Robot Soccer - Strategy Description and Game Analysis

Jan Martinovič, Václav Snášel, Eliška Ochodková
Lucie Žołtá, Jie Wu,
VŠB - Technical University of Ostrava,
FEECS, Department of Computer Science
Ostrava, Czech Republic
Email:jan.martinovic@vsb.cz, vaclav.snasel@vsb.cz,
eliska.ochodkova@vsb.cz, l.zolta@seznam.cz,
defermat2008@hotmail.com

Ajith Abraham
Machine Intelligence Research Labs - MIR Labs,
USA, http://www.mirlabs.org
ajith.abraham@ieee.org

KEYWORDS

Robot Soccer, Cluster Analysis, Strategy, Prediction

ABSTRACT

The robot soccer game, as a part of standard applications of distributed system control in real time, provides numerous opportunities for the application of AI. Real-time dynamic strategy description and strategy learning possibility based on game observation are important to discover opponent's strategies, search tactical group movements and synthesize proper counter-strategies. In this paper, the game is separated into physical part and logical part including strategy level and abstract level. Correspondingly, the game strategy description and prediction of ball motion are built up. The way to use this description, such as learning rules and adapting team strategies to every single opponent, is also discussed. Cluster analysis is used to validate the strategy extraction.

INTRODUCTION

A typical example of a distributed control system with embedded subsystems is the control of robot soccer game (FIRA, 2010). The game can be described as double eleven autonomous mobile robots (home and visiting players), which are situated at the field with size of 220×180cm. Robot soccer is a challenging platform for multi-agent research, including real-time image processing and control, path planning, obstacle avoidance and machine learning. The robot soccer game presents an uncertain and dynamic environment for cooperating agents. (Bezek, 2005; Bezek et al., 2006; Tucnik et al., 2006) describe multi-agent strategic modeling of robot soccer. (Berger and Lämmel, 2007; Fard et al., 2007) give a very broad overview for using Case Based Reasoning techniques for this challenging problem.

Dynamic role switching and formation control are crucial for a successful game (Sng et al., 2002). The entire game can be divided into a number of partial tasks (Kim et al., 2004; Stankevich et al., 2005), such as evaluation of visual information, image processing, hardware and software implementation of distributed control system, hard-wired or wireless data transmission, information processing, strategy planning and control of robots. Complex control tasks can often be solved by decomposing them into hierarchies of manageable subtasks (Whiteson et al., 2005).

Because of the attraction of robot soccer, many interesting approaches were developed to improve robot soccer. An abstract description of the game was developed in (Obitko and Snasel, 2004; Smid et al., 2004; Srovnal et al., 2007), together with the ways to use this description. In (Zhao et al., 2006), it discussed the strategy based on opponent information in the Robot-Soccer Game. We adopted this representation to control robots and learn game strategies.

In our approach, the game is separated into logical and physical parts. The logical part includes the strategy selection, calculation of robot movement and adaptation of rules to the opponent's strategy. The physical part includes robot actual movement on the game field and recognition of the opponent movement. The logical part is independent of the physical part because we can calculate movements of the opponent robots as well as movements of own robots.

By separating the game into two parts, the logical part is independent of the field size and the resolution of the camera used in visual information system. In the logical part, the game is represented as an abstract grid with a very high resolution, which ensures a very precise position of robots and ball. However, this very detailed representation of the game field is not suitable for strategy description. Too many rules are required to describe robot behavior. Therefore, a strategy grid, which has a much lower resolution than an abstract grid, is used. This simplification of reality is sufficient because it is not necessary to know the exact position of robot. It is sufficient to know its approximate position for strategy realization (Fig. 1). When the physical part is used, we only need to transform the coordinates from the abstract grid into coordinates based on the game field size and camera resolution.

The rest of the paper is organized as follows. Section 2 describes possible game strategies. Using the abstract grids and game strategies, it is explained how to learn rules that describes specific game strategies. Our

Proceedings 24th European Conference on Modelling and Simulation ©ECMS Andrzej Bargiela, Sayed Azam Ali David Crowley, Eugène J.H. Kerckhoffs (Editors)
ISBN: 978-0-9564944-0-5 / ISBN: 978-0-9564944-1-2 (CD)

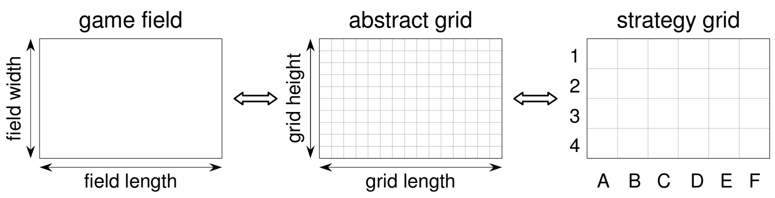

Figure 1: Inner game representation.

approach to predict the movement of ball is shown in section 3. It is based on the grids, respectively on strategy level and abstract level, which is helpful to determine the game strategy and control the robot formation. Section 4 discusses the strategy extraction based on cluster analysis. Section 5 concludes with the discussion of the presented approach.

GAME STRATEGY

The game strategy can be dynamically changed based on game progress (i.e. the history and the current position of the players and the ball (Veloso and Stone, 1998)).

In this section, we describe our approach for learning game strategy from observation. Similar approach we can see in (Ros and Arcos, 2007; Ros et al., 2007). Our objective is to learn an abstract strategy. The main steps of the learning process are:

- Transformation of observations into abstract grids.
- Transformation of observations into strategy grids.
- Learning a strategy based on the observed transitions in the strategy grid.

The movement of a particular robot is determined by the current game class and situation, and by the robot's role. For example, the goalkeeper's task is to prevent the opponent from scoring a goal. In most cases, its movements are limited along the penalty zone and near the goal line. The preferred movements are in the direction of the goal line. The goalkeeper's movements ensure that the ball will be kicked from the defence zone.

We adopt this definition of strategy (Johnson and Scholes, 2001): "Strategy is the direction and scope of an organization over the long-term, which achieves advantage for the organization through its configuration of resources within a challenging environment..."

Strategy application for one movement of players is computed in following steps:

1. Get coordinates of players and ball from camera
2. Convert coordinates of players into strategic grid
3. Convert ball and opponents' positions into abstract and strategic grids
4. Choose goalkeeper and attacker, exclude them from strategy and calculate their exact positions.
5. Detect strategic rule from opponents' and ball positions
6. Convert movement from strategic grid to physical coordinates
7. Send movement coordinates to robots

Each strategy is stored in one file and currently consists of about 15 basic rules.

```
.Strategy "test"
.Algorithm "Offensive"
.Author "Vaclav Snasel"
.Date "19.12.2008"
.Size 6 4
.PriorityMine      100 100 100 100 100
.PriorityOpponent  50   50  50  50  50
.PriorityBall      50

.Rule 1 "Attack1"
.Mine c2  c3  b1  b4
.Opponent d2  d3  e1  e4
.Ball d3
.Move d2  c3  b1  b4

.Rule 2 "Attack2"
.Mine d2  c3  b1  b4
.Opponent d2  d3  e1  e4
.Ball c3
.Move d2  c4  c1  b4

.Rule 3 "Attack3"
.Mine d2  c4  c1  b4
.Opponent d2  d3  e1  e4
.Ball c3
.Move d2  c4  b2  b3
```

Furthermore the file contains following metadata:

- Information about the name of strategy
- The algorithm to strategy choosing
- The author responsible for current strategy
- The date of last modification
- The size of strategic grid (e.g. Fig. 1)
- Strategic rules

Each strategic rule consists of five records:

- The rule ID and description (e.g. Rule 1 "Attack1"),
- the coordinates of our players in strategic grid (e.g. .Mine c2 c3 b1 b4),
- the coordinates of opponent's players in strategic or abstract grid (e.g. .Opponent d2 d3 e1 e4),

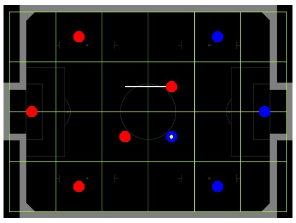
Figure 2: Rule Attack1.

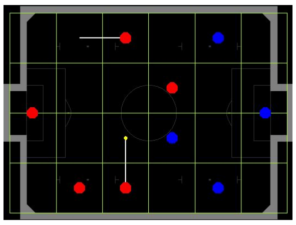
Figure 3: Rule Attack2.

- the ball coordinates in abstract or strategic grid (e.g. .Ball d3)
- strategic or abstract grid positions of the move (e.g. .Move d2 c3 b1 b4).

```
// algorithm for rule selection
// Game.Mine       -- actual positions
// Game.Opponent   -- actual positions
// Game.Ball       -- actual position
maxWeight = 0
SelectRule = 0
foreach r in Rule
{
  ruleTmp = r.Mine
  SumToMine = 0
  foreach p in Game.Mine
  {
    s = nearest position in ruleTmp to p
    w = 1 / (distance(s, p) + 1)
    w = Strategy.PriorityMine * w
    SumToMine = SumToMine + w
    remove s from ruleTmp
  }
  weight = SumToMine

  ruleTmp = r.Opponent
  SumToOpponent = 0
  foreach p in Game.Opponent
  {
    s = nearest position in ruleTmp to p
    w = 1 / (distance(s, p) + 1)
    w = Strategy.PriorityOpponent * w
    SumToOpponent = SumToOpponent + w
    remove s from ruleTmp
  }
  weight = weight + SumToOpponent

  ToBall = 1 / (distance(Game.Ball, r.Ball) + 1)
  ToBall = Strategy.PriorityBall * w
  weight =  weight + ToBall
  if weight > maxWeight
  {
    maxWeight = weight
    SelectRule = r
  }
}
return SelectRule
```

A basic strategy item is a movement in strategy grid. An example may be the following rule:

if $(M_1, M_2, M_3, M_4, M_5)$ is close to $(c2, c3, b1, b4)$
and $(O_1, O_2, O_3, O_4, O_5)$ is close to $(d2, d3, e1, e4)$
and B is close to $(d3)$
then $(M_1, M_2, M_3, M_4, M_5)$ "go to" $(d2, c3, b1, b4)$

By observing opponent's strategy, a new set of rules can be written without the necessity of a program code modification. Furthermore, there is a possibility of automatic strategy (movement) extraction from the game in progress. There are two main criteria in the selection process rules. The selection depends on opponent coordinates, own coordinates and ball position. The strategy file contains rules, describing three possible formations suggesting danger of the current game situation. The opponent's team can be in offensive, neutral or defensive formations. Furthermore, we need to consider ball position risk. Generally, opponent is not dangerous if the ball is near his goal. The chosen rule has minimal strategic grid distance from the current rule.

Optimal movements of our robots are calculated by applying a minimal distance from strategic grid position. The goalkeeper and attacking player, whose distance is closest to the ball, are excluded from strategic movement and their new position is calculated in exact coordinates. To summarize, strategy management can be described in the following way:

- Based on incoming data from the vision system, calculate abstract and strategy grid Coordinates of the players and the ball.
- The abstract grid is then used to decide which player has under the ball control.
- This player is issued a "kick to" command that means that it has to try to kick the ball to a given strategy grid coordinates.
- All other players are given (imprecise) "go to" coordinates. These coordinates are determined by the current game strategy and are determined for each robot individually. The goalkeeper is excluded from this process since its job is specialized, and does not directly depend on the current game strategy.

PREDICTION OF BALL MOTION

Obviously, the prediction of ball motion is very important to the robot soccer. The prediction includes ball track, collision and rebound. Strategically, it is helpful to select proper strategy and regulate robot formation. Tactically, it is contributing to break through, intercept or steal the ball at right position right moment. To sum up, the prediction is a necessary step for defensive and offensive in robot soccer.

In the game representation, there are two kinds of grids, strategy grid and abstract grid. The strategy grid exists in our representation because it is not necessary to know the robot exact position in strategy hierarchy. Similarly, the strategy grid can be used to predict the ball motion in strategy level, because it is enough to predict the ball approximate position for strategy realization. On the other hand, for the prediction, the abstract grid is necessary to ensure the precision. Therefore, the ball motion should be predict on two levels – strategy and abstract.

PREDICTION ON STRATEGY LEVEL

About the prediction, it could be very simple if the ball velocity is known. However, for the ball motion prediction on strategy level, it is not necessary to predict the ball exact position and we just need to predict the ball approximate position at next frame in the strategy grid. Therefore, this prediction could be further simplified.

Figure 4 shows the ball lying in a strategy grid. The velocity of ball can be decomposed into x-component and y-component. In Fig. 4, there are eight potential positions for the ball at next moment. But the final position depends on the relative magnitude of x-component and y-component. That means we just need to compare those two velocity components with each other, then the new position of ball can be predicted. For the case shown in Fig. 4, because the x-component is much greater than y-component, the ball will move into the left grid at next moment. By this means, the prediction is further simplified to boolean operation, which would reduce the operation time. It's very important to the strategy selection.

In addition, there are two noticeable cases in strategy level prediction. The first one is the ball lies close to boundary, which means potential collision and rebound to the wall. In this case, the number of possible new positions is less than eight, so it's simpler again. The second one is the ball is likely to collide with robot during movement. Regardless of which party robot is, the robot will certainly catch the ball. Therefore, in this case, the ball will lie in the grid in which the colliding robot lies.

PREDICTION ON ABSTRACT LEVEL

When the ball is caught by robot, the motion track of ball is under the control of robot, and in this case, it's not necessary to predict. When the ball is free, the ball would move along a line. Generally, the ball is usually under the control of robot. That means the ball would move freely only for a short time, so it's reasonable to neglect the acceleration, then the motion track can be depicted by simple linear model.

For the prediction on abstract level, it's necessary to predict a very precise position of ball, so the grid must be presented with a very high resolution. To solve the new position of ball, coordinate system should be built according to the grid (see Fig. 5). The track equation of ball can be expressed as

$$\begin{cases} x &= x_0 + v_x t \\ y &= y_0 + v_y t \end{cases}$$

where x and y are respectively predictive coordinates, x_0 and y_0 are initial coordinates, v_x and v_y are two components of ball velocity. The initial coordinate should be

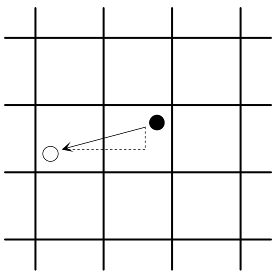

Figure 4: Ball in strategy grid.

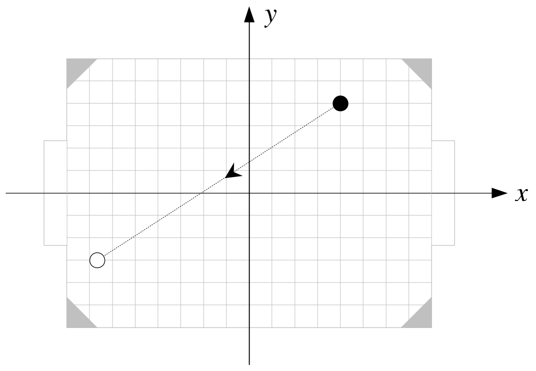

Figure 5: Ball in abstract grid.

re-read from visual information system at the moment of every touch, collision or rebound to something before the ball move freely again.

Rebound track is very important to the prediction of ball motion. In fact, we developed a method to calculate the detailed position of ball, but we have to simplify it because of the time urgency. The rebound mentioned here only refers to the wall rather than to the robot, because the robot will catch the ball and the rebound to robot would happened scarcely.

Given the collision to the wall is elastic collision, so the post-rebound velocity has the same magnitude to the pre-rebound velocity except orientation. In other words, the velocity could be decomposed into two quadrature-components, the new velocity could be gotten by sign-changing one component while maintain another one. According to the new velocity, the ball rebound track could be predicted. For example, if the ball move forward along the line shown in Fig. 5, it will collide to the left of the wall and then rebound. After rebound, the ball track equation can be expressed as

$$\begin{cases} x &= x_0' - v_x t \\ y &= y_0' + v_y t \end{cases}$$

GAME ANALYSIS

A log, which describes in detail the course of the game, is created during the game. Log contains hundreds of lines to describe each game situation (my robots and opponent's robots positions, ball position, what rule was used, etc.). Records in the game log, which can be seen in Example 1, contain real and grid positions. We obtain more than a thousand such records (depending on the performance of your computer) during one game.

Example 1

Entry (one line) from the log-file (starting position)

$14; 0; 0; 4; 3; -36,67; -30; 3; 2; -36,67; 30; 3; 3;$
$-73,33; -60; 2; 1; -73,33; 60; 2; 4; -105; 0; 1; 3;$
$36,67; -30; 4; 2; 36,67; 30; 4; 3; 73,33; -60; 5; 1;$
$73,33; 60; 5; 4; 105; 0; 6; 3;$

■

By analyzing this log, we can derive the opponent's strategy or optimize our strategy. First, it is necessary to reduce the cardinality of logs to a "reasonable number" of representative situations (records). To reduce records, we used clustering using minimum spanning tree. It was therefore necessary to create a graph in which vertices represent records in the log (game situation) and edges represent similarities between them.

One question is how to compare two game situations. Therefore, each record was transferred to a matrix (map), which is a simplified description of the game situation. Each matrix in this new log consists of three maps of the course in the grid coordinate system (see Table 1).

The first part of the matrix shows the field distributed grid in the ratio 4×6 and the position of the ball. Second part (six columns) represents my positions and last six columns represent opponent's positions. Number one in the matrix represents the presence of the object (ball, my robot, opponent robot).
Thus described game situation can be better compared by e.g. the following relationship used in Equation 1, which sums up the differences between the various elements of the matrix. This method is used here only as an illustrative method because it is not completely accurate, since only considers whether or not there was a change (Example 2).

$$\sum_{i,j=0}^{i\leq r, j\leq s} |A[i,j] - B[i,j]| \,, \tag{1}$$

where A, B are compared matrices, r is the number of rows and s number of columns.

Example 2

Comparison of three situations (matrix simplified to vector)

$$\begin{array}{rcccccc} situation_1: & 0 & 0 & 0 & 1 & 0 & 0, \\ situation_2: & 0 & 0 & 1 & 0 & 0 & 0, \\ situation_3: & 1 & 0 & 0 & 0 & 0 & 0, \end{array}$$

$situation_1 - situation_2 = |0-1| + |1-0| = 2,$
$situation_1 - situation_3 = |0-1| + |1-0| = 2.$

In both cases, there was a change toward the $situation_1$. Change is, however, expressed by the same coefficient of similarity, 2, although the $situation_3$ differs from the $situation_1$ more than the $situation_2$ differs from the $situation_1$. ■

For a more precise comparison, the binary matrix was transferred to a matrix "with less contrast"'. Elements with value 1 were increased by a constant c and their vicinity by the value $(c-1)$ etc. (see Table 2 for $c = 1$).

It is now more accurate comparison, as we can see in Example 3. The difference between $situation_1$ and $situation_2$ is less than the difference between $situation_1$ and $situation_3$ now.

Example 3

Comparison of three situations (matrix simplified to vector)

$$\begin{array}{rcccccc} situation_1: & 0 & 0 & 1 & 2 & 1 & 0, \\ situation_2: & 0 & 1 & 2 & 1 & 0 & 0, \\ situation_3: & 2 & 1 & 0 & 0 & 0 & 0, \end{array}$$

$situation_1 - situation_2 = |0-1| + |1-2| + |2-1| + |1-0| = 4,$
$situation_1 - situation_3 = |0-2| + |0-1| + |1-0| + |2-0| + |1-0| = 7.$ ■

Now it was possible to reduce the quantity of records. As mentioned above, we will work with complete graph with vertices representing records and edges representing similarities between them (edges are valued with a number indicating the similarity (the more similar the smaller value and 0 means identical records)). Identical records were written out when creating the graph. From this graph the minimum spanning tree was obtained with the Kruskal's algorithm and by removing the most expensive edges from the spanning tree clusters were created. As a representative element (we named it as centroid) of the cluster, we selected the vertex with the highest degree. The resulting centroids will be used to develop estimates of the opponent's strategy. Each centroid contains information about ball, of my players and opponents. In order to generate the strategy of the opponent, we need to add the following information about shift in the strategic move. This shift is obtained from the original log. In it we find the row i, which represents the selected centroid. Shift can be determined from the $i + k$-th following line, where k will define the jump in the log.

CONCLUSION

The main goal of the control system is to enable a real-time response. The method we described provides fast control. This is achieved by using rules those are fast to process. We have described a method of game representation and a method of learning game strategies from observed movements. The movements can be observed from the opponent's behavior or from the human player's behavior. We believe that the possibility of learning the game strategy that leads to a fast control is critical for

Table 1: Matrix (starting position): position of the ball - of my robots - of the opponent robots

0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0

Table 2: The revised matrix for $c = 1$ (starting position)

0	0	0	0	0	0	1	3	2	1	0	0	0	0	1	2	3	1
0	0	1	1	1	0	2	4	4	2	0	0	0	0	2	4	4	2
0	0	1	2	1	0	3	4	4	2	0	0	0	0	2	4	4	3
0	0	1	1	1	0	2	4	2	1	0	0	0	0	1	2	4	2

success of robot soccer game. Likely to the chess playing program, the database of successful game strategies can be stored in the database and can be used for subsequent matches (Sadikov and Bratko, 2006).

ACKNOWLEDGEMENT

We acknowledge the support of project SP/2010196 Machine Intelligence.

REFERENCES

Barrios-Aranibar, D. and Alsina, P. J. (2005). *Recognizing Behaviors Patterns in a Micro Robot Soccer Game.* HIS IEEE, Pages: 463–468

Berger, R. and Lämmel, G. (2007). *Exploiting Past Experience Case-Based Decision Support for Soccer Agents.* KI 2007: Advances in Artificial Intelligence, LNCS 4667, Pages: 440–443.

Berry, M. W. and Browne, M. (1999). *Understanding Search Engines.* SIAM Society for Industrial and Applied Mathematics, Philadelphia.

Bezek, A. (2005). *Discovering strategic multi-agent behavior in a robotic soccer domain.* AAMAS 2005: Pages: 1177–1178.

Bezek, A., Gams, M. and Bratko, I. (2006). *Multi-agent strategic modeling in a robotic soccer domain.* AAMAS 2006: Pages: 457–464.

Fard, A. M., Salmani, V., Naghibzadeh, M., Nejad, S. K. and Ahmadi H. (2007). *Game Theory-based Data Mining Technique for Strategy Making of a Soccer Simulation Coach Agent.* ISTA 2007: Pages: 54–65.

FIRA robot soccer, http://www.fira.net/ (01-04-2010)

Johnson, G. and Scholes, K. (2001). *Exploring Corporate Strategy: Text and Cases.* FT Prentice Hall.

Kim, J., Kim, D., Kim, Y. and Seow, K. (2004). *Soccer Robotics.* Springer Tracts in Advanced Robotics, Springer-Verlag.

Obitko, M. and Snášel, V. (2004). Ontology Repository in Multi-Agent System. Artificial Intelligence and Applications - Volume I & II. Calgary: Acta Press, vol. 1, Pages: 853–858.

Ros, R. and Arcos, J. L. (2007). Acquiring a Robust Case Base for the Robot Soccer Domain. IJCAI 2007: Pages: 1029–1034.

Ros, R., de Mántaras, R. L., Arcos, J. L. and Veloso, M. M. (2007). *Team Playing Behavior in Robot Soccer: A Case-Based Reasoning Approach.* ICCBR 2007: Pages: 46–60.

Sadikov, A. and Bratko, I. (2006). *Learning long-term chess strategies from databases.* Machine Learning (2006) 63:3 Pages: 329–340.

Smid, J., Obitko, M. and Snášel V. (2004). *Communicating Agents and Property-Based Types versus Objects.* In SOFSEM 2004 - Theory and Practice of Computer Science. Prague: MATFYSPRESS, Pages: 154–163.

Sng, H.L., Gupta, G.S. and Messom, C.H. (2002). *Strategy for Collaboration in Robot Soccer.* The First IEEE International Workshop on Electronic Design, Test and Applications, Pages: 347–251.

Srovnal, V. , Horák, B. , Snášel, V., Martinovič, J., Krömer, P. and Platoš, J. (2007). *Strategy Description for Mobile Embedded Control Systems Exploiting the Multi-agent Technology.* International Conference on Computational Science (2), Pages: 936–943.

Stankevich, L., Serebryakov, S. and Ivanov, A. (2005). *Data Mining Techniques for RoboCup Soccer Agents.* AIS-ADM 2005: Pages: 289–301.

Tučník, P., Kožaný, J. and Srovnal, V. (2006). *Multicriterial Decision-Making in Multiagent Systems.* Computational Science ICCS 2006, LNCS 3993, Pages: 711–718.

Veloso, M. and Stone, P. (1998). *Individual and collaborative Behaviours in a Team of Homogeneous Robotic Soccer Agents.* Proceedings of International Conference on Multi-Agent Systems, 1998, Pages: 309–316.

Zhao, X., Zhang, J., Li, W. and Li, Y. (2006). *Research on Strategy of Robot Soccer Game Based on Opponent Information.* IEEE Machine Learning and Cybernetics, Pages: 230–234.

Whiteson, S., Kohl, N., Miikkulainen, R. and Stone, P. (2005). Evolving Soccer Keepaway Players Through Task Decomposition. Machine Learning, Volume 59:1 Pages: 5–30.