ECM3412 Continuous Assessment - Ant Colony Optimisation

Talhaa Hussain

2023

1 Introduction

1.1 Background

This report is accompanied by source code in the src/ directory. Please consult README.md for more information on the source code.

This paper looks at the ant colony optimisation algorithm, particularly how results and performance vary when modifying parameters. The parameters in question were as follows:

- Evaporation rate - how fast pheromones evaporate
- Colony size - how many ants active per iteration

This paper will also take discuss ACO approach variations, such as elitist ant system, and local heuristic functions.

The main metric when analysing algorithm performance is distance yielded by the algorithm. This is split into three main areas, each recorded by generation/iteration - minimum (best) distance found, maximum (worst) distance found, and average (mean) distance found. Time taken is also discussed in this paper, as a strictly tracked metric.

All experiments shown are run several times to ensure integrity and accuracy. Nevertheless, due to the random nature of the algorithm, it may not be possible to replicate results exactly.

2 Answers to Questions

2.1 Which combination of parameters produces the best results?

The below table shows the parameters that were varied for the experiments, and what possible values they could assume. While another value was being modified, the parameters would default to certain values; the default colony size was 4 and the default evaporation rate was 0.5.

Parameter	Value
Colony Size	2, 4, 6, 8
Evaporation Rate	0.25, 0.5, 0.75

Table 1: Parameters and potential values assumed.

A variety of combinations of parameters were used. A main characteristic observed was that increasing the colony size would mean that iterations would take significantly longer to run, but would also find much more optimal results in fewer generations.

These diagrams are for the Brazil (58) data-set. As shown in the diagrams, as colony size increased, plateaus were reached later and later. Additionally, distance reduced slightly, indicating improved accuracy, on average reducing from a distance of 35'000 to 30'000 as colony size grew from 2 to 10.

However, much more dramatic than this slight accuracy increase was the time taken to process. With a colony size of 2, 18.5 seconds was the average time to complete all runs and hit a plateau. In a stark contrast to this, with 10 ants in a colony, the duration was 112 seconds on average. These statistics are available to view in the included directory for Brazil's results.

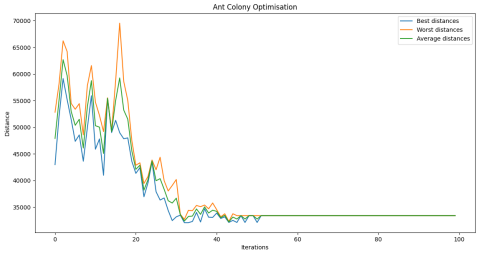

Figure 1: Brazil - Colony Size 2

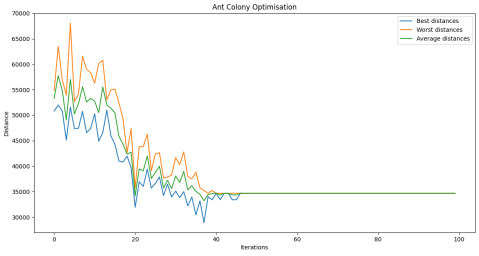

Figure 2: Brazil - Colony Size 4

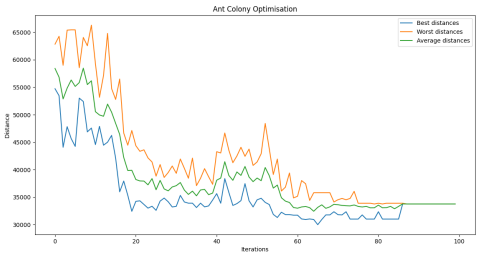

Figure 3: Brazil - Colony Size 6

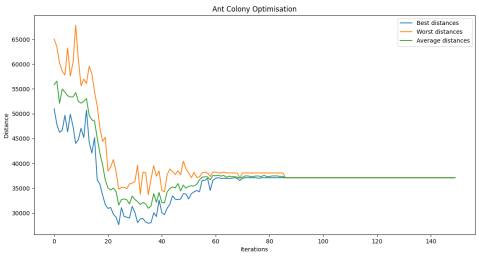

Figure 4: Brazil - Colony Size 8

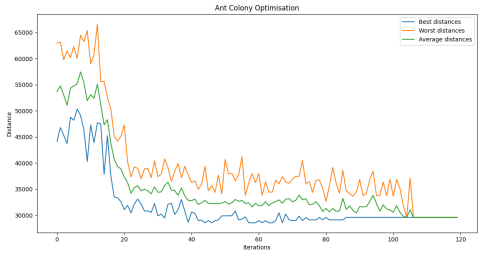

Figure 5: Brazil - Colony Size 10

With evaporation rate, a somewhat similar trend arose. The below diagrams show results for the Burma (14) data-set. As shown, the lower the evaporation rate, the much higher the spread, leading to a lot of "jumps" in the data on average. However, the best results were yielded from a medium evaporation rate of 0.5, which was the only one of the evaporation rates to fall below a distance of 4'000 on average upon plateauing. With regards to time taken, on average, evaporation rate seemed to have a negligible effect, with all durations quite similar to one-another regardless of rate.

Please note that the dis-included results (evaporation rate for Brazil data-set/colony size for Burma data-set) are not shown due to space limitations, but are available to review in their respective directories. The text files show paths taken by ants and their distances, as well as ants best attempts and time taken (at the bottom of every text file).

2.2 What do you think is the reason for your findings in Question 1?

Colony size drastically affected time taken, because it significantly increases the amount of processing per iteration - with more ants, more processing must take place. More optimal results were found in fewer iterations because more ants were working towards a solution per iteration.

Pheromone evaporation rate doesn't affect time taken very much, because it is a very small part of the computing process, and regardless of rate, remains fixed in computer resource consumption. The erratic jumps and variation are caused by significant changes in pheromone level during the iterations, leading to more random city selections. A medium pheromone evaporation rate strikes a balance between these changes, and hence does the best job at converging to an accurate solution.

Plateaus occur due to the algorithm becoming stuck on local optima, and never moving from them due to lack of incentive from the heuristic and pheromone matrices. Once these optima have

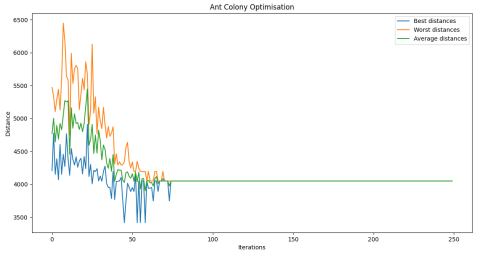

Figure 6: Burma - Evaporation Rate 0.25

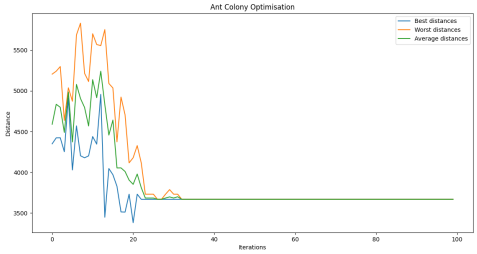

Figure 7: Burma - Evaporation Rate 0.5

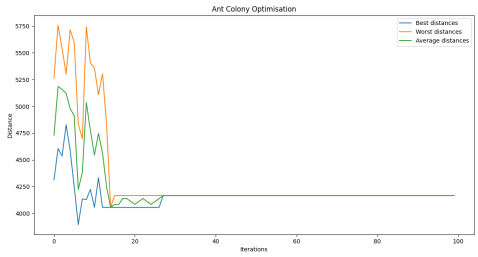

Figure 8: Burma - Evaporation Rate 0.75

been reached, there's no way off of them.

2.3 How does each of the parameter settings influence the performance of the algorithm?

A larger colony size means that better solutions are found in fewer iterations, however, iterations take longer are significantly slower. Because of this, the time complexity performance of the algorithm will undoubtedly suffer.

A higher pheromone evaporation rate leads to a faster plateau; a slower rate means that the plateau happens much later. These affect accuracy of the algorithm, since faster plateaus generally means less search-space has been explored, and the solution found is much more sub-optimal.

2.4 Can you think of a local heuristic function to add?

A good local heuristic function to add would be one that allows the algorithm to escape local optima.

A potential way of doing this would be using elitism. Elitism is an ACO approach variation that involves the algorithm working as usual, but pheromones are deposited on the global best path found (so far) on every turn, regardless of whether or not this path has been visited. Doing this encourages ants to search, even beyond the usual plateau that we've seen throughout these results.

Another potential heuristic function would be one similar to the A-star algorithm, using an informed search to find better solutions. This would be beneficial because it would result in better results being found faster.

2.5 Can you think if any variation for this algorithm to improve your results? Explain your answer.

Adding a stronger local heuristic function would improve results because it would allow the algorithm to converge on an "acceptable" result in much less time. However, this would not necessarily solve the issue of the algorithm becoming stuck on local optima.

A potential improvement that could solve the issue of becoming stuck on local optima is to deploy ants from many locations, instead of sticking to a single location. By doing this, we allow ants to explore more options and come up with a more diverse range of results - this diversity in results may allow ants to break through plateaus and find more optimal paths.

As previously mentioned, I attempted to implement elitism in my ACO implementation. The elitist ACO variation can be found in elitist.aco.py.

This variant was run on both the Brazilian and Burmese data-sets, with a colony size of 4 and an evaporation rate of 0.5. With this setup, the algorithm yielded the below results, with an average of 40 seconds for Brazil (58) and an average of 0.4 seconds for Burma (14). As can be seen below in both cases the elitist algorithm supplies slightly improved results, especially in the case of Brazil - plateaus take place later and finalise on more accurate routes.

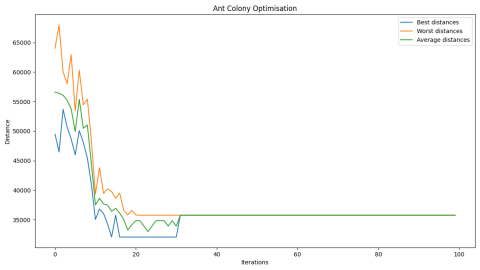

Figure 9: Brazil - Elitist ACO

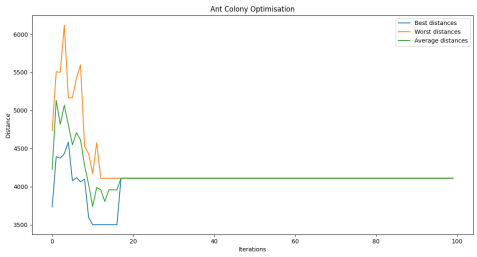

Figure 10: Burma - Elitist ACO

2.6 Do you think of any any other nature inspired algorithms that might have provided better results? Explain your answer.

Evolutionary algorithms may provide better results for this problem. This is because they are more flexible and can be modified to be more resistant to getting stuck on local optima. Additionally, EAs do not converge as quickly, and can be deployed from multiple nodes, greatly reducing the chance of plateauing and making it much more likely that they will reach a near-optimal solution.