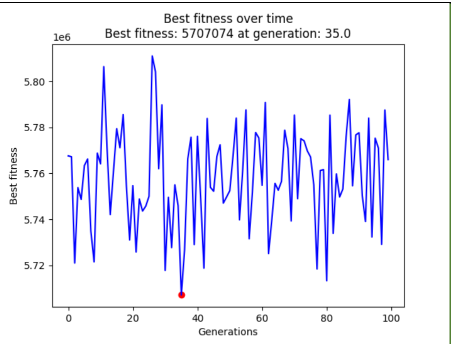
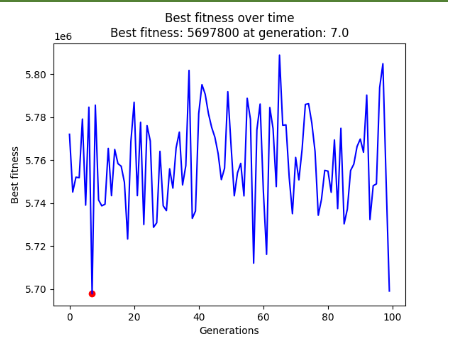
Nature-Inspired Computing Coursework Report

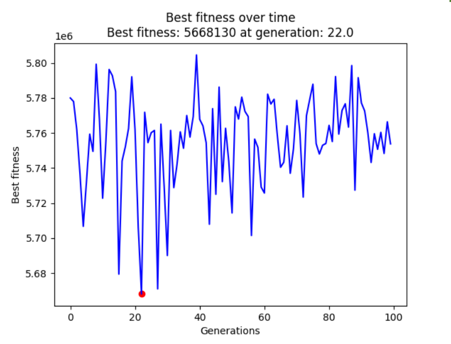
M = 100

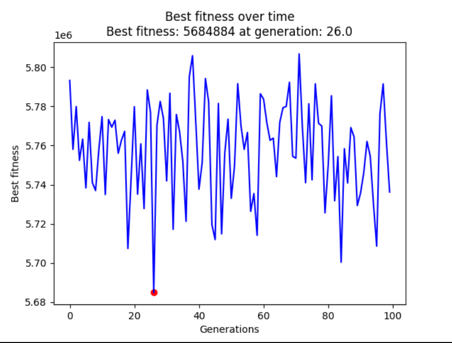
E = 0.9

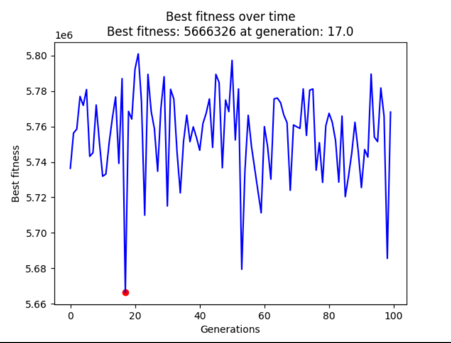
Avg. = 5684843







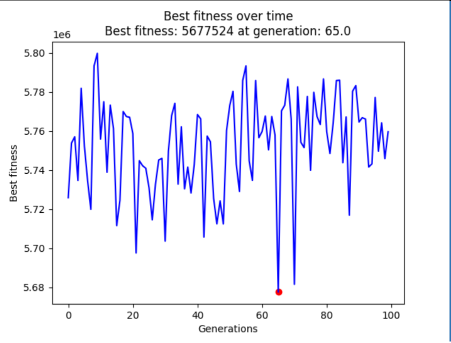


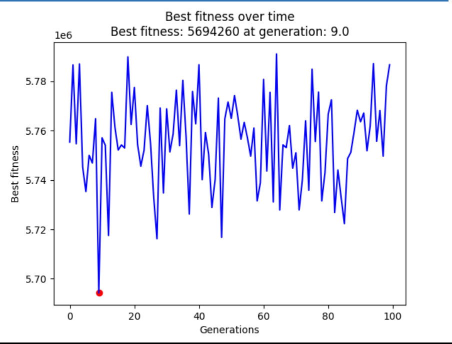


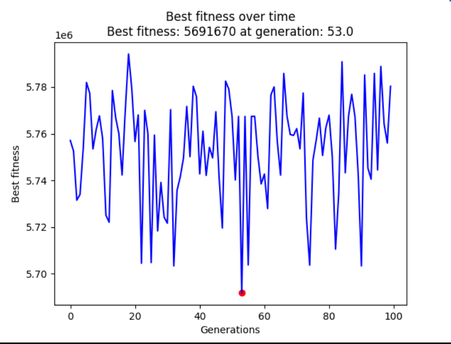
M = 100

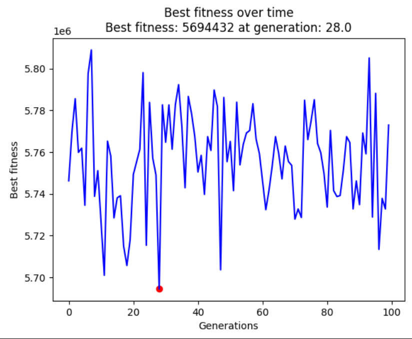
E = 0.5

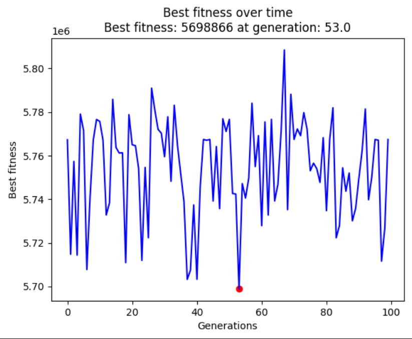
Avg. = 5691350







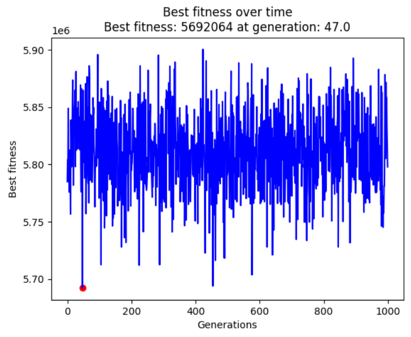


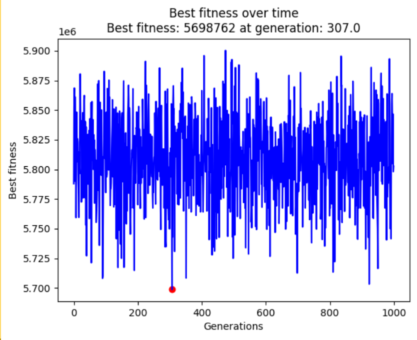


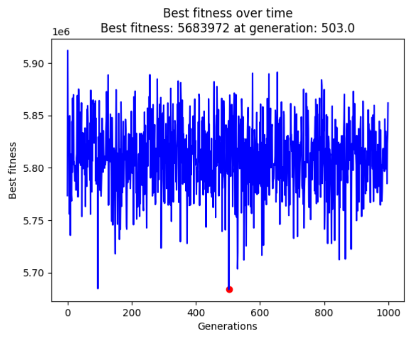
M = 10

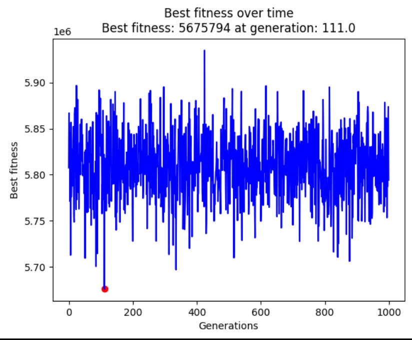
E = 0.9

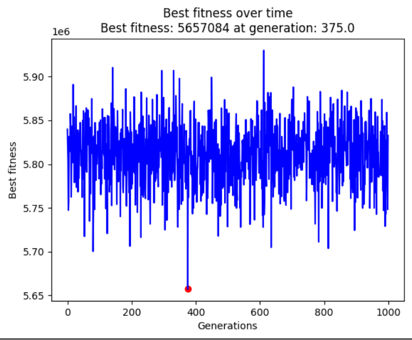
Avg. = 5681535







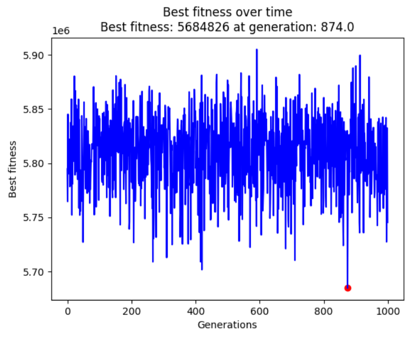


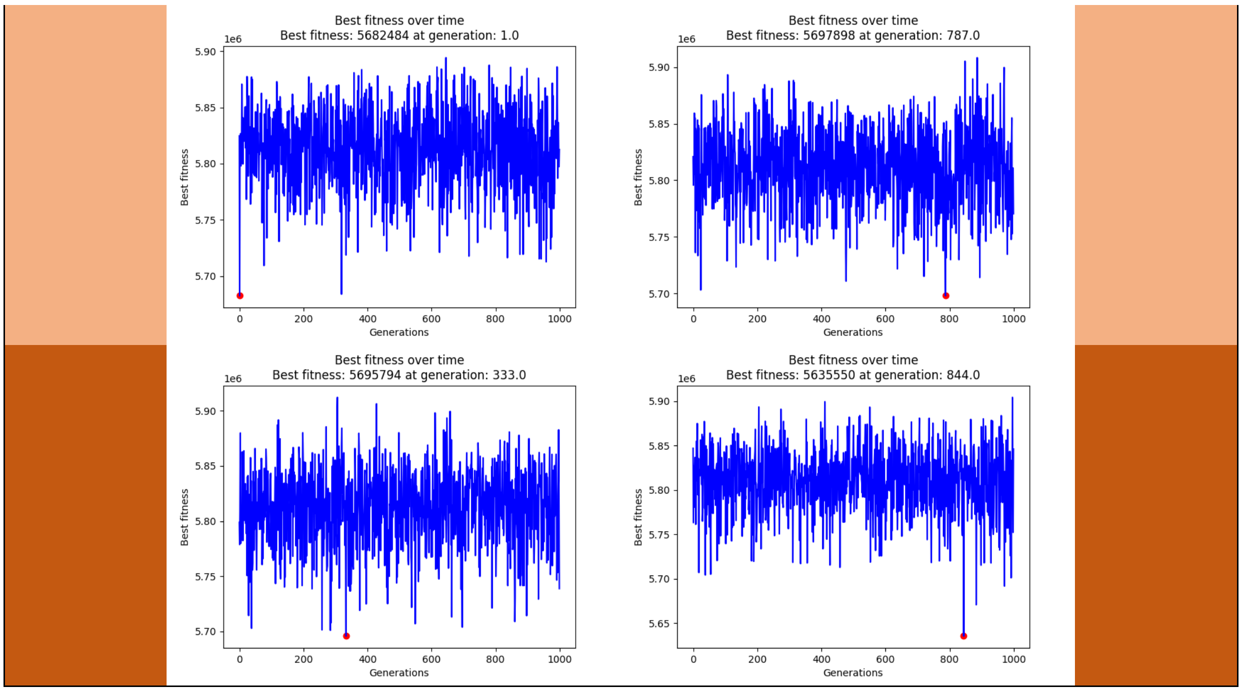


M = 10

E = 0.5

Avg. = 5679310





Note: I have created other resources throughout the development of this project that you may view in '/resources', should you wish. I also discuss them here.

Question 1: The best combination of parameters was M = 10 and E = 0.9, as this provided the lowest average of results, EXCLUDING anomalies. There was an outlier in the above M = 10, 0.5 trials which allowed for a lower average fitness of 5679310, but discarding that anomaly gave us an average of 5690250. Further experiments were observed but not recorded, where ALPHA and BETA were both experimented with (set to 1, and 5 respectively, to ensure a local maximum wasn't reached), however, these seemed to show very few differences. Despite this, we see that there is no obvious trend in the best solution's direction.

Question 2: This is surprising since intuitively, a higher population usually yields better results. The loop, however, was based on number of fitness evaluations instead, allowing a smaller population to explore 10x more results. Since the implementation relies on randomness to provide suitable results, more exploration allows for a result that could be closer to the optimum. Furthering this, a higher evaporation rate ensures that the pheromone matrix, which is inherently built on randomness and whose updates are negligible, is removed of any values that could potentially negatively (as well as positively) sway the direction of travel. This effect can also be seen when comparing the runs of M=100, E=0.5/0.9. We also see that the population (and thus number of solutions) influences the algorithm to produce a better result than the evaporation rate does, going by the average (excl. anomalies) results.

Question 3: There are two aspects to the performance of the algorithm that we may discuss. The first, and most obvious, is the algorithm's ability to tend to an optimal solution. We can see that a higher population of ants gives us a 'best' solution that is closer to the optimum than most, however, due to the entropy inherent in this implementation, larger numbers of generations, or fitness evaluations, aren't especially useful. When ALPHA was set to a higher number, no difference was made to the algorithm's performance than when BETA was set to the higher number. We created test.txt which contained a smaller QAP, so that I was able to quickly test my algorithm. This very quickly converged onto the right solution ([2, 1, 0]), which shows the algorithms works, however, when faced with larger pieces of data, the interpretation of the pheromone matrix complicates the bias of the random choosing. The initial pheromone matrix also had a heavy influence, as the pheromone update function added negligible amounts to it over time. Secondly, we may discuss the performance in terms of speed of the solution. While efforts were made to optimise the final piece of code once it had been fully written, the inherent design was not optimised for a lower time/space complexity, thus slowing down the program when met with higher generations and populations.

Question 4: An interesting heuristic that could've been explored was one that decided an ant's movement based on purely the highest pheromone, or purely the closest distance. The formula given in the specification for calculating cost involves multiplying by the flow matrix. Ideally, we want a **low** distance and a **high** flow. By using the sum of the product of these two, we are creating a confusing outcome whereby a path with a high flow is punished. This should be changed to create a higher performing algorithm.

Question 5: There are a few options that we choose when varying this algorithm. We can create a weighted average of multiple ant colonies that each explore the same problem (with different pheromones and parameters). This could ensure that any local maximum that is reached is overshadowed by the contributions of other colonies. We could absolutely change the pheromone update rule. Since the cost of each path is so high (several millions), the inverse of it so negligible and barely affects the pheromone matrix from its initial values. We could have also implemented a better search heuristic to search the possibilities for an ant's movements in another way, other than randomly choosing and using the transition possibilities as weights.

Question 6: A semantic genetic algorithm could have been used to have reached an optimal solution. The problem could encoded as a set of genes that represent the assignment of buildings to a location. Genetic operators, such as a geometric crossover/ mutation (mutation is something the ACO lacks, though is slightly countered by use of the pheromone matrix) could be used to ensure the fast(er) approach to the optimal solution (since as semantic landscape is unimodal, and thus no other local maximum exists). Fitness would be based in a similar way the cost is calculated in the ACO. Like ACO, SGP should also iteratively converge on a optimal/ near-optimal solution.

To conclude, this implementation of the ACO was very flawed. Changing the parameters yielded no non-random effect, and this may be due to the cost and pheromone update functions being designed sub optimally for the problem. We could improve this experiment by altering these functions in the ways I have described in my answers to the previous questions.