**Bin Packing with Ant Colony Optimisation**

**Candidate Number: 099562**

**ABSTRACT**

The aim of this paper is to review different nature-inspired approaches to the bin packing problem. Then this paper will outline the ant colony optimisation algorithm and apply it to the bin packing problem. The results from the experiments will be used in order to draw conclusions on how the parameters affect the algorithm and results.

# INTRODUCTION

The bin packing problem is defined as a finite set of values O which represent the item sizes and two constants C and N where C is the bins capacity and N is the number of bins. The problem asks: is it possible to pack all the items into N bins. We can further define the specifics of the problem; is there an arrangement of O into N or fewer subsets, such that the sum of elements in any subset does not exceed C.[1]

# Nature-Inspired Algorithms

Here this paper looks at a few different algorithms that have been inspired by nature. Knowing how the algorithms work generally is useful but in this section, this paper will also focus on how the algorithms adapt to represent the bin packing problem.

First fit decreasing (FFD) involves ordering the items in descending order and fitting them in the bins in chronological order, if the item doesn't fit in the current bin it tries the next bin. FFD yields very good results and is not a nature-inspired algorithm. Therefore FFD is generally accepted as the benchmark for testing how good algorithms are at bin packing.[8]

## Genetic Algorithm

The genetic algorithm can be generally outlined as follows: whilst a stopping criterion is not met, choose pairs for mating based on the fitness function, then perform a cross-over to generate off-springs, then evaluate the fitness of the new off-springs and finally generate a new population. This loops through until a stopping criterion is met.[2][3]

A key element of genetic algorithms is the chromosomes, the name is borrowed from Genetics but in this case, it is a coded representation of a solution. A solution is an n-dimensional array, each component of which is termed a gene. Three different representation schemes for the bin packing problem include: bin based representation, object-based representation and group-based representation [3].

In bin based representation, a chromosome has a fixed length n which is the number of objects. Each bin is represented by a gene. The position of each gene in a chromosome then indicates the object number placed in the bin represented by that gene. If a chromosome 2 3 1 4 5 exists, it means that object 1 goes in bin 2, object 2 goes in bin 3, object 3 goes into bin 1 etc. The disadvantage of this representation is that some solutions will be infeasible, and some others will be redundant.[3]

In object-based representation, a chromosome represents a permutation of objects. Then the chromosome is partitioned based on the number of objects placed in bins. For example given the chromosome 453261 then this can be partitioned like 45|32|61 this shows that in bin 1 there is 4 and 5, in bin 2 there is 3 and 2 and in bin 3 there is 6 and 1. There is a disadvantage for this representation in that different permutations can have the same solutions because the partitions can create the same groups. For example, 123|45|6 is the same as 45|123|6.[3]

The group-based representation has two parts, first being similar to bin based representation so for a chromosome 4 5 2 3 2 1 means that object 1 is placed in bin 4 and object 2 is placed in bin 5 etc. Then the second part the bins are given letters so, for instance, DEBCBA and the letters are given where A= {5} B= {3, 5} C= {4} D = {1}, E= {2}. One advantage of this representation scheme is that genes represent both objects and groups. The group part is used in genetic operations while the object part is used to identify elements of a group. The disadvantage is that chromosomes have a variable length.[3]

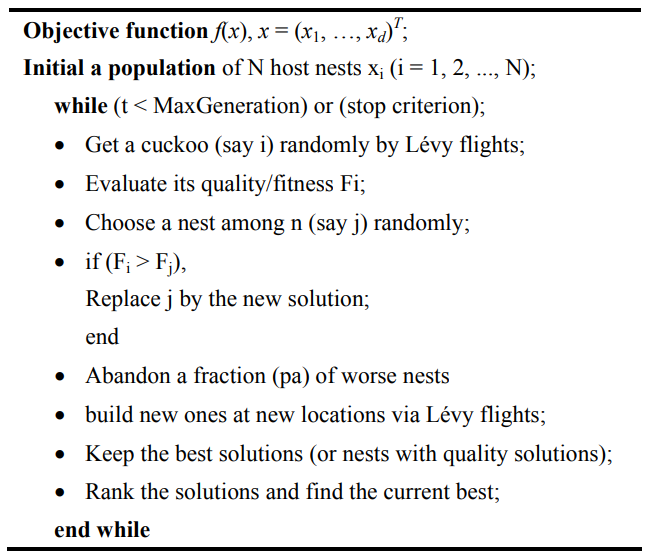
The genetic algorithm, when compared to the first-fit decreasing algorithm, is superior and can outperform it in what is described as ‘tough conditions’. [8]

**2.2 Cuckoo Search Algorithm**

The Cuckoo Search Algorithm was developed in 2009 by Xin-she Yang and Suash Deb. The algorithm is inspired by the routine life of cuckoo birds. The birds tend to lay their eggs in another bird species nest, this is called brood parasitism. Once the cuckoo chick hatches it removes all other eggs out of the nest.[4]

Cuckoo search is based on three rules: the first rule is that each cuckoo lays just one egg and randomly chooses a nest. the second rule is the best nests with the highest quality eggs carries over to the next generations. Finally, the number of existing host nests is fixed and each bird host can find out the alien cuckoo egg with a probability of Pd. In this case, the host bird can either throw the cuckoo egg away or build a new nest in a new location.[4]

The pseudo-code can be seen in figure 1:



**Figure 1: The pseudo-code for Cuckoo Search[5]. For bin packing problem the quality/fitness would be the difference in weights between the largest and smallest bin. The cuckoo’s egg will be potential solutions to the problem.**

Results of the cuckoo algorithm when there’s a 100000 limit capacity of bins outperforms the FFD on the same item set[7]. For item sets with fewer items, the solutions match for both algorithms.

**2.3 Ant Colony Optimisation**

The first Ant Colony Optimisation algorithm was applied to a travelling salesperson problem in 1992 by Dorigo. The original algorithm has since been improved and applied to different problems.[6]

Ant colony optimisation algorithms were inspired by the ant’s ability to find the shortest path between their nest and a food source. They can do this by leaving a trail of pheromones wherever they walk. Other ants can smell the pheromone and follow it. When an ant has an option initially of two paths is chooses randomly between the two. The ants which run on the shortest path will go to the food and back to the nest faster. After they return to the nest there will be more pheromone on the shortest path. The initial random choice of two paths is then biased towards the path with more pheromones influencing more ants to take that path. After some time the entire colony will take the shortest path. [6]

The adaptation of the ant colony optimisation is explained in 3.1. The effectiveness of this algorithm can be compared with Falkenauer’s Hybrid Group Genetic Algorithm (HGGA). This is a variation of the genetic algorithm. The performance of ant colony optimisation is worse with big problems and similar with smaller problems. If the problem involves many items of large weights the genetic algorithm is favourable, else either gets the same result so it does not matter which you use.[6]

**3.1 Ant Colony Optimisation Algorithm Method Used**

The method for which the Ant Colony Optimisation Algorithm is as follows:

The first step is to create an initial construction graph of random pheromones. The pheromones, in this case, are random values between 0 and 1.

The second step involves generating a set of paths. Each ant traversing the graph is one path and so in the algorithm, a list of bin numbers is used to represent a path. for example: [2,3,2,4,5] means the first item is put in bin 2, the second item is put in bin 3 and so on.

Then once the set of paths have been made the pheromone table must be updated. For every path a fitness is calculated, this is the difference in weights of the biggest bin and the smallest bin. In this algorithm to update the table, for each path add 100/fitness to the corresponding nodes in the pheromone table.

Next evaporation takes place, this involves multiplying every pheromone by a value between 0 and 1 specified as the evaporation rate.

The stopping criteria for this algorithm is 10000 fitness evaluations. Until that number is met, return to step 2. The number of fitness evaluations made each time depends upon how many ants you choose to use. With just 10 paths this will loop 1000 times before stopping.

The solution returned at the end may not be the best solution found in the entire 10000 evaluations but its the best solution found out of the number of ants you specify in that loop.

**3.2 Results**

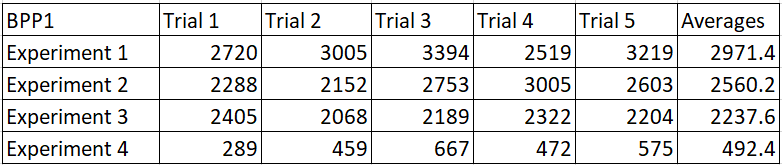
For Bin Packing Problem 1 (BPP1) there were 4 experiments to attempt using the ant colony optimisation. For each experiment, the number of bins remained the same at 10, there were 500 items of weights 1 to 500. The algorithm ran until the number of fitness evaluations was equal to 10000. The number of ants traversing paths; P, and the evaporation rate is different for each experiment:

Experiment 1: P is set at 100, Evaporation Rate set at 0.9.

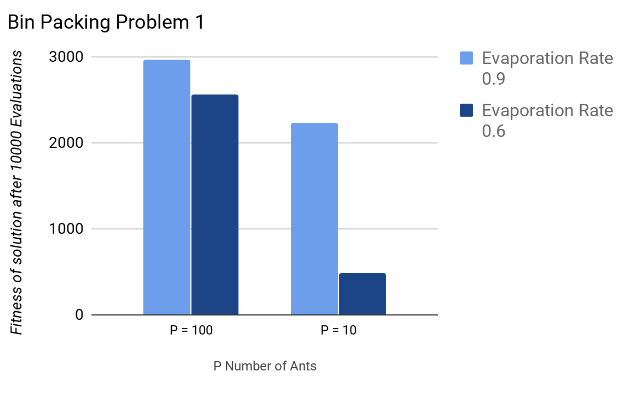
Experiment 2: P is set at 100, Evaporation Rate set at 0.6

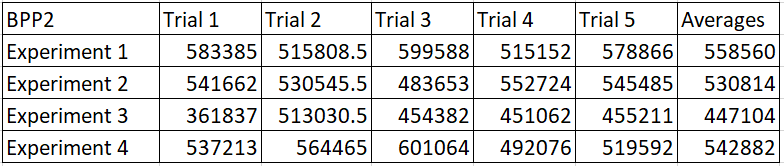
Experiment 3: P is set at 10, Evaporation Rate set at 0.9

Experiment 4: P is set at 10, Evaporation Rate set at 0.6

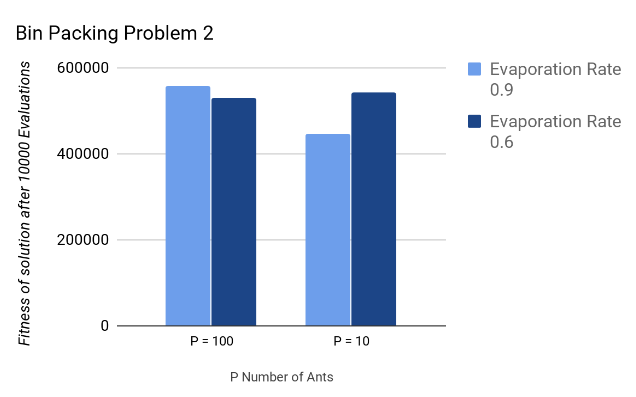
****

This table shows the results of each trial for BPP1, the values in the cells are the fitness of the result given. The averages of each trial were taken and put into a graph see figure 2.

**Figure 2: In this graph are the averages of the trials for Bin Packing Problem 1.**

In Bin Packing Problem 2 (BPP2) the items are different to BPP1. The items are (i^2) /2 for all i where i is numbers 1 to 500. e.g 1,4,9...250000. The number of bins is also changed to 50. 

This table shows the fitness’s of the solution produced by each experiment in BPP2. The averages of each trial were taken and put into a graph see figure 3

**Figure 3: In this graph are the averages for the trials of Bin Packing Problem 2**

**4.1 Discussion and Further Work**

**Question1: Which combination of parameters produces the best results?**

It is clear from both problems that the best parameters will be different depending upon the weight of the items.

For BPP1 we see a clear trend of 0.6 being the better evaporation rate compared to 0.9. We can see that when the number of ants remains the same the best solutions come from the experiments with 0.6 as the evaporation rate. The number of ants also affects the results, out of 100 ants vs 10 ants we see that the 10 ants yield a solution with a lower and therefore better fitness. Thus we can conclude for the BPP1 the parameters for the best results is P=10 where p is the number of ants, and evaporation rate set to 0.6.

For BPP2 we see a similar trend to BPP1. The 10 ants outperformed the 100 ants with exception to experiment 4. 0.6 evaporation rate outperformed 0.9 with the exception again to experiment 4. For this reason, the best parameters in this problem are 10 ants with an evaporation rate of 0.9

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

We see in both BPP1 and BPP2 that using 10 ants is better than 100, this is because the algorithm stops after 10000 fitness evaluations. With 10 ants the pheromones are evaporated 1000 times whereas with 100 ants the pheromones are evaporated 100 times, this makes the probability of bad nodes in the construction graph much less due to the evaporation rate.

In BPP1 we see the 0.6 evaporation rate perform better than the 0.9. This is because 0.6 effectively eliminates the nodes that have a poor pheromone score faster so they are traversed less. 0.9 isn't as effective because it just converges too slow on for these items.

The best parameters differ in BPP2 and that can be explained. First of all, in bin packing problem 1 the items are relatively similar with the biggest difference being 499 (between 1 and 500). In bin packing problem 2 the biggest difference is 249999, this means the pheromone updates will be much smaller in BPP2 as the update is always 100/fitness.

Having a more drastic evaporation rate can be beneficial, work and eliminate certain paths that are less efficient, however with the big data set like in BPP2 the 0.6 converges too fast and so the majority of the time it will get bad results. It is beneficial for big data sets to have a less drastic evaporation rate like 0.9. The results in the experiment reflect this.

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

The number of ants does not affect the time it takes to run the entire algorithm because the algorithm used is based on fitness evaluations, and so no matter how many ants are used the same number of evaluations are done. The ants do however affect the performance in terms of solutions, from the experiment we see that the fewer ants used are yield a better solution because the fitness evaluations are fixed for all experiments.

The bins used affect the timing of the algorithm, the more bins the longer it takes to run, this is because there are more probabilities to calculate for each ant path. The number of fitness evaluations also affects the timing, the more fitness evaluations the longer the algorithm takes to run. With more fitness evaluations the solution improves as a general trend, this occurs until a local optimum has been reached and the solution will stay the same.

Changing the evaporation rate although would not affect the speed of the algorithm, it will get differing results and so it is possible to find an optimum rate for different items. To improve the performance in cases of a big data set for items a less drastic evaporation rate gets a better solution (for example 0.9 is less drastic than 0.6)

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

For the ant colony optimisation algorithm that was used in the experiments, the main limiting factor to getting better results was the limit of 10000 fitness evaluations. Increasing this parameter will gain better results

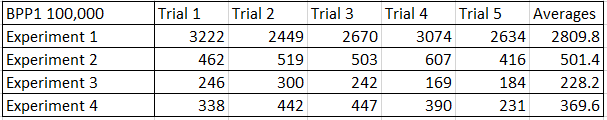
In the literature review, the algorithms were compared to first fit decreasing algorithm, however, in this case, it cannot be done. This is because the bins have no maximum capacity so all the items will be put in one bin for FFD.

Ant colony optimisation or the cuckoo algorithm, for this case I think the cuckoo algorithm will perform better. I think this because the cuckoo algorithm keeps the best solution of all time. The ant colony optimisation problem only shows the best solution of the current ant paths.

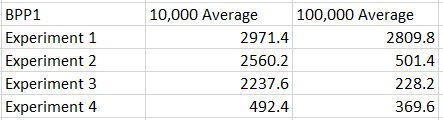
The genetic algorithm given enough generations will converge onto a local optimum. Considering in the experiments the fitness evaluations were limited to 10000, I think the genetic algorithm will outperform it based solely on more time to converge the solution. For experiment 4 of BPP2 I think that the genetic algorithm will perform better. The mutations tend to be small and so it won't converge too fast, unlike the ant colony optimisation example.

**4.2 Further Experimentation:**

From my observations, the 10000 fitness evaluations seemed to be a limiting factor in getting a better solution, this is because the values leading up to the 10000th evaluation showed a trend of the solution improving. To further explore this, I decided to attempt the BPP1 again with 100000 fitness evaluations. If the solutions are better with more fitness evaluations it proves that it is a limiting factor when it comes to getting better solutions.



This table shows the fitness of each experiments solution. The conclusion from these results matches those of BPP2 with 10000 evaluations. Given more evaluations experiment 3 outperforms experiment 4.



This table is a comparison of fitness results for BPP1 with 10,000 and 100,000 fitness evaluations. From comparing the averages of both BPP1 experiments you can see that the 100,000 fitness evaluations experiment performed better in all experiments. I can conclude from these results that given more fitness evaluations, the ant colony optimisation algorithm created gets better results.

**4.3 Conclusions**

A few nature-inspired approaches to the bin packing problem have been presented. Genetic algorithm, Cuckoo Search and Ant Colony Optimisation. The ant colony approach was implemented, and the parameters of the algorithm tested to see how they affect the result. Although it's not the most efficient nature-inspired algorithm, it was useful to discover how depending upon the items of the problem, different parameters give better solutions.

# 

# 

# 5. REFERENCES

[1] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing", *Journal of Heuristics*, vol. 2, no. 1, pp. 5-30, 1996. Available: 10.1007/bf00226291.

[2] E. Burke, M. Hyde, G. Kendall and J. Woodward, "A Genetic Programming Hyper-Heuristic Approach for Evolving 2-D Strip Packing Heuristics", *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 942-958, 2010. Available: 10.1109/tevc.2010.2041061.

[3] M. Nemat, "Application of Genetic Algorithm for the Bin Packing Problem with a New Representation Scheme", Mathematical Sciences Quarterly, vol. 4, 2010. [Accessed 5 November 2019].

[4] R. Yesodha, "A Comparative Study On Heuristic Procedures To Solve Bin Packing Problems", *International Journal in Foundations of Computer Science & Technology*, vol. 2, no. 6, pp. 37-49, 2012. Available: 10.5121/ijfcst.2012.2603.

[5] A. Gherboudj, A. Layeb and S. Chikhi, "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm", *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, p. 229, 2012. Available: 10.1504/ijbic.2012.048063.

[6] J. Levine and F. Ducatelle, "Ant colony optimization and local search for bin packing and cutting stock problems", *Journal of the Operational Research Society*, vol. 55, no. 7, pp. 705-716, 2004. Available: 10.1057/palgrave.jors.2601771.

[7] A. Layeb and S. Boussalia, "A novel greedy quantum inspired cuckoo search algorithm for variable sized bin packing problem", *International Journal of Mathematics in Operational Research*, vol. 6, no. 6, p. 732, 2014. Available: 10.1504/ijmor.2014.065420.

[8] E. Falkenauer and A. Delchambre, "A Genetic Algorithm for Bin Packing and Line Balancing", 1992. [Accessed 10 November 2019].