Programming Task & Report: Ant Colony Optimisation

032588

1. INTRODUCTION

In this version of the 0/1 knapsack problem that was solved in this paper, there are 100 bags with unique weights (kg) and cash values (£), and there is a van with a maximum weight limit. The algorithm in this paper was designed to select the best combination of bags to fit in the van that can maximise the cash that is in the van without exceeding the weight limit of the van.

The algorithm used was the ant colony optimisation (ACO) algorithm, adjusted to fit the requirements of the 0/1 knapsack problem. The 0/1 knapsack problem is a classic combinatorial optimisation problem, it can let us effectively evaluate the efficiency of our ACO algorithm.

2. LITERATURE REVIEW

Different variations of the ACO algorithm have been used to solve the 0/1 knapsack problem, (Schiff, 2014) used a different method of calculating the heuristic value for each bag. The heuristic was calculated using the ratio of the square of the profit to the square of the weight of a bag. The ratio of the profit to the weight of the bag is always greater than one. Therefore, using the ratio of the square of the profit to the square of the weight will greatly amplify the heuristic value of a bag relative to the other heuristic values of bags if it has a larger value to weight ratio than the other bags. This algorithm makes exploration less likely and favors solutions that contain combinations of bags that all have very high profit to weight ratio.

3. METHOD

To solve the problem was I first split the 100 bags into 10 sets of 10 bags. Then I treated each set of 10 bags as if they were 10 nodes in the travelling salesperson problem (TSP). The equation that calculates the probability of an ant moving from one node to another in the TSP is as follows.

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{h \in J_i^k} [\tau_{ih}(t)]^{\alpha} \cdot [\eta_{ih}(t)]^{\beta}}$$

Where usually in the TSP, the heuristic information $\eta_{ij} = {}^{1}/_{d_{ij}}$ where d is the distance between nodes i and j. However, for my algorithm.

$$\eta_{ij} = {}^{v_j}/_{w_j}$$

Where v_j is the value of bag j and w_j is the weight of bag j. This allows the probability of the ant to travel from bag i to j be higher if bag j has a high value to weight ratio. Apart from the difference in calculating η_{ij} the rest of the equation is the same.

The ant must start at a node before choosing the first bag, therefore each set has a start node S which the ant is placed at before choosing the first node. Node S is not listed on the final list of bags that the ant has chosen and does not contribute to the weight of the van or the amount of money placed in the van.

The ant starts at node S and picks out the first bag out of the available ten. The ant keeps on picking out bags until it has picked seven bags in total. Then, the ant is placed at node S of the second batch of ten bags and repeats the process. This continues until the ant cannot fit any more bags in the van without exceeding the weight limit of the van.

If the ant selects seven bags from each batch of ten without exceeding the weight limit, then then ant is placed back on the neutral node from the first batch and selects one new bag. It moves to the neutral node of the second batch, selects one new bag and repeats the process of selecting one new bag from each set until the van weight limit is reached.

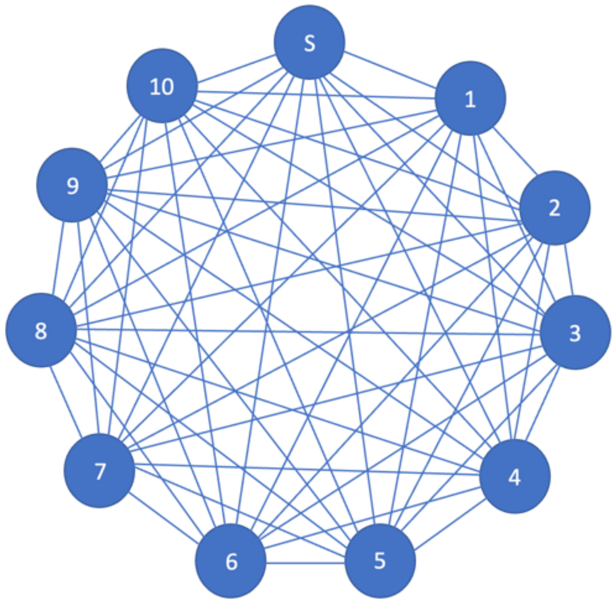

Figure 1. Construction graph, batch of 10 bags and start node.

The construction graph in figure 1 is one of the ten sets of ten bags. Each line joining each bag represents a different pheromone value. Each pheromone value is initialised with a random value from 0-1.

I choose the number of ants n, that are going to find solutions, and get them to calculate solutions, one by one. Each ant finishes generating its solution when the van weight limit is reached.

Once the van weight limit is reached, the pheromone values are updated relative to the value of the ant's solution.

Each pheromone in the ant's path is updated relative to the fitness of the solution, by adding $\Delta\tau_{ij}(t)$ to its value. If an ant did not use a path, then this does not occur.

$$\Delta\tau_{ij}(t) = {}^{fitness}/_{m}$$

Fitness is the solution value which is the amount of cash that the ant fit in the van. The value m is the amount of pheromone deposited relative to the fitness of the solution. Smaller values of m increase $\Delta\tau_{ij}(t)$, the pheromones deposited when an ant uses a path.

Then, all the pheromone values τ are multiplied by $(1-\rho)$ which is what evaporates the pheromone, ρ is the pheromone evaporation rate.

If a path wasn't used.

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t)$$

If a path was used.

$$\tau_{ij}(t+1) = (1-\rho)\cdot(\tau_{ij}(t) + \Delta\tau_{ij}(t))$$

After the ant finds a solution and the pheromones are updates, the process starts over, this repeats p times, where p (population) is the number of ants chosen for this iteration. This process of using p ants to create p solutions is repeated x number of times where $x = {}^{10{,}000}/_{p}$ .Finally, the best solution out of all 10,000 solutions is found.

The values p, m, and ρ will be varied to examine the performance of the algorithm when these have different values.

The values that will be kept constant are: $\alpha = 1, \beta = 2,$ total number of solutions = 10,000. The algorithm will be tested with the following values for p, m, and ρ.

p: 10, 20, 30, 40, 50, 75, 100, 150, 200.

m: 100, 200, 500, 1000, 2000, 3000, 4000, 5000, 6000.

ρ: 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95.

4. Description of Results

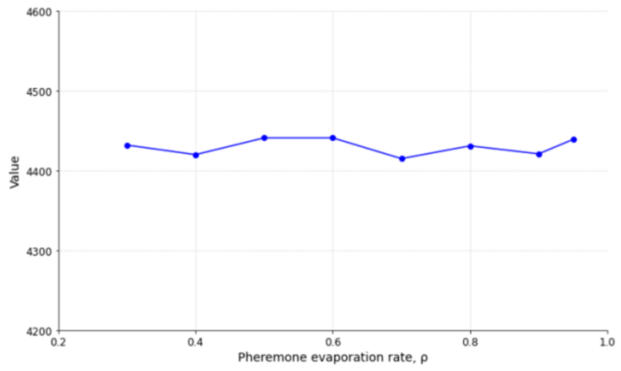

Figure 2. Graph of ρ against maximum values.

Figure 2 shows the variations in pheromone evaporation rate: ρ: 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95. Against the maximum values it produced. Where p = 50, and m = 4000.

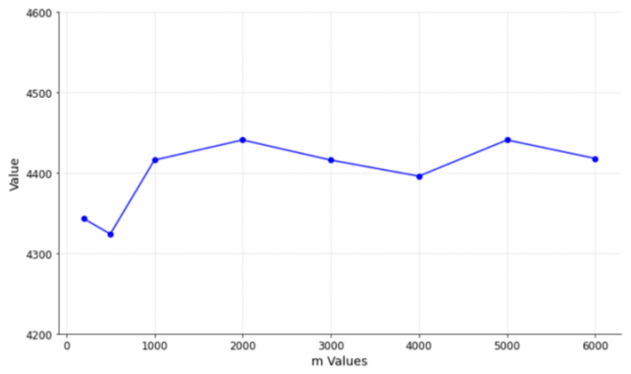

Figure 3. Graph of m against maximum values.

Figure 3 shows the variations in m values: 200, 500, 1000, 2000, 3000, 4000, 5000, 6000. Against the maximum values it produced. Here, p = 50, and ρ = 0.5.

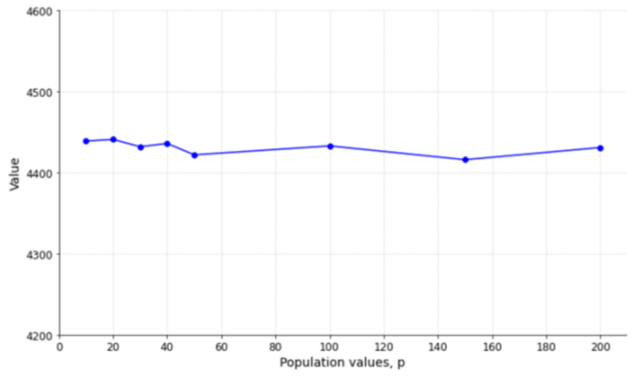

Figure 4. Graph of population against maximum values.

Figure 4 shows the variations in population, p values p: 10, 20, 30, 40, 50, 100, 150, 200. Against the maximum values it produced. Where ρ = 0.5 and m = 4000.

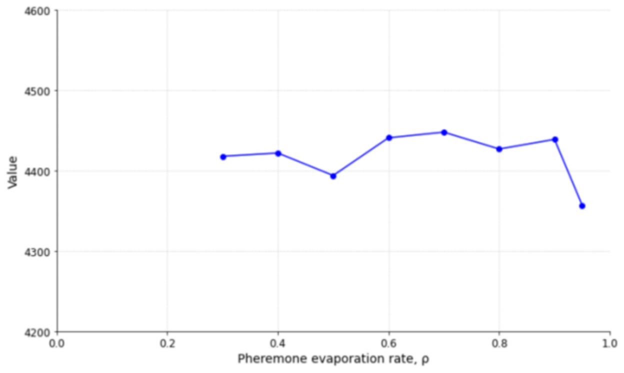

Figure 5. Second graph of ρ against maximum values

In figure 5, m is kept at 2000, p = 50. The values for pheromone evaporation rate are varied ρ: 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95.

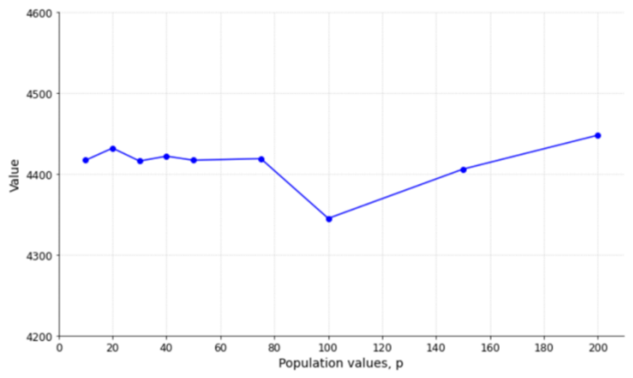

Figure 6. Second graph of population against maximum values.

In figure 6, m is kept constant at 2000, ρ = 0.5. The values for ant population are p: 10, 20, 30, 40, 50, 100, 150, 200.

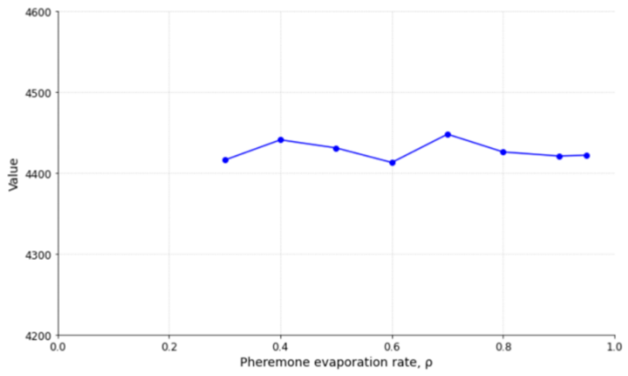

Figure 7. Second graph of population against maximum values.

In figure 7, population p is set at 10, m is set at 4000, and pheromone evaporation rate is varied, ρ: 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95.

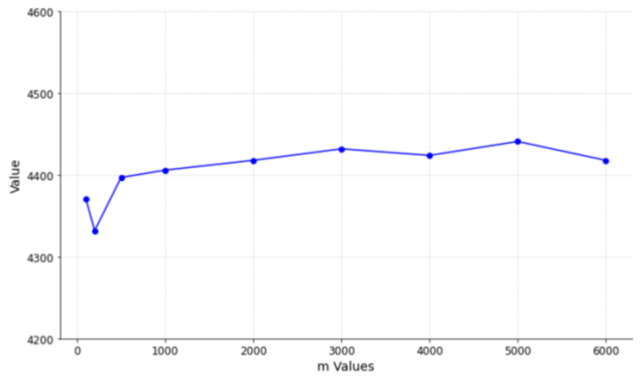

Figure 8. Second graph of m against maximum values.

In figure 8, population p is set at 10, ρ = 0.5, and m is varied, m: 100, 200, 500, 1000, 2000, 3000, 4000, 5000, 6000.

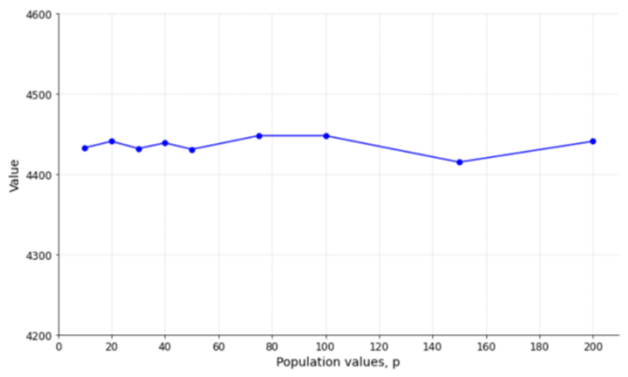

Figure 9. Third graph of population against maximum values.

In figure 9, ρ is set at 0.3, m = 4000. Population is varied as p: 10, 20, 30, 40, 50, 100, 150, 200.

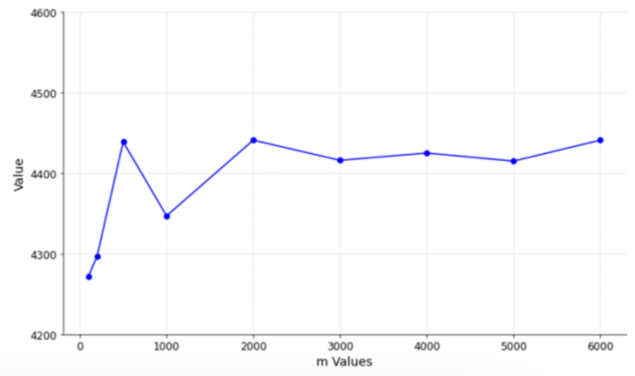

Figure 10. Third graph of m against maximum values.

In figure 10, ρ is set at 0.3, p = 50, and m is varied, m: 100, 200, 500, 1000, 2000, 3000, 4000, 5000, 6000.

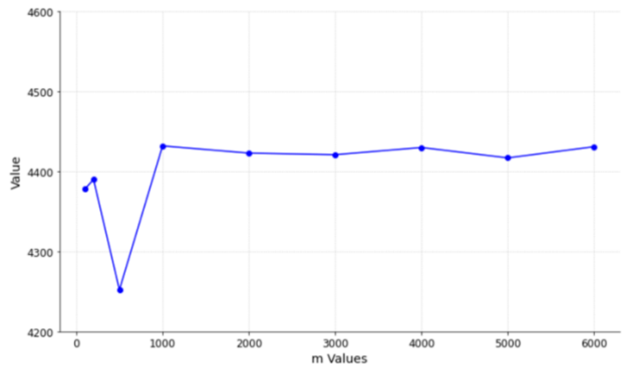

Figure 11. Fourth graph of m against maximum values.

In figure 11, ρ is set at 0.95, p = 50, and m is varied m: 100, 200, 500, 1000, 2000, 3000, 4000, 5000, 6000.

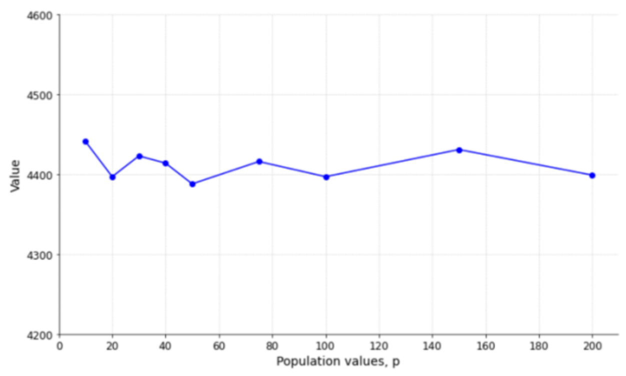

Figure 12. Fourth graph of population values against maximum values.

In figure 12, ρ is set at 0.95, m = 4000, and p is varied p: 10, 20, 30, 40, 50, 100, 150, 200.

5. DISCUSSION AND FURTHER WORK

The best results came from figure 9 where the value for pheromone evaporation rate was set at 0.3, the m value was set at 4000, and the variable on the x-axis was population. The mean of values in these set of results is 4436. The best result(s) were 4448. The ant population values were quite low (10, 20, 30) for the first few sets of results, and their values were very close to the values where the ant population was greater (100, 150, 200). This indicates that in this scenario, the ants converge on good solutions very quicky, and that a larger ant population doesn't allow the ants to explore higher value solutions.

The second-best set of results came from figure 4. Here the mean for the values produced is 4431. In this set of results the pheromone evaporation rate was set at 0.5, and the m value was set at 4000, and the population varies along the x-axis. The results were all very similar to the results in figure 9. This indicates that the values don't have any noticeable difference when the pheromone evaporation is reduced from 0.5 to 0.3.

The reason for the results being so high for figure 4 and 9 is because the algorithm makes it very easy for the ants to find a solution close to the local maximum. This is because for every set of ten bags, the ant can quite quickly choose either the best set of seven to pick, or they can pick a set that is close to the best set of seven. The pheromone evaporation rate was small, 0.3 and 0.5 in the case of figure 9 and 4. Which means that the pheromone values from the unused paths don't become so small that the ants are almost completely prevented from exploring new solutions. So, if the ants at first did not find the best solution, they could very quickly explore a new solution.

The setting for population had the smallest effect on the performance of the algorithm. This is because the algorithm lets the ants converge on good solution without the need for iterating through potential solutions many times.

The parameters $\alpha = 1$ and $\beta = 2$ were kept constant throughout all the tests in the figures above.

The value of m had a large impact on the quality of the solutions the ants created. In figures 3, 8, 10, and 11, we can see that when the value of m was greatly reduced, the solution value was decreased by a large amount. This is because the very low m value means that first few solutions created by the ant will have a very large impact on the pheromone paths. If the first few solutions have a low value, despite being low value, due to the small value of m, they will make it highly likely that the ants converge on a similar solution. If the solution that the ants converge on straight away is of low value, then the small value of m will have been what allowed the algorithm to produce such a poor solution.

In general, the pheromone evaporation rate did not have a large effect on the value of the solutions, this is shown in figure 2, 5, and 7 where the lines produced in the graphs are relatively flat. This is due to the fact that the algorithm had a tendency to converge very quickly on a solution, and so the pheromone evaporation rate was not as impactful on the solution as the other parameters that determined the early convergence of a solution.

The parameter with the biggest influence was the parameter that determined how many times the code was run with a set number of ants. This is what ensured that the values we got were consistently good. Even if a lot of the time the different populations of ants would converge on values that weren't as good, after a large number of run throughs, there would eventually be a set of ants that would converge on a good solution.

The algorithm had a tendency to converge on a solution quickly using the heuristic value mainly to decide on what bags to use. Therefore, the method used by (Schiff, 2014) of using the ratio of the square of the value to the square of the weight for the heuristic value would probably not be as effective since it would further increase the algorithms tendency to converge on a solution prematurely as the heuristic value is far greater.

6. CONCLUSIONS

The proposed algorithm works well at very quickly finding a good solution. The population of ants used did not need to be very large at all for the ants to converge on what is very close to a local maximum.

The main limitation of the algorithm was its structure forced the ants to converge to a local maximum and made them miss out on the global maximum. The justification for splitting the bags into ten different sets that are treated as the travelling salesperson problem was that it would allow the algorithm to converge on a good solution without having to use an excessively large number of ants. As expected, the ants did manage to converge on a good solution quickly. The algorithms' structure limited the number of possible combinations of bags too greatly, which is was constrained the maximum values so much.

A further experiment to improve on the results attained by the algorithm in this paper would be to use four sets of 25 bags instead of ten sets of ten bags. If the algorithm instead split the bags into four sets of 25 bags, then the number of potential solutions would be far greater, and the best possible solution that could be found by the algorithm would be better. It would mean that the algorithm would take slightly longer to converge on a solution, however, it would be worthwhile for the better solutions. Using this architecture would allow the algorithm to get far closer to the global maximum.

7. REFERENCES

[1] Schiff, K. (2014). Ant colony optimization algorithm for the 0-1 knapsack problem. 2013, pp.39–52. doi:https://doi.org/10.4467/2353737xct.14.056.3964.