COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

SPRING SEMESTER 24/25

SPORTS LEAGUE OPTIMIZATION USING GENETIC ALGORITHMS

Group AE

Diogo Duarte, 20240525

Rodrigo Sardinha, 20211627

Rui Luz, 20211628

https://github.com/ruibluz/CIFOProject---GroupAE

1. Introduction

This report addresses the Sports League Optimization problem, where the goal is to build balanced football teams while meeting specific constraints related to player positions and team cost. A Genetic Algorithm was developed to generate valid team configurations and improve balance across the league. The report outlines the problem setup, implementation details, and results of the optimization process.

2. Formal Definition of the Problem

2.1 Problem Overview

The goal of the problem is to form five football teams from a fixed group of 35 players. Each player has a defined position, a skill rating, and a cost. The aim is to build a league where the teams are as balanced as possible in terms of skill. The challenge lies in finding valid team combinations that respect the rules while keeping the overall distribution of talent fair across all teams.

2.2 Constraints

To be valid, a solution must follow these rules:

- Each team must have exactly 7 players, with this structure:
  - 1 Goalkeeper (GK)
  - 2 Defenders (DEF)
  - 2 Midfielders (MID)
  - 2 Forwards (FWD)
- No player can appear in more than one team.
- All 35 players must be used — none can be left out.
- The total budget of each team can't be more than €750 million.

2.3 Individual Representation

In our implementation, an individual represents a complete league made up of 5 teams. Each team is built using the `Team` class and includes exactly 7 players. Every player has three main attributes: position, skill rating, and cost.

The structure of each team is defined in advance (1 GK, 2 DEF, 2 MID, 2 FWD), and the total cost must stay under €750 million. To ensure these rules are followed, individuals are created using the `LeagueIndividual` class, which builds teams by selecting players grouped by position and checking that all constraints are satisfied.

The individual also stores its fitness value, which is calculated based on how balanced the league is, measured by the standard deviation of the teams' average skill levels.

2.4 Search Space

The search space in this project includes all valid ways of forming a league with 5 teams, using the 35 available players. A solution belongs to the search space only if it respects the rules explained earlier.

In theory, the search space covers all possible individuals the algorithm could explore. In practice, though, most random configurations would violate at least one constraint and are therefore not useful. To avoid this, the `LeagueIndividual` class handles team generation through a controlled process, and the `is_valid()` method is used throughout mutation and crossover to ensure that constraints are met. If an invalid configuration is detected, it is simply skipped or not accepted, and the algorithm continues searching for a valid alternative.

This way, the algorithm focuses only on valid solutions, keeping the search within the feasible space by design.

2.5 Fitness Function

The fitness function is used to measure how balanced a league is by evaluating the distribution of player skill across the five teams. The aim is to minimize the differences in average skill between teams, which is done by calculating the **standard deviation** of the teams' average skill levels — the lower the value, the more balanced the league.

Before computing this value, the function checks whether the solution satisfies the constraints. These checks are done inside the `evalute_fitness()` method, which iterates through the teams and uses methods like `is_valid()` to confirm that the structure and composition of the league are correct. If any team is invalid or a player appears more than once, the function returns `float('inf')`, marking the solution as unusable.

This aligns with the general principle in Genetic Algorithms that fitness should reflect both solution quality and constraint satisfaction — guiding the search toward feasible and optimal regions of the solution space.

3. Genetic Operators

3.1 Selection Mechanisms

Mimicking the Darwinian principles of evolution and adaptation, genetic operators are applied to a selection on individuals rather than the entire population. For this effect, we

applied a set of selection algorithms, based on fitness, to approach this “natural selection” requirement.

The Fitness Proportional Selection (or Roulette Wheel) defines the probability of picking a given individual by its fitness proportion relative to the total fitness. In a similar way, the Rank-Based Selection also assigns selection probabilities, but this time based on fitness rank rather than raw value.

Tournament Selection, on the other hand, selects the best individual among a subset of the population taken randomly. This stratification ensures a balanced selection by preserving diversity while favoring stronger individuals without converging as much.

3.2 Crossover Operators

Two crossover operators were implemented, that combine elements from two parent solutions to generate new valid leagues. Both follow the same general logic: they take parts of each parent, avoid repeated players, and only return a child if the result respects all constraints. The difference lies in how they combine information — one works at the team level, and the other at the player level.

The ***team-based crossover*** creates children by selecting a random number of teams from one parent and filling the rest with teams from the other, as long as there are no overlapping players. If the child doesn’t reach the required number of teams, new ones are generated from the remaining player pool.

The ***position-based crossover*** mixes players across all teams, grouped by position. It builds a pool of players from both parents for each role and samples from that pool to build new teams. This approach introduces more diversity, since players are shuffled more freely and end up in different team contexts.

In both methods, the algorithm ensures that the resulting league respects all constraints. If a valid child can't be created on the first try, the process is repeated up to a certain number of attempts. This helps maintain a population made only of feasible solutions, which is important for keeping the search efficient and meaningful.

3.3 Mutation Operators

We implemented three mutation operators building them by scratch. These operators are used to introduce variation into the population by making changes inside each individual (League) while keeping it valid.

The ***mutation_swap_players*** performs a swap per team in each league (individual), ie. for each team attempts to swap one player with the same position of a player from another team.

The ***mutation_regenerate_team*** removes all players from a randomly chosen team (let's call it team x), then removes 7 position-respecting players from the other four teams and fills team x with those 7 players. Finally, redistributes the players removed from team x back into the other 4 teams.

The ***mutation_balance_teams*** attempts to swap one player between the strongest and the weakest teams (teams with the highest and lowest average skill, respectively) in order to reduce the standard deviation of average skill within an individual.

4. Performance Analysis

To evaluate the performance of different configurations we tested every different combination of selection, crossover, and mutation operators. *(Table 1)* summarizes the results across multiple runs, showing the mean and best fitness, standard deviation, and execution time.

After testing the eighteen possible combinations in runs of three per combination and for fifty generations each, we obtained our best fitness (approximately 0.05714) in four different combinations of operators.

In these top-four best fitness results the mutation operator is the same (*mutation_balance_teams*) indicating an almost perfectly balanced league. However, these configurations also took significantly longer to run, except when combined with *position_crossover* instead of *team_crossover*.

Overall, we concluded that the fourth combination (which is composed of the following operators: *sel_roulette*; *position_crossover*; *mutation_balance_teams*) is the best one, since it achieved the lowest standard deviation in average skill (approximately 0.05714) in less time (approximately 21.51 seconds), an excellent trade-off between score and computational cost.

5. Justification of Decisions

We decided to represent an individual (or a League) as a list of teams, this representation allowed us to use the operators in an intuitive way and easy to test and validate, by

analyzing the standard deviation of the average skill of each team compared to the league mean.

The fitness function we implemented computes the standard deviation of the average skill of the teams in a league and the goal is to minimize it, meaning that the league is well balanced in terms of the average skill per team.

6. Conclusion

This project showed that Genetic Algorithms can be effectively used to build balanced sports leagues while respecting complex constraints like team structure, salary limits, and unique player assignments. By designing our own mutation and crossover operators, we made sure that the algorithm always worked with valid solutions and stayed focused on what matters most: team balance.

The best solutions we found reached a fitness of 0.0571, which means the teams were extremely close in terms of average skill. We can't say for sure that this is the absolute best possible solution — Genetic Algorithms don't guarantee that — but based on the results and how consistent they were, we believe the algorithm got very close to an optimal outcome.

The overall results reflect the effectiveness of the approach and the quality of the operators implemented. The algorithm consistently produced strong, valid solutions and successfully met the main objective of creating a well-balanced league under all given constraints.

Appendix

Table 1 - Performance of All Genetic Algorithm Configurations

	selection	crossover	mutation	mean_fitness	std_fitness	best_fitness	time_sec
1	sel_rank	team_crossover	mutation_balance_teams	0.05714285714285552	0.0	0.05714285714285552	379.85
2	sel_tournament	team_crossover	mutation_balance_teams	0.05714285714285552	0.0	0.05714285714285552	406.03
3	sel_roulette	team_crossover	mutation_balance_teams	0.05714285714285552	0.0	0.05714285714285552	417.12
4	sel_roulette	position_crossover	mutation_balance_teams	0.05714285714285552	0.0	0.05714285714285552	21.51
5	sel_rank	position_crossover	mutation_balance_teams	0.07373007035022593	0.023457861879837342	0.05714285714285552	10.81
6	sel_tournament	position_crossover	mutation_balance_teams	0.08475218557682003	0.03904548671932574	0.05714285714285552	13.82
7	sel_rank	team_crossover	utation_regenerate_team	0.09031728355759631	0.023457861879837342	0.05714285714285552	2.41
8	sel_roulette	team_crossover	mutation_swap_players	0.11792661199156083	0.015587624839488393	0.10690449676496672	1.76
9	sel_rank	position_crossover	utation_regenerate_team	0.11792661199156257	0.015587624839490855	0.10690449676496672	9.21
10	sel_roulette	team_crossover	utation_regenerate_team	0.11792661199156257	0.015587624839490855	0.10690449676496672	1.93
11	sel_rank	team_crossover	mutation_swap_players	0.12680253969898056	0.0281400821819645	0.10690449676496672	2.43
12	sel_roulette	position_crossover	utation_regenerate_team	0.1378246549255764	0.024417232221108195	0.10690449676496672	6.47
13	sel_tournament	team_crossover	mutation_swap_players	0.1454655852671522	0.033951269091929484	0.10690449676496672	2.78
14	sel_tournament	position_crossover	utation_regenerate_team	0.14670058263299438	0.0281400821819645	0.10690449676496672	7.95
15	sel_rank	position_crossover	mutation_swap_players	0.1543415129745702	0.03482400003104628	0.10690449676496672	8.1
16	sel_tournament	position_crossover	mutation_swap_players	0.1543415129745702	0.03482400003104628	0.10690449676496672	7.89
17	sel_roulette	position_crossover	mutation_swap_players	0.17300455854274185	0.023358364660678217	0.13997084244475427	6.61
18	sel_tournament	team_crossover	utation_regenerate_team	0.17560567469974156	0.04857906879318291	0.10690449676496672	3.02

Table 2 -Final League Configuration – Best GA Solution

Team 1	**Team 2**	**Team 3**	**Team 4**	**Team 5**
Avg Skill: 86.43	Avg Skill: 86.43	Avg Skill: 86.43	Avg Skill: 86.29	Avg Skill: 86.43
Total Salary: €670M	Total Salary: €707M	Total Salary: €685M	Total Salary: €675M	Total Salary: €687M
GK: Jordan Smith | Skill: 88 | Salary: €100M	**GK: Blake Henderson** | Skill: 87 | Salary: €95M	**GK: Ryan Mitchell** | Skill: 83 | Salary: €85M	**GK: Alex Carter** | Skill: 85 | Salary: €90M	**GK: Chris Thompson** | Skill: 80 | Salary: €80M
DEF: Owen Parker | Skill: 88 | Salary: €100M	**DEF: Ethan Howard** | Skill: 80 | Salary: €70M	**DEF: Jaxon Griffin** | Skill: 79 | Salary: €65M	**DEF: Brayden Hughes** | Skill: 87 | Salary: €100M	**DEF: Daniel Foster** | Skill: 90 | Salary: €110M
DEF: Lucas Bennett | Skill: 85 | Salary: €90M	**DEF: Maxwell Flores** | Skill: 81 | Salary: €72M	**DEF: Caleb Fisher** | Skill: 84 | Salary: €85M	**DEF: Logan Brooks** | Skill: 86 | Salary: €95M	**DEF: Mason Reed** | Skill: 82 | Salary: €75M
MID: Austin Torres | Skill: 82 | Salary: €80M	**MID: Dominic Bell** | Skill: 86 | Salary: €95M	**MID: Ashton Phillips** | Skill: 90 | Salary: €110M	**MID: Gavin Richardson** | Skill: 87 | Salary: €95M	**MID: Spencer Ward** | Skill: 84 | Salary: €85M
MID: Dylan Morgan | Skill: 91 | Salary: €115M	**MID: Hunter Cooper** | Skill: 83 | Salary: €85M	**MID: Bentley Rivera** | Skill: 88 | Salary: €100M	**MID: Connor Hayes** | Skill: 89 | Salary: €105M	**MID: Nathan Wright** | Skill: 92 | Salary: €120M
FWD: Chase Murphy | Skill: 86 | Salary: €95M	**FWD: Sebastian Perry** | Skill: 95 | Salary: €150M	**FWD: Julian Scott** | Skill: 92 | Salary: €130M	**FWD: Tyler Jenkins** | Skill: 80 | Salary: €70M	**FWD: Zachary Nelson** | Skill: 86 | Salary: €92M
FWD: Adrian Collins | Skill: 85 | Salary: €90M	**FWD: Elijah Sanders** | Skill: 93 | Salary: €140M	**FWD: Landon Powell**| Skill: 89 | Salary: €110M	**FWD: Xavier Bryant**| Skill: 90 | Salary: €120M	**FWD: Colton Gray**| Skill: 91 | Salary: €125M