Bin-Packing with Ant Colony Optimisation

1 Introduction

The bin packing problem (BPP) is defined as an optimisation problem where there are different items of weight, and they are to be most efficiently packed into a number of bins of a certain capacity. Computationally, the problem is NP-hard and the corresponding decision problem. The implementation in this report is of the Ant Colony Optimisation applied to two BPP's that will be evaluated and discussed. In this report I will describe my observations of the results with explanations. In addition, I will describe any further experiments I conducted to better answer questions 1-3 above and to demonstrate understanding of the parameters used in the Ant Colony Optimisation (ACO). Each experiment is listed below:

- Experiment 1: Run five trials of the ACO with p = 100 and e = 0.90
- Experiment 2: Run five trials of the ACO with p = 100 and e = 0.50
- Experiment 3: Run five trials of the ACO with p = 10, and e = 0.90
- Experiment 4: Run five trials of the ACO with p = 10, and e = 0.50

2 Evaluation Results for BPP1

For BPP1, there are 10 bins and the weight of item i will be i. Each experiment had 5 trials, and for each, the minimum best fitness, maximum worst and average across all five are calculated and displayed in the Table below.

Experiment	Min Best Fitness	Average Fitness	Max Worst Fitness
1	1225	3881	8384
2	2016	5380	10348
3	1306	3080	5119
4	225	683	867

As seen in the Table above, the results of the ACO performs best for this problem using p = 10, e = 0.50 (Experiment 4). In addition, experiment 4 had the best average fitness, showing strong consistency. The worst performing parameters were p = 100, e = 0.5 (Experiment 2), with the highest average and minimum best fitness.

3 Evaluation Results for BPP2

For BPP2, there are 50 bins and the weight of item i will be i^2. Each experiment had 5 trials, and for each, the minimum best fitness, maximum worst and average across all five are calculated and displayed in the Table below.

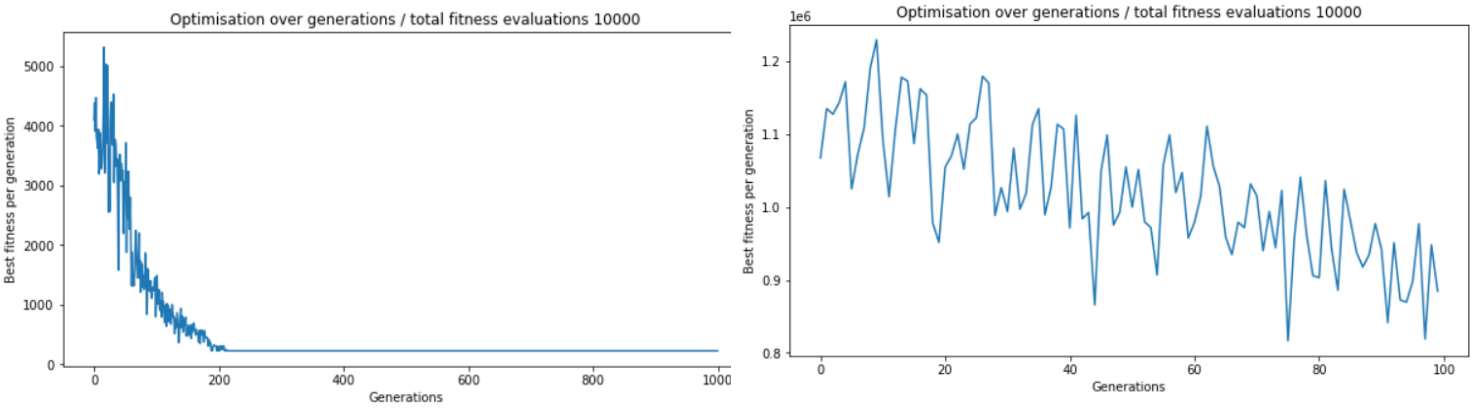
Experiment	Min Best Fitness	Average Fitness	Max Worst Fitness
1	829675	1252083	1952430
2	948067	1476125	2347116
3	972337	1190874	1571185
4	937403	1117936	1266600

As seen in the Table above, the results of the ACO performs best for this problem using p = 100, e = 0.90 (Experiment 1). While Experiment 4 had the best average fitness, showing strong consistency in the result. The worst performing parameters were p = 100, e = 0.5 (Experiment 2), with the highest average fitness.

4 Analysis

To answer the following questions, I will incorporate the visualisations of the best fitness plotted against the generations throughout the individual trials.

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



(a) A trial for Experiment 4 applied to BPP1 (b) A trial for Experiment 1 applied to BPP2

Figure 1: Trials of best performing experiments per BPP 1 and 2

A trial for Experiment 4 applied to BPP1 gave the best result with p = 10, and e = 0.50.
A trial for Experiment 1 applied to BPP2 gave the best result with p = 100 and e = 0.90.

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

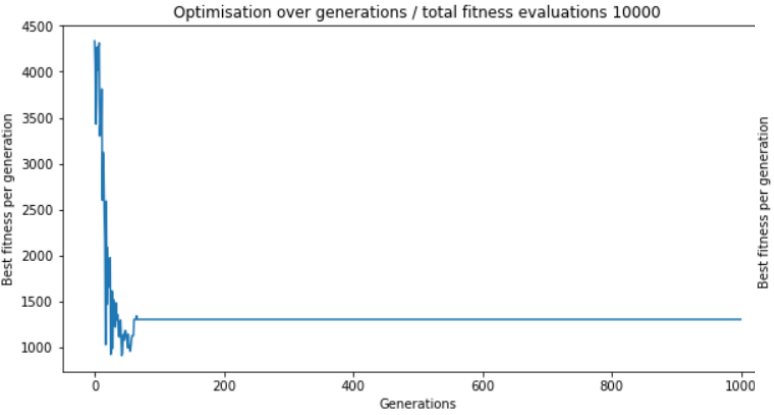
BPP1

It's important to note that BPP1 is an easier problem with 10 bins and items with weights of a smaller range compared to BPP2. From Figure 1, it is clear that the algorithm stopped optimising around the 200th generation. With a population of 10 ants, this allowed 1000 generations to optimise fully and was sufficient in size to traverse the construction graph as this depends on the number of bins. With an evaporation rate of 0.5, the generations could be improved upon from one generation to another without forgetting the paths by evaporating too quickly, such as 0.9. 0.9 evaporation rate often lead to sub-optimal solutions by not continually improving from one generation to the next. These sub-optimal solutions for Experiment 3 can be seen in Figure 2 below:

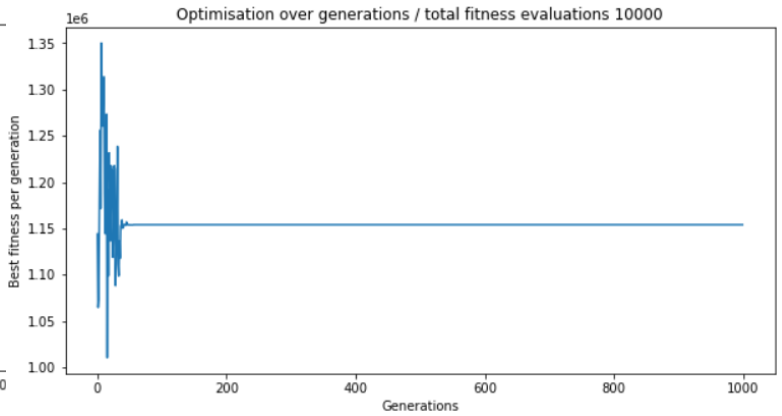
BPP2

It's important to note that BPP2 is a more complex problem with 50 bins and items with weights of an exponentially bigger range compared to BBP1. From Figure 1, it is clear that the algorithm did not stop optimising and probably could have done with more generations. With a population of 100 ants, this allowed for 100 generations to traverse as best possible. Still, with the evaporation rate of 0.9, this feels like a shotgun approach where the highly volatile graph suggests that more ants will attempt random paths as the evaporation rate probably isn't maintaining much memory, but enough for gradual improvement. From looking at the average fitness, the best performing and therefore most consistent is Experiment 4 with p = 10 and e = 0.50. I believe this shows that experiment 4 is still the best at finding the local optimal, but more ants are needed to traverse the graph fully. An evaporation

rate of 0.5 is still sufficient but may require more generations as it is less volatile from one generation to the next.



(a) A trial for Experiment 3 applied to BPP1



(b) A trial for Experiment 3 applied to BPP2

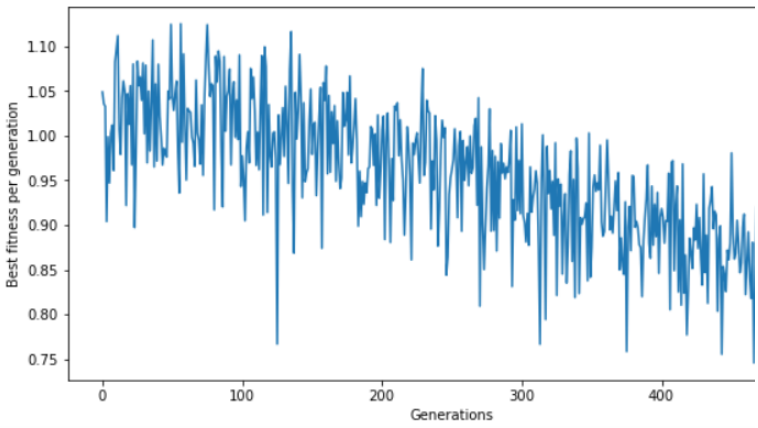
Figure 2: Trials of sub-optimal Experiment 3's per BBP 1 and 2: with p = 10, and e = 0.90

Figure 2 shows the impact of an evaporation rate that is too great to stick with a current best path and improve from one generation to the next. Another observation can be made that after less than 100 generations, the 10 ants and high evaporation rate get stuck at a sub-optimal path, unable to improve from the current best path. This early cut performance is most likely due to the 0.9 evaporation rate resulting in the colony being unable to benefit from being a colony and unable to communicate through pheromones. The biggest difference between Figure 2(a) and Figure 2(b) is that the 10 ants find it harder to traverse the whole construction graph when it is bigger (fig 2(b)), and therefore more difficult for the ant colony to retain its grasp on the best result.

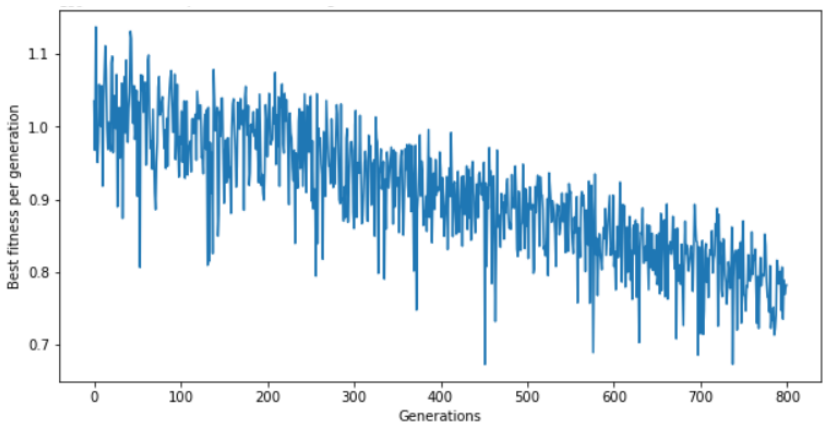
Question 3: How do each of the parameter settings influence the performance of the algorithm?

How the e and p parameter settings influence, the performance was explored and explained in Question 2. But the remainder of this question will be showing my findings through further experimentation to see what other parameters can significantly influence the performance both in time and optimisation result. Focusing on BPP2 as this is still yet to be fully optimised, and with time, the reduction in fitness evaluations is always the way to go while trying to maintain consistent average fitness results.

For BPP2 a two new better fitness were found, with e = 0.50 and p = 300 and then e = 0.50 and p = 400, but also a modification on the max 10000 fitness evaluations had to be made to allow for more generations to run and try and best solve this problem. As seen below in Figure 3, now more closely resembling the optimisation seen in Figure 1 BPP1. Although this trade-off was significantly impacted in performance regarding time, requiring 150000 (300*500) and 320000 (400*800) fitness evaluations, respectively, directly correlates to the amount of time to run the ACO on BPP2. It is now easy to infer with a new and improved best fitness of 782244 for BPP2 that with more generations, this configuration could lead to a further enhanced result; however, this does not seem feasible with this size of BPP as it takes far too long during run time. More bins correlated to performance longer in time as this exponential increases the problem's computational complexity. Generations and population size had an increasing effect on computational complexity by increasing the number of fitness evaluations, again taking longer to complete, but depending on the problem, the parameters for the population of ants and evaporation rate greatly affected the quality of the ant colony optimisation (ACO).



(a) A trial for Further Experimentation applied to BPP2: e = 0.5 and p = 300 and generations = 500



(b) A trial for Further Experimentation applied to BPP2: e = 0.5 and p = 400 and generations = 800

Figure 3: Optimisation on ACO BPP2 to prove hypothesis on parameters: Best fitness's 814700 and 782244 respectively

Question 4: Do you think that one of the algorithms in your literature review might have provided better results? Explain your answer

As suggested, the proposed solution implemented can perform well on smaller BPPs, but the larger the problem becomes less and less feasible. The literature suggests many ways to consider improving upon the Ant colony optimisation. Still, the best approach both in time and optimisation capability is the HGGA [1], and Falkenauer combines genetic algorithms with local search and is still one of the best methods when handling all sizes of BPP.

A way in the literature presented by Martello and Toth [2] suggests that in conjunction with the current implementation that combining a local optimisation speeds up the overall performance. Using the ACO as a global search and then locally optimising local ant paths once distributed enough.

The most sophisticated solution has shown up trying to solve a 3D-BPP which would add a lot more complexity to the problem and solution space but still manage to solve in an acceptable amount of time. [3] This is solved with a hybrid-genetic algorithm called Improved Deepest Bottom Left with Fill (I-DBLF).

A hybrid genetic algorithm rather than an Ant Colony Optimisation has been explored and would provide a better result by introducing a meta-heuristic and correcting infeasible chromosomes. [4] This led to significantly reduced execution time. While also allowing for cost efficiencies in the real-world solution that the BPP was applied to for packing Virtual machines in servers.

Conclusion

An ACO approach for BPP was presented in this paper and parameters were disucussed on the impact and how to choose parameters for performance and trade-offs. Now with understanding of how a larger BPP may not feasible for ACO alone and possibly using a local search or genetic algorithm can drastically improve the optimisation as the surrounding literature suggests.

References

[1] Falkenauer, E. and Delchambre, A., 1992, May. A genetic algorithm for bin packing and line balancing. In ICRA (pp. 1186-1192).

[2] Martello, S. and Toth, P., 1990. Bin-packing problem. Knapsack problems: Algorithms and computer implementations, pp.221-245.

[3] Kang, K., Moon, I. and Wang, H., 2012. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. Applied Mathematics and Computation, 219(3), pp.1287-1299.

[4] Kaaouache, M.A. and Bouamama, S., 2015. Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud. Procedia Computer Science, 60, pp.1061-1069.