Ant Colony Optimization and the Quadratic Assignment Problem – 700047428

I. Implementation

My ant-colony optimization (**ACO**) implementation consists of two files: `world.py` and `main.py`. In `world.py`, a World class is defined, which holds all the information for a particular ACO run (i.e distance, flow, and pheromone matrices), along with methods to generate ant paths, apply pheromones, calculate fitness, and plotting results. `main.py` contains methods for running the program, which enable running multiple attempts concurrently, so we can compare multiple attempts faster. For each run, the program also saves plots under `plots/`, stats under `stats/`, and final saved World objects under `worlds/` (These are useful if we want to do analysis without having to rerun experiments).

II. Results

The cost of the best ant in each experiment ran are depicted in the chart below. m is the number of ants in the simulation, and e is the evaporation rate. Each attempt ran for 10,000 iterations. The lower the cost, the better.

	m=10, e=0.9	m=10, e=0.5	m=100, e=0.9	m=100, e=0.5
Attempt 1	5591286	5591738	5601190	5623208
Attempt 2	5604042	5664992	5594558	5666120
Attempt 3	5644360	5677386	5597064	5614620
Attempt 4	5651770	5662868	5607330	5607728
Attempt 5	5648544	5648482	5638490	5621982
Average	**5,628,000**	**5,649,093**	**5,607,726**	**5,626,732**

Figure I: Best ant cost table per attempts and parameters

To better visualize this, here's a bar chart comparing the average for each parameter pair.

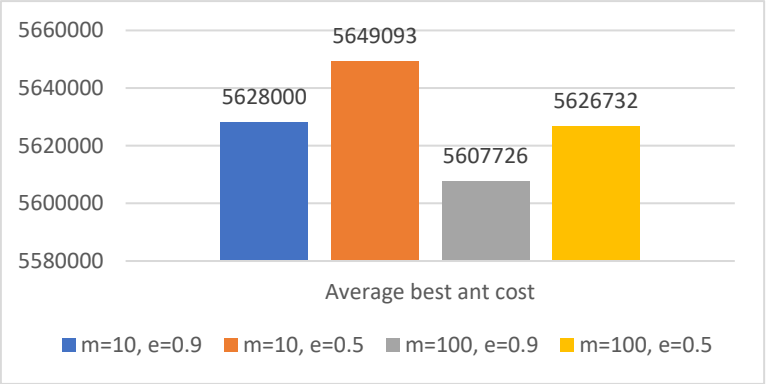

Figure II: Bar graph of average ant costs

III. Questions & Analysis

Question 1: *Which combination of parameters produces the best results?*

We can decipher a couple of things from this data. On average, the best combination of parameters for this problem is an **e** value of 0.9, with an **m** value of 100. Generally, these parameters performed the best, with an average best ant cost of 5607726. Meanwhile, the opposite parameters performed the worst, with **e**=0.5 and **m**=10 having the worst average ant cost, a value of 5649093. These results can be seen in *Figure I* and visualized in *Figure II*, where the differences between parameters become apparent. Interestingly, the best ant was found in the situation with **m**=10 and **e**=0.9, which is likely due to pure randomness (as on average, these parameters performed the second worst)

Question 2: *What do you think is the reason for your findings in Question 1?*

The fact that the best and worst sets of parameters were complete opposites says a lot about the nature of the problem.

Firstly, the problem has a massive number of potential solutions. As we have 50 facilities to place in 50 different locations, this is a permutation that leaves us with 3.04140932 E+64 potential solutions. With just 10 ants, we were not able to explore any notable portion of the solution landscape, and therefore it struggled significantly more to find high quality solutions than when using 100 ants. Using 100 ants meant that more of the search space could be explored, and significantly better solutions could be found.

Beyond that, the fact that the best solution had an evaporation rate of 0.9 is likely due to problems with a higher number of possible solutions benefitting from slower convergence. The worst solution having an evaporation rate of 0.5 meant that it converged on a solution much more quickly, getting stuck in local minima as opposed to exploring different solutions. The best case solution's evaporation of 0.9 meant that different decisions weren't ruled out as quickly, and the ants were able to make more, and better decisions.

In *Figure III,* we compare two samples of ACO fitness over time, with the best parameters on the left and the worst on the right. It's worth noting that the worst parameters had significantly more randomness, whilst the best case was less messy. Beyond that, although the average solution costs were similar across both tests, the best solution costs of the m=100 and e=0.9 experiment were generally much lower than the other. These are both visualizations of the effect of population size and evaporation rate on the ants in the program.

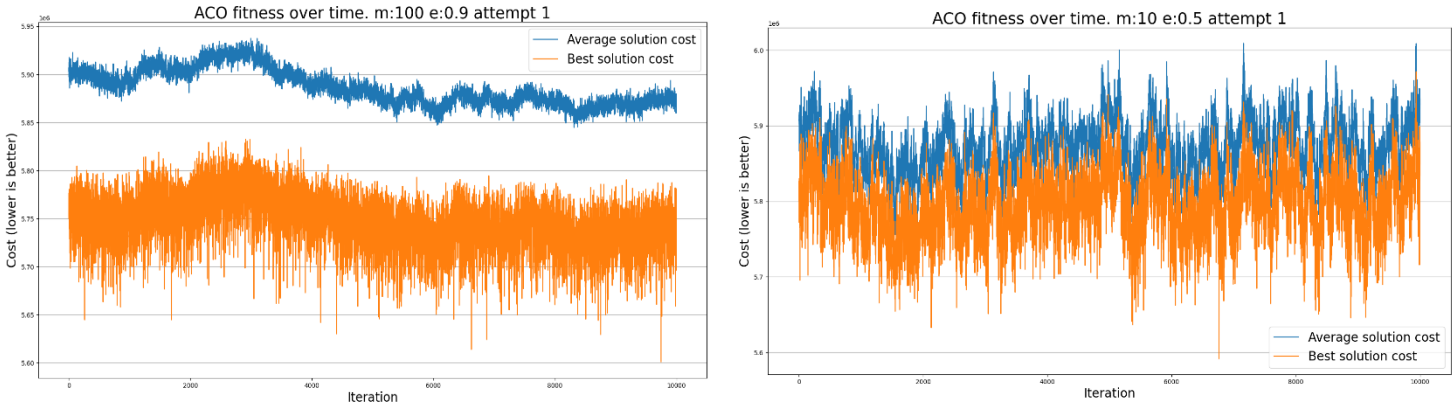

Figure III: Over-time ant costs for the best and worst parameters from Figure I

Question 3: *How do each of the parameter settings influence the performance of the algorithm?*

To determine the influence of each parameter on the results, we had to find the averages of each parameter's results, otherwise known as the parametrized averages. In *Figure IV* we can see the average performance of each parameter setting compared to the other. These two tables shouldn't be compared to each other, but it can be used to compare the performance of m=10 to m=100, and e=0.9 to e=0.5

	m=10	m=100
Average	**5,638,546.5**	**5,617,229**

	e=0.9	e=0.5
Average	**5,617,863**	**5,637,912.5**

Figure IV: Average m and e ant costs

Per *Figure IV*, an **m** value of 100 generally performed than an **m** value of 10. Similarly, an **e** value of 0.9 performed better than an **e** value of 0.5. This is in line with the hypothesis proposed in the answer to *Question 2*, being that m=100 and e=0.9 yielded better results on average than the alternatives. Again, these results are unsurprising due to the large search space of the problem.

Question 4: *Can you think of a local heuristic function to add?*

To improve the results of the ACO, we could add a local heuristic that would "sway" ants into making better decisions when picking a node. There's a few greedy heuristics that could potentially improve the

results of the algorithm. These could be balanced with the pheromones using a beta coefficient, which would be a float value that dictates the influence of pheromones vs heuristics for ant decision making.

When picking the next node in the ant's path we could pick the next facility based on the highest flow value. We could look at the path so far, and the facilities that haven't been placed yet, and place the facility that has the highest flow value between it and the previously placed facilities. This would add moderate computation costs, as you'd have to loop through all facilities to determine which one would maximize flow when placed next.

We could also pick the next facility to be placed based on how it'd affect the cost of the solution. When ants are picking the next node in their path, they would calculate the cost of the next step, with every solution. If the ant has the path [3,5,2,1] so far and they have to place facilities 4 and 6, the ants would calculate the costs of solutions [3,5,2,1,4] and [3,5,2,1,6] to see which is better. This means that we can rank the decisions the ant can make to guide them in the direction of better solutions. The main downside of this strategy is that it would add significant cost in terms of computing power, as fitness evaluations would have to be done hundreds of times when creating ant paths as opposed to at the end of the generation.

Question 5: *Can you think of any variation for this algorithm to improve your results?*

Aside from adding a heuristic, the algorithm would likely benefit from a few variations. First, we could improve results by initializing pheromones to higher values in the range [0,1], which would enable more decisions to be made by the ants, specifically at the beginning of the run. This means that good solutions wouldn't be ruled out purely due to their initial pheromone being randomly lower than others.

Another variation would be to add max and min values of pheromones to be applied per ant – that is to say, if the pheromone is too low, it's set to the minimum value on the range, and if it's too much, it's set to a maximum. This would allow for less elitist ants, and more solutions to be explored.

Finally, probably the best improvement would come about by taking the top 30% of ant solutions and applying more pheromone from those ants than others. By doing this, we'd place more emphasis on the better solutions, allowing us to encourage them, and ideally get more convergence towards them.

Question 6: *Do you think of any other nature inspired algorithms that might have provided better results? Explain your answer.*

I believe that a better solution to the QAP could be achieved by using an evolutionary algorithm. Using an ACO for this problem had various shortcomings, most of which would be resolved by employing an evolutionary algorithm instead.

An EA would need a script to generate solutions, which would initially be random, but then the solutions with the lowest cost would be preserved while the rest are dropped. After that, we cross-over the remaining solutions by combining different parts of each to refill the population and apply random mutations.

With this problem having so many nodes to place, the differences between the best solutions and the worst solutions were somewhat small. According to *Figure I*, the best solution had a cost of 5,591,286, whilst the worst had a cost of 5,677,386, a difference of 86,100. Although this is notable, it is only a difference of around 1.5%, and when it comes to applying pheromone, you're applying 1.79e-7 pheromone from the best solution and 1.76e-7 for the worst, which is essentially a negligible amount. Instead of doing the pheromone approach, an evolutionary algorithm would drop the worst x% of solutions, so they will not have an impact on future decisions. This means that by using an evolutionary algorithm, we can get rid of the poor solutions to encourage better solutions to appear.

Beyond that, an ACO solution to the QAP has too much randomness. As can be seen in *Figure III*, the best ant costs bounce around an extreme amount – there are trends in the data, but the best ants in

adjacent iterations can have differences in fitness of almost 200,000. This is too much noise to find a great solution, which could be resolved with an evolutionary algorithm. Although a key aspect of EAs is controlled randomness, at the end of each iteration we'd drop the worst solutions, meaning that in general, each generation will have more similarities and trend towards lower costs.

Another benefit of using an EA is the lowered computation cost. When using an ACO, we have to calculate the amount of pheromone for each solution, apply it, evaporate it, and take it into account when creating the next generation of ants. When using an EA, this is not a consideration, as instead of having to use pheromones to create solutions, we just focus on crossing over and mutating the existing solutions. This is not to mention, as previously mentioned, that even the best solutions in the ACO apply an almost negligible amount of pheromone to the table – It's a waste of computation time.

IV. Conclusions & Further Experiments

In general, I found that ACO wasn't the best method to solve this problem. I think the pheromones did have an impact, and did lead to some improvements in the results, however I don't think that this yielded the best results. As explained in the answer to question 6, I believe an EA would have been much more suited to this problem.

To quantify how much the pheromones were aiding in finding a solution, I decided to run very similar experiments, except I didn't use any pheromones in the decision making, and instead, ants were selected entirely randomly. This would allow us to determine the effect of the pheromone in the decision making compared to completely random decisions. The results are depicted in *Figure V*. As we can see, compared to the plots in *Figure III* and the results in *Figure I*, there is an improvement in the solutions when using the pheromone as opposed to fully random decisions. The fitness over time in *Figure V* doesn't really follow a trend and is essentially completely random. Beyond that, the best ant cost achieved in this run was 5,648,878, which is lower than runs with the same parameters seen in *Figure I*. However, it presents better results than the worst-case parameters, reinforcing that m=100 and e=0.9 are ideal in this application. This proves that although the ACO was not super effective for this problem, it does represent a tangible improvement over fully randomized solution creation.

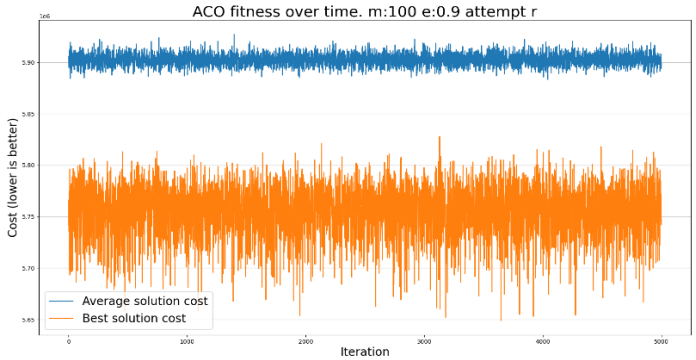

Figure V: Ant fitnesses over iterations with completely random paths

To experiment with different cost formulas, I attempted to apply pheromone per-segment cost instead of per-path cost. Ideally this would have increased the amount of pheromone applied to approx. 1/50 as opposed to 1/5,000,000, however this did not return good results. The final results were worse than the normal algorithm, and the calculation took much longer due to having to contain and calculate the cost of each location-facility pairing. I realized after this, that although extremely small amounts of pheromone were being applied, due to the high number of total iterations (10,000) and the evaporation rate, small amounts of pheromone became much more significant after just a few iterations.

In the future, when addressing this problem, I'd like to attempt the variations presented in the response to question 5. I had my doubts about the effectiveness of this ACO implementation for this problem, however my own experiments with randomness and different cost functions showed me there were emergent properties in the algorithm that aren't noticeable in isolation.