Application of Ant Colony Optimisation Algorithm to the Bin Packing Problem

660045148
College of Engineering, Mathematics,
and Physical Sciences
University of Exeter

ABSTRACT

The Bin Packing Problem is a NP-hard problem with many potential applications. Being NP-hard, there is no known optimal algorithm for BPP running in polynomial time. This paper discusses various nature-inspired algorithms that have been applied to the Bin Packing problem and analyzes results from an Ant Colony Optimisation (ACO).

Keywords

Bin Packing problem, Nature-Inspired algorithms, Ant Colony Optimisation

1. INTRODUCTION

The Bin Packing Problem (BPP) has many potential applications including transportation, parallel machine scheduling and capital budgeting [3]. Garey and Johnson [7] defined the one-dimensional BPP as follows: "Given a finite set O of numbers and two constants C and N, is it possible to pack all the items into N bins - that is, does there exist a partition of O into N or less subsets, such that the sum of elements in any of the subsets does not exceed C?".

In this paper, we will discuss various nature-inspired algorithms that have been used to the Bin Packing problem and analyze results from an Ant Colony Optimisation (ACO) implementation. Section 2 presents literature review. Section 3 describes ACO approach. Section 4 provides the results obtained using ACO. Section 5 discusses the results and presents further work. Finally, section 6 concludes the paper.

2. LITERATURE REVIEW

In this section, we present previous research done in the field and provide overview of different algorithms.

2.1 Bin Packing Problem

BPP was shown to be NP-hard in 1979 Garey and Johnson [7] and it is now a well known and researched problem. Initially, a number of heuristics were proposed for solving BPP. Even though BPP can usually be solved using simple heuristics [5], this method becomes deficient for handling large BPPs. Several methods have been proposed based on nature-inspired algorithms, including various order-based and grouping-based genetic algorithms [14], hybrid methods [6, 11] and evolutionary algorithms [3].

2.2 Ant Colony Optimisation

One of the approaches for BPP was proposed by Levine and Ducatelle in 2004 based on ACO [10]. Previously, ACO has been successfully applied for multiple combinatorial optimization problems including Traveling Salesman Problem (TCP), Quadratic Assignment Problem and Graph Coloring Problem [3]. Originally, ACO was inspired by the capability of real ants to find the shortest path between their nest and a food source. It is, therefore, natural that the first use of the ACO was to a path optimization problem- TCP. After the its publication in 1996 [4], this method became very popular and has been improved and used in various research since.

2.3 Other Nature-Inspired Algorithms

One of the most successfully used algorithm for BPP is Hybrid Grouping Genetic Algorithm (HGGA) introduced by E. Falkenauer [6]. The essence of HGGA is forming offspring by choosing a set of bins from each parent solution, excluding duplicate items and putting the remaining items according to the simple First Fit Decreasing (FFD) heuristic.

After the Levine and Ducatelle publication [10], several hybrid Ant Colony algorithms have been proposed. For example, in [8], the authors successfully mixed the previous technique with the Firefly algorithm.

Other Nature inspired algorithms have been proposed, and in 2012, Layeb and Boussalia merged Cuckoo Search (CS) algorithm with a binary quantum technique for solving BPP [9]. Later, in 2016, the authors in [15] also applied CS, using Ranked Order Value and Ford–Fulkerson method for algorithm adaptation.

More recently, the authors in [12], solved BPP with Simulated Annealing in combination of a swap function for solution enhancement. In 2018, a Whale Optimization Algorithm was proposed for BPP named Improved Lévy-based Whale Optimization Algorithm (ILWOA) by Abdel-Basset et al. [1] achieving promising results.

3. METHOD

The key to ant colony's ability to find the shortest path lies in the fact that ants leave a pheromone trail while walking [2]. Other ants can then follow the trail by smelling the left pheromone. Ants are presented with two untraveled paths, they choose their step randomly. Naturally, the ants

using the shortest path will be back faster. Therefore, after their return there will be more pheromone on the shortest path. This will influence other ants to follow this path. In the end, the whole colony follows the shortest path. We utilise this ants behaviour in out implementation. ACO Implementation:

1. Create construction graph. This is done by generating a 3D array of the following size: number of bins b, number of items i, b. Then, we fill the graph with random pheromone values between 0 and 1.
2. Next, the ant paths are generated. Each ant decision is random, but biased by the amount of pheromone on the choices. This is repeated for each item and bin.
3. The pheromone values are updated on the graph according to ant's path fitness. The pheromone update in our implementation is *100/fitness*.
4. Each path in the graph must be evaporated. This is done by multiplying each value in the construction graph by evaporation rate.
5. Unless the set limit of evaluations is reached, return to step 2. In our experiments the set limit is 10,000.

In order to assess the quality of the solutions, the fitness will be calculated by the difference between the heaviest and lightest bins, with the goal to minimize this difference.

4. DESCRIPTION OF RESULTS

Using the approach described in section 3, we have ran a number of tests. We use two different Bin Packing Problems BPP1 and BPP2. In each, there are 500 items. To check how different parameters change the result, the population (p) and evaporation rate (e) were:

1. $p = 100$, $e = 0.9$
2. $p = 100$, $e = 0.6$
3. $p = 10$, $e = 0.9$
4. $p = 10$, $e = 0.6$

For each of the above parameter five full trials were ran, and the best average fitness was selected from these trials. The results of these runs are presented in Tables 1 and 2.

4.1 BPP1

In BPP1, 10 bins are given and the item weights start at 1 and finish at 500.

Table 1: BPP1 results

	Fitness			Time
Parameters	AVG	MAX	MIN	AVG
p=100, e=0.9	2520	2973	1942	10.6
p=100, e=0.6	2457	2777	2024	10.7
p=10, e=0.9	1338	1688	1008	10.6
p=10, e=0.6	1243	1959	872	10.4

As can be seen in Table 1, the results of the application improve with reduction of population and evaporation rate. Using pheromone evaporation helps eliminate bad solutions from previous environments. However, too high evaporation rate can result in the loss of good paths as well. Therefore in BPP1, reducing evaporation rate and population make the solution better.

The best fitness was achieved using $p = 10$, $e = 0.6$, the worst using $p = 100$, $e = 0.9$. In our implementation, each trial took just over 10 seconds to complete.

It is also interesting to observe the difference between the worse and the best results during each test. The max and min differences are 1031, 753, 680 and 1087 accordingly. This might indicate how stable the results are when using specific parameters.

4.2 BPP2

In BPP2, 50 bins are to be used to pack items with weights $\frac{i^2}{2}$, where i ranges between 1 and 500.

Table 2: BPP2 results

	Fitness			Time
Parameters	AVG	MAX	MIN	AVG
p=100, e=0.9	574818	595771	554872	21.1
p=100, e=0.6	516030	535807	490668	20.7
p=10, e=0.9	491945	524425	427161	21.2
p=10, e=0.6	597238	628951	564010	21.1

In BPP2, decreasing the population improves the result, however, decreasing both the evaporation rate and the ant population leads to worse fitness. Since there are only 10 ants and 50 bins there is a big chance that the ants would not explore enough paths, therefore failing to find the optimal solution. Lower evaporation rate makes it even less likely that the ants would improve their route, staying in the same bad path without making it better.

The best fitness here was achieved using $p = 10$, $e = 0.9$, and the worst with $p = 10$, $e = 0.6$. Each trial took approximately 21 seconds when the number of bins was 50. From results provided in Table 2, the difference between the maximum and minimum fitnesses are 40899, 45138, 97264, 64941 correspondingly.

4.3 Further tests and results

In order to visualise the progress during different ant generations, bellow graphs illustrate how average fitness results improve with each generation. Figures 1 and 2 illustrate a single BPP1 trial.

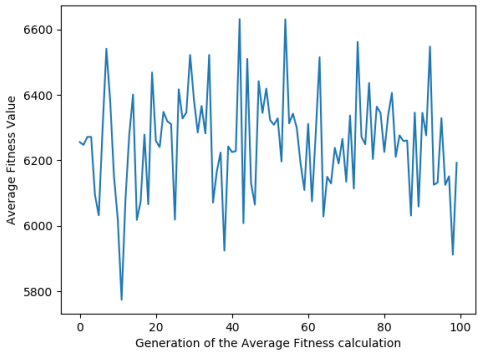

Figure 1: BPP1 trial with e = 0.9 and p = 100. Maps Average fitness values to ants generation.

As per Figure 1, high evaporation rate and population result in almost no improvement in each generation. The ants seem to be exploring the construction graph mostly

at random. Due to high evaporation rate, the best fitness achieved around 16th generation was lost and the rest results were found to be worse.

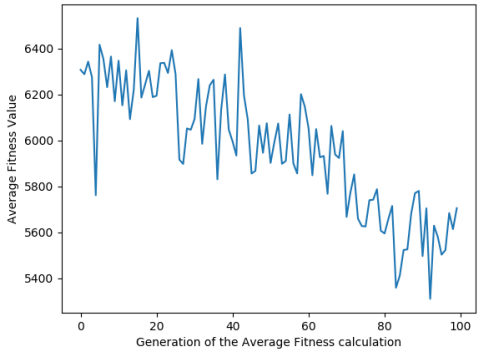

Figure 2: BPP1 trial with e = 0.6 and p = 100. Maps Average fitness values to ants generation.

In Figure 2 an improvement can be seen, however the change is not very significant.

Next, we ran the same program to illustrate how lower population affects results. This is shown in Figures 5 and 4, where population of 10 is used to run a trial.

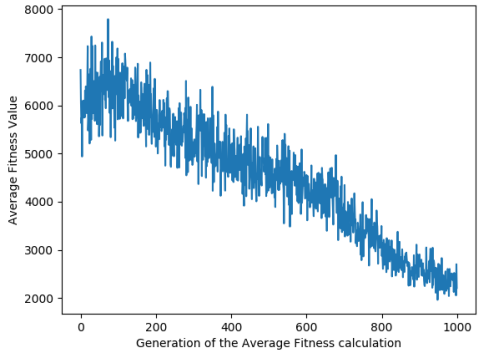

Figure 3: BPP1 trial with with e = 0.9 and p = 10. Maps Average fitness values to ants generation.

Figure 5 proves that lesser population achieves better results in BPP1, as overall, the fitness improves with generations, even with evaporation rate of 0.9.

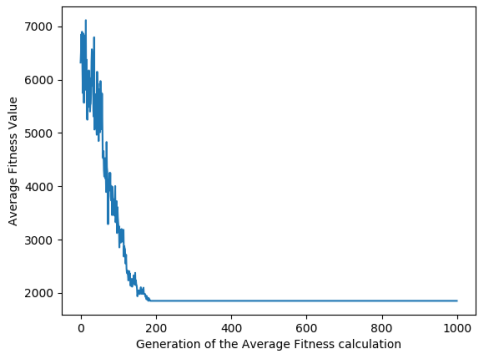

Figure 4: BPP1 trial with e = 0.6 and p = 10. Maps Average fitness values to ants generation.

Lowering evaporation rate as well as population proves to perform better, as per figure 4. This trial achieves better fitness and found it approximately 800 generations earlier.

It is important to note that with the limit of 10000 fitness evaluations, using 10 ants results in 1000 generation, as apposed to 100 generations when we use 100 ants. This is due to each ant having its own path during each generation, meaning number of generations are *the limit* divided by *number of ants.*

5. DISCUSSION AND FURTHER WORK

In this section, we will discuss the results provided in section 4. Performance of the algorithm will also be reviewed, together with potential further improvements.

5.1 Discussion of the best parameters

For BPP1, the best fitness was achieved using population of 10 and evaporation rate of 0.6. Since there were only 10 bins to fill in, using less ants and smaller evaporation achieves best result. The reason smaller population perform better, is less ants result in more generations. This means that the ants are given the chance to improve their decision bigger number of times. This can be clearly seen when comparing figures 1 and 5. If we only consider the first 100 generation in figure 5, the results would be very similar and even worse than in figure 1, where 100 ants are used. But since the 10 ants in figure 5 have more generations, the results eventually improve and outperform the 100 ants by 200th generation.

The evaporation rate is also very important. Comparing Figures 1 and 2, and Figures 5 and 4, there is a definite trend of the ants finding better results quicker with smaller evaporation. This is especially evident with population of 10 and evaporation rate of 0.6. As mentioned in the previous section, pheromone evaporation allows eliminate worse solutions from previous generations. It is important to find the right evaporation rate for each individual problem, as high evaporation rate can result in loss of optimal solutions. Therefore, having lesser rate allows the ants to improve their previous paths, as apposed to starting new path each time. This results in better performance and finding the solution with less generations.

For BPP2, however, the best fitness was achieved using population of 10 and evaporation rate of 0.9. Similar graphs as in figures 1-4 were generated for BPP2. The results were similar apart from the value range. Based on this, one can conclude that the behaviour of ants are similar in both problems. However, in BPP1, 10 ants with evaporation rate 0.6 found the best result. The similarity in the results generated by these parameters are in the shape of the graphs (see figure 5). In both the cases ants stop improving their paths by approximately 200th generation. In BPP2, however, this is not enough to find the optimal solution with 50 bins. Even though reducing evaporation worsen the results, lower population size improved fitness. This is likely for the same reason as BPP1. Additionally, the number of possible paths found by 100 ants could also mislead the choice, making it more challenging to find the best solution.

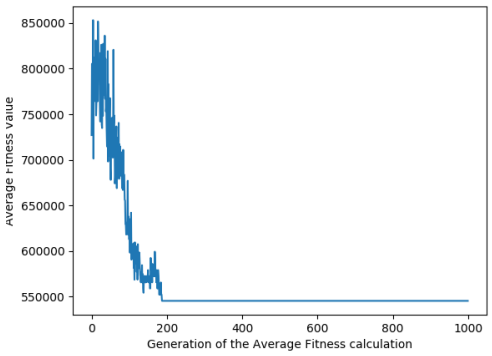

Figure 5: BPP2 trial with with e = 0.6 and p = 10. Maps Average fitness values to ants generation.

5.2 Performance of the algorithm

Changing bin size for the problem greatly influences the performance of the algorithm. Other parameters that greatly affected the computational time and quality of the results were bin capacity, number of items and their weight. In BPP1 each trial took approximately 11 seconds to run, in contrast to 21 seconds in BPP2. This shows that having more bins greatly increase the computational complexity of the problem. This is expected from NP-hard problem. This means our implementation is not feasible for large BPPs.

Population size and evaporation rate didn't affect computational time greatly. In BPP1, changing population to 1000 only increased the computational time by approximately 0.5 seconds. Population of 10000, on average, would take 1 second longer than with population of a 100. Changing the evaporation rate also did not have a noticeable impact on time performance of the algorithm. These parameters do, however, influence the quality of the result.

5.3 Potential improvements

Overall, the proposed approach seems to perform well on smaller BPPs. A better approach would still be Falkenauer's HGGA [6], which combines genetic algorithm with local search and is considered one of the best solution method for the BPP. Main downfall of our method seems to be its slowness. A way to improve this algorithm would be to combining it with a simple local search algorithm, based on Martello and Toth [13], as suggested in [10].

6. CONCLUSION

An ACO approach for the BPP was presented in this paper. Based on the carried out tests this algorithm works well for smaller BPPs, taking around 11 seconds to run for 10 bin problem. However, increasing number of bins to 50 increased computation time to 21 seconds, which proves to be too slow to use for larger problems.

We also found that for BPP of 10 bins and 500 items the best parameters were population of 10 and evaporation rate of 0.6. Increasing the number of bins slowed the computation down, and with 50 bins the best results were achieved using 10 bins with 0.9 evaporation rate. Since simple local search algorithm has proven to greatly improve results [10], further work should combine the approaches.

7. REFERENCES

[1] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, and S. Mirjalili. An improved nature inspired meta-heuristic algorithm for 1-d bin packing problems. *Personal and Ubiquitous Computing*, 22(5-6):1117–1132, 2018.

[2] E. Bonabeau, M. Dorigo, D. d. R. D. F. Marco, G. Theraulaz, G. Théraulaz, et al. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[3] B. Brugger, K. F. Doerner, R. F. Hartl, and M. Reimann. Antpacking–an ant colony optimization approach for the one-dimensional bin packing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 41–50. Springer, 2004.

[4] M. Dorigo, V. Maniezzo, A. Colorni, et al. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[5] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[6] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2(1):5–30, 1996.

[7] M. R. Gamy and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness. *Revista Da Escola De Enfermagem Da USP*, 44(2):340, 1979.

[8] A. Layeb and Z. Benayad. A novel firefly algorithm based ant colony optimization for solving combinatorial optimization problems. *IJCSA*, 11(2):19–37, 2014.

[9] A. Layeb and S. R. Boussalia. A novel quantum inspired cuckoo search algorithm for bin packing problem. *International Journal of Information Technology and Computer Science*, 4(5):58–67, 2012.

[10] J. Levine and F. Ducatelle. Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research society*, 55(7):705–716, 2004.

[11] C. Reeves. Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63(3):371–396, 1996.

[12] E. Sonuc, B. Sen, and S. Bayir. Solving bin packing problem using simulated annealing. In *Proceedings of 65th ISERD international conference, Mecca, Saudi Arabia*, 2017.

[13] P. Toth and S. Martello. *Knapsack problems: Algorithms and computer implementations*. Wiley, 1990.

[14] M. Vink. Solving combinatorial problems using evolutionary algorithms. *Master's thesis, Leiden University*, 1997.

[15] Z. Zendaoui and A. Layeb. Adaptive cuckoo search algorithm for the bin packing problem. In *Modelling and Implementation of Complex Systems*, pages 107–120. Springer, 2016.