Evolutionary Algorithm

Project: Sport Tournament Scheduling

Introduction

Overview:

Sports tournament scheduling is a critical problem in the field of operations research and optimization. It involves creating an efficient and fair schedule for a set of matches between teams or individuals while adhering to specific rules and constraints. This problem is widely studied due to its complexity, which arises from the need to balance fairness, avoid conflicts, and ensure an even distribution of matches over the tournament duration.

After Researching This project focuses on a general approach to sports tournament scheduling without being limited to a specific sport. The solution is designed to be adaptable for different types of tournaments, including round-robin, knockout, and league formats. Our approach utilizes a Genetic Algorithm with an Island Model, a powerful optimization technique, to generate efficient schedules that meet the defined constraints and objectives.

Problem Description:

The primary objective of this project is to design an optimized schedule for a sports tournament that can accommodate various types of games, including team-to-team matches (e.g., football, basketball) and one-to-

one matches (e.g., tennis, boxing). The scheduling process must ensure that:

- Each team or player competes against all other participants as required.
- The schedule avoids conflicts and overlaps, ensuring a smooth tournament flow.
- The distribution of matches is balanced, avoiding scenarios where a team plays excessively on consecutive days or has long idle periods.
- Constraints such as venue availability and team preferences are considered when applicable.

After conducting research on sports tournament scheduling, we identified various approaches used in solving this problem. These approaches vary in their focus, such as optimizing for travel distance, accommodating specific sports, or targeting different tournament formats. We found that many solutions focus on specialized cases, such as:

- Scheduling for specific sports (e.g., football leagues, tennis tournaments).
- Prioritizing travel optimization to minimize team movement.
- Balancing match frequency to prevent player fatigue.

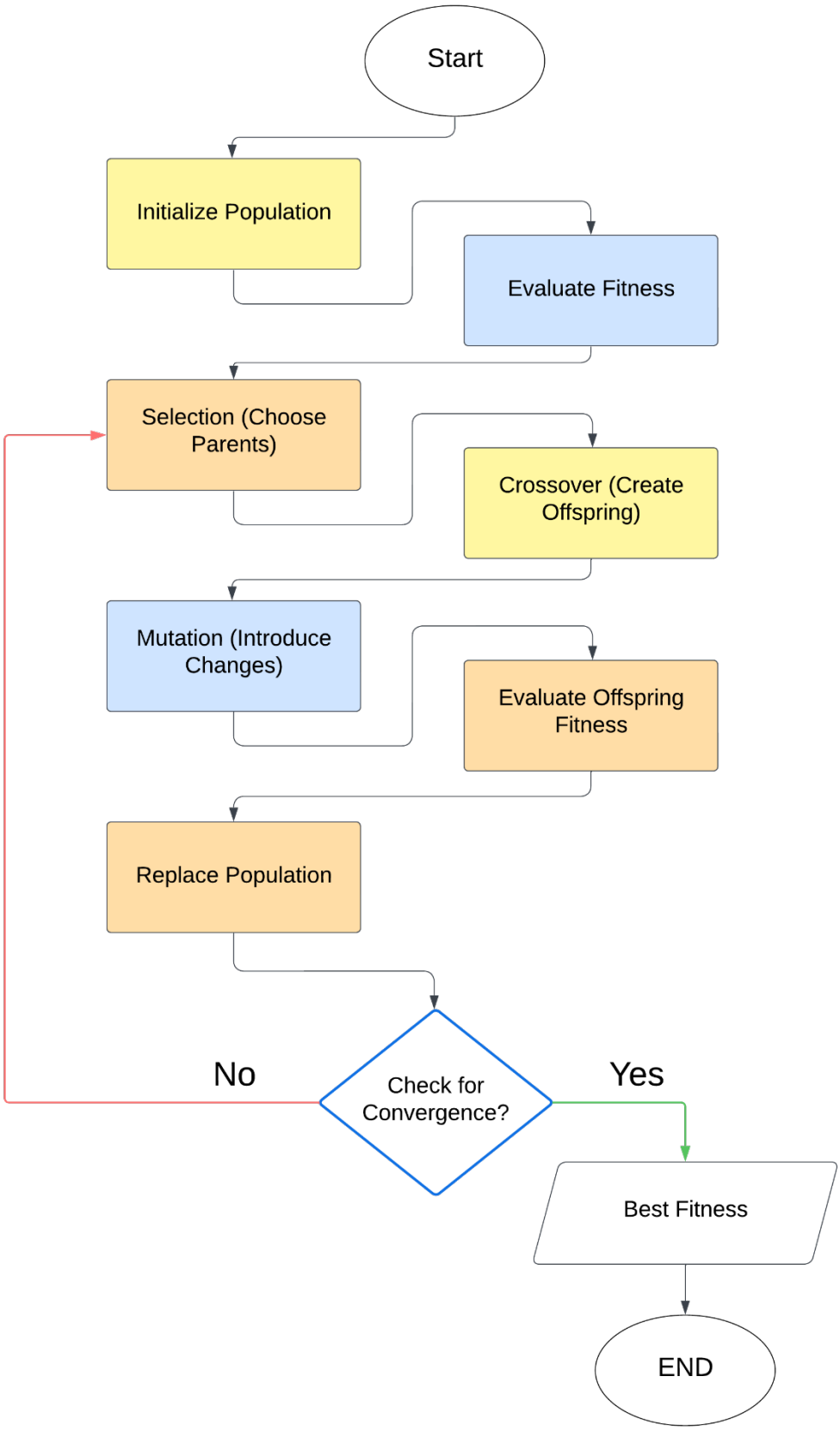
Project Scope:

Our project focuses on a general approach to tournament scheduling that is adaptable to both team-to-team and one-to-one formats. This decision ensures our solution can be applied to a wide range of sports and tournament types, providing greater flexibility and utility. Our chosen method leverages a Genetic Algorithm with an Island Model, which offers a robust optimization framework suitable for this complex problem.

Contents

Solution Approach

Our solution to the sports tournament scheduling problem is based on a Genetic Algorithm (GA), a powerful metaheuristic inspired by the principles of natural selection and evolution. The GA begins with an initial population of randomly generated schedules, which are iteratively improved through the processes of selection, crossover, and mutation. Our implementation is enhanced using an **Island Model**, where multiple sub-populations (islands) evolve independently, periodically exchanging individuals to maintain genetic diversity. This approach ensures a robust search of the solution space and improves the algorithm's ability to escape local optima. The final solution is selected as the best-performing schedule based on our defined fitness criteria, which prioritize conflict avoidance, balanced match distribution, and efficient scheduling.


Start
Initialize Population
Evaluate Fitness
Selection (Choose Parents)
Crossover (Create Offspring)
Mutation (Introduce Changes)
Evaluate Offspring Fitness
Replace Population
Check for Convergence?
No
Yes
Best Fitness
END

Constraints:

In our sports tournament scheduling problem, several constraints are considered to ensure the generated schedule is practical, fair, and efficient. These constraints include:

- Team Availability: Each team can only play one match per day, preventing scheduling conflicts for players.
- Venue Availability: Each venue has a limited capacity and can only host a certain number of matches per day, with enforced rest periods (venue_rest) to avoid overuse.
- Match Duration and Timing: Each match has a specified duration, and matches can only be scheduled within the defined daily operational hours (daily_start_hr to daily_end_hr).
- Balanced Rest Days for Teams: Teams are ensured a minimum number of rest days between matches to avoid player fatigue.
- Fair Distribution: The number of matches is balanced across tournament days to ensure a smooth schedule without excessive congestion or long idle periods.

Stage of Evoluation:

Our project is designed with several key stages that guide the optimization process:

1. Representation

Each individual is represented as **a list of tuples** (match, venue, day, start_hour). This format is chosen because it clearly separates the core scheduling components, making it easy to detect conflicts and optimizes and it can implicit **represent it as premutation.**

(TEAM 1 VS TEAM 2 , VENUE , DAY , TIME)

2. Selection

Methods Used: Tournament Selection and Roulette Wheel Selection.

We Choose Tournament Selection offers a balance of exploitation and exploration, while Roulette Wheel Selection introduces probabilistic diversity. We avoided rank selection due to its sensitivity to minor fitness differences.

3. Crossover

Methods Used: Uniform Crossover and One-Point Crossover.

We Choose uniform Crossover maintains diversity, while One-Point Crossover can preserve sequence integrity. We avoided Two-Point Crossover because it may disrupt scheduling sequences as can make conflict in Venue (same venue used in muiltple match at same time) and can change the order of match representation.

4. Mutation

Methods Used: Swap Mutation and Reschedule Mutation.

We Choose Swap Mutation preserves match assignments but changes their order, while Reschedule Mutation allows direct adjustment of timing and venue.

5. Survivor Selection

Methods Used: Steady-State, Elitism, Generational, $(\mu + \lambda)$ Selection.

6. Island Model

Approach for Diversity: Multiple populations (islands) evolving independently, with periodic migration.

In which enhances genetic diversity, avoids local optima, and accelerates convergence. We avoided a single-population model because it is more prone to stagnation.

Testing & Report

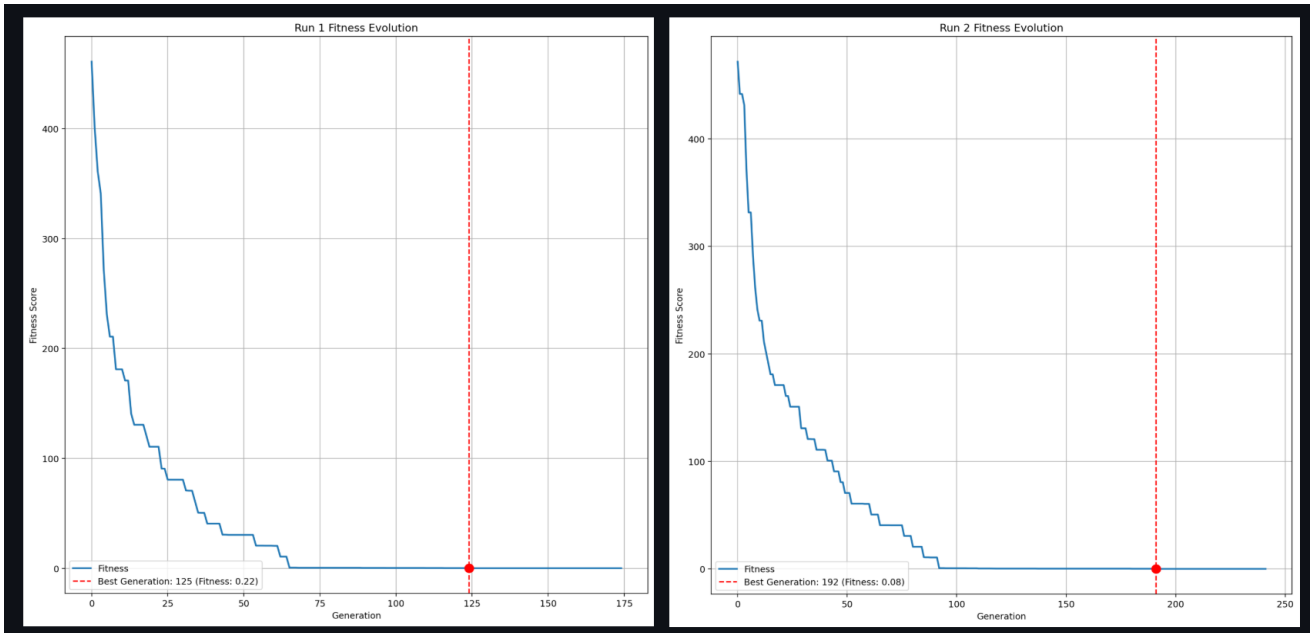
We report the testing results of each method used in the Genetic Algorithm (GA) for sports tournament scheduling. Our approach involves systematically evaluating each component of the GA, starting with the initialization methods.

We begin by comparing the performance of random and greedy initialization to identify which method generates better initial populations.

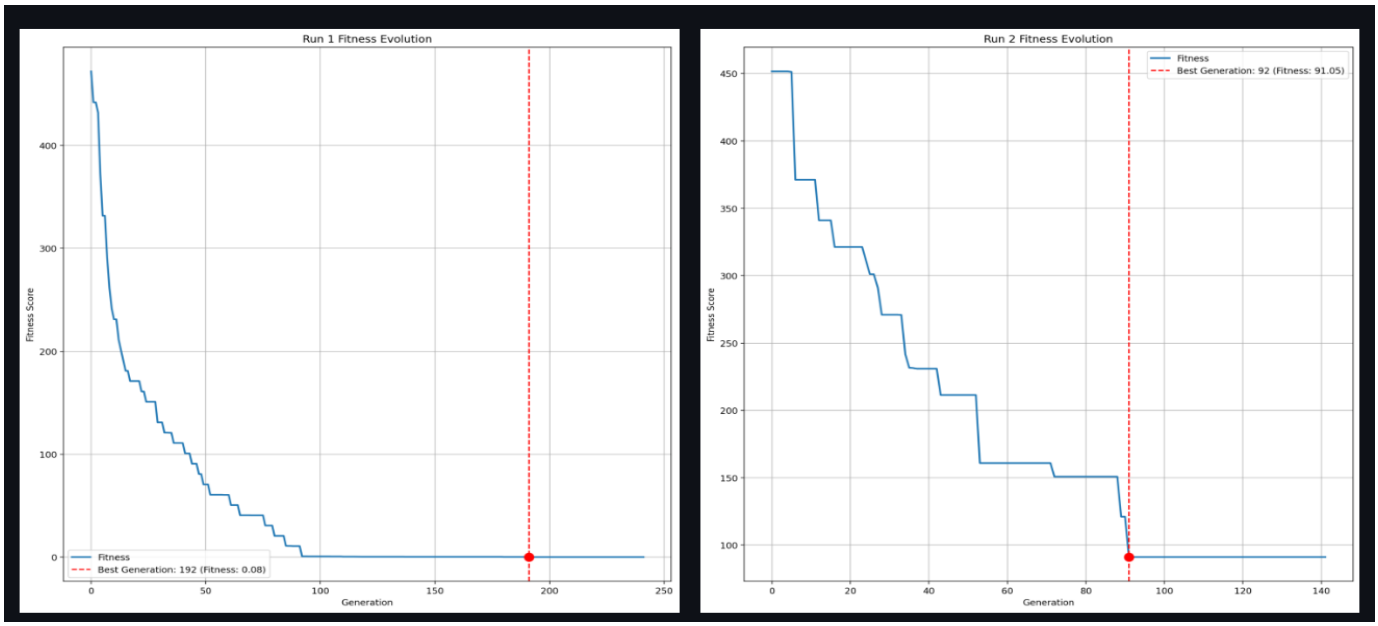
Once the best initialization method is determined, we proceed to test different selection methods (tournament and roulette wheel) while keeping the best initialization method fixed. The process continues sequentially for crossover (one-point and uniform), mutation (swap and reschedule), and survivor selection (elitism, steady-state, and generational).

This step-by-step evaluation allows us to isolate the impact of each method and optimize the configuration of the Genetic Algorithm for the best scheduling results.

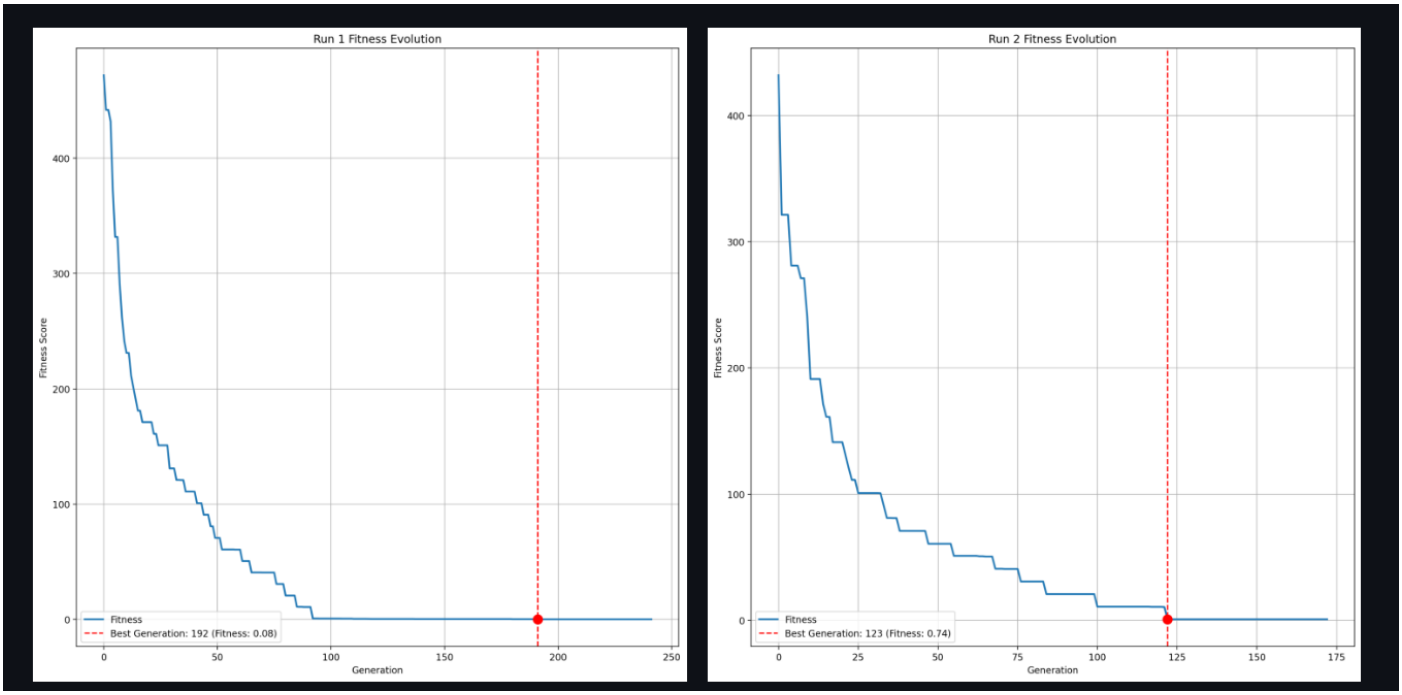
1. Random V.S Greedy Initialization:
    - Greedy fitness function with best one = **0.08**
      (with Best One with Configuration Greedy, Tournament, Uniform, Reschedule, Steady-State)



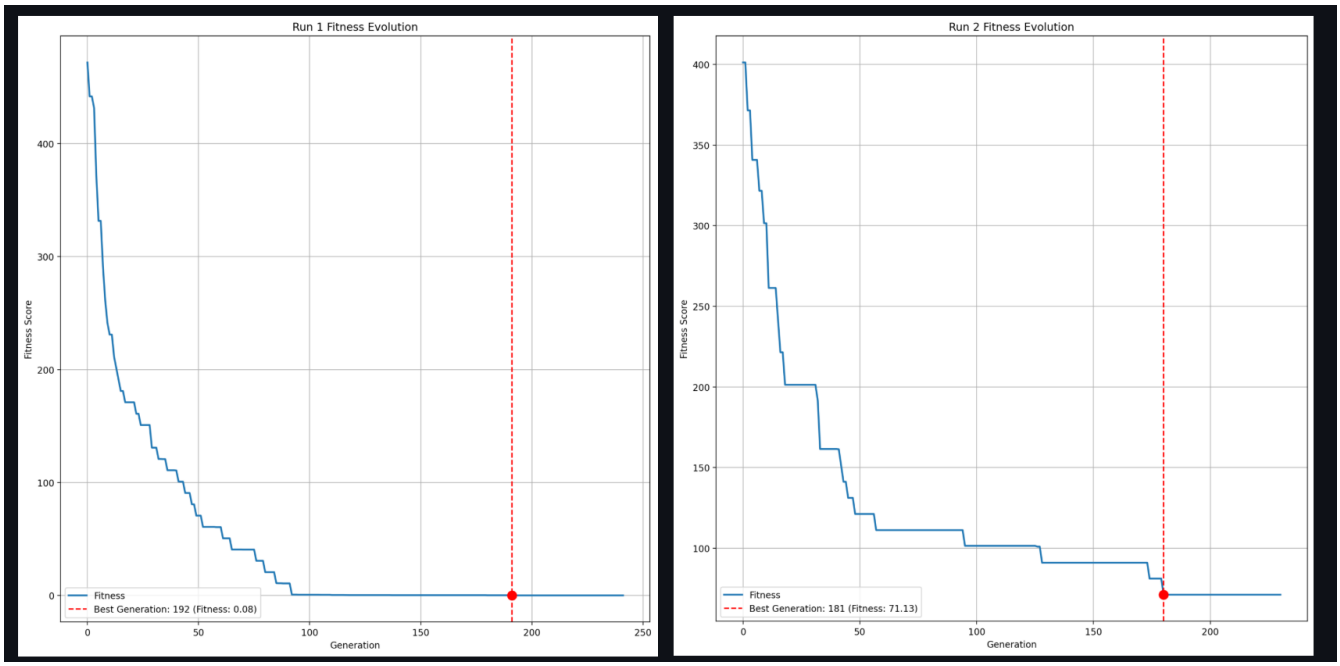
2. Tournament V.S Roulette Selection:
    - Roulette is worst one as it is logic in which converge in point with Better one not discovers the other Area **91.05**.



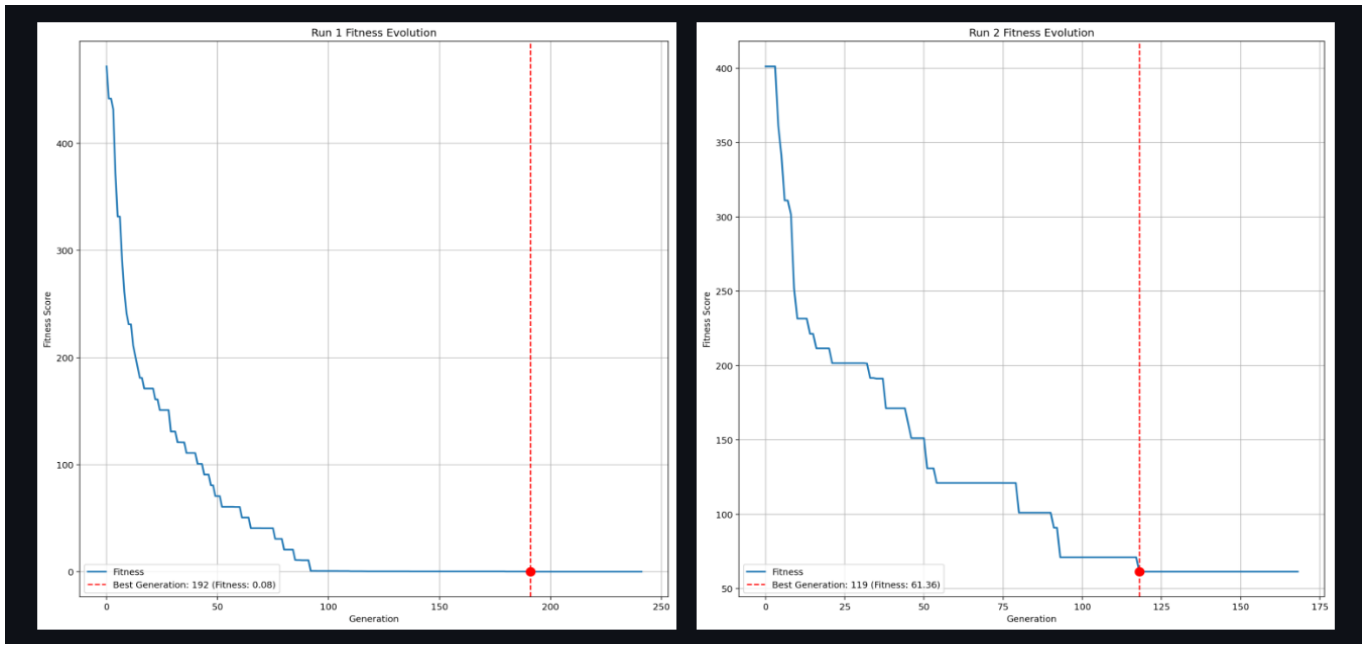
3. Uniform V.S One Point Crossover:
    - The two is the same but with number the uniform is best with very small difference (it totally depends on initialization and random).

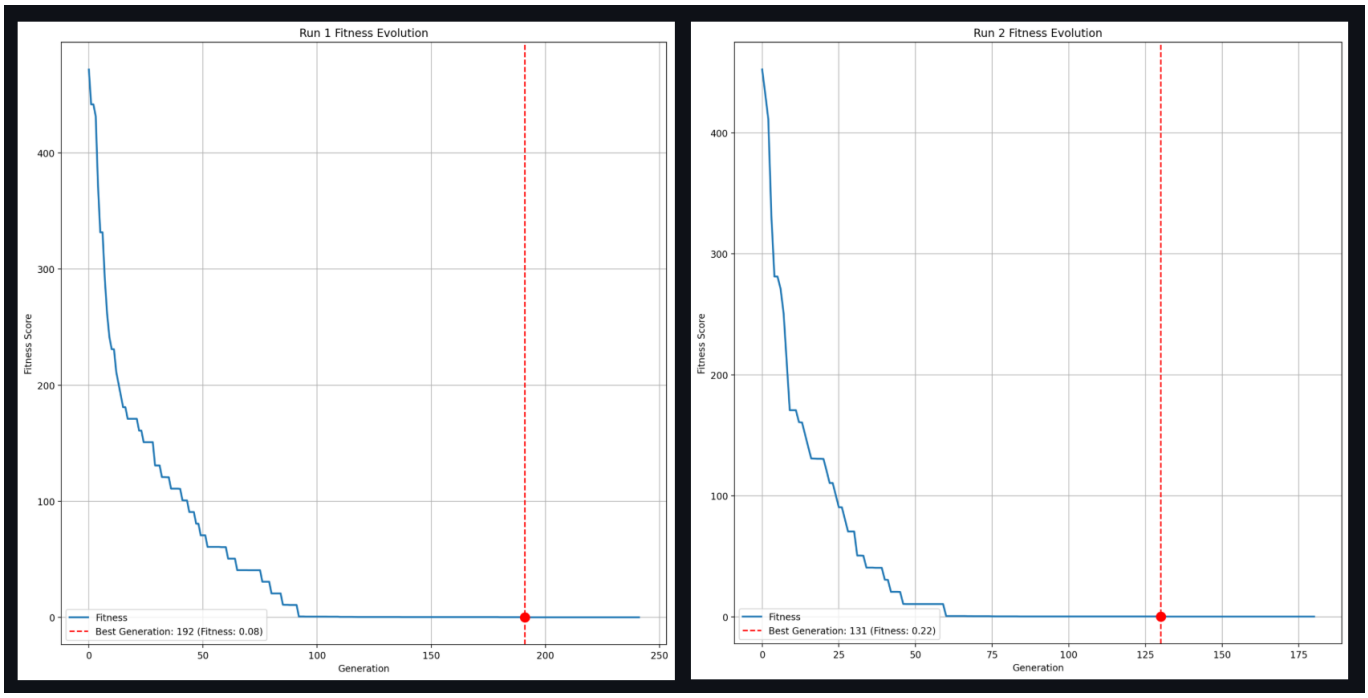


4. Reschedule V.S Swap Mutation:
    - Swap here replace the match with other so if initialization assign parameter value for venue and time this is still exists, so Reschedule play one important point in evaluation for other range of this value.

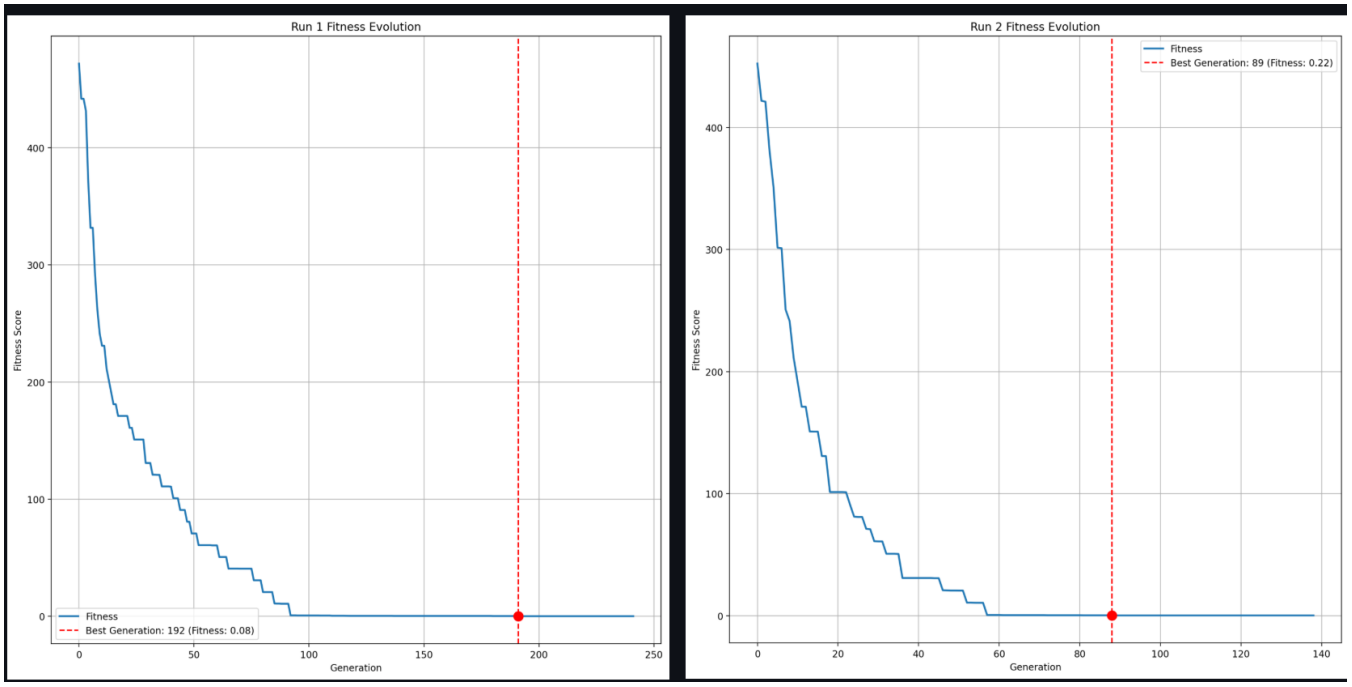


5. Steady-State V.S Generational, Elitism, (μ + λ):
   - Generational is worst one as this gets all new offspring and discard the other with **(61.36)**, but Steady, Elitism, (μ + λ) have some preserve the best old one.





Steady-State VS Elitism



Steady-State VS (μ + λ)

Optimal Configuration Selection:

After extensive testing and evaluation, the best configuration for the Genetic Algorithm was determined to be:

- Initialization: Greedy.
- Selection: Tournament.
- Crossover: Uniform.
- Mutation: Reschedule.
- Survivor Selection: Steady-State.

This configuration consistently produced the best schedules with a final average fitness of 0.08, indicating a highly optimized solution. The greedy initialization provided a strong starting population, while tournament selection ensured that strong candidates were consistently chosen as parents. Uniform crossover-maintained diversity, and reschedule mutation allowed for fine-tuned adjustments. Finally, steady-state selection maintained high-quality solutions across generations without losing diversity.