Self-Organizing Systems - assignement 1

Evelina Danielsson (12108362) Helena Hrženjak (12110167) Maximilian Rieger (1425229)

November 27, 2021

1 Introduction

In this assignment the Maximum Scatter TSP, Job-shop scheduling, and Rastring function have been selected as problems to optimize by using Genetic Algorithms (GA) and Ant Colony Optimizing (ACO) algorithm.

The genetic algorithm is a heuristic algorithm that searches for the optimal solution. The algorithm is inspired by Darwin's theory of natural evolution and uses the theory of natural selection in order to find the optimum. The fittest individuals are selected and produce new offspring. Which such things as crossing-over and spontaneous mutations the generations are evolving.

The ant colony optimizing algorithm is inspired by ants and how they find the nearest path to food. When they return to the colony they lay down pheromone trails. An ant is mor likely to follow a path with higher amounts of pheromone than one with lower amounts. Over time the pheromone starts to evaporate, so the longer distance the ant needs to travel on the path, the more time the pheromone has to evaporate, thus a short path will have more pheromones than other paths as the pheromones on that is released more frequently. When an ant finds a short path to the food, other ants are more likely to follow, and positive feedback single out the best part in the end.

In the following sections the Maximum Scatter TSP, Job-shop scheduling, and Rastring function will be described in more detail together with the result of using the two algorithms.

2 Rastrigin function

The Rastrigin function is a non-convex highly multi-modal non-linear function.
It is used in practice as a performance test problem for optimization algorithms.
It is defined as:

$$f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$$

$$-5.12 \leq x_i \leq 5.12$$

$$\text{minimum at } f(0, \cdots, 0) = 0$$

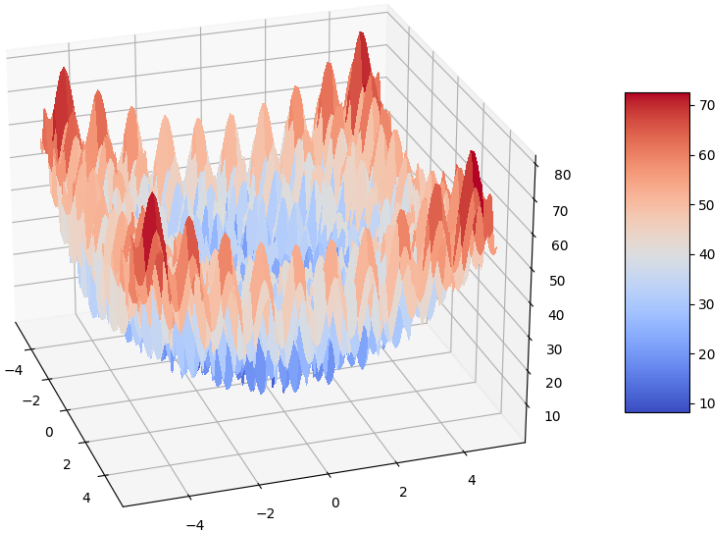


Figure 1: Surface Plot of the rastrigin function

2.1 Evaluation with genetic algorithm

For the evaluation of the GA approach the algorithm was run in a grid search with the following parameter options:

- Number of genes in [2, 10]
- Population size in [10, 100, 1000]:
- Max iterations in [100, 500]:
- Selection strategies ["tournament", "roulette wheel"]:
- Mutation rate in [0.1, 0.5]:

Fitness was defined as the negative rastriging function of the genes as the implementation used could only maximise the function.
The number of genes determines the dimensionality of the rastrigin function used.

The run time for 2 genes is between 0.0625 - 17.4531 and for 10 genes between 0.1718 - 26.7031
The best solution found for 2 genes was: $\begin{bmatrix}-3.03396624e-10 & -8.52448801e-11\end{bmatrix}$ with a fitness of 0.

The best solution found for 10 genes was:

$$\begin{bmatrix}-0.00884496 & -0.01057834 & -0.0042656 & 0.00439743 & -0.0143721 \\ 0.03168057 & -0.01486237 & -0.00646266 & 0.00337322 & 0.01277119\end{bmatrix}$$ with a fitness of -0.3712.

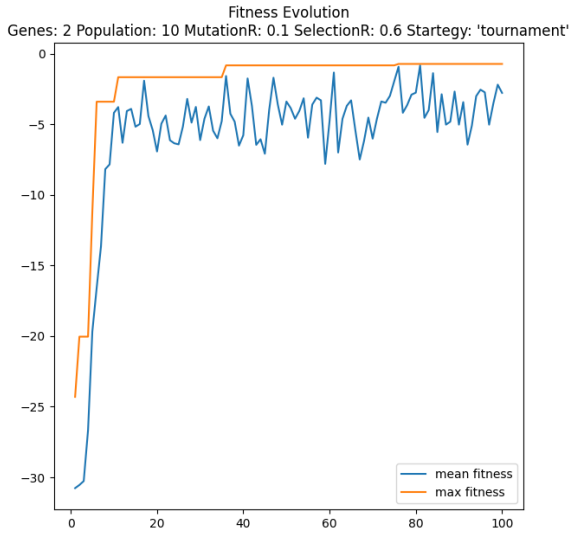


Figure 2: Fitness evolution for GA run with
Genes: 2 — Population: 10 — MutationRate: 0.1

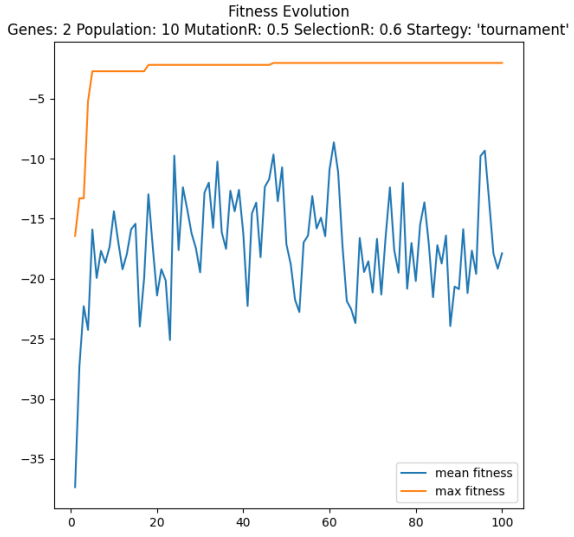


Figure 3: Fitness evolution for GA run with
Genes: 2 — Population: 10 — MutationRate: 0.5

When the mutation rate is chosen to high the mean fitness of the population stays much lower than the best found solution, as can be seen in figure 3.

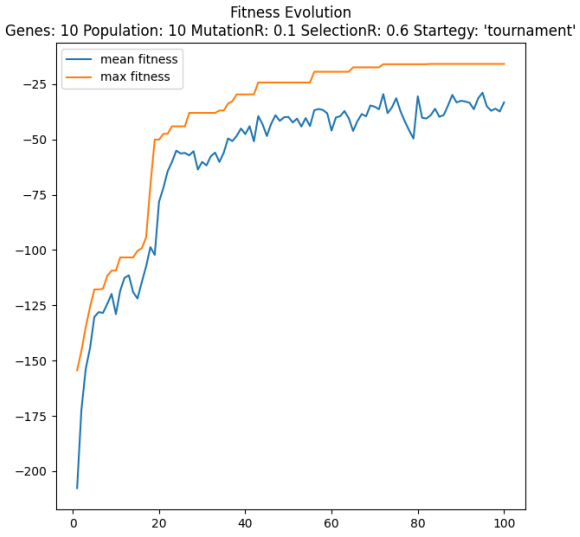


Figure 4: Fitness evolution for GA run with
Genes: 10 — Population: 10 — MutationRate: 0.1

2.2 Evaluation with ant colony optimization

For the evaluation of the ACO approach the algorithm was run in a grid search with the following parameter options:

- Number of variables in [2, 10]
- Population size in [10, 100, 1000]:
- Max iterations in [100, 500]:
- Probability density [0.3, 0.5, 0.8]:
- Evaporation rate [0.3, 0.5, 0.65, 0.8]:

The number of variables determines the dimensionality of the rastrigin function used.
The run time for 2 variables is between 0.21875 - 35.426 and for 10 variables 0.348 - 45.8924

The best solution found for 2 variables was: $\begin{bmatrix}0.01375711 & 0.00108365\end{bmatrix}$ with a fitness of 0.0377.

The best solution found for 10 variables was:

$$\begin{bmatrix}1. & 1. & 0.14559747 & 0.98079985 & 0.89135984 \\ 1. & 0.08977809 & 0.95161219 & 0.11656103 & 0.9153156\end{bmatrix}$$ with a fitness of 18.7112.

2.3 What algorithm is more suitable for the problem?

In the case of the rastriging function it is the genetic algorithm as it finds better solutions and has a shorter run time overall. This was expected though as the ACO algorithm has to be adapted for the continuous domain and the GA is directly applicable. Both in the 2 and 10 variable case GA finds the better solution in a shorter time.

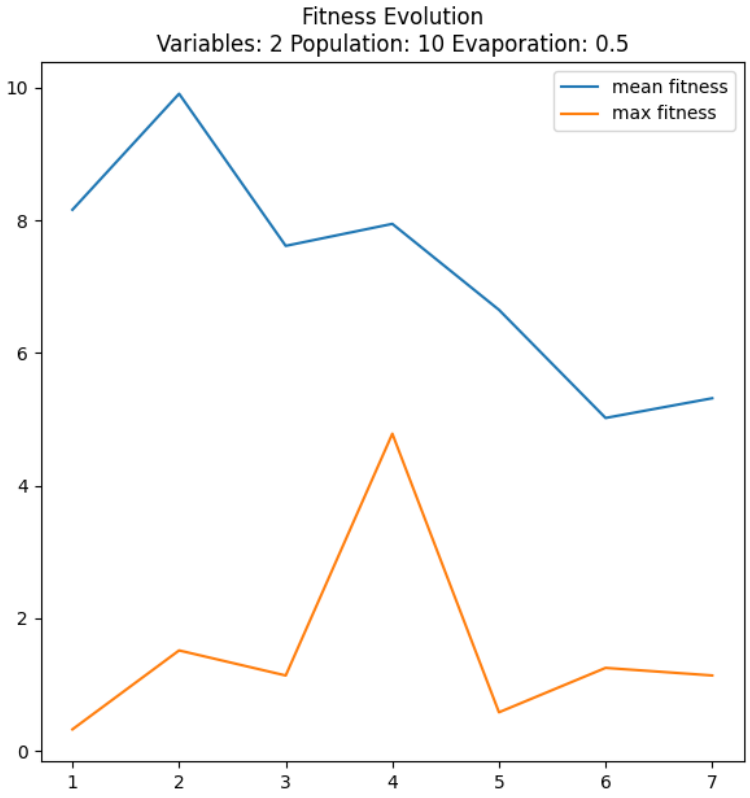


Figure 5: Fitness evolution for GA run with
Variables: 2 — Population: 10 — EvaporationRate: 0.5

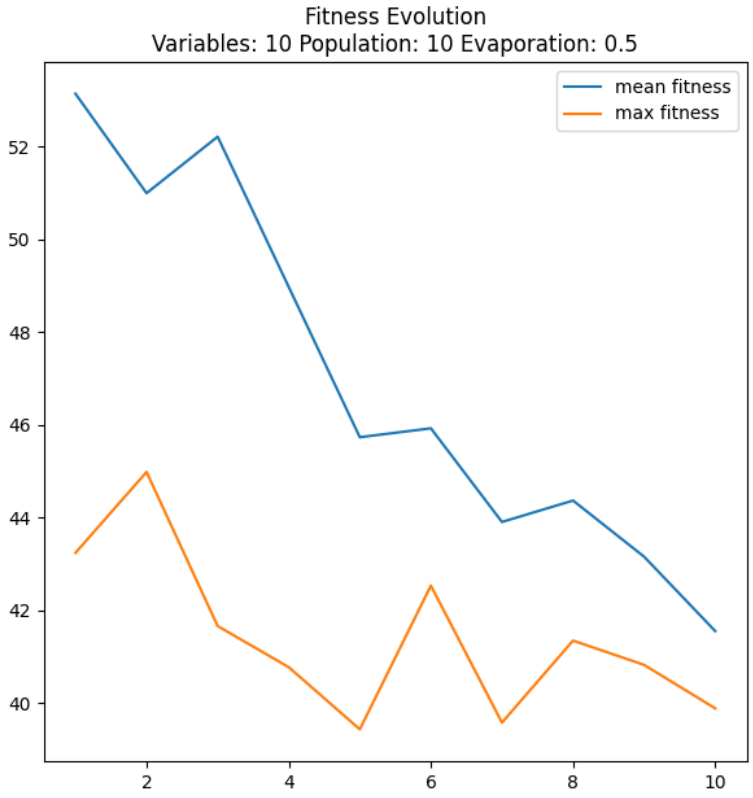


Figure 6: Fitness evolution for GA run with
Variables: 10 — Population: 10 — EvaporationRate: 0.5

3 Maximum Scatter Traveling Salesman

The maximum scatter TSP is a problem where we want to maximize the minimum edge in the graph. This algorithm can be applied to real world problems, for example riveting where you aim to place rivets far from each other, or drilling where the workpiece heats up and you want to drill the next hole far from the one you just drilled.

The problem can be optimized by the use of genetic algorithms and ant colony optimization algorithms.

In this case we are locking for a maximum distance of an edge. Therefore, the algorithms that are usually used to find the shortest distance, were inverted and now designed to find the longest distance of the shortest edge.

3.1 Evaluation with genetic algorithm

When looking at the genetic algorithm, by comparing figure 7 with figure 8, one can see that by decreasing the mutation rate it takes more iterations (generations) in order to reach a good result. Also as expected the best genome (most optimal solution) is not changing as many times in the 300 iterations when the mutation rate is set to 10 as of when the rate is set to 60.

If we instead compare figure 7 with figure 9 we can see that the beginning of the curve is fluctuating a bit more in fig 9, it even decreases at one point. The difference between the two executions was the weakness threshold, where the genomes not living up to the thresholds will be removed from the population. This means that in the beginning a larger part of the population will be removed in the case with the higher threshold than the lower which will decrease the gene pool. This means that there are not enough genomes left to weigh up for the situation of the crossing-over and mutations of the small number of genomes being left in the population results in a less fit combinations than their parents. To many individuals have been eliminated in the beginning.

If we now take a look at the last figure when it comes to GA, figure 10 we can see the result of having a too homogenic population and the importance of mutations. Between somewhere before the 200th iteration and after the 800th iteration the result doesn't improve. It might seem that we have reached an optimum, but we have not. In fact, there is now way of knowing if we have reached the absolute optimal solution in any of the cases.

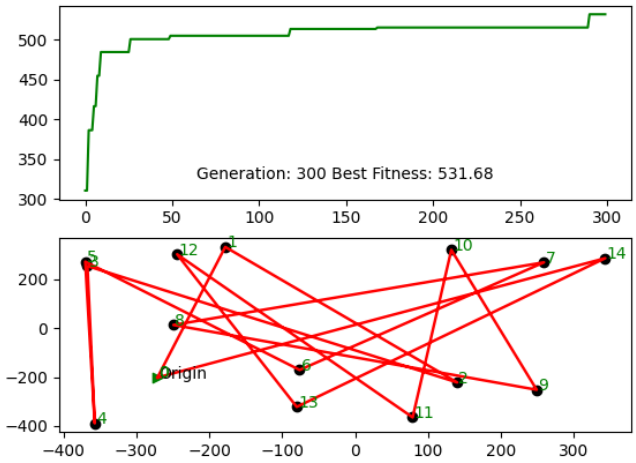


Figure 7: Optimizing the MSTSP problem using GA. Weakness threshold was 300, mutation rate 60 and mutation repeat count 2. The 300 steps took 6.09 seconds.

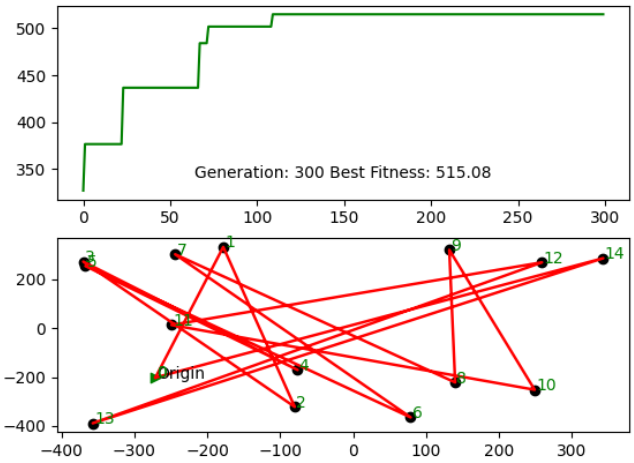


Figure 8: Optimizing the MSTSP problem using GA. Weakness threshold was 300, mutation rate 10 and mutation repeat count 2. The 300 steps took 6.87 secodns.

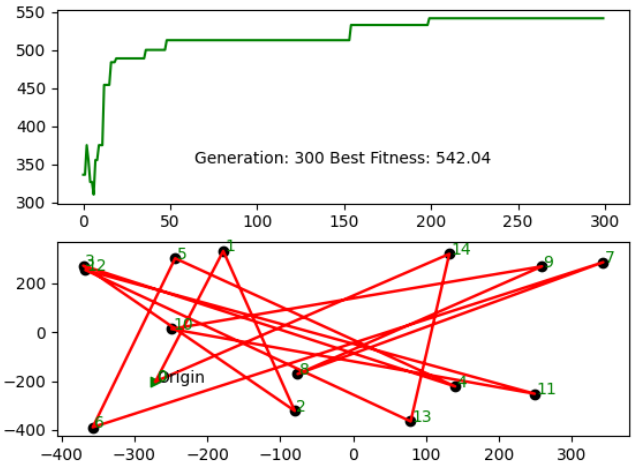


Figure 9: Optimizing the MSTSP problem using GA. Weakness threshold was 450, mutation rate 60 and mutation repeat count 2. The 300 steps took 6.87 seconds.

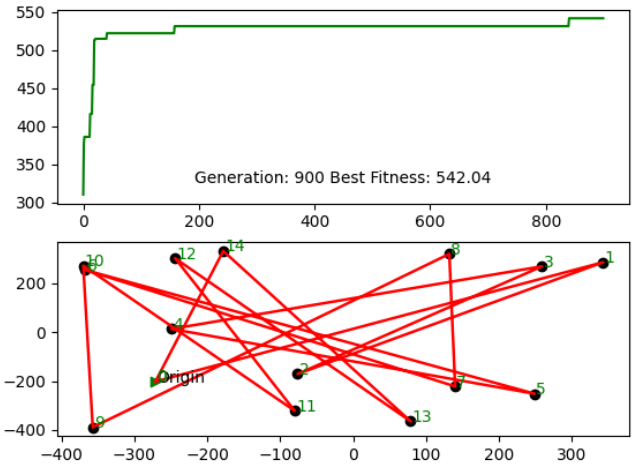


Figure 10: Optimizing the MSTSP problem using GA. Weakness threshold was 300, mutation rate 60 and mutation repeat count 2. The 900 steps took 22.5 seconds.

3.2 Evaluation with ant colony optimization

If we now take a look at the ACO algorithm and the results we got from it we can see that one the parameters that effected the result the most was the size of the ant colony. The figures 11, 12, and 13, shows the results of when the colony is 5, 10, and 25 ants respectively. As the number of ants increases the fluctuation and time to reach a good or the most optimal value found decreases. The best fits for the one with 5 ants was 542.03 and for the runs with 10 and 25 ants the best fits were 553.20.

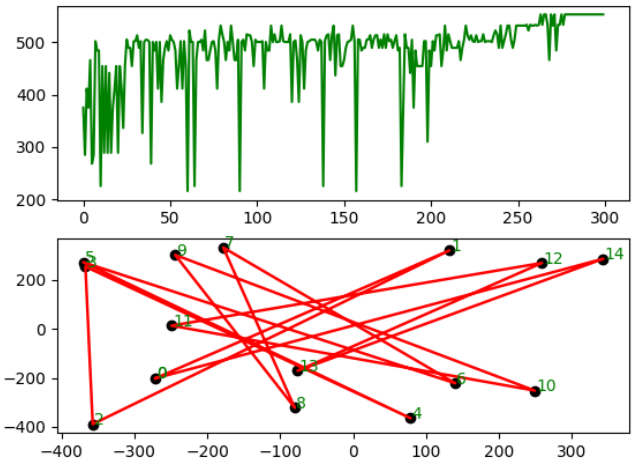


Figure 11: Optimizing the MSTSP problem using ACO. Colony size was set to 5, the 300 steps took 0.993 seconds.

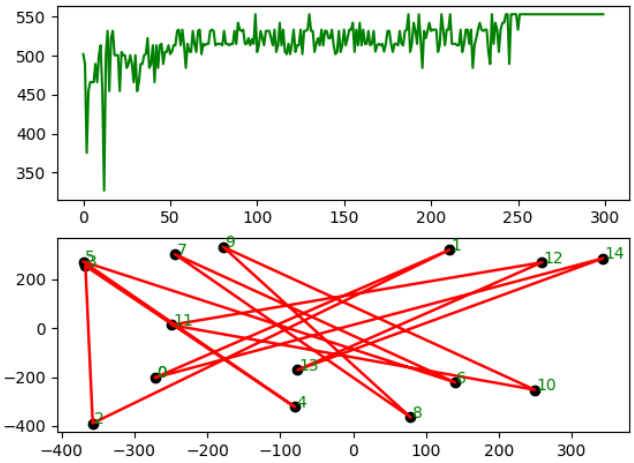


Figure 12: Optimizing the MSTSP problem using ACO. Colony size was set to 10, the 300 steps took 1.43 seconds.

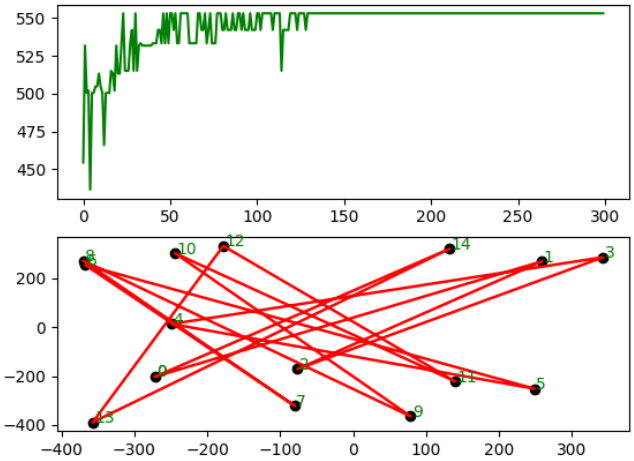


Figure 13: Optimizing the MSTSP problem using ACO. Colony size was set to 25, the 300 steps took 2.44 seconds.

3.3 What algorithm is more suitable for the problem?

In the case of the MSTSP problem we can see that the ACO algorithm is completed in a shorter runtime and is giving a better result. The runtime of the ACO was between 0.933-2.44 seconds depending on the size of the colony (5-25), while the runtime for 300 iterations of GA was about 6.5 second. However, in this implementation of the GA we used a static weakness threshold. If a version where for example the top 20% of the population were to remain and the rest removed, then we would have the same effect of the threshold throughout all iterations.

4 Job-shop scheduling Problem

Job-shop scheduling is a well-known combinatorial optimization problem. In this problem multiple jobs are processed on several machines. Each job consists of tasks that must be performed in a given order. Each task must be processed on a specific machine. (Unlike similar problem - flexible job shop, where each task can be processed on any machine) The problem is to schedule the tasks on the machines with the goal of minimizing the length of the schedule—the time it takes for all the jobs to be completed.

The job shop problem constraints are: a machine can only work on one task at the time, no task can be started until the previous task for that job is completed. Once the task is started it must be done until completion.

In this exercise the task will be solved by using Ant colony optimization and Genetic algorithm. The implementations for algorithms were found on GitHub and adjusted for this exam.

The test data was taken from OR-Library by Dirk C. Mattfeld and Rob J.M. Vaessens. The pair of numbers before each data represents the number of machines and number of jobs. In the data each row represents a machine and time required for each step of the job. The time for each job consists of two columns, first is start time and second is the end time of the job.

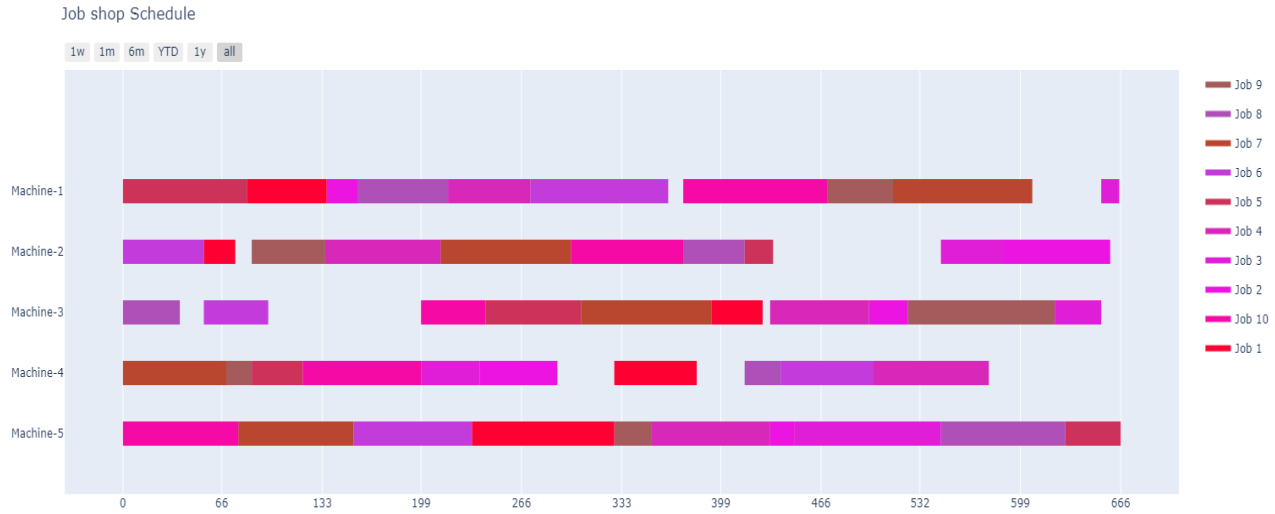


Figure 14: Example of resulting schedule for job shop problem

4.1 Evaluation of Ant colony optimization

Firstly, we make the comparisons for different parameters for ACO and choose optimal parameters. The value we will observe is the makespan, or the time from start of the job project to the end, the aim is that makespan is as short as possible. The parameters were tested on medium sized dataset with 15 machines and 10 jobs (data instance la22). The tested parameters were number of ants, number of cycles and evaporation rate. From the figures 15, 16, 17 we can see that the good choice of parameters would be number of ants = 20, cycle number = 35, pheromone evaporation rate = 0.9.

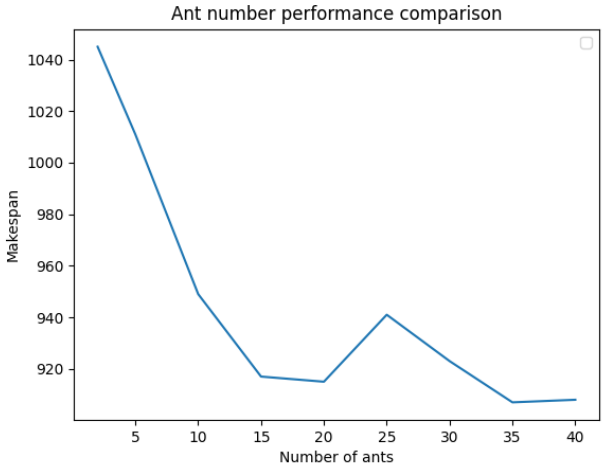


Figure 15: Comparison of results that algorithm gives for a different number of ants

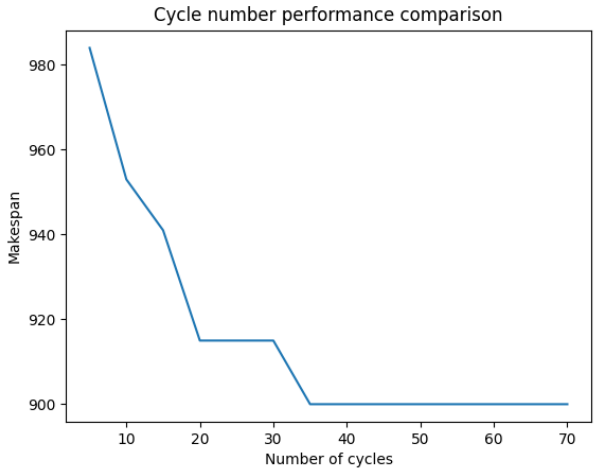


Figure 16: Comparison of results that algorithm gives for a different number of cycles

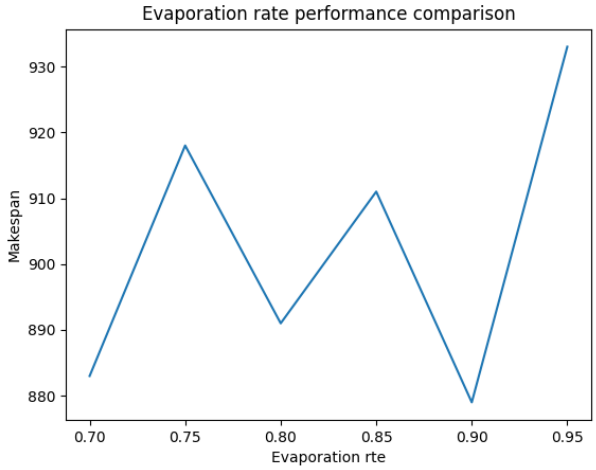


Figure 17: Comparison of results that algorithm gives for a different pheromone evaporation rate

4.2 Evaluation of Genetic algorithm

We repeat process for finding optimal parameters for GA as well. The parameters were tested on the same medium sized dataset with 15 machines and 10 jobs (instance la22). The value we will observe is the makespan, or the time from start of the job project to the end, From the figures 18, **??**, 20 we can see that the good choice of parameters would be population size = 25, mutation rate = 0.2, number of iterations around 3000 is working well.

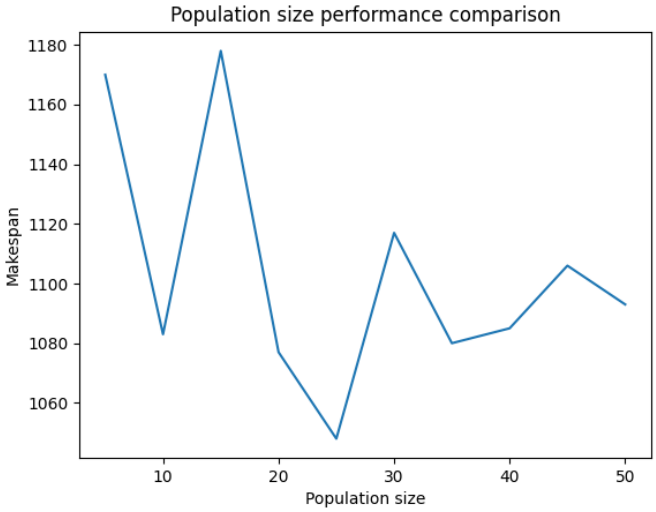


Figure 18: Comparison of results that algorithm gives for a different number of population size

Figure 19: Comparison of results that algorithm gives for a different mutation rate

Figure 20: Comparison of results that algorithm gives for a different number of iterations

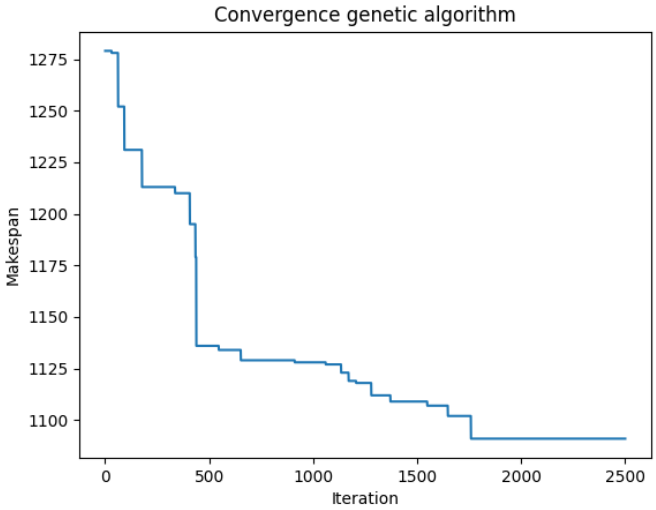
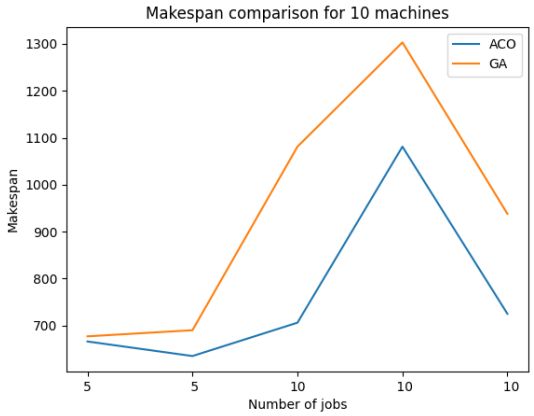
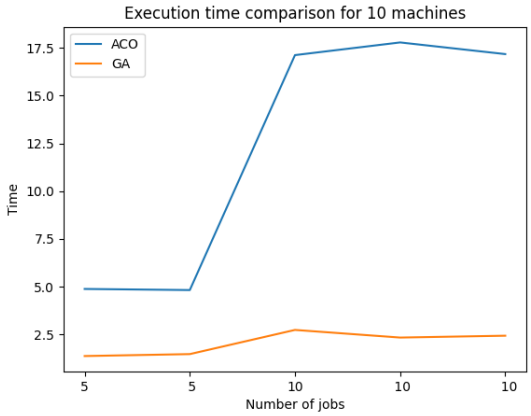


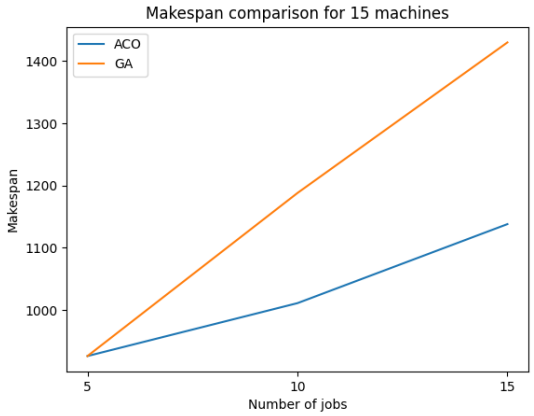
Figure 21: Convergence of genetic algorithm

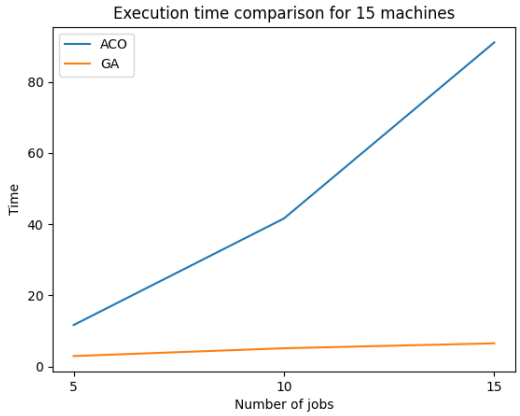
4.3 ACO and GA performance comparison

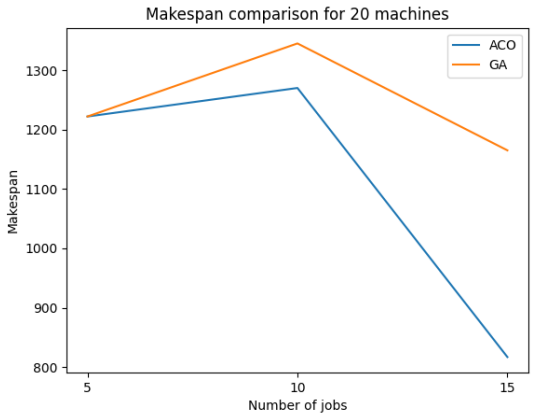
The comparison was done with optimal parameters for each algorithm. The observed properties were makespan, execution time and convergence. Since data can depend both on number of machines and number of tasks per machine, for clearer visualization the plots were split based on number of machines in the data.

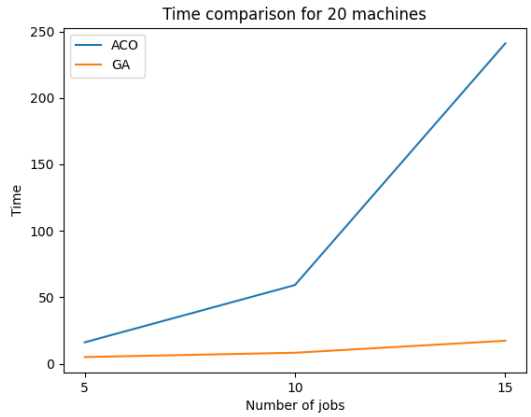












Based on acquired perfomance data we can see that for job-shop scheduling problem the genetic algorithm is noticeably faster than ant colony optimization. The makespan result that ant colony algorithm finds is smaller and therefore better then the result that genetic algorithm gives. If we want to prioritize execution time genetic algorithm will be better for this problem, if we want to get better solution ant colony optimization is better.

5 Comparison

The problems are fairly different as Maximum Scatter TSP is a graph problem, Job-shop scheduling is a constraint optimization problem and the rastrigin function evaluation problem is a continuous non-convex function optimization problem. Despite these differences all these problems can be optimized with the chosen algorithms.
Although the better approach is not always the same as for MSTSP ACO is better and for JSP and rastrigin GA is better.