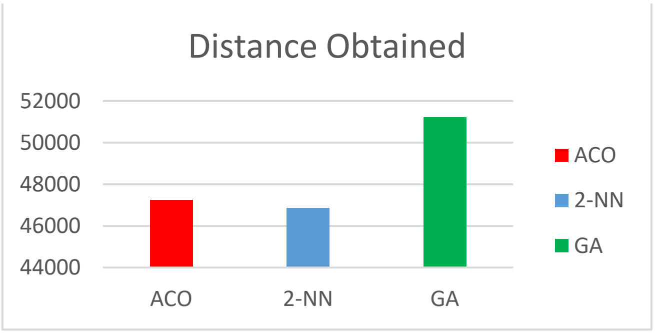

Fig. 13. Distance obtained from the three algorithms

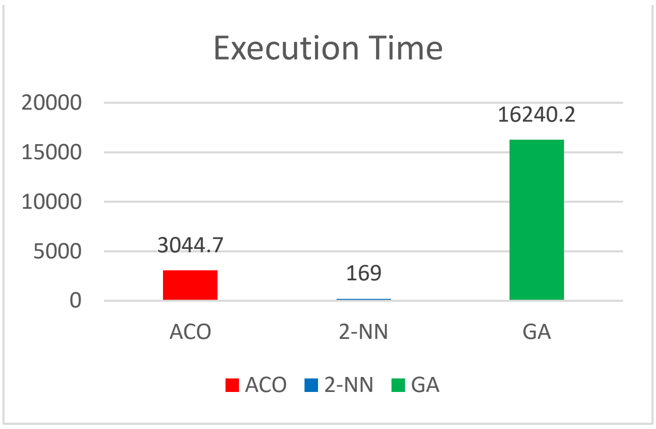

Fig. 14. Execution time of the three algorithms

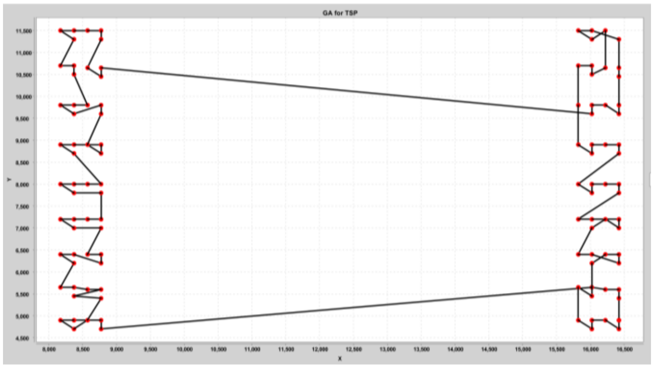

Fig. 15. Path generated by GA

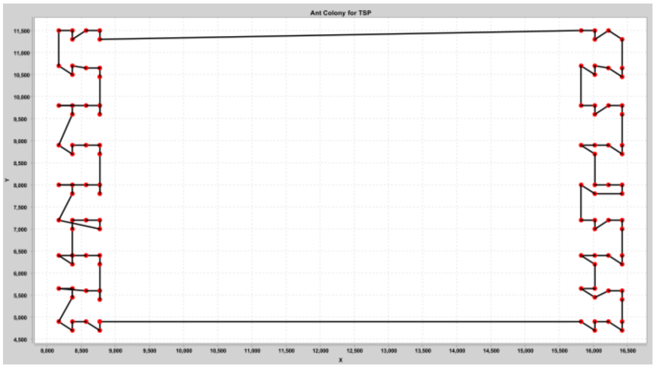

Fig. 16. Path generated by ACO

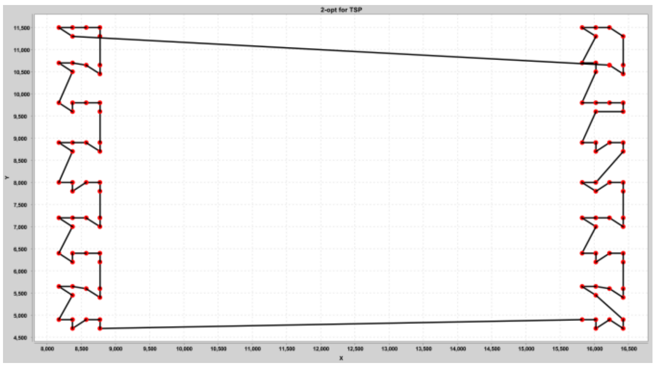

Fig. 17. Path generated by 2-NN

The GA finds global optima better than the other algorithms; however, it takes a really long execution time. However, it produces by far the worst results in terms of distance obtained as well as execution time. This is because GA uses permutations between cities to find the best route, but they are random so there is no guarantee offered on the optimal solution. Furthermore, there are high chances that the GA does not converge to a valid solution.

ACO obtains a better distance than GA and does so in a better time than GA.

Both GA and ACO are difficult to tune because of the large number of parameters that they contain.

2-NN provides solutions closer to the optimal solution as compared to the other two algorithms. However, the execution time of the 2-NN algorithm heavily depends on the comparisons and iterations done by the algorithm during execution as well as the initial solution generated. Nevertheless, it still runs faster than ACO by 18 times and GA by 96 times. Moreover, the 2-NN algorithm is more prone to be stuck in the local minima as compared to the other two.

Conclusion

In this paper, three algorithms are implemented – GA, 2-NN and ACO on a TSP dataset containing 107 cities. It can be observed from the results that the 2-NN hybrid algorithm produced the results closest to the optimal solution, whilst utilising the least computing resources. The ACO algorithm performed slightly worse but consumed more resources whereas the GA had the worst performance and consumed the most resources, producing an execution time of about 100 times more than the 2-NN algorithm.

References

[1] Russell, S. and Norvig, P., Artificial Intelligence: A Modern Approach, New Jersey: Pearson Education Inc, 2010.

[2] Agung Chandra and Christine Nataloa, A Comparative Study Of Optimization Methods For Traveling Salesman Problem, Academic Journal Of Manufacturing Engineering, Vol.19, Issue 4/2021.

[3] A. Philip, A. A. Taofiki, and O. Kehinde, ‘A Genetic Algorithm for Solving Travelling Salesman Problem’, 2011.

[4] Agarwal A and Singh R, Comparative analysis of travelling salesman problem using metaheuristic algorithms, Communication and Computing Systems - Proceedings of the International Conference on Communication and Computing Systems, ICCCS 2016

[5] S. A. Haroun, B. Jamal, and E. H. Hicham, “A Performance Comparison of GA and ACO Applied to TSP,” *Int J Comput Appl*, vol. 117, no. 20, pp. 28–35, May 2015, doi: 10.5120/20674-3466.

[6] A. Gharib, J. Benhra, and M. Chaouqi, “A Performance Comparison of PSO and GA Applied to TSP,” *Int J Comput Appl*, vol. 130, no. 15, pp. 34–39, Nov. 2015, doi: 10.5120/ijca2015907188.

[7] H. H. A. Mukhairez and A. Y. A. Maghari, “Performance Comparison of Simulated Annealing, GA and ACO Applied to TSP,” *International Journal of Intelligent Computing Research*, vol. 6, no. 4, pp. 647–654, Dec. 2015, doi: 10.20533/ijicr.2042.4655.2015.0080.

[8] “Comparative Analysis of Ant Colony Optimization and Genetic Algorithm in Solving the Traveling Salesman Problem Hatem Mohi El Din,” 2021.

[9] N. Sureja, “Random Travelling Salesman Problem using Genetic Algorithms multi-site software ontology View project.” [Online]. Available: www.ifrsa.org

[10] P. Vaishnav, “A Comparative Study of GA and SA for Solving Travelling Salesman Problem.” [Online]. Available: www.ijert.org

[11] A. Q. Ansari, Ibraheem, and S. Katiyar, “Comparison and analysis of solving travelling salesman problem using GA, ACO and hybrid of ACO with GA and CS,” in 2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions, WCI 2015, Institute of Electrical and Electronics Engineers Inc., Jun. 2016. doi: 10.1109/WCI.2015.7495512.

[12] Haider A Abdulkarim, Ibrahim F Alshammari “Comparison of Algorithms for Solving Traveling Salesman Problem”, 2015

[13] D. Nuraiman, F. Ilahi, Y. Dewi and E. A. Z. Hamidi, "A New Hybrid Method Based on Nearest Neighbor Algorithm and 2-Opt Algorithm for Traveling Salesman Problem," 2018 4th International Conference on Wireless and Telematics (ICWT), Nusa Dua, Bali, Indonesia, 2018, pp. 1-4, doi: 10.1109/ICWT.2018.8527878.

[14] Appligate, D.L., Bixby, R.E., Chavatal, V., Cook, W.J.: The Travelling Salesman Problem, A Computational Study. Princeton Univesity Press, Princeton, 2006

[15] Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer, Germany, 1994

[16] Arora, K., Agarwal, S., & Tanwar, R. (2016). Solving TSP using Genetic Algorithm and Nearest Neighbour Algorithm and their Comparison. Paper presented at 2016 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS).