Computational Intelligence For Optimization

Timetable / Scheduling Problem

Git Repository: PatriciaTorres10/CIFO_OlympicTeam (github.com)

Team: Olympic Players

Guilherme Curioso – 20230558

Mafalda Antunes – 20201565

Matilde Simões – 20201502

Patrícia Torres – 20201518

Index

Division of Labour

Guilherme Curioso developed the problem definition and design of the fitness function. Matilde Simões was responsible for the population and individual representation, as well as the implementation of the selection and crossover algorithms. Mafalda Antunes and Patrícia Torres focused on developing the mutation algorithms, optimising the grid search, and the composition of the final report. The entire group actively participated in drawing conclusions and conducting the final review.

Introduction

This project aims to solve an optimisation problem using Genetic Algorithms (GAs), specifically focusing on the Timetable/Scheduling problem for the Olympic Games.

An efficient and well-structured schedule is paramount given that the 2024 Olympic Games will be hosted in Paris. Coordinating multiple events across numerous venues within the city poses a significant challenge that demands an optimal solution to ensure smooth operations and a successful event.

The primary objective of this project is to develop an optimal schedule, considering various constraints to create a seamless and effective event schedule for the 2024 Paris Olympic Games.

Problem Definition

For this project, we created sample data of fifteen venues, fifteen events, and sixty-three athletes. Each event has a specific venue where it can happen, and each athlete is associated with one or two events, depending on the sport.

The starting date was defined as the 26th of July of 2024 and the ending date as the 11th of August of 2024, the games will occur every day between 9am and 10pm.

In addition, several constraints were also created. These constraints add a penalty value to the fitness function each time they are violated.

Therefore, this problem was defined as a minimization problem once a low fitness value means that fewer constraints were violated.

Constraints

Each constraint was classified as hard or soft to determine the appropriate penalties for violations.

Defined constraints:

1. **Exceed Event Duration:** penalises all events that exceed their stipulated duration time. It was considered a soft constraint.
2. **Overlap Avoidance:** there could be several events sharing a venue (for example, both tennis singles and tennis doubles can happen at the Court) so, it is essential to penalise events scheduled simultaneously in the same venue. This was defined as a hard constraint given that it is critical for well-organized games.

3. **Extra Time Between Events:** buffer times help manage transitions between events, ensuring that venues can be prepared adequately, and spectators and athletes can move between locations without time pressure. It was considered a soft constraint.
4. **Daylight vs. Nighttime Events:** some events are mandatory to happen during the daytime while others can easily occur at night. Therefore, daylight start and end as well as the compulsory daylight events were defined. This is a hard constraint given that some events can only occur during the day for safety reasons.
5. **Athlete Rest and Recovery:** given that several athletes compete in more than one event they should have adequate breaks to maintain peak performance and prevent injuries. As this is not crucial for a smooth function of the games, it is a soft constraint.
6. **Multi-Event Conflicts:** the athletes competing in more than one event cannot be required to participate in two events simultaneously. Therefore, it is a hard constraint.

Fitness Function

The fitness function is a critical component of the Genetic Algorithm approach. This function was designed based on previously mentioned constraints to ensure a well-coordinated and successful Olympic Games in Paris 2024.

It started with initialising six variables to store the total penalties applied for each constraint. All the hard constraints were defined with a multiplier of 100. For the soft constraints, Exceed Event Duration has a multiplier of 15, Extra Time Between Events has a multiplier of 10, and Athlete Rest and Recovery has a multiplier of 30.

Each of the defined constraints is verified for all individuals in the population. The correspondent penalty is added for each of the violations verified. The fitness function returns a sum of all the penalties applied.

Furthermore, considering the chosen representation and the complexity of the problem only this fitness function was considered and tested in this project.

Fitness Sharing

A fitness-sharing technique was implemented to promote diversity within the population and prevent premature convergence to suboptimal solutions. Fitness sharing also ensures that multiple regions of the solution space are explored.

A threshold value of 0.1 was defined to determine the degree of similarity below which individuals are considered for fitness adjustment. This function compares two individuals and counts how many events they have in common. Then, each individual's fitness in the population is adjusted based on their similarity to other individuals. The adjustment penalises individuals that are too similar to others, encouraging diversity.

These functions are useful in this problem where maintaining a diverse population is crucial for avoiding local optima and ensuring a robust solution space search.

Population and Individual

In this problem, the schedule was considered as an Individual and a scheduling window was defined where events can happen every day between 9am and 22pm between the 26th of July and the 11th of August.

The Individual representation is presented as follows:

Best schedule: [{‘event_id’: , ‘start_time’: , ‘end_time’: ,’athlete_id’: , ‘venue_id’: }, {…},…]

This representation was selected given its clarity allowing for an easier manipulation of the population throughout the project development. Furthermore, this representation also enhances the interpretation of the achieved solution.

Selection

Selection is a fundamental component of genetic algorithms, responsible for choosing individuals from the current population to create the next generation. The effectiveness of the selection process directly impacts the convergence and diversity of the population, playing a crucial role in the overall success of the algorithm.

This project implemented three different selection methods: Tournament Selection, Fitness Proportionate Selection and Rank Selection.

Mutation

The mutation is a crucial operator in genetic algorithms as it introduces genetic diversity into the population, helping to prevent premature convergence to local optima. In this project, two mutation methods were implemented and analysed: swap mutation and inversion mutation.

Both methods work similarly starting by selecting two positions (events on the schedule(individual)). The swap mutation simply “swaps” those positions in the presentation. Inversion Mutation uses those positions as boundaries. The elements (events) within the selected range are reversed, while elements outside the range remain unchanged.

Crossover

Crossover is another essential operator in genetic algorithms, responsible for combining the genetic material of two parents to produce offspring. In this project, two crossover methods were implemented and compared: single-point crossover and uniform crossover.

Single-point crossover works by combining the representations of both parents at the crossover point. In contrast, uniform crossover iterates over each gene position and randomly selects a gene from one of the parents.

Final Schedule

To define the final schedule several evaluations and analyses were performed. Different combinations of crossover, mutation and selection methods were tested using the 'mutation_search' method.

When running these tests, elitism was kept True to preserve the best individual of each generation in the following ones, and a population of 100 individuals across 50 generations were the parameters applied. Additionally, this analysis was performed across 7 (this number was a compromise between efficiency and efficacy) different runs. The fitness value was obtained by calculating the median fitness of the different runs for each of the possible combinations.

To understand which combination presented the best performance several graphs were plotted:

- *Figure 1 Fitness evolution for different mutation methods using Uniform Crossover and Tournament Selection*
- *Figure 2 Fitness evolution for different mutation methods using Single Point Crossover and Tournament Selection*
- *Figure 3 Fitness evolution for different mutation methods using Uniform Crossover and Fitness Proportionate Selection*
- *Figure 4 Fitness evolution for different mutation methods using Single Point Crossover and Fitness Proportionate Selection*
- *Figure 5 Fitness evolution for different mutation methods using Uniform Crossover and Rank Selection*
- *Figure 6 Fitness evolution for different mutation methods using Single Point Crossover and Rank Selection*

Given the proximity of fitness values presented on the two first graphs (*Figure 1 Fitness evolution for different mutation methods using Uniform Crossover and Tournament Selection, Figure 2 Fitness evolution for different mutation methods using Single Point Crossover and Tournament Selection*), the ones which presented the best fitness values, it wasn't explicit which crossover method was the best. Therefore, a more profound evaluation of the best crossover operator to select was carried out.

The crossover search method, which works in a very similar way to the mutation search method, was applied with a population of 100 individuals across 50 generations using swap mutation (mutation operator which presented lower fitness values across the different graphs) and tournament selection (selection that presented lower fitness values across the different graphs) across 7 runs.

Based on the plotted graphs it was possible to conclude that **swap mutation**, **uniform crossover** and **tournament selection** were the operators who contributed to the lowest fitness values *(Figure 1 Fitness evolution for different mutation methods using Uniform Crossover and Tournament Selection, Figure 7 Fitness evolution for different crossover methods using Swap Mutation and Tournament Selection)*.

Throughout the analysis of the graphs, it was detected that different operators affect the convergence of the genetic algorithm. In the last pair of graphs where fitness proportionate selection and rank selection were tested *(Figure 3 Fitness evolution for different mutation methods using Uniform Crossover and Fitness Proportionate*

Selection, Figure 4 Fitness evolution for different mutation methods using Single Point Crossover and Fitness Proportionate Selection, Figure 5 Fitness evolution for different mutation methods using Uniform Crossover and Rank Selection, Figure 6 Fitness evolution for different mutation methods using Single Point Crossover and Rank Selection) the fitness values are stabilising at an earlier stage. This indicates that the population's fitness is no longer improving significantly which can be a sign of early convergence, suggesting that the genetic diversity in the population has reduced, leading to a lack of exploration in the solution space.

Additionally, parameter tunning was also implemented. Considering the operators selected above, a grid search was used to tune the population size, number of generations, crossover and mutation probabilities, and elitism. The grid search implementation followed the same reasoning as mutation and crossover search methods. It was performed throughout 7 different runs where the median of each generation for all the runs was then calculated.

The performance of the different combinations can be seen in *Figure 8 Median Fitness of the different parameters' combinations.*

The best parameters selected were: **'population_size': 200, 'num_generations': 100, 'crossover_probability': 0.8, 'mutation_probability': 0.1, 'elitism': True**

The best final solution found *(Table 1 Best Schedule)* presented a fitness value of 70. This solution violates the constraint "Extra Time Between Events" 7 times.

Conclusion

The results of our genetic algorithm implementation for the Olympic Games scheduling problem were promising, with only one constraint being violated.

Even though the violation happened several times this was the constraint to which it was attributed less weight. This indicates that the primary objectives of the schedule were largely met.

In the future, to further improve the results, the following could be implemented:

1. Increase dataset: Expanding the dataset would allow the algorithm to evolve more effectively, potentially leading to a more diverse population. This could contribute to a later convergency of the algorithm and consequently, leads to a better solution.
2. Increase the number of runs: With more computational resources, a higher number of runs would be beneficial to provide a better statistical understanding of the algorithm's performance and robustness.
3. Try more methods: Testing different crossover and mutation methods, as well as exploring other selection techniques, could yield improvements.

Overall, the obtained results were good, demonstrating a successful application of genetic algorithms in this complex scheduling problem.

References

Class notes retrieved from Moodle

Genetic Algorithms - mutation. (n.d.). Retrieved from https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm

Annexes

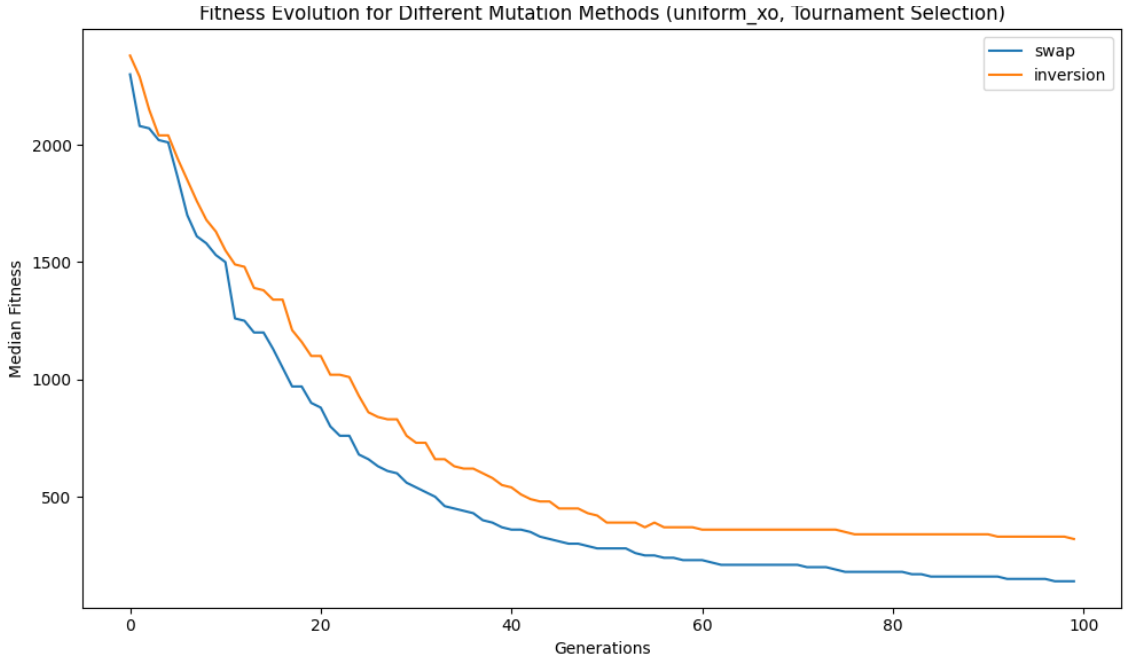

Figure 1 Fitness evolution for different mutation methods using Uniform Crossover and Tournament Selection

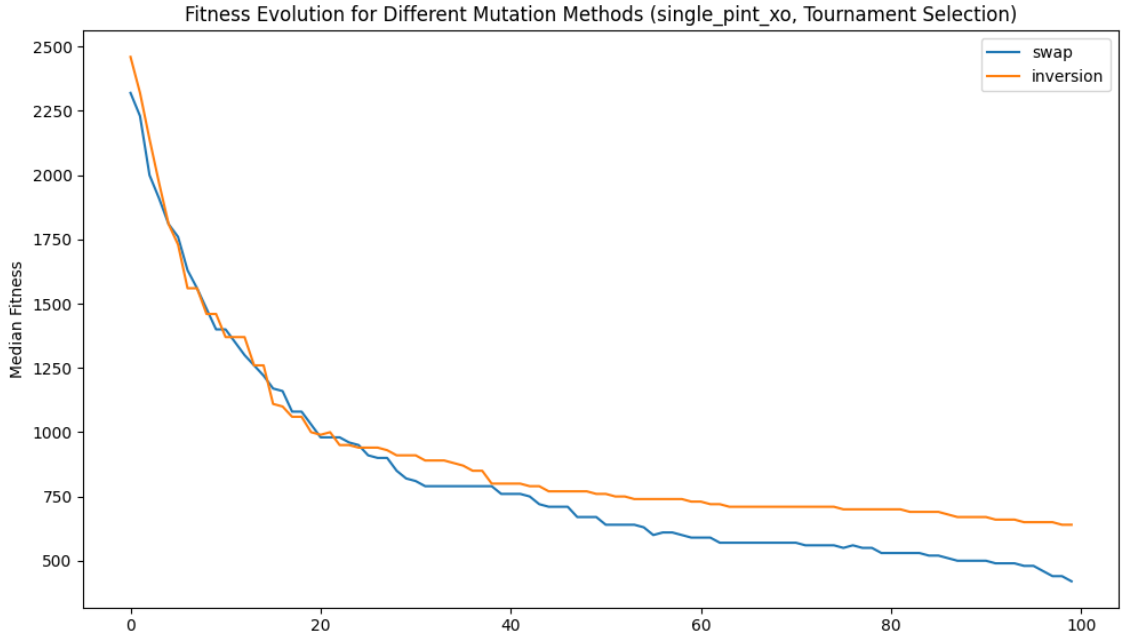

Figure 2 Fitness evolution for different mutation methods using Single Point Crossover and Tournament Selection

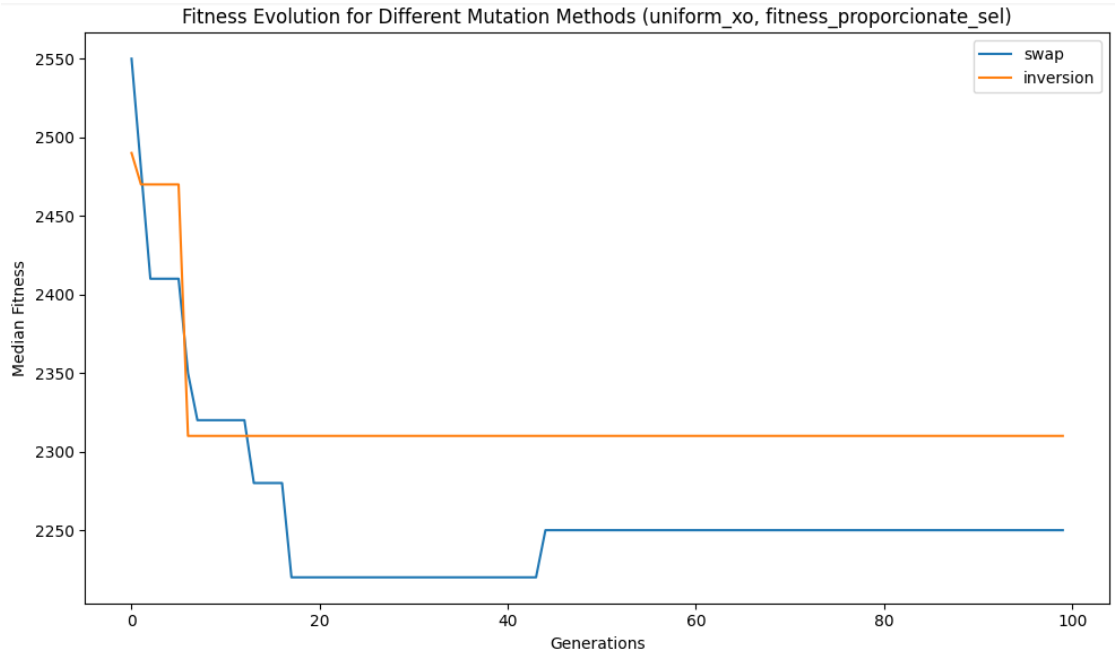

Figure 3 Fitness evolution for different mutation methods using Uniform Crossover and Fitness Proportionate Selection

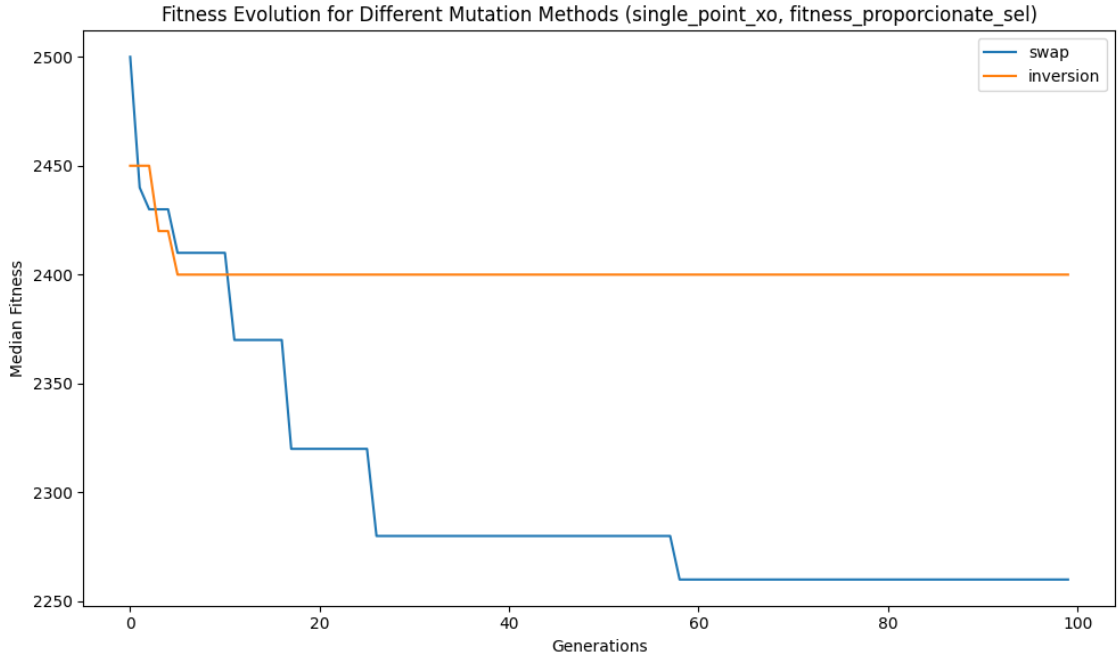

Figure 4 Fitness evolution for different mutation methods using Single Point Crossover and Fitness Proportionate Selection

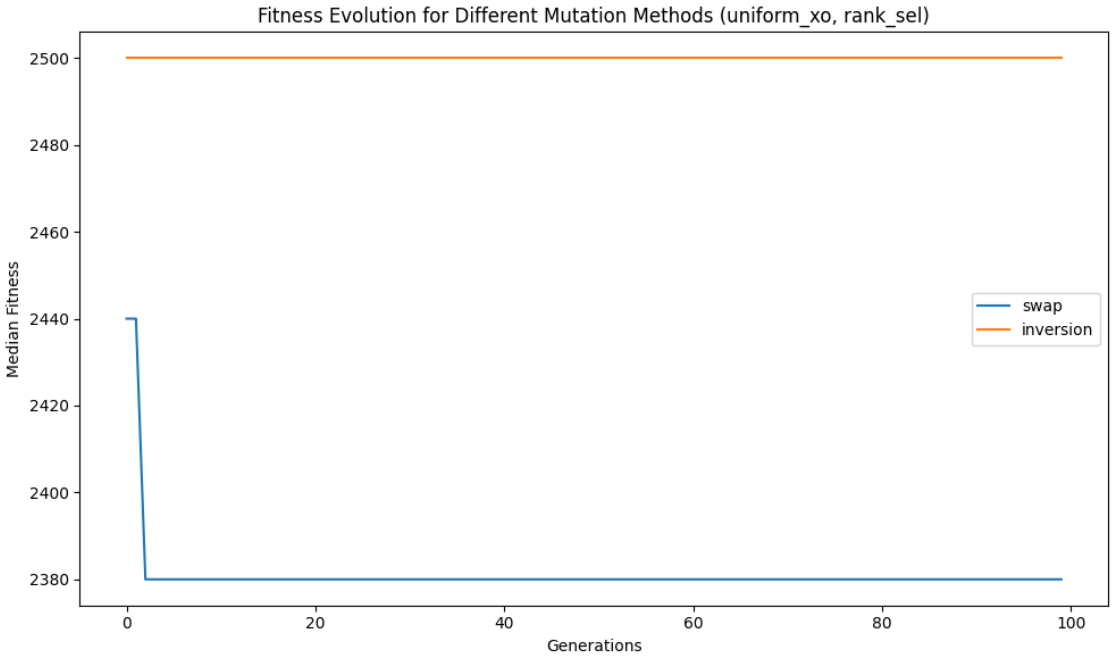

Figure 5 Fitness evolution for different mutation methods using Uniform Crossover and Rank Selection

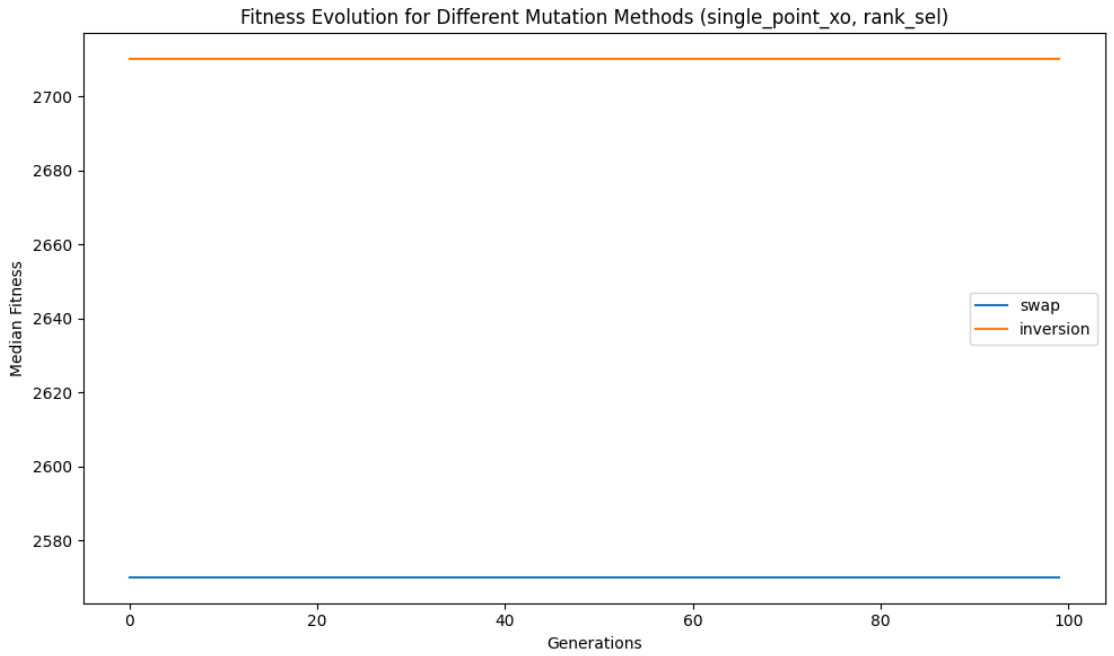

Figure 6 Fitness evolution for different mutation methods using Single Point Crossover and Rank Selection

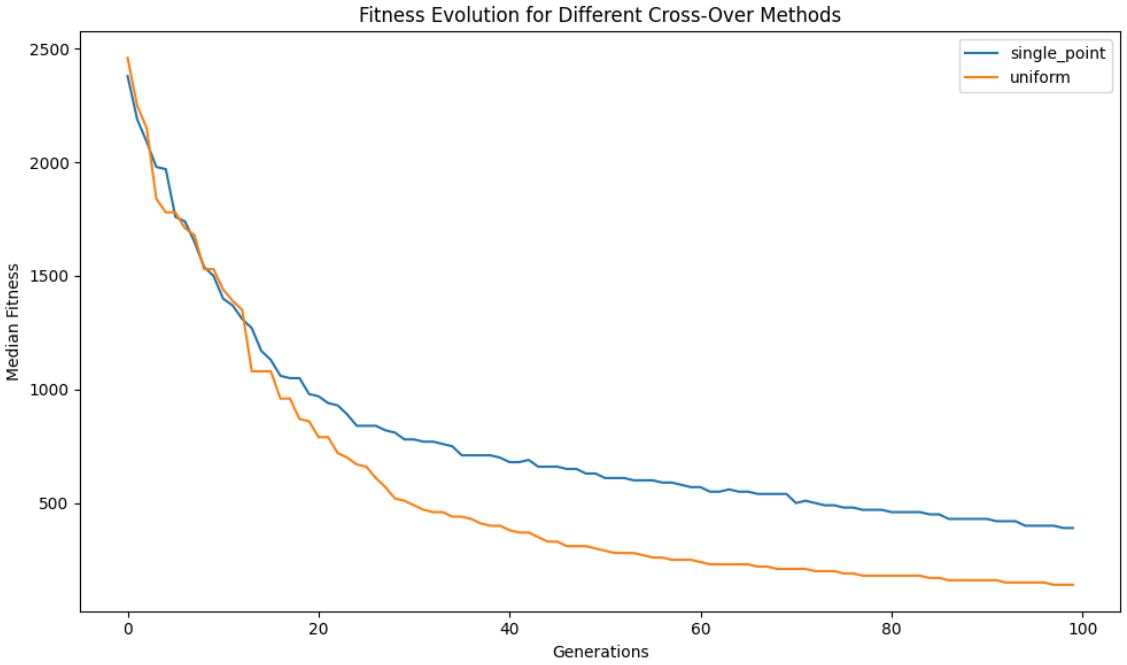

Figure 7 Fitness evolution for different crossover methods using Swap Mutation and Tournament Selection

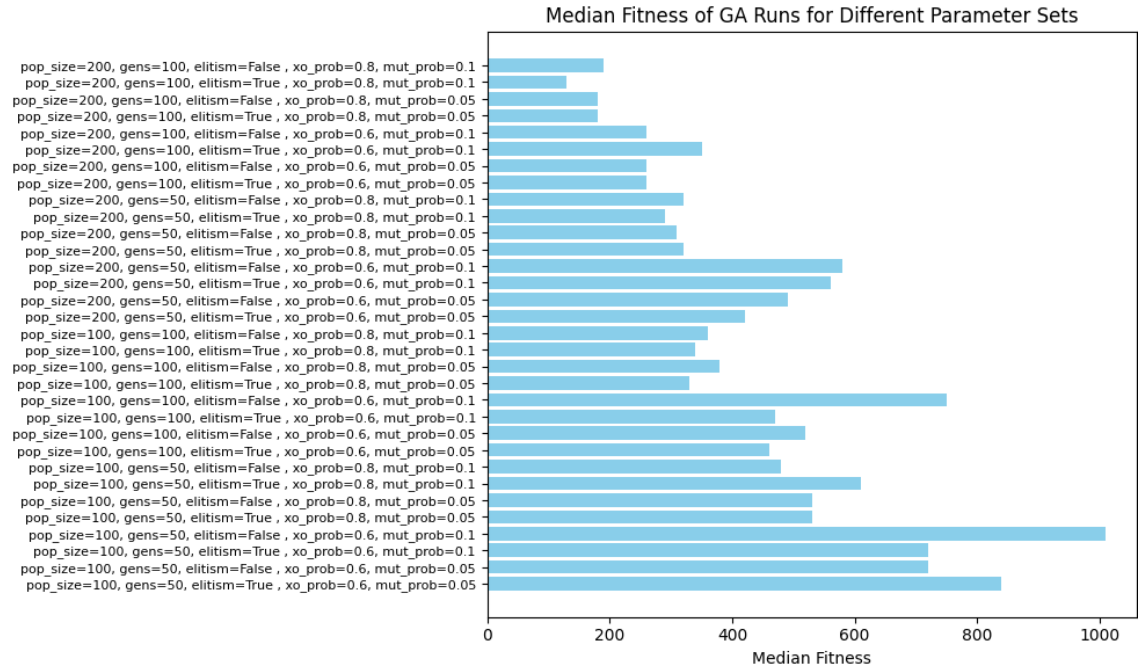

Figure 8 Median Fitness of the different parameters' combinations

Table 1 Best Schedule

event_id	start_time	end_time	athlete_id	venue_id
1	2024-07-26 09:38:00	2024-07-26 12:38:00	1	8
3	2024-07-26 11:49:00	2024-07-26 12:49:00	2	3
5	2024-07-26 14:34:00	2024-07-26 16:34:00	3	5
7	2024-07-26 14:58:00	2024-07-26 17:58:00	4	6
9	2024-07-26 15:29:00	2024-07-26 16:29:00	5	15
11	2024-07-26 18:15:00	2024-07-26 19:15:00	6	13
7	2024-07-27 10:45:00	2024-07-27 12:45:00	17	6
1	2024-07-27 11:34:00	2024-07-27 13:34:00	8	8
3	2024-07-27 17:44:00	2024-07-27 20:44:00	9	3
11	2024-07-27 18:05:00	2024-07-27 20:05:00	6	13
3	2024-07-28 12:38:00	2024-07-28 13:38:00	11	3
3	2024-07-28 14:48:00	2024-07-28 15:48:00	12	3
4	2024-07-28 15:17:00	2024-07-28 17:17:00	13	9
4	2024-07-28 15:51:00	2024-07-28 16:51:00	14	9
8	2024-07-28 18:43:00	2024-07-28 19:43:00	15	7
8	2024-07-29 09:03:00	2024-07-29 11:03:00	16	7
11	2024-07-29 09:23:00	2024-07-29 12:23:00	47	13
7	2024-07-29 12:12:00	2024-07-29 14:12:00	17	6
2	2024-07-29 15:37:00	2024-07-29 17:37:00	18	6
1	2024-07-29 15:52:00	2024-07-29 16:52:00	25	8
2	2024-07-29 16:20:00	2024-07-29 19:20:00	19	6
7	2024-07-29 18:06:00	2024-07-29 20:06:00	19	6
9	2024-07-29 18:14:00	2024-07-29 20:14:00	42	15

7	2024-07-30 12:07:00	2024-07-30 15:07:00	20	6
8	2024-07-30 13:05:00	2024-07-30 15:05:00	37	7
9	2024-07-30 16:59:00	2024-07-30 17:59:00	22	15
11	2024-07-31 09:42:00	2024-07-31 10:42:00	47	13
9	2024-07-31 11:35:00	2024-07-31 14:35:00	24	15
1	2024-07-31 11:38:00	2024-07-31 12:38:00	25	8
1	2024-07-31 15:49:00	2024-07-31 17:49:00	26	8
3	2024-08-01 10:54:00	2024-08-01 12:54:00	27	3
6	2024-08-01 11:42:00	2024-08-01 12:42:00	35	10
4	2024-08-01 16:22:00	2024-08-01 18:22:00	29	9
4	2024-08-01 17:58:00	2024-08-01 18:58:00	30	9
5	2024-08-02 09:20:00	2024-08-02 10:20:00	31	5
5	2024-08-02 09:44:00	2024-08-02 10:44:00	32	5
6	2024-08-02 13:17:00	2024-08-02 14:17:00	33	10
6	2024-08-03 09:14:00	2024-08-03 12:14:00	34	10
6	2024-08-03 09:53:00	2024-08-03 10:53:00	35	10
6	2024-08-03 14:46:00	2024-08-03 15:46:00	36	10
9	2024-08-04 09:17:00	2024-08-04 11:17:00	21	15
8	2024-08-04 09:42:00	2024-08-04 10:42:00	38	7
4	2024-08-04 12:01:00	2024-08-04 14:01:00	14	9
9	2024-08-04 15:43:00	2024-08-04 16:43:00	40	15
9	2024-08-04 17:50:00	2024-08-04 19:50:00	41	15
9	2024-08-04 18:57:00	2024-08-04 20:57:00	42	15
10	2024-08-04 20:03:00	2024-08-04 22:03:00	43	8

13	2024-08-05 10:05:00	2024-08-05 11:05:00	43	8
10	2024-08-06 09:31:00	2024-08-06 10:31:00	44	8
9	2024-08-06 09:40:00	2024-08-06 11:40:00	39	15
13	2024-08-06 16:02:00	2024-08-06 17:02:00	50	8
10	2024-08-06 16:25:00	2024-08-06 18:25:00	46	8
11	2024-08-06 19:05:00	2024-08-06 21:05:00	47	13
4	2024-08-07 14:19:00	2024-08-07 17:19:00	29	9
11	2024-08-07 16:25:00	2024-08-07 17:25:00	49	13
9	2024-08-07 17:21:00	2024-08-07 18:21:00	41	15
13	2024-08-07 17:56:00	2024-08-07 19:56:00	45	8
10	2024-08-08 11:53:00	2024-08-08 12:53:00	51	8
13	2024-08-08 18:46:00	2024-08-08 20:46:00	51	8
14	2024-08-09 09:43:00	2024-08-09 12:43:00	52	14
14	2024-08-09 11:57:00	2024-08-09 12:57:00	53	14
14	2024-08-09 13:37:00	2024-08-09 14:37:00	54	14
14	2024-08-09 14:55:00	2024-08-09 16:55:00	55	14
15	2024-08-09 15:31:00	2024-08-09 16:31:00	56	3
15	2024-08-09 16:05:00	2024-08-09 18:05:00	57	3
15	2024-08-10 09:48:00	2024-08-10 11:48:00	58	3
15	2024-08-10 10:26:00	2024-08-10 11:26:00	59	3
12	2024-08-10 14:12:00	2024-08-10 16:12:00	60	5
12	2024-08-10 15:17:00	2024-08-10 17:17:00	61	5
12	2024-08-11 12:39:00	2024-08-11 14:39:00	62	5
12	2024-08-11 18:49:00	2024-08-11 20:49:00	63	5