Football: A Linear Programming Approach

ASHUTOSH SHARMA - 190100027, AYUSH SINGH - 190100032, KARSHIK MADAVI - 190100070, SATDHRUTI PAUL - 190100105
DEPARTMENT OF MECHANICAL ENGINEERING, IITB

1 Abstract

Our project is based on a game-play of a football game on a 15x15 grid with 3+3 players wherein given particular parameters underlying the game, we figure out the best move for every player in the 2 teams using linear programming. The optimum move is decided at every iteration wherein each iteration corresponds to every player making one move out of the following - dribble,pass or shoot. So essentially we try to optimize the moves given the current player positions and ball position. We consider two major parameters for our model - first is safety of any point on the grid which takes into account enemy team and self team distances; and second is distance from the opponent goal post. For most part of the analysis, we looked at 3v3 scenarios and at later stages, we tried increasing the number players to understand it's effect on the game-play. We used Pulp - a python library for solving the linear programming part and other visualization libraries of python were also used for displaying the game-play.

2 Challenges faced

- The major challenges on the onset of the project were related to the basic formulation of the problem and various forms of constraints to be included in the problem. Formulating the distance and safety functions for optimum game-play was also challenging as our score function is a proxy for higher number of goals secured for any team and various problems could occur during the game-play with a wrong score function like - lower incentive for reducing goalpost distance can make the game-play stretching on for longer duration and even getting stuck in the middle of the field; higher penalty in safety function for grid points near opponents can make players moving away from the goalpost even with few opponents defending the goal, leading to sub-optimal actions.

- At later stages of the project, another challenge we faced was how to include uncertainty in the game given that the whole problem had been modelled and solved in a deterministic sense till that point. Opponent intercept code was modified to include uncertainty, by taking the probability of a successful intercept during a shoot/pass or dribble to be 0.3 for every opponent player lying near the trajectory of the ball.

- Another problem that we faced was that players used to shoot to the goalpost from really large distances when the trajectory scores for other moves were very low. So as to solve this problem, we modified the trajectory scores of such passes and shoots to $-\infty$ so as to avoid such a move. Another problem that occurred was that many times passes kept on happening between same 2 players while other opponent team players kept on advancing and finally surrounded them. This happened because rather than moving forward by dribbling (to a position close to opponent players), the team players had higher scores for passing to team mates. In order to mitigate this issue, we limited consecutive passes

made between same players to 2 by maintaining another variable which was updated at every iteration.

Some other notable challenges that we faced during the course of the project include the "clustering problem" and "consecutive passes", which have been discussed in the later sections.

3 Functions and Implementation Details

3.1 Distance

Distance is a simple function which will be utilised to calculate distance between two players.

$$dist(p_i, p_j) = \sqrt{(p_i.x - p_j.x)^2 + (p_i.y - p_j.y)^2}$$

3.2 Safety

This function is used to mark how safe a point is on the grid which is a measure of current team players' influence versus the opponent team players' influence. The 'reach' of a player is what we define as influence in this context and is taken to be a decaying function radially.

$$S(P(i,j)) = \Sigma_{i=1}^{n_{te}} \frac{k}{1 + k * dist(P, p_i)} - \Sigma_{i=1}^{n_{op}} \frac{k}{1 + k * dist(P, op_i)}$$

Where k = influence factor, P = Point (i, j) on grid, n_{te} = number of current team players, n_{op} = number of opponent team players

This function was chosen because we wanted scores of grid points away from opponents higher and away from team players to be lower. 1 was added in the denominator to avoid the denominator becoming zero and k was kept in the formula to control how fast function decreases away from opponents/team players. The first term had later been replaced with a constant factor to decrease clustering (as awarding grid points positively for positions near team players promoted clustering of team players and led to easier interception by opponent players).

3.3 Trajectory Score

Now that we have all the essential parameters defined, we can calculate the trajectory of any given path with any pair of starting and ending point. Initially this was taken to be the minimum of scores of every point along a path, then it was later taken to be an average along the path.

$$score(P) = S(P) \times \frac{k}{1 + k * dist(P, goalpost)}$$

$$traj(p_0, p_f) = min(\{score(p_i) \forall p_i \in line(p_0, p_f)\}) \rightarrow \frac{\Sigma_{P=p_0}^{p_f} score(P)}{n_{points}}$$

Where $\rightarrow$ implies 'later converted to'.

3.4 Dribble

The score of any surrounding point to which a player can dribble to would be an average of the scores of passing to the others and shooting to the goalpost from the new position. Thus every possible dribble position has a score associated with it, which is returned along with information regarding how many possible positions are there for any player .

$$Dr(P(i,j)) = \frac{\Sigma_{k=1}^{n_{te}-1} traj(P, p_k) + traj(P, goalpost)}{(n_{te}-1)+1}$$

3.5 Intercepting and Uncertainty

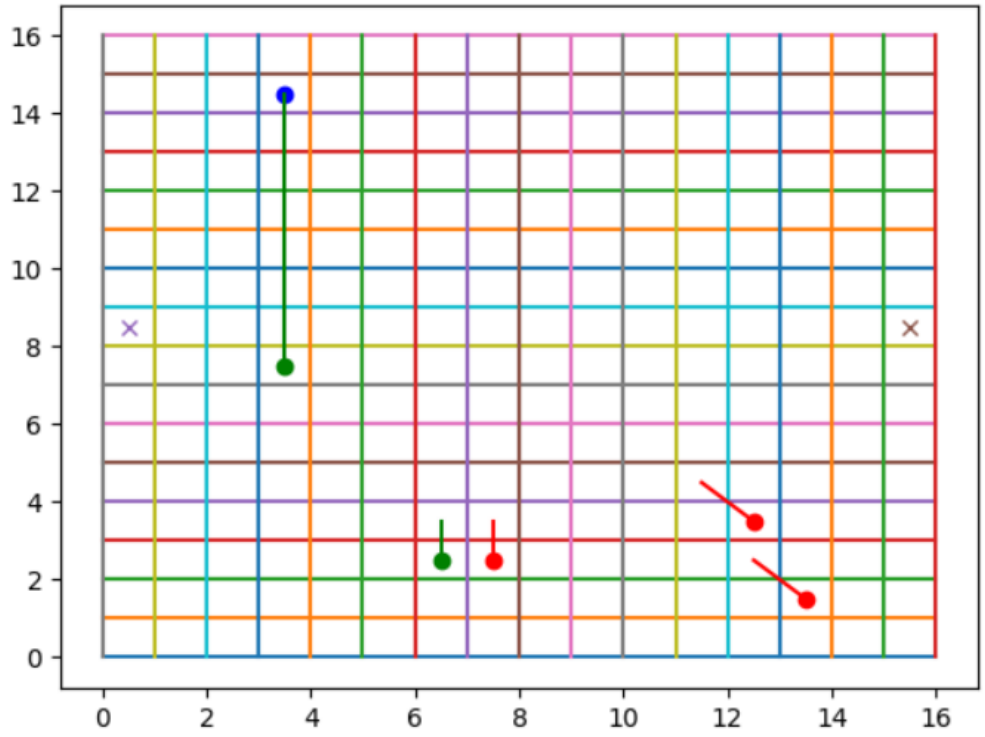
In order to include uncertainty in this deterministic decision making, we included another parameter, the probability of a successful intercept by the opponent team. In our implementation, we consider the opponents closer than 0.5 (each grid point being a distance of 1 apart along x and y axis) to the line of pass/shoot as the opponent players who can intercept the ball. Then each player is assigned the same probability of intercepting the ball and iterating through these opponent players, whomsoever successfully intercepts the ball first gets the ball for the next iteration. For intercepts during a dribble, we consider only those opponents who can reach the previous or new position of the player with the ball, and same uncertainty of intercept has been implemented for these opponents.

Another way to introduce uncertainty which we plan on including in the future is intercept at goal. As of now we don't have a dedicated goalkeeper. We can keep a goalkeeper restricted to the goalpost grid point and we can assign some probability of intercepting the ball within a radius r around the goalpost and once the ball is intercepted, goalkeeper will shoot to the team player with highest score function.

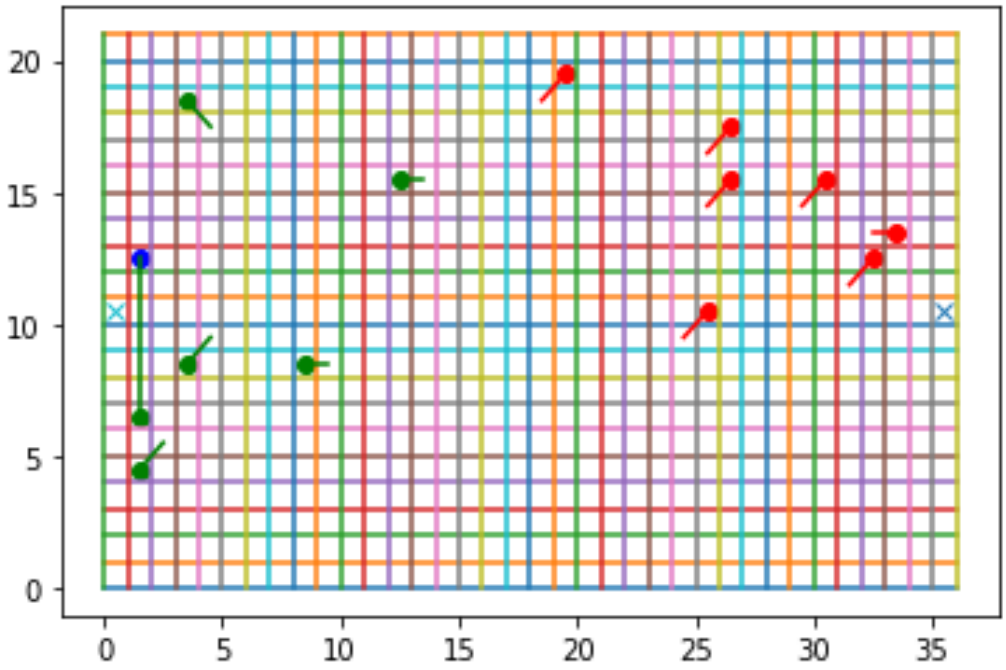
3.6 Opponent moves

We are optimizing for the best move for the team currently having the ball. For opponent move, we first calculate the safety for the opponent team, which is calculated similarly except that the opponent team (op) and team (te) get reversed in the function calculating safety (higher safety for grid points near the players of the team having the ball). The score is simply safety now, instead of multiplication of safety and distance functions because when the team does not possess the ball as their priority should be to get the ball, and not scoring a goal post shoot. Since the actions for the opponent team is limited to dribble only, we simply extract the maximum dribble score from the possible dribble positions for every opponent team player using max function in python as we already have the arrays for the dribble scores for every opponent team player.

4 Animations



Animation 1: 6 Players (3+3) playing in 15x15 grid (green team wins)



Animation 2: 14 Players (7+7) playing in 20x35 grid (red team wins)

Note: The animations are only supported by some pdf readers like AcrobatReader, PDF-XChange, acroread, and Foxit Reader. If not visible, kindly refer to this link for the animations.

5 Constraints and Objective Function

5.1 Constraints

Initially the constraints were designed such that when player with the ball is dribbling, every team player will dribble, and if he passes or shoots to the goal, other team players are stationary. Later on the constraints were modified to give freedom to dribble for the other players when player with the ball is dribbling and the final constraints for a game of 3v3 is as follows-

$$x_a + x_b + x_g + \sum_{i=1}^{8} x_i = 1$$

$$x_a + \sum_{i=9}^{16} x_i = 1$$

$$x_b + \sum_{i=17}^{24} x_i = 1$$

where a,b,c are the team players having the ball (for whom we are optimizing the next move) and x_a,x_b are the binary variables representing pass to players a,b (player c currently has the ball). x_g variable represent shoot to the goal post. x_i binary variables represent dribble positions where $i = 1$ to 8 are for player c, next 8 for player a and the 8 variables after that for player b. The constraints are designed such that when the player with the ball dribbles or shoots to, so other players have no choice but to dribble, but if he passes to another player, let's say player b, then player b must not dribble and should remain stationary to receive the ball, and player c must dribble.

5.2 Objective Function

The objective function that we seek to maximize is as follows -

$$x_a s_a + x_b s_b + x_g s_g + \sum_{i=1}^{24} x_i d_i / 3$$

where s_a, s_b, s_g, d_i are the parameters which represent the trajectory/dribble score (described earlier) for the corresponding action. This objective function ensures that the action leading to highest score is chosen.

The objective function can be seen as the function maximizing the score of the action taken by the player. We take average of the dribble scores of all the players rather than just taking the dribble score for the player with the ball and it works because the dribble scores for any player for any possible dribble position is close to 1 and taking average helps in maximizing the dribbles for the other players as well. If we just take the dribble score of the player with the ball in the objective, the dribbles for other players will be sub-optimal.

5.3 Proposed modifications using Hashing

There is a loophole in the proposed constraints, i.e. there is no restriction for any two players to dribble to the same square on the grid. In order to tackle this we propose to add additional constraints such that whenever dribbling occurs, the final position's x and y coordinate of

any player should not match with another player's coordinates. We also introduce a hashing function $H(i, j)$ that take any coordinate and converts it into a unique (increasing along x and y, for dealing with clustering later on) hash so that we can use this to set constraints and since it is increasing function of x and y, value obtained after hashing will be unique for every coordinate. This technique will help in reducing the constraints by half as instead of checking 2 coordinates x and y for every player, we will be checking only one hash value (for a game of 3v3, the number of these additional constraints will be nominal, but will explode on increasing number of team players, hence motivation to include hashing).

$$H(i, j) = bin(i) << len(bin(j)) \oplus bin(j)$$

Here, $bin(i)$ means the binary representation of number i and the operator $\oplus$ means XOR of binary numbers. Hence we can convert a point P_i into a number $H(P_i) = \alpha_i$. (This hashing technique to resolve the above mentioned issue is not yet coded in the python script, and is part of the future work that we propose to implement)

$$|\Sigma_{k=1}^{p_i} x_k \times \alpha_k - \Sigma_{k=1}^{p_j} x_k \times \alpha_k| = |\beta| \geq 1$$

$$\beta + N \times y \geq 1; -\beta + N \times (1 - y) \geq 1$$

These extra constraints are to take care of dribble positions coinciding, p_i and p_j are the respective number of dribble positions available for player i and player j. α_i in each summation is the hashed value of the point in consideration for that corresponding player. N is a large number and y is a decision variable, that is either 0 or 1, and if it doesn't exist then point coincides.

6 Future scope of the Project

- As we can see in the video, there is a problem of clustering, where the intercept occurs repeatedly as the distance less than 1/2 units condition is satisfied frequently when all the team players are located close to one another and the game stagnates at continuous interception by opponent team. To take care of this we propose to introduce the concept of throwing the ball in a random area in space at a range of 6 units so that game proceeds rather than getting stuck.
- We also plan on including the concept of defenders wherein we can add constraint that a constant fraction of the number of team players need to stay behind half line to defend the goalposts from shoots directly into goalpost
- As described in previous sections, we also plan on including the concept of goalkeeper and hashing function for solving the problem of 2 players at the same grid point.

7 Conclusion

From our analysis and methodology we can see that it is indeed possible to simulate a football game using less than ten constraints and make it interactive as well. Various issues such as clustering, players getting to same positions, and various such issues required innovative solutions and even a simple looking project such as this required thorough understanding of the foundations of optimization and mathematical formulations. Introduction of uncertainty in intercepts

help in making the game-play different even for the same initial positions of all the team players.

In our model we are able to quantify the score of any given trajectory and this is updated every time players move around on the grid. Initially dribbles were coded to not occur during passing but this was modified by getting rid of condition variables and changing the constraints a little. By introducing the idea of hashing function we will be able to get rid of corner cases which had popped up. The model that we made using linear programming and knowledge of football differs from conventional methods of solving such problems like reinforcement learning which make use of historical data. In future, we may even compare such models to our model in order to improvise our own methodology of solving, and to simply observe how simple modeling methods using LP compare against sophisticated techniques such as RL.

Such kinds of hands on project are very helpful in getting to know how to implement various modeling techniques taught in class theoretically and such practical applications make the course even more exciting.

Link for the animations

A 15x15 grid with 3v3 player setup and d = 6.
A 20x35 grid with 7v7 player setup and d = 14.