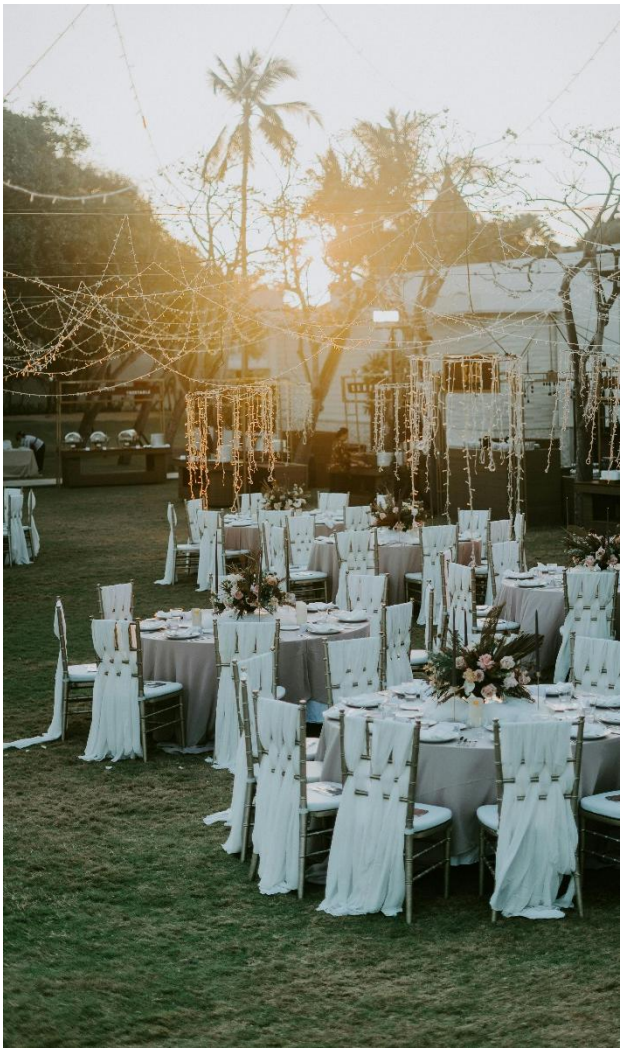
**Computational Intelligence for Optimization Project**
Professors: Leonardo Vanneschi and Inês Magessi

Master in Data Science and Advanced Analytics
**NOVA Information Management School**
Universidade Nova de Lisboa



# Wedding Seating Optimization


**Group AU**
Joana Rodrigues, 20240603
Matilde Street, 20240523
Mara Simões, 20240326
Rafael Silva, 20240511


Github Project Link: https://github.com/joanaramosrodrigues/CIFO_Project

# Table of contents

# 1. Introduction

Deciding where to seat guests can be complex, especially considering the different relationships between them, as some get along very well, and others don't at all. In weddings there are numerous logistical aspects to be dealt with, therefore being able to have an automatic seat allocation can really help the organization.

This project solves the issue of seating 64 people at 8 tables, each with 8 seats. The need for automation lies in the large number of combinations of how to seat the guests (the search space corresponds to more than 4.6 billion combinations, each representing a unique seating allocation of the guests across the tables). The code was implemented in a generalizable way, allowing it to handle any similar seating problem as long as the number of guests per table is the same.

Genetic Algorithm is a useful probabilistic search algorithm based on natural selection [1]. This approach is good for problems like this, where the number of solutions is too high for exhaustive search as this explores the solution space and generates different and improved seating combinations over time. The challenge of this work lies in the risk of getting stuck in local optima, so diversity is maintained through the implementation of different crossover and mutation techniques that allow better combinations to be found.

# 2. Initial Analysis

The dataset is a matrix showing the relationships between different guests, where negative values represent conflicts, and higher values indicate stronger positive relationships.

Since relationships are mutual, the matrix is expected to be symmetric. An initial analysis revealed asymmetries in the matrix, involving guest 54 (the bride's best friend instead of the groom) (Figure 1). To correct this, the matrix was adjusted based on the fitness values of each relationship (Figure 2), ensuring full symmetry.

# 3. Individual Representation and Initialization

Each individual is represented as a list of 8 tables, where each table contains 8 Guest objects. This matrix-like representation (8x8) reflects the full seating arrangement, where each guest is explicitly assigned to a specific table.

Two initialization methods were implemented: random and sequential. Random initialization was chosen by default as it promotes greater diversity in the initial population, which is essential to explore different areas of the solution space and avoid local optima. Sequential initialization fills tables based on the order of guests, resulting in a more structured but less variable setup. The initial randomness helps prevent biased configurations and opens the door to novel guest combinations.

# 4. Genetic Algorithm

To solve the optimization problem using genetic algorithms, we implemented multiple variants of selection mechanisms, crossover, and mutation operators to explore the search space effectively and maintain genetic diversity.

Elitism was implemented to ensure that the best solutions are preserved across generations. In a problem like this, where maintaining well-functioning social groupings is critical, elitism prevents losing good seating arrangements due to random mutations. By carrying the top individuals forward to the next generation, elitism helps the algorithm retain successful partial solutions, improving convergence speed and overall solution quality. This approach stabilizes the search process and increases the likelihood of finding an optimal seating plan.

To promote diversity, fitness sharing and restricted mating were briefly tested. Fitness sharing (applied to FPS, because it was the worst performing selection algorithm) and restricted mating both increased runtime without improving results. Due to their high cost and low benefit both were dropped in favor of optimizing crossover and mutation.

To ensure that all solutions are valid, meaning no guest is repeated or missing and that each table has exactly 8 guests, a verification function was created and applied after all solutions.

## 4.1. Fitness Function

The main objective is creating a seating plan that minimizes social conflicts and maximizes guest comfort. To do this, the total fitness of a solution is achieved by summing the scores of all guest pairs seated at the same table, giving double penalties for negative relationships to emphasize the importance of avoiding tension. As positive relationships increase the total score, the algorithm is made to avoid poor pairings and promote favorable social dynamics, with the goal of maximizing the fitness function.

## 4.2. Selection Mechanisms

Three different selection strategies were implemented and used to choose parents for the algorithm:

- **Fitness Proportionate Selection**: assigns a selection probability proportional to each individual's fitness. Individuals with higher fitness have a greater chance of being selected. As previously mentioned, fitness sharing can be applied to this algorithm but was not used due to its high computational cost.
- **Rank Selection**: Individuals are ranked by fitness, not mattering their differences, then selection probabilities are assigned based on these ranks, reducing sensitivity to large fitness differences.
- **Tournament Selection**: A small group of individuals (tournament size k) is randomly chosen from the population, and the best in this group is selected as a parent. This method is computationally efficient as it avoids sorting and fitness normalization, but it introduces more randomness, and the selection pressure depends directly on the tournament size.

## 4.3. Crossover

Crossover operators take two parent seating plans and produce new ones.

- **Standard One-Point Crossover:** flattens both parents into 1D sequences, selects a random crossover point, and combines the beginning of one parent with the remainder of the other, while ensuring no duplicate guests. The resulting sequence is reshaped into a valid 8x8 seating arrangement.
- **Cycle Crossover:** preserves guest positions by maintaining groups of guests seated together if they are positioned together in both parents. To adapt it to the seating plan, parents are flattened into linear guest lists, cycles are identified based on guest IDs, and offspring are rebuilt into valid 8x8 arrangements, ensuring unique guests and preserving some positional structure.
- **Table Block Crossover:** selects a random block of full tables from one parent and completes the seating plan with guests from the other, preserving original table groupings and avoiding duplicates. This method is particularly effective at maintaining social structures already present in the parents.

## 4.4. Mutation

Mutation operators are responsible for modifying seating arrangements by changing the positions of guests. The following mutation operators were developed in our project:

- **Swap Mutation:** randomly selects two guests from different tables and swaps their seats. This operator is useful for making small, localized improvements in seating compatibility that preserve the 8x8 table structure and guarantee valid solutions.
- **Inversion Mutation:** flattens the entire seating matrix into a 1D list of guests, selects two positions, and inverts the subsequence between them. After mutation, the list is reshaped into a valid 8x8 format. This allows for more disruptive changes that can significantly alter the seating dynamics while maintaining all constraints of the problem.
- **Adaptive Mutation:** selects between swap and inversion mutations depending on whether the individual is highly similar to others in the population. If the solution is very similar to the others, a more intense mutation (inversion) is utilized to enhance diversity. Otherwise, a lighter mutation (swap) is used.

# 5. Results

The creation of 3 different mutations and 3 different crossovers resulted in 9 combinations per selection mechanism, leading to 27 different configurations in total.

Each configuration was run with 200 generations and with a population size of 100 individuals. To evaluate which setup worked best as our genetic algorithm, each combination was run 30 times. This was done to avoid conclusions based on randomness and to get a solid understanding of each configuration's typical behavior and reliability.

From these runs, several plots were generated per selection algorithm (Figures 3–5). The goal of these visualizations was to evaluate the configurations using three key metrics:

- **Mean fitness per generation** – to understand the overall performance trend.
- **Median fitness** – reduces the effect of outliers, giving a more robust measure of performance.
- **Standard deviation** – to evaluate how stable and consistent the performance was across runs.

Since the previous visualizations alone might not be sufficient to clearly identify the best-performing setups, a Wilcoxon signed-rank test was applied to statistically determine whether the differences in performance between configurations were significant and not due to random chance.

From this, five top configurations were selected:

- 2 using **Tournament Selection (TS):** Table Block Crossover with Swap Mutation and Table Block Crossover with Adaptive Mutation
- 1 using **Fitness Proportionate Selection (FPS):** Cycle Crossover combined with Swap Mutation
- 2 using **Rank Selection (RS)**: Table Block Crossover with Adaptive mutation and table block crossover with Swap Mutation.

These five were then directly compared using the same visual metrics (mean, median, and standard deviation). A final Wilcoxon test was performed on just these five configurations, showing that the top 3 (2 TS and 1 RS) were not significantly different from each other (Figure 6). Among these, the combination using **Tournament Selection** and **Table Block Crossover** with **Swap Mutation** stood out by achieving the highest maximum fitness of all runs. This way, it was chosen as our final configuration**.**

The comparison between different implementations clearly showed how the choice of selection, crossover, and mutation methods affects algorithm performance. Configurations using the Tournament Selection proved to be more reliable and consistent with a greater ability to reach high fitness scores. In contrast, setups using Fitness Proportionate Selection performed consistently worse (Figure 6), likely because it can still select weaker individuals, slowing down the algorithm and reducing consistency.

The Table Block Crossover consistently ranked among the top performers by preserving entire table blocks and maintaining meaningful guest groupings. In contrast, the standard one-point crossover was never among the best, likely because the changes it makes are very small and random. Similarly, inversion mutation showed room for improvement, as its disruptive changes often broke useful groupings, reducing effectiveness.

The Table Block Crossover consistently ranked among the top performers, whereas the standard one-point crossover was never among the best (probably because the crossover was not impactful enough), the same with inversion mutation showing signals of possibility of improvement in this.

Although different genetic operators clearly impact performance, there is no notable difference in convergence trends (Figures 3–6). None of the configurations achieved full convergence over the 200 generations.

This final configuration was tuned using Bayesian Optimization to find the best hyperparameters: mutation rate, crossover probability, and the tournament size (k). The best parameters were picked based on the average fitness across the runs (Figure 7).

Using these results, the algorithm was run 30 additional times, and the solution with the highest fitness among these runs was saved as the final seating plan for the wedding guests. With the tuned parameters, the best fitness improved from 79300 to 80300, with no full convergence. When increasing the number of generations from 200 to 300, the algorithm had more time to optimize and reached a final fitness of 80800. The final solution was saved in a pickle file for future use and visualization.

# 6. Comparison with HC and SA

To evaluate the performance of the Genetic Algorithm, Hill Climbing and Simulated Annealing were also tested, each run 30 times (Figure 8).

- **Hill Climbing (HC)** gets stuck fast in local optima once it only accepts better neighbors. In a problem with such a huge search space, it is unable to explore enough solutions. This way it ended up with a much lower average fitness (12,383.33).
- **Simulated Annealing (SA)** achieved better results since it accepts worse solutions, helping it escapes local optima. Still, it didn't quite reach the results of G.A., once contrary to it, S.A. works with a single solution and relies on gradual improvements. It obtained an average fitness of 67,273.33.
- **Genetic Algorithm** performed best by using a population of individuals and genetic operations to explore and improve seating arrangements, keeping the best solutions. This approach is especially effective for complex problems like wedding seating, proving to be the best option for this problem (average fitness of 75,616.67).

# 7. Conclusion

To get the best seating allocation for this wedding, 27 different configurations combining various selection, crossover, and mutation strategies were tested. The best performance came from a Tournament Selection with Table Block Crossover and Swap Mutation. After running a Bayesian search to tune the parameters, it reached a final fitness of 80800 with 300 generations. Even though the genetic algorithm didn't fully converge within the 200 generations, it showed strong signs of getting there. Since the maximum possible fitness is unknown, it's hard to say how close we got to the global optimum but after visualizing the final solution, it's clear that it's a very strong one. It is possible to see that there are no conflicts at any table, and that at the bride and groom's table (table 8), they are seated together with their best friends and date, which is ideal for this table (Figure 9).

Although many configurations and operators were explored, further improvements are possible. Each configuration was tested with 30 runs, the statistical minimum for reliability, but additional runs could

improve robustness. Furthermore, generations until full convergence should be extended to allow the algorithm more time to fully explore the search space and potentially find better solutions. Moreover, new and modified crossovers and mutations could be explored, especially tailored to this specific problem, to improve the least effective operators such as the standard one-point crossover and the inversion mutation that are never among the top performative operators and decrease the probabilities of getting stuck in a local minimum and getting the algorithm to reach full convergence sooner.

# 8. References

1. Kumar, M., Husain, D. M., Upreti, N., & Gupta, D. (2010). Genetic algorithm: Review and application. Available at SSRN 3529843.

# 9. Annex A – Definitions

| Concept | Definition |
|---|---|
| Elitism | Preserves the best seating arrangements to the next generation to keep high-quality solutions. |
| Fitness sharing | Reduces selection chances for similar seating plans to maintain diversity. |
| Restricted Mating | Prevents mating between similar seating arrangements to encourage variety but can limit exploration. |
| Hill climbing | Moves only to better neighboring solutions; fast but prone to local optima |
| Simulated annealing | Moves to neighboring solution but sometimes it accepts worse solutions to avoid local optima. |

# 10. Figures

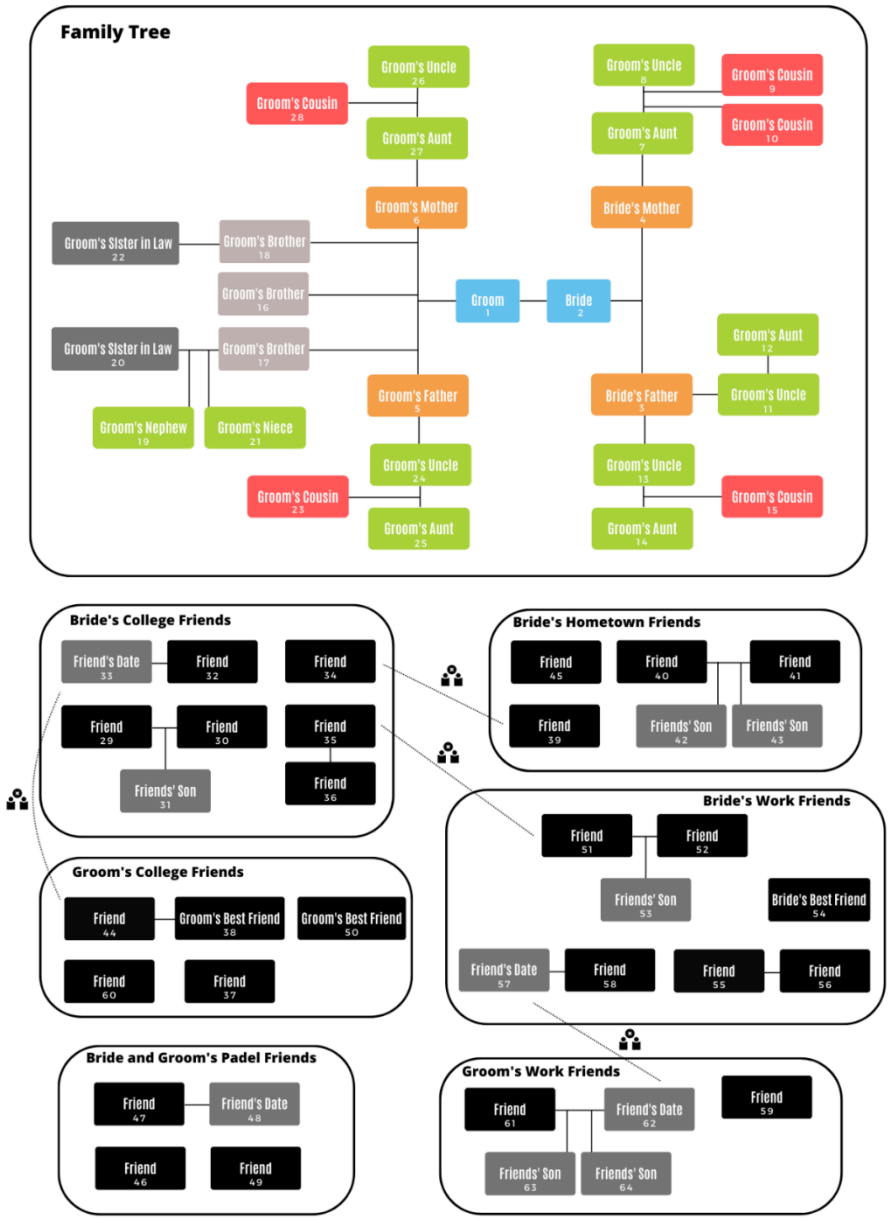



Figure 1 – Relationship between the guests

| Relationship | Value |
|---|---|
| Bride or Groom | 5000 |
| Spouse or Date | 2000 |
| Best Friend | 1000 |
| Siblings | 900 |
| Parent or Child | 700 |
| Cousin | 500 |
| Aunt/Uncle or Niece/Nephew | 300 |
| Friend | 100 |
| Strangers | 0 |
| Enemies | -1000 |

Figure 2 – Fitness / value corresponding to each relationship

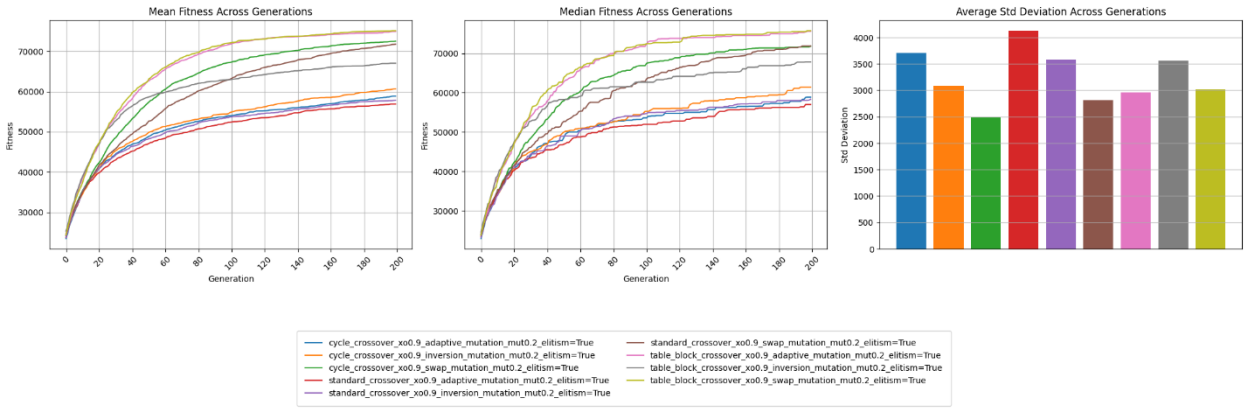



Figure 3 – configuration results across the three metrics defined for Tournament selection

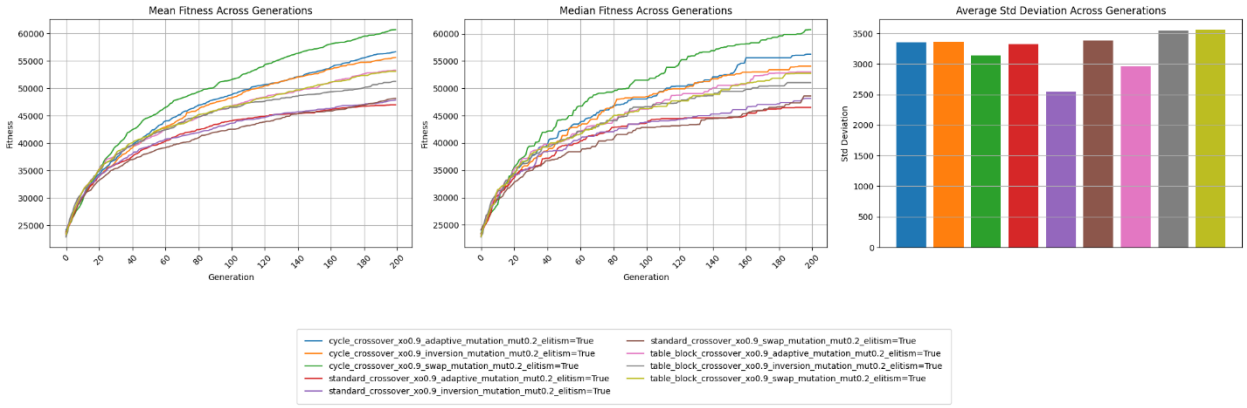



Figure 4 – configuration results across the three metrics defined for Fitness proportionate selection

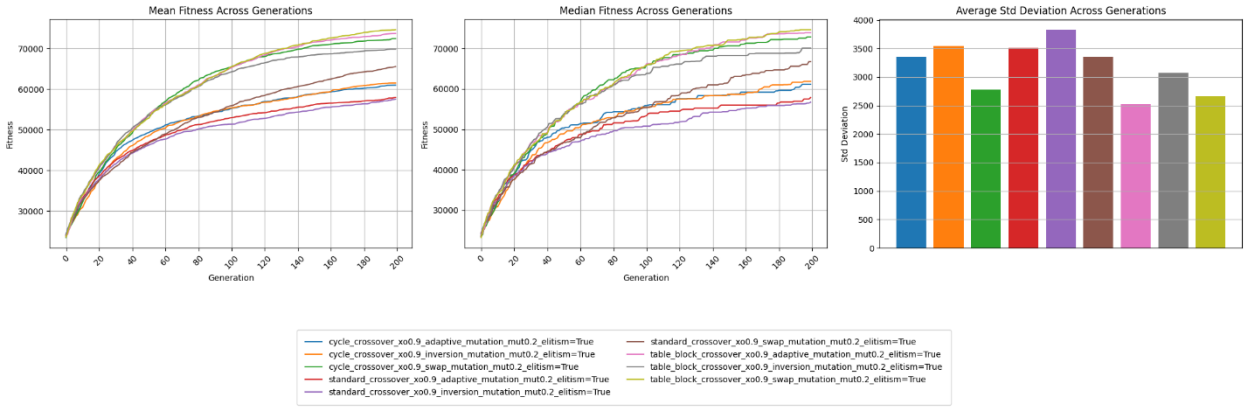



Figure 5 – configuration results across the three metrics defined for Ranking selection

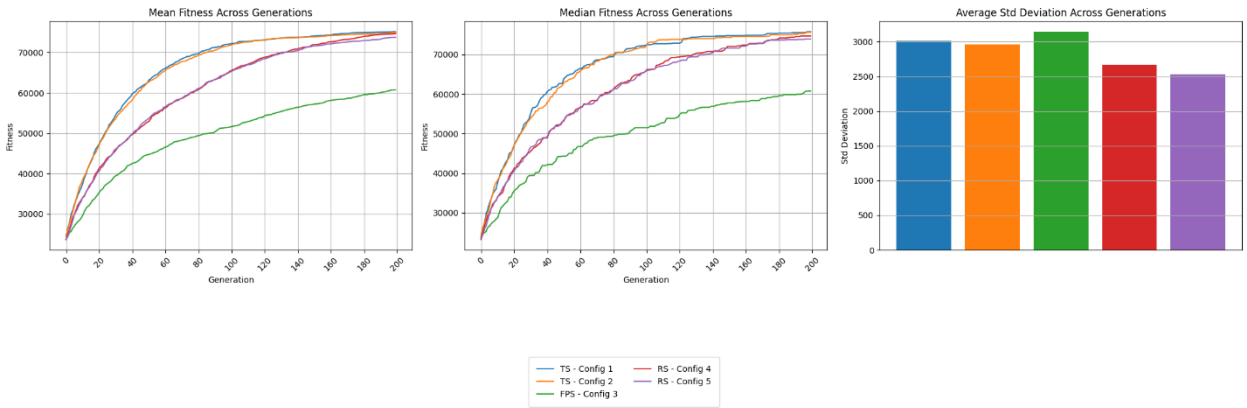



Figure 6 – Best configurations from the different selection strategies

| iter | k | mut_prob | xo_prob | average_fitness |
|---|---|---|---|---|
| 9 | 3 | 0.400 | 0.900 | 76200.00 |
| 12 | 3 | 0.400 | 0.783 | 75903.33 |
| 11 | 3 | 0.315 | 0.899 | 75713.33 |
| 6 | 3 | 0.400 | 0.900 | 75356.67 |
| 13 | 3 | 0.398 | 0.607 | 75280.00 |
| 16 | 3 | 0.358 | 0.628 | 75263.33 |
| 17 | 4 | 0.400 | 0.600 | 75196.67 |
| 20 | 3 | 0.316 | 0.638 | 75186.67 |
| 14 | 4 | 0.400 | 0.900 | 75026.67 |
| 1 | 3 | 0.385 | 0.820 | 75006.67 |
| 8 | 3 | 0.400 | 0.820 | 74906.67 |
| 10 | 3 | 0.400 | 0.900 | 74876.67 |
| 5 | 4 | 0.164 | 0.655 | 74363.33 |
| 19 | 3 | 0.118 | 0.727 | 73916.67 |
| 7 | 3 | 0.206 | 0.900 | 73853.33 |
| 2 | 3 | 0.147 | 0.647 | 73780.00 |
| 4 | 3 | 0.106 | 0.891 | 73700.00 |
| 18 | 3 | 0.177 | 0.751 | 73680.00 |
| 3 | 2 | 0.360 | 0.780 | 73560.00 |
| 15 | 4 | 0.100 | 0.900 | 73283.33 |

Figure 7 - Results of Bayesian search

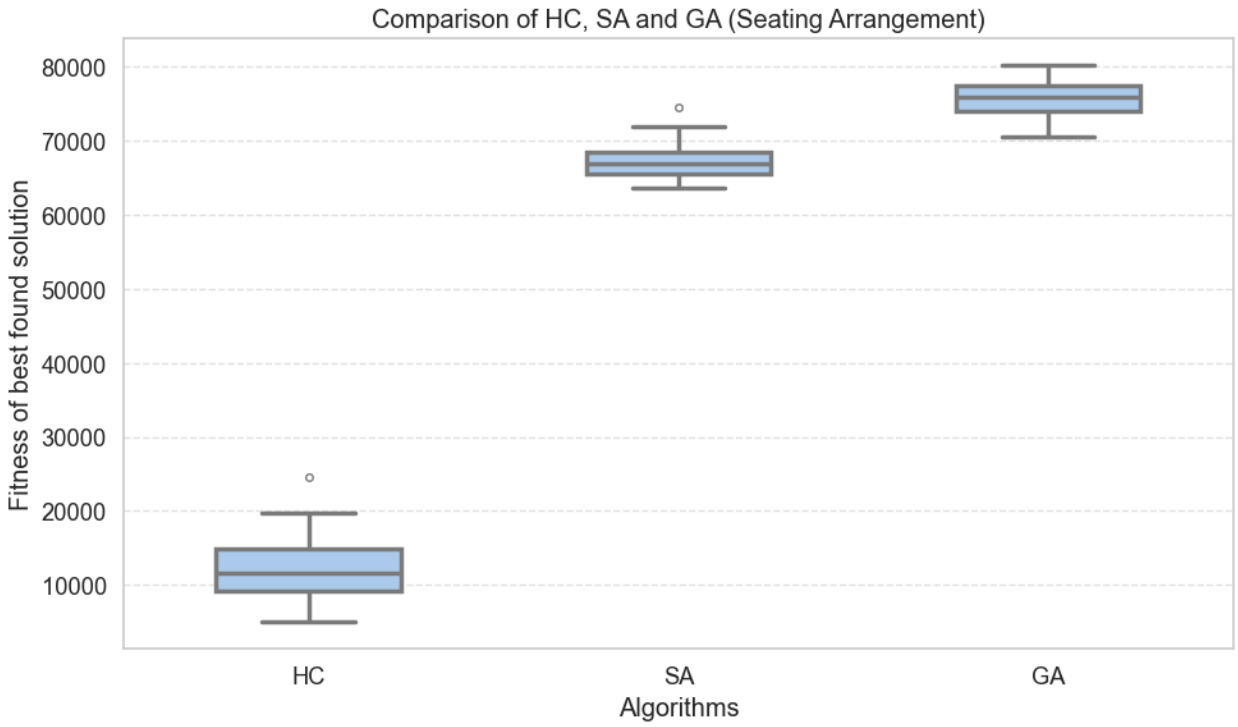



Figure 8 – Comparison between the three algorithms

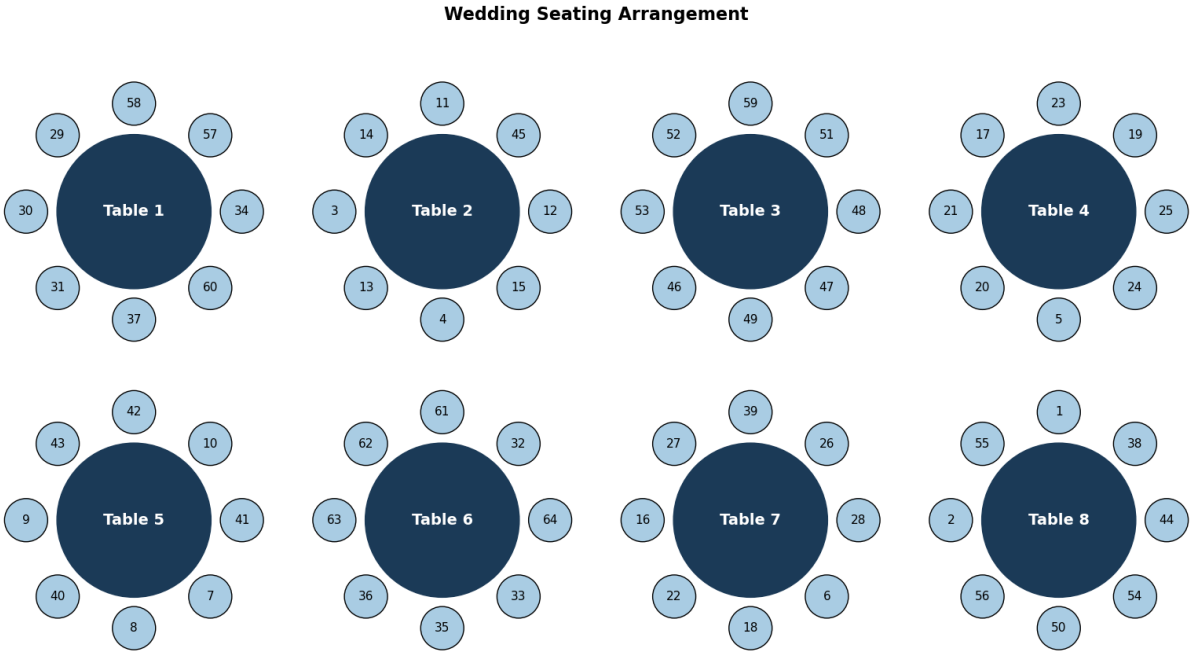



Figure 9 – Wedding final seating allocation