Correcting and Improving Imitation Models of Humans for Robosoccer Agents

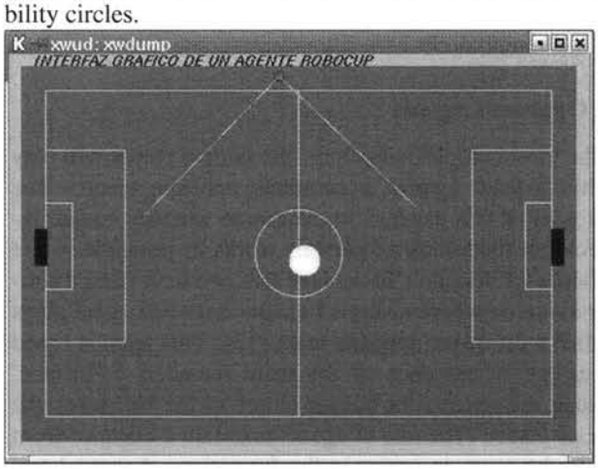
Ricardo Aler, Oscar Garcia, and Jose M. Valls
Universidad Carlos III de Madrid
Avenida Universidad, 30
28911 Leganes (Madrid)
aler@inf.uc3m.es, jvalls@inf.uc3m.es

Abstract- The Robosoccer simulator is a challenging environment, where a human introduces a team of agents into a football virtual environment. Typically, agents are programmed by hand, but it would be a great advantage to transfer human experience into football agents. The first aim of our paper is to use Machine Learning techniques to obtain models of humans playing Robosoccer. These models can be used later to control a Robosoccer agent. However, models did not play as smoothly and optimally as the human. To solve this problem, the second goal of this paper is to incrementally correct models by means of evolutionary techniques, and to adapt them against more difficult opponents than the ones beatable by the human.

1 Introduction

The Robosoccer simulator is a challenging Artificial Intelligence environment where programmers introduce a team of software agents in a simulated football field, to compete against other teams [17, 18]. Machine Learning (ML) is becoming a promising way of automatically endowing these agents with complex skills. The main approach that has been followed so far is to let the agents learn the behaviors by themselves, totally or partially [22, 36, 4, 14]. However, instead of learning behaviors from scratch, control knowledge can be transferred from humans to agents. In this paper, we develop an interface so that a person can play Robosoccer. Every cycle, a pair $(situation, action)$ is saved, that describes the action the human carried out in a particular situation. From these instances, models were learned using PART, a Machine Learning algorithm [15]. Models could then be used to control football agents. They played well, but the style of play was not as smooth or efficient, as the human play. Therefore, the second goal of this paper is to correct these problems by using evolutionary techniques. In particular, we will use the technique of seeding the initial population with a working (but imperfect) individual, and letting the system evolve from there. This technique has been used often with good results (see for instance [3, 20]). In addition, we have found out that in order to build the model, the human had to play against a challenging enough opponent, but beatable. Otherwise, it would not have been possible to produce useful models. But it is interesting that the model is able to play against more difficult opponents. So, in this paper we will also use the same evolutionary technique (seeding the initial population) to adapt the model to new, more difficult, opponents. To summarize, our second goal is to correct and to improve human models composed of sets of rules.

Figure 1: 2D Interface. Objects are represented by probability circles.



This paper has been divided into five sections. First, Section 2 describes our modelling approach in the Robosoccer domain. Section 3 describes how the agent was trained from the instances generated by the human playing Robosoccer. Section 4 improves the model obtained by means of an evolutionary algorithm. Section 5 comments on some research related to building models from other agents. Finally, Section 6 concludes and points towards future work.

2 Our Approach

2.1 The Robosoccer Interface

We have programmed an interface to let a human connect to a Soccerserver [17, 18] and play the game. Soccerserver version 6.07 was used for this purpose. The appearance of our Human-Soccerserver interface is displayed in Figure 1.

This interface displays a complete 2D real time view of the field. Although the whole field is visible, only those objects within the vision cone of the agent are displayed. Also, in Robosoccer, perception of far away objects is noisy. Thus, objects are displayed as probability circles: the radius of the circle depends on the radius of the object and the distance to the object. In Figure 1, the ball is represented as a probability circle. Different colors are used to differentiate the ball, the opponents, and the same team players. Those objects which are no longer visible are represented at the last position they were seen.

In order to improve playability, not all Soccerserver commands are available to the player. For instance, it is possible to kick the ball with any strength, but the interface only allows a standard kick. The commands allowed by the

0-7803-9363-5/05/$20.00 ©2005 IEEE.

interface are:

- turn left/right: the player can turn the agent's body a discrete amount (10 degrees).
- run slow/fast: only these two kinds of dash are allowed. Their power is 60 and 99, respectively.
- kick ball: kicks the ball in the direction of the agent's view line with a strength of 60
- kick to goal: kicks the ball towards the goal, with a strength of 99

2.2 Opponent Agents

In the most complex situations, the human player will play within a team against a complete opposite team. The first goal of this paper is to determine whether human input/output modelling of persons works in principle in the Robosoccer domain. To achieve this, we have used a simpler situation where a single human-controlled agent plays against a defensive opposite team [12]. This team is based on zones, where each of the team members is located. Among the agents, the player closer to the ball takes the role of leader. The rest of agents maintain a distance from the leader so as to maintain the formation. Agents determine who is the leader and pass this information to others. When the agent moves away from its zone, it tries to pass the ball to other agent, if available. Otherwise, it continues towards the goal, in order to score. The goalie follows a similar behavior: it stays within its zone and looks for the ball. If it is close (15 units), goes for it and kicks it towards the opposite goal.

The second goal of the paper is to use evolutionary techniques to correct and improve the human model. We have chosen CMUnited99, the winner team of Robocup 1999 [25]. This is a tough opponent, and human players found very difficult to play against it. Therefore, we will be setting an ambitious goal to the evolutionary technique: besides correcting mistakes, it will have to adapt rules learned with a simpler opponent to the new, tougher, opponent.

2.3 Individuals

In this work, the evolving population is made of individuals, each one consists of several rules. The rules follow an `if(situation) then action` structure, where the `situation` checks the values of the agent's sensors and the `action` tells which action the agent will carry out. Actually, the whole set of rules follows a `if-then-else` structure, so there is no need to decide what to do when sevaral rules are activated. Figure 2 displays two actual rules obtained in the course of our research.

With respect to the `if` part of the rules, it is very important to select only the information available to the human player when making decisions. Attributes selected are the X,Y absolute position of the agent, and the distance and angle to the ball, the opponent goal, and the two closest opponent players. There are also attributes to indicate whether these objects are within the view cone, and therefore can be

Figure 2: Example of individual.

```
IF (angle-ball > -37) AND
   (distance-ball > 1.2) AND
   (angle-ball <= 24)
THEN dash99
ELSE IF
   (angle-ball > 19) AND
   (angle-ball <= 42)
THEN dash99
ELSE turn10
```

actually seen. The values that the right hand side of rules (the action part) can take are: kick60 (kick the ball with medium force), kick99 (kick the ball strongly), dash60 (run slow), dash99 (run fast), turn10, and turnminus10. In order to create and modify individuals and maintain syntactical correctedness, the following grammar is used:

- RULE → CONDITIONS ACTION
- CONDITIONS → CONDITION CONDITIONS
- CONDITION → ATTRIBUTE OPERATOR REAL-VALUE
- CONDITION → BOOLEAN-ATTRIBUTE BOOLEAN-OPERATOR BOOLEAN-VALUE
- OPERATOR → > | < | >= | <= | == |! =
- BOOLEAN-OPERATOR → == |! =
- ATTRIBUTE → X |Y | angle | distance-ball | angle-ball | angle-goal1 | angle-goal2 | distance-goal1 | distance-goal2 | distance-opponent1 | distance-opponent2 | angle-opponent1 | angle-opponent2
- BOOLEAN-ATTRIBUTE → ball-in-view | goal1-in-view | opponent1-in-view | opponent2-in-view
- ACTION → dash99 | dash60 | turn10 | turnmenos10 | kick60 | kick99

2.4 Our Evolutionary Algorithm

Our algorithm has some similarities with Evolution Strategies [32] (with respect to the selection method) and Evolutionary Programming [1] (with respect to the representation and absence of crossover operator). It follows a very simple algorithm which is displayed in Table 1. It contains a population of M individuals. For every individual, N new individuals are created by means of diverse types of mutation from the parent. No crossover exists. Then, all the individuals are evaluated by means of a fitness function that involves running a football software agent in diverse situations. Finally, the best M individuals are kept, and so on. Fitness evaluation is very time consuming, so currently we consider only quite small populations. Therefore, we use no policy for allocating more offspring to the best individuals: all of them have the same number of children. Selection is achieved by keeping the best M individuals. So,

Table 1: Evolutionary Algorithm for Improving Control Rules.

CREATE POP with M individuals
EVALUATE POP
LOOP UNTIL termination criterion
  FOREACH $IND \in POP$
    $OFFSPRING = \emptyset$
    REPEAT N times
      $OFFSPRING \leftarrow$ MUTATE(IND)
    EVALUATE $OFFSPRING$
    $POP \leftarrow$ BEST M individuals from $OFFSPRING$

on the one hand, our system is very exploratory, as all individuals in the population reproduce. But on the other hand, small M values put on a lot of selective pressure. This simple approach has produced good results, but in the future, when larger populations are used, we will use reproduction schemes based on fitness and also the crossover operator.

Robosoccer is a noisy domain and the same initial configuration of objects in the field can produce different results. A simple approach to face this problem is to repeat the execution of every training case several times. In this paper, we have usually carried out 2 repetitions. The drawback is that although the fitness value becomes more accurate, the computation time doubles. The reader might want to check [26, 27] for sophisticated approaches for dealing with time-consuming and noisy fitness functions.

2.5 Genetic Operators

The genetic operators consist of various types of mutation, and have been specially tailored to be able to:

- Insert, modify, or remove complete rules. Which rule is modified or removed is determined randomly
- Insert, modify, or remove rule conditions. Which condition is modified is decided randomly
- Fine tuning of numerical values (add/substract a small number. The amount depends on the attribute and it is determined experimentally)
- Gross adjustment of numerical values (add/substract a larger number)
- Modify rule actions

Insertion of new rules, conditions, and actions is done by randomly creating new structures according to the grammar displayed in Section 2.3. This technique has been described in [2]. Modifications also use the grammar to generate new correct individuals. Rules can be removed, but an individual must contain at least 5 rules. This was determined experimentally, because in some cases, the best that the evolutionary algorithm could do is to remove imperfect rules.

2.6 The Fitness Function

In order to determine the quality (i.e. fitness) of the agent controlled by the evolved rules, it has to play one or several matches, and observe its behavior. Simulated play is done in Soccerserver version 7.1.[1] We have adapted the standard Robosoccer Logplayer to store in a file the following data: number of goals (G), time to score the first goal (T), ball location (X_b, Y_b) at the end of the match, agent location at the end of the match (X_a, Y_a). These values will be used to compute other factors, which are:

- D_g: Distance of the agent to the opponent goal at the end of the game
- D_b: Distance to the ball at the end of the game
- N_r: Number of rules

Equation 1 displays the fitness formula to be maximized. The aim is to score as many goals as possible, to minimize the duration to the first score, to minimize the number of rules, to end as close to the opponent goal as possible, and to minimize the distance to the ball at the end of game, in that order.

$$K_1 G + K_2 \frac{T\text{max} - T}{T\text{max}} + K_3 \frac{30 - N_r}{30} + K_4 \frac{52 - D_g}{52} + K_5 \frac{52 - D_b}{52} \quad (1)$$

The weights have been chosen so that $K_1 >= K_2 + K_3 + K_4 + K_5, K_2 >= K_3 + K_4 + K_5, K_3 >= K_4 + K_5$. Some values that seem to work well are: $K_1 = 50\%, K_2 = 30\%, K_3 = 12\%, K_4 = 3\%, K_5 = 5\%$.

However, fitness evaluation requires more than letting the agent play for some time. Robosoccer is a noisy domain, so every training has to be repeated several times to diminish chance results. More importantly, the controlling rules are required to be general, so the agent has to play under different situations.

3 Training the agent

To produce the agent, we have chosen a relatively complex task where the agent (a striker) must look for the ball, get close to it, conduct it to the goal, dribble 3 defences and 1 goalie, and score. It would have been desirable to use worthy opponents, Robocup champions like CMUnited [25] or FC-Portugal [28]. However, the human player found impossible to beat them. This was due to these teams playing very well, but also to the interface not being responsive enough. We have chosen a challenging but beatable Robosoccer team [12], which has the advantage that although their players have a team behavior, we can use as many players as desired. In this case, only 4 of them were used.

It must be remarked that, although the [12] team is not a Robocup champion, the human generated agent has many more difficulties compared to the team:

[1] Here, we have used a newer version of the Soccerserver than the one used for the interface described in Subsection 2.1. No problems were observed.

- The opponents outnumber our agent and play cooperatively
- The opponent can use more actions (turning the neck, turning left and right, kicking and dashing with any power and angle, ...).
- The opposite team has a goalie, whereas our agent must defend and attack

Therefore, it would be a success if the human-generated agent could score against them and even win some of the matches.

The human played a match against the opponent team and 24594 instances were generated. 164 rules were obtained by using the PART algorithm from the Weka tool [15]. Classification accuracy from a 10 crossvalidation is 92.65%. The agent was able to find the ball on the field, to conduct it towards the goal, to score, to dribble opponents, and to steal the ball. The scores in five testing matches were: 5-4, 2-4, 3-4, 4-5, 2-4, 3-4. One match was won and 19 goals were scored, versus 25 scored by the opponent. The learned rules were then pruned to 69 rules (91.90%). Similar results were observed in five new matches: 3-4, 2-4, 3-3, 2-5, 3-1, 4-3. 2 matches were won and 1 drawn. 17 goals were scored versus 20 goals scored by the opponent. Although modest, these results show that the modelling approach is able to construct working agents able to score against a team. [2]

Although the behavior of the agent is not as optimal as the human, it is quite human-like and manages to dribble opponents and steal the ball some of the time, just like the human player did. But the behavior is far from perfect. Instead of carrying more exhaustive analysis and improving the modelling approach to get better results, we will employ evolutionary techniques to improve the set of rules and to adapt them to more powerful opponents,

4 Correcting and Improving the Agent

The goal of this section is to correct and improve control rules obtained in the previous section. In this case, the CMUnited99 team was used as opponent. This team is much more difficult than the team used in the previous section (it is the winner of Robocup99). It was expected that the evolutionary technique would be able to improve the agent's style of play, and score some goals against CMUnited99. Four experiments of increasing difficulty were carried out. The main configuration parameters are displayed in Table 2. M and N are the population size and the number of offspring per individual. In experiments 1 and 2, single individual populations have been used, whereas experiments 3 and 4 tested larger populations to deal with the complexity of the task. With larger populations, a small number of generations (gen) was used to limit the total computation time. Every training case was executed twice to reduce chance effects (the fitness value is the average of the two), except in the last experiment, to limit the computational effort. Every training case was run for 15 seconds. This value seemed to offer good results experimentally. Total Time is the time required by the system from the first to the last generation. The training cases (positions of objects in the playing field) were designed and created by hand, bearing learning in mind.

Table 2: Configuration parameters of the four experiments carried out

Exp.	M	N	Gen.	Training Cases	Reps.	Training Time	Total Time
1	1	1	200	2	2	15 sec.	3h 33'
2	1	1	200	3	2	15 sec.	8h 40'
3	10	2	50	3	2	25 sec.	41h 35'
4	20	1	27	3	1	15 sec.	21h 30'

A summary of results of the four experiments can be seen in Table 3. Initial training fitness is the fitness of the initial set of rules measured with the training cases. Final training fitness is the fitness of the best individual after evolution. Testing has been measured by counting the number of goals scored on 30 cases generated randomly, with different situations for the agent, the ball, and 1 opponent goalie and 1 moving defence. These 30 cases are the same for the four experiments. The initial set of 69 control rules (the ones obtained in the previous section) scored no goals against CMUnited99, out of the 30 opportunities.

From the first experiment, 69 evolved rules are obtained. Training cases involve a single goalie from the CMUnited99 team. Training fitness increases from 14.89 to 27.70. But the testing value shows almost no variation (only one goal is scored, out of 30 trials).

The set of rules obtained in the first experiment are used as the starting point of experiment 2. Three training cases were created, where the agent starts at the center of the field, must get close to the ball which is in its own field, turn and drive the ball towards the opponent field, dribble the defence and the goalie, and score. Only 1 goalie and 1 static defence were present in the training cases. Although training fitness improves from 15.40 to 28.61, no large improvement is observed in testing: only 2 goals were scored, out of 30 chances.

In experiment 3, the size of the population was increased to 10. We will try to improve the set of rules from experiment 2. Again, in the training cases there is 1 goalie and 1 static defence. Three more training cases were created. Training fitness improves from 24.44 to 41.96. This time, learning transfers to the testing cases, and 9 goals are scored out of 30 opportunities. Compared to experiment 2, results are statistically significant at the 1% level. [3]

In experiment 4, the size of the population was increased to 20 and the set of rules obtained from experiment 3 were used for seeding the initial population. Again, one oppo-

[2] In order to draw statistically rigorous conclusions, more matches should have been played. However, our purpose in this paper is not to use the modelling approach by itself, but to be used as seed for an evolutionary algorithm.

[3] We have followed the hypothesis testing procedure based on binomial distributions explained in [23]. Every testing case is considered a trial, with two possible outcomes: scoring or not. As the number of trials is 30, the binomial can be approximated by a normal distribution.

nent goalie and defence are used in the training cases, but this time the defence can move. A large increase in training fitness is achieved (from 34.28 to 60.80) and results in the testing cases improve (17 goals out of 30 are scored). Compared to experiment 3, results are statistically significant at the 1% level. Visually, the agent plays much more smoothly than the initial set of rules, although this is difficult to quantify. There is also a significant decrease in the number of rules (from 72 to 31), even though the style of play has improved.

Table 3: Training fitness before (initial) and after (final) evolution. The best individuals were given 30 opportunities to score, for testing purposes.

Exp.	Initial Training	Final Training	Goals (Testing)	Rules
1	14.89	27.70	01/30	69
2	15.40	28.61	02/30	70
3	24.44	41.96	09/30	72
4	34.28	60.80	17/30	31

5 Related Work

One of the first attempts at modelling opponents was that of [10]. Here, the goal was to learn finite automatons (DFA) consistent with the behavior of the opponent. The model was used to improve a mini-max search algorithm. This approach was valid only for discrete, round-based games, and required complete non-noisy information. Prediction was the goal, but such models could be used to imitate the opponent. Machine Learning has been used extensively since then in classical and strategic games. [13] contains a good survey with some discussion of opponent modelling.

Behavioral cloning is an attempt to imitate other agents behavior [5]. [31] uses this technique to learn from a human piloting a simulator, from whom input/output traces are obtained. It differs with our work because the testing domains is different and no information about the current goal is supplied to the learning system. KNOMIC [39] is an approach that learns to imitate an expert from input/output traces in dynamic, non-deterministic, and noisy domains. It uses a rich knowledge representation based on SOAR rules. The authors claim that KNOMIC can learn from observing a human in difficult domains such as air combat and Quake II. It differs from our work in that we do not use annotations to help learning. [6] presents an overview of imitation in the field of robotics. However, most work is centered around the seeing sequences of actions and then replicating them exactly, which in robotics is a difficult task. Not much generalization is involved.

An area were human behavior ML modelling has been the focus is that of User Modelling [40]. Early work in this field was about student modelling, which seeks to model the internal cognition of a student's cognitive system (as in [9]). However, [33] casted some doubt on the tractability of the cognitive approach. Since them, many researchers have followed a different approach that models an agent in terms of the relationships between its inputs and outputs [19, 11].The common testing ground is modelling students learning subtraction. In contrast with the Robosoccer, this is a discrete and static domain. Currently, the demands of electronic commerce and the Web have led to a fast growth in research in information retrieval, where ML can be used for acquiring models of individual users interacting with information systems [8] or grouping them into communities with common interests [24]. Their domain and purpose is very different to ours.

ML techniques have also been applied to the prediction of user's actions, also called plan recognition. Kautz and Allen [16] defined it as the problem of identifying a minimal set of top-level actions that are sufficient to explain the observed actions. Most work learns hierarchical plans from user logs, and associate them with the goal the user was trying to achieve [7]. Later, plan libraries can be used to match actual user actions with a plan in the library and determine the user intentions. Some research deals with agents modelling opponents and using this knowledge to beat them. In some cases, models are not learned, but predefined and used to classify opponent teams (not individual agents) by means of a similarity metric [29]. [21] proposed a ML scheme to take advantage of prediction of opponents based on visual observations in the Robosoccer simulator. But these models cannot be used to imitate opponent behavior. In all these cases, models are not used for imitation.

[30] uses ML for a coach agent in the Robosoccer domain. That is, here learning takes place at a more strategic higher level. The coach can learn from previous games three kinds of models about teams: team formations and passing behavior. These models can be used to predict and beat the opposite team, or more closely related to our research, so that the team imitates the modeled team. Differences with our research is that whole teams are modeled, and the coach agent is used to observe the playing field.

In [35], a layered learning architecture is proposed. It is designed to use Machine Learning in complex domains, where learning a direct mapping from sensors to actuators is not tractable, and a hierarchical task decomposition is given. Different Learning mechanisms are used to learn behaviors from the bottom level (simple behaviors) to the highest level (more complex, strategic behaviors). The architecture is instantiated in the Robosoccer to learn an individual skill (ball interceptation), a multi-agent behavior (pass evaluation and selection), and adapting the team behavior (here, a new reinforcement learning algorithm called TPOT-RL is used [37]). No imitation experiments have been carried out under this architecture.

[34] is the first reported work (to our knowledge) where data collected from players playing a dynamic video-game, is used to train an agent. In this case, data was collected from humans playing Tron over the internet and a neural network was trained. The approach is similar to our work, but the domain (Tron game) is simpler. Recent research has shown a lot of interest in First Person Shooter games (FPS), like Quake. In this kind of games, human players must react to situations very quickly. So, it is a very suitable environment for learning reactive behaviors[38] pro-

vides a good summary of how imitation can be used at all levels (reactive, tactical, strategic, and motion modelling) in FPS games. Their approach to reactive behavior is similar to ours, but many of the skills to be learned in Robosoccer are different to Quake's.

6 Conclusions and Future Work

The aim of this paper was twofold. First, we have applied an input-output Machine Learning approach to model a human playing Robosoccer. An interface was built that displayed to the user the objects in the playing field that could be seen according to Robosoccer rules. This interface allowed the user to send low-level commands (dash, turn, and kick) to the Soccerserver. Input/output instances generated by the human player were used by a propositional Machine Learning algorithm (PART) to learn a model. This model was then introduced into a computer agent, that played in a similar way to the human. The final agent was able to score against a computer team that the human found challenging, but beatable. However, the style of play of the final agent was not as smooth and efficient as that of the human. Also, the human found almost impossible to beat much tougher teams, like CMUnited99.

Therefore, our second goal was to correct the style of play and to adapt the obtained rules to more difficult opponent teams. This way, rules can be obtained for teams which are too hard for the human player. To achieve this purpose, a simple evolutionary technique, close to evolution strategies and evolutionary programming, was used. The initial population was seeded with the sets of rules to be improved. Genetic operators were developed to work on rules. A fitness function that confronted the agent controlled by the rules against the opponent team was programmed. Number of scored goals, time to score, etc were used to measure the fitness of the individual. After many generations, the rules were indeed improved. The agent was able to increase from scoring 0 to 17 goals against a goalie and a defence of Robocup99 winner CMUnited99. Also, the number of rules was drastically reduced.

In the future, we will improve our two pieces of work. First, the modelling approach suffers from some weaknesses. The user had to be as consistent and reactive in his style of play as possible. Otherwise, if the human varied too much his style during a match, or if he used a lot his memory or his forecasting abilities (i.e. being deliberative), the learning algorithm would not work well. The reason is that it construct its models based on input-output instances only, and assumes there is no internal state, no memory of the past, and no forecasting of the future . Therefore, the agent should be provided with an internal state (i.e. memory) and higher level cognitive skills (forecasting, planning, computing trajectories, etc) so that the power of the modelling language matches that of the human, as far as possible. Also, it should be easier for the human to play the game, by improving the interface, and by allowing the player to control the agent via higher level actions, each one made of sequences of complex lower level actions (like "dribble", "conduct the ball", or "pass the ball").

It would also be interesting to compare current results from trying to evolve players from scratch. Initial results are not good, although other authors have produced competitive results by means of Genetic Programming [22, 4].

With respect to the evolutionary part, so far, we have used a simple evolutionary algorithm, with small populations and no cross-over operator. Even so, results are positive and very encouraging. We have determined that the size of the population makes a difference and in the future we would like to test the approach with larger populations and evolutionary algorithms closer to Genetic Algorithms (including the cross-over operator). Also, we would like to test the whole approach in more challenging football situations, involving several strikers that could pass the ball between them, more defences, and more difficult teams.

Acknowledgements

This article has been financed by the Spanish founded research MCyT project TRACER, Ref:TIC2002-04498-C05-04M.

Bibliography

[1] *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. Wiley-Interscience, 1999.

[2] Ricardo Aler, Daniel Borrajo, and Pedro Isasi. Learning to solve planning problems efficiently by means of genetic programming. *Evolutionary Computation*, 9(4):387–420, Winter 2001.

[3] Ricardo Aler, Daniel Borrajo, and Pedro Isasi. Using genetic programming to learn and improve control knowledge. *Artificial Intelligence*, 141(1-2):29–56, October 2002.

[4] 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 1998 1999. Springer Verlag.

[5] M. Bain and C. Sammut. *Machine Intelligence Agents*, chapter A Framework for Behavioral Cloning, pages 103–129. Oxford University Press, 1999.

[6] Paul Bakker and Yasuo Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *AISB'96 Workshop in Robots and Animats*, pages 3–11, 1996.

[7] M. Bauer. From interaction data to plan libraries: A clustering approach. In *International Joint Conference on Artificial Intelligence*, pages 962–967, 1999.

[8] E. Bloedorn, I. Mani, and T.R. MacMillan. Machine learning of user profiles: Representational issues. In *Thirteen National Conference on Artificial Intelligence*, pages 433–438, 1996.

[9] J. S. Brown and R. R. Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2:155–192, 1978.

[10] David Carmel and Shaul Markovitch. Opponent modeling in multi-agent systems. In *Adaptation and Learning in Multiagent Systems. IJCAI'95 Workshop*, volume 1042 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 1996.

[11] Bark Cheung Chiu, Geoffrey I. webb, and Mark Kuzmycz. A comparison of first-order and zeroth-order induction for input-output agent modelling. In *Proceedings of the Sixth International Conference*. Springer, 1997.

[12] Fernando Fernandez, German Gutierrez, and Jose M. Molina. Coordinacion global basada en controladores locales reactivos en la robocup. In *Workshop Hispano-Luso de Agentes Fisicos*, pages 73–85, Tarragona, Espaa, 2000.

[13] J. Frnkranz and M. Kubat, editors. *Machines That Learn to Play Games*. Nova Science Publishers, 2001.

[14] S. M. Gustafson and W. H. Hsu. Layered learning in genetic programming for a cooperative robot soccer problem. In *European Conference on Genetic Programming*, pages 291–301, 2001.

[15] Eibe Frank Ian H. Witten. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.

[16] H. Kautz and J.F. Allen. Generalized plan recognition. In *Proceeding of the AAAI National Conference on Artificial Intelligence*, pages 32–37, 1986.

[17] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347. ACM Press, 1997.

[18] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge. In *International Joint Conference on Artificial Intelligence (IJCAI97)*, 1997.

[19] M. Kuzmycz. A dynamic vocabulary for student modeling. In *Proceedings fo the Fourth International Conference on User Modeling*, pages 185–190, 1994.

[20] W. B. Langdon and J. P. Nordin. Seeding GP populations. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin, and Terence C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 304–315, Edinburgh, 15-16 April 2000. Springer-Verlag.

[21] Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo. Predicting opponent actions by observation. In *RoboCup International Symposium 2004 (RoboCup2004)*, Lisbon (Portugal), 2004.

[22] Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *Proceedings of the First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[23] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[24] G. Paliouras, V. Karkaletsis, C. Papatheodorou, and C.D. Spyropoulos. Exploiting learning techniques for the acquisition fo user stereotypes and communities. In *Seventh International Conference on User Modelling*, pages 169–178, 1999.

[25] Manuela M. Veloso Peter Stone, Patrick Riley. The cmunited-99 champion simulator team. In *RoboCup-99: Robot Soccer World Cup III*, pages 35–48, 2000.

[26] A. Di Pietro, L. While, and L. Barone. Learning in robocup keepaway using evolutionary algorithms. In *GECCO 2002: Proc. of the Genetic and Evolutionary Computation Conference*, pages 1065–1072. Morgan Kaufmann Publishers, 2002.

[27] A. Di Pietro, L. While, and L. Barone. Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 1254–1261. IEEE Press, 2004.

[28] L. P. Reis and N. Lau. Fc portugal team description: Robocup 2000 simulation league champion. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV (LNAI 2019)*, pages 29–40. Springer Verlag, 2001.

[29] Patrick Riley and Manuela Veloso. *Distributed Autonomous Robotic Systems 4*, chapter On Behavior Classification in Adversarial Environments, pages 371–380. Springer-Verlag, 2000.

[30] Patrick Riley, Manuela Veloso, and Gal Kaminka. An empirical study of coaching. In *Proceedings of DARS-2002, the Seventh International Symposium on Distributed Autonomous Robotic Systems*, 2002.

[31] C. Sammut, S. Hurst, and D. Kedzierand D. Michie. Learning to fly. In D. Sleeman, editor, *Proceedings of the Ninth International Conference on Machine Learning*, pages 385–393. Morgan Kaufmann, 1992.

[32] H.P Schwefehm. *Numerical Optimization for Computer Models*. John Willey, 1981.

[33] J. A. Self. Bypassing the intractable problem of student modelling. In *Proceedings of the Intelligent Tutoring Systems Conference*, pages 107–123, 1988.

[34] Elizabeth Sklar, Alan D. Blair, Pablo Funes, and Jordan Pollack. Training intelligent agents using human internet data. In *Proceedings of the First Asia-Pacific Conference on Intelligent Agent Technology (IAT-99)*, pages 354–363, 1999.

[35] Peter Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.

[36] Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.

[37] Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999. Also in *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[38] C. Thurau, C. Bauckhage, and G. Sagerer. Imitation learning at all levels of game-AI. In *Proc. Int. Conf. on Computer Games, Artificial Intelligence,Design and Education*, pages 402–408, 2004.

[39] Michael van Lent and John Laird. Learning hierarchical performance knowledge by observation. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 229–238. Morgan Kaufmann Publishers Inc, 1999.

[40] Geoffrey Webb, Michael Pazzani, and Daniel Billsus. Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11(19–20), 2001.