ARTICLE IN PRESS

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Analysis of strategy in robot soccer game

Jie Wu [a,*], Václav Snášel [a], Eliška Ochodková [a], Jan Martinovič [a], Václav Svatoň [a], Ajith Abraham [a,b]

[a] *Department of Computer Science, FEECS, VŠB - Technical University of Ostrava, 708 33 Ostrava-Poruba, Czech Republic*
[b] *Machine Intelligence Research Labs, Auburn, WA 98071, USA*

A R T I C L E I N F O

Keywords:
Robot soccer
Strategy

A B S T R A C T

Strategy is a kernel subsystem of robot soccer game. In our work, we present an approach to describe the strategy of the game, based on which we explain the morphology of strategy set. Loop strategies are likely to make robots be in a trap of executing repeated actions. We analyze the existence criterion of loop strategies, and then present some corollaries and theorems, by which the loop strategies and chain strategies can be found, also superfluous strategies and inconsistent strategies. We present a ranking model that indicates the weak node in strategy set. We also present a probability-based model which is the basis of evaluation of strategy. Additionally, we present a method to generate offensive strategy, and the statistic results of simulation game prove the validity of the method.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Robot soccer game is an emerging field that combines artificial intelligence and mobile robotics with the popular sport of soccer. Robot soccer can be portrayed as a competition of advanced robot technology within a confined space. It offers a challenging arena to the young generation and researchers working with autonomous mobile robotic systems.

There are two biggest organizations in the field of robot soccer, namely FIRA and RoboCup. They have been actively promoting the development of soccer robotics and education in the field of artificial intelligence. Robot soccer not only provides an experimentation test bed for multi-agent robotic systems research, but also a useful education tool for practical courses in this area. It is helpful to knowledge integrating, teamwork, real world issues, critical thinking and creativity.

Particularly, in robot soccer collaboration is desired so that the group of robots work together to achieve a common goal [1]. In robot soccer, the game situation in the playground is typically read in terms of the robots' postures and the ball's position. Naturally, good strategies are needed to decide the positions of the team robots during the game.

In this work, we are particularly interested in the evaluation of strategy in robot soccer game. Description of strategy is the basis of the robot soccer game. Based on the strategic description we can set up strategy set. In strategy set there are many types of strategies, such as isolated strategies, chain strategies and loop strategies. Loop strategies could be a problem for robot soccer game. We present the existence criterion and detection method of loop strategies. We also present a ranking model to find the most important strategy that could be a weak node in the strategy set. And we present a probability-based model to estimate the probability of goal by which we can evaluate strategies. Additionally, we present a method to generate offensive strategy, and our experiment prove the validity of the method.

The paper is organized as follows. We review related work within the robot soccer domain in Section 2. Section 3 introduces our approach to describe strategy. Section 4 discusses the morphology of strategy set. Section 5 focuses on the problem of loop strategies, including existence criterion and detection of loop strategies. Section 6 proposes a method to rank strategies by which we can find the weak nodes in strategy set. Section 7 presents a model to evaluate strategies. Section 8 presents a method to generate offensive strategy. Finally, Section 9 draws the conclusions and discusses future work.

2. Related work

There are many techniques have been applied to decision-making of robot soccer game, including Case-based Reasoning, Learning from Observation, Reinforcement Learning, Pattern Recognition, Fuzzy Theory, Neural Network, Evolutionary Algorithm, Decision Tree, etc. Most of them are on the level of tactic which cannot answer the question that how good are the strategies.

Case-based Reasoning (CBR) [2,3] is a family of artificial intelligence techniques, based on human problem solving, in

* Corresponding author.
E-mail addresses: defermat2008@hotmail.com (J. Wu), vaclav.snasel@vsb.cz (V. Snášel), eliska.ochodkova@vsb.cz (E. Ochodková), jan.martinovic@vsb.cz (J. Martinovič), vaclav.svaton@vsb.cz (V. Svatoň), ajith.abraham@ieee.org (A. Abraham).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.03.021

ARTICLE IN PRESS

which new problems are solved by recalling and adapting the solutions of similar past problems. Given a new situation, the most similar past case is retrieved and its solution is reused after some adaptation process to match the current situation. Ros et al. [4] applied CBR to model the action selection of a team of robots within the robot soccer domain. Cases model the state of the world at a given time and prescribe a "successful" action. A case can be treated as a recipe that describes the state of the problem description and the actions to perform in that solution description. In fact, CBR could be an experience-based system, it depends on how good the experience is; however, CBR cannot evaluate how successful experience it has. If the solution of similar past problem is not good enough, the outcome might be unsatisfied.

Learning from Observation (LFO) aims at modeling agents that learn from observing other agents and imitating their behavior. As in CBR, the learning agent selects the most similar past observed situation with respect to the current problem and then reproduces the solution performed at that time. Lam et al. [5] developed an agent behavior model based on scene recognition in the Simulation League. The main difference between CBR and LFO is that the learning agent is not able to improve the observed agent since there is no feedback in the model. Similar to CBR, LFO cannot evaluate the quality of the behavior.

Reinforcement Learning (RL) [6] is an area of machine learning that is concerned with how an agent ought to take actions in an environment so as to maximize some notion of cumulative reward. Riedmiller et al. [7] proposed an approach that applies RL to robot soccer on tactical level, such as the moves of intercept-ball, wait at position, pass ball to teammate, shoot to goal, etc. Kleiner et al. [8] applied RL to the Middle-Size League of robot soccer game on skill level, such as search-ball, approach-ball, shoot-ball, shoot-away, and so on. All these works are focused on the robotic collaboration on the level of skills and action selection.

Pattern Recognition has been used to solve the opponent modeling problem in the Simulation League. Huang et al. [9] presented an approach that, combining plan representation, plan recognition and retrieval techniques, translates the multi-variable information stream obtained from the observation of the dynamic and adversarial world into streams of agents' behaviors using prediction and backfill techniques. Consequently, frequent behavior sequences or patterns can be found by statistical dependency test, then relevant plans can be extracted. Lattner et al. [10] and Miene et al. [11] proposed an approach that applies unsupervised symbolic off-line learning to a qualitative abstraction in order to create frequent patterns in dynamic scenes.

Fuzzy Theory is another selected approach in robot soccer game. Lee et al. [12] presented a tactics using fuzzy logic mediator that selects proper robot action depending on the positions and the roles of adjacent two robots. Their work is implemented and tested by SimuroSot. Wu and Lee [13] focused their research on the selection of five action categories within the Small-Size League. Given a set of input variables, the output of the fuzzy rules indicates the action to perform by the robot. The approach only considers a single player and therefore, no cooperation can be considered.

Neural Networks (NN) are extensively used in the field of intelligent multi-agent systems. Jolly et al. [14] presented a complete work in which a two-stage approach using NN for action selection in the MiroSot League is proposed. More precisely, their work is focused on deciding which of the two robots near the ball must go after it while the other remains as a supporter.

Evolutionary Algorithms (EA) have been proposed in several occasions within the RoboCup domain. Nakashima et al. [15] proposed a method for acquiring team strategies in the Simulation League. They employed an ES-type generation update scheme after producing new integer strings by using crossover and mutation. They use the scores of the soccer games as performance measure in their evolutionary method. Park et al. [16] used evolutionary algorithms to determine the appropriate fuzzy control rules for the path planning problem in robot soccer.

Konur et al. [17] chose decision-tree learning technique to decide the next action in robot soccer game of Simulated League. However, during gathering the training data, the supervisor should have a good idea of how soccer is played in order to give advice to the agent. The decision-tree learning could not solve this problem.

3. Description of strategy

Description of strategy is the basis of the robot soccer game. Many different forms of strategic description have been developed to support corresponding decision-making system. Bezek et al. [18,19] propose a multi-agent strategic modeling of robot soccer. In their work, the strategy is described as a combination of agent roles, node positions and sequence of actions. Node positions correspond to agent positions in the game field that is divided into several areas, such as left-wing, center-of-the-field, right-wing, left-half, right-half, and so on. Huang and Liang [20] bring forward a strategy-based decision-making system for robot soccer game which implemented by a decision tree. In the tree, there are 12 strategies that are clustered by ball's position, possession of ball, team players, etc. In [4], a case is defined by (P, A, K), where P is the problem description, A the solution description, and K the case scope representation. The problem description corresponds to a set of features, including ball's global position, defending goal, teammates' global positions, and opponents' global positions, which describe the current world state. The solution of a case corresponds to the sequences of actions each teammate performs, such as "get the ball", "kick", or "pass ball to robot t_{mi}", and so on. All of these approaches mentioned above contain two common elements, i.e. grid positions and possession of ball. These common elements are also reflected in our description of strategy.

In our work, the game is separated into logical and physical parts [21]. 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 precise positions of robots and ball. However, this very detailed representation of the game field is not suitable for strategy description. Otherwise, the similar situations of the game might be treated as different ones. Therefore, the strategy grid is used, which has a much lower resolution than the abstract grid. This simplification is sufficient because it is unnecessary to know the robots precise position in the logical part. It is enough to know the robot's approximate position for strategy realization (see Fig. 1). When the physical part is used, based on the game field size and camera resolution, we only need to transform the abstract grid into physical coordinates.

For the robot soccer game, the strategies can be treated as a series of actions under some certain conditions that may contain the information of position, velocity, acceleration, or posture of

ARTICLE IN PRESS

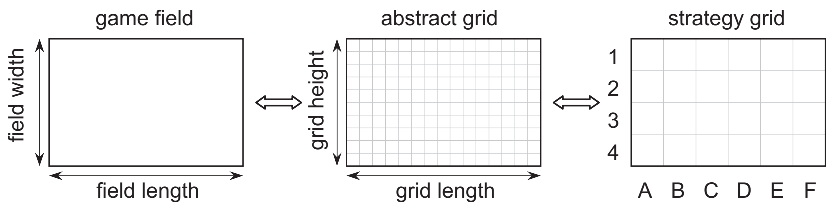


Fig. 1. Inner game representation.

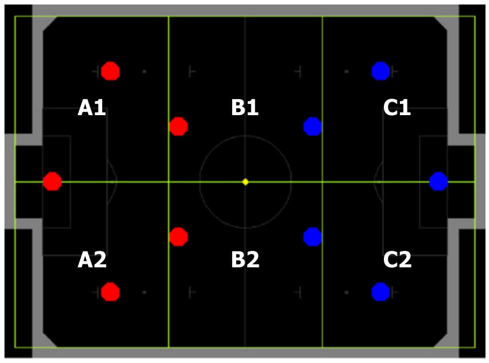


Fig. 2. An example case.

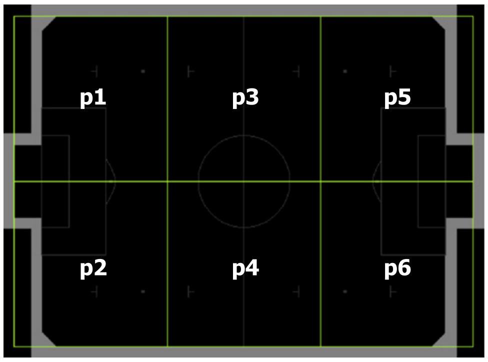


Fig. 3. A strategy grid.

robots, and so on. The position information, representing the situation of both sides, is much more important than the others in the logic part. Abstract grid, velocity, acceleration and posture of robots are associated with motion planning and robot control [22,23]. Apparently, motion planning and robot control are the implementation of strategies, while the strategies should be made through game situation. Therefore we express the game situation by position information [24,25].

Example 1 (*Family a strategy*). According to strategy grid, the strategy can be expressed easily as (*M, O, B, D*), where *M* is the teammates' positions of *mine*, *O*, *opponents*' positions, *B*, *ball* position, and *D*, my teammates' *destination* grids.

- *Mine* *A*1 *A*2 *B*1 *B*2;
- *Oppo* *B*1 *B*2 *C*1 *C*2;
- *Ball* *B*2;
- *Dstn* *A*1 *B*2 *C*1 *C*2.

Example 1 shows a strategy stored in a log file, which corresponds to the case shown in Fig. 2. It means "**If** (M_1,M_2,M_3,M_4) is close to (*A*1, *A*2, *B*1, *B*2), **and if** (O_1, O_2, O_3, O_4) is close to (*B*1, *B*2, *C*1, *C*2), **and if** *B* is close to (*B*2), **then** (M_1, M_2, M_3, M_4) go to (*A*1, *B*2, *C*1, *C*2)". In the strategy, the first part (*M, O, B*) is *condition attributes*, the latter part *D* is *decision attribute*. Now if we represent the grid position by using digital coordinates, the strategy in Example 1 can be denoted as follows.

- *Mine* 11 12 21 22;
- *Oppo* 21 22 31 32;
- *Ball* 22;
- *Dstn* 11 22 31 32.

In the field of strategy, there are two types of features, controllable features and non-controllable features [4]. Teammates' positions are controllable features because the robots can move to more appropriate locations if needed. The ball's and opponents' positions are non-controllable features because we cannot directly modify them. The idea of separating the features into controllable and non-controllable ones is that the two types of features have different significance at different moment. If the ball is under control, the controllable features are more important. However, if the ball is out of control, the non-controllable features are more important. It means that if the ball is under the control of our robot, we only need to consider how to control our robots to next positions, it is unnecessary to care the opponents' positions anymore; while if the ball is controlled by opponent robot, we have to control our robot to catch the ball.

Here we ignore the ball position when we analyze the strategy set, because the ball position must be same to one of our or opponent robots' position. By the control tag of the ball, the strategies can be divided into two sets which correspond to attack and defence respectively. It is easy to switch the team's state between attack and defence in this way.

Consequently, according to the strategy grid in Fig. 3, the strategy of Example 1 can be simplified as (111100100111) that means "**if** *M* is close to (*p*1, *p*2, *p*3, *p*4) **then** *M* goes to (*p*1, *p*4, *p*5, *p*6)". Here *M* means (M_1, M_2, M_3, M_4).

Example 2. Based on the strategy grid shown in Fig. 3, we have the following rules. These rules can be denoted as digital numbers in Table 1, in which the digital "1" means there exists only one robot in the corresponding grid, while the digital "0" means there is not any robot in the grid.

Rule 1: If *M* is close to (*p*1 *p*2 *p*3 *p*4), then *M* goes to (*p*1 *p*4 *p*5 *p*6).
Rule 2: If *M* is close to (*p*1 *p*2 *p*5 *p*6), then *M* goes to (*p*1 *p*2 *p*4 *p*5).
Rule 3: If *M* is close to (*p*3 *p*4 *p*5 *p*6), then *M* goes to (*p*1 *p*2 *p*5 *p*6).
Rule 4: If *O* is close to (*p*2 *p*3 *p*4 *p*6), then *M* goes to (*p*1 *p*2 *p*2 *p*4).

Clearly, our strategic description simplifies the complexity of strategy, reduces the cardinal number of strategy set greatly. If we consider all controllable and non-controllable features in our strategy, there would be about 6^9 strategies in the case of 3×2

ARTICLE IN PRESS

Table 1
Strategy description of rules in Example 2.

Rules	*p*1	*p*2	*p*3	*p*4	*p*5	*p*6	*Destination*
*rule*1	1	1	1	1	0	0	100111
*rule*2	1	1	0	0	1	1	110110
*rule*3	0	0	1	1	1	1	110011
*rule*4	0	1	1	1	0	1	120100

Table 2
A decision table.

U	c_1	c_2	c_3	c_4	c_5	c_6	d_1	d_2	d_3	d_4	d_5	d_6
x_1	0	0	1	1	1	1	0	1	0	1	1	1
x_2	0	1	0	1	1	1	0	1	1	1	1	0
x_3	0	1	1	0	1	1	0	0	1	1	1	1
x_4	0	1	1	1	0	1	1	1	1	0	0	1
x_5	0	1	1	1	1	0	1	0	0	1	1	1
x_6	1	0	0	1	1	1	1	0	1	0	1	1
x_7	1	0	1	0	1	1	0	1	1	0	1	1
x_8	1	0	1	1	0	1	1	0	1	1	0	1
x_9	1	0	1	1	1	0	1	1	0	0	1	1
x_{10}	1	1	0	0	1	1	1	0	1	1	1	0
x_{11}	0	0	1	1	1	1	0	1	0	1	1	1
x_{12}	1	1	1	0	1	0	0	0	1	1	1	1
x_{13}	1	0	0	1	1	1	1	1	0	1	0	1

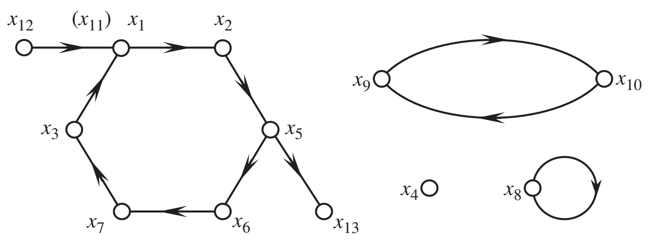


Fig. 4. A simple example of loop strategy detection.

strategy grid. Obviously, it is impossible to enumerate all these strategies, even though sorted by some certain similarity algorithm, the number is still very huge. Therefore, it is necessary to reduce the description of strategy.

4. Morphology of strategy set

A *directed graph* [26,27] or, for short, a *digraph* is a pair $G=(V,E)$ consisting of a finite set V and a set E of ordered pairs (a,b), where $a \neq b$ are elements of V. The elements of V are called *vertices*, those of E *edges*.

A group of strategies can be drawn as a digraph, in which a vertex represents a strategy (or a rule), the direction of edge indicates that execution of the head strategy would lead to executing the tail strategy. Table 2 shows a strategy set containing 13 rules. The corresponding digraph is shown as Fig. 4.

According to the digraph, we can find that there are three types of strategies in strategy set. In Fig. 4, x_4 is an *isolated strategy*, which means there is no other strategy connects up x_4, and x_4 does not connect up any other strategy; $x_{12}-x_1-x_2-x_5-x_{13}$ constitute *chain strategies*; $x_1-x_2-x_5-x_6-x_7-x_3-x_1$ constitute *loop strategies*.

About the loop strategies, there are three types again. The first one is *self-loop strategy*, such as x_8; the second type is two-component loop strategies which are constituted by two strategies, such as x_9 and x_{10}; the third one is formed by more than two strategies, called multi-component loop strategies, such as $x_1-x_2-x_5-x_6-x_7-x_3-x_1$. If we cast off any one strategy in the third type of loop strategies, we can obtain a chain strategies.

In the digraph of Fig. 4, x_{11} is same to x_1, they are superfluous rules; x_{13} and x_6 have the same condition attribute but different decision attribute, therefore they are inconsistent rules.

As in the digraph, there also exist superfluous strategies and inconsistent strategies in strategy set. Two strategies are superfluous strategies means they are totally same, including the same condition attributes and the same decision attributes. Two strategies are inconsistent strategies means they have the same condition attribute but different decision attribute. The superfluous strategies and inconsistent strategies can be detected by rough set theory, and by condition-decision relation matrix too. For details of condition–decision relation matrix, see Section 5. Due to space limitation, we just illustrate the application of rough set theory in detecting superfluous strategies and inconsistent strategies (Table 2).

According to indiscernibility relation in rough set theory, we can get the following relations:

$$IND(\{c_1\})=\{\{x_1,x_2,x_3,x_4,x_5,x_{11}\},\{x_6,x_7,x_8,x_9,x_{10},x_{12},x_{13}\}\}, \tag{1}$$

$$IND(\{c_2\})=\{\{x_1,x_6,x_7,x_8,x_9,x_{11},x_{13}\},\{x_2,x_3,x_4,x_5,x_{10},x_{12}\}\}, \tag{2}$$

$$IND(\{c_3\})=\{\{x_2,x_6,x_{10},x_{13}\},\{x_1,x_3,x_4,x_5,x_7,x_8,x_9,x_{11},x_{12}\}\}, \tag{3}$$

$$IND(\{c_4\})=\{\{x_3,x_7,x_{10},x_{12}\},\{x_1,x_2,x_4,x_5,x_6,x_8,x_9,x_{11},x_{13}\}\}, \tag{4}$$

$$IND(\{c_5\})=\{\{x_4,x_8\},\{x_1,x_2,x_3,x_5,x_6,x_7,x_9,x_{10},x_{11},x_{12},x_{13}\}\}, \tag{5}$$

$$IND(\{c_6\})=\{\{x_5,x_9x_{12}\},\{x_1,x_2,x_3,x_4,x_6,x_7,x_8,x_{10},x_{11},x_{13}\}\}, \tag{6}$$

and

$$IND(\{c_1,c_2,c_3,c_4,c_5,c_6\})=\{\{x_1,x_{11}\},\{x_6,x_{13}\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\},\{x_{12}\}\}. \tag{7}$$

Similarly, we have

$$IND(\{d_1,d_2,d_3,d_4,d_5,d_6\})=\{\{x_1,x_{11}\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\},\{x_{12}\},\{x_{13}\}\}. \tag{8}$$

Obviously, x_1 and x_{11} are superfluous strategies, while x_6 and x_{13} are inconsistent strategies.

5. Loop strategies

Based on the strategy description, because the strategies can be treated as a series of actions, there is a possibility of automatic strategy extraction from the game log file. It is easy to record the game process in real time, and then some data can be selected from the record file. By adding these selected data, or called rules, to the antecedent of rules set, a new strategy set is created.

However, the new selected rules are likely indiscernible or inconsistent with the original rules, which is reflected in three aspects. The first one is some of the new rules might be superfluous data, which means the new data have same condition and same decision to some certain rules in the antecedent of rules set. This kind of superfluous data should be deleted from the rules set. The second one is some of the new rules come into conflict with the original rules. For example, a new rule and an old rule hold the same position condition but different destination grid. In this case, it is necessary to identify the conflictive rules, since they indicate different decisions for the same conditions, and divergent decisions might lead to different follow-up situation of game. The third one is some new selected rules and some old rules form a cycle, called *loop strategies*. In robot soccer game, the loop strategies would make robots be in a trap of executing repeated actions. Fig. 5 shows a simple situation of repeated action in robot soccer simulation, in which the white arrow indicates the robot's

ARTICLE IN PRESS

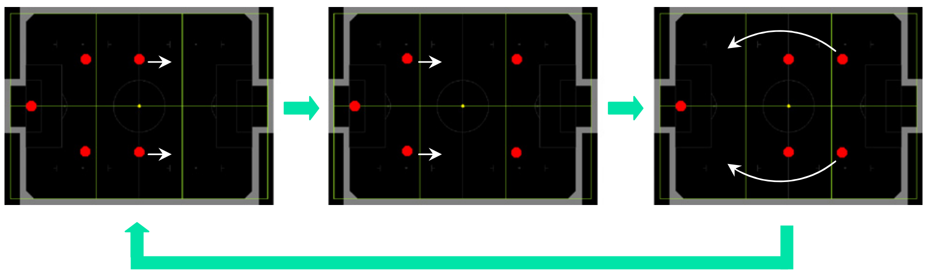


Fig. 5. Repeated action in robot soccer game.

Table 3
An example of loop strategies.

U	c_1	c_2	c_3	c_4	c_5	c_6	d_1	d_2	d_3	d_4	d_5	d_6
x_1	0	0	1	1	1	1	0	1	0	1	1	1
x_2	0	1	0	1	1	1	0	1	1	1	1	0
x_3	0	1	1	0	1	1	0	0	1	1	1	1
x_4	0	1	1	1	1	0	1	0	0	1	1	1
x_5	1	0	0	1	1	1	1	0	1	0	1	1
x_6	1	0	1	0	1	1	0	1	1	0	1	1

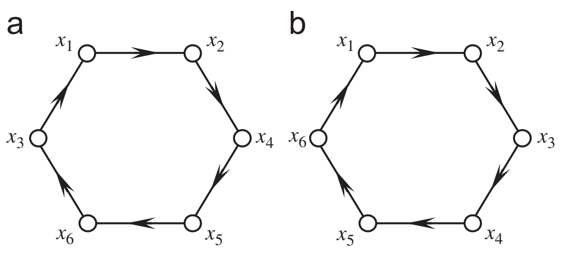


Fig. 6. An example of digraph.

movement direction. Apparently, endless repeated actions are harmful to the game. We must know which rules constitute loop strategies. When the game goes into loop strategies, we must do something to prevent the game from endless repeated actions.

5.1. Existence criterion of loop strategies

Before we begin to discuss the existence criterion of loop strategies, we introduce the concept of condition–decision relation matrix firstly.

Definition 1. Let information system $\mathbb{A}=(U,C\cup D)$ be given, $U=\{x_1,\ldots,x_n\}$, $n=|U|$, $V_C=V_D$. The condition–decision relation matrix, or C–D matrix, denoted as $R_{CD}=[r_{ij}]$, is an $n\times n$ matrix, where

$$r_{ij}=\begin{cases}1 & \text{if } D(x_i)=C(x_j),\ i,j=1,\ldots,n,\\ 0 & \text{otherwise.}\end{cases}$$

In Definition 1, $D(x_i)$ means the sequence of decision attributes over object x_i, $C(x_j)$ means the sequence of condition attributes over object x_j. V_C and V_D are domain of condition set and domain of decision set, respectively.

An example of loop strategies is illustrated in Table 3. Fig. 6(a) shows the corresponding digraph. According to the concept of condition–decision relation matrix, the loop strategies in Table 3 can be expressed as the following matrix:

$$R_{CD}=\begin{bmatrix}0&1&0&0&0&0\\0&0&0&1&0&0\\1&0&0&0&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\\0&0&1&0&0&0\end{bmatrix}. \tag{9}$$

In fact, each matrix that describes loop strategies can be transformed into a matrix with the following form:

$$J_k=\begin{bmatrix}0&1&&\\&0&\ddots&\\&&\ddots&1\\1&&&0\end{bmatrix}_{k\times k}. \tag{10}$$

To get this type of matrix, the only thing we need to do is to relabel the rules in strategy set. This operation can be realized by permutation matrix [28]. For example, in Fig. 6(a) the permutation can be expressed in two-line form by

$$\begin{pmatrix}x_1&x_2&x_4&x_5&x_6&x_3\\x_1&x_2&x_3&x_4&x_5&x_6\end{pmatrix}.$$

Then the corresponding permutation matrix is

$$P_\pi=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&0&0&0&1\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\end{bmatrix}.$$

Consequently, the permutation operation results in

$$P_\pi X=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&0&0&0&1\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\end{bmatrix}=\begin{bmatrix}x_1\\x_2\\x_6\\x_3\\x_4\\x_5\end{bmatrix}.$$

Finally, we relabel x_4 as x_3, x_5 as x_4, x_6 as x_5, and x_3 as x_6, then we get Fig. 6(b), by which we can obtain a new relation matrix as follows:

$$J_{CD}=\begin{bmatrix}0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\\1&0&0&0&0&0\end{bmatrix}, \tag{11}$$

where the corresponding eigenvalues are $\lambda_1=1$, $\lambda_2=0.5+0.866i$, $\lambda_3=-0.5+0.866i$, $\lambda_4=-1$, $\lambda_5=-0.5-0.866i$, $\lambda_6=0.5-0.866i$.

ARTICLE IN PRESS

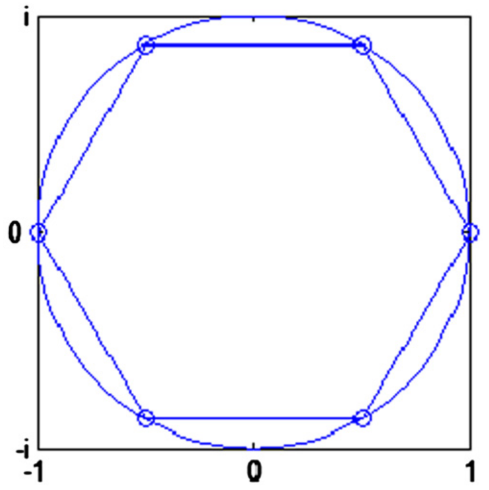


Fig. 7. Eigenvalues of R_{CD} on the complex plane.

These eigenvalues can be drawn as six points on the complex plane. Apparently, these points divide the unit circle on the complex plane into six equivalent parts, which means there are six rules that constitute loop strategies. Fig. 7 shows this situation.

Generally speaking, the matrix of J_k with the form in Eq. (10) has eigenvalues which satisfy the following equation:

$$(-\lambda)^k-(-1)^k=0. \qquad (12)$$

The eigenvalues can be drawn as points on the complex plane. These points divide the unit circle on the complex plane into k equivalent parts.

Now, if we delete a rule from the loop strategies, the rest rules would form chain strategies, and the new condition–decision relation matrix will be *Jordan normal form*. For example, if we delete any one rule from Table 3, the rest five rules will become chain strategies, the corresponding condition–decision relation matrix is

$$J^*_{CD}=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\0&0&0&0&0\end{bmatrix}. \qquad (13)$$

Apparently, J^*_{CD} is the Jordan normal form, its eigenvalue is 0. Therefore we can conclude that, the condition–decision relation matrix of loop strategies always has non-zero eigenvalues, and all the eigenvalues of non-loop strategies' relation matrix are zero.

Additionally, there is another important thing we should notice. The matrix of J_k with the form in Eq. (10) is a *column-stochastic matrix*, because J_k is a square matrix, all of its entries are nonnegative and the entries in each column sum to 1. We should notice that every column-stochastic matrix has 1 as an eigenvalue, which is helpful to count the numbers of rules that constitute loop strategies.

5.2. Detection of loop strategies

There are already well-known cycle detection algorithms in graph theory, while all of them are based on special measures, such as the distance matrix and shortest path in cycles of negative length [26]. Before establishing the concept of these special measures, those existing algorithms could not be applied to detect loop strategies. Consequently, we need to find a novel method to detect the loop strategies.

Now we consider a new matrix $S=R_{CD}+R^t_{CD}$, where R^t_{CD} is the transpose of R_{CD}, then we can obtain the following corollaries, lemmas and theorem. Due to space limitation, we do not prove them.

Corollary 1. *Let* $i\in[1,n]$, *if* $s_{ii}=2$, *then* x_i *is a self-loop strategy.*

Corollary 2. *The matrix S is a symmetric matrix.*

Corollary 3. *Let* $i,j\in[1,n]$, $i\neq j$, *if* $s_{ij}=2$, *then* $r_{ij}=r_{ji}=1$, x_i *and* x_j *constitute loop strategies.*

Corollary 4. *Let* $i,j\in[1,n]$, $i\neq j$, *if there exists* $k\in[1,n]$, *such that* $r_{ik}=r_{jk}=1$, *then* $r_{i\cdot}=r_{j\cdot}$.

Corollary 5. *Let* $u,v\in[1,n]$, $u\neq v$, *if there exists* $i\in[1,n]$, *such that* $r_{iu}=r_{iv}=1$, *then* $r_{\cdot u}=r_{\cdot v}$.

In Section 4, we have known that the loop strategies have three types, that is, self-loop strategy, two-component loop strategies and multi-component loop strategies. Corollary 1 indicates that self-loop strategy would make the corresponding main diagonal element of S to be 1. Corollary 2 is the basis of Corollary 3 that shows how to detect two-component loop strategies. Corollaries 4 and 5 depict the characters of superfluous and inconsistent data. Now we consider x_u and x_v. For the superfluous data, they have the same condition attributes and the same decision attributes, which means $r_{\cdot u}=r_{\cdot v}$ and $r_{u\cdot}=r_{v\cdot}$ if their entries are not all zero. For the inconsistent data, they have the same condition attributes and different decision attributes, which means $r_{\cdot u}=r_{\cdot v}$ and $r_{u\cdot}\neq r_{v\cdot}$ if their entries are not all zero. Consequently, we can deduce that if $r_{\cdot u}=r_{\cdot v}$ and if their entries are not all zero, x_u and x_v must be the data with problems.

In order to detect multi-component loop strategies, we assume that superfluous and inconsistent objects have been deleted from the universe in the following discussion.

Before we discuss the next theorem about detection of loop strategies, firstly we introduce an important concept of *chain strategy*, which will be helpful to understand the detection of loop strategies. As we know, the loop strategies form a cycle of action. If we cut the cycle, we would get a chain. The chain strategies form a series of consecutive actions.

Definition 2. Let $\mathbb{A}=(U,C\cup D)$, $V_C=V_D$, if $D(x_a)=C(x_b)$, then we say $\{x_a,x_b\}$ are directed strategies, x_a and x_b are connectible, more precisely, x_a connects up x_b; if $D(x_a)\neq C(x_b)$, then x_a does not connect up x_b.

Definition 3. Let $\mathbb{A}=(U,C\cup D)$, $|U|=n$, $V_C=V_D$. Suppose there exists a sequence of objects $\{x_1,x_2,\dots,x_m\}$ such that $D(x_1)=C(x_2)$, $D(x_2)=C(x_3)$, …, $D(x_{m-1})=C(x_m)$ and $m\le n$, i.e. $\{x_1,x_2,\dots,x_m\}$ are directed strategies. If x_m does not connect up x_1, then $\{x_1,x_2,\dots,x_m\}$ are chain strategies, x_1 is head of the chain, $\{x_2,\dots,x_{m-1}\}$ are bodies of the chain, x_m is tail of the chain. If x_m connects up x_1, then $\{x_1,x_2,\dots,x_m\}$ are loop strategies.

Definitions 2 and 3 help to understand the chain strategies, and we can deduce the following lemmas according to them.

Lemma 1. *Let* $\mathbb{A}=(U,C\cup D)$, $x_i,x_j\in U$, *and matrix* R_{CD} *is given. Then* x_i *connects up* x_j *iff* $r_{ij}=1$.

Lemma 2. *Let* $\mathbb{A}=(U,C\cup D)$, $x_i,x_j\in U$, *and matrix S is given. Then* x_i *and* x_j *are connectible iff* $s_{ij}\ge 1$.

Lemma 3. *Let* $\mathbb{A}=(U,C\cup D)$, $|U|=n$, *and matrix S is given. Let* $i\in[1,n]$, *then* x_i *belongs to the body of chain strategies iff there are two entries with the value of*1 *in the ith row* (*or column*) *of matrix S.*

Connection is the key of chain strategies. Lemmas 1 and 2 indicate the relationship of two objects, we can judge whether two objects are connected according to them. In addition, together with the definitions, they become the basis of Lemma 3 which makes clear the significant step for the proof of the following theorem.

Theorem 1. *Let $\mathbb{A}=(U,C\cup D)$, matrix S is given, x_i is a body of chain strategies, and $s_{iu}=s_{iv}=1$. The all chain strategies are loop strategies iff for all body x_i, the x_u and x_v are bodies too.*

Theorem 1 shows that, for each object x_i in a loop, there are two entries with the value of 1 in the ith row (or column) of matrix S, these two entries correspond to the head and tail of x_i respectively. Informally speaking, Theorem 1 means that, in loop strategies, everybody is a head, and everybody is a tail.

Based on the corollaries and theorem above, it is easy to find out loop strategies by observing matrix S. Corollary 1 shows that the self-loop strategy only lies in the main diagonal of the matrix S. Corollary 3 expounds the situation that two rules constitute loop strategies. Theorem 1 explains how to detect the loop strategies constituted by multi-rules.

6. Ranking of strategies

Bryan and Leise [29] rank the importance of web pages according to an eigenvector of a weighted link matrix. This model is helpful to rank strategies. But strategy ranking is different to page ranking.

In [29], a core idea in assigning an importance score to any given web page is that the page's score is derived from the links made to that page from other web pages. The links *to* a given page are called the *backlinks* for that page. The web thus becomes a democracy where pages vote for the importance of other pages by linking to them.

While in strategy ranking, the strategy that has more links *to* other strategies is more important because that strategy has more influence on the process of game, the game situation is mostly depends on the follow-up execution of that strategy. In other words, the strategy becomes a power where strategy influence other strategies by linking to them. In addition, this kind of strategy is a *hub* because many other strategies have to be achieved through it. The links to other strategies are called *forwardlinks* for the hub strategy. If we can destroy opponent's hub strategy, then we eliminate most of the follow-up rules of opponent, and vice versa.

Suppose the strategy set of interest contains n strategies, each strategy indexed by an integer k, $1\le k\le n$. An example is illustrated in Fig. 8, in which there are four strategies, an arrow from strategy A to strategy B indicates a link from strategy A to strategy B. Such a strategy set is an example of a *directed graph*. We use x_k to denote the importance score of strategy k in the set.

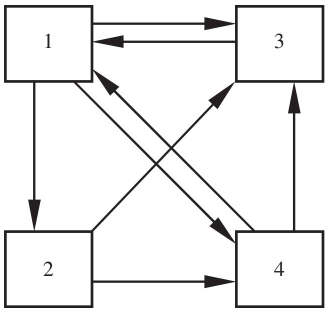


Fig. 8. An example of a strategy set with only four strategies.

x_k is nonnegative, $x_j>x_k$ indicates that strategy j is more important than strategy k, and $x_j=0$ indicates that strategy j has the least possible importance score.

A very simple approach is to take x_k as the number of forwardlinks for strategy k. In the example in Fig. 8, we have $x_1=3$, $x_2=2$, $x_3=1$, and $x_4=2$, so that strategy 1 is the most important, strategies 2 and 4 tie for second, and strategy 3 has the least importance. A link from strategy k becomes a power for strategy k's importance.

This very simple approach ignores an important fact that a link from strategy k to an important strategy should boost strategy k's importance score more than a link to an unimportant strategy. For example, in the strategy set of Fig. 8, strategies 2 and 4 both have two forwardlinks: each links to the other, but strategy 4's second forwardlink is to the seemingly important strategy 1, while strategy 2's second forwardlink is to the relatively unimportant strategy 4. As such, perhaps we should rate strategy 4's importance higher than that of strategy 2.

The most fundamental problem of the simple approach mentioned above is that a single individual could gain influence merely by sending out multiple arrows. Therefore, we seek a scheme in which a strategy does not gain extra influence simply by linking to lots of other strategies. If strategy j contains n_j backlinks, one of which links from strategy k, then we will boost strategy k's score by x_j/n_j, rather than by 1. To quantify a set of n strategies, let $L_k\subset\{1,2,\ldots,n\}$ denote the set of strategies with a backlink from strategy k, that is, L_k is the set of strategy k's forwardlinks. For each k we require

$$x_k=\sum_{j\in L_k}x_j/n_j,$$

where n_j is the number of forwardlinks to strategy j (which must be positive since if $j\in L_k$, then strategy j links from at least strategy k). We will assume that a link from a strategy to itself will not be counted, and we should delete this kind of self-loop strategy from strategy set because it is harmful to the robot's action, which would make robots be in a trap of executing repeated actions.

Now we go back to the example in Fig. 8 again. For strategy 1, we have $x_1=x_2/1+x_3/3+x_4/2$, since strategies 2, 3 and 4 are *forwardlinks* for strategy 1 and strategy 2 contains only one link, strategy 3 assembles three links, while strategy 4 receives two links. Similarly, $x_2=x_3/3+x_4/2$, $x_3=x_1/2$, $x_4=x_1/2+x_3/3$. These linear equations can be written as

$$\begin{bmatrix}0&1&\frac{1}{3}&\frac{1}{2}\\0&0&\frac{1}{3}&\frac{1}{2}\\\frac{1}{2}&0&0&0\\\frac{1}{2}&0&\frac{1}{3}&0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}. \qquad (14)$$

In this case we obtain $x_1=0.3750$, $x_2=0.1875$, $x_3=0.1875$, $x_4=0.2500$. Thus strategy 1 gets the highest importance score.

It should be noted that the node with highest importance score may be foible of the system. Opponent may hinder the implementation of our strategy by attacking this node, and vice versa.

7. Evaluation of strategy

Game strategies can be represented as a digraph as shown in Fig. 9, in which the nodes 1–3 are opponent positions. In addition, the nodes 1, 2 and 3 can be considered as *event*1, 2 and 3, respectively. Event i means the robots stand at the position corresponding to node i. Each arrow pointing to event j from event i represents a rule, r_{ij}, that the decision attribute would be position j under the condition of position i.

ARTICLE IN PRESS

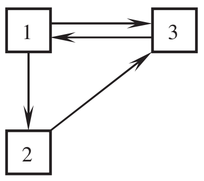


Fig. 9. A digraph of a game.

The probability of event i, denoted by $P(i)$, means the probability of robots' standing at the position i. The probability $P(j|i)$ denotes the probability of rule r_{ij} being selected:

$$P(1) = P(1|3)P(3), \tag{15}$$

$$P(2) = P(2|1)P(1), \tag{16}$$

$$P(3) = P(3|1)P(1) + P(3|2)P(2), \tag{17}$$

$$P(1) + P(2) + P(3) = 1, \tag{18}$$

$$P(1|3) = 1, \tag{19}$$

$$P(3|2) = 1, \tag{20}$$

$$P(2|1) + P(3|1) = 1. \tag{21}$$

These linear equations can be written as

$$\boldsymbol{Ax} = \boldsymbol{x}, \tag{22}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & P(1|3) \\ P(2|1) & 0 & 0 \\ P(3|1) & P(3|2) & 0 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} P(1) \\ P(2) \\ P(3) \end{bmatrix}.$$

Definition 4. A square matrix is called a *column-stochastic* matrix if all of its entries are nonnegative and the entries in each column sum to 1.

Proposition 1. *Every column-stochastic matrix has*1 *as an eigenvalue*.

Proof. Let $\boldsymbol{A}$ be an $n \times n$ column-stochastic matrix and let $\mathbf{e}$ denote an n-dimensional column vector with all entries equal to 1. Recall that $\boldsymbol{A}$ and its transpose $\boldsymbol{A}^T$ have the same eigenvalues. Since $\boldsymbol{A}$ is column-stochastic, it is easy to see that $\boldsymbol{A}^T\boldsymbol{e} = \boldsymbol{e}$, so that 1 is an eigenvalue for $\boldsymbol{A}^T$ and hence for $\boldsymbol{A}$.

Proposition 1 shows there must exist a solution of Eq. (22). Apparently, the matrix $\boldsymbol{A}$ has the eigenvalue

$$\lambda = 1. \tag{23}$$

A decision-making is a *non-preference decision-making* if the rule is selected stochastically at the node with multiple options. Otherwise, it would be *preference decision-making*.

Before game starting, we know nothing about the opponent. So we can assume that they select their rules stochastically if there are multiple options at a certain node. For example, we can assume that $P(2|1) = P(3|1) = 0.5$ in Eq. (22). Then the eigenvector with eigenvalue 1 for matrix $\boldsymbol{A}$ is

$$\boldsymbol{x} = \begin{bmatrix} P(1) \\ P(2) \\ P(3) \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.4 \end{bmatrix}. \tag{24}$$

After a period of time, when we get some records of opponent behaviors, we can count the number of times an event has occurred, then we can obtain the probability of that event. If the statistic probability is unequal to the probability calculated by Eq. (22), then opponent would do their selection as preference decision-making. Apparently, according to the statistic probability we can calculate the unknown conditional probability, based on which we can predict the opponent behaviors.

Now we should add some special nodes to the digraph, such as start node, penalty kick, free kick, goal-kick, free-ball, and so on. According to the probability-based model presented above, we can predict the opponent behaviors, calculate the probability of goal score, based on which we can evaluate strategies, because good strategies could achieve good results. It means better strategies would achieve higher probability of goal score.

8. Generation of offensive strategy

Robot soccer simulator is a good test bed for strategy set. In the simulator, two teams, such as red and blue team, will share the same strategy selection algorithm, prediction of movement, tactics, and so on, which would eliminate the impact of the simulator, and distinctly display differences between the two strategy sets.

According to the game simulation result, it is easy to know which strategy set is better, because the better strategy set could achieve better competition result. In the game log file, we can find the process of goal, which is useful to improve the generation of strategy set. For example, we can find the last action before goal score in the log file of game, and then transform it to be an offensive strategy.

Tables 4 and 5 show the strategy sets of red and blue team, named Redset and BluesetI, respectively. In the strategy set of red team, there are 10 strategies, while 34 strategies in blue team. Table 6 lists the results of 10 games between Redset and BluesetI, where blue team got three wins, five defeats and two ties, with five gains and seven loss balls.

In the first game, the last action before red team made score can be represented as follows:

- *Mine* 42 53 62 63;
- *Oppo* 13 42 42 53;
- *Ball* 63;
- *Dstn* 42 53 63 63.

which can be added as a new strategy to the strategy set, and then the blue team get a new strategy set BluesetII. Table 7 lists the results of 10 games between Redset and BluesetII, where blue team got eight wins and two ties, with 11 gains and only two loss balls. Apparently, the strategy set of blue team is improved greatly by the last strategy that greatly enhances blue team's offensive.

Table 4
Redset—strategy set of red team.

Rule	*Mine*	*Oppo*	*Ball*	*Dstn*
01	21243233	42435154	43	31334243
02	31334243	33425253	42	32435253
03	32435253	43525354	53	33536263
04	23313243	42435253	31	33414253
05	33414253	41425152	41	41435152
06	42435152	41425152	42	52525362
07	42525253	51525252	52	43536263
08	32334243	33424344	43	42435253
09	22232323	23323334	33	22233233
10	12222333	13233233	23	12132323

ARTICLE IN PRESS

Table 5
BluesetI—strategy set of blue team.

Rule	*Mine*	*Oppo*	*Ball*	*Dstn*
01	21243233	42435154	32	22334243
02	22334243	42435154	42	22335253
03	22335253	51525254	52	32335253
04	32335362	51525254	62	32335362
05	21243233	42435154	43	21243233
06	21243233	42435154	33	22334243
07	22334243	42435154	43	32335253
08	32335253	51525354	53	32335253
09	32335263	51525354	63	32335263
10	21243233	42435154	43	21243233
11	21243233	32435154	32	22232432
12	22232432	32335253	32	22232431
13	22232431	32335253	22	22232431
14	22232431	22235253	22	22243133
15	22243133	32335253	33	21222324
16	21222324	32335253	23	21243233
17	32333353	31323334	53	32323353
18	32323352	31323334	52	32323362
19	32323362	31323334	62	32323362
20	32323353	31323334	53	32323363
21	32323363	31323334	63	32323363
22	41424344	23424343	23	41424344
23	41424344	13424343	13	41424344
24	41424344	22424343	22	41424344
25	41424344	12424343	12	41424344
26	32335263	51525354	63	32335253
27	32335253	51525354	44	32344244
28	32344244	41434452	44	22233334
29	22233334	41434452	34	22233334
30	22233334	31323442	34	22232434
31	22232434	31323442	34	22232434
32	32335253	51525354	62	32335243
33	32334352	51525354	52	32334243
34	32334243	41424453	42	22233233

Table 6
Redset vs BluesetI.

Team	*Score*									
Red	1	0	1	0	0	2	0	2	1	0
Blue	0	0	0	1	1	1	2	0	0	0

Table 7
Redset vs BluesetII.

Team	*Score*									
Red	0	1	0	0	0	0	0	1	0	0
Blue	1	2	1	1	0	2	1	1	1	1

9. Conclusions and future work

In this work, we discuss the strategies of robot soccer game in detail. Firstly, we present the description of strategy, describe the morphology of strategy set. Secondly, we analyze the existence criterion of loop strategies, and then present some corollaries and theorems, based on which the loop strategies and chain strategies can be found, also superfluous strategies and inconsistent strategies. After we make clear the structure of strategy set, the importance of every strategy in the strategy set could be computed based on ranking model. Consequently, we can distinguish the important strategies from the others. We should pay more attention to the strategies with high importance score. Based on the idea that good strategy would increase the probability of victory, together with the probability-based model presented in Section 7, we can evaluate the strategies. Additionally, we present a method to generate offensive strategy, and the statistic results of simulation game prove the validity of the method.

As future work, we are interested in integrating the ranking model and probability-based model with robot soccer simulator since as we could predict the follow-up game situation to some extent, we could select strategy more suitably. Hence, by integrating both approaches, we can benefit from their advantages.

Strategies are different to strategy set. Combination of strategies plays an important role to the performance of strategy set. We believe that evolutionary algorithms could be helpful to find good combination of strategies so that the performance of strategy set could be improved.

Acknowledgment

This work was supported by the European Regional Development Fund in the IT4Innovations Center of Excellence project (CZ.1.05/1.1.00/02.0070).

References

[1] J. Kim, D. Kim, Y. Kim, K. Seow, Soccer Robotics, vol. 11, Springer Verlag, 2004.
[2] J. Kolodner, An introduction to case-based reasoning, Artif. Intell. Rev. 6 (1992) 3–34.
[3] A. Aamodt, E. Plaza, Case-based reasoning: Foundational issues, methodological variations, and system approaches, AI Commun. 7 (1994) 39–59.
[4] R. Ros, J. Arcos, R. Lopez de Mantaras, M. Veloso, A case-based approach for coordinated action selection in robot soccer, Artif. Intell. 173 (2009) 1014–1039.
[5] K. Lam, B. Esfandiari, D. Tudino, A scene-based imitation framework for robocup clients, in: Workshop on Modeling Other Agents from Observations, AAAI-06, AAAI Press, Boston, MA, USA, 2006.
[6] R.S. Sutton, A.G. Barto, Introduction to Reinforcement Learning, 1st edition, MIT Press, Cambridge, MA, USA, 1998.
[7] M.A. Riedmiller, A. Merke, D. Meier, A. Hoffman, A. Sinner, O. Thate, R. Ehrmann, Karlsruhe brainstormers—a reinforcement learning approach to robotic soccer, in: RoboCup 2000: Robot Soccer World Cup IV, Lecture Notes in Computer Science, vol. 2019, Springer-Verlag, London, UK, 2000, pp. 367–372.
[8] A. Kleiner, M. Dietl, B. Nebel, Towards a life-long learning soccer agent, in: RoboCup 2002: Robot Soccer World Cup VI, Lecture Notes in Computer Science, vol. 2752, Springer, 2003, pp. 126–134.
[9] Z. Huang, Y. Yang, X. Chen, An approach to plan recognition and retrieval for multi-agent systems, in: Workshop on Adaptability in Multi-Agent Systems, First RoboCup Australian Open, AORC2003, CSIRO, 2003.
[10] A.D. Lattner, A. Miene, U. Visser, O. Herzog, Sequential pattern mining for situation and behavior prediction in simulated robotic soccer, in: RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Computer Science, vol. 4020, Springer, 2005, pp. 118–129.
[11] A. Miene, U. Visser, O. Herzog, Recognition and prediction of motion situations based on a qualitative motion description, in: RoboCup 2003: Robot Soccer World Cup VII, Lecture Notes in Computer Science, vol. 3020, Springer, 2003, pp. 77–88.
[12] J. Lee, D. Ji, W. Lee, G. Kang, M. Joo, A tactics for robot soccer with fuzzy logic mediator, in: Computational Intelligence and Security, CIS 2005, Springer, Xi'an, China, 2005, pp. 127–132.
[13] C. Wu, T. Lee, A fuzzy mechanism for action selection of soccer robots, J. Intelligent Robotic Syst. 39 (2004) 57–70.
[14] K. Jolly, K. Ravindran, R. Vijayakumar, R. Sreerama Kumar, Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks, Robotics Autonomous Syst. 55 (2007) 589–596.
[15] T. Nakashima, M. Takatani, M. Udo, H. Ishibuchi, M. Nii, Performance evaluation of an evolutionary method for robocup soccer strategies, in: RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Computer Science, vol. 4020, Springer, 2005, pp. 616–623.
[16] J. Park, D. Stonier, J. Kim, B. Ahn, M. Jeon, Recombinant rule selection in evolutionary algorithm for fuzzy path planner of robot soccer, in: Advances in Artificial Intelligence, KI 2006, Springer, 2007, pp. 317–330.
[17] S. Konur, A. Ferrein, E. Ferrein, G. Lakemeyer, Learning decision trees for action selection in soccer agents, in: Workshop on Agents in Dynamic and Real-time Environments, ECAI 2004, IOS Press, Valencia, Spain, 2004.
[18] A. Bezek, Discovering strategic multi-agent behavior in a robotic soccer domain, in: Autonomous Agents and Multiagent Systems, AAMAS'05, ACM, New York, NY, USA, 2005, pp. 1177–1178.
[19] A. Bezek, M. Gams, I. Bratko, Multi-agent strategic modeling in a robotic soccer domain, in: Autonomous Agents and Multiagent Systems, AAMAS'06, ACM, New York, NY, USA, 2006, pp. 457–464.

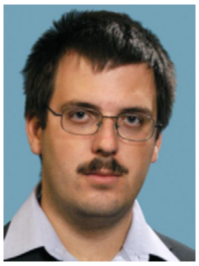
ARTICLE IN PRESS

[20] H. Huang, C. Liang, Strategy-based decision making of a soccer robot system using a real-time self-organizing fuzzy decision tree, Fuzzy Sets Syst. 127 (2002) 49–64.
[21] J. Martinovič, V. Snášel, E. Ochodková, L. Zołtá, J. Wu, A. Abraham, Robot soccer—strategy description and game analysis, in: Modelling and Simulation, ECMS 2010, Kuala Lumpur, Malaysia, pp. 265–270.
[22] K.G. Jolly, R. Sreerama Kumar, R. Vijayakumar, An artificial neural network based dynamic controller for a robot in a multi-agent system, Neurocomputing 73 (2009) 283–294.
[23] R. Kala, A. Shukla, R. Tiwari, Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, Neurocomputing 74 (2011) 2314–2335.
[24] B. Horák, M. Obitko, J. Smid, V. Snášel, Communication in robotic soccer game, in: B.J. d'Auriol (Ed.), Communications in Computing, CSREA Press, Las Vegas, NV, USA, 2004, pp. 295–301.
[25] V. Srovnal, B. Horák, R. Bernatík, V. Snášel, Strategy extraction for mobile embedded control systems apply the multi-agent technology, in: M. Bubak, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science, ICCS 2004, Lecture Notes in Computer Science, vol. 3038, Springer, Krakow, Poland, 2004, pp. 631–637.
[26] D. Jungnickel, Graphs, Networks and Algorithms, vol. 5, Springer Verlag, 2008.
[27] J. Bondy, U. Murty, Graph Theory, Graduate Texts in Mathematics, vol. 244, Springer, New York, 2008.
[28] I. Bronshtein, K. Semendyayev, K. Hirsch, Handbook of Mathematics, Springer, New York, NY, 2007.
[29] K. Bryan, T. Leise, The $25,000,000,000 eigenvector: the linear algebra behind google, SIAM Rev. 48 (2006) 569–581.

Jie Wu received the M.Eng. degree from Hubei University of Technology, China. At present he is a Ph.D. student in VŠB - Technical University of Ostrava, Czech Republic, major in applied mathematics and information technology. He works in a multi-disciplinary environment involving artificial intelligence, rough set theory, graph theory, electrical engineering, etc.

Václav Snášel has over 25 years of research experience. He works in a multi-disciplinary environment involving artificial intelligence, multidimensional data indexing, conceptual lattice, information retrieval, semantic web, knowledge management, data compression, machine intelligence, neural network, web intelligence, data mining and applied to various real world problems. He has given more than 20 plenary lectures and conference tutorials in these areas. He has authored/co-authored several refereed journal/conference papers and book chapters. He has published more than 400 papers and 147 are recorded at Web of Science.

Eliška Ochodková is an assistant professor at the Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava, Czech Republic. Her research interests include combinatorics, complex networks, graph theory, robot soccer, computer security and cryptography.



Jan Martinovič received the M.Eng. degree and Ph.D. degree in computer science from VŠB - Technical University of Ostrava, Czech Republic, in 2004 and in 2008 respectively. Nowadays, he works as an assistant professor at the same university. He is a researcher dealing with data mining and data processing, with emphasis on web search, data compression and social networks.

Václav Svatoň received the M.Eng. degree in Computer Science from VŠB - Technical University of Ostrava, Czech Republic, in 2010. Currently he continues studying for Ph.D. degree at the same university. His main area of interest is data mining and data analysis.

Ajith Abraham received the M.S. degree from Nanyang Technological University, Singapore, and the Ph.D. degree in Computer Science from Monash University, Melbourne, Australia. He has a worldwide academic experience with formal appointments in many Universities. He serves/has served the editorial board of over 50 international journals and has also guest edited 40 special issues on various topics. He has published more than 750 publications, and some of the works have also won best paper awards at international conferences. His research and development experience includes more than 20 years in the industry and academia. He works in a multidisciplinary environment involving machine intelligence, network security, various aspects of networks, e-commerce, Web intelligence, Web services, computational grids, data mining, and their applications to various real-world problems. He has given more than 50 plenary lectures and conference tutorials in these areas.