

Genetic Algorithm: Bin Packing Problem

Matthew D. Branson

Department of Computer Science

Missouri State University

Springfield, MO

branson773@live.missouristate.edu

Abstract

This paper presents the implementation and analysis of a...

Index Terms

Genetic Algorithms, Bin Packing Problem, Optimization, Heuristic Search

I. Q1: ENCODING AND INITIALIZATION

An integer-based representation is used to encode bin assignments. For an instance with N items, each individual is a vector of length N , where the value at index i denotes the bin to which item i is assigned.

For example, the encoding $[0, 1, 0, 2, 1, 1, 2, 1, 3, 0]$ represents a solution for 10 items in which items 0, 2, and 9 are assigned to bin 0; items 1, 4, 5, and 7 to bin 1; items 3 and 6 to bin 2; and item 8 to bin 3, resulting in the use of 4 unique bins.

Initial populations are generated by assigning each item independently to a random bin in the range $[0, N - 1]$. This produces diverse initial solutions while satisfying the constraint that the maximum number of bins does not exceed the number of items.

II. Q2: FUNCTION EVALUATION

A. Objective Function

The objective function f measures the number of unique bins used in a given individual. For instance, the encoding $[0, 1, 0, 2, 1, 1, 2, 1, 3, 0]$ uses bins $\{0, 1, 2, 3\}$, yielding $f = 4$. The

algorithm minimizes f to reduce the total number of shipping boxes required.

B. Constraint Handling

The constraint violation function g counts the number of bins whose total weight exceeds the 10 kg capacity. For each bin, the weights of assigned items are summed. A bin contributes 1 to g if its cumulative weight exceeds the limit. A solution is considered feasible when $g = 0$.

C. Evaluation Function

Each individual is evaluated as a tuple (f, g) , where f represents the number of bins used and g the number of constraint violations. This formulation enables the algorithm to balance minimization of bin usage with satisfaction of feasibility constraints.

III. Q3: GA OPERATIONS

A. Tournament Selection

Tournament selection is used to construct a mating pool of N individuals. For each selection, two candidates are drawn at random, and the winner is determined by a constraint-aware comparison rule. This process is repeated N times. Mating pairs are then selected at random from the resulting pool.

B. Winner Selection Rules

Selection is guided by the following criteria:

- If both individuals are feasible ($g_1 = 0, g_2 = 0$), the one with lower f is preferred.
- If only one is feasible, the feasible individual is selected.
- If both are infeasible, the one with fewer violations (g) is selected.

These rules guide selection toward feasible individuals with fewer bins and penalize constraint violations when feasibility is not yet achieved.

C. Crossover

Two-point crossover is applied with a probability of 0.9. Two cut points are chosen to divide each parent chromosome into three segments. Offspring are generated by exchanging the middle segments, preserving the integer-based encoding and promoting genetic diversity.

D. Mutation

Each gene (bin assignment) is mutated independently with probability $p_m = 1/N$, where N is the number of items. When selected, a gene is reassigned to a random integer in $[0, N - 1]$. This maintains valid encodings while introducing variation for local exploration.

IV. Q4: GA EXECUTION

A. Baseline Configuration

The genetic algorithm was initially executed using the specified baseline parameters: population size of 20 and 50 generations. This configuration was tested on four problem instances containing 10, 25, 50, and 100 orders, with order weights uniformly distributed between 0 and 2 kg and a bin capacity of 10 kg.

B. Parameter Variation Study

To assess the impact of population size and generation count on performance, additional configurations were tested. Population sizes of 10 and 40 individuals were compared against the baseline of 20, while generation counts of 25 and 100 were compared against the baseline of 50. Additionally, each configuration was evaluated across all four problem sizes using three random seeds (42, 773, 2025) to examine performance variability.

C. Fitness Plots

The evolution of best and average fitness values across generations is shown for various configurations.

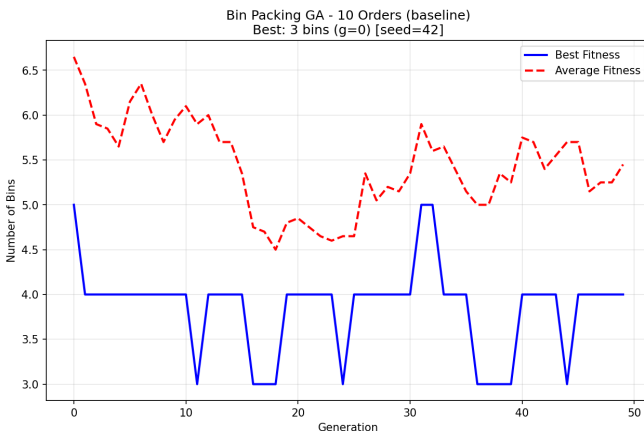


Fig. 1: Baseline: 10 orders

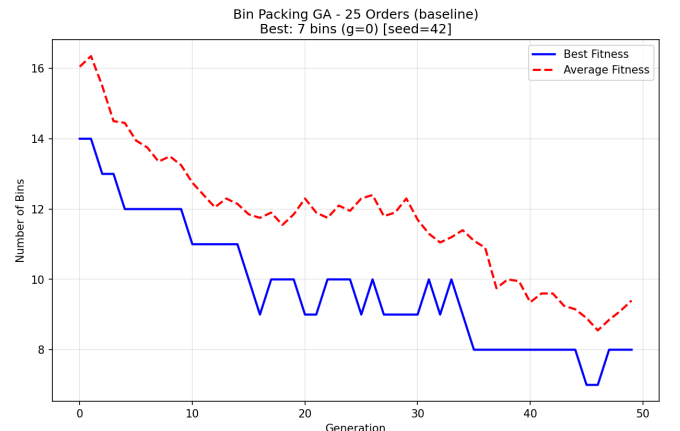


Fig. 2: Baseline: 25 orders

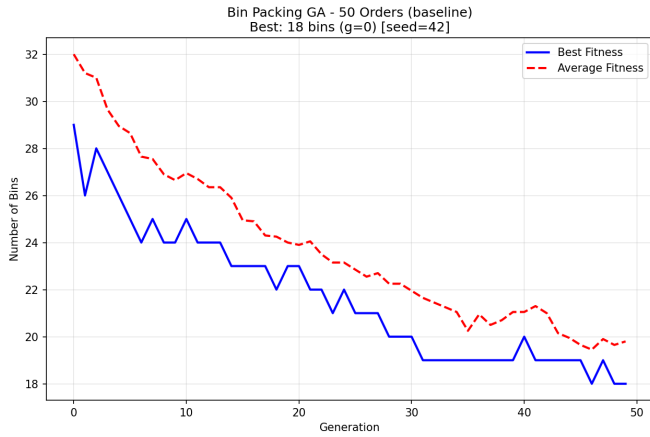


Fig. 3: Baseline: 50 orders

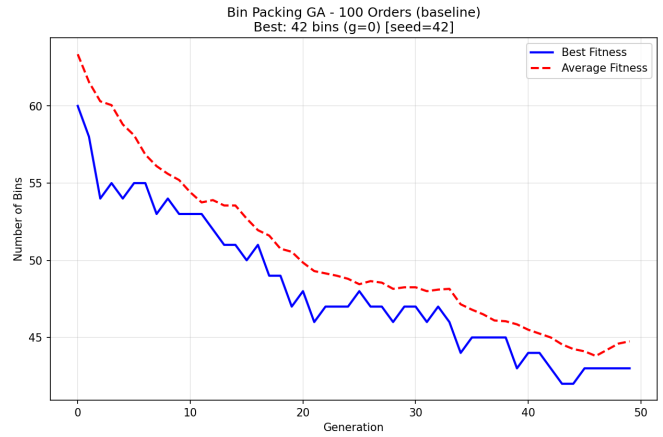


Fig. 4: Baseline: 100 orders

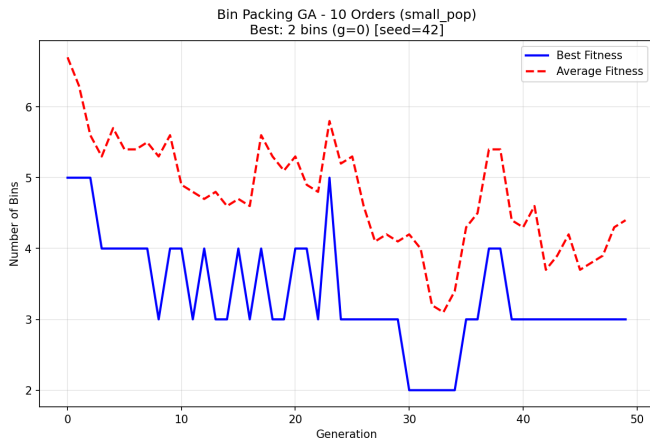


Fig. 5: Small population: 10 orders

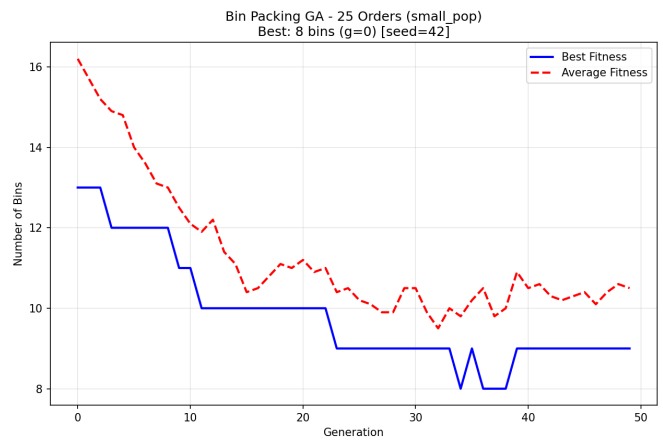


Fig. 6: Small population: 25 orders

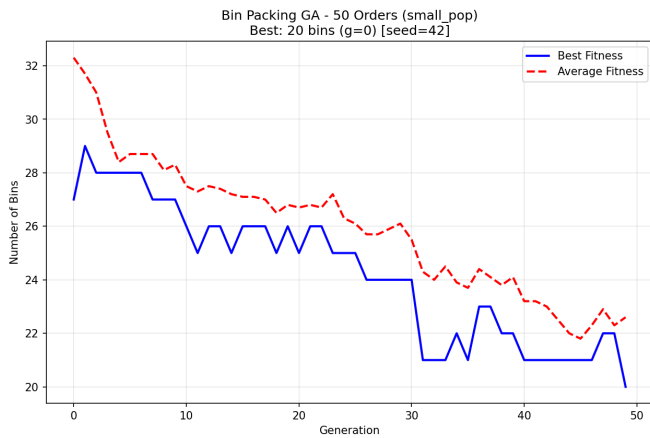


Fig. 7: Small population: 50 orders

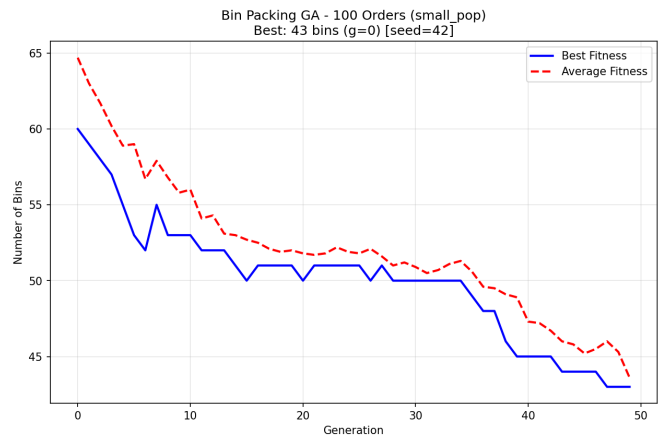


Fig. 8: Small population: 100 orders

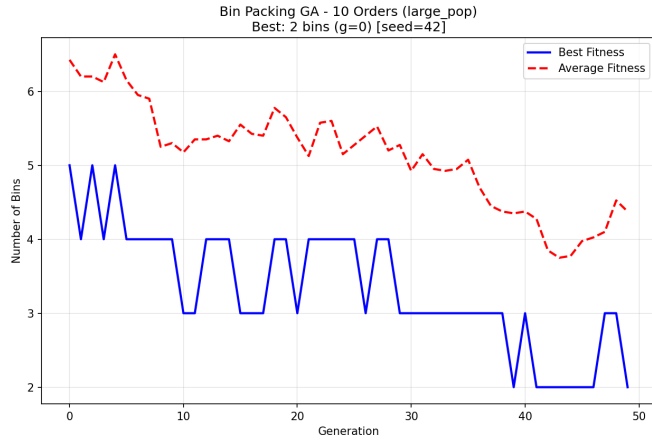


Fig. 9: Large population: 10 orders

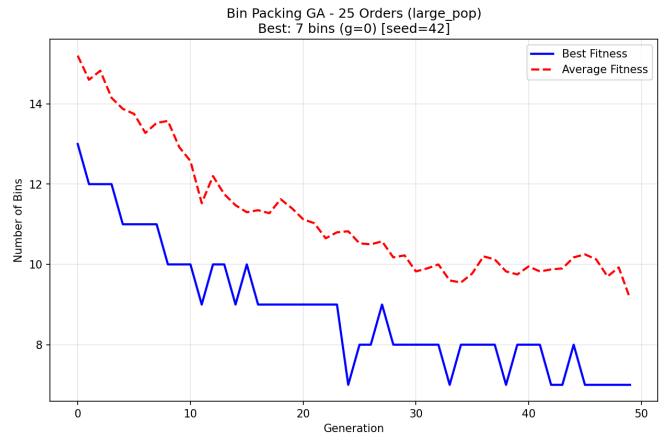


Fig. 10: Large population: 25 orders

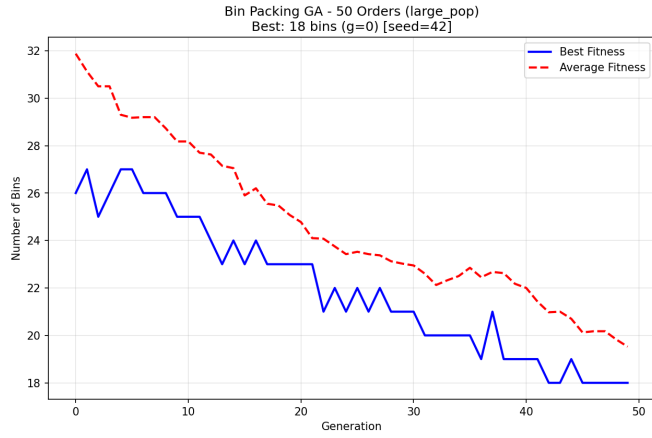


Fig. 11: Large population: 50 orders

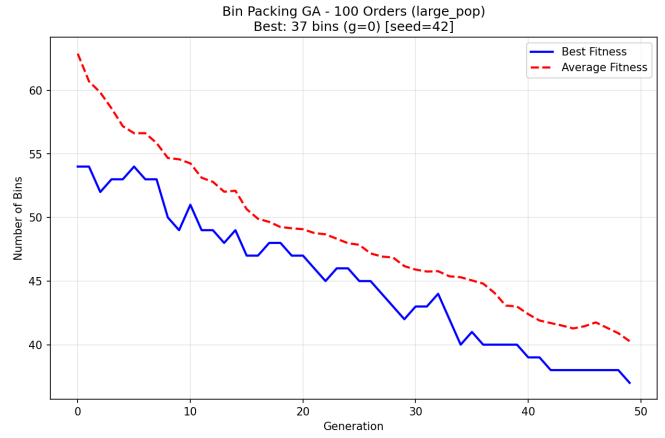


Fig. 12: Large population: 100 orders

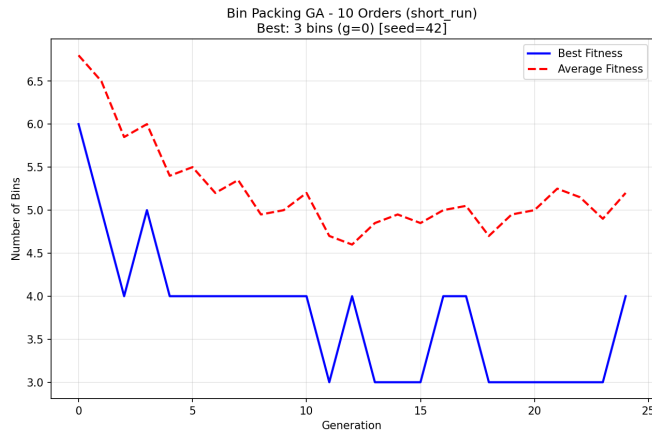


Fig. 13: Short run: 10 orders

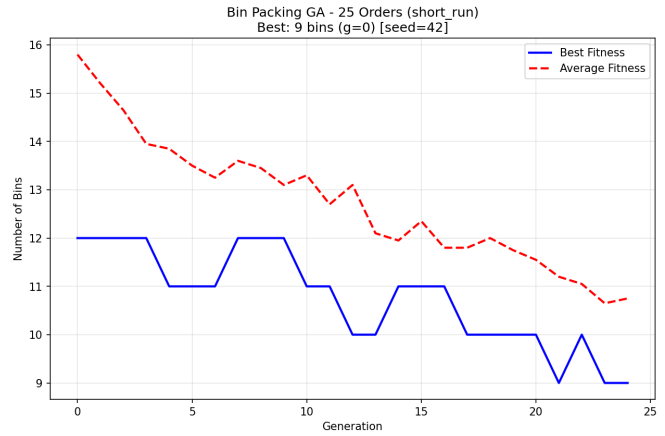


Fig. 14: Short run: 25 orders

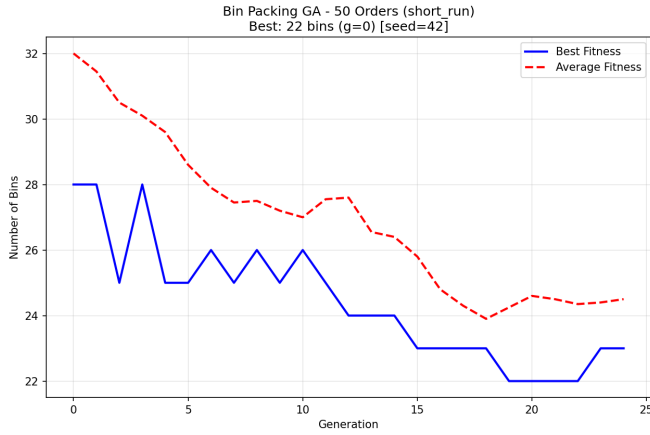


Fig. 15: Short run: 50 orders

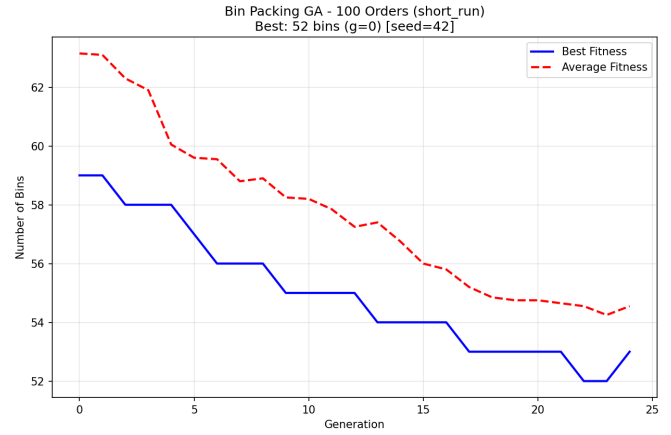


Fig. 16: Short run: 100 orders

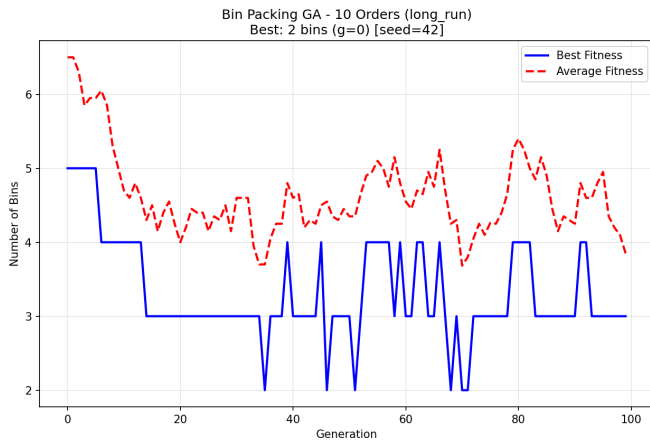


Fig. 17: Long run: 10 orders

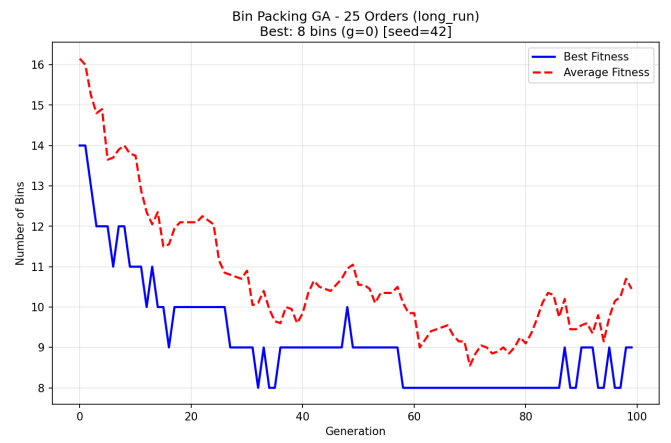


Fig. 18: Long run: 25 orders

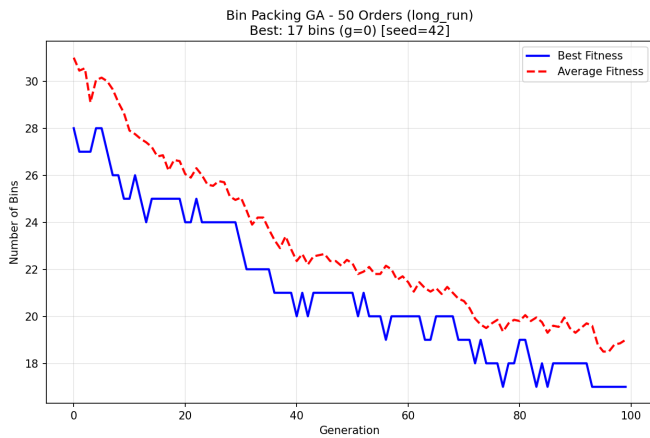


Fig. 19: Long run: 50 orders

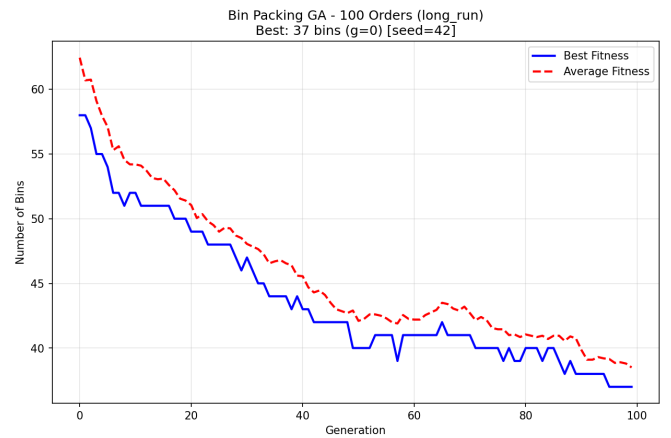


Fig. 20: Long run: 100 orders

D. Best Solutions

Table I summarizes the best solutions found across all configurations tested with seed 42.

TABLE I: Best solutions achieved for each configuration

Configuration	10 orders	25 orders	50 orders	100 orders
Baseline (pop=20, gen=50)	3	7	18	42
Small pop (pop=10, gen=50)	2	8	20	43
Large pop (pop=40, gen=50)	2	7	18	37
Short run (pop=20, gen=25)	3	9	22	52
Long run (pop=20, gen=100)	2	8	17	37

The following tables detail the bin-wise packing configurations for the baseline solutions:

TABLE II: Baseline packing configuration for 10 orders

Bin	Items (weights in kg)	Total (kg)
3	#1(1.49), #4(1.60), #8(0.18)	3.27
6	#0(1.22), #3(1.57), #5(1.19), #6(1.89), #9(0.60)	6.47
9	#2(0.60), #7(0.12)	0.72

TABLE III: Baseline packing configuration for 25 orders

Bin	Items (weights in kg)	Total (kg)
0	#17(0.23), #20(1.40), #23(1.15)	2.78
3	#7(0.11), #14(1.38), #22(1.65)	3.14
4	#13(0.06), #16(0.83)	0.89
17	#8(1.94), #9(0.88)	2.82
19	#1(0.57), #2(1.23), #3(1.79), #5(1.29), #6(0.50), #19(1.63)	7.01
21	#0(1.87), #10(1.73), #15(0.35)	3.96
22	#4(1.28), #11(1.01), #12(1.08), #18(1.98), #21(1.15), #24(0.39)	6.90

V. Q5: ANALYSIS AND COMPARISON

A. Convergence Analysis

Performance varied significantly across configurations and problem sizes. Testing with seed 42 revealed clear patterns.

For the smallest problem (10 orders), multiple configurations found 2-bin solutions, including the small population (10 individuals) and long run (100 generations). Small problems proved easy to solve regardless of parameter settings.

TABLE IV: Baseline packing configuration for 50 orders

Bin	Items (weights in kg)	Total (kg)
1	#2(0.30), #6(1.75), #40(0.17)	2.21
9	#13(1.07), #15(1.38), #48(0.90)	3.35
10	#26(1.41)	1.41
14	#0(1.67), #9(1.92), #12(1.59), #35(0.34), #46(0.92), #49(0.54)	6.98
15	#3(1.44), #44(1.73)	3.18
17	#19(1.97), #21(1.96)	3.93
20	#5(1.82), #37(1.69)	3.51
21	#1(0.93), #20(1.67), #28(0.81), #42(0.41), #43(0.70)	4.52
23	#22(0.46), #25(0.21)	0.67
31	#31(0.30)	0.30
32	#14(1.39), #27(1.78)	3.17
34	#17(0.19), #18(0.92), #23(1.36), #33(1.82)	4.28
36	#30(0.58)	0.58
37	#4(0.47), #11(1.46)	1.93
39	#29(0.98), #34(1.94), #39(1.44), #41(0.71), #47(1.91)	6.99
41	#24(0.08), #32(1.08), #36(1.30), #45(0.18)	2.64
48	#10(0.65), #16(1.75)	2.40
49	#7(0.58), #8(1.52), #38(0.96)	3.06

Population size mattered more for larger problems. With 100 orders, the small population (10 individuals) needed 43 bins versus the baseline’s 42, while the large population (40 individuals) achieved 37 bins - a significant improvement. More diversity helped on complex problems.

Generation count showed similar effects. Short runs (25 generations) performed poorly, especially on large problems—using 52 bins for 100 orders compared to 42 for the baseline. Long runs (100 generations) performed best overall, achieving 17 bins for 50 orders and 37 bins for 100 orders.

Both population size and generation count improved solution quality, with larger problems benefiting most from increased resources in either dimension.

B. Feasible Solution Analysis

All configurations found feasible solutions ($g=0$) from generation 0 across all problem instances. This was expected given the problem parameters: with items weighing 0-2 kg and bins holding 10 kg, random initialization rarely produces overloaded bins.

Since each of N items can be assigned to any of N bins, items tend to spread out during random initialization. Even when multiple items land in the same bin, reaching the 10 kg limit requires at least 5 items, which is unlikely when items distribute randomly across many bins.

As a result, the genetic algorithm focused entirely on minimizing the number of bins used rather

TABLE V: Baseline packing configuration for 100 orders

Bin	Items (weights in kg)	Total (kg)
1	#71(0.75), #94(0.78)	1.53
4	#60(1.30)	1.30
7	#3(1.65), #20(0.70), #69(1.74), #74(0.67)	4.77
8	#1(0.50), #34(1.23)	1.74
12	#23(1.78)	1.78
17	#14(1.73), #28(1.66), #58(0.31)	3.71
20	#65(0.60)	0.60
21	#67(0.51), #84(1.94), #96(0.43)	2.88
29	#38(0.34), #73(1.49), #80(1.33)	3.17
31	#21(1.40), #68(0.58)	1.98
34	#22(1.56), #35(0.94)	2.50
36	#55(0.01)	0.01
37	#0(1.49), #47(1.99), #49(0.87), #54(0.47), #64(1.92)	6.74
42	#42(1.95), #70(1.01), #88(0.25)	3.21
44	#56(0.88), #61(1.77)	2.65
46	#11(0.31), #19(0.10), #75(1.55), #78(1.02), #91(1.40), #92(0.07)	4.45
49	#16(1.64), #53(0.09), #63(0.92)	2.64
51	#29(0.17), #89(1.73)	1.91
52	#13(1.51), #46(1.07)	2.58
53	#18(0.35), #39(1.07)	1.42
55	#27(1.56), #52(1.66), #72(1.18)	4.40
57	#12(0.33), #40(1.25), #43(0.28), #85(1.46)	3.32
59	#24(0.03)	0.03
61	#44(0.79), #57(1.98)	2.77
62	#10(1.37), #51(0.53), #82(0.92), #83(0.89)	3.71
66	#31(1.12), #45(1.80)	2.92
68	#7(1.81), #9(0.21), #37(1.26), #95(1.82)	5.11
69	#59(1.73), #79(1.91), #99(0.12)	3.76
71	#33(0.33), #90(0.68), #98(0.61)	1.63
77	#25(0.26), #26(0.71), #50(1.29)	2.26
78	#4(1.95)	1.95
81	#15(0.05), #77(0.71)	0.77
82	#6(0.63), #17(1.64)	2.28
83	#30(1.28)	1.28
85	#81(0.95), #93(0.01)	0.97
86	#32(1.45), #36(0.85)	2.29
89	#8(0.16), #76(0.86)	1.01
90	#41(0.47), #62(0.46), #87(1.37)	2.30
91	#48(1.27)	1.27
93	#2(1.97), #66(0.74)	2.71
95	#86(1.87), #97(0.55)	2.43
97	#5(0.52)	0.52

than finding feasible solutions. The constraint handling mechanisms in the tournament selection were implemented but never needed, as no infeasible solutions appeared during evolution. This shows that for bin packing with generous capacity relative to item size, the challenge is optimization, not constraint satisfaction.

VI. CONCLUSION

This study successfully implemented...