Analysis of the PSO parameters for a robots positioning system in SSL*

Marcos Aurelio Pchek Laureano[1,2][0000-0002-9399-7633] and Flavio Tonidandel[2][0000-0003-0345-668X]

[1] Federal Institute of Parana, Curitiba – PR, Brazil marcos.laureano@ifpr.edu.br
http://curitiba.ifpr.edu.br/
[2] University Center of FEI, São Bernardo do Campo – SP, Brazil
flaviot@fei.edu.br
https://portal.fei.edu.br/

Abstract. The changes in the Small Size League rules have brought greater possibilities of playing. With the increased complexity of soccer matches, the positioning of the robots has become important as a defense and attack mechanism. The learning of opposing team game playing has been shown to be effective, but an SSL soccer match indicates the need for solutions that analyze the strategy of the opposing team during the game and make any necessary adaptations. This paper proposes the use of the Particle Swarm Optimization (PSO) algorithm as an option to determine the positioning during the match. A prototype has been developed to validate the configuration parameters. Experiments in a simulator, analysis of game logs and results in a real matches have demonstrated the feasibility of applying the PSO algorithm to find the robots positions.

Keywords: robot soccer · Particle Swarm Optimization (PSO) · Small Size League (SSL)

1 Introduction

RoboCup is a robotics competition designed to encourage research into artificial intelligence techniques through friendly matches. RoboCup Small Size League (SSL) competition focuses on the problem of cooperation and multi-agent control in a dynamic environment. SSL-related surveys focus on analyzing the historical records with learning algorithms to identify and classify sets of opponent's moves. The learning has proved to be effective [8, 12], but the complexity of a soccer match indicates the need for solutions that can analyze the strategy of the opposing team during the game and make any necessary adaptations. In this context, it's necessary to use techniques that result in low computational time.

* This study was financed in part by Instituto Federal do Paraná – Brasil and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

SSL has already implemented certain changes and others are being addressed for the near future [17]. The main changes in the SSL category in 2018 were the division into two leagues (A and B), a new field design and an increased number of robots in the field. Since these changes enable various combinations of defense and attack moves, a dynamic system for positioning the defense is important for the success of a soccer team [4].

Many papers involving swarm intelligence and robotic algorithms can be found in the literature [1,11,13,16]). Studies on the social behavior of organisms have inspired the development of efficient optimization algorithms. The Particle Swarm Optimization (PSO) proposed by Kennedy and Eberhart [6] is an optimization algorithm based on a population of particles. It has been acknowledged for solving several problems with simplicity and a few computational resources.

The first objective of this paper is to test, verify, and determine the configuration parameters (inertia, confidence, number of iterations and population size) of the PSO algorithm in order to optimize robot positioning in the field. The second is to demonstrate the effective positioning of robots based on the defense fitness function developed for this study.

The remainder of the paper is organized as follows: section 2 describes the PSO algorithm; section 3 refers to papers on soccer robots applications; section 4 explains the defense fitness function applied to the PSO; section 5 explains the PSO parameter choices for this proposal; section 6 addresses the simulations; section 7 analyzes the simulations, and section 8 presents the conclusions of this paper.

2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here called particles, and moving these particles around in the search space according to the mathematical formula over the particles position and velocity [6]. To find the ideal solution, each particle moves toward its best position (*pbest*) and the best overall position (*gbest*) in the swarm. These equations can be observed in (1) [19]:

$$\begin{aligned} pbest(i,t) &= min[f(P_i(k))], i \in 1,...,N_p, k=1,...,t \\ gbest(t) &= min[f(P_i(k))], i=1,...,N_p, k=1,...,t \end{aligned} \tag{1}$$

Where i is the index of the particle; N_p indicates the total number of particles; t informs the current iteration; f is the function fitness, and, P_i indicates the position of the particle. Each particle has a randomly associated velocity, thus allowing it to move through the search space of potential solutions to the problem being optimized. In the implementation of the PSO algorithm, the particle velocity is updated based on inertia, cognition and social components. The velocity V and the position P of the particles are updated from equations (2) and (3), respectively:

$$V_i(t+1) = \underbrace{\underbrace{\omega V_i(t)}_{\text{inertia}}}_{\text{diversification}} + \underbrace{\underbrace{c_1 r_1 (pbest(i,t) - P_i(t))}_{\text{cognitive}} + \underbrace{c_2 r_2 (gbest(t) - P_i(t))}_{\text{social}}}_{\text{intensification}} \quad (2)$$

$$P_i(t+1) = P_i(t) + V_i(t+1) \quad (3)$$

Where V denotes velocity, ω is the inertia factor used to balance global and local exploration, r_1 and r_2 are randomly distributed in the range $[0, 1]$, and c_1 and c_2 are constant parameters called acceleration coefficients. The variable c_1 represents how much the particles tend to follow their past behaviors in case of success, and c_2 represents their tendency to follow the success of other particles.

The PSO algorithm can work with several neighborhood topologies, the most frequent of which are best global (*gbest*) and local best (*lbest*) topologies [7, 19]. The neighborhood topology determines the set of particles that contribute to the calculation of the best result of a particle. The *gbest* topology is represented by a fully connected graph. Each particle is influenced by its best position and also by the best position of a swarm particle. In the *lbest* topology, each particle is bound to k particle. Each particle is influenced by its personal best position and the best neighborhood position.

A PSO algorithm with global best topology converges rapidly because all particles are strongly attracted by the best particle in the swarm, thus producing sub-optimal results when the best particle is trapped in a local minimum. On the other hand, a PSO algorithm with local best topology converges slowly, but with greater chances of finding the global optimum, because each particle is influenced only by its adjacent neighbors and, therefore, groups of neighbors can explore different regions or great places in the search space [7].

The PSO was chosen as the base algorithm due to some of its characteristics: it is versatile, applied in several areas of knowledge, converges rapidly to a set of satisfactory answers, has low computational cost in comparison with other optimization algorithms, is less complex and requires a small number of parameters to be optimized, and allows a greater flexibility between local and global exploration in the desired search space when compared to other algorithms. The authors Wahab et al. [15] demonstrated that the PSO is considered one of the best optimization algorithms.

3 Related Studies

Many techniques have been applied to the decision-making of a robot soccer game, including: Case-based Reasoning, Learning from Observation, Reinforcement Learning, Pattern Recognition, Fuzzy Theory, Neural Network, Evolutionary Algorithm and Decision Tree.

Wang et al. [16] proposed to use the PSO to calculate route planning and detour. Their proposal was applied and simulated in a soccer game of robots with five members on each side. According to the authors, the use of PSO has proved to be possible, simple and easy to implement. However, the speed of the ball and the robots were considered constant, which does not reflect a real game situation. Saska et al.'s proposal [13] uses the PSO algorithm combined with Fergunson splines for the calculation of trajectories of robots on a soccer field. According to the authors, the use of splines is an implementation of movement that is natural for robots and easy to implement. Such calculation of trajectories reduces the computational cost of processing since one does not need to explore all possible paths as in traditional path-planning algorithms.

In Okada et al. [11], PSO is used to find soccer team positions in the 2D Simulator League field. In the simulations, the ball had 15 fixed positions and the PSO was used to find possible configurations of the position of the players in the field. According to the authors, the formations found showed no tendency for offensive or defensive positions, so the players can remain close to midfield.

There are other studies related to SSL. Mendoza et al. [9] suggested a Selectively Re-active Coordination (SRC) algorithm, which contains two layers: a coordinated opponent-agnostic layer enables the team to create its plans, setting the space for an offensive game, and an individual opponent-reactive action selection layer enables the robots to maintain re-activity to different opponents. This approach allows a tradeoff between team's ability to create team plans independently of the opponents, and its ability to react appropriately to different opponent behaviors. Quintero et al. [12] applied machine learning techniques to the problem of predicting soccer plays. The authors demonstrated experimentally that it is possible to predict the play that certain team will perform in a SSL game. Schawb et al. [14] applied deep reinforcement learning (DRL) to train skills. They have demonstrated learning of two different skills: navigating to a ball and aiming and shooting. Although they are not perfect, these learned skills allow a performance close to that of hand-coded baseline skills.

Albad et al. [1] applied the swarm algorithm to robot soccer to try to solve strategy, learning or positioning problems. For RoboCup community, solutions about strategy, learning or positioning apllied in robot soccer is an open problem, even of exists many researches about this.

This proposal differs from other approaches because it uses an optimization algorithm for defensive positions that can respond during a SSL match and based on the movements of the opponent team's robots. A fitness function for a PSO algorithm was created to meet certain behavioral conditions respecting some criteria based on human soccer games.

4 Proposal of this paper

In this paper, the PSO algorithm is used to find a positioning that meets some characteristics of human soccer. According to Costa et al. [4], a soccer match is divided into two concepts: defense and attack. The defense of a team must a) hinder offensive passes (that might result in a goal) and kicks on goal, b) recover ball possession, c) prevent opponent's progression, d) protect the goal and e) reduce opponent's playing space. The attack must: a) maintain ball possession,

b) build up offensive actions, c) progress through the opponent's half, d) create situations of shooting and e) shoot on opponent's goal.

To meet the defense requirements similar to those in human soccer, a fitness evaluation function is used in simulations applied to the defense position. It evaluates four desirable situations for a defense formation:

– A minimum distance among the robots in order to make opponent's movements more difficult and decrease the opponent's chances to receive the ball, make passes or kick to goal;
– The view of the opponent's robots in relation to a certain point of interest is blocked;
– The view of the goal of all the opponent's robots is blocked by least one robot of team, especially opponent robot with ball possession;
– Respect for the SSL rules on collisions between robots and invasion of the goal area.

In equations, field size is considered in centimeters and the *Goal* term can be any point of interest in the search space. For example: the left or right corner of the goalkeeper or a set of points from the opponent's field attack.

4.1 Robot movements

The new robots positioning in the field is calculated by the motion of the particle $p(i, t)$ in search space. The velocity and the position of the robots are based in equations (2) and (3). The new calculated position (new particle $p(i, t + 1)$) is checked in (4) to determine if the new fitness value of particle has become smaller than the current value of *pbest*. If so, then new particle is the new *pbest* ($pbest(i, t) = p(i, t + 1)$). Subsequently, the PSO checks whether the new *pbest* is lower than the current *gbest* value. If so, then the new particle is also the new *gbest* ($gbest(t) = pbest(i, t)$) of the swarm.

4.2 Punishments and notations

Table 1 shows the punishments applied in the equations. These values represent centimeters in the SSL field. The punishments are used to differentiate situations that may occur during a SSL match.

Table 1: Types of Punishments

Type	Value	Meaning
PLOW	100	Low Punishment (only to differentiate minor cases)
PMLOW	500	Mid Punishment (low impact)
PMID	1000	Mid Punishment
PMHIGH	2000	Mid Punishment (high impact)
PHIGH	5000	High Punishment, violation of SSL rules

4.3 General Equation

Equation (4) represents the four desirable situations for a defense formation. It comprises of another five functions (detailed in the following subsections).

$$\begin{aligned}\text{fitness}(A, p(i,t), goal) = & \, \text{f}_{\text{MinDistance}}(A, p(i,t)) + \\ & \text{f}_{\text{CheckStraight}}(A, p(i,t), goal) + \\ & \text{f}_{\text{ProtectGoal}}(A, p(i,t)) + \\ & \text{f}_{\text{Colission}}(A, p(i,t)) + \\ & \text{f}_{\text{InvasionGoalArea}}(p(i,t))\end{aligned} \tag{4}$$

Where A represents the set of opponent's robots, $p(i,t)$ represents the set of robots that make up the defense (it is the particle of the swarm), i is a particle in iteration t, and *goal* represent the goal or another point of interest. The set A considered in the other equations follows the rule described in (5).

$$\begin{gathered} RA = \downarrow_{1}^{9}(\measuredangle)A \\ A = \begin{cases} RA_{1\ldots l} \in RA, & \text{if } l = k \\ RA_b + RA_{1\ldots(l-k+1)} \subset RA, & \text{if } l > k \\ RA_{1\ldots l} \in RA, & \text{if } l < k \end{cases} . \end{gathered} \tag{5}$$

Where RA is the set A ordered according to the angular view of the goal, RA_b represents the opponent robot with ball possession, l is the number of opposing robots, and k is the number of robots that make up the defense. Robots with lower angles of vision of the goal will have more difficulties to kick the ball in its direction.

4.4 Minimum distance

The $\text{f}_{\text{MinDistance}}$ (6) function sums the distances between the opponent's robots and the robots that make up the defense. A minimum distance is desirable to obstruct kicks and passes.

$$\begin{aligned} \text{f}_{\text{MinDistance}}(A, P) = \{ & \\ & \text{distance}(p,a) = \sqrt{(p_x - a_x)^2 + (p_y - a_y)^2} \\ & d = \text{distance}(p,a) \\ & I_{c_d}(p,a) = \begin{cases} (1 - \frac{40}{d}), & \text{if } (1 - \frac{40}{d}) > 0 \\ \text{PMID}, & \text{otherwise} \end{cases} \\ & \sum_{a \in A} \sum_{p \in P} I_{c_d}(p,a) \\ & \} \end{aligned} \tag{6}$$

P represents the set of robots that make up the defense, A represents the set of opponent's robots, a is a robot from set A, and p is a robot from set P. The I_{c_d} operator is used to ensure the minimum distance between a team robot and an opponent robot. Value 40 indicates a dimension of two robots in SSL ($\approx 40\ cm$). One value > 0 indicates that the robot is behind the opponent's robot.

4.5 Blocking the view of points of interest

The $f_{CheckStraight}$ (7) function verifies if the opponent robots' view of the goal or another point of interest is blocked, which is desirable to make passes and kicks difficult. A high sum indicates that several of the opponent's robots have vision of the goal.

$$\begin{aligned}
\mathrm{f}_{\mathrm{CheckStraight}}(A, P, goal) = \{ & \\
& \mathrm{straight}(a,p,g) = \{((g_y - p_y) \times a_x + \\
& (p_x - g_x) \times a_y + (g_x \times p_y - p_x \times g_y))\} \\
& s = \begin{cases} 0, & \text{if } a \text{ has the ball} \\ \mathrm{PLOW}, & \text{otherwise} \end{cases} \\
& I_{c_s}(p, a, goal) = \mathrm{straight}(a, p, goal) + s \\
& \sum_{a \in A} \sum_{p \in P} I_{c_s}(p, a, goal) \\
\} &
\end{aligned} \tag{7}$$

The straight operators returns the distance between robot p and the line formed between a and $goal$. If a does not have the ball, $PLOW$ punishment is applied, so the formations that protect the goal of the player with the ball tend to be privileged.

4.6 Blocking the view of the goal

The $\mathrm{f}_{\mathrm{ProtectGoal}}$ is a function to penalize whenever robot p does not protect its team's goal from the opponent's robot with the ball. A high sum indicates that the opponent's robot A_b has a view of the goal and is free to kick or pass the ball. In that case, if there are more robots in the defense than opponent's robots, the opponent's robot with ball possession may be blocked by more robots.

$$\mathrm{f}_{\mathrm{ProtectGoal}}(A, P) = \mathrm{PMID} \times (\sum_{p \in P} \ \forall\ p \notin A_{ball}) \tag{8}$$

4.7 Respect SSL rules

The function $\mathrm{f}_{\mathrm{Colission}}$ (9) prevents a collision among a team's own robots or the opponent's robots.

$$\begin{aligned} \mathrm{f_{Colission}}(A,P) = \mathrm{PHIGH} \times ((\sum_{a\in A}\sum_{p\in P} \mathrm{pos}(a) = \mathrm{pos}(p) + \\ (\sum_{p\in P}\sum_{q\in P} \mathrm{pos}(p) = \mathrm{pos}(q) \ \wedge \ p_{id} \neq q_{id})) \end{aligned} \quad (9)$$

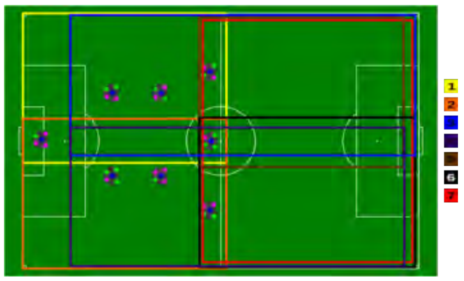
In (9) the comparison between robots uses the field coordinates. Example: $\mathrm{pos}(a) = \mathrm{pos}(p)$ indicates that $a_x = p_x$ and $a_y = p_y$, and $p_{id} \neq q_{id}$ comparing team robots id.

Function $\mathrm{f_{InvasionGoalArea}}$ (10) prevents the invasion of the goalkeeper's area.

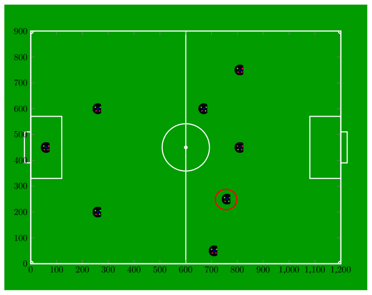
$$\mathrm{f_{InvasionGoalArea}}(P) = \mathrm{PHIGH} \times (\sum_{p\in P} \forall\ p \in \text{Goal area}) \quad (10)$$

5 Simulation parameters

To perform the experiments, the opponent's robots were positioned in the classic 2–2–3 formation applied 8 × 8 soccer. In this formation, the players have a well defined field of attack and defense (figure 1a). Figure 1b shows an example of the position of the opponents used in the simulations, chosen randomly according to the distribution in figure 1a and respecting the SSL rules. One robot was randomly chosen to possess the ball.



(a) 2–2–3 formation – colors indicate area of action for each robot



(b) Example of distribution in simulations

Fig. 1: 2–2–3 Formation in 8 versus 8 Soccer.

Inertia weight (ω) plays a key role in providing balance in the local and global exploitation process. It determines the contribution rate of the particle's previous velocity to its velocity in the next step. Bansal et al. [2] compared several proposals of inertia weights and concluded that the constant value 0.7298 (proposed by Clerc and Kennedy [3]), the random value $0.5 + \frac{rand(0,1)}{2}$ (proposed by Eberhart and Shi [5]), and the linear value $(\omega_{start} - \omega_{end}) \times \frac{it_{Max} - it}{it_{Max}} + \omega_{end}$ proposed by Xin et al. [18] (where $\omega_{start} = 0.9$ and $\omega_{end} = 0.4$, it_{Max} is the maximum of iterations, and *it* the current iteration) have a smaller number of iterations or less error.

The acceleration coefficients (c_1 and c_2) were chosen based on literature, according to which the most used values are: $c_1 = c_2 = 2$ and $c_1 = c_2 = 1.496$, in the simulations the values $c_1 = c_2 = 1$, $c_1 = 2, c_2 = 1$ and $c_1 = 1, c_2 = 2$ were adopted. Coefficient c_1 controls the diversity of each particle and c_2 controls their the global diversity. When $c_1 > c_2$, the particle tends to move to the *pbest* position and when $c_1 < c_2$, the particle tends to move to the *gbest* position in the swarm. Neighborhood topologies for global best (*gbest*) and local best (*lbest*) were used in the simulations.

Fifteen simulation scenarios were chosen to find the best parameters for inertia (ω) and the cognitive (c_1) and social coefficients (c_2). Each scenario comprised 5 instants (15 seconds of movement) to simulate a real game situation. The population size (from 50 to 200) and the number of iterations (from 50 to 500) varied in each scenario.

The configurations that provided best results were with 100 particles in the swarm population and 300 iterations, above this there was no improvement in results. Each particle consists of five robots positioned in the defense (with the coordinates x, y and velocities vx, vy – a robot is only one point in the search space), goalkeeper was not considered. The current position of the robots in the field is always part of the swarm population, since the movement may have been minimal and the current position can continue to have the best fitness. The search space dimensions are defined by 900 (D_{maxY}) $\times$ 1200 (D_{maxX}) points to represent the total dimensions of a SSL field. In order to to make up a particle in the swarm, each robot's position and velocity are defined as ($x = rand(0, D_{maxX})$, $y = rand(0, D_{maxY})$) ($vx = 1, vy = 1$), and fixed position representing the goal is defined at 450,1150. The *lbest* topology with the same parameters obtained results similar to those for the *gbest* topology but with a 30% higher average execution time.

Figure 2 presents the results from simulation scenarios with 300 iterations, population of 100 particles, and *gbest* topology after 10.000 simulations. As figure 2 shows, the minimum value (the desired value, since it is a minimization function) is the same in all simulations (variation ≈ 0.1). Other field positions get similar values. Figures 3a, 3b and 3c show some examples of the positions obtained for each inertia factor.

All scenarios with $c_1 = c_2 = 2$ have the best results, followed by the scenario with $c_1 = c_2 = 1.496$. In the tests performed, the inertia adopted showed little influence on the final result of the algorithm. A fixed inertia ($\omega = 0.7298$) was chosen (proposed by Clerc and Kennedy [3]). These values are the most used in several studies about PSO.

6 Experiments

Three types of experiment were performed using (2) and (3) with parameters $\omega = 0.7298, c_1 = c_2 = 2$ to calculate the velocity and position of each robot that

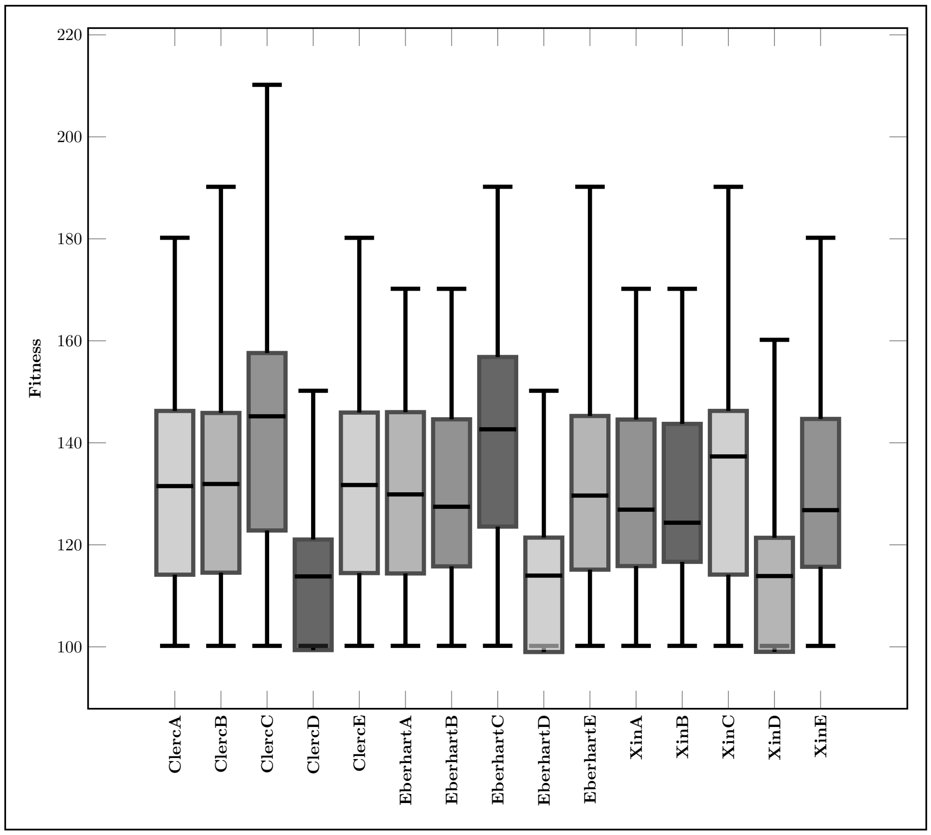
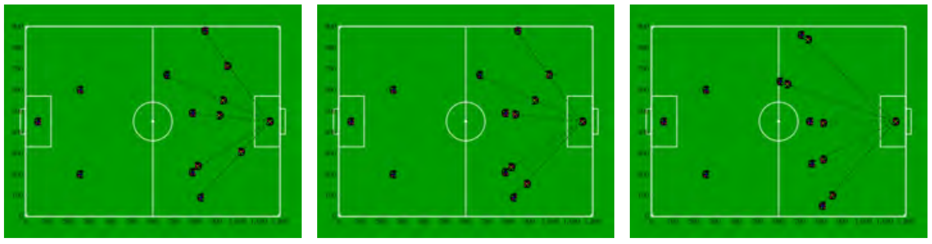


Fig. 2: Scenario settings: a) $c_1 = 1$ and $c_2 = 2$; b) $c_1 = 2$ and $c_2 = 1$; c) $c_l = c_2 = 1$; d) $c_1 = c_2 = 2$ and e) $c_1 = c_2 = 1.496$.

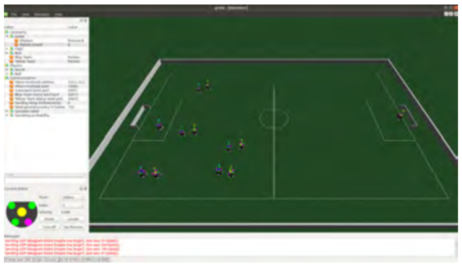


(a) Inertia proposed by [3]. (b) Inertia proposed by [5]. (c) Inertia proposed by [18].

Fig. 3: Some results for each inertia.

make up the particle, a grSim simulator [10], log analysis of RoboCup 2018 and the *Latin American Robotics Symposium* 2018 (LARS 2018) in five real matches.

In the simulator and in real matches the algorithm runs every 100ms. The current positioning is a particle that makes up the swarm of the next execution of the algorithm. The algorithm returns the new positioning and the strategy system decides which robot moves to each position (based on shortest distances). Figure 4 demonstrates the strategy system running together with the grSim simulator.


(a) grSim

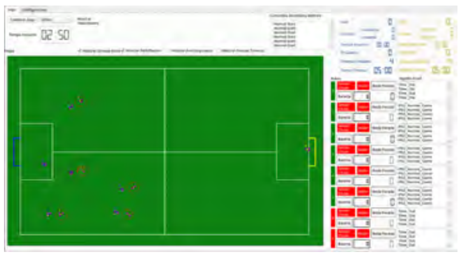

(b) Strategy System

Fig. 4: Experiments in simulator.

In the grSim simulator, the ball and the opponent's robots were positioned differently for visual verification of the behavior of the team robots. These validations allowed us to identify and adjust some parameters of the fitness function.

To validate the proposal of this paper, five situations of passing and goal were selected from the RoboCup 2018 playoffs. Each scenario was evaluated 1,000 times with the parameters of the algorithm found in the previous experiments. In the goal scenario, the particle had 4 robots to defend the goal (the other team's robots were not considered). In the pass scenario, in addition to the defense particle (previously with 4 robots), another particle was made up with the remaining 3 robots in order to block the pass. The algorithm was run in parallel with two particle populations (one to defend the goalkeeper's area and another to block passes). In the ball interception scenario, the fitness function was adjusted to receive the opponent's robots that did not have ball possession as points to defend. The results can be seen in figure 5.

Finally, during LARS 2018 the proposal was applied in five real games. These experiments allowed us to verify the positioning limitations caused by the time spent in processing and sending commands to the robots.

7 Analysis

The various simulations and matches have show that using parameters for $c_1 = c_2 = 2$ leads to the positions that are closest to the ideal. Even the worst positioning found, based on the worst minimized value that was calculated (figure 6a) provides a satisfactory positioning for the goal defense.

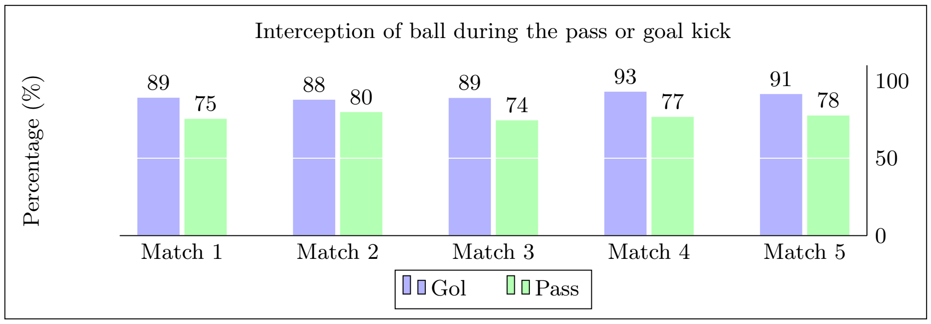
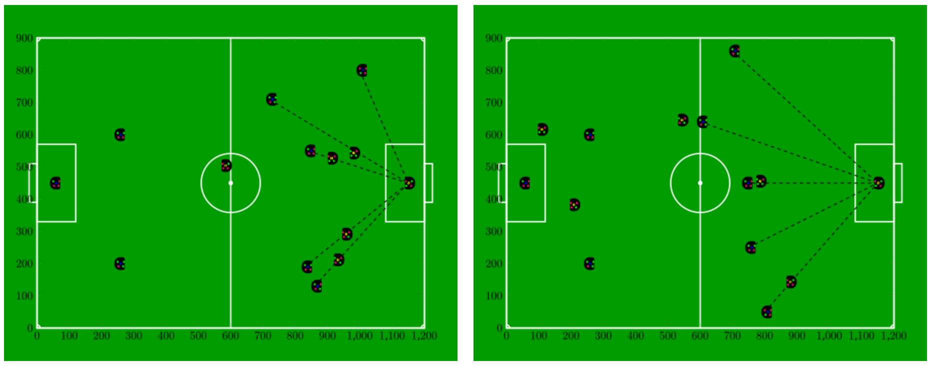


Fig. 5: Percentage of interception success in five RoboCup 2018 matches.

Figure 6b shows the worst positioning situation ($c_1 = c_2 = 1$). In this case, the solution found is far below the ideal for a defense positioning system. For this scenario and search space, higher values for c_1 and c_2 are more effective because they result in the gradual decrease of the amplitude of the trajectory of the particles, thus ensuring the convergence of the algorithm [19].



(a) $C_1 = C_2 = 2$. (b) $C_1 = C_2 = 1$.

Fig. 6: Worsts cases – Inertia proposed by Clerc and Kennedy [3].

Experiments with logs have demonstrated the effectiveness of the proposal of this paper, but the opponent's strategy system would probably make other moves to succeed in goal-kicking or pass-through. In the scenario to block passes, as there were more opponent's robots in the attack area that could receive a pass than defenders to block them, the percentage of success was lower than in the scenario to defend the goalkeeper's area.

Experiments conducted during LARS 2018 have shown that it is necessary to anticipate position of the opponent's robot and ball. For the time between the processing of the new positions is no longer desired due to the movement of the opposing team (about 3 seconds for calculating the positioning, the system sending the commands to robots and robots performing the movement). In games with dead-ball situations, the positioning found by the system was always a difficult one for passes or kicks to goal. This proposal does not consider high kicks since the robots do not have the mechanical ability to jump.

8 Conclusions

The contribution of this paper was to verify the application of the PSO algorithm to robot soccer matches. We have analyzed the optimal parameters for the algorithm configuration. The initial prototype proved the feasibility of using the algorithm. The experiments indicate that the best global neighborhood topology, the number of iterations (300), acceleration coefficients ($c_1 = c_2 = 2$), and the size of the population (100) meet the project requirements in terms of computational costs to be run during a real soccer match. For inertia, all the evaluated strategies were effective, and we adopted the value $\omega = 0.7298$.

The main advantage of this approach is the possibility of optimizing, in real time and without previous knowledge, combinations of positioning with a low computational cost, which would not be possible with other techniques [12, 14].

The analysis of RoboCup playoff logs has demonstrated the effectiveness of the proposal for this paper. Experiments conducted during LARS 2018 have shown that it can be used to find field positioning during an real SSL soccer game, especially in games with dead-ball situations (e.g. indirect kicks). However, for a dynamic game, the movements of the opponent's must be considered in order to calculate the future positioning, which is the main challenge to be improved on future studies.

References

1. Albab, R.T.U., Wibowo, I.K., Basuki, D.K.: Path planning for mobile robot soccer using genetic algorithm. In: 2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA). pp. 276–280 (Sep 2017). https://doi.org/10.1109/ELECSYM.2017.8240416
2. Bansal, J.C., Singh, P.K., Saraswat, M., Verma, A., Jadon, S.S., Abraham, A.: Inertia weight strategies in particle swarm optimization. In: 2011 Third World Congress on Nature and Biologically Inspired Computing. pp. 633–640 (Oct 2011). https://doi.org/10.1109/NaBIC.2011.6089659
3. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1), 58–73 (Feb 2002). https://doi.org/10.1109/4235.985692
4. da Costa, I.T., da Silva, J.M.G., Greco, P.J., Mesquita, I.: Princípios táticos do jogo de futebol: conceitos e aplicação. Motriz. Revista de Educação Física **15**(3), 657–668 (julho 2009)
5. Eberhart, R.C., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). vol. 1, pp. 94–100 vol. 1 (2001). https://doi.org/10.1109/CEC.2001.934376

6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on. vol. 4, pp. 1942–1948 vol.4 (Nov 1995). https://doi.org/10.1109/ICNN.1995.488968
7. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. vol. 2, pp. 1671–1676 (2002). https://doi.org/10.1109/CEC.2002.1004493
8. Larik, A.S., Haider, S.: A survey of nature inspired optimization algorithms applied to cooperative strategies in robot soccer. In: 2018 International Conference on Advancements in Computational Sciences (ICACS). IEEE (feb 2018). https://doi.org/10.1109/icacs.2018.8333485, https://doi.org/10.1109/icacs.2018.8333485
9. Mendoza, J.P., Veloso, M., Simmons, R.: Selectively reactive coordination for a team of robot soccer champions. In: Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI) (2016)
10. Monajjemi, V., Koochakzadeh, A., Ghidary, S.S.: grSim – RoboCup Small Size Robot Soccer Simulator, pp. 450–460. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32060-6-38
11. Okada, H., Wada, T., Yamashita, A.: Evolving robocup soccer player formations by particle swarm optimization. In: SICE Annual Conference 2011. pp. 1950–1953 (Sept 2011)
12. Quintero, C., Rodríguez, S., Pérez, K., López, J., Rojas, E., Calderón, J.: Learning Soccer Drills for the Small Size League of RoboCup, pp. 395–406. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-18615-3-32
13. Saska, M., Macas, M., Preucil, L., Lhotska, L.: Robot path planning using particle swarm optimization of ferguson splines. In: 2006 IEEE Conference on Emerging Technologies and Factory Automation. pp. 833–839 (Sept 2006). https://doi.org/10.1109/ETFA.2006.355416
14. Schwab, D., Zhu, Y., Veloso, M.: Learning Skills for Small Size League RoboCup. In: Proceedings of the RoboCup Symposium. Springer, Montreal, Canada (June 2018), nominated for Best Paper Award
15. Wahab, M.N.A., Nefti-Meziani, S., Atyabi, A.: A comprehensive review of swarm optimization algorithms. PLoS One **10** (May 2015). https://doi.org/10.1371/journal.pone.0122827
16. Wang, L., Liu, Y., Deng, H., Xu, Y.: Obstacle-avoidance path planning for soccer robots using particle swarm optimization. In: 2006 IEEE International Conference on Robotics and Biomimetics. pp. 1233–1238 (Dec 2006). https://doi.org/10.1109/ROBIO.2006.340104
17. Weitzenfeld, A., Biswas, J., Akar, M., Sukvichai, K.: RoboCup Small-Size League: Past, Present and Future, pp. 611–623. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-18615-3_50, http://dx.doi.org/10.1007/978-3-319-18615-3_50
18. Xin, J., Chen, G., Hai, Y.: A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In: 2009 International Joint Conference on Computational Sciences and Optimization. vol. 1, pp. 505–508 (April 2009). https://doi.org/10.1109/CSO.2009.420
19. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical Problems in Engineering **2015**, 38 (2015). https://doi.org/http://dx.doi.org/10.1155/2015/931256