NOVA IMS Information Management School

2024/2025

COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

WEDDING SEATING OPTIMIZATION

MASTER IN DATA SCIENCE AND ADVANCED ANALYTICS

GROUP AW

- AFONSO LOPES, 20211540
- ANA ISABEL DUARTE, 20240545
- PEDRO CAMPINO, 20240537
- RITA MATOS, 20211642

GITHUB LINK REPOSITORY

Table of Contents

1. Introduction 1
2. Problem Definition 1
2.1. Formal Definition of the Optimization Problem 1
2.2. Solution Representation 1
2.3. Search Space 2
2.4. Fitness Function 2
3. Implementation of Genetic Operators 2
3.1. Selection 2
3.2. Crossover 3
3.3. Mutation 3
4. Performance Analysis 3
4.1. Initial Configuration Tuning 3
4.2. Focused Probability Tuning 4
4.3. Comparative Evaluation of Top Configurations 4
4.4. Algorithm Comparison 5
5. Final Results 5
6. Conclusion 5
7. References 6
8. Anex 7

1. Introduction

According to a survey conducted by Zola on over 4,000 engaged couples, 94% reported feeling stressed while planning their wedding [1]. Additionally, with the average U.S. wedding costing $35,000 [2], and $2,100 of that being spent on a wedding planner [3], it's clear that logistics play a major role in both the emotional and financial components of the event. Among these tasks, the seating arrangement plays a crucial role in the overall success of a wedding. As such, developing an effective seating arrangement that not only enhances guest satisfaction but also automates this complex task emerges as a valuable and impactful problem to solve.

This project aims to address that challenge using Genetic Algorithms - a class of evolutionary optimization techniques inspired by natural selection. The goal is to automatically generate seating arrangements that maximize guest satisfaction, based on a relationship matrix that quantifies how well each pair of guests gets along. In doing so, we also explore how different configurations of genetic algorithm components - including selection methods, crossover operations, and mutation techniques - affect the quality and efficiency of the solutions.

While optimizing seating arrangements can significantly improve guest experience, it relies on the availability of detailed data regarding interpersonal relationships. In real-world events like weddings, collecting such information for every pair of guests can be challenging or impractical, especially when dealing with large guest lists, privacy concerns, or limited knowledge of interpersonal dynamics.

2. Problem Definition

2.1. Formal Definition of the Optimization Problem

The objective of this problem is to find an optimal seating arrangement for a wedding, where 64 guests must be assigned to 8 tables, with 8 guests per table. Each guest pair has a relationship score that reflects how positively or negatively they interact. The goal is to maximize the overall happiness across all tables, defined by the sum of pairwise relationship scores among guests seated at the same table.

This is a combinatorial optimization problem with the following constraints:

- Each guest must be assigned to exactly one table.
- Each table must have exactly 8 guests.
- No guest can appear more than once or be left out.

The problem is suitable for a Genetic Algorithm (GA) approach because of the vast number of possible configurations and the lack of a straightforward analytical solution.

2.2. Solution Representation

In this problem, a solution is represented as a seating arrangement for the wedding guests. Each solution consists of a list of 8 subgroups (tables), where each subgroup contains exactly 8 guests. The guests are identified by unique IDs, and each table represents a group of guests assigned to

that particular table. The arrangement ensures that each guest is assigned to exactly one table, and each table contains a fixed number of 8 guests.

This representation makes it straightforward to manipulate and evaluate, particularly when applying genetic operators such as crossover and mutation. The design ensures that valid solutions are preserved, and no guest is ever seated at multiple tables or left out.

2.3. Search Space

The search space for this problem consists of all possible seating arrangements for the 64 guests across 8 tables, each containing exactly 8 guests. The number of possible seating arrangements is extremely large, making it infeasible to explore every combination exhaustively.

Each solution involves selecting 8 guests from a pool of 64 and assigning them to one of the 8 tables. Therefore, the total number of distinct seating arrangements is given by the multinomial coefficient:

$$\frac{64!}{(8!)^8 \cdot 8!} = 450538787986875167583433232345723106006796340625$$

2.4. Fitness Function

The relationship between two guests is quantified by a value from a relationship matrix, where positive values indicate a good relationship and negative values indicate a poor relationship. For each table in the seating arrangement, the fitness function sums up the relationship scores for all possible pairs of guests seated at that table. This is done for all tables, and the total happiness score is calculated by summing the individual table scores.

The fitness function aims to maximize this total happiness score, so a higher fitness value corresponds to a better seating arrangement. In mathematical terms, the fitness F of a seating arrangement is computed as:

$$F = \sum_{tables} \sum_{(a,b) \in \binom{table}{2}} relationship(a, b)$$

Where $\binom{table}{2}$ represents all unique pairs of guests seated at the same table and $relationship(a, b)$ is the relationship score between guests a and b.

The fitness function provides a clear metric for evaluating solutions, enabling the genetic algorithm to prioritize arrangements that maximize guest happiness while adhering to the problem constraints.

3. Implementation of Genetic Operators

3.1. Selection

In our project, we implemented three selection algorithms which were taught in class: Fitness Proportionate Selection, Ranking Selection and Tournament Selection. Out of these, Ranking Selection yielded the best results. This is likely because Ranking Selection assigns selection

probabilities based on an individual's rank rather than raw fitness values. This means higher-ranked solutions have a greater chance of being chosen, but lower-ranked or even negative-fitness individuals retain some probability of selection—helping avoid premature convergence.

3.2. Crossover

To introduce genetic diversity and effectively combine parent solutions, we implemented several crossover operators tailored to the constraints of the wedding seating problem.

Firstly, we designed a custom table-level crossover operator. Instead of operating on a flat list of guests, this method performs one-point crossover at the table level. As this can lead to duplicate or missing guests, a repair mechanism is applied to ensure all 64 guests are present exactly once by replacing duplicates with the missing individuals.

Next, we adapted Partially Matched Crossover (PMX) to our problem. Each solution is first flattened into a linear representation, then PMX is applied, and the result is then restructured back into tables. Any duplicates introduced during this process are replaced by mapping them to the missing guests, preserving the uniqueness constraint.

Finally, we implemented Order Crossover (OX) [4], a standard operator for permutation problems. OX preserves the relative order and position of elements by copying a segment from one parent and filling the remaining positions with genes from the other parent in the order they appear, skipping duplicates. To better ilustrate the mechanics of OX, we created an explanatory video, accessible via the following link.

3.3. Mutation

To maintain diversity and explore new areas of the solution space, we adapted several mutation operators to respect the structure and constraints of the wedding seating problem.

Since swapping individuals seated at the same table does not affect the solution, we modified two standard mutations. Swap Mutation was adjusted to ensure that the two selected individuals for swapping come from different tables. Inversion Mutation was altered so that the selected subsequence for reversal does not both start and end within the same table.

We also implemented a version of Scramble Mutation [5], which typically permutes a randomly selected group of individuals. In our adaption, one random guest from each table is chosen, and the resulting group is permuted such that no individual returns to their original position - a derangement (Figure 1).

4. Performance Analysis

This section presents an evaluation of our Genetic Algorithm (GA) approach, both in isolation and in comparison with other optimization strategies. The analysis is divided into several phases:

4.1. Initial Configuration Tuning

In the first phase of our performance analysis, we focused on identifying effective combinations of genetic algorithm components. Specifically, we tested various selection algorithms, mutation

and crossover operators, explored the impact of enabling elitism, and assessed how different values for crossover and mutation probabilities influenced performance.

To avoid bias introduced by arbitrarily chosen fixed values, we employed a structured grid search approach for tuning probabilities. Instead of using static values, we generated one random number from each of the following intervals: [0, 1/3], [1/3, 2/3], and [2/3, 1], both for the crossover probability and the mutation probability. This strategy ensured a broad and balanced exploration of the [0, 1] range, without favoring specific values. Each configuration was evaluated based on the quality of solutions produced over multiple runs, and the top 5 best-performing combinations were selected for further refinement in the next phase of our analysis.

4.2. Focused Probability Tuning

After conducting an initial grid search to explore all possible parameter configurations, we proceeded with a more targeted tuning phase focused specifically on the probability parameters. The goal was to refine the performance of the genetic algorithm by narrowing the search around the most promising probability ranges identified earlier.

To do this, we took the broader interval in which a probability value had shown strong results and subdivided it into three equally sized subranges. From each of these narrower intervals, we sampled one random value. This maintained the variability of random selection while concentrating the search on the most effective regions of the parameter space.

This iterative narrowing allowed finer control over the algorithm's behavior, revealing how performance responds to subtle parameter changes without relying on arbitrary values.

The refined top configurations revealed consistent patterns: all employed ranking selection, most used table-level crossover, swap mutation, and elitism. Mutation probabilities were generally low (below 0.35, with several close to this threshold), while crossover probabilities were more varied but typically high, often above 0.75 (Table 1).

4.3. Comparative Evaluation of Top Configurations

After narrowing down and fine-tuning the crossover and mutation probabilities, we selected the five best-performing configurations from the grid search phase for further evaluation. Each of these configurations was executed 30 times to ensure that the results were statistically representative and suitable for robust comparative analysis.

Prior to performing the statistical testing, we visually inspected the average fitness evolution curves for each configuration. We observed that configurations 2, 3, and 4 achieved the highest average fitness by the final generation (Figure 2). Although these three configurations performed very similarly and reached almost the same fitness in the end, a closer look at each curve revealed that configurations 3 and 4 exhibited greater stability across runs (Figure 4, 5 and 6).

While these visual insights were informative, a more rigorous and objective evaluation was necessary to validate the observed differences. To this end, we applied the Wilcoxon signed-rank test to compare the performance of the different genetic algorithm configurations. This non-parametric test is well-suited for our case because it does not assume a normal distribution, which aligns with the variability often seen in stochastic algorithms. By comparing the fitness

distributions from multiple runs, the test allowed us to determine whether observed differences were statistically significant or due to random chance, helping us identify the most consistently effective configuration. The results aligned with our visual inspection, and configuration 4 was considered the best from a statistical point of view.

4.4. Algorithm Comparison

According to the No Free Lunch Theorem, Leonardo Vanneschi says that "averaged over all possible problems, every algorithm has the same performance; hence no single method can outperform all others on every problem" [6]. This implies that algorithm effectiveness is highly problem-specific and must be evaluated empirically. With this in mind, we compared the performance of three algorithms - Genetic Algorithm (GA), Hill Climbing (HC), and Simulated Annealing (SA) - on the wedding seating optimization problem.

Before the comparison, we performed a parameter tuning process for Simulated Annealing using grid search, allowing us to identify a configuration that consistently achieved good performance. We then compared this tuned version of SA with the Hill Climbing algorithm and the best configuration we found for GA. All three algorithms were runned under the same conditions, including an equal number of maximum iterations, to ensure fairness in the evaluation. Each algorithm was executed 30 times independently to obtain statistically meaningful results and reduce the influence of randomness.

5. Final Results

After conducting a thorough comparison of the three algorithms (GA, HC and SA), we analyzed their performance for our problem (Figure 8).

Across multiple independent runs, the GA consistently produced the highest fitness values, with scores ranging from 79,300 to 81,700, significantly outperforming both HC and SA. Hill Climbing yielded fitness scores between 67,800 and 79,100, while Simulated Annealing's results varied from 66,600 to 76,100. These results clearly indicate that GA was the most effective algorithm at finding high-quality seating arrangements under the tested conditions.

As a final step, we applied Hill Climbing to the best GA solution to check for further improvements. However, it had no effect, suggesting the GA had already reached a local optimum, with a best fitness of **81,700**.

Qualitatively, the seating plans generated by GA better respected guest compatibility and social preferences, effectively grouping friends and minimizing conflict pairs. This demonstrates the practical application of GA for real-world seating arrangement problems, where social harmony and guest satisfaction are essential.

6. Conclusion

This project investigated the use of Genetic Algorithms (GA) to optimize wedding seating arrangements, aiming to maximize overall guest satisfaction based on interpersonal relationship scores. By representing seating plans as grouped assignments and applying tailored genetic operators, the GA effectively navigated the vast search space to find high-quality solutions.

Comparative analysis against Hill Climbing and Simulated Annealing demonstrated that GA consistently achieved superior fitness values, proving the robustness of our implementation. The results show that the GA not only improves solution quality but also maintains computational efficiency suitable for practical applications.

To assess the difficulty of the problem and establish a baseline, we conducted random trials by generating and evaluating one million seating arrangements. The highest fitness score observed during these trials was 34,700, while the average fitness consistently converged around 12,200. These results underscore the impracticality of exhaustively searching the solution space using random sampling alone, due to its vast combinatorial complexity. In contrast, our Genetic Algorithm was able to discover solutions with fitness scores as high as 81,700, clearly demonstrating its superior ability to efficiently explore and exploit the search space.

In summary, the project validates Genetic Algorithms as a powerful tool for complex social optimization problems like seating arrangements, providing a scalable and adaptable method that can be extended or refined for even better performance in future work.

7. References

[1] "The First Look Report 2023 - Zola Expert Wedding Advice." Accessed: May 22, 2025. [Online]. Available: https://www.zola.com/expert-advice/the-first-look-report-2023?clickid=Qa1UeQRHXxyPRxwW6uTkeXNXUksTlOz9ByRxRE0&irgwc=1&utm_medium=Affiliate&utm_source=Impact&utm_campaign=249354&affsrc=1&pkey=Affiliate

[2] "Chart: How Much Do Weddings Cost Around the World? | Statista." Accessed: May 22, 2025. [Online]. Available: https://www.statista.com/chart/32638/average-cost-of-a-wedding/

[3] "Average costs for a wedding in the U.S. in 2023, by item| Statista." Accessed: May 22, 2025. [Online]. Available: https://www.statista.com/statistics/254722/average-costs-for-a-wedding-by-item/

[4] "(28) Crossovers in genetic algorithms | LinkedIn." Accessed: May 22, 2025. [Online]. Available: https://www.linkedin.com/pulse/crossovers-genetic-algorithms-ali-karazmoodeh-tthjf/

[5] "(28) Mutations in genetic algorithms | LinkedIn." Accessed: May 22, 2025. [Online]. Available: https://www.linkedin.com/pulse/mutations-genetic-algorithms-ali-karazmoodeh-u94pf/

[6] L. Vanneschi and S. Silva, "Natural Computing Series Lectures on Intelligent Systems".

8. Anex

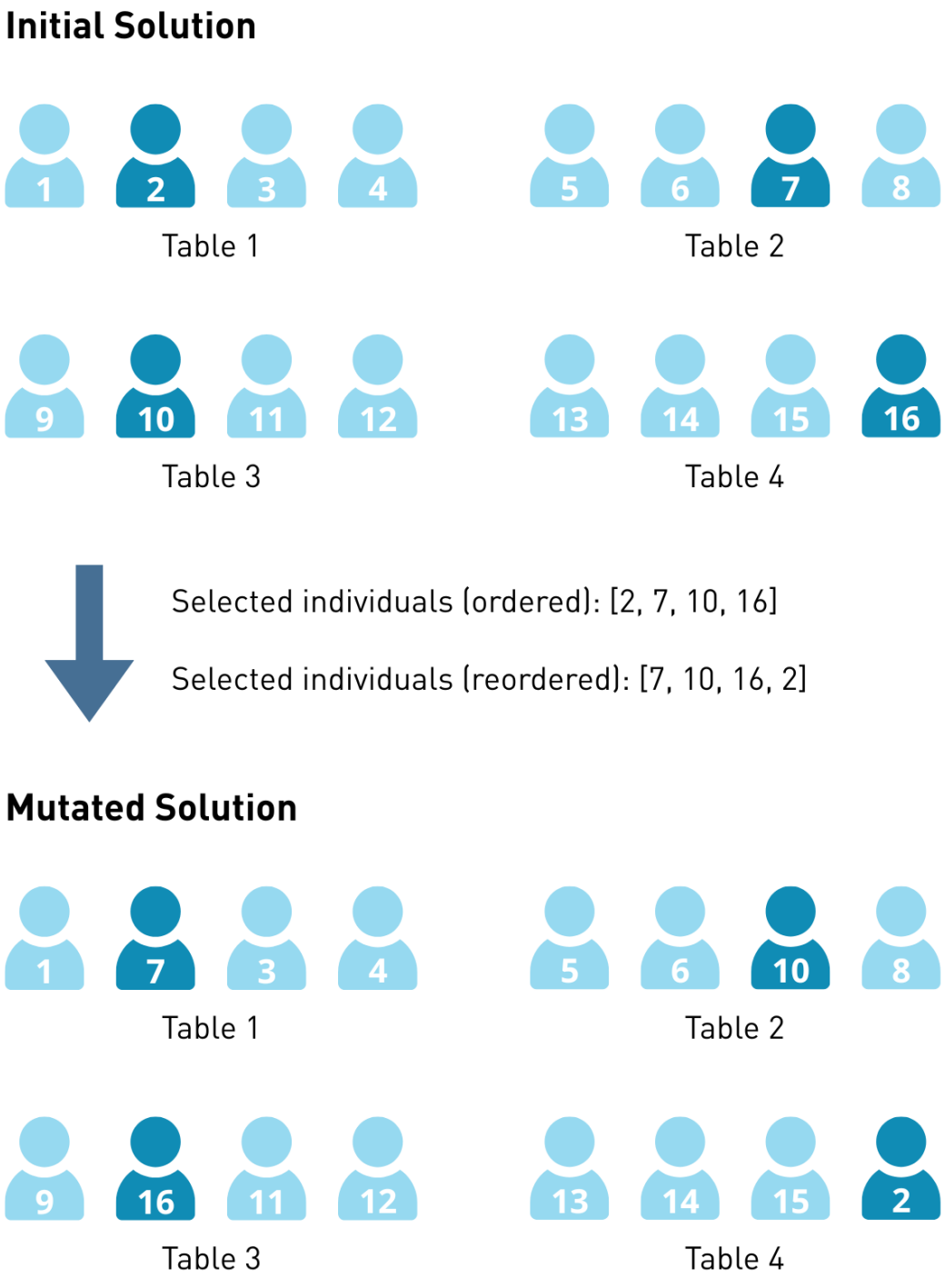

Figure 1 - Scramble Mutation Schema

Config.	Fitness	Selection	Crossover	Mutation	Mutation Probabilty	Crossover Probability	Elitism
1	65400	Ranking	Table Level	Swap	0.1373	0.7984	False
2	67200	Ranking	Table Level	Swap	0.3415	0.5134	True
3	68300	Ranking	Partially Matched	Swap	0.3415	0.7984	True
4	65100	Ranking	Table Level	Swap	0.2322	0.9780	True
5	72500	Ranking	Table Level	Inversion	0.3415	0.4001	True

Table 1 - Top 5 Best-Performing GA Configurations

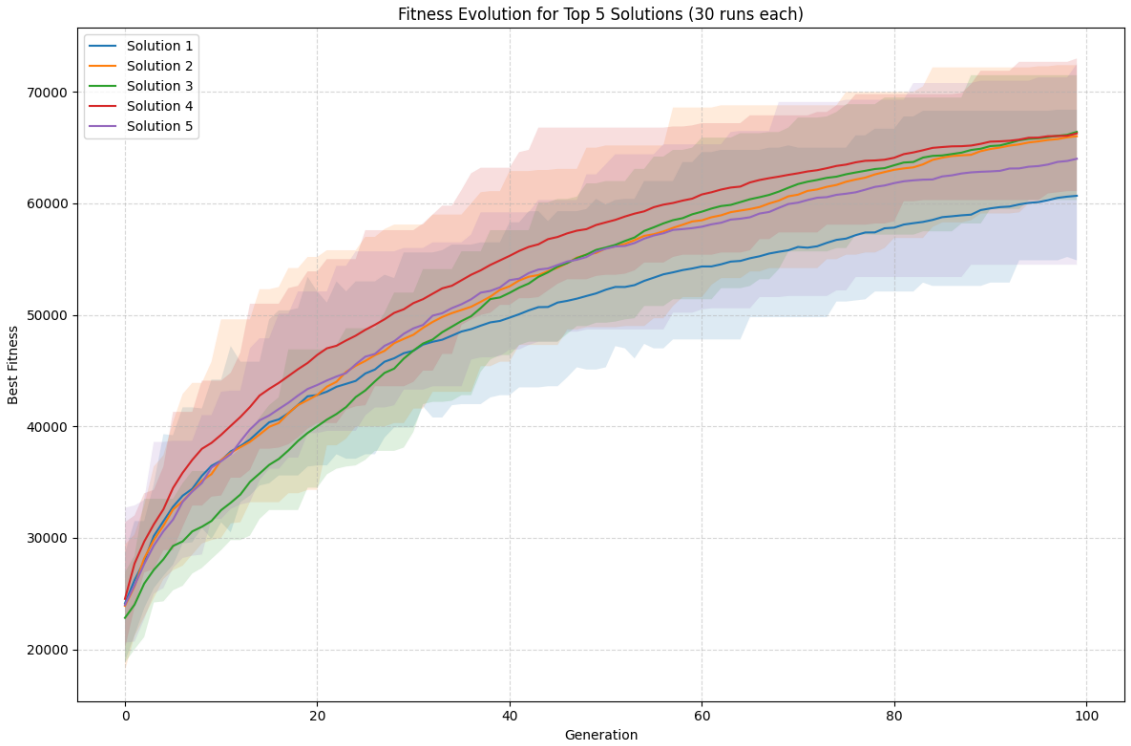

Figure 2 - Fitness Evolution of Top 5 Best-Performing GA Configurations

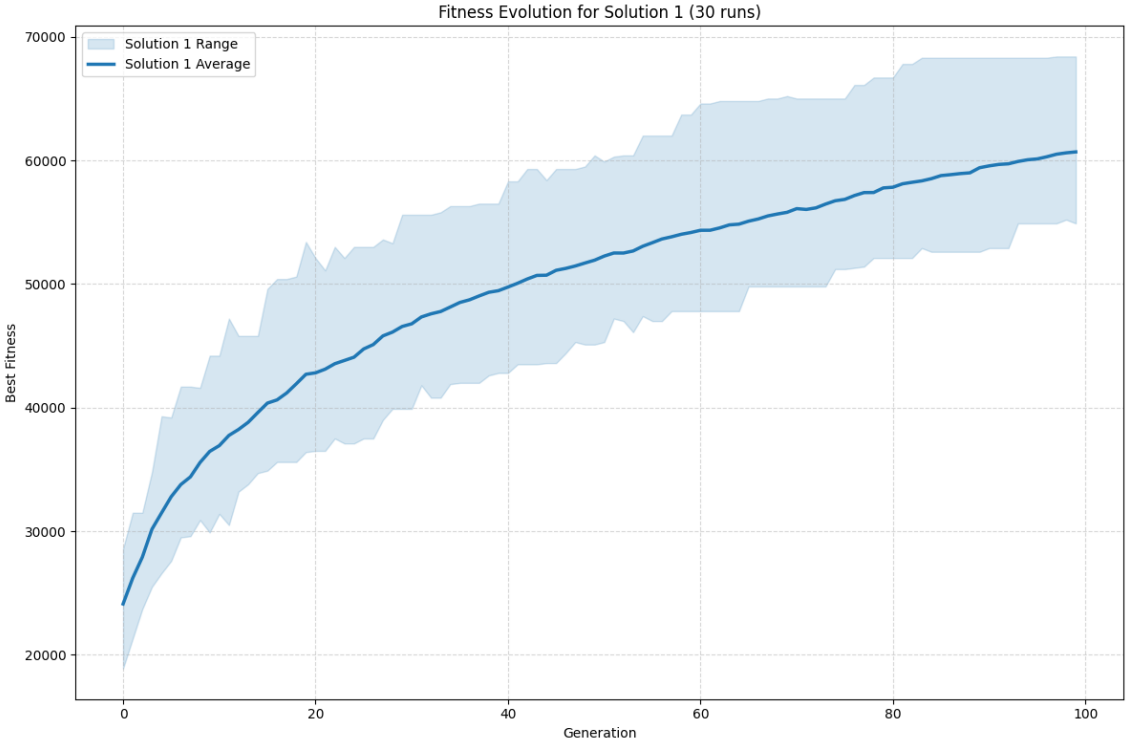

Figure 3 - Fitness Evolution of Configuration 1

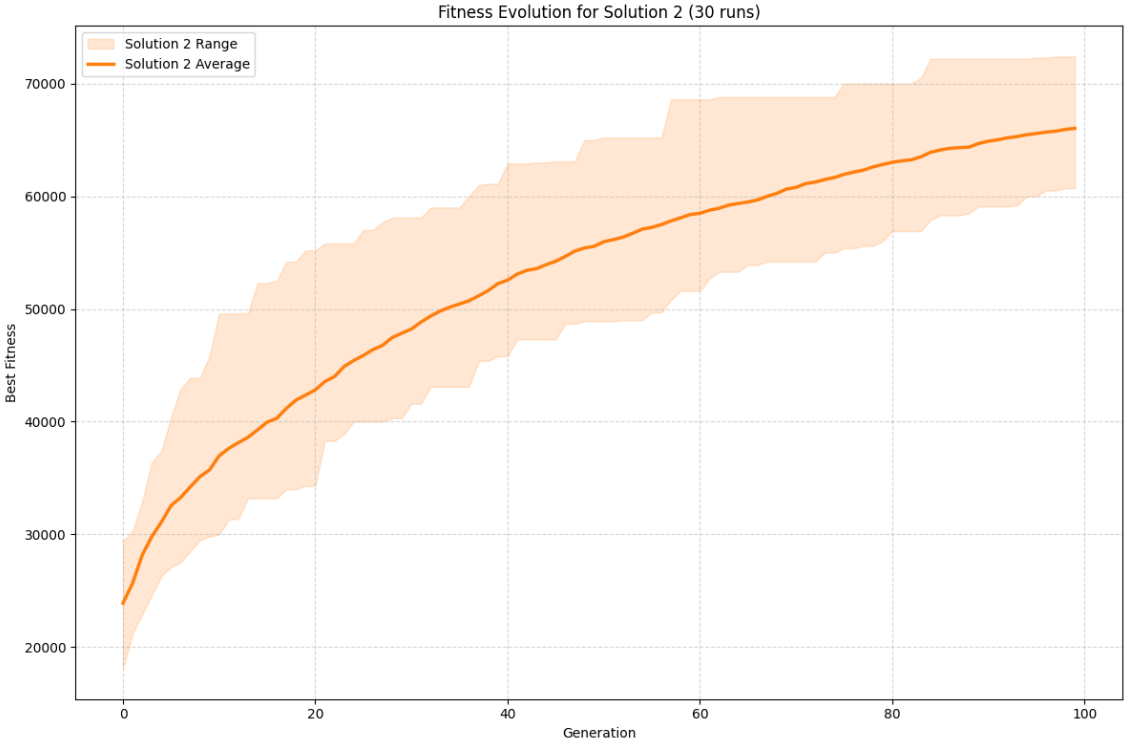

Figure 4 - Fitness Evolution of Configuration 2

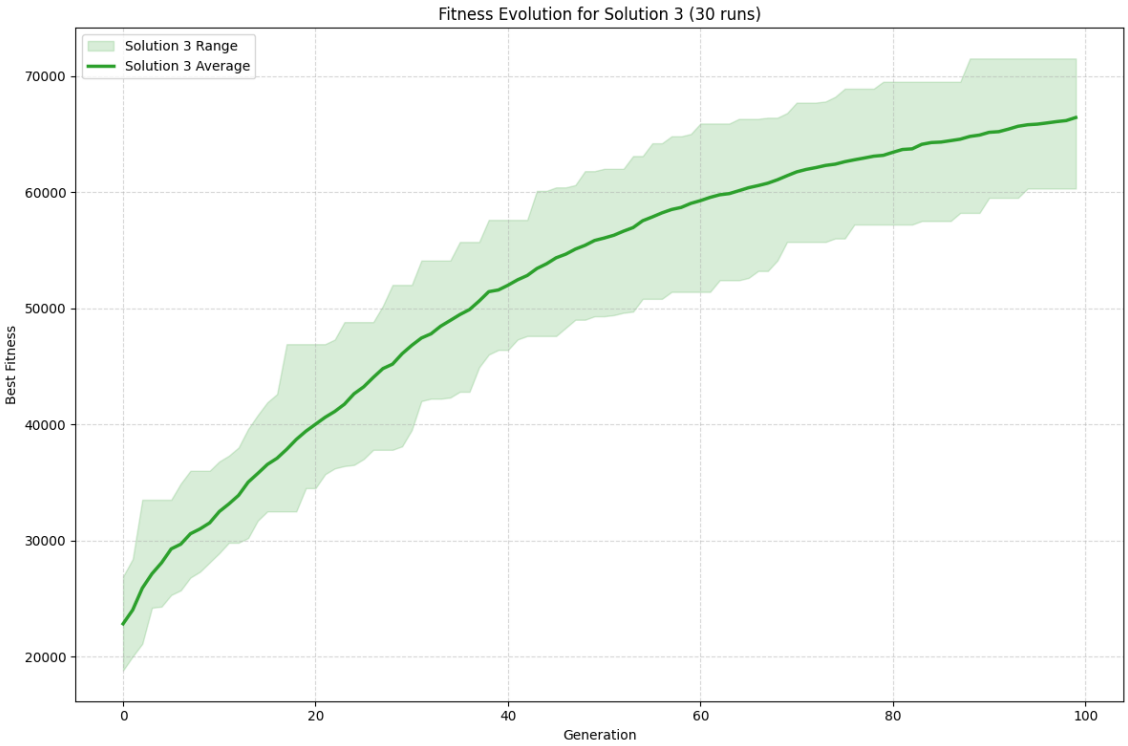

Figure 5 - Fitness Evolution of Configuration 3

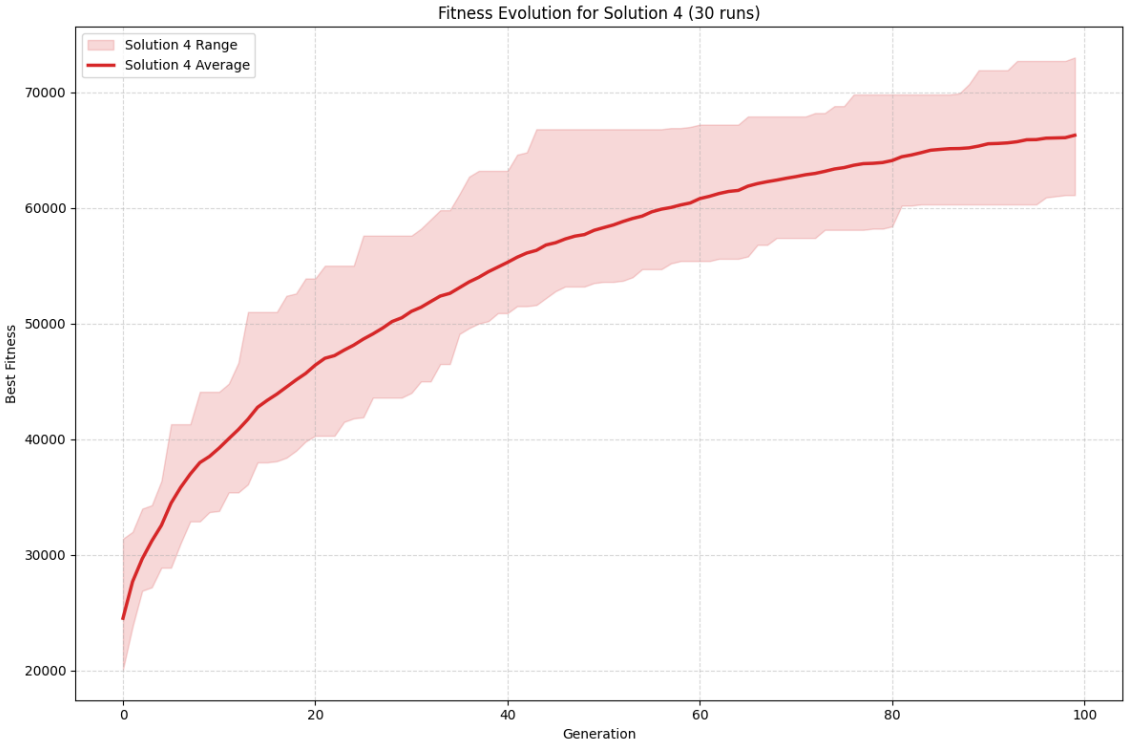

Figure 6 - Fitness Evolution of Configuration 4

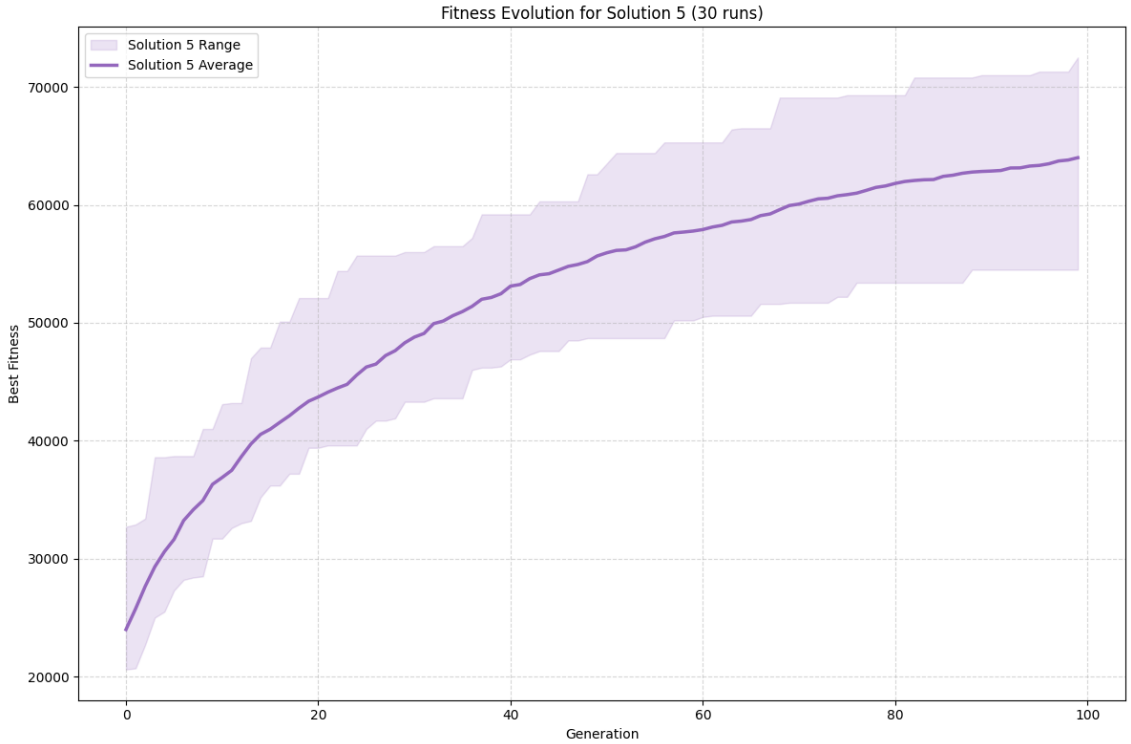

Figure 7 - Fitness Evolution of Configuration 5

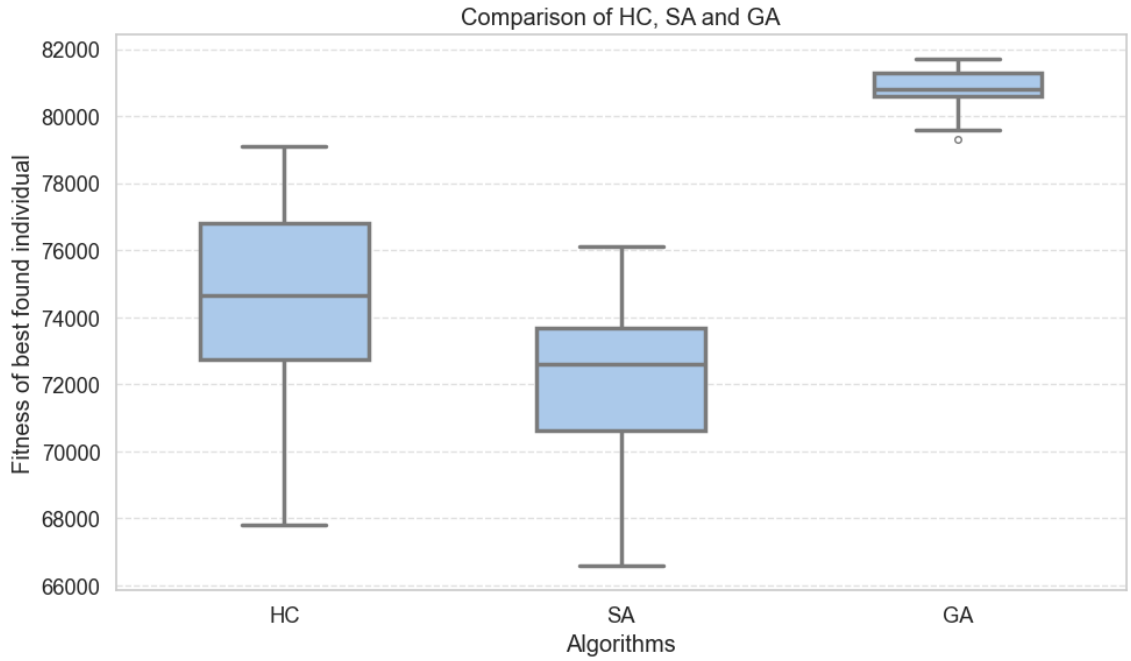

Figure 8 - Fitness Boxplots of different Optimization Algorithms (HC, SA and GA)