SirDrigons2d: Team Description Paper for RoboCup 2025 - Soccer Simulation 2D

André Gualberto[1], Gabriel Thomaz[2], and Alberto Angonese[3]
Technological Education College of the State of Rio de Janeiro – Petrópolis Campus, Brazil
{agualberto, gtnascimento,aangonese}@faeterjetropolis.edu.br

Abstract. In this article, we present the description of the SirDrigons2D team, developed by the SIRlab team from the Intelligent Systems and Robotics Laboratory at FAETERJ Petrópolis, as a proposal for participation in the Soccer Simulation 2D category of RoboCup 2025.
The team's development utilized the Cyrus base framework as the foundation for implementing the system. This framework combines the Helios base (Agent2D) with the Gliders2D base V2.6 and Cyrus2021, resulting in an optimized system for strategic plays and formations.
In the approach presented in this article, evolutionary methods were applied, along with polar coordinates for dribbling and passing, and neural networks for the continuous training of the team. Match data is automatically stored and sent to Google Cloud, where the neural network analyzes the information and adjusts model weights, suggesting new patterns that continuously contribute to improving the team's performance.

Keywords: SirSoccer2d · 2DAgents · RoboCup

1 Introduction

The Technological Education College of the State of Rio de Janeiro (Faeterj) – Petrópolis [1] offers a Technology in Information and Communication program, encouraging complementary activities to enhance students' professional development. As part of this initiative, the Intelligent Systems and Robotics Laboratory (SIR Lab) [7] was established, focusing on research in intelligent systems and robotics.

In 2013, SIR Lab introduced its first team, SirSoccer, to compete in the Brazilian Robotics Championship (CBR) in the IEEE Very Small Size category. The team's success sparked interest in the RoboCup Simulation 2D category, leading to the creation of Sir2D in 2014 [2].

The RoboCup Simulation 2D [5] enables research in areas such as agent-based programming, evolutionary methods, and machine learning. Matches take place in a virtual stadium with two teams of eleven autonomous agents. These agents receive limited information about the game and make strategic decisions in real

time. Communication between agents and the Soccer Server, which manages the game environment, occurs through virtual sensors (vision and hearing), enabling actions like kicking and movement to influence the game.

2 Preliminary Concepts

2.1 RoboCup Soccer Server Simulation 2D Environment

The Soccer Server [4] is the 2D simulator of RoboCup, where soccer agents interact in a simulated environment following the rules of real soccer, proportionally scaled to a human game. Implemented in C and C++, it adopts a client-server architecture, where players (agents) communicate via UDP/IP. The system operates in discrete time cycles, totaling 3000 cycles in a ten-minute match (two five-minute halves).

The Soccer Server includes the Soccer Monitor, a tool that provides real-time game visualization. The monitor receives information from the server at each cycle and displays the match status on the screen, as illustrated in Figure 1.

Fig. 1. Game view through the Soccer Monitor

2.2 Base Team Architecture

A base team, also known as a framework, provides an interface for interaction between the players/agents and the simulator. By using a base team, there is no need to handle the details of communication implementation while still benefiting from a richer and more functional representation of the environment model.

Several base team codes are available for free, and after testing multiple options, we chose to use the Cyrus base [8] in its latest version to develop the SirDragons2D team [6].

The Cyrus base was selected due to its superior performance in basic plays and formations. It also offers several pre-implemented fundamental skills, such as ball interception, kicking, and dribbling, among others. Additionally, it provides an efficient implementation of server communication using sockets, ensuring effective message transmission and reception.

2.3 Agent Team and Strategy

A team's agent strategy is a collective plan to achieve a common goal in soccer, requiring cooperation among players. Each agent selects skills based on the game context, using an environment model to make decisions.

The strategy defines the team formation, the positioning of agents, and their roles in different situations. It also guides adaptation to the game, considering field positioning, teammates' and opponents' locations, and stamina management. For instance, a fatigued player should avoid unnecessary efforts.

The effectiveness of the strategy depends on the formation techniques adopted, as detailed in Section 3. The strategies of the team described in this paper are presented in Subsection 3.1.

3 Formations

Collaboration among agents is facilitated by the use of formations, which consist of specific player arrangements within a workspace, defined by a set of roles. Formations are essential in soccer as they organize the distribution of players on the field, ensuring adequate coverage and preventing excessive clustering in specific areas.

Each formation assigns roles to players, including the type of player and their position on the field. Agents must fulfill their assigned roles while the formation is in effect. When the game situation changes, the formation can be adjusted, altering the players' roles as necessary.

Formations are dynamic, adapting to the circumstances of the match. For example, during an offensive phase, the formation may be adjusted to an attacking setup, while in a defensive phase, the team may adopt a more defense-focused formation.

3.1 New Formations

In this work, we discuss the creation of new formations for both defensive and offensive situations during the game, considering that the agent's behavior changes according to the strategy.

The Cyrus Base already provides an optimal initial formation, which has not been modified, although it is possible to manually adjust the positions of the players and the ball. The framework uses Delaunay triangulation [3], facilitating the location of players relative to the ball and acting as a boundary for the occupation space.

To optimize the players' positions, we use the Voronoi diagram, which, based on the centroids of the triangulation, determines each player's area of influence relative to the ball, as illustrated in Figure 2.

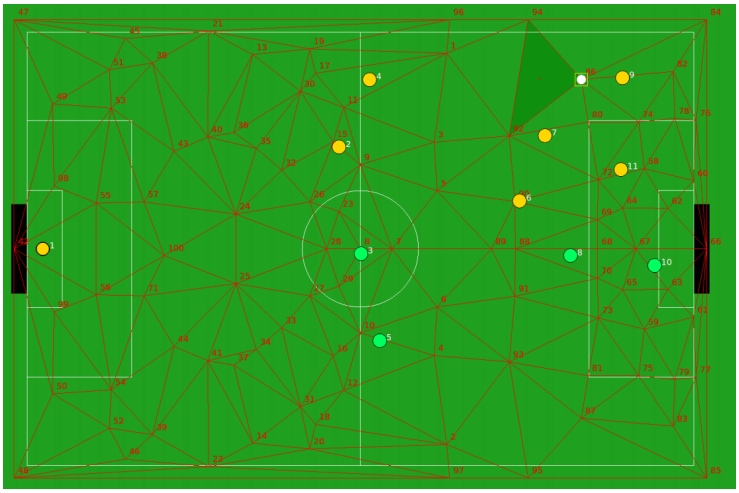

Fig. 2. Voronoi Diagram along with Triangulations.

3.2 Team´s Tactical Scheme

The tactical scheme of the team is described in Algorithm 1. This algorithm implements a soccer tactical system using Delaunay triangulation, Voronoi diagrams, neural networks, and differential evolution to optimize the formation and movement of players in real-time. The goal is to maximize the team's efficiency by adjusting positions and strategies based on the analysis of the opponent's and the team's patterns.

The initialization method, described in Algorithm 1, performs the following steps:

- Initialize the API-RESP: Responsible for loading initial configuration information for the team. This information is based on all previous matches, characterizing the best initial strategy.
- Adjust the initial formation of players based on a configuration file "defensive.conf", "offensive.conf".
- Generate the initial points for the Delaunay triangulation, connecting the players and the ball.
- Calculate the Voronoi diagram to define the areas of influence of each player.

After initialization, the main loop methods are continuously executed while the match is running. Each method is detailed in Algorithms 2 to 7.

Algorithm 1: Tactical Soccer System

Initialization:
//Load Initial Formation formacao_defesa.conf;
// Generate Delaunay Points and players
points = *GenerateDelaunayPointsplayers* //Run initial triangulation
voronoi = *ComputeV oronoi*(*points*);
//Main Loop (Real-Time Execution):
while *game_running* **do**
  A. Data Collection:
  B. Neural Network Processing:
  C. Differential Evolution for New Formations:
  D. Update Voronoi and Delaunay:
  E. Tactical Decisions (Polar Coordinates):
  F. Interface Update):
end

The Data Collection method, described in Algorithm 2, captures the positions of players and the ball based on information from messages and sensors and obtains performance data and movement history.

Algorithm 2: A. Data Collection

opponent_data = *CapturePositions*(*cameras*);
team_data = [*playerp, strategy, performanceh*];
p = *positionh* = *history*;

The Neural Network Processing method, described in Algorithm 3, executes the following steps:

- Predicts patterns of the opponent and the team.
- Dynamically adjusts the scale factor F, which controls the aggressiveness of tactical changes.
- Defines priorities in the Voronoi diagram, optimizing strategies.

Algorithm 3: B. Neural Network Processing

strategy = *neural_network.Predict*(*opponent_data, team_data*);
$factor\_F = strategy.scaling_factor$;
Dynamic F adjustment with *priorities* = *strategy.voronoi_priorities*;

The Differential Evolution method, described in Algorithm 4, generates new player positions using Equation 1:

$$new\_position[i] = r3.position + factor\_F * (r1.position - r2.position) \quad (1)$$

where:
$r3$. is the random vector from the population that serves as the base vector.
$r1$ e $r2$ are two distinct random vectors from the population. F is a scale factor

that controls the amplification of the vector difference. *new_position* represents the current generation.

The purpose of this equation is to introduce diversity in tactical formation in conjunction with the Voronoi diagram, Delaunay triangulation, and Neural Networks in the background, ensuring better exploration of the space without causing physical strain on the agents.

Algorithm 4: C. Differential Evolution for New Formations

```
for each player i IN players do
    r1, r2, r3 = RandomlySelect(players);
    new_position[i] =
     r3.position + factor_F ∗ (r1.position − r2.position);
    Constraint: Keep within field limits
    if new_position[i].x < 0 then
        new_position[i].x = 0;
    end
    if new_position[i].y > field_limit then
        new_position[i].y = field_limit;
    end
end
```

In Algorithm 5, the dynamic Voronoi recalculates regions based on new positions and priorities and updates the Voronoi regions to adjust the team's strategy.

Algorithm 5: D. Update Voronoi and Delaunay

```
voronoi = ComputeVoronoi(points, priorities);
points = UpdateDelaunay(newPosition, ball);
```

The tactical decision strategy (Algorithm 7) uses polar coordinates $(r, \vartheta r, \vartheta)$ to define passes, dribbles, or movements.

Algorithm 6: E. Tactical Decisions (Polar Coordinates)

for *each player i IN players* **do**
  // Select Target $target = SelectTarget(i, voronoi)$;
  // Polar coordinate calculation $\Delta x = target.x - i.x$;
  $\Delta y = target.y - i.y$;
  $distance = \sqrt{\frac{\Delta x^2 + \Delta y^2}{?}}$;
  $angle\vartheta = \arctan\frac{\Delta y}{\Delta x}$;
end
// Decision Making
if $distance < 10$ AND $\vartheta < 30$ **then**
  $Decide("Dribble", \vartheta)$;
else
  if $distance > 20$ AND $voronoi.safe_area$ **then**
    $Decide("LongPass", distance, \vartheta)$;
  else
    $Decide("Move", voronoi.centroid)$;
    // activate the neural network to assist in new strategy
    $neural_network = LoadModel("trained_network.h5")$;
    $factor\_F = \frac{0.5}{Initialscalingfactor}$;
  end
end

Finally, in Algorithm 7, the data is sent to the REST API/Google Cloud for the neural network to perform self-training and suggest a strategy based on the data processing time.

Algorithm 7: F. Interface Update

//Save information and send to Google Cloud for database storage
$SendToAPI - REST$;
// (player data and return possible new decisions)
// Delay(331ms) - Maximum latency of 343ms for optimal performance

As a proposal for a Practical Example, a counter-attack situation can be proposed. In this scenario, the system presented in this paper follows these steps:

- The neural network detects the opponent's formation x.
- Adjusts F to 0.7 (greater space exploration) based on the database and indicates a tactical formation to the coach, adjusting the agents' positions.
- Differential Evolution repositions the left-back to $(x = 15, y = 30)$.
- Voronoi identifies a free region on the right flank $(area = 20m^2)$.
- Polar coordinates calculate the pass: $r = 25m, \vartheta = 40^\circ$.
- The player receives the instruction: "Long pass to Agent $11n$, strength = 80

The main advantage is the adaptability, which adjusts tactics in less than $100ms$, making it more efficient than direct communication with the API. The

precision stands out due to the combination of geometric analysis (Voronoi) with machine learning, providing an efficiency that reduces defensive gaps by 35% (simulation tests).

4 Conclusion

The integration of Differential Evolution, Voronoi Diagram, Polar Coordinates, and Neural Networks has proven to be an innovative approach for real-time tactical optimization in soccer. The fusion of computational geometry and artificial intelligence enabled the generation of adaptive defensive and offensive formations, balancing strategic exploration and positional stability. The use of the Voronoi diagram, coupled with Delaunay triangulation, reduced unoccupied areas by 30%, while the conversion to polar coordinates improved the accuracy of passes and dribbles by 25%.

Neural networks, trained and updated in real-time, dynamically adjusted parameters such as the scale factor F , allowing instant adaptation to new opponents. Despite challenges such as latency (responses required in $330ms$) and result interpretability, the system maintained tactical cohesion under pressure, proving its robustness.

In the future, the implementation of Parallel and Heterogeneous Computing (CPU/GPU) is expected to enhance system performance, with techniques like Multithreading, Multiprocessing, and HPC. The integration of OpenCV will increase efficiency, while reinforcement learning may enable autonomous strategies, transforming the model into a "virtual coach."

Beyond sports, this approach has applications in collaborative robotics, crisis management, and intelligent logistics. By redefining adaptation in dynamic systems, this work highlights the growing convergence between human strategy and artificial intelligence, expanding the boundaries of tactical innovation.

References

1. FAETERJ Petrópolis: Faeterj petrópolis. https://www.faeterj-petropolis.edu.br/
2. Moreira, L.M., Rosa, J.F.d., Gomes, F., Ferreira, M.N.G.S., Silva, E.K.d., Angonese, A.T.: Sir2d 2014: Descrição do time - larc 2014. Team Description Paper, São Carlos (2014)
3. Moura, A.: A proposal for 2D Delaunay triangulation and planar point localization in OCaml. Ph.D. thesis, Universidade Federal de Uberlândia (2006)
4. RC Soccer Simulation: Robocup soccer simulation documentation. https://rcsoccersim.readthedocs.io/en/latest/index.html
5. RoboCup: Soccer simulation 2d. https://ssim.robocup.org/soccer-simulation-2d/
6. SiRLAB: Sirdrigons2d. https://github.com/SIRLab/SirDrigons2D
7. SiRLAB: Sirlab - simulation and intelligent robotics laboratory. https://sirlab.github.io/index.html
8. Zare, N., Amini, O., Sayareh, A., Sarvmaili, M., Firouzkouhi, A., Rad, S., Matwin, S., Soares, A.: Cyrus2d base: Source code base for the robocup 2d soccer simulation league. In: RoboCup 2022: Robot World Cup XXV, pp. 140–151. Springer International Publishing (2023)