A team-based co-evolutionary approach to multi agent learning

Vanessa Frias-Martinez
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, Mailcode 0401
New York, NY 10027, USA
vf2001@cs.columbia.edu

Elizabeth Sklar
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, Mailcode 0401
New York, NY 10027, USA
sklar@cs.columbia.edu

Abstract

A vast amount of the work developed for learning the roles of agents in a multi agent team has focused on the individual. Each agent learns within a selfish reward system. In this paper, we introduce "adaptable auctions", a cooperative, co-evolutionary mechanism in which agents learn using a team-based reward system with the goal of obtaining the best team for achieving a task in an environment that requires coordination to succeed. The agents use a simple auction mechanism to negotiate their roles dynamically. Each agent bids individually according to their perceptions. The system then chooses the best combination of bids for the team; the chosen bids may not be optimal for each individual, but the system learns to bid as a team and develops the best team-based strategy. Our test-bed is based on the RoboCup Four-Legged Soccer League, and we develop our learning algorithm in a simple, simulated version of this environment.

Keywords: Multi Agent Systems, Auctions, Multiagent Learning, Genetic Algorithms, Co-evolution.

1. Introduction

Socially speaking, humans tend to work better in collaborating groups rather than alone. Adding regulation and communication rules for coordination will improve the output of the group [2]. The same is true for team-based multi agent systems.

Multi agent systems (MAS) has been applied to the field of robotics with increasing frequency over the last 10 years [3]. Soccer robotic teams [4] are a good training ground for a multi agent system: a set of agents, possibly heterogeneous by playing different positions, collaborate in a team and at the same time compete against an opposing team. A very important issue in a heterogeneous multi agent system is the distribution of roles within the group. It is hence necessary to define a mechanism for changing roles within this dynamic environment. Many RoboCup teams use hand-coded solutions to this problem. Here we explore the use of an automated auction mechanism in which the bidding strategy is learned by the players using a co-evolutionary algorithm.

Many other authors have studied automatic bidding strategies for agent-based systems applied to other areas, for example fishmarket auctions [9] and trading agent competitions (TAC) [10] with adaptable single-agent auctions or competitive negotiation scenarios by [11]. We believe that the first proven application of auctions in a physical multi robot system was developed by [5].

The field of Multiagent Learning is described in [12] and in [13] as a fusion between Multiagent Systems and Machine Learning (ML). Applying ML techniques to MAS allows us to build evolving agents: agents that learn from and adapt to their experience and their interactions with their environment. Learning techniques have usually been applied to one agent at a time[1], even if it is part of a multi agent system. Some research into learning as a team within an MAS has been done by [15] and [14].

The work developed by [15] is one of the early attempts at demonstrating the utility of self-organization in an agent system driven to achieve a common objective. Our environment and approach center around a multi agent approach where agents are inherently selfish, but must learn to act as contributing members of a team.

[14] applies reinforcement learning techniques to a multi agent box pushing system. In this case, the agents have to work as a team to move a box. Our approach is different in the sense that we use auctions for the role distribution and genetic algorithms for learning to allow adaptation over time in the highly dynamic soccer domain.

We define *adaptable auctions* in a multi agent environment as auctions where agents' bidding strategies are not fixed but improve over time in order to enhance agents' performance at a certain task. The improvement is achieved through the use of a genetic algorithm (GA), tuned using a co-evolutionary learning process. We demonstrate the use of adaptable auctions in a simulated robot soccer environment. The auctions represent the bids made by the agents in order to take on a specific role at a given time during the game play. The system learns to choose the best allocation of roles using the GA.

Our approach is novel in combining both adaptable auctions with a real-time, dynamic multi agent environment as well as our focus on team-based learning where the fitness function is based on the performance of the team rather than the performance of the individual.

The paper is organized as follows. In section 2, we describe SimRob, our simulation environment for bidding and playing out soccer games. Section 3 describes the genetic algorithm that we have used and its paradigm. Section 4 describes some early experimental results, and section 5 contains the discussion of the set-up and future work.

2. SimRob: our soccer simulation environment

Our overall direction with this work is the development of methodologies for organization and coordination of agents in team-based heterogenous multi agent systems operating in dynamic, real-time environments. Currently, we are working with the Sony AIBO robots and the RoboCup Four-Legged Soccer League. We have developed a high-level abstract simulation of the coordination scheme within our AIBO team in order to rapidly prototype different strategies and interaction mechanisms. The experiments described here have all been run using this simulator, which we call *SimRob* [8]. The choice to use a simulator instead of real robots for this phase of the work is due to the fact that we are working with genetic algorithms, which typically take many hours to converge on optimal values. The role that SimRob plays in our methodology is that of a rapid prototyping environment where we can iterate back and forth between development of possible schema in simulation and evaluation of the schema on the physical robots.

The skeleton of our simulator is divided into components, as illustrated in figure 1. We describe three of the components here and detail the learning component in the next section (3):

- **Agent strategy.**
  This component consists of the development of a bidding strategy for each agent which will operate based on calculations of a set of perception parameters for each agent in the field. By perception parameters, we mean values calculated from the sensory data of the robot that indicate the current state of the soccer field.

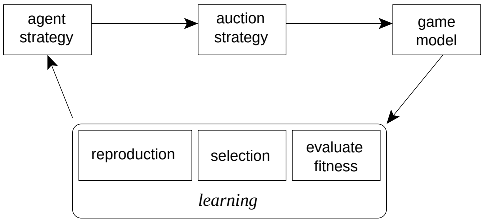

Figure 1. SimRob high-level architecture.

  The agent must construct a bidding strategy such that at any point in the game, based on the current values of its perception parameters, it will use that strategy to bid for a certain role to play at that moment. Here is where the GA is applied so as to learn the optimal role to associate to each set of perceptions. The bid is not specified by the programmer, but automatically learned by the genetic algorithm.

- **Auction strategy.**
  This component consists of the development of an auction clearing strategy for a simulated "auctioneer". The auctioneer takes each agent's bid for certain roles to play in the game, coordinates those bids and returns roles to each agent. In the agent parameter component, the agent bids depending on its perceptions on the current state of the field (opponent seen, where is the ball, is the goal too far away?, etc.). The auctioneer clearing is fixed manually, but we plan to implement this strategy also using GAs as a future work. This auction represents the application of our multiagent learning approach.

- **Game model.**
  This component represents the game play itself. Once the roles have been distributed among the players of each team, the game is simulated. The simulator confronts two equal teams. In order to obtain real results, one team plays against itself.

3. Cooperative co-evolutionary learning

The agents on the team learn their bidding strategies over time by playing many games and evaluating the results of the games in relation to the bidding strategies used. We use the term *cooperation* because the agents share information (i.e., sensory data) amongst team members. The term *co-evolution* refers to evolving individuals for different roles where the *fitness* of an individual (or a team) is based on that individual's (or team's) performance compared to that of another individual (or team) operating in the same environment.

In our case, we are using team-based co-evolution, thus we consider the performance in the environment to mean the result of one team playing a (short) series of soccer games against another team.

3.1. Bidding strategy framework

Each agent constructs its bidding strategy according to the following framework. We have defined three different roles that agents can bid for: *primary attacker* (PA), *deffensive supporter* (DS) and *offensive supporter* (OS). Note that the role of *goalie* is always fixed to one particular agent.

The agents construct a bidding strategy according to their perceptions of the state of the soccer field. For initial experimentation purposes, we greatly simplify the perception state to the values: (1) can I see the ball? (2) minimum distance to the ball, and (3) minimum distance to the goal. These relative perceptions are represented by a 3-digit *percept code*, containing one bit per perception, which is set to 0 or 1 depending on the presence of the perception in the field. Table 1 illustrates the eight possible relative perceptions and corresponding percept code.

ball seen?	closest to ball?	closest to goal?	percept code
no	no	no	000
no	no	yes	001
no	yes	no	010
no	yes	yes	011
yes	no	no	100
yes	no	yes	101
yes	yes	no	110
yes	yes	yes	111

Table 1. Percept code definitions.

For each of the $2^3 = 8$ possible sets of perceptions, the agent's bidding strategy defines a bid where a bid consists of a preference order of the three roles described above for each possible percept code value. Given three roles, six orderings are possible. Table 2 contains the possible preference orders and the corresponding bid code.

Thus a single bid can be represented by 8 values, where each value is between 0 and 5. In our genetic algorithm (described below), this translates into a 24-bit binary string, grouped into 8 3-bit values, where each 3-bit value is between 000 (0) and 101 (5).

role ordering	bid code
PA-OS-DS	0 (000)
PA-DS-OS	1 (001)
OS-PA-DS	2 (010)
OS-DS-PA	3 (011)
DS-PA-OS	4 (100)
DS-OS-PA	5 (101)

Table 2. Bid code definitions.

For each of the (2^3) 8 possible sets of perceptions p_i, $0 <= i < 8$, the agent can bid for a certain role (PA, OS or DS). Our SimRob environment, allows the agent to do a weighted bid of three different roles. This means that the agent can bid for more than one role defining a bid-list that contains the prefered roles in decreasing order of preference. With this setup, the search space of all possible bid strategies for one agent (for the 8 perceptions and the 6 different role bids) is[1]:

$$VR(6,8) = 6^8 = 1.6 * 10^6, \text{ possible bids} \tag{1}$$

These calculations are for only one agent. Each team is composed of three agents (plus a goalie). This means that each of the possible bids that an agent can make are going to be combined with two others to make a team bid. In these terms, we have more than a million possibilities to be combined by a team of three:

$$VR(1.6 * 10^6, 3) > 10^{18}, \text{ possible bid teams} \tag{2}$$

Given this combinatorial explosion within the bidding space even within our highly simplified experimental framework, it is clear case where an evolutionary learning algorithm can be used to identify strong bidding strategies.

3.2. Genetic Algorithm Paradigm

Our genetic algorithm, “GenRob”, plays a series of matches between two teams in the SimRob simulator. Each team consists of three players and a goalie, and our focus is to build the best team, that is, the one that either scores the most, or at least, puts the ball nearest to the opponent's goal within the time limit of the game. The fitness of the players is measured in terms of optimal play of the team as a whole (of which the agent is part of), and not in terms of an optimal individual player.

The genetic algorithm begins by randomly initializing bidding strategies for a population of n players. At each generation, we randomly select six players (3 players per team) from this population. Then the two teams play a series of games against each other, called a “round”. Each round consists of g games; each generation consists of r rounds.

For the experiments described below, we used $g = 5$ games per round. The games are played for a limited amount of simulated time, and after each game, the fitness

[1] where $VR(n,p) = n^p$ is the formula for computing the number of variations with reputation of selecting an ordered set of p elements from a set of possible n elements

of the three agents belonging to the winning team is increased. The best team is chosen after each round. A team is considered better than another one when their fitness measure is better. In our case, the fitness is measured per game, and then summed up for each round. After 'r' rounds, a new generation is obtained with GenRob, our soccer genetic algorithm.

If we have an initial population of 'players' players, and we run GenRob, we have to choose 6 agents(3 per team) from the total population. Once we have chosen one team, the other will be chosen among half of the initial set.This means that we have:

$$combinations(players/6)/2, \text{ possibilities} \qquad (3)$$

If we let this selection to be completely random, we may be testing the same team several times, or the same players. We propose the use of table 3 with all the posible combinations of teams (according to value in formula 3). Every time we select two teams, we will read this selection from the table instead of chosing the players randomly. This definitely speeds the convergence of GenRob.

g1	ABC	DEF
g2	ABD	CEF
g3	ABE	DCF
g4	ABF	DEC
g5	ADC	BEF
g6	AEC	DBF
g7	AFC	DEC
g8	DBC	AEF
g9	EBC	DAF
g10	FBC	DEA

Table 3. Games List: all possible teams and their opponents for a 6-player population.

4. Results

We have designed a population of 6 randomly generated agents. We use the combination of teams explained in 3.

This means that we will have $6choose3 = (6,3)/2 = 10$ different team configurations possible (see table 3). In order to analyze all 10 possible configurations, we perform a complete search, playing a round with each of the 10 possible team combinations. Hence, each generation will have $10rounds = g * 10games$. At the end of each round, we increment the fitness of each player from the winning team, i.e., the team that has won more than $g/2$ games.

After the rounds for each generation are complete, we enter the *selection* phase of the genetic algorithm. We select to use a $50 : 50$ exploitation:exploration strategy, keeping the 3 best players and reproducing 3 new players using the first three as parents, employing mutation and crossover as operators. Each of the new players come from one of the best players and a 1 point mutation to avoid big jumps in the new population generated.

We conducted experiments with these parameters for several different lengths of game time. In the graphical display, we can see how the teams evolve so that the distances of the ball to the opponent's goal decrease over time towards zero. Depending on the length of the game, we can see that for longer games, the progression towards zero distance to the goal is smoother.

Figures 2, 3 and 4 show the distance of the ball to the opponent's goal (for the better team). The x-axis contains the generations of the simulation(measured in time ticks of the simulator), the length of the generation depends on the length of the game. Each length is specified under its graph. The y-axis is the distance between the ball and the opponent's goal at the end of the game, measured in "cells". To aid our localization algorithm (not discussed here), we have divided the soccer field into 54×27 square cells.

Figure 2 shows the learning of the team for games of length 70 ticks. The coevolutionary learning is shown for three different random seeds. We have also added a polynomial curve fitting function for each of the representations in order to see clearly the convergent learning. As we can see, the team learns to take the ball to approximately one cell (or less) away from the goal.

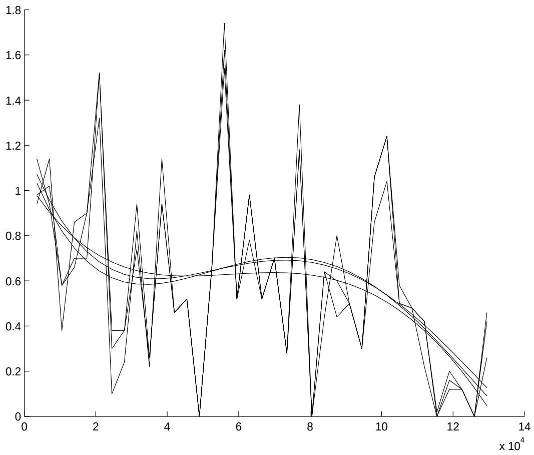

Figure 2. Co-evolution over 37 generations for games of length = 70 ticks.

Figure 3 shows the learning for shorter games, of length 15 ticks, and its corresponding polynomial curve fitting. In this case, the distance of the ball to the goal is bigger (a median of 7 cells), and the learning is more unstable than for longer games. Still, we can see that for the same number of generations, we obtain distances no bigger than 10 cells to the goal, which means that at least the best team we are obtaining is keeping the ball in the opponents' half of the field. Thus, we have obtained an "attacking team".

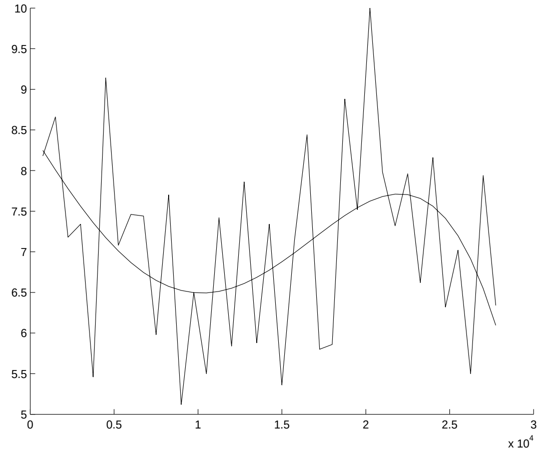

Figure 3. Co-evolution over 37 generations for games of length = 15 ticks.

From the results obtained, we can say that it makes sense that for shorter lengths the learning is more difficult, since there is not enough time for the team to apply its game strategy. Hence, in order to obtain significative team-learning results, we recommend running games of length 70 ticks (and longer) with our current game strategy.

In order to demonstrate the usefulness of the learning algorithms, we have also played games without learning (that is, agents bidding a fixed and not learned value). With these "control" runs, we have obtained distances of the ball to the goal much higher than any of the learning experiments developed.

Figure 4 shows the distance of the ball to the goal for both a coevolutionary learning team and a random team. We define random team as one in which the agents bids are made randomly without any learning strategy. As we can see, random games do worse that learning games in the sense that they do not take the ball as close to the goal as learning teams do. Figure 4 also contains the polynomial curve fitting that shows the convergent behavior of the learning team versus the non convergence of the random team.

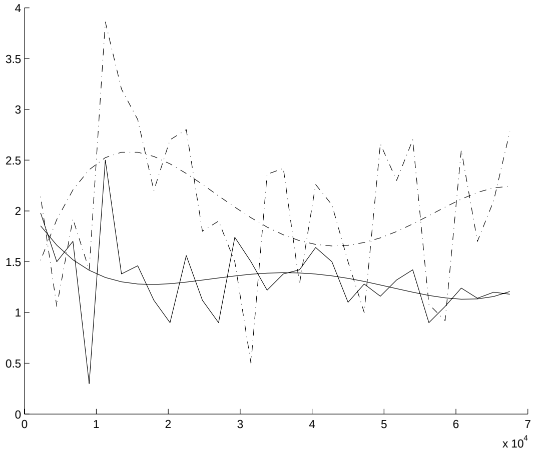

Figure 4. Co-evolutioned vs Random Generations for games of length = 45 ticks.

The important result is that we have obtained convergent behavior (convergence speed depending on the length of the game). This means that our team is learning to bid so as to make their team strategy be the winner, that is, scoring or at least taking the ball as close to the goal as possible.

5. Discussion and Future Work

In this paper we have designed a learning algorithm for a multiagent system to maximize its performance as a team. In our case, we have used as testbed a simulated RoboCup Four-Legged Soccer League environment with a team of four agents playing soccer against another team of 4, with the objective of scoring the maximum number of goals within a fixed time period. We have demonstrated that genetic algorithms help us to learn how to negotiate role distribution in order to obtain the best team results.

We plan to use our best co-evolved teams on real AIBO robots for playing games in upcoming RoboCup tournaments. Here we will see the output and the improvement of our game by opposing it to another random generated team.

We also plan to develop a GA for the goalies, that is, the agents responsible for the distribution of the roles once the bids are done. The policies to be learned to assign roles, are also a problem solvable by means of GAs.

References

[1] Balch T.: Learning Roles: Behavioral Diversity in Robot Teams. (1996)

[2] Reicher S.D., Turner J.C.: Rediscovering the social group: a self-categorization theory. Oxford: Blackwell. 1987.

[3] Stone P., Veloso M.: Multiagent Systems: A survey from a Machine Learning Perspective Autonomous Robotics, volume 8, number 3. July 2000.

[4] Kitano H., Asada M., Kuniyoshi Y.,Noda I., Osawa E. The robot world cup initiative. In Proc. Autonomous Agents 97. ACM. Marina del Rey, California.

[5] Mataric M., Gerkey B.: Sold!: Auction methods for multirobot coordination IEEE Transactions on Robotics and Automation,volume 18,number5, october 2002.

[6] Dias M.B., Stentz A.: A free market architecture for distributed control of a multirobot system In Pro. Autonomous Agents, Marina del Rey, CA, Feb. 1997,pp.323-331.

[7] Thayer S., Digney M., Dias M.B., Stentz A.: Distributed Robotic mapping on extreme environments. In Proc. SPIE, vol 4195,Mobile Robotx XV and Telemanipulator and Telepresence Technologies VII, Nov.2000.

[8] Frias-Martinez V., Sklar E., Parsons S.: Exploring auction mechanisms for role assignment in teams of autonomous robots Robocup International Symposium, 2004.

[9] Rodriguez-Aguilar J.A, Martin F.J., Noriega P., Garcia P., Sierra C.: Towards a test-bed for trading agents in electronic auction markets. AI Communications. 2001.

[10] Stone P., Littman M.,Singh S., Kearns M.: ATTac-2000: An Adaptive Autonomous Bidding Agent. Journal of Artificial Intelligence Research. pp. 189-206, 2001.

[11] Zeng D., Sycara K.: Bayesian Learning in negotiation. In Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium, pp.99-104. AAAI Technical Report SS-96-01.

[12] Weib G.: Distributed reinforcement learning. Robotics and Autonomous Systems,15, 135-142.

[13] Stone P., Veloso M.: Towards Collaborative and Adversarial Learning: A case study in robotic soccer. International Journal of Human Computer Studies, 48(1):83–104, Jan.1998.

[14] Sen S., Sekaran M.: Learning to Coordinate without Sharing Information Proceedings of the Twelfth National Conference on Artificial Intelligence, pp.426-431, Washington 1994.

[15] Nagendra M.V., Lesser V., Lander S.: Learning Organizational Roles in a Heterogeneous Multi-agent System,. 1995.