WEDDING SEATING OPTIMIZATION

COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

https://github.com/Gomsofi06/CIFO

DAVID CASCÃO, 20240851
JAN-LOUIS SCHNEIDER, 20240506
MARTA BOAVIDA, 20240519
SOFIA GOMES, 20240848

Index

1.Introduction 3
2.Problem Formalization 3
2.1 Individual and Search Space Definition 3
2.2 Fitness Function Design 3
3. Genetic Operators Overview 3
3.1 Selection Mechanisms 4
3.1.1 Simple Selection Mechanisms 4
3.1.2 Advanced Selection Mechanisms 4
3.2 Crossover Operators 4
3.3 Mutation Operators 5
4. Experiments 5
4.1 Setup of phases 5
4.2 Success Measurement 6
4.3 Methods Comparison 6
4.4 Simple vs Advanced Comparison 6
4.5 Best configs and solutions 7
5. Conclusion and future work 7
References 8
APPENDIX 9

1.Introduction

The wedding seating arrangement is a challenging optimization problem that wedding planners frequently face. It involves balancing interpersonal relationships and individual preferences while satisfying a specific set of constraints. Manual arrangements are time-consuming and rarely optimal, making this problem a strong candidate for computational intelligence techniques.

The objective is to optimize the seating of 64 guests across 8 tables in a way that maximizes overall guest happiness. This is achieved using a relationship matrix, which assigns pairwise scores representing how much each guest likes the others values indicating stronger positive relationships. Each guest must be assigned to exactly one table.

Through this report, you can get a detailed explanation of everything that was done to achieve our goal. The code that made this project possible is available at the following link https://github.com/Gomsofi06/CIFO.

2.Problem Formalization

2.1 Individual and Search Space Definition

In our genetic algorithm, each individual represents a full seating plan for the wedding. We chose to model this as a list of lists, where each inner list stands for a table and contains the IDs of the guests sitting there.

Example of a solution.: [[1, 5, 12, 23, 42, 55, 60, 63], [0, 7, 15, 24, 35, 44, 51, 59], …]

This structure makes it easy to keep things organized and ensures that the rules of the problem are always followed every guest is seated at one, and only one, table, all tables are used, and it doesn't matter who sits where within each table, only who they sit with. Arrangements that leave someone out or assign them to more than one table are considered invalid, and our implementation makes sure that none of those slip through during initialization or when applying crossover and mutation.

The space of possible solutions includes every valid way of grouping the 64 guests across the 8 tables, with each guest placed exactly once. While this space is very large, it's limited to valid options only, which helps the algorithm focus on promising solutions.

2.2 Fitness Function Design

To measure how good a seating plan is, we use a fitness function based on how much the guests like the people they're seated with. We rely on a relationship matrix that gives a score for every pair of guests. Since these scores aren't always symmetric, for example, someone might rate another person highly even if the feeling isn't mutual we made sure our fitness function looks at both directions of each relationship. For every table, we calculate the total relationship score by adding up the scores for all guest pairs sitting together, and the overall fitness of the seating plan is the sum of those scores across all tables. The higher the score, the happier the guests are likely to be.

3. Genetic Operators Overview

To guide the evolutionary process of our genetic algorithm toward better seating arrangements, we carefully designed and implemented specific strategies for selection, crossover, and mutation. Each of these components plays a distinct role: selection determines which individuals are chosen to pass on their traits,

crossover combines parts of those individuals to create new seating configurations, and mutation introduces small changes that help maintain diversity and avoid premature convergence Figure 1.

3.1 Selection Mechanisms

We explored several selection algorithms to determine how individuals are chosen for reproduction based on fitness, each offering a different trade-off between exploration and exploitation.

3.1.1 Simple Selection Mechanisms

We explored several selection algorithms to determine how individuals are chosen for reproduction based on fitness, each offering a different trade-off between exploration and exploitation.

The first method we implemented was Roulette Wheel Selection, where each individual is assigned a slice of a wheel proportional to their fitness. The wheel is spun randomly to select parents, giving fitter individuals a higher chance of being chosen while still allowing weaker ones to contribute. This helps maintain diversity and prevents the algorithm from converging too quickly Figure 2.

In contrast, Tournament Selection works by randomly selecting a group of individuals, in our case, k = 3, and choosing the fittest among them as the parent. This process is repeated for each parent needed. The method introduces a controlled level of selection pressure while preserving randomness, effectively balancing the search between strong candidates and broader population diversity Figure 3.

Finally, ranking Selection takes a different approach by sorting individuals by fitness and assigning selection weights based on their rank rather than their actual fitness values. For example, the individual with the lowest fitness receives rank 1, the next best rank 2, and so on. Higher-ranked individuals have a greater chance of being selected, but the gap between selection probabilities is more controlled than in fitness-proportional methods. This helps maintain diversity and reduces the risk of domination by a few exceptionally fit individuals early on. Figure 4

3.1.2 Advanced Selection Mechanisms

We also implemented more advanced selection algorithms that improve on basic methods by offering better sampling consistency and dynamic control over selection pressure during the evolutionary process.

Stochastic Universal Sampling builds on Roulette Wheel Selection by placing evenly spaced pointers across the fitness wheel rather than using repeated random spins. Calculates pointer spacing using the formula total_fitness/num_selected, and starts from random position. This approach provides a more consistent and representative sample of the population and significantly reduces selection variance Figure 5.

Meanwhile, Boltzmann Selection introduces a temperature-controlled exponential weighting temperature-based mechanism to dynamically adjust selection intensity. It calculates each individual's selection probability using the formula exp(fitness/temperature). At high temperatures, selection is nearly uniform, encouraging broad exploration. As the temperature decreases, the algorithm becomes more selective, focusing on fitter individuals and supporting exploitation. This allows a smooth transition from exploration to refinement over time Figure 6.

3.2 Crossover Operators

We tried several crossover methods to see which ones best combined parent solutions while keeping the seating valid and high-quality.

One of the simpler methods we used is Group-Based Crossover, which merges both parents' seating arrangements into a single pool of guests. This pool is shuffled to introduce randomness, and new tables are then

formed by selecting unique guests from the shuffled list. This ensures all guests are seated without duplication while preserving genetic diversity Figure 7.

In contrast, Greedy Table Merge emphasizes the preservation of complete tables. It alternates between the two parents to select non-conflicting tables for the offspring. Once overlaps occur, the algorithm systematically assigns the remaining guests in a way that continues to satisfy the problem's constraints Figure 8.

Table Preservation Crossover takes a more evaluative approach by measuring the fitness contribution of each table using the actual relationship scores among guests. It greedily selects the highest-scoring tables from both parents, provided they don't share any guests. After the optimal groupings are secured, the remaining unassigned guests are placed to complete the seating arrangement Figure 9.

Lastly, Partially Mapped Crossover (PMX) introduces a more traditional genetic algorithm technique. A random segment of one parent is copied directly into the offspring, and a mapping is created between corresponding genes in the selected segment from both parents. This mapping is then used to resolve conflicts and fill the remaining positions in the child, ensuring that each guest appears only once Figure 10.

3.3 Mutation Operators

To maintain diversity and avoid premature convergence, we implemented several mutation operators that introduce variation into the population while preserving valid seating arrangements.

The most straightforward of these is Swap Mutation, which iterates through each guest and, with a 0.2 probability, selects two guests from different tables and swaps their positions. This probabilistic approach introduces subtle, randomized changes that help the algorithm explore nearby solutions and escape local optima when needed Figure 11.

A more focused alternative is One-Point Mutation, which moves a single guest from their current table to a different one. This small, controlled change causes minimal disruption but can still be effective in navigating the search space and fine-tuning solutions Figure 12.

To enable larger structural shifts, Multi-Point Mutation takes a more deterministic approach by selecting and moving a fixed number of guests—specifically ten—across tables in a single operation. This broader change allows the algorithm to make more substantial leaps in the solution space, which can be particularly useful for escaping deep local minima or exploring new configurations Figure 13.

Finally, Inversion Mutation takes a different approach by flattening the seating arrangement into a linear sequence. It randomly selects two boundaries and reverses the order of guests within that segment. This kind of reordering can lead to the discovery of new groupings and seating patterns that wouldn't easily arise from swaps or moves alone. Figure 14

4. Experiments

4.1 Setup of phases

To better understand how different configurations impact the quality of our solutions, we structured the experimentation process into four progressive phases, each one building on the learnings of the previous Figure 15.

In the first phase, we started with a small setup — just 10 individuals and 10 runs — to quickly explore a wide range of hyperparameter combinations. We tested different crossover and mutation rates (from 0.1 to 0.9), varied elitism between 0% and 25%, used a high delta threshold of 3000 to allow big changes in fitness. In phase two, we increased the population to 50 and ran 30 experiments and, keeping the other settings the same. This helped us see which configurations consistently worked well under slightly more demanding conditions. Phase three focused more on refining the results. We increased the population to 250, doubled the number of generations, and reduced the delta threshold to 250 to encourage smaller, more precise improvements. Early stopping was used if there was no fitness improvement for three generations. Finally, in phase four, we kept the same population and generation limits but ran 100 experiments using the best settings found so far. This helped reinforce convergence and ensure more consistent results.

4.2 Success Measurement

At first, we selected configurations manually by looking at results individually, but we soon realized this approach was far too simplistic and didn't give us a reliable way to compare performance. To better compare different configurations, we created a single score that combines solution quality, reliability, and efficiency. This score is based on four things: average fitness, maximum fitness, how much the results vary (standard deviation), and how many generations it takes to reach a stable solution. We gave the most importance to average fitness because it shows how good the solutions are overall. Maximum fitness mattered too, but a bit less it highlights when a configuration can find outstanding solutions. To favor reliable setups, we gave a negative weight to the standard deviation fitness, which means more stable results are rewarded. We also gave a negative weight to the average number of generations, so configurations that reach good solutions faster score higher.

4.3 Methods Comparison

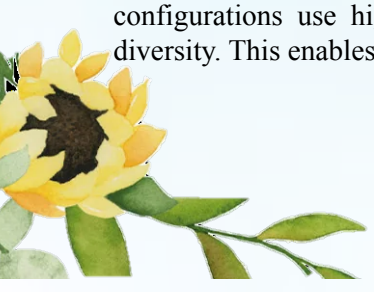
To gain a broad understanding of operator effectiveness, we analyzed the configurations tested in Phase 1, which included approximately 32,000 variations. One key finding was the strong performance of the Boltzmann selection method. As shown in Figure 16 it achieved both higher average and maximum fitness scores compared to other selection strategies, indicating its strength in balancing exploration of new solutions with the exploitation of existing high-performing ones. When comparing mutation and crossover, Figure 17 shows that mutation operators have a greater impact on achieving maximum fitness. This highlights their crucial role in helping the algorithm escape local optima and discover better solutions. Figure 18 further reveals that group-based crossover leads to the fastest convergence rate. This can likely be attributed to its ability to preserve complete table groupings from parent solutions, reducing disruption to well-formed guest clusters and accelerating the accumulation of high-quality solutions.

4.4 Simple vs Advanced Comparison

A noticeable performance gap can be seen between simple and advanced configurations, as shown in Figure 19. From Phase 3 to Phase 4, advanced setups show only a small drop in average fitness (about 1%), indicating consistent convergence rather than random fluctuations. This suggests that advanced configurations effectively refine and maintain high-quality solutions, even as parameters are adjusted.

Under identical generational constraints, advanced methods outperform simpler ones, achieving 15–20% higher average fitness by generation 50 Figure 20. This highlights the value of well-tuned operators and parameters in accelerating convergence. Figure 20 and Figure 21 further reveal key differences in strategy. Simple configurations apply consistent elitism to preserve strong solutions, while advanced setups reduce or remove elitism later on to promote diversity and avoid early convergence.

Operator rates also vary: simple setups use conservative values to limit sudden changes, whereas advanced configurations use higher crossover and mutation rates to combine strong solutions and introduce controlled diversity. This enables a more focused and effective exploration of the solution space.



4.5 Best configs and solutions

To confirm our findings, we ran a final comparison using the best-performing simple and advanced configurations. This test was carried out with a large population size of 1,000 and repeated 100 times to ensure reliable results Figure 22. The simple configuration, which used tournament selection and greedy merge, achieved an average fitness of 149,814 and a maximum fitness of 161,900. However, it required 128 generations on average to converge. In contrast, the advanced configuration, which relied on Boltzmann selection and partial elitism, performed better in all aspects. It reached a higher average fitness of 153,922, a maximum fitness of 162,200, and converged in just 37 generations on average, making it about 3.5 times faster and more efficient.

The best solution from the advanced configuration had a fitness score of 162,200. We also discovered a slightly better solution with a score of 162,500, but it wasn't consistent across runs, meaning it couldn't be reliably reproduced. While both solutions achieved very high fitness score, they did not consider important social preferences, such as seating the bride and groom with their parents. This highlights a key limitation of the algorithm: its focus on mathematical optimization may overlook cultural or emotional factors that are harder to quantify.

162500: [[0, 1, 23, 37, 43, 49, 60, 61], [2, 3, 6, 7, 8, 9, 14, 53], [4, 5, 15, 16, 17, 18, 19, 20], [10, 11, 12, 13, 54, 55, 56, 57], [21, 31, 32, 34, 35, 38, 46, 47], [22, 24, 33, 39, 40, 41, 42, 44], [25, 26, 27, 48, 50, 51, 52, 58], [28, 29, 30, 36, 45, 59, 62, 63]]

162200: [[0, 1, 37, 43, 46, 47, 49, 53], [2, 3, 12, 13, 14, 34, 35, 45], [4, 5, 15, 16, 17, 18, 19, 20], [6, 7, 8, 9, 10, 11, 56, 57], [21, 38, 39, 40, 41, 42, 44, 58], [22, 23, 24, 25, 26, 27, 31, 32], [28, 29, 30, 33, 60, 61, 62, 63], [36, 48, 50, 51, 52, 54, 55, 59]]

5. Conclusion and future work

Our genetic algorithm effectively optimized wedding seating arrangements, consistently producing high-quality solutions that maximize guest happiness based on interpersonal relationships. The study highlighted the critical role that genetic operator selection and parameter tuning play in overall performance.

Advanced methods proved to be highly effective, but they demand precise calibration. For instance, using high crossover rates (such as 0.9) can lead to overfitting on temporary patterns, while reducing elitism to low levels (0–5%) may cause instability in the later stages of evolution. On the other hand, simple methods offer more robustness through conservative settings, but this comes at the cost of slower convergence, making them less suitable for scenarios where time or computational resources are limited.

Ultimately, our findings reinforce an important point: optimization algorithms are powerful tools, but they cannot replace human judgment. Although advanced configurations excel at maximizing quantitative happiness scores, they can unintentionally overlook social or cultural nuances for example, by separating close family members or isolating introverted guests.

Looking ahead, one clear opportunity for improvement lies in moving beyond static hyperparameters. Future research should explore self-adaptive mechanisms that allow the algorithm to adjust its behavior dynamically as the population evolves, leading to more resilient and context-sensitive solutions.

References

[1] Vanneschi, L., & Silva, S. (2023). *Lectures on Intelligent Systems*. Springer. https://doi.org/10.1007/978-3-031-17922-8

[2] Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6), 2193–2196.

[3] Fang, Y., & Li, J. (2010). A review of tournament selection in genetic programming. In Z. Cai, C. Hu, Z. Kang, & Y. Liu (Eds.), *Advances in Computation and Intelligence* (Vol. 6382, pp. 181–192). Springer.

[4] Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. *Proceedings of the 1st International Conference on Genetic Algorithms*, 101–111.

[5] Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms and Their Application* (pp. 14–21). Lawrence Erlbaum Associates.

[6] de la Maza, M. (1995). The Boltzmann selection procedure. In L. Chambers (Ed.), *Practical Handbook of Genetic Algorithms: New Frontiers* (Vol. II, pp. 111–138). CRC Press.

[7] Falkenauer, E. (1998). Genetic Algorithms and Grouping Problems. Wiley.

[8] Kazakovtsev, L., Rozhnov, I., & Shkaberina, G. (2021). Increasing population variability in parallel genetic algorithms with a greedy crossover for large-scale p-median problems. *ResearchGate*. Retrieved from https://www.researchgate.net/figure/Results-of-running-new-Algorithm-11-and-original-genetic-algorithm-with-greedy-crossover_tbl2_266676040:contentReference[oaicite:11]{index=11}

[9] Goldberg, D. E., & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 154–159). Lawrence Erlbaum Associates.

[10] Fontes, F. A. C., & Fontes, D. B. M. M. (2008). Predicting the outcome of mutation in genetic algorithms. *FEP Working Papers*, (294). Retrieved from https://www.fep.up.pt/docentes/fontes/FCT

[11] Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., & Prasath, V. B. S. (2019). Choosing mutation and crossover ratios for genetic algorithms—A review with a new dynamic approach. *Information*, 10(12), 390. https://doi.org/10.3390/info10120390

[12] Eiben, A. E., & Smith, J. E. (2015). Introduction to Evolutionary Computing (2nd ed.). Springer.

[13] Mitchell, M. (1998). An introduction to genetic algorithms. MIT Press.

APPENDIX

Figure 1

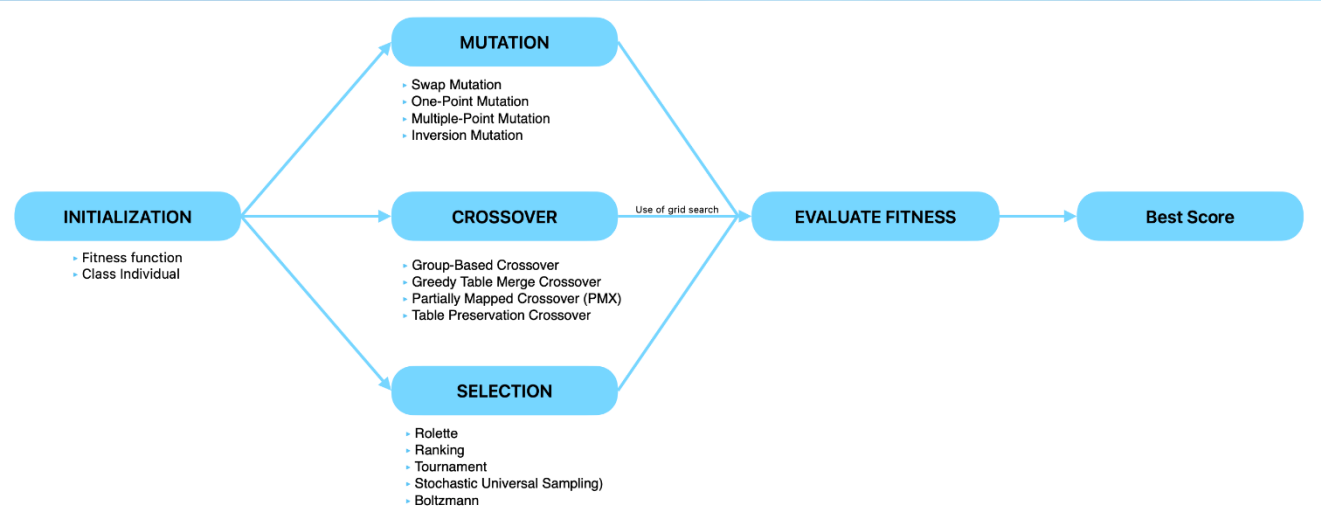


Figure 2

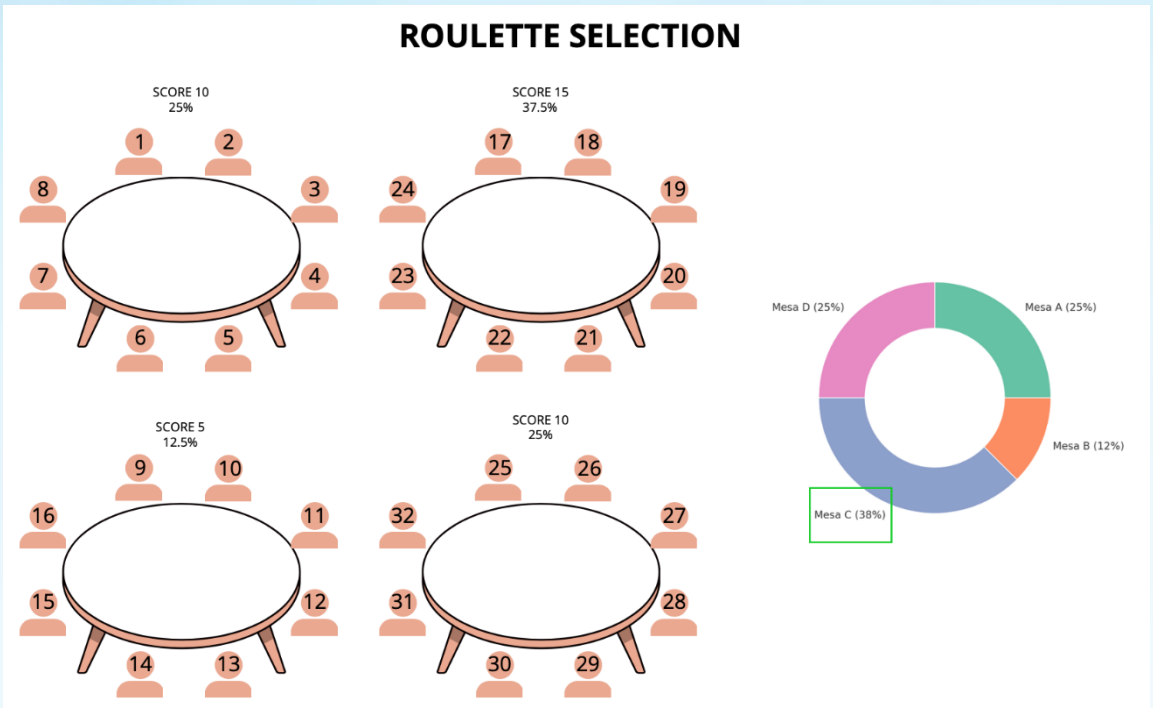
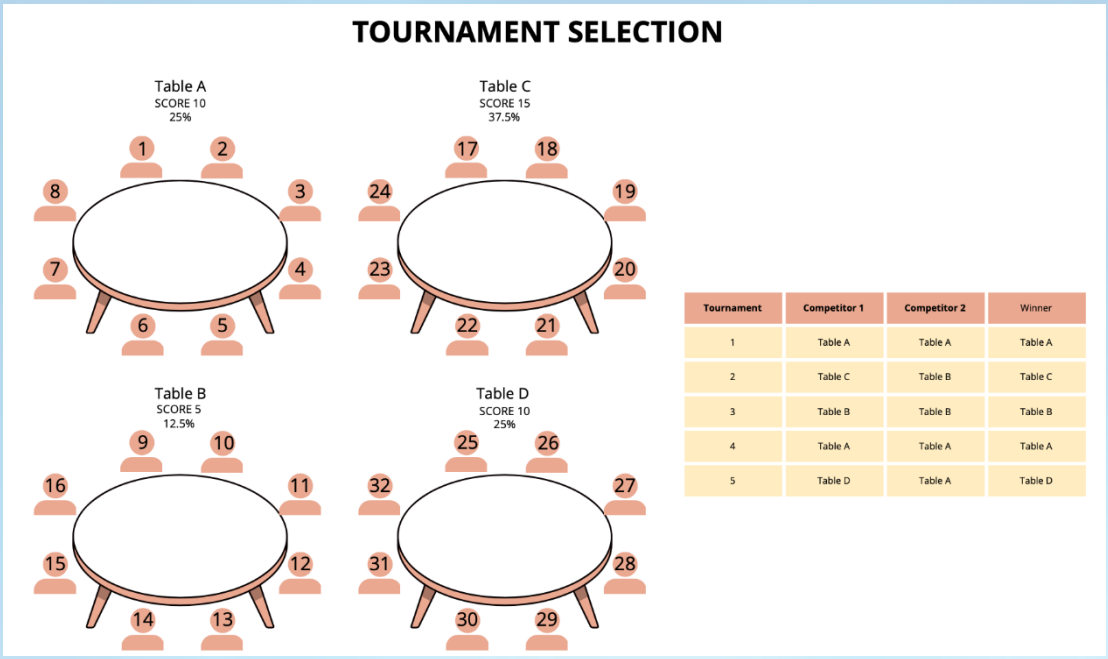
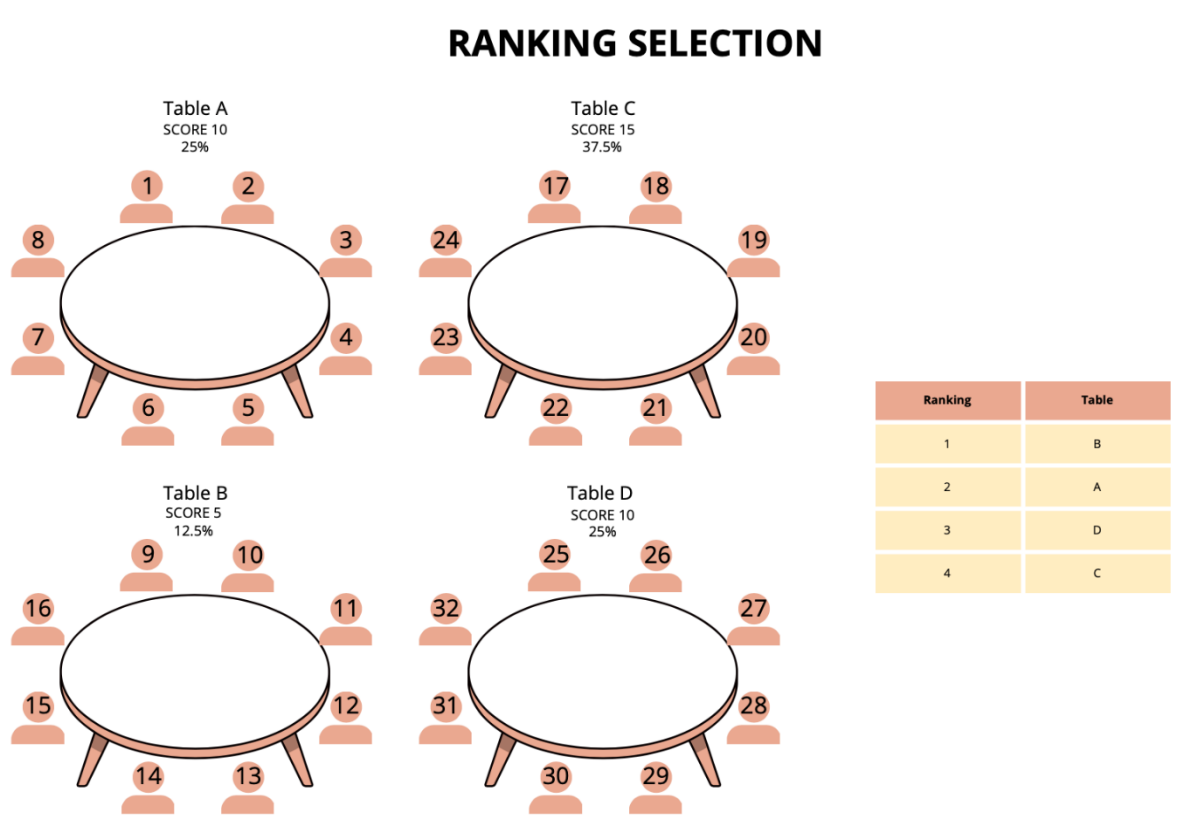


Figure 3



Tournament	Competitor 1	Competitor 2	Winner
1	Table A	Table A	Table A
2	Table C	Table B	Table C
3	Table B	Table B	Table B
4	Table A	Table A	Table A
5	Table D	Table A	Table D

Figure 4



Ranking	Table
1	B
2	A
3	D
4	C

Figure 5

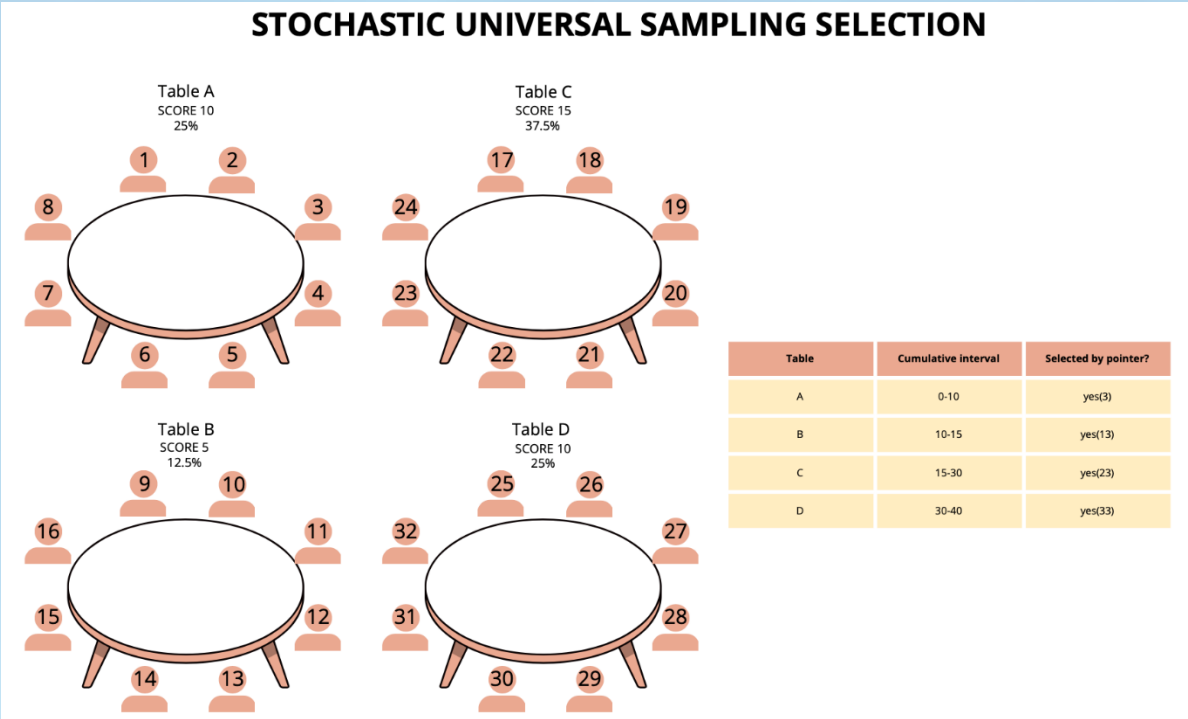
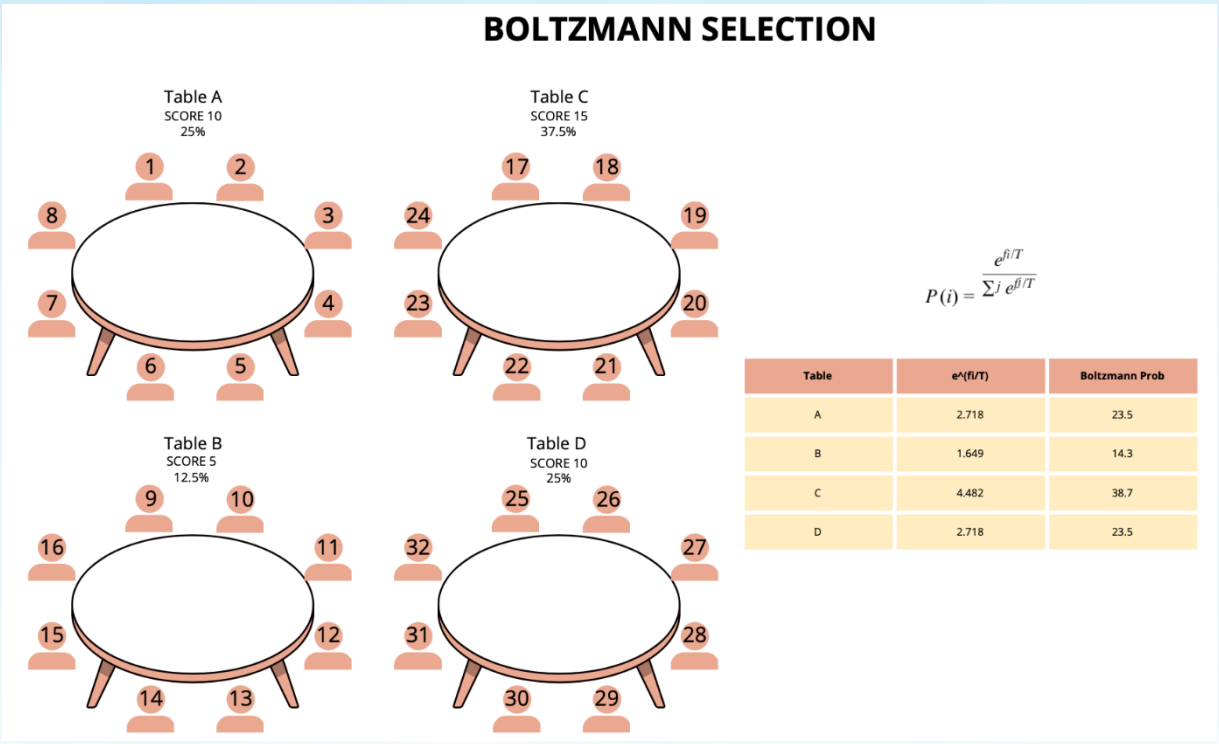


Table	Cumulative interval	Selected by pointer?
A	0-10	yes(3)
B	10-15	yes(13)
C	15-30	yes(23)
D	30-40	yes(33)

Figure 6



$$P(i) = \frac{e^{fi/T}}{\sum_j e^{fj/T}}$$

Table	e^(fi/T)	Boltzmann Prob
A	2.718	23.5
B	1.649	14.3
C	4.482	38.7
D	2.718	23.5

Figure 7

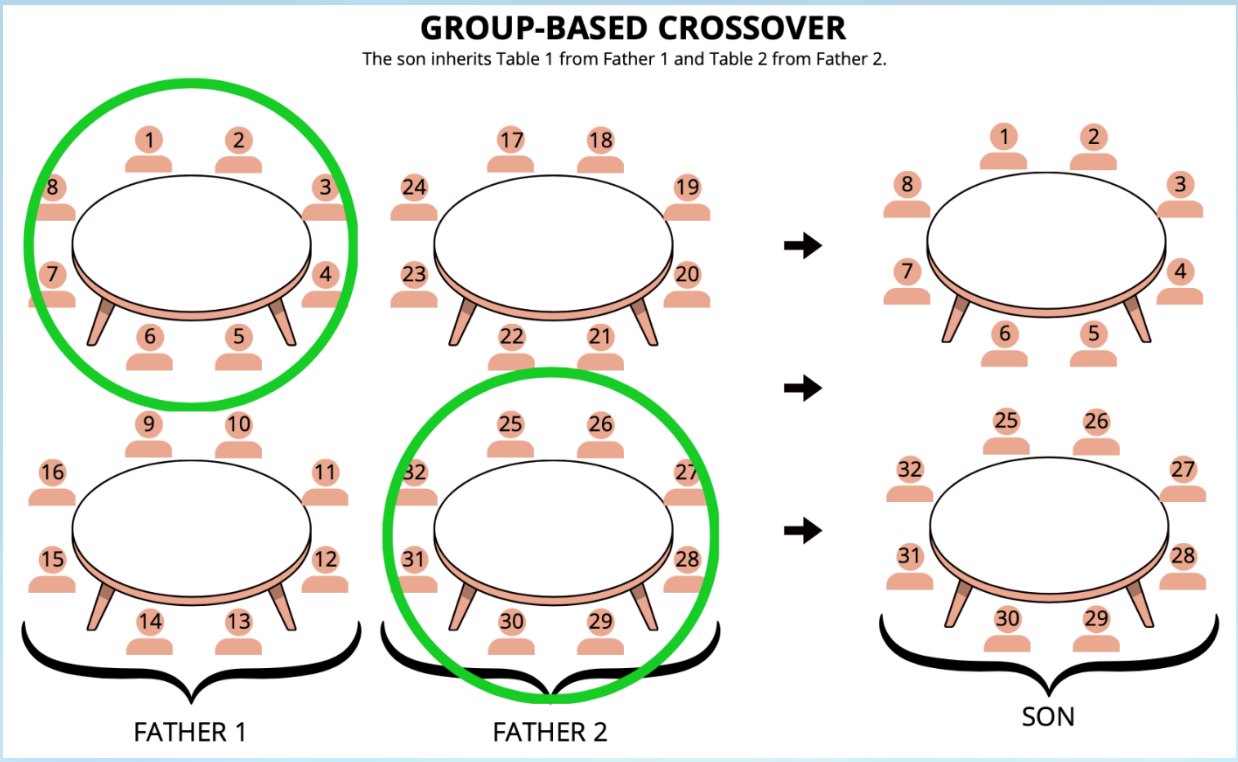


Figure 8

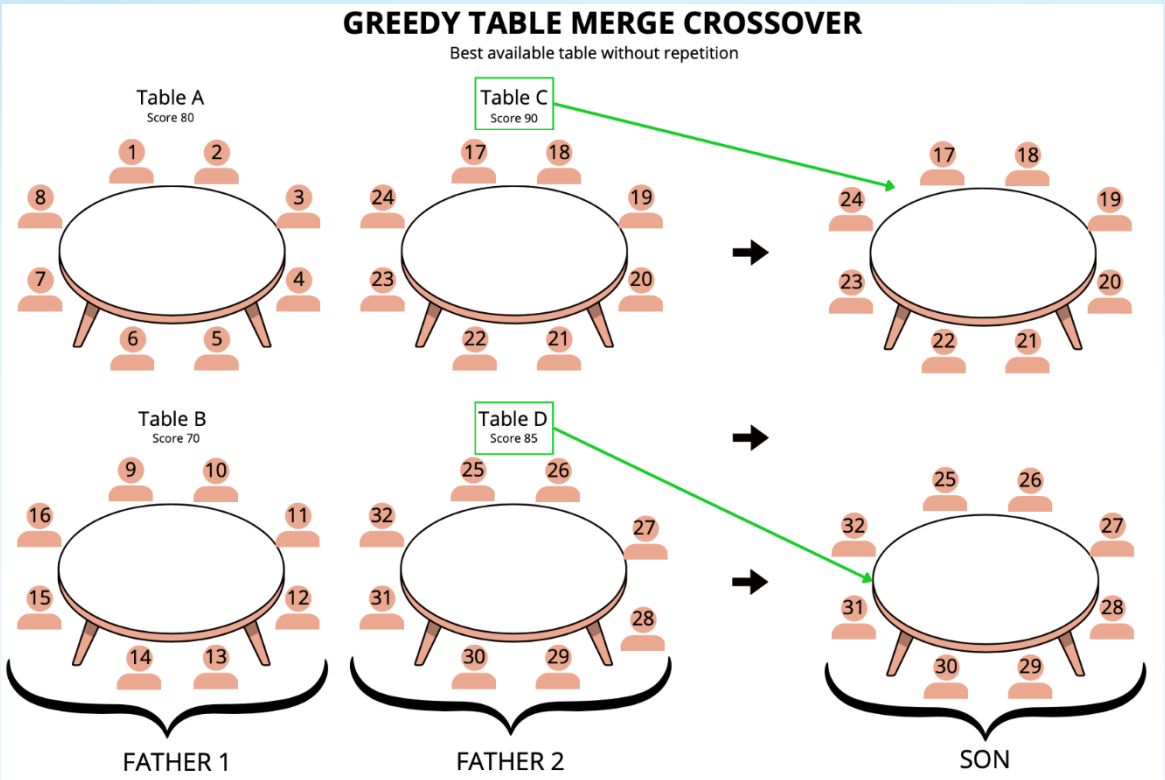


Figure 9

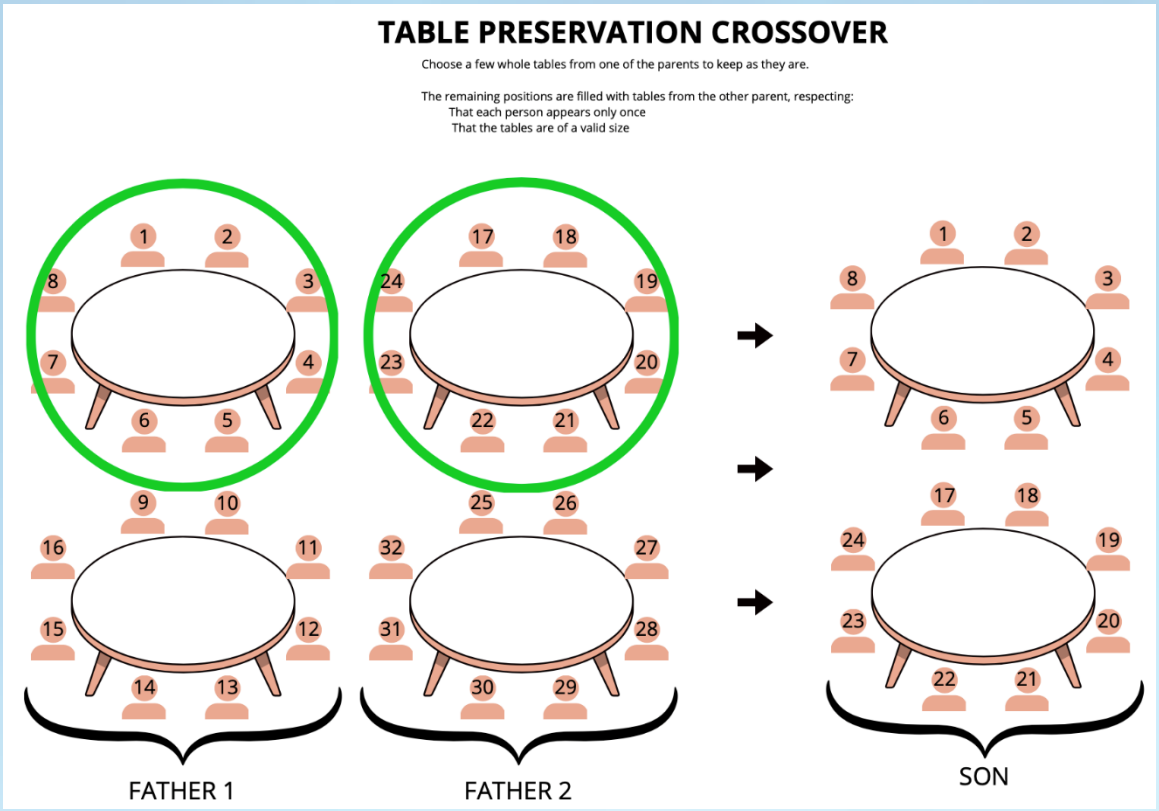


Figure 10

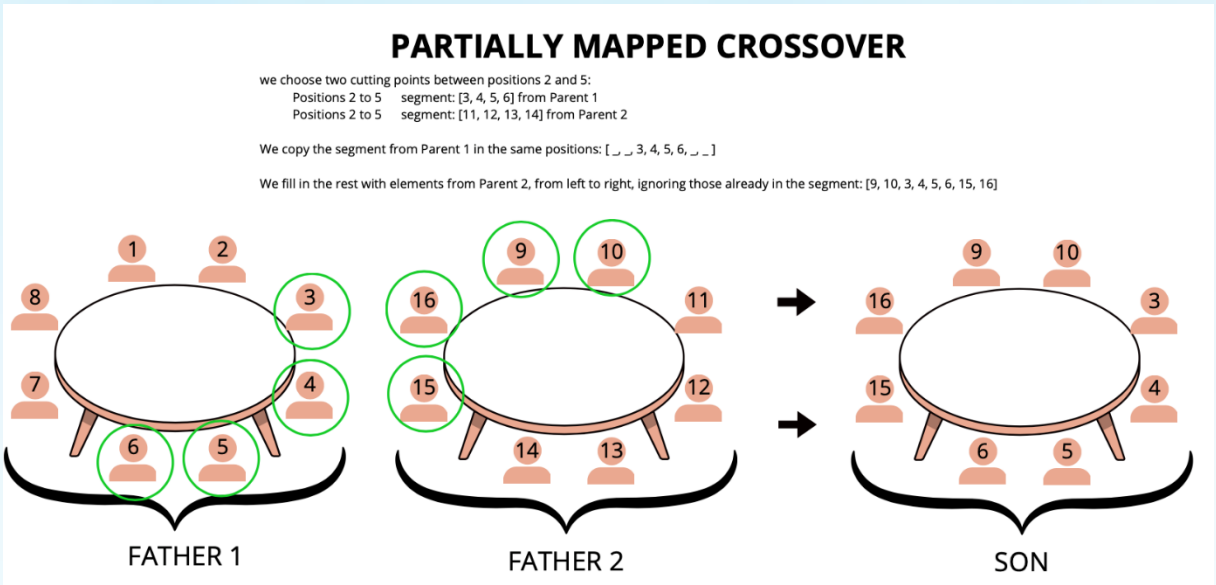


Figure 11

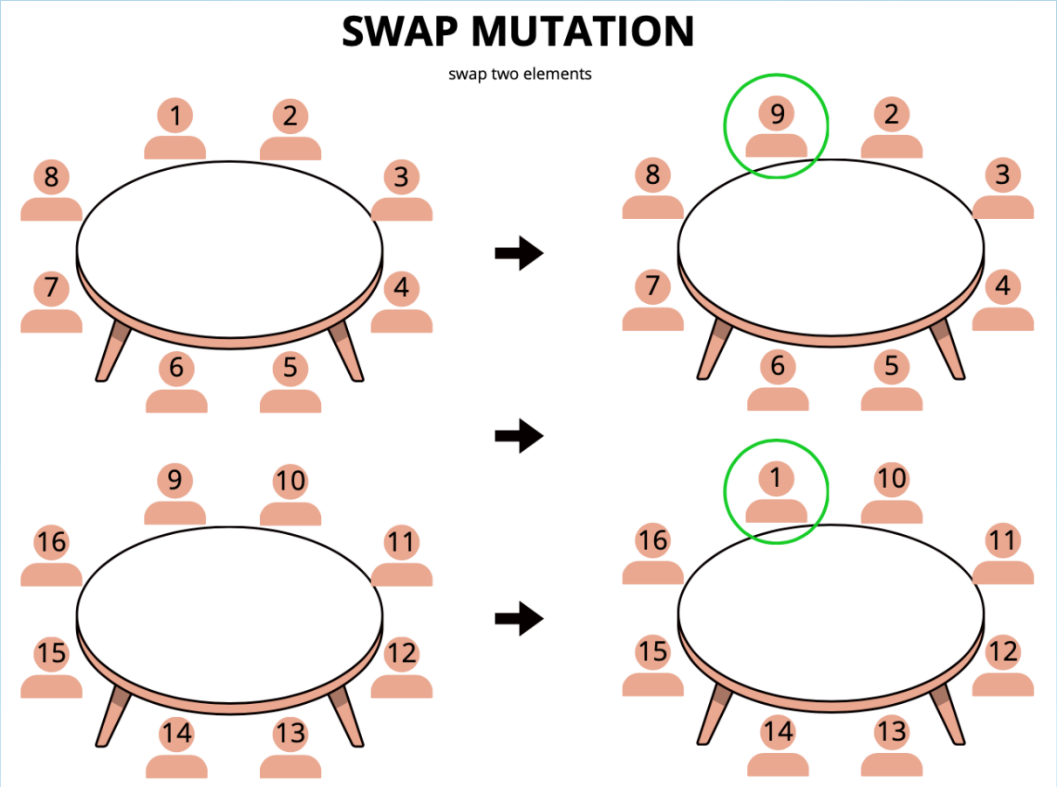


Figure 12

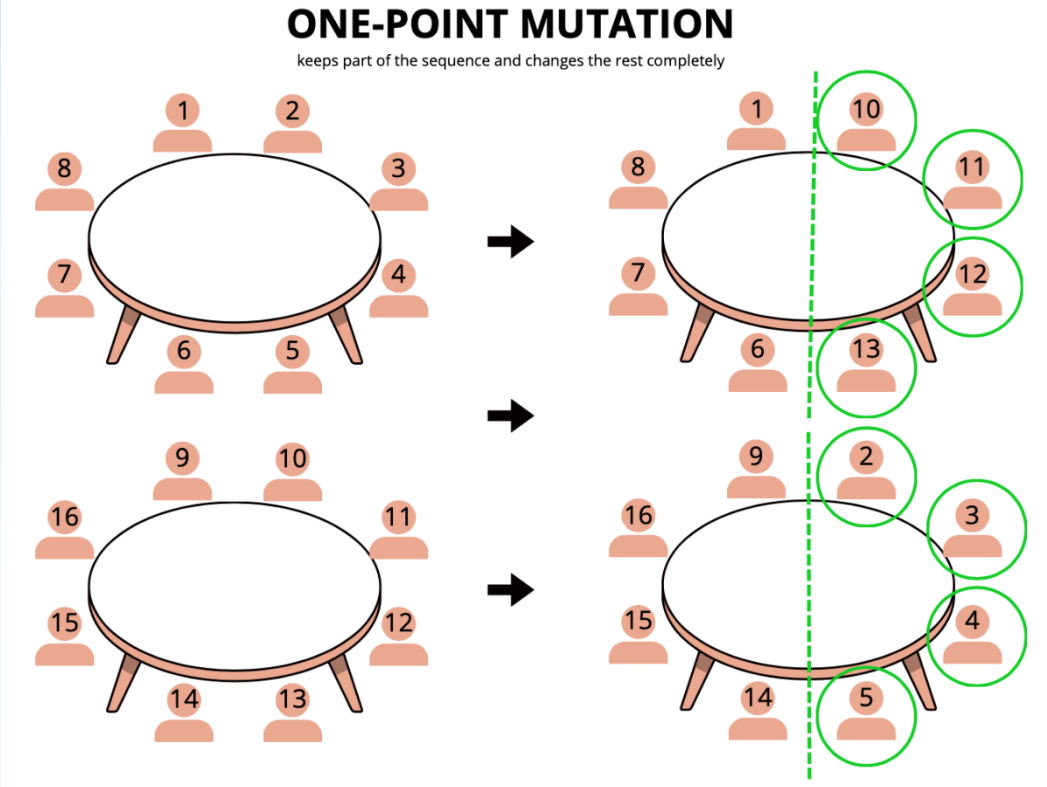


Figure 13

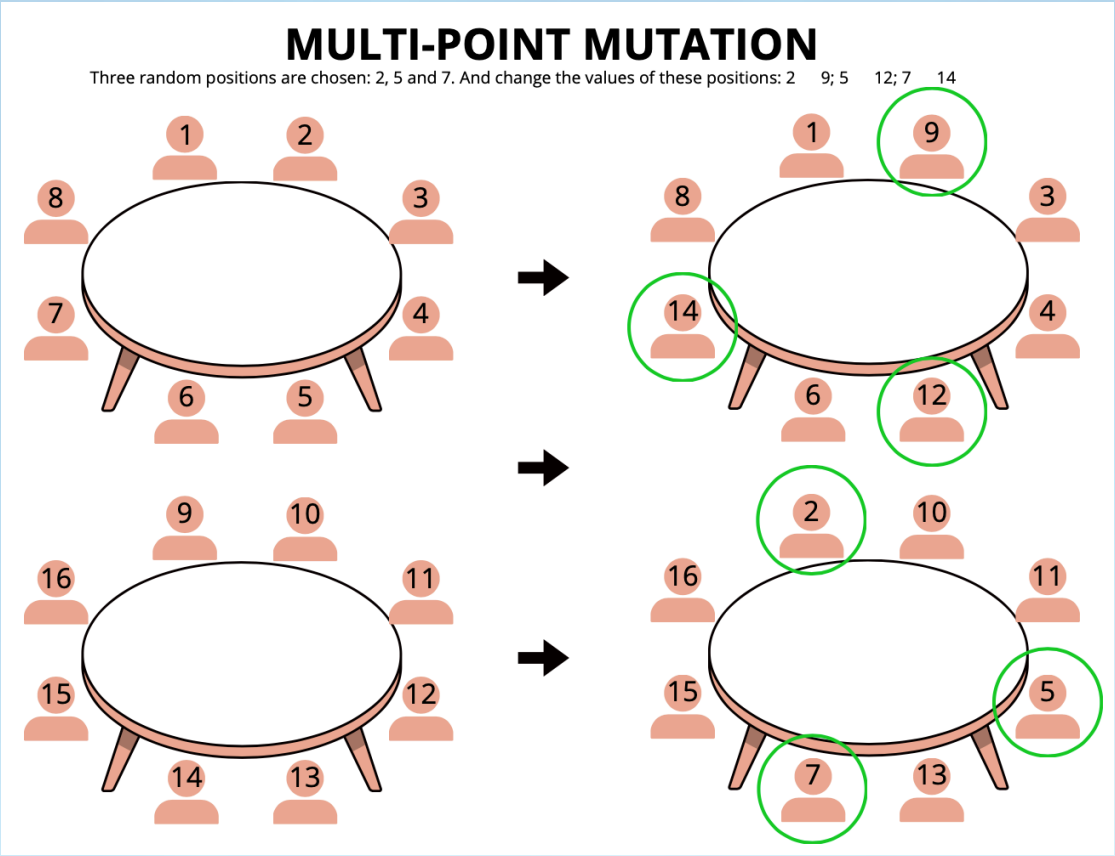


Figure 14

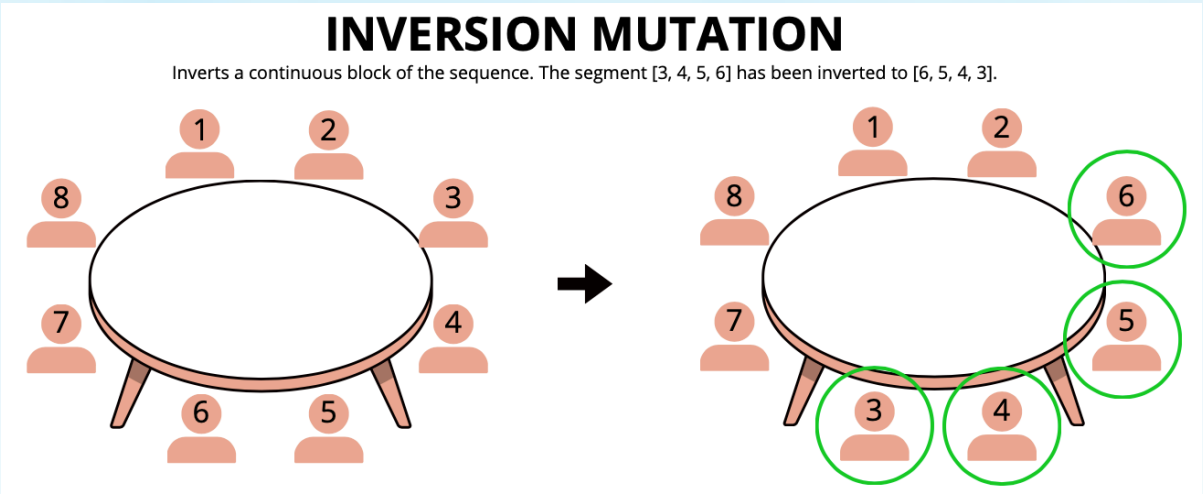


Figure 15

Phase	Population size	Runs	Generations	Early stop gen	Delta threshold
1	10	10	50	3	3 000
2	50	30	50	3	3 000
3	250	50	100	3	250
4	250	100	100	3	250

Figure 16

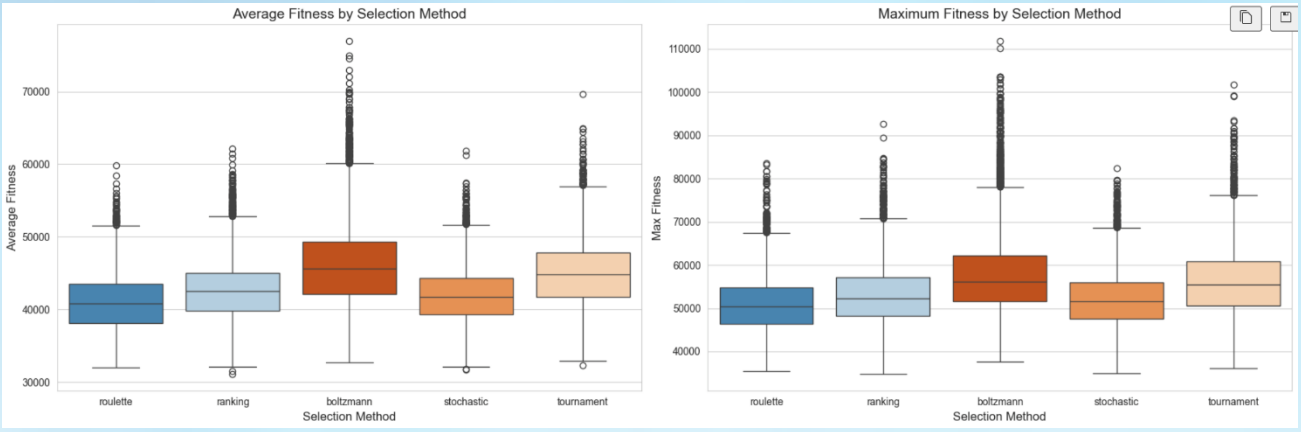


Figure 17

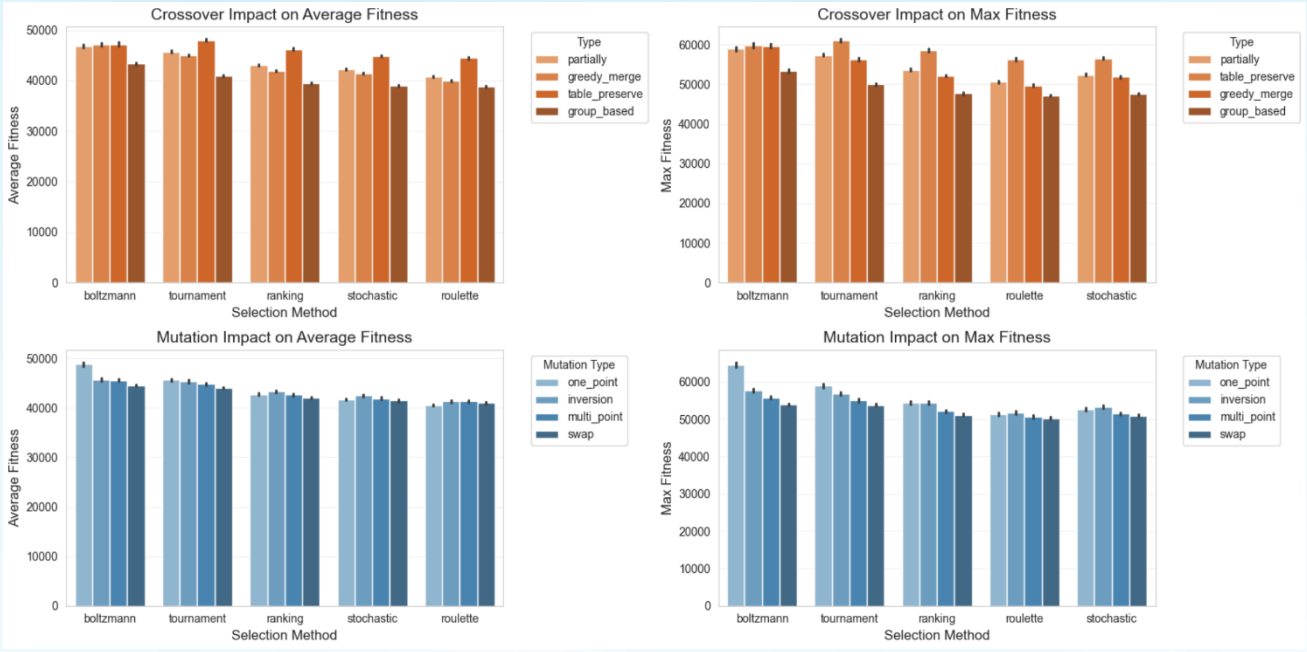


Figure 18

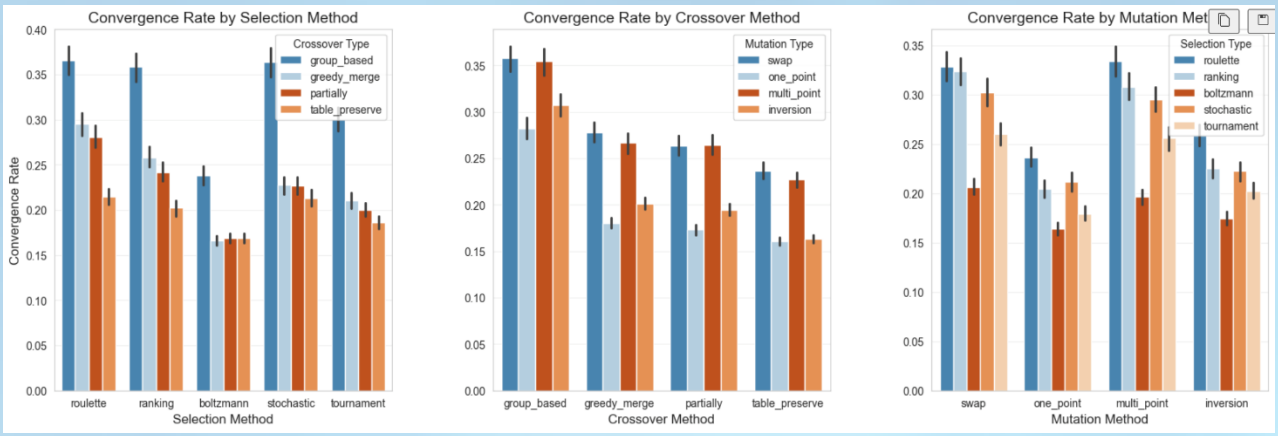


Figure 19

Top 5 Simple Configurations per Phase

Metric	avg_fitness				avg_generations				max_fitness				std_fitness			
Phase	Simple Phase 1	Simple Phase 2	Simple Phase 3	Simple Phase 4	Simple Phase 1	Simple Phase 2	Simple Phase 3	Simple Phase 4	Simple Phase 1	Simple Phase 2	Simple Phase 3	Simple Phase 4	Simple Phase 1	Simple Phase 2	Simple Phase 3	Simple Phase 4
Rank																
1	69660.00	80973.00	89732.00	90281.00	11.90	19.60	20.90	21.69	99200.00	102700.00	121600.00	138100.00	14068.00	12663.00	15410.00	15121.00
2	60190.00	79870.00	89738.00	92803.00	9.80	19.67	20.94	22.95	101700.00	102900.00	120200.00	125800.00	17693.00	13151.00	14240.00	13012.00
3	62740.00	78690.00	89902.00	86251.00	11.40	18.10	22.16	19.60	93500.00	103900.00	124200.00	122700.00	17535.00	12705.00	12457.00	13338.00
4	64860.00	77133.00	87768.00	88314.00	9.60	18.73	20.24	20.52	93200.00	100600.00	117400.00	114000.00	13990.00	13233.00	14136.00	12977.00
5	61340.00	75553.00	84830.00	84850.00	9.50	17.47	19.64	17.83	90700.00	106000.00	114600.00	117500.00	16009.00	13182.00	15450.00	12036.00

Top 5 Advanced Configurations per Phase

Metric	avg_fitness				avg_generations				max_fitness				std_fitness			
Phase	Advanced Phase 1	Advanced Phase 2	Advanced Phase 3	Advanced Phase 4	Advanced Phase 1	Advanced Phase 2	Advanced Phase 3	Advanced Phase 4	Advanced Phase 1	Advanced Phase 2	Advanced Phase 3	Advanced Phase 4	Advanced Phase 1	Advanced Phase 2	Advanced Phase 3	Advanced Phase 4
Rank																
1	74960.00	110427.00	145926.00	144852.00	13.00	19.10	28.94	28.68	99800.00	142500.00	162500.00	159700.00	17597.00	16505.00	6720.00	7008.00
2	69650.00	111060.00	146256.00	146463.00	14.30	19.23	30.40	29.54	102000.00	139200.00	161100.00	157500.00	18995.00	15600.00	5971.00	5676.00
3	74580.00	108340.00	144220.00	143941.00	13.70	17.80	30.46	29.14	99800.00	138400.00	159700.00	157700.00	16573.00	17363.00	6877.00	6911.00
4	63300.00	107467.00	144812.00	144909.00	10.50	17.40	29.60	28.90	111800.00	145000.00	159400.00	159100.00	19752.00	15343.00	6914.00	5986.00
5	76940.00	110433.00	144750.00	144879.00	13.70	17.47	28.80	28.07	96200.00	137800.00	160700.00	158100.00	14190.00	14995.00	6860.00	6574.00

Figure 20

```
==================================== Phase 2 - SIMPLE Configurations ====================================
Rank | Selection    | Crossover      | Mutation | CX Rate | Mut Rate | Elitism | Avg Fitness | Max Fitness
-----|--------------|----------------|----------|---------|----------|---------|-------------|------------
259 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 10%     | 80973.33       | 102700.00
263 | tournament   | greedy_merge   | one_poin | 0.60    | 0.70     | 10%     | 79870.00       | 102900.00
267 | tournament   | greedy_merge   | one_poin | 0.60    | 0.80     | 10%     | 78690.00       | 103900.00
268 | tournament   | greedy_merge   | one_poin | 0.60    | 0.80     | 25%     | 77133.33       | 100600.00
303 | tournament   | greedy_merge   | one_poin | 0.80    | 0.90     | 10%     | 75553.33       | 106000.00


==================================== Phase 3 - SIMPLE Configurations ====================================
Rank | Selection    | Crossover      | Mutation | CX Rate | Mut Rate | Elitism | Avg Fitness | Max Fitness
-----|--------------|----------------|----------|---------|----------|---------|-------------|------------
  2 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 10%     | 89732.00       | 121600.00
  1 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 5%     | 89738.00       | 120200.00
  8 | tournament   | greedy_merge   | one_poin | 0.60    | 0.80     | 10%     | 89902.00       | 124200.00
  5 | tournament   | greedy_merge   | one_poin | 0.60    | 0.70     | 10%     | 87768.00       | 117400.00
  3 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 25%     | 84830.00       | 114600.00


==================================== Phase 4 - SIMPLE Configurations ====================================
Rank | Selection    | Crossover      | Mutation | CX Rate | Mut Rate | Elitism | Avg Fitness | Max Fitness
-----|--------------|----------------|----------|---------|----------|---------|-------------|------------
  1 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 5%     | 90281.00       | 138100.00
  2 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 10%     | 92803.00       | 125800.00
  3 | tournament   | greedy_merge   | one_poin | 0.60    | 0.60     | 25%     | 86251.00       | 122700.00
  4 | tournament   | greedy_merge   | one_poin | 0.60    | 0.80     | 10%     | 88314.00       | 114000.00
  5 | tournament   | greedy_merge   | one_poin | 0.60    | 0.90     | 10%     | 84850.00       | 117500.00
```

Figure 21

```
==================================== Phase 2 - ADVANCED Configurations ====================================
Rank | Selection    | Crossover      | Mutation | CX Rate | Mut Rate | Elitism | Avg Fitness | Max Fitness
-----|--------------|----------------|----------|---------|----------|---------|-------------|------------
 60 | boltzmann    | greedy_merge   | one_poin | 0.80    | 0.90     | 10%     | 110426.67       | 142500.00
156 | boltzmann    | partially      | one_poin | 0.40    | 0.90     | 10%     | 111060.00       | 139200.00
 34 | boltzmann    | greedy_merge   | one_poin | 0.40    | 0.90     | 0%      | 108340.00       | 138400.00
163 | boltzmann    | partially      | one_poin | 0.70    | 0.80     | 0%      | 107466.67       | 145000.00
311 | boltzmann    | table_preser   | one_poin | 0.90    | 0.90     | 5%      | 110433.33       | 137800.00


==================================== Phase 3 - ADVANCED Configurations ====================================
Rank | Selection    | Crossover      | Mutation | CX Rate | Mut Rate | Elitism | Avg Fitness | Max Fitness
-----|--------------|----------------|----------|---------|----------|---------|-------------|------------
 16 | boltzmann    | greedy_merge   | one_poin | 0.70    | 0.90     | 0%      | 145926.00       | 162500.00
 56 | boltzmann    | partially      | one_poin | 0.90    | 0.80     | 5%      | 146256.00       | 161100.00
 36 | boltzmann    | partially      | one_poin | 0.30    | 0.90     | 10%     | 144220.00       | 159700.00
 62 | boltzmann    | table_preser   | one_poin | 0.30    | 0.80     | 5%      | 144812.00       | 159400.00
 49 | boltzmann    | partially      | one_poin | 0.80    | 0.80     | 0%      | 144750.00       | 160700.00


==================================== Phase 4 - ADVANCED Configurations ====================================
Rank | Selection    | Crossover      | Mutation | CX Rate | Mut Rate | Elitism | Avg Fitness | Max Fitness
-----|--------------|----------------|----------|---------|----------|---------|-------------|------------
 10 | boltzmann    | partially      | one_poin | 0.80    | 0.90     | 5%      | 144852.00       | 159700.00
  5 | boltzmann    | greedy_merge   | one_poin | 0.90    | 0.90     | 0%      | 146463.00       | 157500.00
  3 | boltzmann    | greedy_merge   | one_poin | 0.30    | 0.80     | 10%     | 143941.00       | 157700.00
  1 | boltzmann    | greedy_merge   | one_poin | 0.30    | 0.80     | 0%      | 144909.00       | 159100.00
  8 | boltzmann    | greedy_merge   | one_poin | 0.70    | 0.90     | 0%      | 144879.00       | 158100.00
```

Figure 22

sel	Cx	Mut	cx_rate	mut_rate	Elitism	early_stop	Delta	avg_fitness	max_fitness	avg_gen	std_fitness
Tournament	greedy_merge	one_point	0.6	0.6	0	10	50	149814	161900	128.08	5180.46
Boltzmann	Partially	one_point	0.9	0.8	0.05	10	50	153922	162200	36.97	4713.95