ELSEVIER

Journal of Cognitive Systems Research 2 (2001) 67–79

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Learning about other agents in a dynamic multiagent system ☆

Action editor: Ron Sun

Junling Hu[a,*], Michael P. Weliman[b]

[a]*Simon School of Business, University of Rochester, Rochester, NY 14627, USA*
[b]*Computer Science & Engineering, University of Michigan, Ann Arbor, MI 48109-2110, USA*

Received 17 October 2000; accepted 30 October 2000

Abstract

We analyze the problem of learning about other agents in a class of dynamic multiagent systems, where performance of the primary agent depends on behavior of the others. We consider an online version of the problem, where agents must learn models of the others in the course of continual interactions. Various levels of recursive models are implemented in a simulated double auction market. Our experiments show learning agents on average outperform non-learning agents who do not use information about others. Among learning agents, those with minimum recursion assumption generally perform better than the agents with more complicated, though often wrong assumptions.

Keywords: Multiagent learning; Multiagent systems; Computational market

1. Introduction

Multiagent systems are in general dynamic systems. The dynamics arise not only from changes in environment state, but also from evolution of agent behaviors. Agents that adapt their behavior, or learn, based on their experience with the environment and other agents, are thus themselves a source of dynamics, providing further impetus for others to adapt and learn.

The effectiveness of learning depends not only on the learning method, but also on how much information is available to the agent. When an agent cannot observe other agents' actions, it has to rely on indirect evidence of the effect of those actions. Such partially observable or incomplete information makes learning more difficult, sometimes imposing strict limitations. In a previous study Wellman & Hu (1998), we investigated a scenario where agents observed only overall effects of joint actions, and attempted to learn a model at this aggregate level. Our study showed that an agent's learning in such a dynamic system can be trapped in a suboptimal equilibrium by a self-fulfilling bias. Our results were confirmed by the study of Sandholm & Ygge (1997), which demonstrated that an imperfect learning agent

☆Revised and extended version of a paper presented at the Second International Conference on Autonomous Agents, Minneapolis, May 1998.

*Corresponding author.
E-mail address: huju@simon.rochester.edu (J. Hu).

1389-0417/01/$ – see front matter
PII: S1389-0417(01)00016-X

in a market with many consumers and producers could perform worse than a non-learning agent.

The present paper is a further attempt to explore the relationship between information available and the effectiveness of learning. Given more information, we might sometimes avoid the the pitfall of self-fulfilling bias, and thus produce better overall performance. In our auction market, each agent is able to observe others' local states and history of actions. The learning agent uses that information to predict the individual actions of others. Our agent can make its prediction in two ways. In one, the agent learns a time-series model of another agent's actions, ignoring the underlying decision process of that agent. In another, the agent learns an underlying decision model that determines another agent's action. Based on rational decision making, the learning agent assumes that every agent is trying to maximize its payoff, therefore there exists a functional relation between that agent's actions and its local states. There are many possible functional forms. We distinguish them by the levels of recursion that our agent assumes about others. We would like to investigate which method of modeling others is more advantageous: a model-free approach based on time series analysis, or a model-based approach derived from assumptions of other agents' recursive level? Furthermore, what is the effect of level of recursion on an agent's performance?

Explicit recursive modeling of other agents was proposed by Gmytrasiewicz et al. (1991). Vidal & Durfee (1998a) formalized the nesting of models in an abstract knowledge framework, and implemented recursive modeling agents in a market for information goods. In the present work, we adopt their concept of k-level to describe the depth at which an agent models others.

We define our problem in the framework of *stochastic games* (Filar & Vrieze, 1997; Thusijsman, 1992). This framework combines game-theoretic solution concepts (Owen, 1995) with individual agents' Markov decision processes. Unlike models based on static or repeated games, stochastic games can account for state transitions and how they are influenced by agent actions. Much of the recent study of multiagent reinforcement learning (Hu & Wellman, 1998; Laner & Riedmiller, 2000; Littman, 1994) has been conducted within the stochastic game framework.

We adopt regression methods to derive functional relations between other agents' actions and their internal states. Regression methods are particularly useful for online learning, because we can often generate meaningful behaviors with a minimal number of observations. Regression methods have been commonly applied to domains with continuous actions or states. In Nadella and Sen's work (Nadella & Sen, 1997), for example, a soccer agent uses linear regression to estimate the relation between its shooting error and shooting distance, and employs predicted error as a guide for planning a shot.

In the following sections, we present the framework of stochastic games and characterize recursive-modeling agents within this framework. We introduce an instance of this framework — a dynamic market trading game where agents interact through synchronous double auctions. We propose a series of regression-based learners for this domain, varying in strategy and depth of reasoning about others. In all cases, we assume that the learning must be online because the other agents do not make themselves available for free trials. In our experiments, learning agents on average outperform non-learning agents who do not use information about others. Among learning agents, those with minimal assumptions about other agents tend to perform better than agents that attribute excess sophistication to their counterparts. However, when such attribution is warranted, agents can do better by learning the more sophisticated models.

2. Multiagent systems

A *multiagent system* consists of multiple agents who are autonomous and make their decisions independently. By this definition, we rule out those systems where a central planner or designer controls the decision processes of local agents. In other words, we define multiagent systems as fully decentralized systems. Among decentralized systems, only the systems allowing for meaningful agent interaction can be truly classified as multiagent systems. If the agents' actions do not effect each others' outcomes, then we may as well consider the agents' situations independently.

There are two types of multiagent systems. In the first, agents can form binding agreements to achieve

mutual benefits. In the second, such binding agreements cannot be formed. The former systems are called *cooperative*, and the latter are called *noncooperative*. Note that this terminology is at variance with common usage in AI (which equates cooperation with acting in common interest), but is standard in the game theory literature (Harsanyi & Selten, 1988). In a noncooperative system, agents' rewards may be positively, negatively, or arbitrarily related to each other. Zero- or constant-sum games are examples where agents' rewards are always negatively related. In coordination games (an example is displayed in Table 1), agents' rewards are always positively correlated. Even with positively correlated rewards, absent an ability to arrange joint behavior through binding agreement, agents need to form their own strategies for playing the game.

We are concerned with the class of noncooperative systems in which agents encounter different stage games at different time periods, and each stage game is associated with a particular state. A stage game here is defined as a one-shot simultaneous-move game. The state changes based on the current state and the agents' joint action. In the most general case, the state transition is assumed to be stochastic, and such games are called *stochastic games* (Filar & Vrieze, 1997; Thusijsman, 1992). When the state transition is deterministic, the games become special cases of stochastic games, and we call such games *deterministic-transition dynamic games*. The precise definition is given as follows:

Definition 1. An n-player *deterministic-transition dynamic game* is a tuple $\langle S, A, r, T\rangle$. $S = S^1 \times \cdots \times S^n$ is the state space, with S^i the part of the state space relevant to agent i. $A = A^1 \times \cdots A^n$ is the joint action space, with A^i the action space for agent i. $r = (r^1, \ldots, r^n)$ represents the agent reward functions, with r^i: $S^i \times A \to \Re$ the payoff function for player i. $T = (T^1, \ldots, T^n)$ is the state transition function, with T^i: $S^i \times A \to S^i$ as the transition function for player i.

Table 1. A coordination game

	Agent 2	
Agent 1	*Left*	*Right*
Up	30, 30	0, 0
Down	0, 0	30, 30

At each time point t, after observing state s_t, agents choose their actions independently and simultaneously. They then receive their rewards r_t^i, for $i = 1, \ldots, n$. The rewards are functions of s_t and the agents' joint actions at time t, $(a_t^1, \ldots, a_t^n)$. The state transits to a new state s_{t+1} based on s_t and the agents' joint actions, with $s_{t+1} = (s_{t+1}^1, \ldots, s_{t+1}^n)$ and $s_{t+1}^i = T^i(s_t^i, a_t^1, \ldots, a_t^n)$ for $i = 1, \ldots, n$.

An agent's *strategy* is a plan for playing the game. A strategy $\pi = (\pi_0, \ldots, \pi_t, \ldots)$ is defined over the whole course of the game, where π_t is called the *decision rule* at time t.

2.1. Special assumptions

We assume that agents can observe the state and actions of other agents, but not others' reward functions. All agents know that the game will end in a finite period, but they are uncertain when that period will be.

Given a measure of utility as a function of local state, we can define agent i's reward r_t^i as the improvement in utility at time t:

$$r_t^i = U^i(s_{t+1}^i) - U^i(s_t^i).$$

Since a state is determined by joint actions, agent i's reward at a given time period,

$$r_t^i = U^i(T^i(s_t^i, a_t^i, a_t^{-i})) - U^i(s_t^i),$$

depends on the actions of other agents, where a^{-i} represents the joint action of all the agents other than agent i. The object of the learning problem is to predict these actions — explicitly or implicitly — so that the agent can effectively select its own.

To simplify the learning problem, we assume that the agent makes its decisions *myopically*, that is, considering only the current time period. To maximize the current reward at t, the agent solves

$$\arg\max_{a_t^i \in \mathcal{A}^i} U^i(T^i(S_t^i, a_t^i, a_t^{-i})).$$

We can drop the $U^i(s_t^i)$ term since it is a known constant at time t.

Note that agent i's true objective is to maximize the expected sum of rewards over time, given the strategies of all the agents,

$$\max_{\pi^i} \sum_{t=0}^{T} E(r_t^i | \pi^1, \pi^2, \ldots, \pi^n, s_0 = s),$$

where π^i is the strategy of agent *i*.

Since agents choose their actions simultaneously, other agents' actions a_t^{-i} are unknown to agent *i* at time *t*. A generally applicable (albeit not optimal) approach is to form an estimate, $\hat{a}_t^{-i}$, and solve the problem as if the estimate were correct. We consider several approaches in the sections below, all of which further assume that the agent models others as having stationary strategies.

2.2. Modeling other agents

2.2.1. Levels of recursive modeling

Perhaps the most straightforward way to estimate a_t^{-1} is based on a time series analysis of the prior observations, $\{a_\tau^{-i}\}$, $\tau < t$. Agents taking this approach do not explicitly attempt to reason about what determines the a^{-i}. We call such agents 0-*level* agents, meaning that they do not model the underlying behavior of other agents. One example of a 0-level estimate is to take $\hat{a}_t^j$ as a linear function of prior observations,

$$\hat{a}_t^j = \sum_{k=1}^{t} w_k a_{t-k}^j. \tag{1}$$

A somewhat deeper way to learn a_t^{-1} is to learn, for each agent *j*, a model of the relationship between *j*'s action, a_t^j, and its state, s_t^j. This amounts to learning a policy function for agent *j*. The policy function can be written as a function of *j*'s state and *j*'s estimate,

$$\hat{a}_t^j = f^j(s_t^j, \hat{a}_t^{-j}). \tag{2}$$

Such an agent who attempts to model the policy functions (rather than just actions) of the others is called a 1-*level* agent.

Note that our 1-level agent assumes that the policy function of agent *j*, f^j, is stationary over time. Although it is never true of learning agents, it might be reasonable for an agent to adopt it when learning about others, as it greatly simplifies the modeling problem.

A natural extension is to consider agents that model others as 1-level. What this means is that the agent attempts to learn the policy function of other agents, taking into account that in this policy the subject agent is itself modeling the policies of others. Expressed mathematically, the 2-*level* agent models the policy of agent *j* as

$$a_t^j = f^j(s_t^j, \{f^k(s_t^k, \hat{a}_t^{-k}) | k \neq j\}). \tag{3}$$

where $f^k(s_t^k, \hat{a}_t^{-k})$ is its model of agent *j*'s model of agent *k*'s policy. Since all agents have the same observations (except about their own policies), *i*'s model of *j*'s model of *k* (for $k \neq i$) is exactly what *i*'s model of *k* would be if *i* were acting as a 1-level agent.

And of course one can define 3-level, . . . , *n*-level agents by repeated deepening of this model.

2.2.2. 0-level non-learning agents

Although as one might expect, 0-level is as low as we can go, there are varying degrees of sophistication within levels — including the zeroth — that are worth distinguishing in a taxonomy of learning approaches. In particular, an agent might learn a policy without explicit consideration of a_t^{-1} at all, in which case its policy, $f^i(s_t^i)$ is a function of its local state alone.

An example of such an agent, introduced in our trading scenario below, is the *competitive* agent, named in accord with the sense of this term in economics (Varian, 1992). To be competitive in a market context means to assume that one's own effect on the environment is negligible, in which case there is no advantage to speculating about the actions of others. This type of agent tends to behave 'reactively', although this characterization is not as precisely defined as is 'competitive' in the market context.

3. A double auction market

Auctions come in a wide range of types, distinguished by the way bidders submit their bids and how the allocations and prices are determined (McAfee & McMillan, 1987). In a *double-sided* (or *double*) auction, both buyers and sellers submit bids. A single agent may even submit both, offering to buy or sell the same type of good with different prices. Based on the timing of the bidding protocol,

double auctions can be classified as *synchronous* (or *synchronized*) double auctions and asynchronous double auctions. In a synchronous double auction, all agents submit their bids in lockstep. Bids are 'batched' during the trading period, and then cleared at the end of the period. In real life, the clearinghouse is a prime example of such auctions. Asynchronous double auctions are also called *continuous* double auctions where bids can come in at any moment. A prototypical example of such auctions is a stock market (Friedman, 1993).

The book edited by Friedman & Rust (1993) collects several studies of double auctions, including both simulations and game-theoretic analyses. Game-theoretic studies (Friedman, 1993; Satterthwaite & Williams, 1993) on double auctions generally adopt the framework of static (one-shot) games with incomplete information, for which the equilibrium solution is Bayesian Nash equilibrium. However, most double auctions are dynamic systems where agent interaction takes more than one round. We need a framework that captures the basic dynamics of the system.

Among the studies of agent design for auctions, Gode & Sunder (1993) designed *zero-intelligence* agents who submit random bids within the range that their utilities never decrease. Such agents can be viewed as one type of 0-level agents defined in this paper. To improve upon zero-intelligence agents, Cliff (1998) designed *zero-intelligence-plus* agents for continuous double auctions. Such an agent submit bids that always brings a certain profit margin to the agent. The profit margin is adjusted based on how well the agent does in the auction. The adjusting process can be seen as a learning process. The zero-intelligence-plus agents never model other agents. Such agents are suitable for the environment where information about other agents is not available or there are so many agents that the modeling of other agents is computationally infeasible.

The Santa Fe tournament (Rust et al., 1993) suggests another type of bidding agent with good performance. Such agents wait in the background and let the others do the negotiation, but when bid and ask get sufficiently close, they jump in by bidding slightly higher than the previous ask price and get the deal. The success of these agents are conditional on two features of the particular auction. First, the total rounds of the auction are known. This allows the agent to calculate how far it is to the end of the auction. Second, the agent has a limited number of items to sell or buy, thus it can afford to miss the trading opportunities at the first few rounds. Neither condition is satisfied in our auction. When the agent has uncertainty about when the auction will end, its bidding strategy will be very different. However, the Santa Fe tournament suggests that an agent can benefit from forming a strategy against what other agents do.

3.1. Specifications of a synchronous auction market

Our auction process can be described as the following: At the start of the auction, each agent is endowed with designated amounts of m goods, each — except the last — associated with an auction. Each time period constitutes a bidding round for one auction, rotated in turn. Agents bid in an auction by posting buy and sell prices for one unit of the good for that auction market. All prices are expressed in units of good m.

After all agents submit their bids, the auction matches the highest buyer to the lowest seller, if the buy price is greater than the sell price. The trading price for the match is equal to $\frac{1}{2}P^{\text{buy}} + \frac{1}{2}P^{\text{sell}}$, where P^{buy} is the price offered by the buyer, P^{sell} is the price offered by the seller. The auction then matches the second highest buyer with the second lowest seller, and so on, until the highest remaining buyer is offering a lower price than the lowest remaining seller. At this point, the market proceeds in turn to the next auction. In the new round, agents post their buy and sell prices for the next good, and they are matched according to the prices they post. The market continues until no agents can be matched in any auction. Fig. 1 shows the matching process in one period of an auction.

3.2. Synchronous double auctions as stochastic games

Our double auction can be modeled as a deterministic-transition dynamic game, where the state is a vector consisting of all agents' individual endowments. The action is the vector of bids sub-

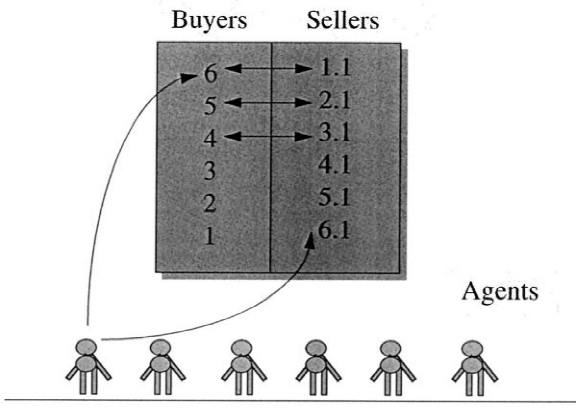

Fig. 1. One round of bidding in the synchronous double auction.

mitted by all agents. At time t, agent i's local state s_t^i, is described by its endowment vector, $e_t^i = (e_1^i(t), \ldots, e_m^i(t))$. Agent i's action at time t, $a_t^i = (P_t^{i,\text{buy}}, P_t^{i,\text{sell}})$, is its buying and selling price offered for one unit of good in the current auction.

The reward for agent i at time t is given by

$$\begin{aligned} r_t^i &= U_{t+1}^i - U_t^i \\ &= U(e^i(t+1)) - U(e^i(t)) \end{aligned}$$

where

$$\begin{aligned} &e_g^i(t+1) = e_g^i(t) + z_t \\ &e_m^i(t+1) = e_m^i(t) - P_t z_t, \text{ and} \\ &e_l^i(t+1) = e_l^i(t),\ l \neq g \text{ and } l \neq m. \end{aligned}$$

In these state update equations, P_t is the trading price, and $z_t \in \{-1, 0, 1\}$ is the quantity the agent trades at time t. When $z_t = 0$, the agent does not trade, and $r_t = 0$.

3.3. Economic analysis of the double auction

We assume that each agent i has a CES (Constant Elasticity of Substitution) utility function,

$$U(x) = \left(\sum_{g=1}^{m} \alpha_g x_g^{\rho} \right)^{1/\rho} \tag{4}$$

where $x = (x_1, \ldots, x_m)$ is a vector of goods, the α_g are preference weights, and ρ is the substitution parameter. We choose the CES functional form for its convenience and generality — including quadratic, logarithmic, linear, and many other forms as special cases.

In constructing agent strategies, we dictate that they always choose actions leading to nonnegative payoffs. We can characterize this in terms of the agents' *reservation prices* (Varian, 1992). The reservation price is defined as the maximum (minimum) price an agent is willing to pay (accept) for the good it wants to buy (sell). More specifically, it is a price at which an agent's utility keeps constant when buying or selling one unit of good.

$$U(e_g - 1, e_m + \bar{P}_s, e_{-g,-m}) = U(e_g, e_m, e_{-g,-m}), \tag{5}$$

$$U(e_g + 1, e_m - \bar{P}_b, e_{-g,-m}) = U(e_g, e_m, e_{-g,-m}). \tag{6}$$

where $\bar{P}_b$ and $\bar{P}_s$ are reservation buy price and sell price.

We prove below that for an agent with quasi-concave (such as CES) utility function, its reservation buy price is always lower than the reservation sell price.

Theorem 1. *For any agent with strictly quasi-concave utility function, the agent's reservation buy price $\bar{P}_b$ is always less or equal to its reservation sell price $\bar{P}_s$.*

Proof. Let an agent's utility function be $U(x_j, x_m, x_{-j,-m})$ where j is the good for current auction, m is the numerair good to be traded with good j, $x_{-j,-m}$ is agent's holding of goods other than j and m.

Let $U(x_j, x_m, x_{-j,-m}) = \bar{U}$, by definition of reservation price, we have $U(x_j - 1, x_m + \bar{P}_s, x_{-j,-m}) = \bar{U}$ and $U(x_j + 1, x_m - \bar{P}_b, x_{-j,-m}) = \bar{U}$. Since $x_{-j,-m}$ and $\bar{U}$ are constants, we can re-write the previous equations to express x_m as a function of x_j:

$$x_m \bar{P}_b = f(x_j + 1) \tag{7}$$

$$x_m + \bar{P}_s = f(x_j - 1) \tag{8}$$

Adding (7) and (8) we have $\bar{P}_s - \bar{P}_b = f(x_j + 1) + f(x_j - 1) - 2x_m$.

Since the utility function $U(x_j, x_m, x_{-j,-m})$ is quasi-concave, $U(x_j, x_m, x_{-j,-m}) = \bar{U}$ yields a convex function $x_m = f(x_j)$. By definition of convex function, $f(x_j + 1) + f(x_j - 1) \geq 2f(x_j)$. Therefore $\bar{P}_s \geq \bar{P}_b$. □

We can further show that if an agent's buy price P_b and sell price P_s satisfy $P_s \geq \underline{P}_s$ or $P_b \leq \bar{P}_b$, the agent's utility never degrade. This result comes from the fact that the utility function is increasing in each of its components. Therefore, if an agent's sell bid P_b and buy bid P_s should always satisfy $P_s \geq \underline{P}_s$ and $P_b \leq \bar{P}_b$ if the agent intends to maintain its utility level at any time. It follows then from Theorem 1 that we always have $P_b \leq P_s$ (Fig. 2).

Below we prove that an agent will never be both a buyer or seller in the same market.

Theorem 2. *Given that an agent's sell bid exceeds its buy bid, the agent can never be matched both as buyer and a seller on the same market.*

Proof. Let the agent be i. Suppose agent i can be matched to agent j as a buyer, and be matched to agent k as a seller. Then we have $P_b^i \geq P_s^j$ and $P_b^k \geq P_s^i$. Since agent i's bids satisfy $P_b^i \leq P_s^i$, we have

$$P_s^j \leq P_b^i < P_s^i \leq P_b^k. \tag{9}$$

According to the matching rule in our double auction, the highest buyer is matched to the lowest seller, the 2nd highest buyer is matched to the 2nd lowest seller and so on. Thus for any 2 pairs of traders, if the buy price in the first pair is higher than the buy price in the second pair, the sell price in the first pair is always lower than the one in the second pair. Now we have two pairs of traders {buyer k, seller i}, {buyer i, seller j}. If $P_b^k \geq P_b^i$, then we will have $P_s^i \leq P_s^j$. This contradicts the inequality (9). □

3.4. Four types of agents

3.4.1. The competitive agent

Our first type of agent, the *competitive*, does not attempt to model the other agents. Thus, it is a 0-level non-learning agent, as described in Section 2.2.2. The competitive agent always chooses its reservation prices $\bar{P}_b$ and $\underline{P}_s$. Based on definition of reservation prices, the competitive agent will never be worse off if trading happens at these prices. Furthermore, the auction rule requires that the price offered from the buyer should always be greater than or equal to the price offered by the seller when they are matched. If the competitive agent is a buyer, it will be better off from the trade since the trading price is anywhere between its offer and the seller's offer and will always be lower than the buy price. Similarly the competitive agent will be better when it is a seller.

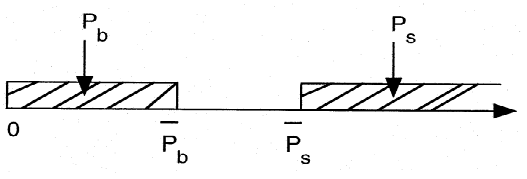

Fig. 2. An agent's reservation prices and its actual bids.

Since the trading price is the average of the buyer's and seller's bids, it is always at least $\underline{P}_s$ (when the competitive agent sells), or at most $\bar{P}_b$ (when the competitive agent buys).

The other three types of agents choose their bidding prices (P_b, P_s) based on their predictions of other agents. Although they differ in how they estimate other agents' actions, they are identical in how to choose bidding prices as best responses to their estimates.

3.4.2. Three types of learning agents

A 0-level learning agent does not model the underlying policy functions of other agents. It predicts the actions of other agents by looking at the history data of those actions. The 0-level learning agent uses the following formula to predict bidding price made by another agent j:

$$\hat{P}_t^j = P_{t-1}^j + \alpha(P_{t-1}^j - P_{t-2}^j). \tag{10}$$

This model suggests that if agent j's price increases from time $t-2$ to $t-1$, our learning agent will assume the same trend will continue from period $t-1$ to t.

A 1-level agent, say agent i, tries to estimate a fixed policy functions of other agents. It assumes that other agents are 0-level non-learning (competitive) agents, whose actions are based on their individual optimization problems. A competitive agent always bids its reservation prices, which are calculated according to Eqs. (5) and (6), and are functions of the agent's own states, its endowments. The exact form of such policy functions is unknown to our learning agent because the competitive agent's utility function is unknown. We assume a linear functional

form. Let another agent be agent j, thus our learning agent's estimate of agent j's bidding price, P_t^j, is

$$\hat{P}_t^j = \alpha + \beta e_t^j.$$

where e_t^j is a vector of dimension m (m is the number of different goods the agent holds).

A 2-level agent, like a 1-level agent, models the policy functions of other agents. But a 2-level agent assumes others are 1-level learning agents. Based on Eq. (3), we adopt a simplified 2-level model: $P_t^j = f(e_t^j, e_t^{-j})$. We found that a linear regression method does not work well. One reason is the correlation between the independent variables e_t^j, e_t^{-j}. In addition, the high dimensionality of input data requires large amount of data in order to get unbiased estimation. Since we assume agents take only a small sample of history data (fixed window length), we need to adopt a nonparametric regression method. We use the K-Nearest Neighbor method in this paper. For current joint state $(e_t^1, \ldots, e_t^m)$, take its K nearest neighbors $\{(e_{t1}^1, \ldots, e_{t_1}^m), \ldots, (e_{t_k}^1, \ldots, e_{t_k}^m)\}$ as the inputs and the corresponding actions of agent j, $\{P_{t_1}^j, \ldots, P_{t_k}^j\}$ as the outputs. The estimate of P_t^j is:

$$\hat{P}_t^j = \sum_{l=1}^{K} W_l P_{t_l}^j$$

where W_l is the weight of data point e_l and is defined as

$$W_l = \frac{\frac{1}{d_l}}{\sum_{l'=1}^{k} \frac{1}{d_{t_{l'}}}}$$

where d_l is the distance between the data point $(e_l^1, \ldots, e_l^m)$ and the query point $(e_t^1, \ldots, e_t^m)$, $d_l = \sqrt{(e_l^1 - e_t^1)^2 + \cdots + (e_l^m - e_t^m)^2}$.

3.4.3. Best-response bidding

Let agent i be the learning agent, with reservation prices $\bar{P}_b^i$ and $\bar{P}_s^i$. Let $\{\hat{P}_b^1, \ldots, \hat{P}_b^n\}$ and $\{\hat{P}_s^1, \ldots, \hat{P}_s^n\}$ be agent i's projected buying and selling prices of other agents. Suppose agent i can be matched as a buyer according to its reservation price, and is matched to seller j. The matching rule is shown in Fig. 1. The highest buyer is matched to the lowest seller, the second highest buyer is matched to the second lowest seller, and so on. The matching stops when the buy price is lower than the sell price. According to this rule, if agent i is the xth highest buyer, then agent j is the xth lowest seller. By the matching criterion, we must have $\bar{P}_b^i \geq \hat{P}_s^j$.

Agent i's goal is to choose a bidding price such that it can still trade with agent j, but will squeeze all the surplus from the trade. In order to trade with seller j, agent i's price has to be at least as high as j's price. That means the lowest price agent i can get away with is $\hat{P}_s^j$. Furthermore, agent i has to keep its bidding price higher than all the buyers currently ranked lower than itself. Let agent k be the buyer with the next highest bid other than agent i. Agent i has to bid a little higher than agent k in order to keep its current ranking. Thus agent i has to bid at least $\hat{P}_b^k + \epsilon$, where ϵ is a small positive constant representing the minimal bid increment (Fig. 3).

Combining the above two lower bounds, which have to be satisfied at the same time, agent i's optimal buy price is

$$P_b^i = \max\{\hat{P}_s^j, \hat{P}_b^k + \epsilon\}, \tag{11}$$

Note that the above equation automatically satisfies the condition $P_b^i \leq \bar{P}_b$ because $\hat{P}_s^j \leq \bar{P}_b^i$ and $\hat{P}_b^k \leq \bar{P}_b^i$.

By similar analysis, suppose agent i can be matched as a seller to buyer j, and the next lowest seller is agent k, agent i should choose its sell-price such that

$$P_s^i = \min\{\hat{P}_b^j, \hat{P}_s^k - \epsilon\}.$$

If agent i predicts that it cannot be matched either as a buyer or a seller according to its reservation prices, it simply bids its reservation prices.

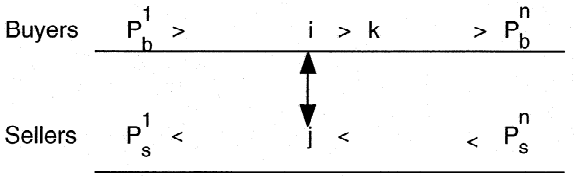

Fig. 3. Choose best-response bid.

3.5. Convergence

By convergence, we mean that the system reaches an equilibrium in which the state stays the same as before. In our double auction case, the state is a vector of all agents' endowments. An equilibrium is reached when, for all goods, all the buy prices are lower than all the sell prices. That means no buyer can be matched to a seller, no transaction happens, and therefore the endowment of every agent is the same as last period. An auction terminates once it reaches an equilibrium. Note that agents need not to have correct belief in such an equilibrium, which can happen whenever all buy bids are lower than the sell bids.

The trading process in our auction can be cast as a Edgeworth process in the broad sense that agents' utilities continually increase throughout the process (Varian, 1992). However, the specific trading rules in our auction differ in a significant way from the ones in standard Edgeworth process (Uzawa, 1962). In a standard Edgeworth process, a central price is announced at each period by the auctioneer, and each agent decides whether to trade based on that price. Thus agents have no role in determining the prices in the market. This leaves little room for strategic behavior. Uzawa (1962) proved that such a process converges to a Pareto optimum distribution where no price exists such that at least two agents can increase their utilities by trading.

In our auction, agents submit bidding prices which will determine the trading prices. Therefore agents have direct influence on the market, and they have incentives to act strategically. Such behavior may cause the auction end prematurely before reaching a Pareto optimum. To see how this can happen, consider an auction that is one step from the Pareto optimum, with the only trading opportunity between buyer A and seller B left. Now suppose A's reservation buy price is 5, B's reservation sell price is 4. If A behaves strategically, A may offer a price far lower than 5, say at 3. Then the market will terminate and agent A and B will never get a chance to trade.

Below we prove that the synchronous double auction market converges. The proof relies on the simple fact that all agents' utilities keeps increasing in the auction.

Theorem 3. *The synchronous double auction market converges.*

Proof. Let $U_t = \sum_{i=1}^{n} U^i(e_t^i)$ be the sum of utilities of all agents. Let $\bar{U} = \sum_{i=1}^{n} U^i(E)$, where E is the total endowment in the system. In a pure exchange system such as our auction market, the total endowment E remains constant. For each agent i, $e_t^i \le E$. Thus $U_t \le \bar{U}$ for any t, that is, U_t is bounded above by $\bar{U}$. Since we stipulate that agents always bid to get non-negative payoff, their utility will always be greater than or equal to that in last period. Thus the sum of all agents' utility, U_t, is nondecreasing. That is, $(\partial U_t/\partial t) \ge 0$. Therefore the bounded increasing sequence $\{U_t\}$ converges to a point U^*, such that $U^* = \lim_{t\to\infty} U_t$, and $U^* \le \bar{U}$. □

4. Experiments

We implement a synchronous auction with six agents and two types of goods. Each agent has the same CES utility function as defined in (4). The parameters for the utility function as set as $\alpha_j = 1$ for all j, and $\rho = 1/2$. Thus the utility function becomes $U(x) = (\Sigma_j (x_j)^{1/2})^2$. This utility function leads to the following reservation buy and sell prices:

$$\bar{P}_b = 2(\sqrt{e_1(e_1+1)} + \sqrt{e_2(e_1+1)} - \sqrt{e_1 e_2} - e_1) - 1,$$

$$\bar{P}_s = -2(\sqrt{e_1(e_1-1)} + \sqrt{e_2(e_1-1)} - \sqrt{e_1 e_2} - e_1) - 1$$

Agents differ in their initial endowments, which are randomly assigned. Prices are in units of good 2; thus, all auction activities are for good 1.

We let our primary agent, Agent 1, to take on four different agent types. We then compare the performance of these four types in a given environment where all other five agents are of the same type. They are all either: competitive agents, 0-level learning agents, 1-level agents, or 2-level agents. Just to reiterate our definitions, a competitive (0-level non-learning) agent does not use any information about other agents. It chooses its action based on its individual optimization problem. A 0-level learning agent makes no assumption about other agents but

uses the data of other agents' actions. A 1-level agent assumes other agents are 0-level non-learning (competitive) agents. A 2-level agent assumes other agents are 1-level agents.

In each of the four environments, Agent 1 and the other five agents' initial endowments are randomly assigned and remain the same when Agent 1 changes its type. The performance under the four different strategies is then compared and ranked. These comparisons are repeated for 50 times for different sets of initial endowments, and we record the number of times that each type is ranked first, second, third and fourth out of 50 total runs.

Fig. 4 shows the experimental results for one set of initial endowments in the environment where all other five agents are competitive agents. As we can see, Agent 1's utility increases over time under any learning strategy. The utility levels of these strategies are the same for the first three rounds, where Agent 1 collects data for its learning methods. After 3 rounds, agent starts bidding based on learned models, and the utility level of different learning methods diverges. A 0-level learning agent has the fastest growth in utility level and remain so in the end. The competitive agent has the lowest utility level in the end, even though its utility increases faster than the 2-level agent at the beginning. The auction converges at round 290.

Fig. 5 presents the average results for the environment where all other five agents are competitive agents. For this environment, we predict that being a

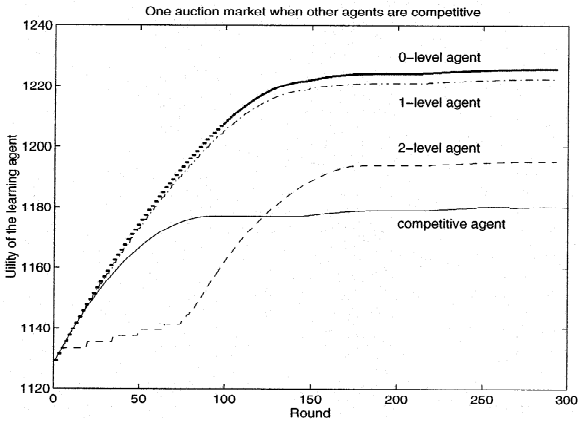

Fig. 4. Agent l's performance over time.

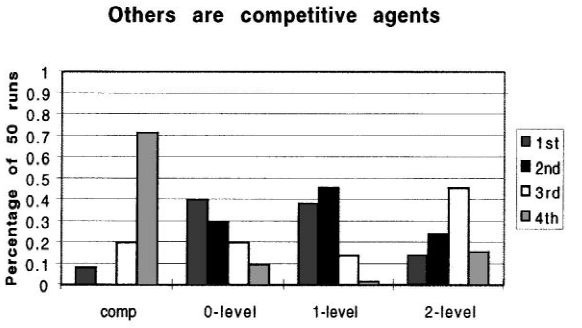

Fig. 5. Performance of Agent 1 under different types in the environment where all others are competitive agents.

1-level agent will give Agent 1 the best performance because the it has correct assumption about other agents of their being competitive. As we can see, out of 88% of all experiments, the 1-level agent ranks the best and the second best. The 0-level agent has similar performance only 70% of time. The 2-level agents and competitive agents perform much worse than the other types.

Fig. 6 presents results for an environment where all other agents are 0-level agents. For this environment, we do not expect being a 1-level or 2-level agent will lead to better performance than being a 0-level agent because these types of agents make incorrect assumptions about others. From Fig. 2, we see that the 0-level agent, which achieves first- or second-best performance 82% of the time, is clearly the best among the types. The competitive agent still performs the worst.

When we change the environment to one in which all other agents are 1-level, the experimental results (Fig. 7) show that the 2-level agent performs worse

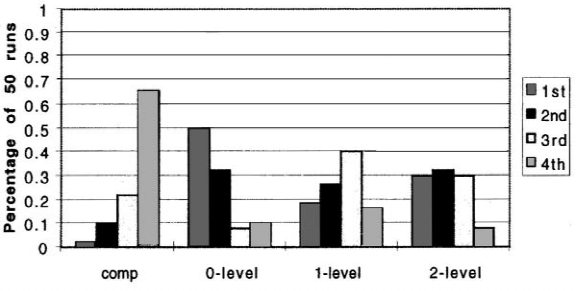

Fig. 6. Performance of Agent 1 under different types in the environment where all others are 0-level learning agents.

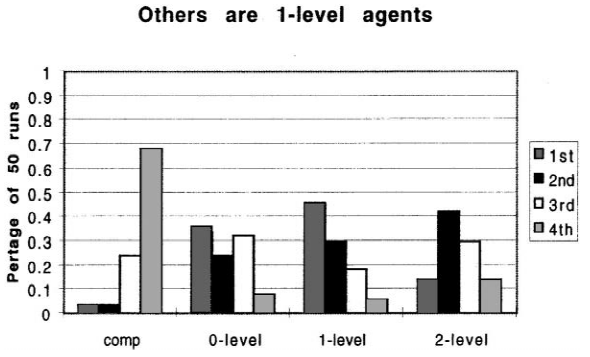

Fig. 7. Performance of Agent 1 under different types in the environment where all others are 1-level agents.

than the 1-level agent, but almost as well as the 0-level agent. This is a little unexpected. We expected that the 2-level agent should do well since it correctly assumes other agents are 1-level agents.Our explanation is that the 2-level agent we implemented is a simplified version of a full 2-level agent, whose model should include all possible variables. Computationally, this is not feasible, and it remains an open questions how to form a computationally efficient and correct model of 2-level agent.

Fig. 8 shows the results of an environment where all other agents are 2-level agents. In this environment, we did not expect either 1-level and 2-level agent to perform well, because these agents make incorrect assumptions about the other agents. The experimental results confirm our expectation. Both 1-level and 2-level agents perform worse than the 0-level agent, who achieves first- or second-best performance 73% of the time.

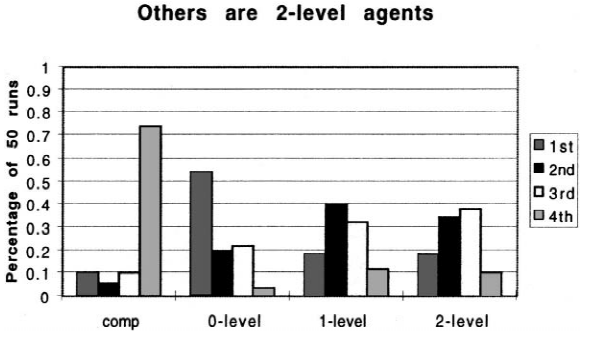

Fig. 8. Performance of Agent 1 under others are 2-level agents different types in the environment where all.

Our experiments suggest that, in most cases, the 0-level learning agent performs better than any the other types, especially when these others make incorrect assumptions about the environment. Finally, our experiments show that the competitive agent performs worst in all four cases. This suggests that if an agent does not use any information about other agents in the system, the agent will perform worse than those who do use the information.

One thing to notice from the graphs is that no single agent is uniformly superior. This is due to the prediction error made by the learning agents. Such error leads to wrong bidding prices and causes the premature convergence of the system. In this case, the learning agent will get worse payoff than the competitive agent since the trading opportunities in the system have not been fully explored.

5. A comment on recursive modeling

Even though recursive modeling is a natural extension to simple modeling of others, it has its limitations. In recursive modeling, an n-level agent always assumes others to be $n-1$ level agents, who in turn assume everyone else to be $n-2$ level agents. Since the n-level agent itself is in the system, the $(n-1)$-level agent always has a wrong belief. By believing others to be $(n-1)$-level agents, our n-level agent always assumes that others have incorrect models.

Another limitation of recursive modeling is that there are many possible levels of recursion, ranging from level 0 to infinity. It is impossible for our agent to test each of them and find a correct level in limited interactions with other agents. Therefore this model presents intrinsic computational difficulty.

A further limitation of recursive modeling, which also applies to many other modeling methods, is that it assumes others' belief and decision models remain constant. If other agents change their beliefs or decision models during the interaction, the models our agent tries to learn become obsolete. This is called the 'moving target problem' by Vidal & Durfee (1998b). A complete solution or remedy to this problem has not been found.

A better way to capture agents' mutual conjectures would be an equilibrium concept, which allows

every agent's conjecture to be correct. Nash equilibrium is such a concept. In a Nash equilibrium, every agent has the correct belief, and acts optimally given its belief. We have investigated using a multiagent reinforcement learning method to learn stationary Nash equilibrium strategies (Hu & Wellman, 1998; Hu & Wellman, 2000) in a domain of discrete state space and action space. It remains to be seen what learning method would be more appropriate for a system with continuous state space and action space.

6. Conclusions

In this paper we analyze the issue of learning about other agents in a dynamic multiagent system. We model the multiagent interaction in the framework of stochastic games. We investigate the effect of learning when information is *complete*, meaning other agents' local states and history of actions are observable.

We apply various regression methods to learn about other agents' actions in a continuous state space and action space. The regression model is based on an agent's recursive belief about other agents. Different levels of recursion define different types of agents. Our 0-level agent makes no assumption on other agents decision model, our 1-level agent model others as 0-level agents, and 2-level agent models others as 1-level agents. In a simulated double auction market, we test the performance of the primary agent when it takes on these different types, relative to a fixed type of all other agents. Our experiments show that the primary agent's performance is worst when it is a 0-level non-learning agent, regardless the types of other agents. Among different learning types, the primary agent in general performs best when it is a learning agent of 0-level, who makes no behavioral assumption about other agents.

As in many experiments, our results are not conclusive. There are many possible ways to construct 1-level and 2-level agents, which might lead to better performance than 0-level agents. But our experiments do suggest that an agent's performance is sensitive not only to its own beliefs (which is reflected in the level of recursion), but also to other agents' beliefs (levels of recursion). More sophisticated modeling does not always lead to better performance than less sophisticated ones.

Our future research remains to explore the Nash equilibrium solution of multi-agent interactions, which is viewed as a further step from recursive modeling. An even further step is to explore the solution to the changing behaviors of others, which can arise from mutual learning.

References

Cliff, D. (1998). Evolving parameter sets for adaptive trading agents in continuous double-auction markets. In: Agents-98 Workshop on Artificial Societies and Computational Markets, Minneapolis, MN, pp. 38–47.

Filar, J., & Vrieze, K. (1997). *Competitive Markov decision processes*, Springer-Verlag.

Friedman, D. (1993). The double auction market institution: a survey, in. In: Friedman, D., & Rust, J. (Eds.), The Double Auction Market, pp. 3–26.

Friedman, D., & Rust, J. (Eds.), (1993). The double auction market, Addison-Wesley.

Gmytrasiewicz, P. J., Durfee, E. H., & Wehe, D. K. (1991). A decision-theoretic approach to coordinating multiagent interactions. In: Twelfth International Joint Conference on Artificial Intelligence, Sydney, pp. 62–68.

Gode, D., & Sunder, S. (1993). Lower bounds for efficiency of surplus extraction in double auctions. In: Friedman, D., & Rust, J. (Eds.), The Double Auction Market, pp. 199–220.

Harsanyi, J. C., & Selten, R. (1988). *A general theory of equilibrium selection in games*, MIT Press.

Hu, J., & Wellman, M. P. (2000). Experimental results on Q-learning for general-sum stochastic games. In: Seventeenth International Conference on Machine Learning, Stanford, CA, pp. 407–414.

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In: Fifteenth International Conference on Machine Learning, AAAI Press, Madison, WI, pp. 242–250.

Laner, M., & Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: Seventeenth International Conference on Machine Learning, Stanford, pp. 535–542.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In: Eleventh International Conference on Machine Learning, New Brunswick, pp. 157–163.

McAfee, R. P., & McMillan, J. (1987). Auctions and bidding. *Journal of Economic Literature 25*, 699–738.

Nadella, R., & Sen, S. (1997). Correlating internal parameters and external performance: Learning soccer agents. In: Weiß, G. (Ed.), Distributed artificial intelligence meets machine learning: learning in multi-agent environments, Vol. 1221 of lecture notes in artificial intelligence, Springer-Verlag, pp. 137–150.

Owen, G. (1995). *Game theory*, 3rd ed., Academic Press, San Diego.

Rust, J., & Miller Palmer, J. R. (1993). Behavior of trading automata in a computerized double auction market. In: Friedman, D., & Rust, J. (Eds.), The Double Auction Market, pp. 155–198.

Sandholm, T. W., & Ygge, F. (1997). On the gains and losses of speculation in equilibrium markets. In: Sixteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan, pp. 632–638.

Satterthwaite, M., & Williams, S. (1993). The bayesian theory of the k-double auction. In: Friedman, D., & Rust, J. (Eds.), The Double Auction Market, pp. 99–124.

Thusijsman, F. (1992). In: Optimality and Equilibria in Stochastic Games, Centrum voor Wiskunde en Informatica, Amsterdam.

Uzawa, H. (1962). On the stability of Edgeworth barter process. *International Economic Review 3*(2), 218–232.

Varian, H. R. (1992). *Microeconomic analysis*, 3rd ed., W. W. Norton, New York.

Vidal, J. M., & Durfee, E. H. (1998b). The moving target function problem in multi-agent learning. In: Third International Conference on Multiagent Systems, IEEE Press, Paris, France.

Vidal, J. M., & Durfee, E. H. (1998). Learning nested agent models in an informational economy. *Journal of Experimental and Theoretical Artificial Intelligence 10*(3), 291–308.

Wellman, M. P., & Hu, J. (1998). Conjectural equilibrium in multiagent learning. *Machine Learning 33*, 179–200.

ELSEVIER

Journal of Cognitive Systems Research 2 (2001) 233

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Erratum

Erratum to ‘‘Learning about other agents in a dynamic multiagent system’’

[Cognitive Systems Research 2 (2001) 67–79]☆

Action editor: Ron Sun

Junling Hu[a,*], Michael P. Wellman[b]

[a]*Simon School of Business, University of Rochester, Rochester, NY 14627, USA*
[b]*Computer Science & Engineering, University of Michigan, Ann Arbor, MI 48109-2110, USA*

The Publisher wishes to correct the spelling of the name of the second author of this paper: Michael P. Wellman.

☆PII of original article: S1389-0417(01)00016-X
*Corresponding author.
E-mail address: huju@simon.rochester.edu (J. Hu).

1389-0417/01/$ – see front matter
PII: S1389-0417(01)00034-1