Nature Inspired Computation:

Ant Colony Optimisation

Candidate Number: 239106
Alfie Walliss

All figures in this report are from a sample run. Ant colony optimisation is non-deterministic therefore on a different run of the program the results may not be the same.

1 Question 1

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

Required Experiments

	M=100, e=0.9	M=100, e=0.5	M=10, e=0.9	M=10, e=0.5
Mean Fitness	5753521.6	5749229.6	5841254.0	5824146.8
Best Fitness	5738812	5719140	5801938	5768352
Worse Fitness	5769888	5818092	5887412	5891900

Extra Experiments

	M=50, e=0.9	M=50, e=0.5	M=200, e=0.7 (elitist)
Mean Fitness	5762767.6	5777069.2	5724496.0
Best Fitness	5712010	5754378	5705038
Worse Fitness	5802806	5802140	5753972

Figure 1: Table to show the Mean Fitness, Best Fitness and Worse Fitness produced over five iterations with each configuration

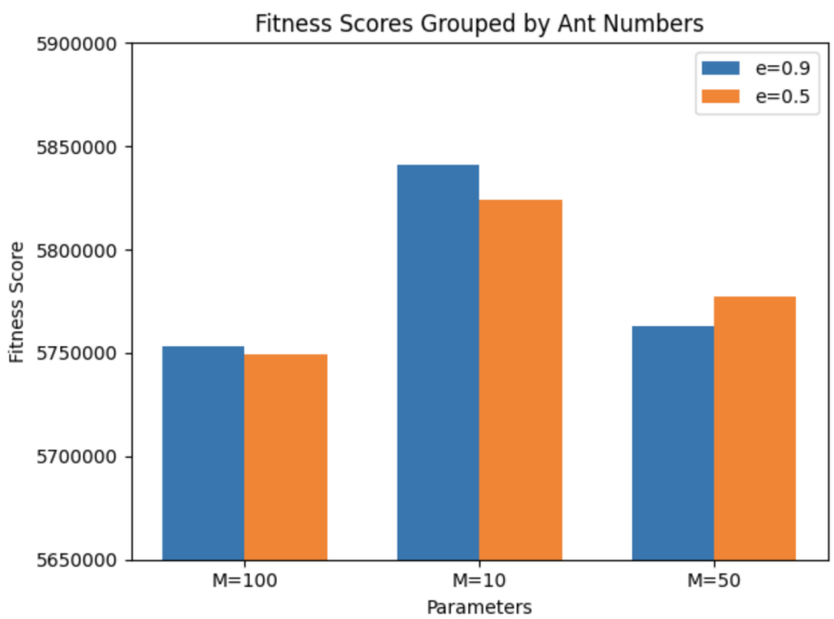

Figure 2: Grouped bar chart to show the effect of changing the number of ants and the evaporation rate on the mean fitness

From the set of parameters required in the specification, it is clear from Figure 1 and 2 that a value of 100 ants produces better average results than that of 10. The reason for this is that both experiments using 100 ants have lower finesses scores for all fields in the table than any experiment using 10 ants. As for the evaporation rate, in the results a larger evaporation rate appears to produce a worse fitness score. However, from running the program multiple times, it seems this can change and sometimes a lower evaporation rate produces a better fitness score. As extra experiments, I also decided to implement two more trials using 50 ants with the same evaporation rates. As you can see from Figure 2, an evaporation rate of 0.5 has produced worse results with 50 ants than that of 0.9. This is contrasting with both 10 and 100 ants. As expected, the mean fitness scores with 50 ants is approximately half way between that of 10 and 100 ants. I decided to implement the 50 ant experiment to be confident that the relationship between the number of ants was near linear and that there wasn't a far optimal number of ants between 10 and 100.

2 Question 2

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

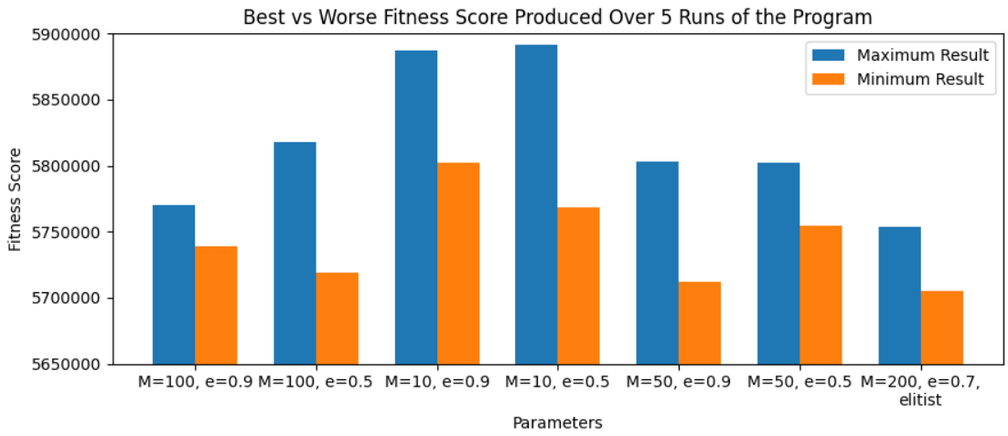

Figure 3: Grouped bar chart to show the difference between the best and worse fitness scores over the five iterations of the algorithm with different parameters

The reason that the greater number of ants produces a better mean fitness score is due to the likelihood of more paths being explored increasing. When there is few ants, fewer paths will be explored. This will increase the pheromone concentration on these paths even if they are non-optimal solutions increasing the chance of these paths being explored again. There is a chance when using few ants that one of the paths explored has a low fitness value, however the probability of this is far lower than when the algorithm is producing 100 solutions per iteration. With 100 ants, the probability of more paths being explored increases, which will likely lead to better solutions being found. 100 ants constantly produced the most similar results between the two evaporation rates. The reason for this is that there are enough ants that over the course of the algorithm, it is more likely that paths with low pheromone levels will still be explored. These paths will then be updated if they produce low cost function values. This reduces the effect of the evaporation rate with high levels of ants. When the algorithm is using 10 ants, the evaporation rate has a greater effect. A lower evaporation rate reduces the amount of pheromone on each path resulting in a further decrease in exploration. This will cause a faster convergence rate on an local optimal solution. Figure 3 is useful to highlight the effect of the evaporation rate. A lower evaporation rate usually results in a greater disparity between the best and worse solutions over the five iterations. The reason for this is that it is likely that the program will converge on a highly nonoptimal local minimum quickly, rather than exploring the whole fitness landscape. In comparison to this, there is a chance that one of the original paths that the ants choose to explore is a near optimal solution. This will therefore result in convergence on a global optimum which is why the best results with a 0.5 evaporation rate are usually better than the best results with a 0.9 evaporation rate. With a 0.9 evaporation rate, even if a near optimal path is selected near the beginning of the algorithm, there is still a high change that the algorithm will move away from this solution in the fitness landscape.

3 Question 3

Question 3: How do each of the parameter settings influence the performance of the algorithm?
For the ant colony optimisation algorithm implemented, there are three main parameters which can be altered: the number of ants, the pheromone evaporation rate and the termination criteria. The number of ants effects the exploration rate; the more ants used in the algorithm, the more paths which are able to be explored. Even paths with low pheromone will eventually be explored if the number of ants is high enough. The evaporation rate is the multiplication factor of which pheromone is removed from the paths after each iteration of the algorithm. The evaporation rate effects how quickly the ants converge on a single solution. If the evaporation rate is low, lots of pheromone will be removed from all paths increasing the effect of the update pheromone function. The same amount of pheromone will be deposited regardless of the evaporation rate. As a result of this there will be few paths with high levels of pheromone leading to early convergence on a local minimum. Finally, the termination criteria will also effect the performance of the algorithm. For all trials and experiment the program was terminated when 10,000 fitness evaluations had been completed. It is important that the program converges on a solution before the algorithm is terminated. In a future experiment it may be beneficial to run the program with a different number of fitness evaluations to ensure the program had converged by 10,000 iterations but also to ensure that the program isn't being run an unnecessary large number of times. An alternative termination criteria could be running the program until a certain fitness threshhold is reached or until contiguous fitness values are achieved within a prescribed limit.

4 Question 4

Question 4: Can you think of a local heuristic function to add?
For a local heuristic, a method is required which is able to accurately represent the distance and flow between each location. The relationship between distance and flow is inversely proportional therefore a heuristic such as $\frac{flow}{distance}$ which represents the flow per unit of distance between two facilities would be appropriate. Adding a heuristic to the algorithm allows the ants to make more informed choices about which path they will enter. This will result in a greater number of more optimal solutions being explored which should increase the chance of conversion on a

global minimum solution. It is important when implementing a heuristic approach that the state transition rule with appropriate fixed parameters is chosen. This is because if there is too much weight on the heuristic, the ant colony optimisation algorithm could turn into a greedy search and find a solution far from optimal getting stuck in a local minimum. Having a heuristic also allows more exploration of more optimal paths before the termination criteria of maximum iterations is reached.

5 Question 5

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

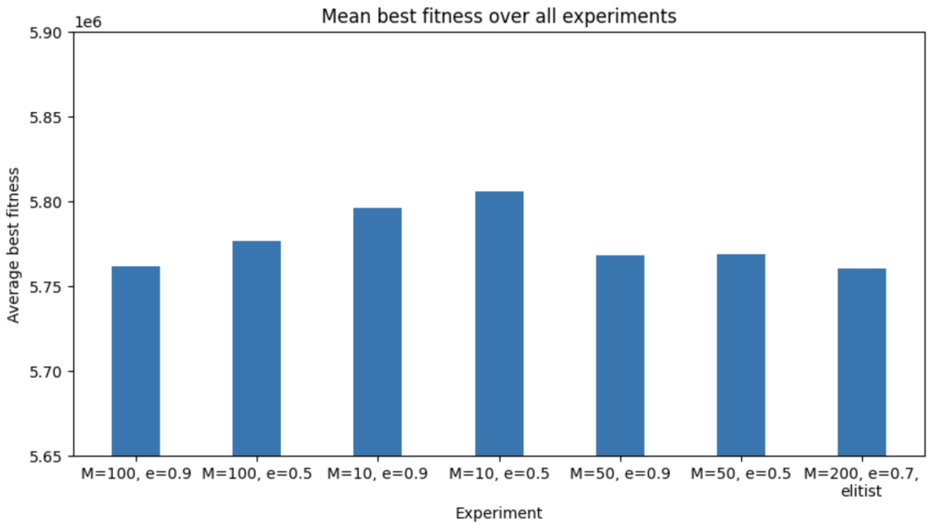

Figure 4: Bar chart to show all the mean average fitness produced in the final run of the ACO on a sample run

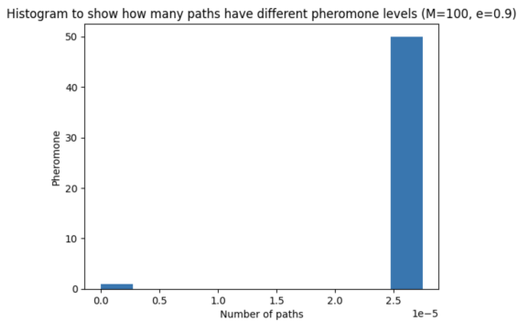

Figure 5: Shows a histogram of pheromone for a pheromone matrix on the final iteration with M=100, e=0.9 from location 4 to all other locations

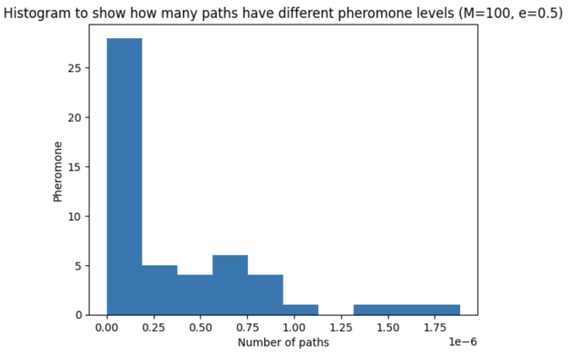

Figure 6: Shows a histogram of pheromone for a pheromone matrix on the final iteration with M=100, e=0.5 from location 4 to all other locations

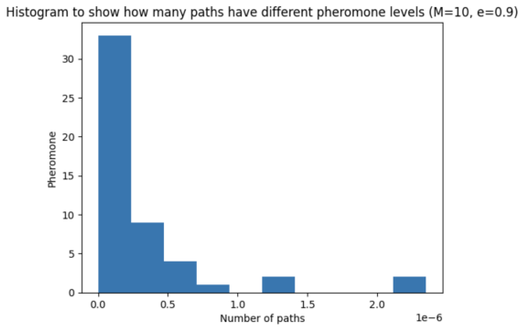

Figure 7: Shows a histogram of pheromone for a pheromone matrix on the final iteration with M=10, e=0.9 from location 4 to all other locations

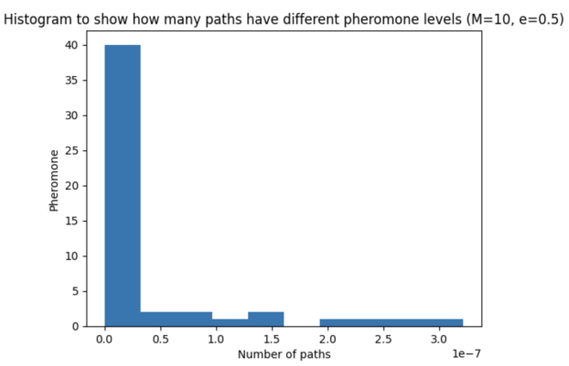

Figure 8: Shows a histogram of pheromone for a pheromone matrix on the final iteration with M=10, e=0.5 from location 4 to all other locations

One variation of the algorithm to improve the results would be to implement an elitist version. Elitist ant colony optimisation is where you only run the deposit pheromone function on the solutions which produce the best fitness scores. This means that solutions that are far from optimal do not receive any extra pheromone on their paths

therefore reducing the likelihood of ants using them in future iterations. This will increase the probability of ants exploring better solutions which should result in convergence on a more optimal solution. As an extra experiment, I decided to implement the elitist ant colony optimisation only running the update pheromone function with the top 10% of solutions. The algorithm was run five times with 200 ants and with an evaporation rate of 0.7. This means that per each iteration of the algorithm, only 20 paths will be updated due to the 0.1 selection rate of the elitist mechanism. Figure 1 and 4 show the results of this experiment and it is clear that the elitist algorithm outperformed all other trials using non-elitist with a lower mean fitness, best fitness and worse fitness score. Another variation of the algorithm could be to increase the number of fitness evaluations required for the termination criteria. It could be possible that the program is terminating before the algorithm has fully converged on a solution. The histograms in Figures 5 to 8 highlight how for 100 ants and a 0.9 evaporation rate, the algorithm hasn't been given adequate time to converge. Figure 5 shows that there are many paths with similar levels of pheromone when the program finishes running. Contradicting to this Figures 6 to 8 indicate that there are very few paths with high quantities of pheromone highlighting the convergence on a solution with these parameters. Running the program with 50,000 fitness evaluation rather than 10,000 should mean that the algorithm always converges on a solution. This could be an interesting extra experiment to conduct however, the compute time for the algorithm to run on the same hardware running the other experiments would be far too long with the current implementation. Another variation for the algorithm that could improve the results would be to implement the heuristic mentioned in **Question 4** which should improve the results. This is because the ants will be informed about the paths they are going down reducing the degree of randomness especially in the early stages of the algorithm. You could also use a max-min ant system which is used to prevent stagnation on a local minimum by having a maximum and minimum amount of pheromone which is allowed to be distributed on each solution. This means that if one solution is by far the most optimal in a given iteration, not too much pheromone will be deposited on it. This will prevent conversion on a good path as it may still not be the optimal solution. It also means that if a solution produces very poor results, a moderate amount of pheromone will still be distributed on it meaning that the paths may still be explored in future iterations as they could still be involved with the optimal solution. The algorithm may have to travel through the search space of these non-optimal paths to find the global minimum point on the fitness landscape.

6 Question 6

Question 6: Do you think of any other nature inspired algorithms that might have provided better results?

One nature inspired algorithm that might have provided better results is an evolutionary algorithm. Evolutionary algorithms are based on the process of evolution; the algorithm begins by generating a certain number of solutions. These solutions will then be evaluated with a fitness function and the top solutions will be selected. The selected solutions will then undergo a cross-over function which is where two solutions are combined producing either one or two further solutions. Finally, the mutation function will occur where random parts of the solution will be altered simulating mutation in evolution. This process will then be repeated with these solutions being the next to be evaluated. The algorithm will stop when a certain number of iterations has been reached or when a solution is produced with a low enough fitness value specified by the implementation. The reason a genetic algorithm could provide a better result than that of the ant colony optimisation algorithm implemented, is that genetic algorithms can converge on more optimal solutions faster meaning that if both algorithms run over 10,000 iterations, genetic algorithms are more likely to converge on a final solution.

Another nature inspired algorithm that may provide better results is simulated annealing. Simulated annealing is similar to a hill climbing algorithm however it allows worse solutions to be chosen to leave a local minimum. The algorithm is initialised with a high probability of choosing worse solutions and then slowly reduces this probability choosing better states. This means that in a fitness landscape over the first iterations while the probability of choosing a poorer solution is high, the aim of the algorithm is to find the area that contains the global minimum. As the probability decreases this local area will be explored and the global optimal solution will be found. Simulated annealing is seen to be far more efficient than both ant colony optimisation and also genetic algorithms whilst still producing quality results.

7 Conclusion

Overall, it is clear that running ant colony optimisations with 100 ants is optimal over all of the trials conducted. Further research would be required to confidently decide which evaporation rate produces better result. This is because it is clear that there is a degree of randomness due to low evaporation rates converging faster which could be on a global minimum if lucky, but will more likely be on a local minimum and a poorer solution. Having an average over more iterations than five could also solve this problem, however once again the hardware running the simulation would take too long for these results. It is highly probable that implementing the heuristic mentioned in question 4 would improve the performance of the algorithm which could be another further experiment to implement. Implementing the elitist algorithm did improve the performance of the algorithm and an experiment varying the elitist selection rate could be beneficial. It would be interesting to use different nature inspired algorithms as stated in question 6 to see whether ant colony optimisation is the optimal algorithm for the question or whether there are other algorithms with better performance.