An Evolutionary Approach to Optimal Robot Location in a Soccer Match

N. Porras, J. Prieto, V. Ortega, and A. Ramirez-Jaime

Faculty of Engineering
Universidad de La Sabana
Puente del Comun Campus, Km. 7, Autopista Norte de Bogota, Chia.
https://www.unisabana.edu.co

Abstract. This paper presents an evolutionary algorithm, based on game theory, to determine the optimal location of a set of NAO robots on a field of SPL Robotics. This algorithm is implemented on the B-Human platform and it is demonstrated (by means of the computation time) that this algorithm can work in real time during a match. Several examples of the optimal location of the robots at a time of the match are also presented.

Keywords: Game Theory · Genetic Algorithm · Optimal Location

1 Introduction

When studying systems with multiple decision-makers, the one of best tools for modeling this type of scenario is game theory. Game theory has been widely used in many fields of engineering, e.g., for control of urban drainage systems [12], for the design of smart energy networks [10], and for the study of sensor networks [14]. This is because game theory is very versatile in the sense that it can be used in static and dynamic environments [2], as well as in systems with few or many interacting agents [6]. However, for some applications finding a solution that satisfies all agents, i.e., the Nash equilibrium [9], may not be trivial and may spend many computing resources. For this reason, it is desirable to unite game theory with heuristic optimization algorithms [7] that can determine the solution of a game in real time without taking long.

Usually, when algorithms based on game theory are used, the fundamental objective is to determine the strategies that lead agents to an equilibrium in the cost functions that each of them is trying to optimize. These strategies can be the control signals for a set of valves in a drinking water system [1], or the position that a set of agents must occupy in a territory to ensure a rapid response to an external stimulus [5]. In the case of Robocup SPL, game theory can help determine the optimal trajectories (through the strategies of certain agents) for scoring a goal in a match between two teams.

In a robot football match, as in many multi-agent problems, knowing the optimal location of the set of entities in question is of great relevance as it can

help to take actions in very short times and help to be prepared for atypical scenarios that the programmer did not consider. For instance, an autonomous multi-agent ambulance system can be considered where optimal parking (considering location variables) can mean the difference between saving a patient's life at the time of an emergency [8]. However, optimizing the area of coverage by a robot is not the only benefit of finding an optimal position. When one knows an optimal location, an optimal rout to that specific point may be determined and thus it is possible to minimized energy consumption made by the robot. These consumptions become relevant when the scenario, e.g., a football match, begins to increase its duration, since the batteries adjusted within the agents may be limited (due to the rules of a game they may be playing), providing 90 minutes of charge in normal activity and 60 minutes in activities of effort, for example in [3] is guaranteed optimal consumption of energy resources arranged in a system of multiple agents only managing to find an optimal location within a structure.

This article proposes a game that allows to determine the optimal location of a set of robots, under a football game, considering several objectives that can be optimized simultaneously and considering several restrictions that concern a football match. The proposed algorithm allows to maximize the area of coverage at any moment of time taking into account that moving away from the ball in a match may not be desirable, as well as taking into account that having erratic positions on the field allows opponents to have an advantage when winning the match. The proposed scheme uses a combination of game theory and evolutionary algorithms that can be implemented under the operating system of an NAO robot and can run in real time satisfying all match restrictions. The major contribution of this paper is the novel implementation of an evolutionary algorithm on the standard B-Human platform that can be used in Robocup SPL matches, as well as in demonstration matches.

This paper is organized as follows: Section 2 presents the main optimization problem that is considered herein. Section 3 proposes a computational solution for determining the optima for the optimization problem shown in section 2. Section 4 presents results where the algorithms is show, and in section 5 the conclusions and future work is presented.

2 Problem Statement

In order to optimize the positioning of the robots on the field, it is proposed to solve an optimisation problem with multiple objectives. Each one of the robots will solve a local problem using the information it has available at a determined instant of time. Given that for this system there are multiple agents solving multiple optimization problems that are linked together it is possible to say that it can be modeled through game theory [9]. It is important to realize that the problem that each robot is solving is local nature because even though all robots know the information about all partners, their own decisions, i.e., where they will move optimally, only affect their own position[5].

Consider the representation in Fig. 1; the information that a robot has is represented by a vector $\boldsymbol{X}$ that contains the position of all the fellow robots and the position of the ball on the field. This figure shows the location of all allied robots (marked as white circles with white rectangles) and the ball (masked as a white circle with spiral) at some point during the match in order to construct the vector $\boldsymbol{X}$. Consider also that the location of the i-th ally is given by a point $p_i = [x_i, y_i]^\top$ and the location of the ball is given by $p_b = [x_b, y_b]^\top$. In that case the vector $\boldsymbol{X}$ can be constructed as

$$\boldsymbol{X} = [x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_b, y_b]^\top \tag{1}$$

and every robot is able to measure it. This vector will now be refered as the State Vector (SV).

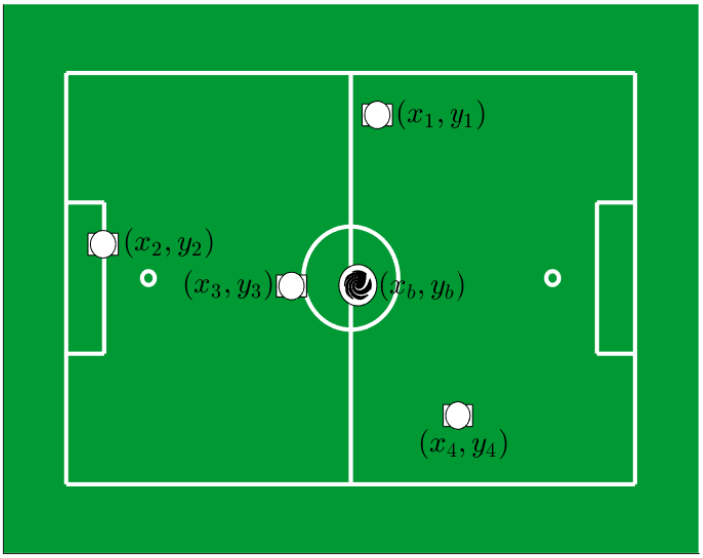

Fig. 1. Example of the information that a robot has at a certain instant of time. The white circles with white rectangles represent the allied robots and the circle with spiral represents the ball. For the implementation of this first algorithm is not necessary the location of the rivals so they are not shown.

The objective of the robots is to maximize the total team coverage area in order to be prepared in case of a rival attack or an unexpected movement of the ball. To do this, each robot seeks to minimize a cost function that depends on the position of all the allied robots interacting with the ball, i.e., all but

the goalkeeper, the current position of the ball and the distance to the center of the field. Therefore, the cost function that the i-th robot is minimizing is a multiobjective function with three objectives that will be described herein. The coverage area depends on the position of all the robots because their position on the field forms a polygon whose area is equal to the area covered by the team. However, if only the largest possible area is searched the robots would simply tend to the four corners of the field as this would generate the polygon with the largest area. For this reason, the position of the ball is included in the cost function to avoid large deviations to the corners of the field. Finally, a parameter is added that seeks to avoid large deviations from the center of the field as this ensures a uniform formation at all times.

Consider the optimization problem described in eq. 2. This is optimization problem that the i-th robot is solving and condenses the requirements described above where A represents the total area covered by the robots, d represents the distance from the robot to the ball and r represents its distance to the center. For this representation, α and β are weigth parameters that are ajusted to determined the relevance of each objective and X_{MAX} and Y_{MAX} are the maximum feasible areas given by the size of the field.

$$\begin{aligned} \underset{x_i,y_i}{\text{minimize}} \quad & -A(\boldsymbol{X}) + \alpha d(x_i, y_i) + \beta r(x_i, y_i) \\ \text{subject to} \quad & 0 \leq x_i \leq X_{MAX}, \\ & 0 \leq y_i \leq Y_{MAX}. \end{aligned} \tag{2}$$

For this optimization problem, A is described by Gauss' Area Formula (shown in eq. 3) [11], and d and r are described by the Euclidean distances squared to the required points (shown in eq. 4).

$$A(\boldsymbol{X}) = \frac{1}{2}\left|\sum_{k=1}^{n} \det\begin{pmatrix} x_k & x_{k+1} \\ y_k & y_{k+1} \end{pmatrix}\right| \tag{3}$$

$$\begin{aligned} d(x_i, y_i) &= (x_i - x_b)^2 + (y_i - y_b)^2 \\ r(x_i, y_i) &= (x_i - \frac{X_{MAX}}{2})^2 + (y_i - \frac{Y_{MAX}}{2})^2 \end{aligned} \tag{4}$$

Since each agent in the system is solving an optimization problem like the one presented in 2, the state vector will dynamically evolve to a balance in the cost functions of all robots. This equilibrium of the system is reached when

$$J_i(\boldsymbol{X}) \leq J_j(\boldsymbol{X}) \qquad \forall i, j \in \mathit{Robots} \tag{5}$$

and it's known as the Nash Equilibrium of the system[9]. It is important to point out that given that all the objectives in the multi-objective optimization problem are norms in nature, the problem described in 2 is a convex optimization problem that only has a global optima.

3 Evolutionary Optimal Locator

In order to solve the optimization problem proposed in the previous section it is possible to choose several paths. As already mentioned, this problem is convex in nature so one might think that a gradient based algorithm would be the best solution. However, the structure of this problem is dynamic because it strongly depends on being able to form a convex polygon using the positions of the robots and apply the Gauss formula. In addition, in a real implementation of the solution, a very high precision is not required given the uncertainty produced by the walk of the robots, e.g., a robot cannot go to a position with millimetric accuracy . For these reasons, it is proposed to implement a heuristic algorithm that allows to find an acceptable solution in a short time.

For this case, and thinking about an implementation on a NAO robot, it is proposed to implement a genetic algorithm [4] that allows to solve the optimization problem previously proposed in real time. This algorithm was implemented on the B-Human platform [13] that allows total control over the behavior of NAO robots and allows easy incorporation of modules into the usual game strategies. The detailed implementation of the algorithm (within the B-Human platform) is described below:

A module and a representation must be created in order to adapt with the B-Human standard. The GeneticLocator representation is powered by the GeneticLocatorProvider module, which consists of a method called optLocator and three attributes: optimalX, optimalY and timeSinceLastSeen. The module itself only assigns the attributes their same value, for example, the initial value that these attributes will have is zero, the module calls the same representation, and takes each value to assign it back to its respective attribute, so that until otherwise these representations are changed, this in order to save the most recent values obtained in the method.

The method has a single parameter, sent to be called from any state machine, this value is the current time, with this it is possible to limit the times by fraction of time in which the complete algorithm is executed. When called, the function takes values from the representations RobotPose, TeamData and TeamBallModel which are fundamental in the equation in charge of assigning a fitness value to each individual of the algorithm and also takes data from the GeneticLocator itself since it has the time of the last time the algorithm was executed. Within this unique method the values of representation, optimalX and optimalY are updated with the result of the algorithm and timeSinceLasTime with the argument sent from the status machine. All this as long as the condition of the time distance between the current time and the time stored in the representation as the last time the algorithm was executed is met.

However, this method not only updates the values but also returns a Pose2f object with the optimum position found by the algorithm, it always takes the values from the representation so that in case the condition has not been fulfilled and has not been executed again, it sends the information from the previous execution.

When this method of representation is called and the time that is sent as an argument fulfills the condition of the temporal difference between calls, the heuristic algorithm in charge of finding the optimal position will run as follows:

- First, defined as a constant, it generates a thousand (1000) random positions with whole numbers for the X axis from -450 to 450 to ensure a precision of the order of centimeters, but then this value is multiplied by 10 because in the BHuman platform the measurements are given in millimeters and for the Y axis with whole numbers from -300 to 300 under the same conditions as the X axis. The order of the axes is thus established as a standard on the BHuman platform. These positions are stored in a vector called curr_generation.
- For each position your fitness is evaluated. The function that determines this value takes four aspects into account:
  - Maximize the area covered by the NAOs participating in the algorithm. This requires the position of the individual generated and information from the TeamData representation.
  - Minimize the distance to the ball. To find this value you need data from the TeamBallModel representation and the position of the individual generated.
  - Maximize the distance to the corners. For this it is enough with simple calculations with the position of the generated individual.
  - Bring the NAOs closer to their own arch so that they are between the ball and it, thus achieving a defensive position. In the same way as the previous point, simple calculations between the position of the individual generated are enough.

For the second, third and fourth points, it is enough to calculate the distance to each target and according to the weight assigned to each entry, calculate how much it affects the total fitness. However, for the first point, since players can be forming any geometric figure on the field, finding the area of that figure cannot be determined through the formula of a particular figure, but needs to be found through a shoe formula that allows you to find the area only with the vertices, no matter what they are forming, however, the only condition is that they are ordered so that no straight line intersects with another so before calculating this area, it is necessary to first make sure that these vertices are ordered and this is done in the following way as follows before calculating this area:

- A vector is created with all the vertices.
- The vertex with the lowest value in X is searched, if there are two or more equal, the lowest Y is taken into account.
- This vertex is taken out of the vector and from now on it will be called point.
- From here it is necessary to find a subfitness for each remaining vertex that allows to find the correct order, this is done in the following way:
  - If the value of Y of the vertex is equal to the value of point, the fitness will be its value in X added to the maximum possible slope, in this case, 4500.

  - If the slope is negative, its absolute value is taken as a divisor of 20250000 and 1 is added.
  - If the slope is positive, its normal value is left.
- They are ordered according to their subfitness.
- The point value is added again.

With this vector of ordered vertices, you can find the area of any figure you are forming and you can calculate the fitness correctly for the individual generated.

- When all individuals have a fitness value, they are ranked from highest fitness to lowest.
- It takes the best 10% and goes straight to the new generation represented by a new vector called neo_generation.
- Of the best 50% random genomes are taken to make crossover which is carried out as follows:
  - Two individuals of the best resulting 50% are taken.
  - For each genome gene, a random number between 1 and 100 is generated called a mendel.
  - If mendel is between 1 and 40, the gene of the first genome is taken, if it is between 41 and 80, the gene of the second genome is taken, if it is between 81 and 100, it means that it mutated and a random number is generated from 0 to 9 and this passes as the gene of the new individual.
- Crossovers are made until neo_generation has 1000 individuals.
- The new vector becomes the current vector so curr_generation equals neo_generation.
- The process begins again until the 500 generations are reached.

When all generations have passed, one would expect the individual with the optimal position to be in the first place of the last generation, so from that position the X is taken, assigned to optimalX and in the same way with Y to optimalY. Finally the object Pose2f is generated with the new information and sent to the state machine that called it.

4 Results

The objective of the robots is to maximize the total team coverage area in order to be prepared in case of a rival attack or an unexpected movement of the ball. To do this, each robot seeks to minimize a cost function that depends on the position of all the allied robots that are able to interact with the ball, i.e., all but the goalkeeper, the current position of the ball and the distance to the center of the field. Fig. 2 shows an example of a set of trajectories based on the proposed strategy. In this example it can be seen how the players look for an optimal position around the ball following the trajectories represented by discontinuous lines.

As expected, the optimal location that seeks to generate the largest possible area is a square. Logically, if the ball were in a position for which a square is

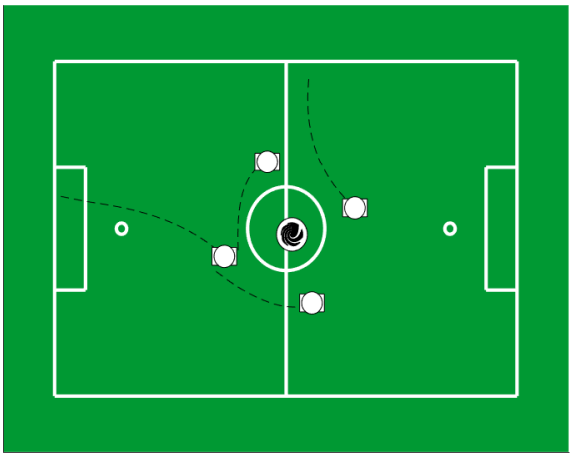

Fig. 2. Optimal location of a group of 4 robots based on the proposed strategy. In this figure, the dashed line represents the trajectory that the robots followed in order to achieve the optimal location. It is important to point out that the optimal polygon for this particular ball position is a square as it is expected.

not a feasible polygon, e.g., if it were very close to the out line, the polygon could change to maximize the given criterion. Fig. 3 shows an example in which the ball is not centered around the optimal polygon. This is the optimal solution for the constrained optimization problem but would be suboptimal for the unconstrained one.

Since this algorithm must be implemented in real time in a game of robots, the time required to determine the optimal solution is a relevant parameter and must be studied in detail. For the implementation in C++ on the B-Human platform, the average time that a robot uses to solve the genetic algorithm over all its generations oscillates around 50ms. This indicates that it is feasible to use it in a match against real rivals, rather than in a simple simulation.

5 Concluding Remarks and Future Work

This paper has presented an algorithm that can determine the optimal location of a set of NAO robots on a playing field and be used in a game of Robocup SPL. A detailed step by step has been shown, as well as the final location that these robots adopt on the field. In addition, it has been seen that this algorithm can be implemented in real time on the robots given the short time required for its execution (about 50ms). It has also been shown that given the nature of the algorithm presented, it can be modeled through game theory, which broadens

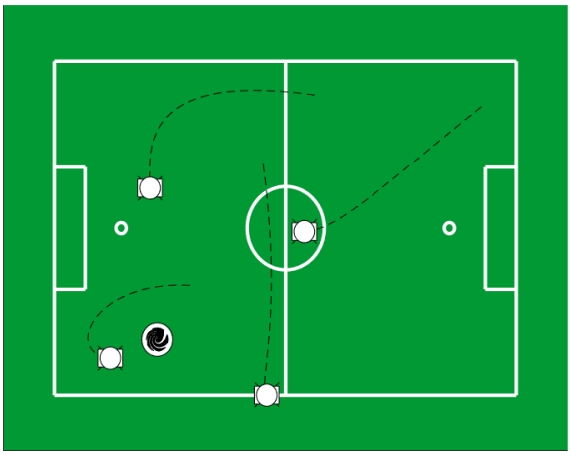

Fig. 3. Optimal location of a group of 4 robots when the optimal square around the ball cannot be reached. In this figure, the dashed line represents the trajectory that the robots followed in order to achieve the optimal location. For this case, the robots find the optimal polygon, but they do no center around the ball due to the fact that it would lead to a quite small square.

the horizon of study, because it can be spoken of a much deeper optimality, i.e., the Nash equilibrium, than only determining the minimum of a convex function.

As a future work, the aim is not only to determine the optimal location of the robots on the field, but also to determine the best possible route, considering obstacles and restrictions, to reach that location. This can be implemented through another evolutionary algorithm that guarantees that all party restrictions are satisfied and that an optimal solution can be found.

References

1. Barreiro-Gómez, J., Quijano, N., Ocampo-Martinez, C.: Distributed control of drinking water networks using population dynamics: Barcelona case study. In: 53rd IEEE Conference on Decision and Control. pp. 3216–3221. IEEE (2014)
2. Basar, T., Olsder, G.: Dynamic Noncooperative Game Theory: Second Edition. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104) (1999)
3. Chang, C.T.: Multi-choice goal programming model for the optimal location of renewable energy facilities. Renewable and Sustainable Energy Reviews **41**, 379–389 (2015)
4. Davis, L.: Handbook of genetic algorithms (1991)
5. Gu, D.: A differential game approach to formation control. IEEE Transactions on Control Systems Technology **16**(1), 85–93 (2008)

6. Lasry, J.M., Lions, P.L.: Mean field games. Japanese journal of mathematics **2**(1), 229–260 (2007)
7. Liu, Y., Tang, W., He, J., Liu, Y., Ai, T., Liu, D.: A land-use spatial optimization model based on genetic optimization and game theory. Computers, Environment and Urban Systems **49**, 1–14 (2015)
8. Lopez, B., Innocenti, B., Busquets, D.: A multiagent system for coordinating ambulances for emergency medical services. IEEE Intelligent Systems **23**(5), 50–57 (Sept 2008). https://doi.org/10.1109/MIS.2008.76
9. Nash, J.: Non-cooperative games. Annals of mathematics pp. 286–295 (1951)
10. Nguyen, H.K., Song, J.B., Han, Z.: Demand side management to reduce peak-to-average ratio using game theory in smart grid. In: 2012 Proceedings IEEE INFOCOM Workshops. pp. 91–96. IEEE (2012)
11. Pure, R., Durrani, S.: Computing exact closed-form distance distributions in arbitrarily-shaped polygons with arbitrary reference point. The Mathematica Journal **17**, 1–27 (2015)
12. Ramirez-Jaime, A., Quijano, N., Ocampo-Martinez, C.: A differential game approach to urban drainage systems control. In: 2016 American Control Conference (ACC). pp. 3796–3801 (July 2016). https://doi.org/10.1109/ACC.2016.7525504
13. Röfer, T., Laue, T., Kuball, J., Lübken, A., Maaß, F., Müller, J., Post, L., Richter-Klug, J., Schulz, P., Stolpmann, A., et al.: B-human: Team Report and Code Release 2016. Universität Bremen (2016)
14. Shi, H.Y., Wang, W.L., Kwok, N.M., Chen, S.Y.: Game theory for wireless sensor networks: a survey. Sensors **12**(7), 9055–9097 (2012)