SPORTS TEAM FORMATION USING GENETIC ALGORITHM

Abstract

This study explores the application of a genetic algorithm (GA) for optimizing soccer team formations based on player skills and positions. This project aims to remove subjective bias from the process and provide a fair and performance-based approach to forming the world team of the year. In addition, the GA's performance in this project was analysed via systematic experimentation with varying population sizes and mutation rates. Moreover, the study experimented with different crossover and mutation strategies. Finally, this study compared the experiments and discussed the observations.

1. Introduction

This report is for the project which is for finding the best world players' team formation in soccer by implementing a Genetic Algorithm (GA). This Algorithm is an adaptive heuristic search algorithm that forms the majority of evaluation algorithms. This algorithm can help to solve the complex problem by iteratively improving a population of viable solutions similar to the evolutionary process (AKBASLI, 2022).

As Kanade (2023) said, GA algorithms can be used in different fields such as Optimization problems, Combinatorial optimisation, Machine learning, Evolutionary robotics, Financial modelling, image processing, etc.

Each player in soccer sport has attributes like speed, strength, accuracy, and teamwork. In addition, every player plays in a different position in their team formation. Thus, the objective of this project is to find the best players based on their skills and position in their team formation.

Every year, the best player and the best formation of the team are chosen by humans in FIFA, and it can be said that emotions are involved in this selection. Hence, every year when the best soccer player is chosen by FIFA, then it brings many protests from people. Therefore, by doing this project, trying to solve this problem and find the best team line-up with fair.

2. Literature Review

I searched for this problem and checked how others tried to solve this type of problem before starting the project. This section of the report discusses some of the papers studied for this project.

One of the papers is "New Mathematical Models for Team Formation of Sports Clubs Before the Match," (Budak et al., 2017). This paper focuses on developing mathematical models to assist in team formation for volleyball teams. Budak et al addressed the limitations of the previous models and then suggested two new models. These models complete the coach's ideas into the decision-making process to a novel approach in the field. In addition, this research shows the application of these models using a real-life case study of a volleyball team and explains their practical applicability.

Budak et al., (2018), focused on creating the best team composition in sports clubs, particularly volleyball by maximising team harmony through a mathematical model. Their methodology is linear programming based on players' preferences and coach strategies. Additionally, their finding emphasises the psychological aspects of players and the integration of coaching strategies in team formation.

Finally, the last paper studied before starting the project was "Team Formation: Matching Quality Supply and Quality Demand" (Boon and Sierksma, 2003). This paper mostly focuses on optimizing soccer and volleyball team formations using the Computer Coach system. In addition, the methodology which they used is linear programming by using player-score and position-score tables to match players to positions. And finally, their finding is that they described the approach to team formation by considering technical and tactical aspects, and the complexities of player rotations in volleyball.

3. Method

As mentioned, for this project wanted to solve this problem by using the GA. First tried to implement the algorithm very basic and simply to make sure the algorithm would work fine. For example, in the first experiment, to make it simple, I considered a small team of 3 players without any position. In other words, it was showing the best three players out of eleven players. It means that created a GA which worked with the small size of data.

After understanding the algorithm is working with the data of players without position, then added position and label for each player. In the beginning, the player data was in a list of Python, but because wanted to add the position and label (name), then I changed the list to a dictionary.

The final GA takes into consideration individual player skills and the positional requirements of a standard soccer team. In addition, the GA evolves a population of potential team formations over a series of generations. Additionally, this GA aims to maximise the collective skill level while adhering to formation constraints.

3.1. Components of The Algorithm

The GA which developed for this project is contains the following components and functions:

Data: as mentioned. the hypothetical data used in this algorithm contains the skills of each player as well as their position. The skills of players are modelled as a tuple of integers, reflecting different aspects of their abilities, such as speed, strength, and accuracy.in addition, the positions of players include forward, midfielder, defender, and goalkeeper.

Team formation: this component is predefined as the dictionary for team formation which algorithm follows it. the team formation in this GA is defined as: {'goalkeeper': 1, 'defender': 4, 'midfielder': 3, 'forward': 3}. Thus, this ensures that each team consists of eleven players, including one goalkeeper, four defenders, three midfielders, and three forwards.

Fitness function: This function is for calculating the sum of the skills of all players in the team. A team's fitness is considered only if it meets the positional requirements; otherwise, its fitness is set to zero. This approach incentivizes the formation of balanced teams.

Population Initialisation function: this function is for creating a specified number of teams or population size, each randomly assembled while respecting the team formation rules.

Selection function: this function is for sorting the current population based on fitness scores and selects the top half. Also, this simulates a survival-of-the-fittest scenario where only the best-performing teams are chosen for breeding.

Crossover function: this function is for generating a child team by combining players from the two parent teams. Then, for each position required by the team formation, and then, this function randomly selects players from one of the parents, and finally, it ensures adherence to positional requirements.

Mutation function: this function is for introducing variability by a predefined mutation rate. Additionally, it randomly alters a player in a randomly selected position with another player not currently in the team, and then, it maintains the team's positional structure.

Parameters: this algorithm contains some parameters as well. For example, the **population size** parameter which is for the number of teams in each generation, the **number of generations** parameter, which is the number of iterations the GA runs, and finally, the **mutation rate** parameter which is for the probability of mutation per team.

4. Result

This section of the report indicates some of the experiments taken for this project as well as shows the results.

4.1. Experiment with The Small Size of Data

As said in the method section, first tried to implement the algorithm for the simple scenario. That is why the data of players in this experiment are without the position and just considered a small team of 3 players. The table below shows the hypothetical data of players which used in this experiment:

ID	Speed	Strength	Accuracy	Teamwork	Stamina
1	80	85	75	90	70
2	78	80	85	88	75
3	82	78	80	85	80
4	75	88	82	80	77
5	81	86	76	90	76
6	82	87	77	90	72
7	84	89	79	90	74

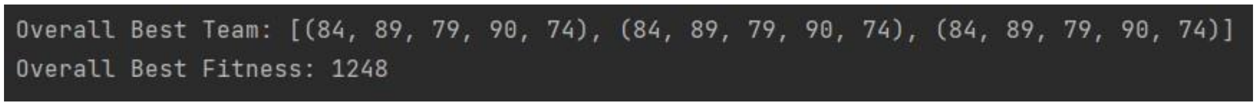
Table 1: Data of players' skills (min + 1, max = 100)

Additionally, the table below shows the values of parameters used in this experiment:

Parameters	Values
Team size	3
Population size	10
Number of generations	50
Mutation rate	0.1

Table 2: values of parameters used in Experiment 1

First, in this experiment, the result was not the same as expected. As shown below, the output is only three player skills, but it is just one player which is repeated three times.

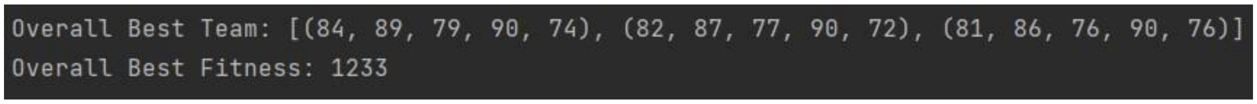


```
Overall Best Team: [(84, 89, 79, 90, 74), (84, 89, 79, 90, 74), (84, 89, 79, 90, 74)]
Overall Best Fitness: 1248
```

Figure 1: Output for Experiment 1

To fix this issue, then I changed the implementation of the crossover and mutation function in a way that makes it diverse. For example, I added the condition to prevent the same player from being included multiple times in a team during the crossover and mutation.

Then as represented in Figure 2, the outcome is three players with the highest skills.



```
Overall Best Team: [(84, 89, 79, 90, 74), (82, 87, 77, 90, 72), (81, 86, 76, 90, 76)]
Overall Best Fitness: 1233
```

Figure 2: the output of experiment 1 after fixing the issue

4.2. Experiment with the position

In this experiment added the position for players to make more real. Then after adding positions to players, I needed to change every function to work based on team formation. After a lot of changes and testing, then the functions work the same as what is explained in the method section.

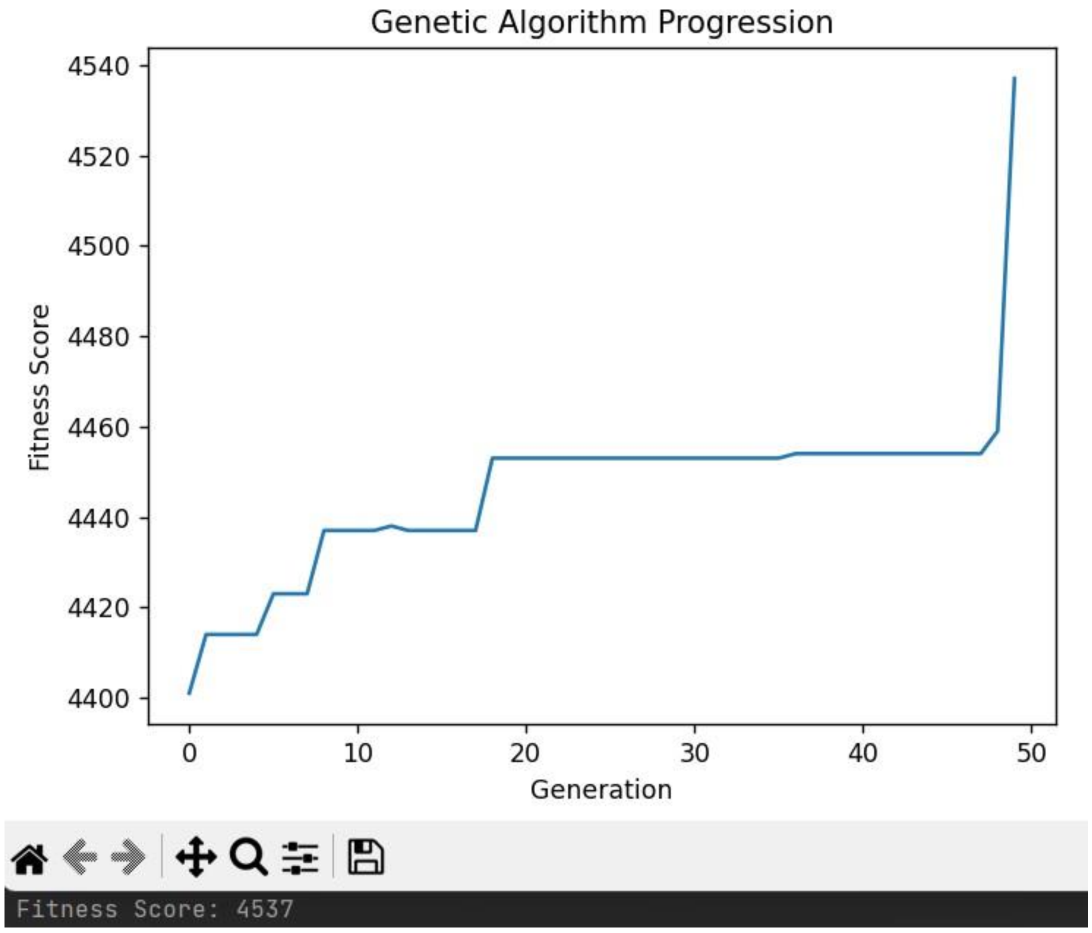
4.3. Experiments with Different Values of Population and Mutation Rate Parameter

After adding the position to the data, then experimented with the following values of the parameters:

Parameters	**Values**
Population size	10
Number of generations	50
Mutation rate	0.1

Table 3: values of parameters of the Experiment 3

Thus, the figure below shows the result with values of table 3:



```
Fitness Score: 4537
```

Figure 3: Result of the experiment 3

As indicated in Figure 3, the fitness score is equal to 4537 for a population size equal to 10, and the mutation rate is equal to 0.1. However, every time running with these values, the fitness score changes, and it is usually between 4530 and 4560. In other words, can say that the fitness score shows gradual improvement over generations. Also, the variability in the fitness score from run to run indicates that the GA is exploring a diverse set of solutions within a smaller population. Therefore, can conclude that the smaller population size may limit the search space but allow for quicker convergence to a solution.

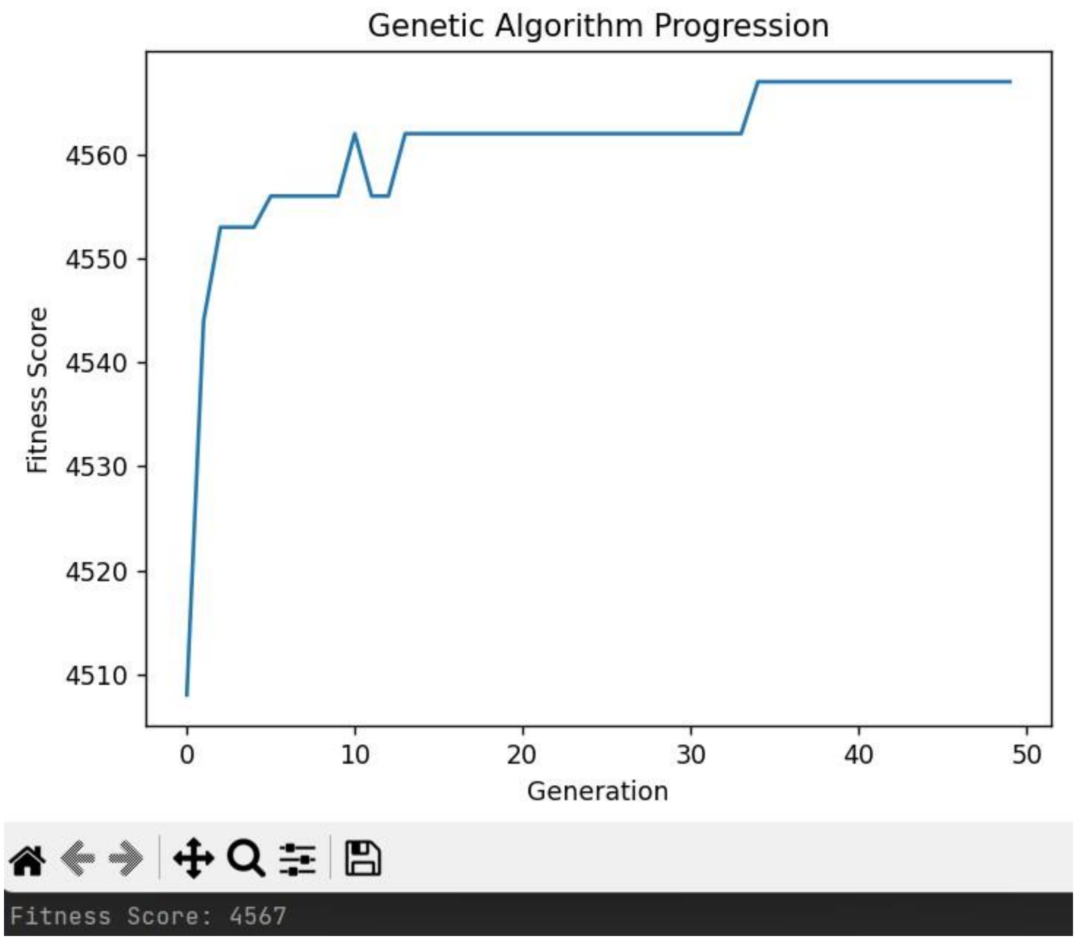
In the next experiment tried the algorithm with population size equal to 50 and mutation rate equal to 0.2.

The table below indicates the values in this experiment.

Parameters	**Values**
Population size	50
Number of generations	50
Mutation rate	0.2

Table 4: the parameters' values in Experiment 4

Therefore, the figure 4 represents the result for the experiment 4:



```
Fitness Score: 4567
```

Figure 4: the output for experiment 4

As shown in Figure 4, the fitness score is 4567 for the population size 50 and a mutation rate of 0.2. in addition, many times tested with these values the fitness score did not change and always it was equal to 4567. Hence, can say that the higher mutation rate and larger population size likely contribute to a thorough exploration of the search space, Also, resulting in a higher fitness score. However, this stability might also suggest that the algorithm is consistently finding a local maximum due to the higher mutation rate disrupting beneficial combinations of genes.

4.4. Experiment with Uniform Crossover and Swap Mutation

For this GA tried different types of crossover and mutation. This section explains how this GA works with uniform crossover and swap mutation. In the uniform crossover, each player in the child team is chosen independently from one of the two parents with equal probability. Hence, this crossover can introduce more diversity.

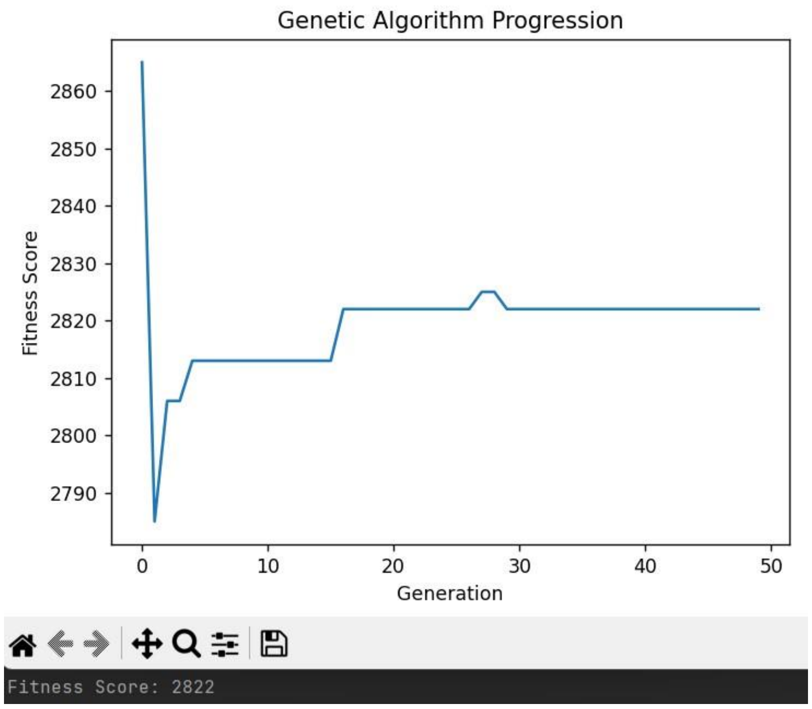
In swap mutation, instead of replacing a player in a certain position, then need to swap two players within the team.

The parameter values used for this experiment are indicated in the following table:

Parameters	**Values**
Population size	10
Number of generations	50
Mutation rate	0.1

Table 5: values of parameters in Experiment 5

the figure below is the result of when the uniform crossover and swap mutation.



```
Fitness Score: 2822
```

Figure 5: fitness score in uniform crossover and swap mutation

It is obvious from Figure 5 that the fitness score is generally lower, with the final value reaching only 2822. Moreover, the progression fluctuates more, with a sharp drop at the beginning and then plateaus with minor improvements. Thus, the overall fitness score is much lower than with the previous crossover and mutation.

5. Discussion

This section of the report discusses and compares the experiments explained in the previous section of this project.

5.1. Comparing Values of Population and Mutation Rate Parameters

As shown in Experiment 3 and Experiment 4, understood that the higher value of population size and mutation rate then will achieve a higher fitness score compared to lower values of these parameters. Therefore, can conclude that the larger population size and higher mutation rate are more effective for this problem.

Another observation in this experiment is that Experiment 4 (with higher values) shows less variability in the fitness score across multiple runs compared to Experiment 3 (with lower values). In my opinion, this could imply that Experiment 4's parameter settings led to a more consistent convergence towards an optimal or near-optimal solution.

Another conclusion which can be made via this experiment is about the impact of the mutation rate. Probably higher mutation rate could provide a more extensive search of the solution space as well as prevent premature convergence and potentially escaping local optima.

In addition, for the impact of population size can say that the GA with a higher population size can maintain a greater diversity within the gene pool, which can be beneficial for finding better solutions.

In summary, the Experiment with a larger population and a higher mutation rate seems to provide a more stable and potentially more optimal solution than the Experiment with a smaller population and lower mutation rate.

5.2. Comparing with Different Crossover and Mutation

As observed in the experiment, the first crossover and mutation seem to achieve a higher fitness score compared to the uniform crossover and swap mutation. Hence, can say this shows that the first crossover and mutation are better at finding a more optimal solution in the given problem space.

Another conclusion for this experiment is that the first crossover and mutation show a more consistent improvement over generations. However, the uniform crossover and swap mutation results in a more erratic progression with less overall improvement.

Additionally, this experiment indicates that the first crossover and mutation methods may be more efficient for this specific problem in this project because they achieve higher fitness scores. Moreover, this could be because of the better exploitation of good genes when they appear in the population.

One thing that needs to be mentioned here is that It is important to be aware that genetic algorithms are highly dependent on the specifics of the problem being solved. Therefore, it is possible that the uniform crossover and swap mutation could perform better in a different problem or with a different fitness landscape.

In summary, by observing this experiment can understand that the first crossover and mutation methods seem to outperform the uniform crossover and swap mutation in this specific example.

6. Future Work or Improvement

Because of the limited time in this project, then it was not possible to experiment with everything in this GA. One thing we can do in the future is to use the data of the real players and check how this GA will work with that.

Another work for the future is to implement multi-objective optimisation. The current GA focuses on maximizing the collective skill levels while adhering to team formation constraints. Additionally, future iterations could consider multi-objective optimization to include additional criteria such as the number of goals and assists for each player.

This GA could be also adapted to other team sports that require strategic formation and player selection. Hence, future work will investigate the transferability of the algorithm to sports such as basketball, volleyball or American football.

7. References

Akbasli, I.T. (2022). *Tutorial of Genetic Algorithm*. [online] kaggle.com. Available at: https://www.kaggle.com/code/zzettrkalpakbal/tutorial-of-genetic-algorithm [Accessed 19 Jan. 2024].

Boon, B.H. and Sierksma, G. (2003). Team formation: Matching quality supply and quality demand. *European Journal of Operational Research*, 148(2), pp.277–292. doi:https://doi.org/10.1016/s0377-2217(02)00684-7.

Budak, G., Kara, İ., İç, Y.T. and Kasımbeyli, R. (2017). New mathematical models for team formation of sports clubs before the match. *Central European Journal of Operations Research*, 27(1), pp.93–109. doi:https://doi.org/10.1007/s10100-017-0491-x.

Budak, G., Kara, İ., Tansel İç, Y. and Kasımbeyli, R. (2018). Optimization of Harmony in Team Formation Problem for Sports Clubs: A Real-Life Volleyball Team Application. *International Journal of Applied Science and Technology*, 8(2). doi:https://doi.org/10.30845/ijast.v8n2a2.

Kanade, V. (2023). *Genetic Algorithms - Meaning, Working, and Applications*. [online] Spiceworks. Available at: https://www.spiceworks.com/tech/artificial-intelligence/articles/whatare-genetic-algorithms/.