Application of genetic algorithms to the problem of bin packing

ECM3412 - Nature-Inspired Computation

Candidate Number: 720037997
University of Exeter
Exeter, United Kingdom

***Abstract*—This paper investigates the effectiveness of a Genetic Algorithm in solving the Bin Packing Problem across two problems of varying complexity (linear BPP1 and quadratic BPP2). Through experimental analysis of the mutation rate (pm) and tournament size (t) the study identified the configuration $pm = 0.01, t = 7$ as optimal. This demonstrates that high selective pressure and minimal disruption are important for performance. Despite this, the resulting absolute load difference (d) remained high. This reveals the GA's limitation in achieving high precision local exploitation. Analysis suggests that the GA is inefficient in its final convergence phase. This led to the conclusion that a Memetic Algorithm (MA) would be better for achieving true load balancing precision.**

***Index Terms*—Genetic Algorithms, Bin Packing Problem, Load Balancing, Parameter Optimization, Exploitation, Memetic Algorithms**

I. Introduction

Optimisation lies right at the core of computer science. It demands efficient algorithms that can navigate massive solution spaces to achieve the best possible outcomes. The Bin Packing Problem (BPP) is a well-established NP-hard problem which can be used to effectively model real world logistics, resource allocation, and job scheduling. In the BPP, the number of bins is fixed, and the objective is to achieve the fairest distribution of load by minimising the total difference in weights (d) between the heaviest and lightest bins.

Due to the combinatorial size of the BPP's search space, exact methods become computationally intractable for large cases. Therefore, nature inspired metaheuristics have become the standard approach. This paper investigates the effectiveness of the Genetic Algorithm (GA): A population based evolutionary strategy in solving the BPP. The implementation uses an assignment based chromosome representation and generational replacement incorporating elitism.

The objective of this study is to understand the GA's sensitivity to key parameters when applied to problems of varying complexity. We define two distinct problems: BPP1, linearly increasing item weights, and BPP2, quadratically increasing weights. The BPP2 represents a significantly more rugged search landscape for the GA to tackle compared to BPP1, thus providing a strong basis for comparison. To achieve the objective, we conduct an analysis across four parameter settings; varying the mutation rate (pm) and the tournament selection size (t). The findings will clarify the balance between the GA's exploration and exploitation capabilities across different problem characteristics.

II. Literature Review

A. Genetic Algorithm Variants

The complexity of the Bin Packing Problem (BPP) is well established, as the problem is known to be NP-hard [2]. Given that packing tasks are combinatorial problems with very large search spaces, the use of specialised metaheuristic search methods, particularly Genetic Algorithms, is encouraged by recent studies [1], [2]. The main challenge lies in chromosome encoding. Two strategies are employed: Permutation Encoding, where the chromosome defines the sequence of items, and decoded by a packing heuristic (e.g .First-Fit) to ensure a feasible solution [1]. While Assignment Encoding is simpler, the risk of solution disruption during crossover and mutation is high, which can potentially destroy fit assignment patterns [2].

To manage this complexity, advanced techniques like Grouping Genetic Algorithms (GGAs) are often used. Many successful GA applications are structured as Memetic Algorithms (MAs) [5], which combine global population based search with local improvement operators [4]. This hybrid approach is critical for preventing premature convergence and efficiently managing the precise demands of the BPP. Experimental comparisons confirm that the hybrid combination yields an algorithm superior to both of its components [3]. It is able to perform equally well, and in some cases better than existing heuristics on benchmark problems [4].

B. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based metaheuristic inspired by the social dynamics of bird flocking, where 'particles' move through the search space influenced by their own best position $(pbest)$ and the swarm's best position $(gbest)$. This method is defined by concepts of position and velocity vectors [6], making it ideal for the continuous optimisation of non linear functions.

However, this architecture presents an issue for the BPP, where the solution space consists of integer bin assignments, not floating point coordinates, meaning the continuous PSO cannot be applied directly. The algorithm must be made into a Discrete PSO (DPSO) model [7]. This often involves complex

mapping mechanisms, such as converting continuous position or velocity into a probability of assignment change. Studies on DPSO have pointed out that this transformation ruins the simplicity of the original method. This introduces representational challenges and new convergence issues, meaning the algorithm may not converge well. The straightforward efficiency of continuous PSO is lost when it is adapted for discrete combinatorial problems.

C. Ant Colony Optimization

Ant Colony Optimization (ACO) is a constructive metaheuristic inspired by the foraging behaviour of ants, that find the shortest path using chemical trails (pheromones). For the BPP, ACO constructs a solution sequentially as the 'ants' incrementally build solutions on a construction graph [9]. An ant places an item into a bin using a decision making process entailing a combination of the current pheromone level and a heuristic measure [9]. This makes ACO highly suitable for constructive combinatorial problems for example minimising the amount of bins if b was not fixed [9]. However, for the BPP (with a fixed size of bins(b)), the objective function depends on the total system balance (d), which is a global metric. Because basic ACO agents make decisions based on limited local information [10], the algorithm struggles to integrate this global feedback. Therefore basic ACO can be inefficient or costly for this problem. It would require hybrid extensions to handle the demands of the BPP [10].

D. Other Nature-Inspired Approaches

Single solution metaheuristics are crucial for problems demanding local searches. Two such methods frequently applied to the BPP are Simulated Annealing (SA) and Tabu Search (TS). Simulated Annealing (SA) is a technique inspired by the process of annealing in metallurgy. SA explores the solution neighbourhood by accepting a move to a worse solution (one with a higher d) with a controlled probability. This probability decreases according to a temperature parameter that is gradually lowered according to a cooling schedule [11]. This allows for effective exploration and escape from local optima, making SA great for the final stages of optimisation where precise adjustments are required to minimise d. Tabu Search (TS) is a local search method that employs a short term memory structure called the tabu list. It does not allow recently visited moves. This is done to avoid cycles and encourage exploration [12]. This approach to local exploitation provides an alternative mechanism for fine tuning bin assignments and is often deployed as a local refinement procedure within hybrid algorithms.

E. Critical Analysis

The optimal metaheuristic strategy for the BPP must balance global exploration and local exploitation. Because of the large solution space the Genetic Algorithm (GA) is the best choice for the initial global exploration phase. It excels at combining effective partial assignments through its uniform crossover operator, identifying the most promising region of the landscape. In contrast, Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) demonstrate significant limitations: PSO requires complex discretisation, losing its efficiency, while ACO's reliance on sequential local decision making struggles with the global nature of the load balancing objective (d).

However, the pure GA is inefficient in the final stage of search. The requirement to minimise d demands high precision local exploitation, a task not suitable for the GA's stochastic operators. This is where single solution methods like Simulated Annealing (SA) and Tabu Search (TS) shine, which perform incremental adjustments in the solution neighbourhood.

Therefore, the ideal approach is a Hybrid or Memetic Algorithm (MA) [5]. This structure leverages the GA's ability to globally locate a near optimal solution and then deploys the local search capabilities of SA or TS to perform the fine tuning required to achieve the minimum load difference [4]. This combination addresses the weaknesses of both population based (GA) and single solution (SA/TS) methods, making it the best way to tackle BPP.

III. Description of Results

A. Experimental Setup

The investigation was done by implementing a Genetic Algorithm (GA) in Python, specifically configured for the Bin Packing Problem (BPP). The chromosome was represented as an array of length $K = 500$, where each gene indicates the assigned bin $(1$ to $b)$ for the corresponding item. The fitness function $F = 100/(1 + d)$ was used to maximise the solution quality, where d is the difference between the heaviest and lightest bin loads. The GA used Uniform Crossover $(pc = 0.8)$ and Random Reassignment Mutation as well as Generational Replacement with Elitism (one 'elite' chromosome preserved per generation).

Two problems were tested: BPP1: $b = 10$ bins, item weights are linear BPP2: $b = 50$ bins, item weights are quadratic

For each of the four required parameter combinations, five trials were executed using a different random seed each time. The population size was fixed at $p = 100$, and the termination criteria was a maximum of 10,000 fitness evaluations. This means 100 generations of the GA were executed for each trial. The parameter settings varied the Mutation Rate(pm) and Tournament Selection Size(t) to assess the trade off between exploration and exploitation.

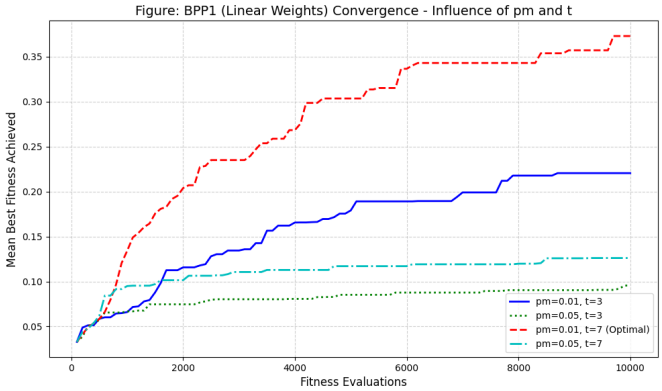
B. Experimental Results

The experiment was executed across 40 distinct trials to assess the influence of mutation rate (pm) and tournament size(t) on the GA's performance for the Load Balancing BPP. The mean final difference(d), standard deviation(σd) and best achieved d $(bolded)$ are presented in the tables below. Mean Fitness is calculated using the final mean d with the formula provided $F = 100/(1 + d)$

1) BPP1 Results (10 bins, 500 items):

TABLE I: BPP1 Results Summary

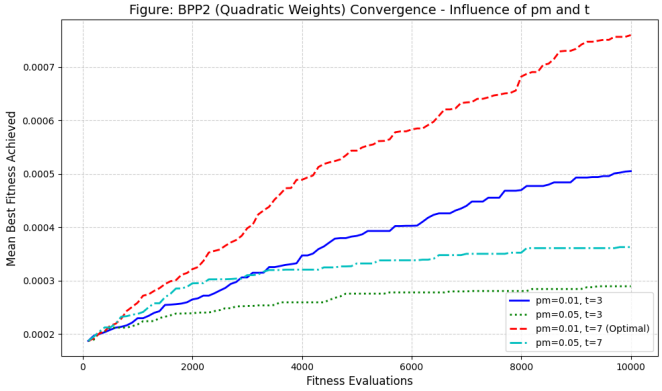
p_m	**Tournament Size**	**Best Fitness**	**Mean Fitness**	**Std Dev**
0.01	3	0.252	0.217	60.64
0.05	3	0.102	0.095	89.70
0.01	**7**	**0.495**	**0.358**	**56.40**
0.05	7	0.139	0.125	50.94



2) BPP2 Results (50 bins, 500 items):

TABLE II: BPP2 Results Summary

p_m	**Tournament Size**	**Best Fitness**	**Mean Fitness**	**Std Dev**
0.01	3	0.000581	0.000500	20634.49
0.05	3	0.000301	0.000289	8056.00
0.01	**7**	**0.000885**	**0.000755**	**9891.35**
0.05	7	0.000405	0.000361	17436.85



IV. Discussion and Further Work

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

The experimental results consistently identify the parameters of $pm = 0.01$ (low mutation rate) and $t = 7$ (high tournament size) as the optimal configuration for both BPP1 And BPP2.

For BPP1 this setting achieved the lowest Mean d of 278.40 and therefore the highest Mean Fitness of 0.358. This outperformed the next best setting $(pm = 0.01, t = 3)$ by over 39

For BPP2 it achieved the highest Mean Fitness of 0.000885 and the lowest Mean d, showing its superiority even on a significantly more rugged search landscape.

This finding demonstrates that the most effective strategy for the BPP problem, regardless of the scale or complexity. It prioritises high selection pressure over high genetic diversity. These settings successfully balance the forces of exploitation and exploration to achieve the lowest final load imbalance.

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

The consistent performance of the $pm = 0.01, t = 7$ configuration is a direct result of the search strategy that prioritises strong exploitation with minimal solution disruption. The high tournament size $(t = 7)$ is the primary driver of the GA's success, as it introduces high selective pressure. This pressure forces the algorithm to identify and propagate the best suited individuals, focusing computational effort on exploiting beneficial assignments found via crossover. Conversely, the low mutation rate $(pm = 0.01)$ acts as a critical protective mechanism. Because the load balancing metric (d) is highly sensitive any significant random change has a high probability of destroying an already balanced solution. By minimising the disruptive effect of mutation the GA preserves the integrity of high quality solutions. This allows the $t = 7$ selection process to operate efficiently and achieve better convergence. The winning strategy avoids excessive random exploration in favour of the preservation and refinement of the best bin assignments.

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

The two parameters, tournament size and mutation rate control the GA's search balance as shown in the convergence graphs. The tournament size dictates the rate of exploitation. The increase from $t = 3$ to $t = 7$ led to a significant and consistent increase in Mean Fitness across all mutation combinations. The graphs demonstrate that $t = 7$ settings show a much faster convergence rate. This validates the conclusion that high selective pressure is essential for the refinement required by the problem. The $t = 7$ pressure quickly filters the less suitable solutions. That makes sure that the population's focus is on refining the best solution possible.

The mutation rate influences performance by controlling the amount of destructive random change introduced. The high rate of $pm = 0.05$ shows to be very detrimental as shown by the drop in mean fitness. For BPP1 the Mean d got worse with the higher mutation rate proving that excessive random assignment is not efficient for this problem. This proves that random exploration frequently undoes the load balancing achieved by the crossover. Therefore the optimal strategy requires that pm should be kept low to only ensure

stagnation does not occur and making sure the primary search and refinement processes are not corrupted.

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

Yes, I believe that using a Memetic Algorithm (MA) which takes advantage of GA with a local search technique would give a much lower load difference score.

The high Mean d values for both BPP1 and BPP2 shows GA's limitation. While GA has located the optimal search region using the $pm = 0.01, t = 7$ configuration, its reliance on stochastic crossover and mutation is inefficient in the end stages. These operators are too harsh on the small changes needed to reduce d down to a lower value closer to zero.

As shown in the literature review, the final minimisation of d requires high precision local exploitation. This is where the likes of Simulated Annealing (SA) or Tabu Search (TS) perform better due to the ability to maintain memory or accept moves systematically [12].

A Memetic Algorithm would be superior by separating the search tasks: the GA would be used for global search and then the SA or TS would do the final evaluations. This local search component would then perform targeted swaps (e.g. between heaviest and lightest bin) to perform fine tuning the GA failed to do well. This would minimise d to a much smaller value. Experimental comparisons validate this thus proving the hybrid approach is superior [3], [4].

E. Further Experiments

To provide more insight into the search mechanics, two additional experiments were conducted to test the influence of the fixed parameters: Population Size (p) and Crossover Rate (pc). Both new settings were tested on the BPP1 problem, using the optimal $pm = 0.01, t = 7$ configuration as the performance baseline (Mean $d = 278.40$).

1. Influence of Population Size (p) The population size was doubled to $p = 200$, reducing the total number of generations from 100 to 50 within the 10,000 evaluation budget. The Mean d for $p = 200$ was found to be 384.80 which is 38.2

2. Influence of Crossover Rate (pc)

The crossover rate was lowered from $pc = 0.8$ to $pc = 0.5$ to test the reliance on the recombination operator. The Mean d for $pc = 0.5$ was found to be 276.60 ($\sigma = 53.73$). This result is slightly better than the baseline Mean d of 278.40, which contradicts the initial hypothesis that a high crossover rate is essential for structural improvement. This finding suggests that the original $pc = 0.8$ rate was slightly too high which induced unnecessary change that interfered with stable convergence. The lower $pc = 0.5$ rate provides a near optimal balance allowing necessary recombination while simultaneously maintaining the structural stability required for the aggressive selection pressure $(t = 7)$ to succeed. This provides more information in understanding the interplay between recombination and exploitation in load balancing problems.

F. Future Work

The main direction for future work should address the weakness identified in this study: the inability of the pure Genetic Algorithm to achieve high precision results. This is proven by the high d values across BPP1 and BPP2. The next step should be to implement an test the Memetic Algorithm (MA), using the proven optimal GA settings($pm = 0.01, t = 7$), with some sort of local search heuristic either SA or TS. This would provide evidence to prove the literature supported view that MAs are superior for load balancing precision.

V. Conclusion

This study successfully investigated the application of a Genetic Algorithm to the Bin Packing Problem across two instances. The experimental analysis demonstrated that the optimal search strategy relies on strong exploitation via high tournament selection $(t = 7)$ with minimal disruption from a low mutation rate $(pm = 0.01)$. This strategy consistently outperformed all other tested settings. However, the high final load difference (d) values highlighted the pure GA's inherent weakness in high precision tasks. The main contribution is the validation that the BPP needs a hybrid approach: using the GA for global search and a local search technique for the final refinement.

References

[1] C. Munien and A. E. Ezugwu, "Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 248-271, 2021.

[2] E. Hopper and B. C. H. Turton, "A review of the application of meta-heuristic algorithms to 2D strip packing problems," *Artificial Intelligence Review*, vol. 16, no. 4, pp. 257-300, 2001.

[3] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," in *Proc. of the 4th IEEE Conf. on Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1996, pp. 433-438.

[4] J. A. Fernández et al., "A New Memetic Algorithm for the Two-Dimensional Bin-Packing Problem with Rotations," *Journal of Heuristics*, vol. 16, pp. 823-847, 2010.

[5] P. Moscato and C. Cotta, "A Gentle Introduction to Memetic Algorithms," in *Handbook of Evolutionary Computation*, A. E. Eiben, Ed. Berlin, Germany: Springer, 2003, pp. G1-G24.

[6] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proc. of IEEE Int. Conf. on Neural Networks*, Piscataway, NJ: IEEE Press, 1995, vol. 4, pp. 1942-1948.

[7] Q. Kang et al., "A novel discrete particle swarm optimization algorithm for the set-covering problem," *Expert Systems with Applications*, vol. 39, no. 1, pp. 915-925, 2012.

[8] M. Macedo et al., "Overview on Binary Optimization Using Swarm-Inspired Algorithms," *IEEE Access*, vol. 9, pp. 149814-149858, 2021.

[9] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Berlin, Germany: Springer, 2010.

[10] M. S. Nadimi-Shahraki et al., "Efficient load balancing using ant colony optimization," *Journal of Advanced Computer Science and Technology*, vol. 12, no. 3, pp. 18-35, 2023.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[12] F. Glover, "Tabu Search—Part I," *Journal on Computing*, vol. 1, no. 3, pp. 190-206, 1989.