IB Computer Science

Extended Essay

—---

Title: Comparison of different heuristic algorithms for solving the travelling salesman problem

Research Question: How does ant colony optimization compare to genetic algorithms in terms of time complexity and the shortest solution found for solving the travelling salesman problem?

—---

Student Code: Jwq262

Exam Session: November 2022

Word Count: 3037

Table of Contents

1 Introduction

Travelling salesman problem (TSP) is a well-known optimization problem in computer science related to - when given a number of cities and the distances between them - finding the shortest single path that goes through each city and returns to starting position. Its applications include logistics, planning and scheduling. The two important factors when measuring the effectiveness of solutions to the travelling salesman problem are the distance and time. The distance matters as a shorter distance travelled in a real world context would result in less fuel consumption and less time travelled, thus resulting in less cost. However, the time necessary to find the solution is also essential as, especially in problems with a large number of points to travel to, a very large amount of time necessary to find the solution would make it redundant.

Various methods for solving the travelling salesman problem have been created. These methods can be divided into two groups as heuristic and linear algorithms. The travelling salesman problem can be expressed as an integer linear programme and can be solved using linear programming, for instance using the brute force approach. However, as the number of cities and the complexity increases, linear algorithms aren't able to solve the travelling salesman problem in a reasonable amount of time. Thus, new methods called heuristic algorithms have been created by sacrificing precision, accuracy, and completeness for speed.

Genetic algorithms (GA) and ant colony optimization (ACO) are two types of heuristic algorithms used to solve the travelling salesman problem that are inspired from nature and biology. Thus, this investigation seeks to answer the research question "How does ant colony optimization compare to genetic algorithms in terms of time complexity and the shortest solution found for solving the travelling salesman problem?".

2 Background Information

2.1 Travelling Salesman Problem

The travelling salesman problem is an example of a combinatorial optimization problem. It is classified as NP-hard as no quick solution exists and increasing the number of cities exponentially increases the complexity of the problem. While TSP can be described as finding the shortest path to visit a number of locations with set distances between each other and returning to the same location, it can also be expressed as an undirected weighted graph and an integer linear program.

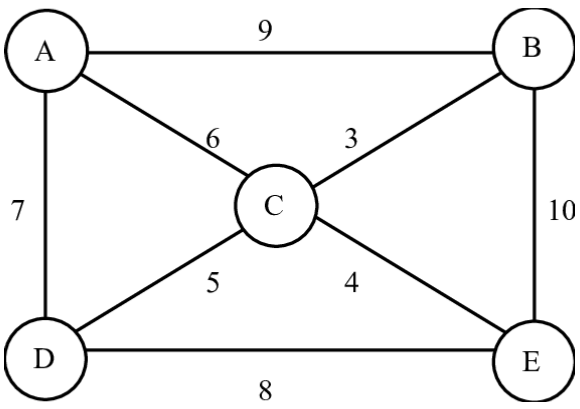


Fig.1 Graph Representation *(Research Gate)*

In the undirected weighted graph representation, the cities are shown as the vertices of the graph while the paths are shown as the edges of the graph. The cost of one path could be calculated by summing the distances between the cities visited in order. For instance, according to the graph shown in figure 1, the path A-D-C-E-B would result in travelling a distance of 26 (7 + 5 + 4 + 10) units.

Another way of representing the travelling salesman problem is integer linear programs. While multiple formulations of TSP exist, the two most notable ones are the Miller-Tucker-Zemlin (MTZ) formulation and the Dantzig-Fulkerson-Johnson (DFJ) formulation.

<u>The MTZ formulation</u> (Baobab)

Primarily certain variables need to be determined:

I: *set of nodes*

u_i : *a dummy variable*

$x_{ij} = \{1\text{: } \textit{the path goes from node i to node j},\ 0\text{: } \textit{otherwise}\}$

$c_{ij} > 0$: *the distance between node i and node j*

Using these variables the formula could be constructed as:

$$min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$$

With the constraints:

$u_i \in Z$

$x_{ij} \in \{0, 1\}$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \quad \forall j$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \quad \forall i$$

$u_i - u_j + nx_{ij} \leq n - 1 \quad 2 \leq i \neq j \leq n$

$0 \leq u_i \leq n - 1 \qquad 2 \leq i \leq n$

The DFJ formulation (Baobab)

The DFJ formulation uses the same variables and formula as the MTZ formulation. However, it is constructed with different constraints:

$$x_{ij} \in \{0,\ 1\} \quad i,\ j\ =\ 1,...,n;$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \quad j\ =\ 1,...,n;$$

$$\sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \quad i\ =\ 1,...,n;$$

$$\sum_{i\in Q}^{n} \sum_{j\neq i,\, j\in Q}^{n} x_{ij} \leq |Q| - 1 \quad \forall Q \subseteq \{1,...,n\},\ |Q| \geq 2$$

The first three constraints function the same as the ones in the MTZ formulation. The last constraint, however, makes sure that there are no sub-tours. Thus, the returned solution is formed as a single tour rather than a group of smaller tours. This leads to an exponential number of constraints, so it is solved via column generation algorithms.

2.2 Traditional and Heuristic Algorithms

There are two ways of solving travelling salesman problems. The traditional approach consists of trying all permutations and checking which one is the shortest using brute force search. While this approach always gives a perfect solution, it runs on the factorial time complexity of O(n!). Thus, even a travelling salesman problem with 15 cities needs all 15! = 1.3076744e+12 possibilities to be checked which makes the traditional way impractical.

Due to the inefficiency of the traditional methods, heuristic algorithms have been created. While the heuristic methods do not output a perfect solution unlike the traditional algorithms, they make solving the travelling salesman practically possible by greatly reducing the execution time in cases where there are very large amounts of data. Examples of heuristic algorithms that are inspired from nature include genetic algorithms and ant colony optimization. While many heuristic algorithms could be used to solve the travelling salesman problem, the choice of which one to use depends on many factors such as the ease of implementation, accuracy, and speed of execution. The importance of each factor depends on the situation and the purpose.

2.3 Genetic Algorithms

Genetic algorithms are search and optimization algorithms constructed based on the concept of natural selection where the fittest organisms survive and reproduce. The characteristics of organisms (the expressed phenotypes) are determined by certain genes inside the chromosomes of eukaryotic organisms. GAs aim to replicate this natural process on the computer level in order to solve the travelling salesman problem.

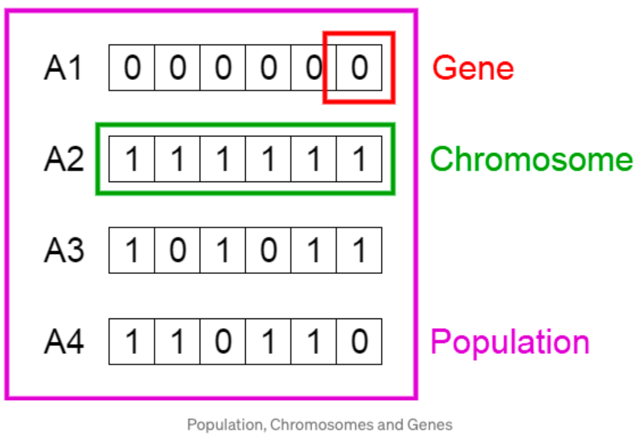


Fig.2 Population in GAs *(Research Gate)*

There are certain steps necessary to achieve this. Primarily an initial P(0) population needs to be created which is usually done randomly in order to make the diversity of the initial population high. The diversity of the population needs to be kept high throughout the generations in order to prevent premature convergence. Premature convergence is a case where the algorithm reaches a solution before the global optimum solution is reached. Low diversity in the selection pool causes the algorithm to be trapped in local extrema rather than the global extremum due to the lack of diversity causing successive generations to not be able to search for new avenues of improvement. After the initial population is created, certain predetermined selection operators are applied to each solution in the population. Then a mating pool is created from the ones that the selection operators selected according to their fitness levels which is determined by a fitness function. These solutions are called parents. These parents undergo the crossover and mutation operations in order to create a secondary population consisting of children solutions derived from the parents. Then, the steps after the creation of the initial population are repeated until a satisfactory solution is reached.

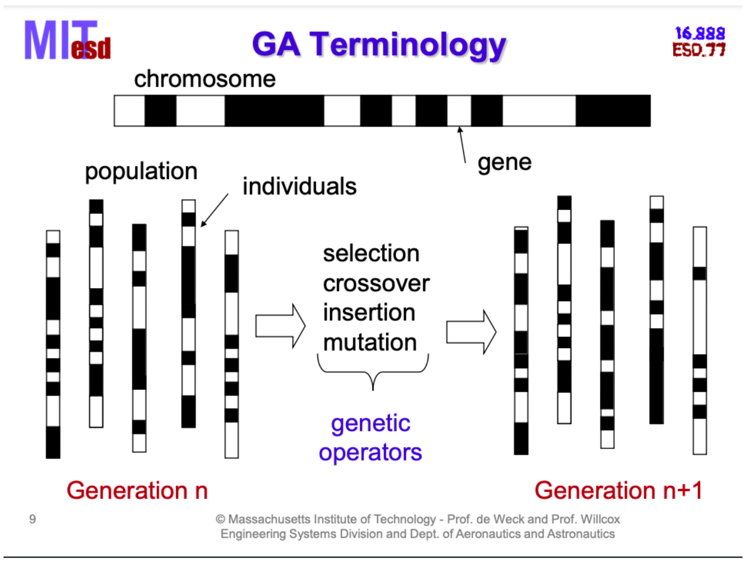
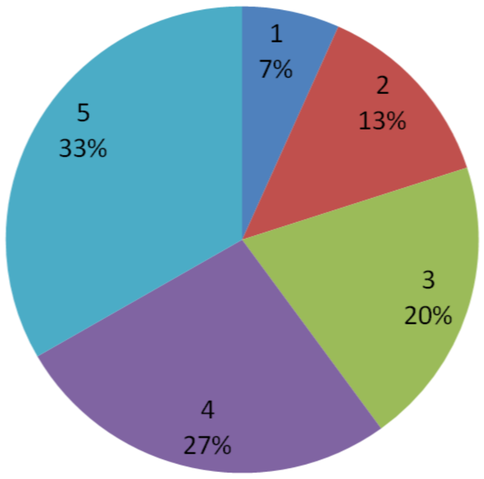


Fig.3 Creation of Successive Generations *(Baobab)*

Different techniques for selection, crossover, and mutation can be used to modify the genetic algorithm. Methods of mating pool selection are divided into three groups: fitness based selection, ordinal based selection, and threshold based selection. In fitness based selection models the selection is done randomly, but the solutions with higher fitness levels have a higher chance of being selected. Roulette wheel selection and stochastic universal sampling are examples of fitness based selection. Roulette wheel selection can be modelled as a wheel where the sizes of the slices are determined according to the fitness of the solutions.



Individual	Fitness
1	1.0
2	2.0
3	3.0
4	4.0
5	5.0

Fig.4 Roulette Wheel Selection *(Research Gate)*

Ordinal based selection includes tournament selection and ranking selection. In tournament selection solutions are selected randomly and then compared according to their fitness levels, the ones with the higher fitness levels called "winners" are added to the mating pool.

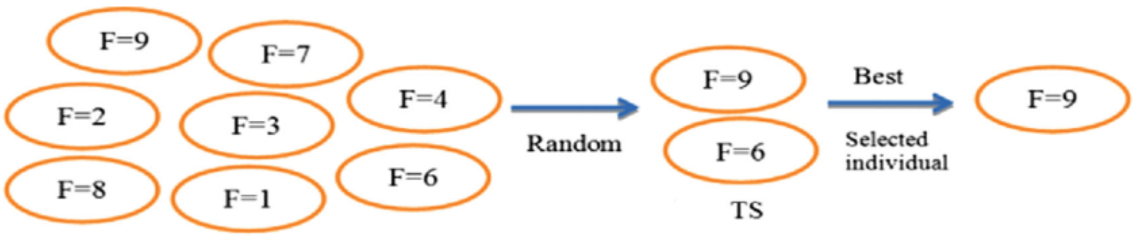


Fig.5 Tournament Selection *(Research Gate)*

In ranking selection, on the other hand, the solutions are sorted according to fitness levels and the highest ranked ones are added to the mating pool. Finally, threshold based selection consists of truncation selection where solutions are sorted according to a special trait and the ones above a certain threshold are selected as parents.

There are many types of crossovers that could be applied to cross chromosomes in genetic algorithms. One notable one is ordered crossover (OX) where a subsequence of genes is chosen from the chromosome of each parent from the same loci and passed down to the offspring in a way that the subsequence derived from parent 1 goes to the offspring 2 and vice versa. The rest of the chromosomes of the offspring are filled from the corresponding parent (parent 1 to offspring 1) such that no gene is repeated, as displayed in figure 6 below.

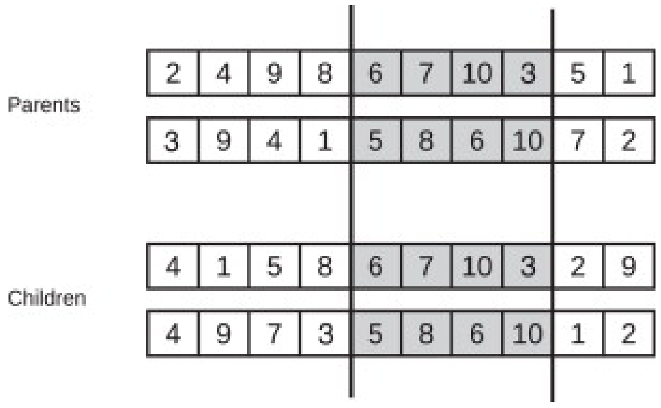


Fig.6 Ordered Crossover (*Alexander, A., & Sriwindono)*

In the concept of natural selection and evolution, organisms have a chance to experience mutations although the chances are very low. Mutations change one or more gene values of a chromosome which changes the properties of the solution. Unlike crossover, mutation is essential for genetic algorithms as it gives birth to new solutions and prevents premature convergence. If the rate of mutation is too low the execution time of the genetic algorithm greatly increases while if the rate of mutation is too high the genetic algorithm gets reduced to a random search. There are multiple types of mutations in genetic algorithms:

- **Bit Flip Mutation**: One or more random bits on a chromosome are selected and then flipped

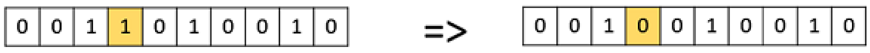


Fig.7 Bit Flip Mutation

- **Swap Mutation**: Two random positions on a chromosome are selected and then swapped

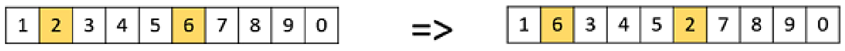


Fig.8 Swap Mutation

- **Scramble Mutation**: A subset of positions are selected on a chromosome and then they are shuffled randomly

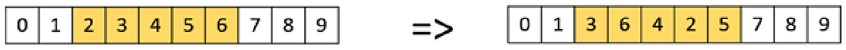


Fig.9 Scramble Mutation

- **Inversion Mutation**: A subset of positions are selected on a chromosome and then they are inverted

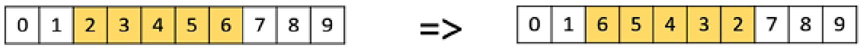


Fig.10 Inversion Mutation

2.4 Ant Colony Optimization

Understanding how ants behave in nature is crucial for understanding how ant colony optimization algorithms work. Ants constantly roam around the environment around their nests for many reasons such as exploration, looking for materials, and searching for food. As the ants wander they leave a trail consisting of a substance called pheromone behind them in order to be able to travel back to the nest and to communicate with other ants. Other ants may follow these pheromones in order to go to the food source that an ant has found. However, not every ant will follow every trail and the ants may diverge from the pheromone path after following it for a while. Pheromones weaken as time passes and completely evaporate after a while. Ants are more likely to follow newer and stronger pheromones rather than older and weaker ones. If an ant diverges from a pheromone path going to a food source and finds another path to the food source, it will grab a piece of the food and come back. If the new path is shorter, the ant will return to the nest quicker than the others, so its pheromone trail will be newer and stronger - causing other ants to follow it. Thus, after a certain amount of time, an optimal path will be found.

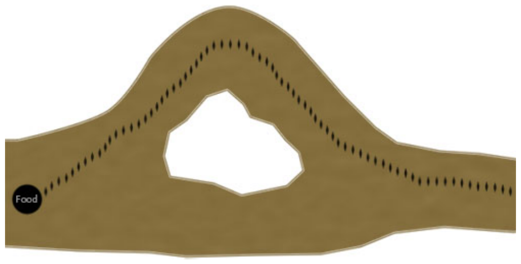


Fig.11 Ant Pathfinding 1*(The Project Spot)*

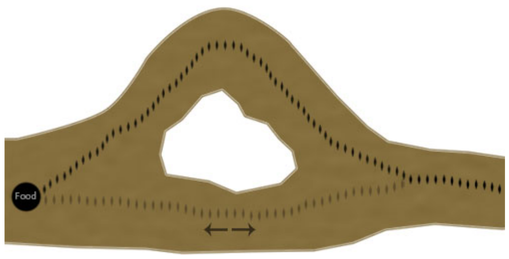


Fig.12 Ant Pathfinding 2 *(The Project Spot)*

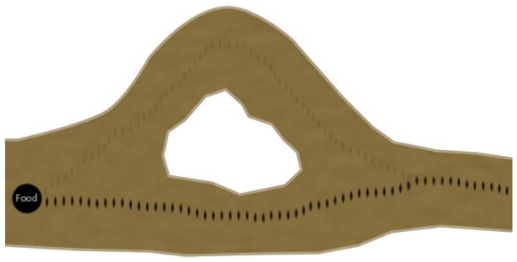


Fig.13 Ant Pathfinding 3*(The Project Spot)*

As an algorithm is created according to the aforementioned ruleset, a value needs to be determined for the chance that an ant diverges from a pheromone trail. This variable can be compared to the mutation rate in genetic algorithms. If it is too low the execution time of the algorithm greatly increases while if it is too high the algorithm turns into a random search. The probability that a single component of the solution (path) gets selected can be expressed mathematically as:

$$p_{ij} = \frac{(r_{ij})^{\alpha} (s_{ij})^{\beta}}{\sum_{k \in neighbour(i)} (r_{ik})^{\alpha} (s_{ik})^{\beta}}$$

Where r_{ij} denotes the pheromone concentration on the edge ij, s_{ij} denotes the heuristic value, α and β are the parameters that determine the significance of the pheromone concentration and heuristic value when selecting the next component of the solution.

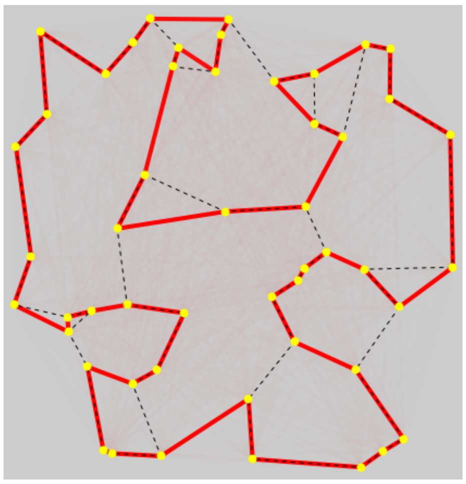


Fig.14 ACO Example *(The Project Spot)*

In nature pheromones are stronger on the better paths due to ants being able to traverse them quicker as explained earlier. This can be replicated on ant colony optimization algorithms by varying the pheromone amount deposited according to how quick the path is so that shorter paths have a higher pheromone concentration.

3 Experiment

3.1 Methodology

To test the difference in efficiency between genetic algorithms and ant colony optimization algorithms, primarily a program that creates n amount of locations inside certain boundaries is coded. Then a genetic algorithm and an ant colony optimization algorithm that iterates through the code 100 times, and outputs the distance of the best solution and the execution time is developed .

For the genetic algorithm: tournament selection is used as the selection method for the mating pool, ordered crossover is used as the crossover method and bit flip mutation is used as the type of mutation applied to the chromosomes. The mutation rate is determined as 1 in 1000. For the ant colony optimization algorithm, on the other hand, the α value is set to 1 (pheromone concentration), the β value is set as 0.5 (heuristic value) and the pheromone evaporation rate is set to 0.9. The code is then run through 10 data points consisting of the city amounts of 10, 20, 30, 40, 50, 60,70, 80, 90 and 100.

3.2 Variables

Independent Variables	**Dependent Variables**	**Control Variables**
. Type of Heuristic Algorithm (GA or ACO) . City Amount	. Shortest Path Found . Execution Time	. Selection Method . Crossover Method . Mutation Type . Mutation Rate . α Value . β Value . Pheromone Evaporation Rate

3.3 Procedure

1. Generate 10 cities in random locations
2. Run the genetic algorithm and record the distance of the shortest path found and the execution time
3. Run the ant colony optimization algorithm and record the distance of the shortest path found and the execution time
4. Repeat the steps 1 to 3 with 10 more cities until the maximum city number of 100 is reached

4.1 Raw Data

The tables below show the exact values of all of the measurements.

Distance of the Shortest Path	10 Cities	20 Cities	30 Cities	40 Cities	50 Cities	60 Cities	70 Cities	80 Cities	90 Cities	100 Cities
GA	1297	2783	4492	6679	7967	9552	10199	11730	14089	17992
ACO	1124	1567	2033	2511	2958	3586	3975	3765	4506	4987

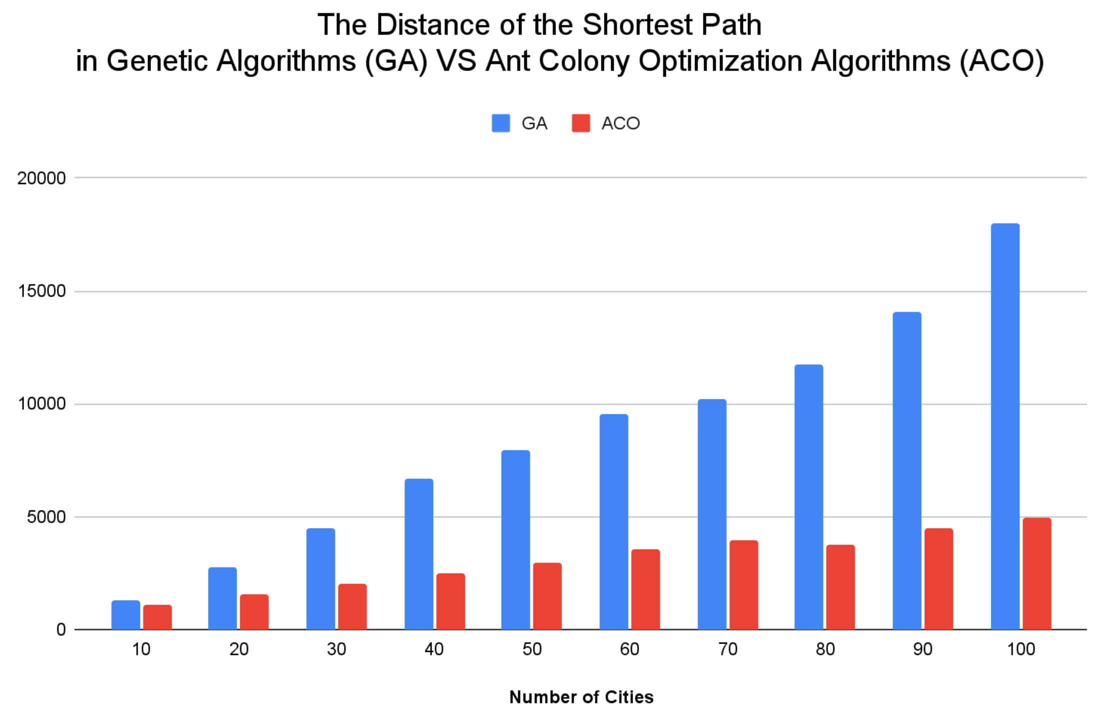
Table 1

Execution Time (ms)	10 Cities	20 Cities	30 Cities	40 Cities	50 Cities	60 Cities	70 Cities	80 Cities	90 Cities	100 Cities
GA	2.38	5.31	11.56	24.68	53.74	112.89	230.4	781.31	1276.88	2045.26
ACO	8.16	13.92	15.75	17.12	21.05	25.9	30.68	43.29	67.32	82.09

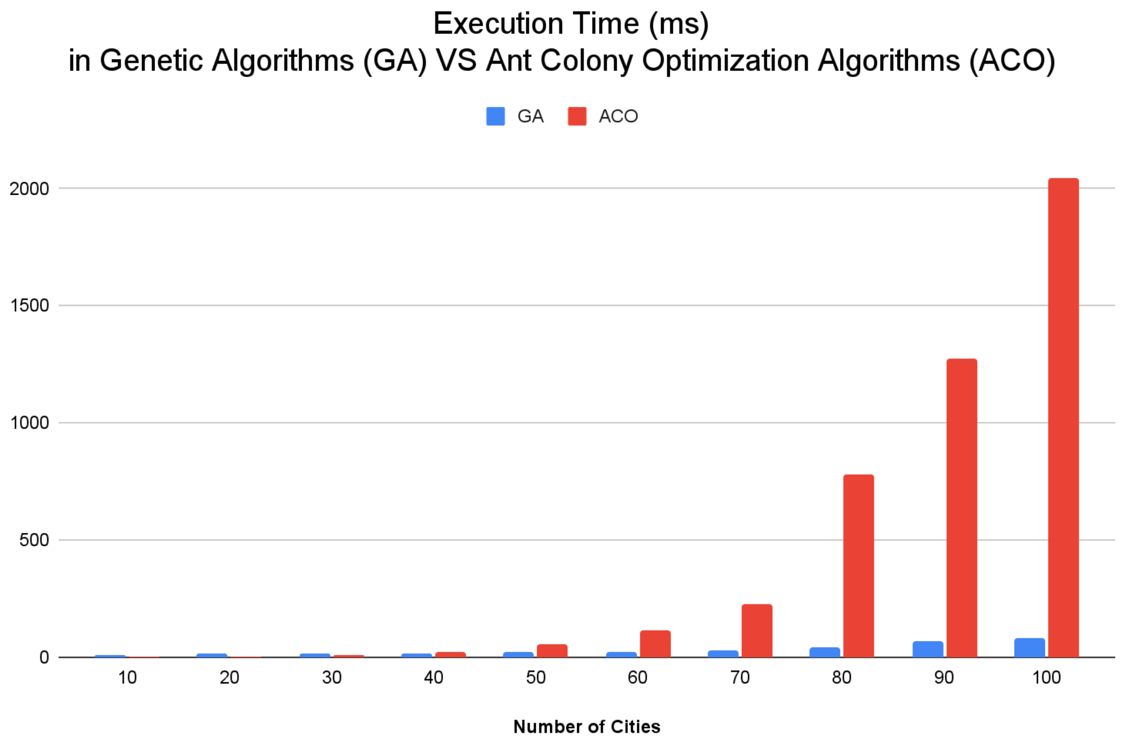
Table 2

4.2 Graphs

In order to better understand the trends, the data has been represented as column charts.



Graph 1



Graph 2

5 Data Analysis

Overall, the values for both the distance of the shortest path and execution time are similar in lower city numbers and the differences increase exponentially as the number of cities increases. As it can be observed from graph 1, at all data points ant colony optimization algorithm has been more effective in finding the shortest possible path than the genetic algorithm. However, as shown on graph 2, the execution time of the genetic algorithm has remained much less than the ant colony optimization algorithm, especially at higher city numbers.

6 Limitations and Further Improvements

The limitations of the experiment lie within how the genetic algorithm and the ant colony optimization algorithm have been constructed. Using a different selection method, crossover method, mutation type, mutation rate for the genetic algorithm and using a different α value, β value and pheromone evaporation rate for the ant colony algorithm might have changed the results significantly. As it isn't feasible to try all different types of methods and values for variables, the results of the experiment documented in this paper are true for the structures of the GA and ACO and their characteristics as explained in the methodology section. However, the results may differ if the algorithms are constructed differently.

In addition, this research could be expanded by comparing other heuristic algorithms as well. There are many heuristic algorithms that can be used to solve the travelling salesman problem such as particle swarm optimization and simulated annealing which can be more effective than both the GAs and AOCs in certain situations.

7 Conclusion

In this paper the travelling salesman problem and two heuristic algorithms used to solve it were investigated. The mathematics and logic behind the travelling salesman problem, genetic algorithms, and ant colony optimization algorithms were explained. Finally, genetic algorithms and ant colony optimization algorithms were compared according to their accuracy and time efficiency at multiple city amounts for solving the travelling salesman problem.

In conclusion, according to the experiment’s results, the ant colony optimization method is more precise and accurate than the genetic algorithm method in solving the travelling salesman problem, but the genetic algorithm is much more time efficient, especially at higher data amounts.

At low city amounts, the ease of application is the most important factor when choosing between the two heuristic algorithms as their results and execution times are very similar. However, in situations where large amounts of data exist the importance of precision versus time efficiency should be determined and the algorithm should be chosen accordingly.

8 Works Cited

Alexander, A., & Sriwindono, H. (2020, March 3). The comparison of genetic algorithm and ant colony optimization in completing travelling salesman problem. EUDL. Retrieved August 22, 2022, from https://eudl.eu/doi/10.4108/eai.20-9-2019.2292121

Genetic algorithm versus ant colony optimization algorithm comparison ... (n.d.). Retrieved August 22, 2022, from http://vigir.missouri.edu/~gdesouza/Research/Conference_CDs/IFAC_ICINCO_2010/ICINCO/ICINCO/Intelligent%20Control%20Systems%20and%20Optimization/Short%20Papers/ICINCO_2010_100_CR.pdf

Relationship between genetic algorithms and ant colony optimization ... (n.d.). Retrieved August 22, 2022, from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.323.8685&rep=rep1&type=pdf

Miller-Tucker-Zemlin formulation. To. (n.d.). Retrieved August 22, 2022, from https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html#:~:text=The%20Miller%2DTucker%2DZemlin%20

Explicit Dantzig-Fulkerson-Johnson formulation. To. (n.d.). Retrieved August 22, 2022, from https://how-to.aimms.com/Articles/332/332-Explicit-Dantzig-Fulkerson-Johnson-formulation.html

Simple example of TSP with 1 depot and 1 robot. (n.d.). Retrieved August 22, 2022, from https://www.researchgate.net/figure/Simple-example-of-TSP-with-1-depot-and-1-robot_fig1_338046714

GeeksforGeeks. (2022, July 13). *Travelling salesman problem: Set 1 (naive and dynamic programming)*. GeeksforGeeks. Retrieved August 22, 2022, from https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

Classical and heuristic algorithms used in solving the ... - dergipark. (n.d.). Retrieved August 22, 2022, from https://dergipark.org.tr/en/download/article-file/763390

Roulette wheel selection example | download scientific diagram. (n.d.). Retrieved August 22, 2022, from https://www.researchgate.net/figure/Roulette-wheel-selection-example_fig2_251238305

Binary tournament selection | download scientific diagram. (n.d.). Retrieved August 22, 2022, from https://www.researchgate.net/figure/Binary-tournament-selection_fig3_257336491

International Journal of combinatorial optimization problems ... - redalyc. (n.d.). Retrieved August 22, 2022, from https://www.redalyc.org/pdf/2652/265219618002.pdf

Wikimedia Foundation. (2022, August 18). *Travelling salesman problem*. Wikipedia. Retrieved August 22, 2022, from https://en.wikipedia.org/wiki/Travelling_salesman_problem

Ant colony optimization for Hackers. The Project Spot. (n.d.). Retrieved August 22, 2022, from https://www.theprojectspot.com/tutorial-post/ant-colony-optimization-for-hackers/10

Baobab. (2021, August 9). *Three different methods to solve the travelling salesman problem*. baobab soluciones. Retrieved August 22, 2022, from https://baobabsoluciones.es/en/blog/2020/10/01/travelling-salesman-problem-methods/