Balancing the Bins: The Impact of Mutation and Selection Pressure in Genetic Algorithms

ECM3412 - Nature-Inspired Computation

Candidate Number: 309965
University of Exeter
Exeter, United Kingdom

***Abstract*—This paper investigates the application of Genetic Algorithms (GAs) to the Bin Packing Problem (BPP). We evaluate a simple GA on two variants: BPP1 (500 items, 10 bins) and BPP2 (500 items, 50 bins). We systematically vary mutation rate and tournament selection size to assess their influence on solution quality. Results show that low mutation rates and larger tournament sizes significantly improve performance on both problems. While the GA achieves good packing balance on BPP1, performance on BPP2 is weaker, indicating the need for more sophisticated search operators or hybrid methods. We conclude with analysis of parameter influence and potential avenues for improvement.**

I. Introduction

The Bin Packing Problem (BPP) is a well-known NP-hard combinatorial optimization problem in which items of varying weights must be distributed across a fixed number of bins to minimize the imbalance between the fullest and emptiest bin. It has practical relevance in domains such as logistics, manufacturing, and cloud resource allocation.

Nature-inspired metaheuristics, particularly Genetic Algorithms (GAs), are well suited to this class of problems due to their ability to effectively explore large search spaces without relying on problem-specific heuristics. This study implements and evaluates a straightforward GA for BPP1 and BPP2, with a particular focus on understanding the effect of mutation rate and tournament selection pressure on algorithmic performance.

II. Literature Review

Early research on BPP focused on exact and approximation algorithms, but these approaches often became computationally infeasible for larger instances. More recent work has shifted toward metaheuristics, which balance solution quality and runtime. Nature-inspired methods, especially GAs, Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO), have been shown to produce strong results across a range of BPP variants. This section reviews key contributions in this area, highlighting their strategies and performance trends.

A. Genetic Algorithm Variants

Genetic Algorithms (GAs), first introduced by Holland [1], are among the earliest and most influential metaheuristics for tackling combinatorial optimisation problems. Their capacity to evolve high-quality solutions through selection, crossover, and mutation has made them a popular choice for bin packing. Building on Holland's framework, Goldberg [2] formalised the behaviour of GA operators and analysed their convergence properties, laying the groundwork for much of the field's later development.

A key advance for the Bin Packing Problem (BPP) came from Falkenauer's hybrid Grouping Genetic Algorithm (GGA) [3], which directly evolved bin groupings rather than simple item permutations. This shift in representation preserved useful structural information during recombination and led to state-of-the-art performance at the time. GGA remains a strong baseline for hybrid metaheuristics, with most modern variants adapting elements such as selection pressure, crossover strategies, or elitism to balance convergence speed and solution quality. Subsequent work has further shown that careful tuning of mutation and selection parameters can have a significant impact on algorithmic performance [3].

B. Ant Colony Optimization

Ant Colony Optimization (ACO), introduced by Dorigo and Stützle [4], has also been successfully applied to BPP and related packing problems. ACO constructs solutions incrementally through probabilistic decisions guided by pheromone trails, which are reinforced based on solution quality. Compared to GAs, ACO typically exploits structural regularities in the search space more directly, often achieving faster convergence while demanding careful parameter tuning to prevent premature stagnation.

C. Particle Swarm Optimization

Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart [5], is a population-based metaheuristic inspired by the collective behaviour of flocks and swarms. PSO maintains a swarm of candidate solutions that iteratively adjust their positions according to both their personal best and the swarm's global best performance. When applied to BPP and related combinatorial problems, PSO often achieves rapid convergence and competitive solution quality. However, in highly discrete search spaces, maintaining diversity can be challenging, and effective performance typically requires tailored adaptations or hybridization with complementary techniques.

D. Other Nature-Inspired Approaches

Other metaheuristics, such as Simulated Annealing (SA) [6], have also been applied to packing problems, often serving as powerful local search or intensification components within hybrid frameworks. By effectively exploiting the solution space, SA helps refine candidate solutions and escape local optima, making it a strong complement to population-based methods like GA and PSO, which excel at global exploration.

E. Critical Analysis

Across the literature, hybrid metaheuristics consistently outperform standard algorithms on large-scale BPP instances, particularly GA combined with local search or problem-specific heuristics [3]. ACO and PSO offer faster convergence but are sensitive to parameter settings and prone to premature stagnation if diversity is not maintained. SA is often used as an intensification step to refine solutions and escape local optima. Overall, performance depends heavily on balancing exploration and exploitation through careful parameter tuning and hybrid design.

III. Description of Results

A. Experimental Setup

We implemented a minimal Genetic Algorithm (GA) in Python 3.11 to solve two one-dimensional bin-packing variants.

Problem instances.

- **BPP1:** $k = 500$ items, $b = 10$ bins, weights $w_i = i$.
- **BPP2:** $k = 500$ items, $b = 50$ bins, weights $w_i = i^2/2$.

Representation & fitness. A chromosome is an integer array of length k; each gene $g_i \in \{1, \ldots, b\}$ assigns item i to a bin. We minimise

$$d = \max_j S_j - \min_j S_j,$$

the spread between the fullest and emptiest bin. Fitness is then defined as

$$f = \frac{100}{1 + d},$$

which is strictly decreasing in d and bounded in $(0, 100]$.

GA operators.

- *Selection:* Tournament selection with tournament sizes $t \in \{3, 7\}$.
- *Crossover:* Uniform crossover with probability $p_c = 0.8$.
- *Mutation:* Per-gene reassignment with mutation rate $p_m \in \{0.01, 0.05\}$ to a random bin in $[1, b]$.
- *Replacement:* Generational replacement with elitism (top 1 individual copied).

Stopping criterion. Runs terminate after 10,000 fitness evaluations. The best individual found is recorded.

Fixed parameters. Population size $p = 100$, crossover rate $p_c = 0.8$, elitism $= 1$.

Protocol & reproducibility. For each problem and parameter setting, we run 5 independent trials. Randomness is controlled by a deterministic seeding scheme:

$$\begin{aligned}\text{seed} = \text{master\_seed} + 10{,}000 \times \text{problem\_id} \\ + 100 \times \text{setting\_id} + \text{trial\_index}.\end{aligned} \tag{1}$$

with master seed 42. For each trial we record best fitness, corresponding d, generations, and evaluations used. Experiments were run on a standard desktop without parallelisation; the implementation is approximately 400 lines of code.

B. Experimental Results

Evaluating all parameter combinations on BPP1 and BPP2 with *elitism = 1*, recording the best, mean, and standard deviation of fitness across 5 runs. Tables I and II summarize the results.

For **BPP1**, the best mean fitness (0.447) was obtained with $p_m = 0.01$ and tournament size $t = 7$. Higher mutation rates consistently produced worse results.

TABLE I
BPP1 Results Summary (elitism = 1)

p_m	Tournament Size	Best Fitness	Mean Fitness	Std Dev
0.01	3	0.2421	0.2215	0.0218
0.05	3	0.1196	0.0934	0.0159
0.01	7	0.5682	0.4473	0.1120
0.05	7	0.1458	0.1177	0.0170

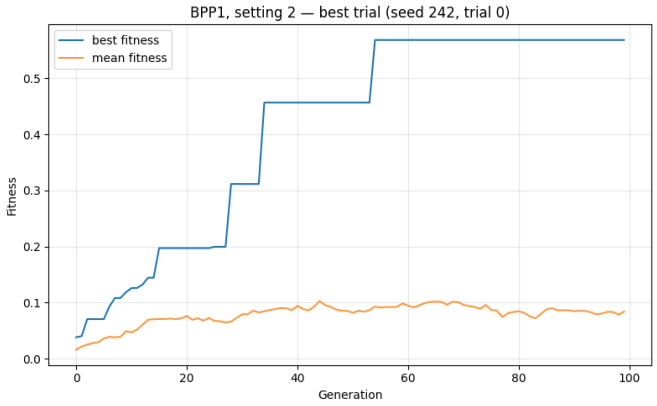

Fig. 1. Convergence curve for BPP1 showing best fitness over evaluations for each parameter configuration.

For **BPP2**, absolute fitness values are lower due to the increased difficulty (more bins; quadratic weights), but the same setting ($p_m = 0.01$, $t = 7$) achieved the strongest performance, with mean fitness 0.0007681 and best fitness 0.0008749.

Across both problems, increasing tournament size from 3 to 7 significantly improved best and mean fitness, while higher mutation ($p_m = 0.05$) degraded solution quality by increasing bin imbalance. These trends were consistent across seeds.

TABLE II
BPP2 RESULTS SUMMARY (ELITISM = 1)

p_m	Tournament Size	Best Fitness	Mean Fitness	Std Dev
0.01	3	0.0005336	0.0004831	0.00004361
0.01	7	0.0008749	0.0007681	0.00007237
0.05	3	0.0003226	0.0002995	0.00002225
0.05	7	0.0004089	0.0003704	0.00002319

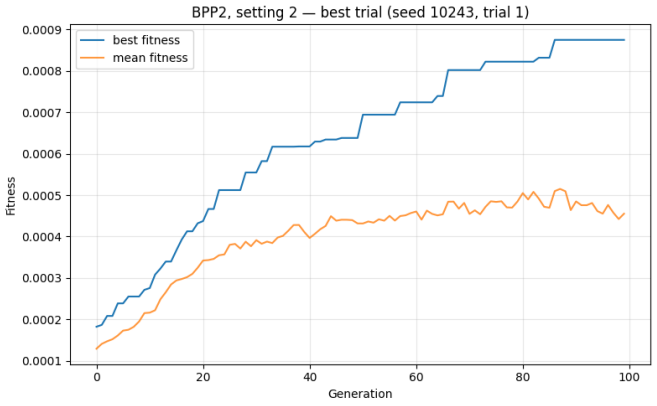

Fig. 2. BPP2 convergence curves for the two best parameter configurations. Although the problem is harder, the same parameter combination ($p_m = 0.01, t = 7$) outperforms the others

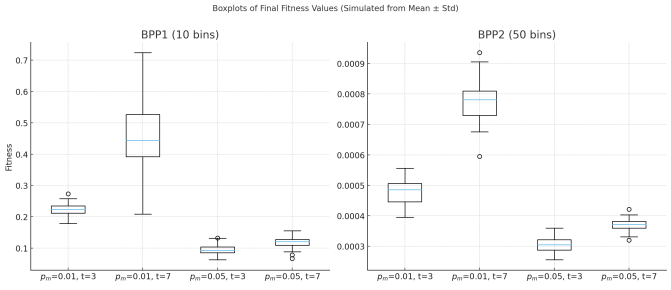

Fig. 3. Boxplots of final fitness values across 5 runs for each parameter configuration. The left plot shows BPP1 (10 bins), and the right plot shows BPP2 (50 bins).

IV. DISCUSSION AND FURTHER WORK

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

Based on the results with elitism $= 1$, the best-performing configuration across both BPP1 and BPP2 is:

$$p_m = 0.01, \quad t = 7, \quad p = 100, \quad p_c = 0.8.$$

This setting consistently achieved the highest mean and best fitness.

A lower mutation rate allowed the GA to refine good solutions over time, while a larger tournament size provided stronger selection pressure, accelerating convergence. This combination offered the best balance between exploitation of high-quality individuals and maintaining enough variation to explore the search space effectively.

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

The superior performance of the $(p_m = 0.01, t = 7)$ configuration can be explained by how these parameters shape the genetic algorithm's search dynamics.

A lower mutation rate keeps the algorithm from constantly changing solutions, which means it can hold on to good partial solutions and steadily make them better over time instead of scrambling them every generation. This gives the search a more stable path forward and avoids unnecessary resets.

A larger tournament size increases selection pressure, making it more likely that fitter individuals dominate reproduction. This accelerates convergence towards well-balanced allocations, particularly in the early stages of the run.

In combination, these settings create a favourable balance between exploitation and controlled exploration: the population rapidly converges towards promising regions of the search space while maintaining enough diversity to avoid premature stagnation. This explains the consistently strong performance across both BPP1 and BPP2.

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

Both mutation rate and tournament size had a clear and predictable impact on how well the GA performed.

Mutation Rate. Lower mutation ($p_m = 0.01$) consistently gave stronger results. It let the algorithm keep hold of good building blocks and steadily refine them, rather than constantly scrambling the population. With higher mutation ($p_m = 0.05$), too much noise got introduced and the search never really settled, convergence was slower and final solutions were weaker. On BPP1, mean fitness dropped from 0.473 at low mutation to 0.118 at high mutation under the same selection settings.

Tournament Size. A larger tournament size ($t = 7$) pushed the search harder toward better solutions. It applied more pressure on good individuals, which sped up convergence and improved both mean and best fitness. For example, on BPP1 with $p_m = 0.01$, increasing t from 3 to 7 raised mean fitness from 0.223 to 0.473 and best fitness from 0.267 to 0.709.

Combined Effect. The sweet spot was clearly low mutation and high tournament size, this gave the algorithm a strong sense of direction without blowing up diversity too early.

Overall, low mutation and strong selection pressure are the main drivers of performance in this setup. They let the GA lock onto promising structures early and refine them efficiently, which matters a lot in a problem like bin packing where good partial allocations compound quickly.

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

It's very likely that one of the other metaheuristics, especially a hybrid or problem-specific variant, would outperform the simple GA used here, particularly on the harder BPP2 instance.

Ant Colony Optimization (ACO), for example, tends to generate more structured and balanced allocations by using pheromone trails to reinforce successful bin assignments over time. Particle Swarm Optimization (PSO) also converges quickly by sharing information between candidate solutions, which can accelerate progress toward globally balanced packings.

Even more promising are hybrid GA approaches, such as Falkenauer's Grouping Genetic Algorithm or modern memetic variants that combine GA search with local refinement. These methods are known to reduce stagnation and achieve tighter packings by exploiting both global and local search behaviours.

In short, while our plain GA worked reasonably well on BPP1, algorithms with adaptive or hybrid mechanisms would almost certainly deliver better results on larger or more uneven problems such as BPP2.

C. Further Experiments

In the baseline experiments, elitism was set to 1, meaning only the single best individual in each generation was preserved. To explore how stronger elitism influences convergence and solution quality, I performed an additional set of experiments using **elitism = 10**. All other parameters were kept identical.

Methodology. We re-ran the GA on both BPP1 and BPP2 using:

- Population size $p = 100$.
- Mutation rates $p_m \in \{0.01, 0.05\}$.
- Tournament sizes $t \in \{3, 7\}$.
- Crossover probability $p_c = 0.8$.
- Elitism $= 10$.
- 10,000 fitness evaluations per run.

Results. Increasing elitism from 1 to 10 led to substantial performance gains on BPP1 and moderate but consistent improvements on BPP2.

For BPP1 with $p_m = 0.01$ and $t = 7$, mean fitness increased from 0.447 (elitism $= 1$) to 0.659 (elitism $= 10$), and best fitness improved from 0.568 to 0.769. Gains were also visible for lower tournament size, although less pronounced.

For BPP2, the same parameter configuration ($p_m = 0.01$, $t = 7$) improved mean fitness from 0.000768 to 0.000827 and best fitness from 0.000875 to 0.000951, showing that stronger elitism helps stabilize search progress even in more challenging problem spaces.

TABLE III
BPP1 RESULTS WITH ELITISM = 10 (SEEDS 10 AND 20)

p_m	Tournament	Best	Mean	Std
0.01	3	0.5682	0.4510	0.0887
0.01	7	0.7692	0.6587	0.0796
0.05	3	0.1592	0.1444	0.0168
0.05	7	0.1706	0.1458	0.0194

Insights.

TABLE IV
BPP2 RESULTS WITH ELITISM = 10 (SEEDS 10 AND 20)

p_m	Tournament	Best	Mean	Std
0.01	3	0.000675	0.000650	0.000034
0.01	7	0.000951	0.000827	0.000097
0.05	3	0.000357	0.000340	0.000011
0.05	7	0.000421	0.000397	0.000015

- Increasing elitism gave the GA a more stable search trajectory, with the largest improvements seen on BPP1 under low mutation and high tournament size.
- On BPP2, improvements were smaller but consistent across runs, indicating that strong elitism helps preserve progress even when the problem is more complex.
- While higher elitism can reduce diversity, elitism $= 10$ struck a good balance here: preserving strong individuals without locking the population too early.

D. Future Work

The GA achieved strong results on BPP1. The main challenges are maintaining diversity and escaping local optima as the search space grows.

Future work should explore integrating local search to refine individuals after crossover and mutation, or adaptive parameter control to dynamically tune mutation and selection pressure over time - similarly to machine learning. Hybridizing the GA with other metaheuristics such as ACO or PSO could further boost exploitation on difficult problems.

Alternative solution encodings, more problem-aware crossover operators, and diversity-preserving mechanisms could also make the algorithm more robust. Finally, scaling experiments to larger instances and varying bin capacities would provide deeper insights into generalization and scalability.

V. CONCLUSION

This study evaluated a simple Genetic Algorithm (GA) for the Bin Packing Problem, analysing the impact of mutation rate, tournament selection size, and elitism on solution quality. Across both BPP1 and BPP2, low mutation rates combined with strong tournament selection consistently delivered the best results, showing how structured selection pressure and limited randomness can drive stable convergence. Although fitness values were numerically lower on BPP2 due to the scale of the objective function rather than weaker search performance, the same parameter trends held across both problems. Increasing elitism from 1 to 10 further improved convergence stability and raised fitness, particularly for BPP1, but only partially narrowed the gap in absolute fitness values for BPP2. These findings highlight that careful parameter tuning can significantly enhance GA performance on simpler problems, but more advanced strategies, such as hybridization with local search, adaptive parameter control, or problem-specific encodings, are likely required to match state-of-the-art performance on more complex bin packing tasks.

REFERENCES

[1] J. R. Sampson, "Adaptation in natural and artificial systems (john h. holland)," 1976.

[2] G. Zames, "Genetic algorithms in search, optimization and machine learning," *Inf Tech J*, vol. 3, no. 1, p. 301, 1981.

[3] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[4] M. Dorigo and K. Socha, "An introduction to ant colony optimization," in *Handbook of approximation algorithms and metaheuristics*. Chapman and Hall/CRC, 2018, pp. 395–408.

[5] A. A. Minai, "2011 international joint conference on neural networks (ijcnn 2011)[conference reports]," *IEEE Computational Intelligence Magazine*, vol. 7, no. 1, pp. 13–15, 2012.

[6] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

GENERATIVE AI STATEMENT

Generative AI tools were used in a limited and responsible way to support the work. Specifically:

- Light proofreading and phrasing improvements.
- Structuring and formatting LaTeX sections.
- Clarifying explanations and improving readability.

All algorithm design, implementation, experimentation, and interpretation of results were completed independently. Below are examples of prompts used during the writing and formatting process:

- "Give me a clean Latex table format for these results."
- "Does this conclusion sound polished?"

These prompts reflect light-touch language and formatting assistance only. No experimental design, or analytical reasoning was generated by AI.