Exploring auction mechanisms for role assignment in teams of autonomous robots

Vanessa Frias-Martinez[1], Elizabeth Sklar[1], and Simon Parsons[2]

[1] Department of Computer Science
Columbia University
1214 Amsterdam Avenue, New York, NY 10027, USA
`vf2001,sklar@cs.columbia.edu`

[2] Department of Computer and Information Science
Brooklyn College, City University of New York
2900 Bedford Avenue, Brooklyn, NY 11210, USA
`parsons@sci.brooklyn.cuny.edu`

Abstract. We are exploring the use of auction mechanisms to assign roles within a team of agents operating in a dynamic environment. Depending on the degree of collaboration between the agents and the specific auction policies employed, we can obtain varying combinations of role assignments that can affect both the speed and the quality of task execution. In order to examine this extremely large set of combinations, we have developed a theoretical framework and an environment in which to experiment with and evaluate the various options in levels of collaboration and policies. This paper describes our framework and experimental environment. We present results from examining a set of representative policies within our test domain— a high-level simulation of the RoboCup four-legged league soccer environment.

1 Introduction

Multi agent research has recently made significant progress in constructing teams of agents that act autonomously in the pursuit of common goals [11, 14]. In a multi agent team, each agent can function independently or can communicate and collaborate with its teammates. When collaborating, the notion of *role assignment* is used as a means of distributing tasks amongst team members by associating certain tasks with particular roles. The assignment of roles can be determined *a priori* or can change dynamically during the course of team operation.

Collaboration enables a team of agents to work together to address problems of greater complexity than those addressed by agents operating independently. In general, using multiple robots is often suggested to have several advantages over using a single robot [4, 7]. For example, [10] describes how a group of robots can perform a set of tasks better than a single robot. Furthermore, a team of robots can localize themselves better when they share information about their environment [7]. But collaboration in a team of robots may also add undesirable delays through the communication of information between the agents.

We are exploring — within dynamic, multi-robot environments — the use of auction mechanisms to assign roles to agents dynamically and the effect of different approaches

to collaboration. In particular, we are studying the effect on both of the above on the effectiveness of teams of soccer players. Varying the degree of collaboration between the agents and the specific auction policies employed, we can obtain an extremely large set of possible mechanisms for role assignment that can affect both the speed and the quality of task execution. In order to evaluate this set, we have developed a theoretical framework and a simulation environment. The theoretical framework helps us to identify the space of possibilities, and the simulation environment helps us to evaluate the various options.

Our simulation environment allows us to perform a systematic study of the different strategies that we may want our group of robots to follow in order to accomplish certain tasks. It allows the user to define the degree of collaboration between agents (for now we restrict collaboration to the sharing of information) and to define different bidding techniques and auction policies. The user can use simulation to explore the range of strategies, and we are currently implementing tools that use learning to automate this process.

This paper begins by highlighting some background material on auctions and the use of auction mechanisms in multi agent systems. Then we describe our theoretical framework, which provides the basis for a systematic exploration of the vast space of possible role assignments. Next we detail our experimental environment — a high-level simulation of the RoboCup four-legged soccer league. RoboCup soccer is a very good testbed for collaboration algorithms and role assignment and in general for all kinds of algorithm applied to any distributed intelligent system. While the work described in this paper was developed using a simulator, our longterm plan is to integrate its use into our RoboCup four-legged team[1]. We present results of simulation experiments evaluating both collaborative and non-collaborative models of information sharing as well as various auction policies. Finally, we close with a brief discussion and directions for future work.

2   Auctions

Following Friedman [8], we can consider an *auction* to be a mechanism that regulates how commodities are exchanged by agents operating in a multi agent environment. Two characteristics of the exchange are important from our perspective. First, the exchange typically deals with two kinds of commodity — indivisible commodities (usually known as *goods* in the economics literature) and divisible commodities (usually money). Second, agents reach an agreement by passing messages that identify how much of the divisible commodity will be exchanged for the indivisible good.

The amount of divisible commodity (money) they choose to trade is related to the value the agent places on the indivisible commodity (the good). This value, which is usually not provided to other traders (though they may infer it from the agent's bids), is known as the trader's *private value*. If a buyer pays more than its private value, or a seller accepts less, then we consider that the agent is trading at a loss.

An *auction mechanism* defines how the exchange takes place. It does this by laying down rules about what the traders can do — what *messages* they can exchange in an

---

[1] http://agents.cs.columbia.edu/metrobots

interaction — and rules for how the allocation of commodities is made given the actions of the traders. When the exchange occurs, the process is known as the market *clearing*. An *auctioneer* is an agent that effects this process and clears the market using a *matching policy* that matches sellers with buyers and sets the prices at which the exchanges of commodities occur.

Associated with each trader is a component that shows the change in the amount of money and goods that they hold. Traders with positive goods components are *buyers*; those with negative goods components are *sellers*. Typically we will deal with so-called *one-way* traders, meaning that traders are either buyers or sellers but not both. The traders' messages (generically called *offers*) include some price information; this information may be an offer to buy at a given price, in the case of a *bid*, or an offer to sell at a given price, in the case of an *ask*. *One-sided* auctions allow only buyers or only sellers to make offers; *two-sided* or *double* auctions allow both.

While the most common auctions are those in which physical items are sold for money, the concept of an auction is much more general than this. Auctions have been used in different environments for resource allocation, such as electronic institutions [6], distributed planning of routes [12] or giving roles to a set of robots to complete a common task [9].

3   Theoretical framework

In our auction, there are two types of agents: the *auctioneer* and the trader — a *player* in the RoboCup soccer game. The player makes an *offer* and the auctioneer's job is to coordinate the offers from all the players and perform role assignment. There are five main components to our model.

First, we define $\mathcal{R}$ to be the set of possible roles:

$$\mathcal{R} = \{PA, OS, DS\} \tag{1}$$

where PA is a primary attacker, OS is an offensive supporter, and DS is a defensive supporter. Note that the goalie is not considered a role to be assigned in this manner, since the goalie has to be predesignated and cannot change during the course of the game. The roles are considered to be the indivisible commodity in the auction. Next, we define $\mathcal{P}$ to be a set of player attributes:

$$\mathcal{P} = \{d_{ball}, d_{goals}, d_{mates}, d_{opps}\} \tag{2}$$

where d_{ball} contains the distance from the player (who is making the offer) to the ball; d_{goals} contains the distance from the player to each goal; d_{mates} contains the distance from the player to each of its teammate; and d_{opps} contains the distance from the player to each player on the opposing team.

We interpret these attributes collectively as the divisible commodity a player will trade for an indivisible commodity (a role). As will be seen below, a player can modulate the value of each attribute in an offer, in order to keep the actual value of each attribute private. Third, we define F to be a set of functions which define the method for sharing perception information between agents. The agents this information is shared

with could be teammates, auctioneer, or both. Fourth, we define $\mathcal{M}$ to be a *matching function*, the method used by the auctioneer for clearing the auction, i.e., matching the offers with roles. In other words, the matching function captures the coordination strategy. Finally, we define an *auction*, $\mathcal{A}$, to be:

$$\mathcal{A} = \langle P, R, M, f \rangle \qquad (3)$$

where $P \subseteq \mathcal{P}$ and $P \neq \emptyset$; $R \subseteq \mathcal{R}$ and $R \neq \emptyset$; $M \subseteq \mathcal{M}$ and $M \neq \emptyset$; and $f \in F$. Our work is systematically exploring the space of all possible auctions $\mathcal{P} \times \mathcal{R} \times \mathcal{M} \times F$. $\mathcal{B}$ denotes the set of possible types of offers in a particular auction, $A \in \mathcal{A}$:

$$\mathcal{B} = \{\mathbf{r}, \mathbf{w}\} \qquad (4)$$

where: $\mathbf{r} \subseteq R$ is a set of roles for which the player bids; $\mathbf{w}$ is a set of real-valued weights, one weight corresponding to each of the roles in r (a weight of 0 means that the player is not interested in making an offer for the corresponding role); and $f(p)$, $p \subseteq P$, is the mechanism by which perceptual data is used to determine $\mathbf{r}$ and $\mathbf{w}$. To date, we have defined two different types of auctions within this framework — a *simple* auction [5] and a *combinatorial* auction [3].

In the simple auction, we consider each player as a "seller" whose offer consists of a statement that it is willing to undertake a single role at a value that is based on its temporal perception. The specific $b_t \in \mathcal{B}$ at time t is:

$$b_t = \{r, w\} \qquad (5)$$

where the role r and w are singletons. Note that there is only one role in the offer, this is what makes this a simple auction. The auctioneer collects all the offers from each of the three players on the team needing roles (i.e., everyone except the goalie) and then uses a matching strategy M to make the matches.

In a *combinatorial auction*, an agent's offer consists of a set of roles and the value the agent is prepared to accept for any of the roles. So in our case, an offer consists of:

$$b_t = \{(r_0, r_1, r_2), (w_0, w_1, w_2)\} \qquad (6)$$

where r_i and w_j are singletons. Using different combinations of weights allows the agent to bid for different combinations of roles, and this makes the auction combinatorial [1].

4   The Simulation Environment: RePast

RePast (*REcursive Porous Agent Simulation Toolkit*) [13] was developed by the University of Chicago's Social Science Group. This tool is a software framework for creating agent-based simulations using the Java language[2]. It provides a library of classes for creating, running, displaying and collecting data from an agent-based simulation. In addition, RePast can take snapshots of running simulations and create Quicktime movies of simulations.

---

[2] RePast requires version Java 1.4 or greater.

RePast envisions a simulation as a state machine whose state is constituted by the collective states of all its components. These components can be divided up into *infrastructure* and *representation*. The infrastructure is the various mechanisms that run the simulation, display and collect data and so forth. The representation is what the simulation modeler constructs, i.e., the simulation model itself. The state of the infrastructure is then the state of the display, the state of the data collection objects, etc. The state of the representation is the state of what is being modeled, the current values of all the agents' variables, the current value of the space or spaces in which they operate, as well as the state of any other representation objects (e.g., aggregate quasi-independent "institution" objects). The history of the simulation as a software phenomenon is the history of both these states, while the history of the simulation as a simulation is the history of the representational states. In RePast, any changes to the states of the infrastructural components and the representational components occur through a Schedule object which emulates the passage of time as sequential *ticks*, as in ticks of a clock.

RePast allows a user to build a simulation as a state machine in which all the changes to the state machine occur through a schedule. This provides clarity and extensibility both for the simulation writer/user as well as the software designer seeking to extend the toolkit.

5 SimRob: our Simulated Approach to a RoboCup Game

In order to model a RoboCup soccer game in RePast, we need to define the agents, the environment and the state machine that RePast will execute at each scheduled tick.

5.1 Agent parameters

In order to simulate the RoboCup Legged-League field environment, we define four robots per team and a ball. Each one of the robotic agents is associated with an array containing the values that define their perception and localization —

$$(x, y, \phi, d_{ball}, d_{goals}, d_{opps}, d_{mates}, b_{ball}, b_{goals}, b_{opps}, b_{mates}) \tag{7}$$

where: (x, y) are the 2D coordinates of the robot on the field[3]; ϕ is region of orientation of the robot[4]; d_{ball} is the distance from the robot to the ball, d_{goals} is the distance from the robot to each goal, d_{opps} is an array containing the distance from the robot to each opponent, and d_{mates} is an array containing the distance from the robot to each teammate. The values above are correct, meaning that we can interpret them as correctly calculated, as long as they are really detected by the perception system on the agent.

The agent also contains a set of Boolean variables that determine whether the ball, goal, opponents or teammates have been seen or not. These values comprise the second half of equation (7) and indicate if the ball has been detected by the player (b_{ball}), if each goal has been detected by the player (b_{goals}), if each opponent has been detected nearby (b_{opps}) and if each teammate has been detected nearby (b_{mates}).

---

[3] The field itself is broken down into the same 24×14 grid that we use for localization on the AIBOs.

[4] The $360°$ of orientation are divided into eight $45°$ sections, numbered 0 through 7.

5.2 Simulation skeleton

We use RePast in order to simulate the development of a game with the agents. The simulation is run for the period of time that we desire. When the scheduler of RePast reaches that time, it stops. We will call *runtime* the amount of time that the simulation is executed. At the beginning of the simulation, we define four agents (per team) and a ball in the field. Each of the agents is defined as explained above, by means of an array as in equation (7).

The simulation run in RePast can be divided into the following steps: (1) generation of the agent parameters, (2) definition of the amount of shared information among the agents, (3) definition of bidding policies for the agents, (4) definition of the auction policies, and (5) game development.

Generation of the agent parameters In this first step, we obtain the parameters of each of the agents in the field. The localization of the robot is expressed with the coordinates (x, y) in a 2D field. It is constructed to give a "realistic" sequence of positions during a soccer game and works by moving the agent one cell forward in the grid model every time a movement is made. The ball localization is updated according to the game development. We add a parameter to the behavior of the ball, $bounce$. This variable defines the number of cells that the ball moves back when a wall is hit. Once the coordinates of the ball are obtained, we can calculate the distance to the ball, d_{ball}. In order to calculate the distance to the goals, d_{goals}, we retrieve the coordinates of the agent in the field. Finally, we can calculate the distances to the opponents d_{opps} and the mates d_{mates}.

Amount of information shared by the agents In order to simplify information sharing and make the scenario relatively tractable, we have adapted the following strategy for information sharing between agents. We build a boolean table that contains the perception data p_t at time t and two columns for each of the possible 1 or 0 values of each variable. The tool allows the user to select, via the boolean variables, which of the percepts $p_i \in P$ are going to be shared by the agents in the field. Hence, if the user does not want the agents to share any info all the variables in the table will be 0, 1 if we want all the data to be shared between the agents, any other combination of values will generate a partial-collaboration among the agents.

When the collaborative approach is selected, some common measures can be calculated from the shared information among the users. In this way, the tool allows the user to use three more measures when the perceptions shared allows us to define them:

- $mingoal$ is a boolean variable that is true when the agent is the one closest to the goal. This variable can be defined when the agents share the variable d_{goals} among them.
- $maxopp$ is a boolean variable that is true when the agent is farthest away from the opponents in the field. This variable can be defined when the agents share d_{opps}.
- $maxball$ is a boolean variable that is true when the agent is farthest away from the ball. This value can be defined when the variable d_{ball} is shared among the agents.

Defining a bidding policy for the agents The simulation makes it possible to define which offer should be made by an agent depending on the perception data that agent has. The amount of information shared or not shared (elements $p_i \in P$) defines a set of values that can be represented in a table with two possible boolean values, 0, when perception is false, 1 when perception is true. The user can associate a certain role to each of the possible entries of the table. Hence, for each simulation tick of the game development, the agent's bid will be the role associated by the user to the set of perceptions gathered by the agent at that simulation tick.

Defining an auction policy for the auctioneer The auction is responsible for distributing the roles between the agents on the field. The behavior of the auctioneer can be defined by a boolean table whose entries are the different possible combinations of true and false values of the perception variables $p_i \in P$. Depending on the collaboration policy, those variables will come from the agent itself or from any of the agents of the team if it is a shared parameter. The simulation makes it possible to define each entry in the table as a suitable situation for the role to be assigned or not. In a simple auction, the auctioneer will try to assign the role in the bid depending on the set of perceptions. In a combinatorial auction, the auctioneer will try to assign each of the roles in the bid in decreasing order of weights. The auctioneer will go through the different roles in the bid until one of the roles in the array is assigned to the agent, meaning that the bid is won.

Game Development Once the agent-roles are defined, we have to actually simulate the joint task to be developed by the agents. As stated before, our aim is that of simulating a soccer game. The game model is very simple. Each role has a state graph that will output a certain behavior depending on the perceptions gathered by the agent:

- PA BEHAVIOR: If the agent sees the goal and the ball, then, kick it, otherwise turn to look for the ball without loosing track of the goal.
- OS BEHAVIOR: If the ball is seen, the agent kicks it.
- DS BEHAVIOR: If the ball is seen, the agent follows it in order to avoid that an agent from the opponent team scores.

The output of the state graphs of each role may generate one of the following methods: `turn()`: the robot turns (changes angle of view region 0..7) to look for perceptions; `kick()`: the coordinates of the ball change according to the hit intensity and robot's coordinates; and `followball()`: agent is assigned a position that is as close to the ball as its closest teammate.

Finally, if a goal is scored, the robots are sent back to their initial positions and the ball randomly changes location. Then, the three step (parameter's generation, auction policy and game development) simulation is run again. We consider that a goal is scored when its coordinates are inside a predefined square around to the goal coordinates.

6 Experiments

This section describes our experimental work to date. This has started to explore the range of possible auctions and their effect on the coordination of a team, as measured

Table 1. Non-collaborative simple auction

Ball seen	Opponent seen	Mate seen	Role
0	0	0	DS
0	0	1	DS
0	1	0	DS
0	1	1	DS
1	0	0	DS
1	0	1	DS
1	1	0	OS
1	1	1	OS

Table 2. Matching policy between roles and perceptions.

Ball seen	Opponent seen	Mate seen	Role
0	0	0	PA
0	0	1	PA
0	1	0	PA
0	1	1	PA
1	0	0	PA
1	0	1	PA
1	1	0	OS,DS
1	1	1	OS,DS

by their performance in simulated games. We have experimented with four very simple types of coordination.

6.1 Non-collaborative simple auction

This approach defines a team of agents that don't share any perception data. Hence, each one relies on the information that it gathers independently of the others. The offers made by the agents follow the policy in Table 1. This shows that we have defined the agent to offer to be OS when both ball and opponent are seen. In any other case, our agent will offer to be DS. The matching policy that we have defined is represented by Table 2. This is very simple and just associates a fixed role to each of the possible sets of perceptions.

6.2 Non-collaborative combinatorial auction

In this case there is still no sharing of perception, but the bid now contains a vector defining the agent's role preferences For our experiments, we have defined two different bidding policies. The *offensive* policy, defined in Table 3, represents a team with an attacking approach, always looking for the goal and aiming to score. The other policy

Table 3. Non-collaborative combinatorial auction

Ball seen	Opponent seen	Mate seen	Role
0	0	0	[OS,.7,DS,.2,PA,.1]
0	0	1	[OS,.7,DS,.2,PA,.1]
0	1	0	[OS,.7,DS,.2,PA,.1]
0	1	1	[OS,.7,DS,.2,PA,.1]
1	0	0	[OS,.7,DS,.2,PA,.1]
1	0	1	[OS,.7,DS,.2,PA,.1]
1	1	0	[OS,.7,DS,.2,PA,.1]
1	1	1	[OS,.7,DS,.2,PA,.1]

Table 4. Collaborative simple auction

Ball seen	Opp seen	Mate seen	MinGoal	MaxOpp	MaxBall	Role
0	0	0	0	0	0	[DS]
0	0	0	0	0	1	[DS]
...	.	.	.	.	.	.
1	1	1	1	1	1	[OS]

is more defensive. This one assigns the array of roles [DS,.7,OS,.2,PA,.1] to each of the agents. The matching table is the same as before.

6.3 Collaborative simple auction

In this case, the agents share all the perception data. Hence, when defining the bids, we can also share the three variables related to the minimum and maximum distances to the ball, opponents and goal. The table defining the bidding policy is huge. In Table 4 we show a few lines to give the sense of it, but it is deliberately similar to the policy for the non-collaborative auction to give a reasonable comparison. When no elements are seen by any of the agents, the agent bids for the role DS. When everything is seen and the distances are minimum, the agents bid to be OS. The matching policy is also the same as for the non-collaborative examples.

6.4 Collaborative combinatorial auction

Here the bidding table (Table 5) is similar to the previous one, but contains a vector of bids and weights instead of only one role, and this vector is like that for the non-collaborative combinatorial auction. Again we ran experiments with an attacking bidding policy and a defensive bidding policy, and the matching table is the one used in the previous examples.

Table 5. Collaborative combinatorial auction

Ball seen	Opp seen	Mate seen	MinGoal	MaxOpp	MaxBall	Role
0	0	0	0	0	0	[OS,.7,DS,.2,PA,.1]
0	0	0	0	0	1	[OS,.7,DS,.2,PA,.1]
...	.	.	.	.	.	..
1	1	1	1	1	1	[OS,.7,DS,.2,PA,.1]

Table 6. Results

	unique		not unique	
	offensive bid	defensive bid	offensive bid	defensive bid
noncollab simple	16	–	16	–
noncollab comb	33	43	30	47
collab simple	40	–	67	–
collab comb	49	37	78	67

6.5 Results

Teams using each of the types of coordination described above (including separate of-
fensive and defensive techniques in the combinatorial auction) were run in simulation
against the same, simple, opponent in order to evaluate the effectiveness of the col-
laboration policy. The opposing team moved randomly around the field, but was not
intended as serious opposition, rather it was intended as a baseline against which all
mechanisms could be judged equally. For each coordination mechanism, we ran two
sets of experiments. In one, the "unique" experiments, we made the auctioneer assign
unique roles to agents. In the "not unique" experiments, the auctioneer was allowed to
assign duplicate roles. The average number of goals scored for each of the different
kinds of collaboration are given in Table 6, and plots of the goals scored over time for a
sample game are given in Figures 1 and 2.

In the non unique approach, collaborative teams score almost double the number
of goals of the non-collaborative teams. In the unique role approach, differences in
the score of the games between the collaborative and non-collaborative approaches for
both simple and combinational auctions are not so marked. This is due to the fact that
our matching policy is very demanding and since we do not allow repeated roles, the
auctioneer often ends up distributing roles randomly. In order to prove this last assertion,
we defined a parameter called ratio in the simulation. The ratio is associated to the
acceptance of the bids made by an agent. The higher the ratio, the more times its bid
has been accepted. In the not uniqueness experiments, we obtained very low ratios,
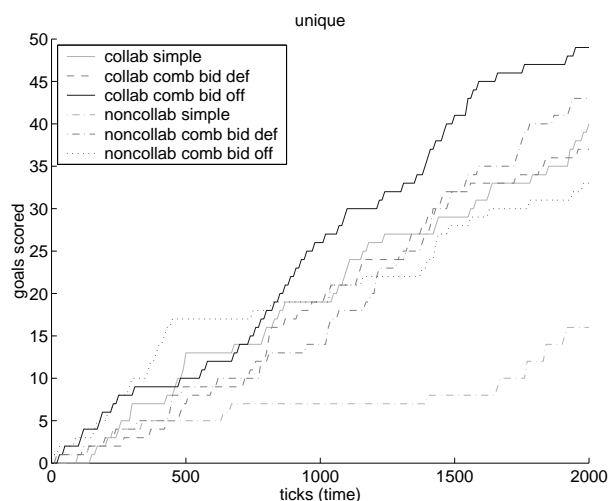meaning that the agents almost never won a bid, and so, the roles were distributed
randomly.

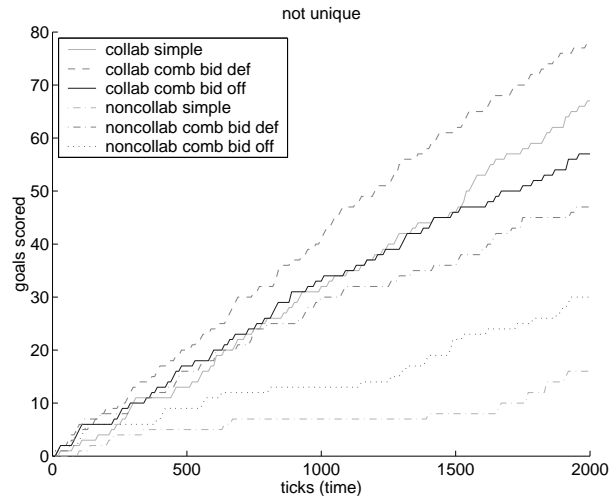**Fig. 1.** Goals scored over the course of a game — unique matching policy

7 Conclusions and Future work

This paper has described our preliminary work in exploring the use of auction mechanisms to coordinate players on a RoboCup team. While this work is only just beginning, we believe that the results demonstrate the potential of the approach to capture a wide range of types of coordination, and to be able to demonstrate their effectiveness through simulation. In addition, this approach makes it simple to explore more complex, and potentially more flexible, kinds of role allocation than have been previously used in the legged-league, for example [2, 15].

Our future work is to build on this foundation, exploring a wider range of possible auctions through simulation, and moving towards using learning techniques to automatically explore the space of auctions. We further intend to implement the most effective bidding and matching policies on our legged-league team.

References

1. C. Boutilier and H. H. Hoos. Bidding languages for combinatorial auctions. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1211–1217, San Francisco, CA, 2001. Morgan Kaufmann.
2. D. Cohen, Y. Hua, and P. Vernaza. The University of Pennsylvania Robocup 2003 Legged Soccer Team. In *Proceedings of the RoboCup Symposium*, 2003.
3. S. de Vries and R. Vohra. Combinatorial auctions: A survey. *INFORMS Journal of Computing*, (to appear).
4. G. Dudek, M. Jenkin, E. Emilios, and D.Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4), 1996.

Fig. 2. Goals scored over the course of a game — not unique matching policy

5. R. Engelbrecht-Wiggans. Auctions and bidding models: A survey. *Management Science*, 26:119–142, 1980.

6. M. Esteva and J. Padget. Auctions without auctioneers: distributed auction protocols. In *Agent-mediated Electronic Commerce II, LNAI 1788*, pages 20–28. Springer-Verlag, 2000.

7. D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Collaborative multi-robot localization. In *Proceedings of the 23rd German Conference on Artificial Intelligence*. Springer-Verlag, 1999.

8. D. Friedman. The double auction institution: A survey. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 1, pages 3–25. Perseus Publishing, Cambridge, MA, 1993.

9. B. Gerkey and M. Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 2000.

10. D. Guzzoni, A. Cheyer, L. Juli, and K. Konolige. Many robots make short work. *AI Magazine*, 18(1):55–64, 1997.

11. G. A. Kaminka, D. V. Pynadath, and M. Tambe. Monitoring deployed agent teams. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 308–315. ACM Press, 2001.

12. T. L. Lenox, T. R. Payne, S. Hahn, M. Lewis, and K. Sycara. Agent-based aiding for individual and team planning tasks. In *Proceedings of IEA 2000/HFES 2000 Congress*, 2000.

13. Repast. `http://repast.sourceforge.net`.

14. M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

15. M. Veloso and S. Lenser. CMPAck-02:CMU's Legged Robot Soccer Team. In *Proceedings of the RoboCup Symposium*, 2002.