Soccer Without Intelligence[*]

Tekin Meriçli and H. Levent Akın
Department of Computer Engineering
Boğaziçi University
34342, Bebek, Istanbul, Turkey
{*tekin.mericli, akin*}*@boun.edu.tr*

***Abstract*—Robot soccer is an excellent testbed to explore innovative ideas and test the algorithms in multi-agent systems (MAS) research. A soccer team should play in an organized manner in order to score more goals than the opponent, which requires well-developed individual and collaborative skills, such as dribbling the ball, positioning, and passing. However, none of these skills needs to be perfect and they do not require highly complicated models to give satisfactory results. This paper proposes an approach inspired from ants, which are modeled as Braitenberg vehicles for implementing those skills as combinations of very primitive behaviors without using explicit communication and role assignment mechanisms, and applying reinforcement learning to construct the optimal state-action mapping. Experiments demonstrate that a team of robots can indeed learn to play soccer reasonably well without using complex environment models and state representations. After very short training sessions, the team started scoring more than its opponents that use complex behavior codes, and as a result of having very simple state representation, the team could adapt to the strategies of the opponent teams during the games.**

***Index Terms*—Braitenberg vehicles, robot soccer, reinforcement learning.**

I. Introduction

Multi-agent Systems (MAS) is the subfield of artificial intelligence (AI) that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of individual agents' behaviors [1]. MAS are becoming popular because of their robustness and success rate in achieving a given task especially in real-time, complex domains since these kinds of domains require agents to act effectively both autonomously and as part of a team. Being a complex and dynamic environment, and having a goal that can be achieved more successfully with multiple agents than a single agent, soccer is an excellent testbed for MAS research.

The main goal in a soccer game is to score more goals than the opponent team and win the game. In order to achieve this goal, the team should play in an organized manner. That requires having well-developed individual skills; such as dribbling and kicking the ball, and collaborative skills; such as passing and proper positioning. However, individual robots and the team as a whole should preferably perform the best possible action in all states in order to increase the chance of scoring a goal. Therefore, effective action selection in a given state has a vital role in being successful in a soccer game. This work also aims to learn when to select which action; that is, learning a mapping between states and actions since the actions of an agent are determined by the state of its environment.

The motivation behind this work is to create a team of autonomous robots that are able to play soccer by using combinations of very primitive behaviors as ants do when accomplishing a complex task in nature. The main source of inspiration is the complexity of behaviors that Braitenberg vehicles [2] can demonstrate although their underlying architectures are extremely simple.

Because of the inherent complexity of MAS, Machine Learning (ML) is an interesting and promising area to combine with MAS. Using ML techniques, effective individual and collaborative skills can be learned in a multi-agent system. However, most of the ML applications require a large amount of labeled examples; that is, one has to provide information about thousands of different situations in order to make a machine learn a concept. On the other hand, in robot soccer case, it is impossible to provide labeled examples to the system because of the complexity and dynamic structure of the environment. Therefore, a trial/error and reward/punishment approach is necessary to be able make learning possible in this domain. Reinforcement Learning (RL) is a learning method that can be used when the agent is only informed about the degree of correctness (or incorrectness) of a sequence of actions. Specifically $Q(\lambda)$ algorithm [3] , which is a RL method, is used in this work. According to the training results, a significant decrease in the number of opponent goals was observed, which means that the team learned a defensive behavior, as well as an increase in the number of own goals, which is an indicator of a learned offensive behavior.

The rest of this paper is organized as follows. Section II provides information about related work and Section III elaborates on our proposed approach. Experiments are explained in detail and results are discussed in Section IV. Section V summarizes and concludes the paper, and proposes some further extensions.

[*]This work is supported by Boğaziçi University Scientific Research Projects Grant 06HA102 and by TUBITAK with Project Grant 106E172.

II. Related Work

Robot soccer has been a very fruitful domain for intelligent robotics and multi-agent systems researchers. There have been many research efforts on various problems related to both individual robot skills, which are necessary to play soccer, and team coordination, which is essential to improve the efficiency of the players and hence the quality of the game.

Related work can roughly be divided into two groups as learning specific instances or sub-games and learning large scale tasks.

A. Learning Specific Instances or Sub-Games

Stone et al. [4], [5] introduced keepaway as a benchmark for machine learning research. The keepers learned individually when to hold the ball and when to pass to a teammate. The proposed method was implemented on the RoboCup simulated soccer environment [6] and they represented the state with 13 state variables for 3 keepers versus 2 takers and with 18 state variables for 3 keepers versus 3 takers versions. Considering that keepaway is only a very small subproblem of the complete robot soccer domain, the model is way too complex compared 3 state variables for 4 players that is defined in our proposed approach.

Kalyanakrishnan et al. [7] extended the keepaway problem to half-field offense, analyzed the learning algorithm that has been most successful for keepaway, and scaled it to meet the demands of the half field offense task. Inter-agent communication was used for more frequent and reliable learning updates. In this work the authors only focused on learning the behavior of the offense player who has possession of the ball; the rest of the team followed a pre-defined formation.

Merke and Riedmiller [8] modelled the soccer domain as a multi-agent Markov Decision Process, and applied both theoretically founded distributed reinforcement learning algorithms and emprically and heuristically motivated way of modified single agent Q-learning algorithm to the problem. They trained their team for learning only coordinated offensive behavior and they obtained promising results, though they did not deal with partial observability of state information.

Noda et al. [9] considered a very small sub-problem of offensive behavior which requires cooperation among the attackers. They trained a neural network that provided two values for shooting directly towards the goal and passing to the teammate. Positions of all objects including the players, the ball, and the goal were fed to the neural network as input, and a binary feedback in the form of success or failure was used to adjust the weights of the network. After training, they obtained a strategy where the robots passed the ball to their teammates when the opponent goalie blocked the shoot way, and they directly shot the ball towards the goal when the goalie was closer to their teammate.

B. Learning Large Scale Tasks

Luke et al. [10] applied genetic programming to behavior-based team coordination in the RoboCup Soccer Server domain. They initialized the players of the team as random movers, then the players evolved to “kiddie-soccer” players, in which all players chased the ball almost aimlessly, then some of the players learned to perform defender behavior, and finally the team converged to an acceptable solution, where the players began to disperse throughout the field and to pass to teammates when appropriate instead of kicking straight to the goal.

The Darwin United team by Andre and Teller [11] was evolved as a team of coordinated agents in the RoboCup simulator. Given the lowest level perceptual inputs, each member of the team learned to execute series of the most basic actions; such as *turn*, *kick*, and *dash* in order to be able to play soccer as a team. Inspired from the work of Luke et al. [10] they also used genetic programming for evolution, providing initial generations as a set of automatically generated functions that encode some simple functionality such as running to the ball and kicking.

In his research, Andou [12] focused on the positionings of the players on the field; that way, the team could adjust their formation according to the opponent’s strategy. The soccer field was divided into grids and a neural network was trained to map the distance measurements of almost all important objects on the field to the best position for the player. In order to refine the result, reinforcement learning was used to update the weights of the neural network. The resulting team had a better performance than the teams with fixed player positions.

The most similar work to our approach was published by Riekki and Röning [13]. They used a goal-oriented and behavior-based architecture, called the *Samba* control architecture, which is inspired by Brooks’ *subsumption architecture* [14], [15], where they propose a mode of executing tasks by modifying and combining primitive reactions. Using this approach, they could obtain some reasonable team behavior in both offensive and defensive halves of the field. However, all these behaviors were manually defined and there was no machine learning and optimization included in their work.

There have been some significant amount of research done in allocating tasks to robots in a team given all primitive actions required to play soccer and some pre-defined roles, such as *attacker*, *supporter*, *midfielder*, and *defender*. Meriçli proposed some metrics calculated from positions of robots and ball on the field, and used a subset of these metrics that are proved to be informative statistically [16], [17]. These most informative metrics were then used to build a task allocation algorithm. The algorithms were implemented and tested on the TeamBots mobile robot simulator environment [18]. Köse et al. [19], [20] presented a task allocation algorithm based on free-market approach, in which the players evaluate their fitnesses to each role by calculating a cost function for accomplishing the main task of that role. These costs are exchanged among teammates and whichever player offers the lowest cost for a role is assigned to that role. They used low level potential field denitions proposed by Kaplan [21] for motion planning. Balch *et.al.* and Coradeschi *et.al.* also proposed some different methods for role assignment and proper formation tasks [22]–[24].

All of these methods propose solutions for different aspects of robot soccer in different scales; however, there has not been any attempt to try to learn soccer as a whole using very primitive behaviors and extremely small state-action space, which is the main focus and contribution of this work.

III. Braitenberg Soccer

In his book [2], Valentino Braitenberg describes a number of wondrous vehicles that behave in unexpectedly complex ways based on the use of a few electronic neurons. Braitenberg gives this as evidence for the "law of uphill analysis and downhill invention". What this means is that it is much more difficult to try to guess internal structure just from the observation of behavior than it is to create the structure that results in the behavior. Similar in concept to Grey Walters's seminal neural work with his robot tortoises [25], Braitenberg's vehicle behavior is more straightforward, making it somewhat easier to follow, both theoretically and logically. Therefore, it is easier to implement into real robotic designs.

The main motivation behind this work is that playing soccer reasonably well does not require highly complicated models for each skill, vast state spaces, and long training sessions that last for thousands of episodes; at least for non-robot soccer players. Hence, we aimed keeping the representation simple yet informative enough in order to make it possible for our robot soccer players to learn both how to use individual skills; such as kicking, dribbling, and avoiding opponents efficiently and how to improve collaborative skills; such as proper positioning and passing. If we take a closer look at the general structures of these skills, we see that they all have two primitive behaviors in common: *moving towards a point* and *moving away from a point*. These behaviors can easily be implemented as the behaviors of Braitenberg vehicles; such as "aggression" and "fear", and the implementation can be approximated using force fields as shown in Figure 1 [21]. Force fields are one of the simplest and most effective controllers for mobile robots. A force field is basically a combination of some attractive and repulsive forces on an object. For example, being the primary object of interest on the field, the ball has an attractive force field on it whereas an obstacle on the field, such as an opponent defender, has a repulsive force field on it. In Figure 1, an attractive force field is placed on the ball (represented as an orange circle in the middle) as well as a circular field that makes it possible

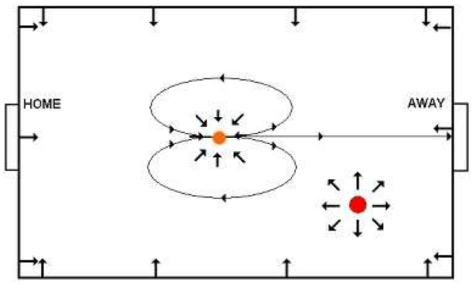

Fig. 1. Force fields used as the underlying motion mechanism.

for the robot to end up facing towards a point when it meets the ball. Other robots (represented as a red circle) and the border lines have repulsive force fields on them in order to keep our robot away from them and inside the border lines, respectively.

The main goal in a soccer game is to score more goals than the opponent team, which can be achieved by carrying and/or pushing the ball towards the opponent goal, specifically behind the goal line. Making an analogy from the nature, the behavior that we observe in ants while they are carrying food to their nests is a perfect biological inspiration for modeling this task. If we treat the robots as ants, the ball as food, and the opponent goal as their nests, then what we actually ask our "soccer playing ants" to do is to carry the food to their nests as quickly and easily as possible while avoiding all the obstacles that they sense on the way. In fact, all of the "complex" behaviors required for achieving this task emerge from combinations of the two primitive behaviors that we have mentioned previously, *moving towards a point* and *moving away from a point*, which in some sense very similar to the *subsumption architecture* [15]. If the robot

- *moves towards* the ball **and** *moves towards* the opponent goal, ***attacker*** behavior is observed
- *moves towards* the ball **and** *moves towards* the home goal, ***defender*** behavior is observed
- *moves away from* the ball **and** *moves towards* the opponent goal, ***supporter*** behavior is observed
- *moves towards* the ball **and** *moves towards* a teammate, ***passing*** behavior is observed.

Figure 2 illustrates those behaviors. Passing and attacking have similar structures; moving towards a point while aiming towards another point, and when the point falls within a certain angle range, kicking the ball towards the point. Defense and support behaviors are different in the sense that they are implemented as trying to move towards / away from two different points at the same time, hence ending up at where the vector sum of those directions point. The intermediate lines with terminal dots in Figure 2 indicate the points where the robot ends up staying while performing defender and supporter behaviors. Also, an implementation detail should be given in order to understand how long distance kicks and

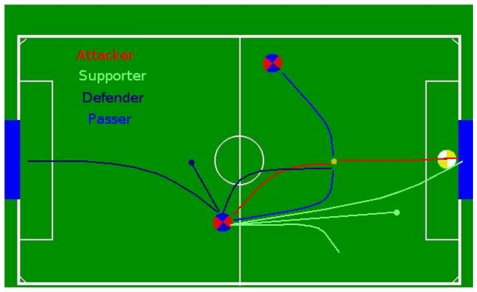

Fig. 2. Combination of the primitive behaviors *moving towards a point* and *moving away from a point* to create more “complex” behaviors, such as offense (red), defense (dark blue), support (green), and passing (blue).

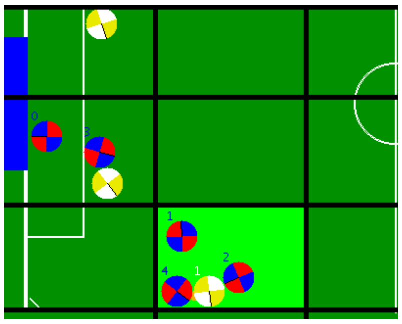

Fig. 3. Portion of the quantized robot soccer field.

passes work; the robots kick the ball as soon as the point that they aim falls within a certain orientation threshold.

Now the problem reduces to determining when to choose which action. Since soccer environment is continuous, the first step to take is to discretize the environment by dividing the field into grids. Quantization makes the representation of the positions of each robot and the ball much simpler, that is just a single number representing the *cell ID*. Another important factor that affects the action selection mechanism is the state of the immediate surrounding of the ball, which is represented as a single *dominance value*. Dominance in a cell is calculated as the difference between the number of teammates (including the robot itself) and the number of opponent robots. Dominance value is set as 2 if our robots are dominant, 1 if the opponents are dominant, and 0 if the number of robots from the two teams is the same. Figure 3 illustrates the discretization of the field and the dominance in a given cell. Blue-red robots belong to our team and yellow-white ones are the opponents. The ball is in the light-green colored cell and the dominance value in that cell is 2.

Another important environmental information for the robots is their distance to the ball; whether they are the closest, a teammate is the closest, or an opponent player is the closest. Similar to the dominance value, this information is also represented as three discrete values.

In a given state, the robot can perform one of the five different actions that are observed when the robots combine different primitive behaviors, namely attacking the opponent goal, supporting the attacker, defending the home goal, passing to the closest teammate, and passing to the teammate that is closest to the opponent goal. It is more convenient and easier to implement to assign probability values to these actions in a given state and pick the one that will result in the highest reward; that is the one with the highest probability value. Therefore, the state-action mapping representation is

$$ballCell, closest, dominance, action \rightarrow probability$$

Initially the probability values are all equal and 0.2, which means that all five actions have equal chances to be selected in all states. Adjustment of those probability values is done through learning, in particular *reinforcement learning*. Specifically, the $Q(\lambda)$ [3] algorithm is used in this work. After the Q values are computed, they are normalized so that they could be treated as probability values and the sum of the probabilities of possible individual actions in a given state would be 1.0.

Reward, that is an increase in the probability of selecting an action, or punishment, that is a decrease in the probability of selecting an action is given in two different ways as being immediate or delayed. Immediate rewards and punishments are given to the robot based on whether the last action it performed resulted in pushing the ball closer to the opponent goal, or one of the teammates or the robot itself touching the ball again in the next step. A delayed reward or punishment is given after a goal is scored and the last N actions are affected from this assignment inversely proportional to their temporal distances from the last action since these series of actions resulted in scoring or receiving a goal. The exploration / exploitation rate is decreased throughout the game; therefore, initially the robots tend to explore the state-action space. However, after some time, they start exploiting the actions that they decided are more beneficial to perform in a given state.

Since it is almost impossible for any two robots on the field to have the exact same distance to the ball, most of the time each robot in the team has a different state-action representation tuple for the same state. Therefore, at each time step, each robot “experiences” a different situation, makes a decision based on what it observes, and modifies a different part of the probability table. This experience table is shared among the teammates; that is, the robots implicitly share their experiences. What actually happens is that they communicate through changing the environment, which is very similar to what swarms do in nature (stigmergy). That eliminates the need for an explicit communication protocol, simplifying the model even further.

In some situations proper positioning may be more beneficial than chasing the ball. One of the preferred positioning styles is the diamond formation in which one of the robots

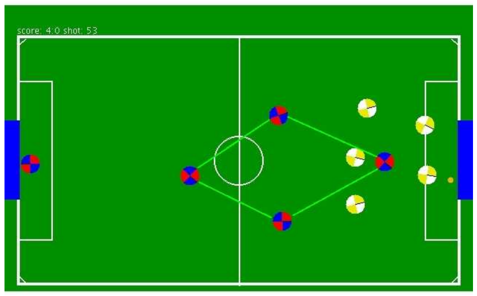

Fig. 4. Diamond formation as a result of emergent assignment of the attacker, supporter, and defender behaviors.

attack the opponent goal, two robots support the attacker from two sides, and one robot stays back to defend the own goal. Figure 4 shows the diamond formation that emerged as the assignment of the positioning related and chasing related roles among the robots.

One of the main advantages of the approach that we propose is that since the state-action space is very small it is possible to keep updating the experience table while playing against a team, that is learning to deal with the opponent team on the fly. That also provides observation of reasonable behaviors very quickly without training the team for thousands of episodes.

IV. Experiments

The experiments were run on the TeamBots simulation environment [18], which can be thought of as a simulation of the FIRA - MiroSot league [26], and an approximation of the RoboCup Small Size League [27]. A team of five players was considered and only the non-goalie players were trained. The goalie ran a very simple positioning code which placed the robot on the intersection of line that connects the ball and the center of the goal box and the goal line.

There are five different hyper-parameters that affect the course of the game which are the constant determining the exploration / exploitation rate K, the dimensions of the grid GD, the length of the state-action history HS, and the values of immediate I and delayed D rewards / punishments.

An incremental method is followed for training. Initially the team is trained on an empty field to promote the actions that carry the ball towards the opponent goal without any intervention. After a relatively short training period, the team was forced to play against the *NullTeam* which has stationary robots on the field. With the presence of opponent robots on the field the team learned to deal with the opponent robots and reach the opponent goal without colliding with the opponents and losing the possession of the ball. In order to let the team develop some defensive skills, the team was trained against the *BrianTeam* team, the codes of which come with the TeamBots simulator. Finally, the robots were trained against a strong team named *MarketTeam*, which

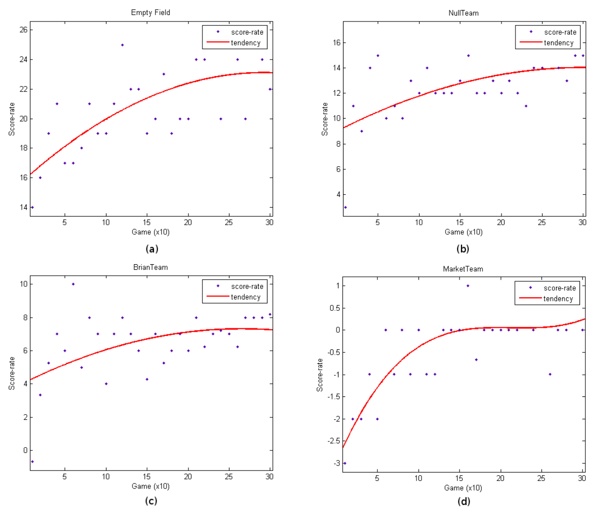

Fig. 5. Results of the games played **(a)** on the empty field, then against **(b)** *NullTeam*, **(c)** *BrianTeam*, and **(d)** *MarketTeam*.

uses a market-driven role allocation algorithm and a similar potential field approach in which the coefficients of the field forces are trained by using Genetic Algorithms (GA) [19], [20]. Being a strong and offensive team, *MarketTeam* forced our team to learn some defensive behavior. At the end of each game, the *score-rate*, which is defined in terms of the difference between the own score and the opponent score is used as an evaluation criterion (Equation 1).

$$ScoreRate = \begin{cases} ownScore\frac{(ownScore-oppScore)}{(ownScore+oppScore)} & \text{if } ownScore > oppScore, \\ oppScore\frac{(ownScore-oppScore)}{(ownScore+oppScore)} & \text{if } ownScore < oppScore, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

After playing against three teams with different K, GD, HS, I, and D values, initial results showed that keeping K in the range of $0.5 < K < 1.0$ works better since it keeps the exploration / exploitation rate balanced. A slightly bigger history size ($HS = 10$) helped making long term plans by learning a path from a specific position towards the one that leads to an own score. Keeping grid size not as big ($GD = 3 \times 3 = 9$) helped the robots have a more compact representation of different portions of the field. Since these three coefficients with these specific values led to more own scores, which is the desired result, we kept them unchanged and trained the team incremantally using these values. Results of the 2-minute games played on the empty field, against *NullTeam*, *BrianTeam*, and *MarketTeam* are provided in Figure 5. 300 games were played in each configuration, and the average score-rates after every $10th$ game were recorded. The blue dots represent the score-rate whereas the red curves represent the general tendency, which is towards bigger score-rates as a sign of scoring more goals and receiving less goals.

As seen in the results, there is a general tendency towards scoring more goals (represented as the red curves) while preventing the opponent from scoring. Very powerful attacking strategy of *MarketTeam* team prevented our team from scoring many goals; that is why the score rate is mostly negative or zero until the last couple of games in Figure 5 (d). However, those results indicate that our team learned how to defend its goal against this team, since the opponents started scoring less towards the last games. Also when the team starts the next game with some experience from the past games, the score rate tends to be very low if negative, and reasonably high if positive, which indicates that the knowledge from the past games is transferred to the current game.

V. CONCLUSIONS & FUTURE WORK

We proposed a biologically inspired approach based on the principles Braitenberg vehicles for creating a team of soccer playing robots that use combinations of very primitive skills to implement “complex” behaviors such as attacking the opponent goal, supporting the attacker, defending the own goal, and passing. We used $Q(\lambda)$ learning to learn the mapping between states, which are represented with only 3 state variables, and one of five possible actions.

The experiments show that our team was able to learn how to score goals efficiently and how to defend their own goal, converging to one of the optimal formations, which is a diamond-shaped formation at the end of a short training period, which started on an empty field and ended after playing against a very strong team. The algorithm is implemented in TeamBots simulation environment, in which differential drive robots are used.

The three main contributions of this paper are as follows.

- It demonstrates that a group of robots can indeed learn how to play soccer as a team using combinations of very primitive behaviors,
- It proves that reasonable results can be obtained without complex state representations and the amount of time necessary for training can be decreased significantly by using a very small state space,
- It proposes a very simple yet effective mechanism that makes learning and adaptation on the fly during a game possible.

This approach can be extended to more than five players by simply defining more targets on the field that the robots can move towards or away from; for instance, the opponent players can be marked in that way. Tests on scalability of this approach are left as future work.

REFERENCES

[1] P. Stone and M. Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. In Autonomous Robots, Volume 8, pages 345-383, July, 2000.

[2] V. Braitenberg. Vehicles: Experiments in Synthetic Psychology. MIT Press / Bradford Books, 1984.

[3] E. Alpaydın. Introduction to Machine Learning. MIT Press, 2004.

[4] P. Stone and R. S. Sutton. Keepaway Soccer: A Machine Learning Testbed. In RoboCup-2001: Robot Soccer World Cup V, pages 214-223, Berlin, 2002. Springer-Verlag.

[5] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement Learning for RoboCup-Soccer Keepaway. In Adaptive Behavior, volume 13, pages 165-188, 2005.

[6] The RoboCup Soccer Simulator. http://sserver.sourceforge.net/.

[7] S. Kalyanakrishnan, Y. Liu, and P. Stone. Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study. In RoboCup-2006: Robot Soccer World Cup X, pages 72-85. Springer-Verlag, Berlin, 2007.

[8] A. Merke and M. A. Riedmiller. Karlsruhe Brainstormers - A Reinforcement learning approach to robotic soccer. In RoboCup 2001: Robot Soccer World Cup V, pages 435-440, London, UK, 2002. Springer-Verlag.

[9] I. Noda, H. Matsubara, and K. Hiraki. Learning Cooperative Behavior in Multi-agent Environment - A Case Study of Choice of Play-plans in Soccer. In PRICAI’96: Proceedings of the 4th Pacic Rim International Conference on Articial Intelligence, pages 570-579, London, UK, 1996. Springer-Verlag.

[10] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-Evolving Soccer Softbot Team Coordination with Genetic Programming. In Proceedings of the First International Workshop on RoboCup, pages 115-118, Nagoya, Japan, 1997.

[11] D. Andre and A. Teller. Evolving Team Darwin United. In M. Asada and H. Kitano, editors, RoboCup-98: Robot Soccer World Cup II, volume 1604 of LNCS, pages 346-351, Paris, France, July, 1999. Springer-Verlag.

[12] T. Andou. Renement of Soccer Agents’ Positions Using Reinforcement Learning. In RoboCup-97: Robot Soccer World Cup I, pages 373-388, London, UK, 1998. Springer-Verlag.

[13] J. Riekki and J. Röning. Playing Soccer by Modifying and Combining Primitive Reactions. In RoboCup-97: Robot Soccer World Cup I, pages 74-87, London, UK, 1998. Springer-Verlag.

[14] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. In IEEE Journal of Robotics and Automation, volume 2, pages 14-23, March, 1986.

[15] R. A. Brooks and A. M. Flynn. Fast, Cheap and Out of Control: A Robot Invasion of the Solar System. pages 478-485, October, 1989.

[16] Ç. Meriçli. Developing a Novel Robust Multi-Agent Task Allocation Algorithm for Four-Legged Robot Soccer Domain. MS Thesis, July, 2005.

[17] Ç. Meriçli and H. L. Akın. A Layered Metric Definition and Validation Framework for Multirobot Systems. RoboCup International Symposium 2008, Suzhou, China, July 15-18, 2008.

[18] T. Balch. TeamBots Mobile Robot Simulator. http://www.teambots.org 2000.

[19] H. Köse, Ç. Meriçli, K. Kaplan, and H. L. Akın. All Bids For One And One Does For All: Market-driven Multi-agent Collaboration in Robot Soccer Domain. In ISCIS 2003, pages 529-536, November, 2003.

[20] H. Köse, K. Kaplan, Ç. Meriçli, and H. L. Akın. Genetic Algorithms Based Market-driven Multi-agent Collaboration in the Robot-soccer Domain. In FIRA Robot World Congress 2003, Vienna, Austria, October, 2003.

[21] K. Kaplan and H. L. Akın. A Controller Design for Soccer-robot Teams. In TAINN, 2003.

[22] T. Balch. Learning Roles: Behavioral Diversity in Robot Teams. In College of Computing Technical Report GIT-CC-97-12, Atlanta, Georgia, March 1997.

[23] T. Balch and R. C. Arkin. Behavior-Based Formation Control for Multi-Robot Teams. In IEEE Transactions on Robotics and Automation, 1999.

[24] S. Coradeschi and L. Karlsson. A Role-Based Decision-Mechanism for Teams of Reactive and Coordinating Agents. In RoboCup-97, pages 112-122, 1997.

[25] W. G. Walter. The Living Brain. New York, 1963.

[26] FIRA - Federation of International Robot-soccer Association. http://www.fira.net/soccer/mirosot/overview.html

[27] RoboCup Small Size League. http://small-size.informatik.uni-bremen.de/