On measuring the usefulness of modeling in a competitive and cooperative environment

Leonardo Garrido

Ramón Brena
Centro de Inteligencia Artificial, Tecnológico de Monterrey

Katia Sycara
The Robotics Institute, Carnegie Mellon University

Abstract

This paper presents recent results of our experimental work in quantifying exactly how useful is building models about other agents using no more than the observation of others' behavior. The testbed we used in our experiments is an abstraction of the meeting scheduling problem, called the *Meeting Scheduling Game*, which has competitive as well as cooperative features. The agents are selfish, and use a rational decision theoretic approach based on the probabilistic models that the agent is learning. We view agent modeling as an iterative and gradual process, where every new piece of information about a particular agent is analyzed in such a way that the model of the agent is further refined. We present our *Bayesian-modeler agent* which updates his models about the others using a Bayesian updating mechanism. We propose a framework for measuring the performance of different modelling strategies and establish quantified lower and upper limits for the performance of any modeling strategy. Finally, we contrast the performances of a modeler from an individual and from a collective point of view, comparing the benefits for the modeler itself as well as for the group as a whole.

Introduction

Several approaches in the field of multiagent systems (MAS) (Durfee 1991; Wooldridge & Jennings 1995) make heavy use of *beliefs* as an internal model of the world (Bratman 1987) One form of belief of particular importance in multiagent systems are the agent's beliefs about other agents (Vidal & Durfee 1997b). This kind of belief could come from a preexisting knowledge base (a kind of "prejudice"), or could be inferred from observing others' behavior.

The purpuse of a modelling activity could be to benefit a specific agent, in the case of "selfish" agents, or to improve the performance of a group as a whole, in the case of cooperative agents -or even a combination of both. In real life there are many situations where cooperation and competition are present simultaneously. Collective sports are a good example, and for automated agents, we have simulated sports like the Robocup automated soccer competition (Stone, Veloso, & Riley 1999).

Now, concerning the modelling activity, in the soccer competition example it could be beneficial to learn the other teams' strategy in order to exploit their weaknesses. Of course, in this soccer example it is not interesting for one player to model its team mates, as all the players in one team are normally designed by the same programmers. Indeed, good real soccer teams improve as the different players get used to their mates' styles and reactions. We see a broad spectrum ranging from purely competitive situation (like zero-sum games), in one extreme, to purely cooperative situations where everybody gets benefits from a better performance, in the other extreme; there are of course many intermediate situations. In this work, we investigate such an intermediate scenario, where there are individualistic as well as collective issues.

Research on modeling other agents has been approached from different perspectives. Carmel and Markovitch (Carmel & Markovitch 1996), for example, have presented an heuristic algorithm to infer a model of the opponent's strategy, represented as a Deterministic Finite Automaton (DFA), from its input/output behavior. The work of Mor et al (Mor, Goldman, & Rosenschein 1996) also sees agent strategies as DFA, showing that a class of automata can be learned in polynomial time. Another interesting work on opponent modeling has been presented by Sen and Arora (Sen & Arora 1997) who propose a scheme for learning opponent action probabilities and a maximum expected utility strategy for exploiting weaker opponents.

Tambe et al (Tambe & Rosenbloom 1996) have proposed an approach for tracking recursive agent models based on a plan recognition task. Gmytrasiewicz (Gmytrasiewicz 1996) has presented the Recursive Modeling Method (RMM) which uses nested models of other agents, combining game-theoretic and decision-theoretic mechanisms. Suryadi and Gmytrasiewicz (Suryadi & Gmytrasiewicz 1999) have proposed the use of influence diagrams for learning models about other agents. Vidal and Durfee (Vidal & Durfee 1996) have developed an algorithm in order to see which of the nested models are important to choose in an effective manner. These authors have also presented a framework for determining the complexities of learning nested models (Vidal & Durfee 1997a).

In the robotic soccer domain there have been other related papers. For instance: mechanisms for learning partners and

competitors' skills as ratios of effectiveness (Nadella & Sen 1997) and the use of Hidden Markov Models to recognize strategic behaviors (Han & Veloso 1999).

We view agent modeling as an iterative and gradual process, where every new piece of information about a particular agent is analyzed in such a way that the model of the agent is further refined, using a Bayesian updating mechanism. There have been other papers sharing this view, for instance: Gmytrasiewicz et al (Gmytrasiewicz, Noh, & Kellogg 1998) have proposed a framework for Bayesian updating of agent models within the formalism of the RMM; Zeng and Sycara (Zeng & Sycara 1998) have presented an experimental research where a buyer models the supplier under a Bayesian representation in Bazaar, a sequential decision making model of negotiation,

In this particular research, we are interested in evaluating, in an experimental way, the advantage an agent can obtain by building models about the others' roles and strategies. This advantage is, in the first place, taken from a "selfish" or individualistic point of view. Later on, we complemented this research with an analysis of the collective benefits (or damages) resulting from a modelling activity. In (Garrido, Brena, & Sycara 1998) we presented our experimental framework and reported preliminary experiments using some non-modeling strategies. In this paper, we present our experimental research in exploring a range of strategies from least- to most-informed in order to evaluate the upper- and lower-limits of the modeler agent performance. Later on, we discuss the benefits that could be drawn from the modelling activity from a collective point of view.

In the following sections, we first review our experimental framework. We present the basic non-modeling and modeling strategies. Then, we present our experimental scenarios and discuss the results we have obtained. Finally, we present the conclusions of this paper.

Experimental Framework

We have implemented the *Meeting Scheduling Game (MSG)* (Garrido & Brena 1998) as our experimental testbed which models some characteristics of the distributed meeting scheduling problem. Our main concerns creating this test bed were: to allow self-interested as well as cooperative behavior, show or hide players' private information, and define different players' roles and strategies.

In this game, a group of agents try to arrange a meeting in such a way that certain meeting slot is available for as many as possible players. So that each player tries to arrange a meeting at a convenient and free time slot with an acceptable *utility* for him.

Each player's *role* is defined by a preference profile which is coded as a calendar slot utility function, ranking each slot from the most preferable slot to the least preferable one. We have defined several agent roles. For example, the following are some basic and familiar agent roles:

The Early-Rising. It prefers the early hours of the day.

The Night-Owl. It prefers the meetings to be scheduled as late as possible.

The Medium. It prefers the meetings to be around noon.

The Extreme. It prefers to have meetings early in the morning or late in the afternoon.

Figure 1 shows examples of these roles with four arbitrary eight-slots calendars.

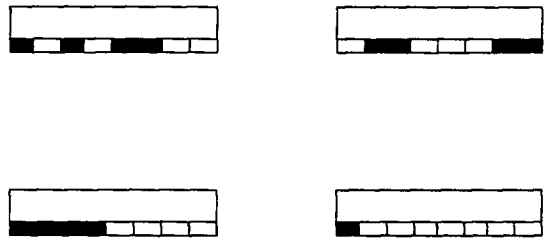

Figure 1: Four basic agent roles with eight-slots calendars. Black bars represent arbitrary busy slots.

Player's *strategies* are rules that tell them what actions to choose at each decision point. Strategies can take into account only the own player's preference profile or they can even use models about the others. In the subsequent sections we will define several different strategies.

Since a combination of a role and a strategy defines a player's preferences and behavior, the conjunction role/strategy of a player is seen as his *personality* in the MSG.

Each player proposes a slot taken from his own calendar composed of a working day with eight hours. Each player's calendar is set at a specific *calendar density* which is the proportion of busy hours in the calendar. The goal of a player in the MSG is to accumulate more points than his competitors in the game. A *game* consists of a predefined number of *rounds* and each player tries to accumulate points after each round.

There is a *referee* who ensures that all the players obey the rules of the game. He is also responsible for accumulating points for each agent after each round in an individual point counter for each player through the whole game.

After each round, each player's calendar is randomly reset, scrambling free and busy slots, maintaining the same predefined calendar density. Then, another round is started and the process is repeated until the predefined number of rounds is accomplished. Note that this implies we are not really "schedulling" any meetings, as the winning slots does not stand from a round to the next.

In each round, every player simultaneously proposes a slot according basically to his own individual strategy and role. However, the players' proposals are not completely determined by their own personalities because some slots can be busy in their calendars. In the first round, each player randomly proposes an available slot. These initial random proposals are needed as a "bootstrap" for the collaborative strategy defined in the following section. The other strategies are not affected by this initial round, since this is the only round where nobody accumulate points.

After all the players make their proposals, several *teams* are formed. Each team is composed of all those players who

proposed the same calendar slot. Then, each *team joint utility* is calculated, summing up all the team members' calendar utilities:

$$TJU(t) = \sum_{\forall m \in t} U_m(s_t)$$

Here, t is a team, m is a member of the team, s_t is the slot proposed by members in t, U_m is the slot utility of member m. Finally, the round is won by the team which accumulates the greatest team joint utility.

Once the winning team is selected, each agent earns points according to the following predefined *scoring procedure:* all the players outside the winning team accumulate zero points for that round and each agent a in the winning team t accumulates his own slot utility plus the team joint utility:

$$G_a(s) = TJU(t) + U_a(s_t)$$

The purpose of this mixed procedure is to promote a balance between selfish and collaborative attitudes. Finally, the winner of the whole game is who gets the highest accumulated points at the end of the last round.

Basically, this game is a competitive one, since each player's goal is to accumulate more points than his competitors over a series of independent meetings (i.e. rounds). However, each player needs to collaborate by joining in a team that will eventually make him win. Furthermore, sometimes some meetings agreements are more convenient than others from a collective point of view, in the sense that the collective utility is greater, but that can not be necessarily true from the individual point of view of some players.

Although this game is based on the general distributed meeting scheduling problem, it resembles only some of its characteristics.

Basic strategies

We set a framework for characterizing all the possible strategies in the MSG, ranging from a least-informed to the most-informed one. This allows us to place every given strategy in a framework where it can be better compared to others, and in particular to place modelling strategies in context.

The lower and upper limits of our framework are given by the following strategies:

Indifferent Strategy: An agent using this strategy chooses his next proposal among his action set using an uniform (equiprobable) distribution.

Oracle Strategy: An agent using this strategy can see in advance the others' next move because he knows the other agents' calendars, roles and strategies. For each free slot s in his calendar, he calculates his possible gain $G_o(s)$, if he proposed that slot. Then, he finds the agent m who would earn the maximum gain $G_m(s)$ among the rest of the players, if he proposed that slot. Then, he calculates the utility of each slot s as his gain with respect to the profit of agent m:

$$\mathbf{U}(s) = G_o(s) - G_m(s)$$

After checking all his free slots, he proposes the slot with the highest utility: $\arg\max_s \mathbf{U}(s)$.

An *indifferent agent* does not take into account any information about the other agents. He does not even take into consideration his own preferences. However, he must propose a free slot in his calendar, as must do all the other strategies as well. This strategy is considered as the lower limit for every "reasonable" strategy, since a strategy performing worse than the random is hardly worth considering.

An *oracle agent* knows the roles and strategies of the other agents (i.e. he has the correct models about the others). Furthermore, he even knows the others' calendars. So that an oracle agent is able to see in advance the others' moves and then he just chooses to propose the slot that maximizes his utility in each round of the game. Although an oracle agent has the best chances of winning each round, he can not always win! This is because of his random calendar availability, according to the fixed calendar density.

In order to have additional points of reference, we have also defined the following two heuristic strategies:

Self-Centered Strategy: This strategy tells the agent always to choose the free slot which just maximizes his own calendar slot utility.

Collaborative Strategy: Using this strategy, the agent chooses the free slot that was proposed by the biggest team (greatest number of members) at the previous round. In case of ties, the agent ranks them according to his own calendar slot utility.

These strategies were motivated by the observation of real human beings playing the MSG. A *self-centered agent* does not consider information about the other agents but he takes into account his role. A *collaborative agent* also takes into account the agent's own role. However, it also takes into consideration information about the previous round, trying to join in the biggest observed team.

Modeling strategies

Let us first introduce our term *model* about another agent. We just see it as a vector which records a probability distribution of the actual character of the modeled agent. In the context of the MSG, each agent has two basic models about each other agent a. The first one is the *role model*:

$$\vec{r}_a \stackrel{\text{def}}{=} (r_1, \ldots, r_n)$$

Where each r_i is the probability that agent a has the particular role i and n is the amount of different predefined roles. The notation $r_a(i)$ refers to the probability r_i of role i. The second model used in the MSG is the *strategy model*:

$$\vec{s}_a \stackrel{\text{def}}{=} (s_1, \ldots, s_m)$$

Where each s_i is the probability that agent a has strategy i and m is the amount of different predefined strategies. The notation $s_a(i)$ refers to the probability s_i of strategy i.

Since we are assuming independence between roles and strategies in the MSG (section), it is easy to construct a new combined model for each other agent: the *personality model.* This model is just a two-dimensional matrix, $\vec{rs}_a$, where each element $rs_a(i,j)$ is just calculated as follows:

$$rs_a(i,j) = r_a(i)s_a(j)$$

Now, let us define an decision-theoretic strategy that take explicit advantage of knowing the others' models:

Semi-Modeler Strategy: This strategy tells the agent to choose the slot which maximizes his expected utility based on predefined fixed models about the other agents.

It is assumed that a *semi-modeler agent* already have models about the others and his strategy just uses these probabilistic models to choose the action that maximizes his expected utility. The models are given to the semi-modeler agent at the beginning of the game and they never change during all the game. It is also important to note that the given models are not necessarily correct models about the others.

The detailed semi-modeler's strategy is as follows:

1. For each other agent a, generate his personality model $\vec{rs}_a$ and generate a set O all the possible opponent scenarios that the semi-modeler could face. Each possible opponent scenario $o \in O$ is just a possible particular combination of the possible personalities of the other agents.
2. For each $o \in O$:
   (a) Assuming that this possible opponent scenario o represents the actual personalities of the other agents, run the oracle strategy in order to get the best slot s_o, and its utility $\mathbf{U}(s_o)$, to propose under this assumption. Let us call r the outcome due to the action of choosing slot s_o.
   (b) Calculate the probability $\mathbf{P}(r|s_o)$ which is indeed equal to the probability of this possible scenario o: just the product of the probabilities in $\vec{rs}_a$ corresponding to each agent a's personality involved in this opponent scenario o. On the other hand, the utility of this outcome, $\mathbf{U}(r)$, is precisely the utility $\mathbf{U}(s_o)$ that was obtained in the previous step.
   (c) In order to incrementally get the expected utility of s_o:

$$\mathbf{EU}(s_o) = \sum_i \mathbf{P}(r_i|s_o)\mathbf{U}(r_i)$$

   Calculate the product $\mathbf{P}(r|s_o)\mathbf{U}(r)$ and accumulate it to previous products in other previous possible scenarios where the slot s_o had been chosen.
3. Propose the slot s_m with maximum expected utility: $\arg\max_{s_o} \mathbf{EU}(s_o|o)$

In order to build a modeler agent, model construction is required. Let us define a modeler strategy that uses an Bayesian updating mechanism in order to build the others' models in an incremental and iterative way:

Bayesian-Modeler Strategy: An agent using this strategy incrementally builds models about the others using a Bayesian belief updating approach and chooses the action which maximizes his expected utility:

A *Bayesian-modeler agent* does not have any information about the others. However, as stated in section , the set of predefined roles and strategies are public knowledge. At the beginning, the modeler agent can behave as a semi-modeler agent with equiprobable models about the others. That is, with no other knowledge about the others, it is reasonable to start with equiprobable probability distributions of the possible traits about the others. Then, the modeler agent can start to update those models based on the others' behavior.

This agent builds models about the other agents in an incremental and iterative way, updating those models after each round during the whole game. All the probabilities of each model are incrementally updated, trying to reach the actual character of the agent being modeled.

The detailed Bayesian-modeler strategy is as follows:

1. At the first round, start with equiprobable models about the others, run the semi-modeler strategy, and propose the resulting slot.
2. At the next round, for each other agent a:
   (a) Observe what was the a's proposal, s_a, in the previous round and update a's personality model, $\vec{rs}_a$, using a Bayesian updating mechanism to obtain the corresponding posterior probabilities of the a's personality, $per_a(i,j)$, given that a proposed slot s_a, $pro_a(s_a)$, in the previous round:

$$rs_a(i,j) = P(per_a(i,j)|pro_a(s_a))$$

   (b) Decompose the updated a's personality model in order to build two new separated role and strategy models. That is, update each element in $\vec{r}_a$ and $\vec{s}_a$:

$$r_a(i) = \sum_{\forall j} rs_a(i,j)$$

$$s_a(j) = \sum_{\forall i} rs_a(i,j)$$

3. Using the new updated models about the others, run the semi-modeler strategy and propose the slot s_m with the maximum expected utility.
4. If it was the last round, the game is over. Otherwise go to step 2.

The model-updating mechanism is based on the well known *Bayes' rule*. The simplest form of this rule is in the case of boolean random variables:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Where A and B are random variables with boolean domain. Bayes' rule provides us a way of calculating a posterior probability based on known prior probabilities and a conditional probability. From basic probability axioms and algebra is it easy to see that:

$$P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B)$$

Combining the two last equations and taking into account multi-valued random variables we can get a more general

form of Bayes' rule. Let us rewrite it using the probability P notation:

$$\mathbf{P}(Y|X) = \frac{\mathbf{P}(X|Y)\mathbf{P}(Y)}{\sum_{\forall y \in Y} \mathbf{P}(X|Y)\mathbf{P}(Y)}$$

Here, P denotes probability distributions and this last formula actually denotes a set of equations relating individual entries in the probability distributions (it does not denote matrix or vector multiplications).

In the case of our Bayesian-modeler agent, we indeed have multi-valued random variables: the personality models. In fact, a personality model $\vec{rs}$ represents a probability distribution of personalities. So that the probability that an agent a has the personality resulting from combining role i and strategy j, $P(per_a(i,j))$, is precisely the value $rs_a(i,j)$ in matrix $\vec{rs}_a$ and the equation used to update each personality model (step 2.1) can be rewritten as follows:

$$rs_a(i,j) = \frac{P(pro_a(s_a)|per_a(i,j))P(per_a(i,j))}{\sum_{\forall x}\sum_{\forall y} P(pro_a(s_a)|per_a(x,y))P(per_a(x,y))}$$

The prior probabilities $P(per_a(i,j))$ are taken from the last recorded value $rs_a(i,j)$ in matrix $\vec{rs}_a$. On the other hand, the conditional probabilities $P(pro_a(s_a)|per_a(i,j))$ can be calculated from the known calendar density and the known agent behavior due to the personality $per_a(i,j)$.

Let us see a simple example in order to see how this conditional probabilities can be calculated. Suppose you have an agent a which is being considered to have the personality resulting of combining the early-rising role and the self-centered strategy (i.e. $per_a(early, self)$). Now, suppose that the calendar density has been predefined and is equal to 0.5. Furthermore, suppose also that we have eight-slots calendars from slot s_0 to s_7. In this case, the conditional probabilities for each slot s_i, for $i = 0, ..., 7$, given this personality are:

$$P(pro_a(s_i)|per_a(early, self)) = 1/2^{i+1}$$

It is easy to see that the conditional probability of choosing slot s_0 given $per_a(early, self)$ is $1/2$ (because this value is precisely the calendar density and it is the most preferred slot of a early-rising agent), the probability of choosing slot s_1 is $1/2^2$ (the probability that the first slot is busy and the second one is free), the probability of choosing slot s_2 is $1/2^3$, and so forth.

Thus, going back to step 2.1, the Bayesian-modeler is able to get all the posterior probabilities from the calculated conditional probabilities and the known prior probabilities. Then, this $\vec{rs}$ matrix is updated with these new probabilities in order to be used as prior probabilities in the following round.

Experimental results

Here, what we call an *experiment* is a series of games with the same characteristics and groups of different and related experiments are called *experimental scenarios*. At the beginning of each game, all the agents are initialized with random roles taken from a set of two opposite roles (the early-rising and night-owl roles presented in section) and eight-slots calendars with the calendar density fixed at 50%. All the games are composed of ten rounds (the fourth and fifth experimental scenarios are the exceptions). Also in all these experiments, we run three agents (the exception is the second experimental scenario). Furthermore, when we run a Bayesian-modeler agent, he is always learning the models about the others and playing the game at the same time.

We have set up series of games in order to measure how agent performance is affected by different strategies. Once a game is completed, we call it a "success", if the strategy under consideration wins. Otherwise it is considered a "failure". Our experiments are composed of 500 independent games and we have calculated that the results obtained in these experiments has a 95% confidence of getting an error not greater than about 0.05. In all tables presented here, we show the performance of each strategy as the percentage of success.

The goal of the first scenario is to compare the performance of the non-modeling strategies discussed in section . Thus, we run here an indifferent agent first against self-centered agents, then against a collaborative ones, and finally against both:

Experimental Scenario 1			
Experiments	Strategies		
	Indifferent	*Self-Centered*	*Collaborative*
Experiment 1.1	7.59%	92.41%	—
Experiment 1.2	18.15%	—	81.85%
Experiment 1.3	3.86%	80.59%	15.45%

As expected, the performance of the indifferent strategy is always the worst, giving us a lower-limit performance to compare other reasonable strategies. We intuitively thought that the performance of the collaborative agents should be better because they can team each other. However, as we can see, in the first two experiments, the self-centered strategy appears to be better than the collaborative one against the indifferent agent. In the last, experiment, we can see that the self-centered strategy clearly outperforms the collaborative one, while the indifferent's performance is very low.

As it is shown elsewhere (Garrido, Brena, & Sycara 1998), when incrementing the number of agents, the collaborative's performance increases, outperforming the self-centered.

The goal of the second experimental scenario is to characterize the performance of the oracle and modeling strategies presented in section . Here we run four experiments with a self-centered agent, a collaborative one, and we vary the strategy of the third agent in each experiment. In the first experiment we run an oracle agent who has the correct models about the others. In the second one, we run a semi-modeler agent who uses fixed equiprobable models. In the third experiment, we again run a semi-modeler agent but now with fixed opposite models about the others. In the last one, we finally run a Bayesian-modeler who is learning the models and playing at the same time during the ten rounds of each game:

Experimental Scenario 2				
Exp	Strategies		Modeling	
	Self-Centered	*Collaborative*	*Models*	*Perform.*
2.1	20.58%	10.88%	Correct	68.54%
2.2	33.13%	11.31%	Equiprobable	55.56%
2.3	51.96%	20.19%	Opposite	27.85%
2.4	30.31%	7.67%	Learning	62.02%

In the first experiment, we get the empirical upper-limit performance given by the oracle strategy. On the other hand, running a semi-modeler with the incorrect fixed opposite models, we expect to have a lower-limit performance. We can see, in the third experiment, that this limit is indeed so low, being even the self-centered strategy the winner. The second experiment, shows a new more refined lower-limit performance, given by a semi-modeler with fixed equiprobable models. So that, our expectations for a good modeler performance is to get a performance somewhere between the upper-limit given by the oracle and the lower-limit given by the semi-modeler with fixed equiprobable models. As we can see, our expectations were confirmed in the last experiment.

The goal of the third scenario is to evaluate the performance of the Bayesian-modeler, varying the number of rounds needed to learn the models about the others in each experiment:

Experimental Scenario 3				
Exp	# Rounds	Strategies		
		Self-Centered	*Collaborative*	*Modeler*
3.1	1	40.21%	15.90%	43.89%
3.2	3	35.60%	16.75%	47.65%
3.3	5	33.40%	14.63%	51.97%
3.4	7	29.92%	12.35%	57.73%
3.5	9	29.02%	9.41%	61.57%
3.6	11	25.15%	10.40%	64.45%
3.7	13	25.61%	8.86%	65.53%

In the first experiment, we can observe how the Bayesian-modeler performance is very low after the first round but it is not as bad as the semi-modeler with fixed opposite models (previous scenario). Looking at the results of all the experiments, it is also clear how his performance improves as the number of rounds increases. As we can see, after eleven or thirteen rounds the performance is indeed already very close to the oracle performance.

In figure 2, we show a summary of the Bayesian-modeler performance. Here, we can directly compare the different performances we have obtained with the indifferent, oracle, semi-modeler, and Bayesian-modeler strategies when playing against the self-centered and collaborative ones. As we can observe in games with only one round, the Bayesian-modeler strategy performance starts with a performance between the limits of the semi-modeler strategies using fixed opposite and equiprobable models. This performance increases when we increase the number of rounds in the games, trying to reach upper-limit given by the oracle strategy.

The group perspective

So far the modeller performance has been measured in terms of the individualistic utility it obtains from the modelling activity. In this paragraphs we examine how the group's utility is affected by a modelling activity carried out by one of the agents in the group.

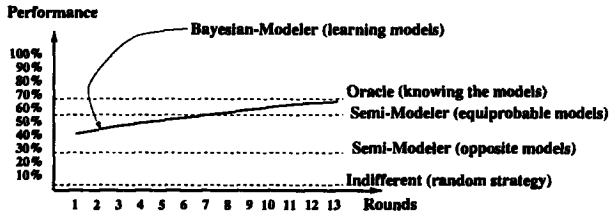

Figure 2: Graphical summary of the Bayesian-modeler strategy performance.

In this scenario we had 3 agents, one self-interested, one collaborative, and the third which is the modeller. Actually we used, instead of the Bayesian modeller itself, the two extreme cases of it, which are the oracle (representing a case when the modeller has learned correct models), and the semi-modeller, with equiprobable models (representing a case when the modeller is about to start learning, thus cancelling the modelling activity).

We ran two experiments, each one of 500 games with 10 rounds each. In them we measured the group utility by simply adding the utilities of the group members.

In the first experiment, with the oracle, the group joint utility was 333.292.

In the second experiment, with the semi-modeler with equiprobable models, the joint utility was 326.912.

We can see that the group performance was not, in this case, very much affected by the modelling activity.

Conclusions

We presented our Bayesian-modeler agent which is capable of building probabilistic models of its competitors in an incremental and iterative way. The modeling mechanism used by the Bayesian-modeler has two main characteristics:

- The decision-theoretic approach chooses the rational decision at each round of the game maximizing the modeler's utility with respect to the gain of the most dangerous opponent.
- The Bayesian updating mechanism is capable of building models about the others in an iterative and incremental way after each round.

We have used a collection of reference points for the characterization of the modeler agent's performance. The indifferent and oracle strategies provide the extremes of the spectrum, ranging from least- to most-informed strategies. We have also obtained other more refined performance limits given by the semi-modeler strategy with fixed opposite and equiprobable models. Our experimental results have shown how the Bayesian-modeler strategy performance is indeed better than the empirical lower-limits we have obtained and, in fact, we have also observed how this performance increase as the number of rounds increases. Our experiments

have also shown that after thirteen rounds the modeler performance is really close to the oracle one.

Our experiments also showed that, though the individualistic performance is greatly raised, the group utility is not really affected by the modelling activity of one of its members. Of course, having just one modeller could not be as effective as if each member of the group is a modeller, but in this case we would need to recursively model the other modellers, using methods like RMM (Gmytrasiewicz & Durfee 1995). Though this is in principle possible, the computational complexity of the simulations would be increased far too much, even for off-line simulations like ours.

We conjecture that our Bayesian modelling methods are optimal with respect to the use of the available information. This means that no other learner could outperform our Bayesian learner in getting correct models as fast as possible. This conjecture has been strengthen after comparisons with other learning methods, like reinforcement learning (Singh, Norvig, & Cohn 1997) (this comparison is reported in a forthcoming report), but we also intend to establish a formal proof of this optimality.

References

Bratman, M. E. 1987. *Intentions, Plans, and Practical Reason*. Cambridge, MA, USA: Harvard University Press.

Carmel, D., and Markovitch, S. 1996. Opponent modelling in a multi-agent systems. In Weiss, G., and Sen, S., eds., *Lecture note in AI, 1042: Adaptation and Learning in Multi-agent Systems*, Lecture Notes in Artificial Intelligence. Springer-Verlag.

Durfee, E. H. 1991. The distributed artificial intelligence melting pot. *IEEE Transactions on Systems, Man and Cybernetics* 21(6):1301–1306.

Garrido, L., and Brena, R. 1998. The meeting scheduling game: A multiagent testbed. Technical Report CIA-RI-037, Center for Artificial Intelligence, ITESM-Campus Monterrey, Monterrey, N.L., México.

Garrido, L.; Brena, R.; and Sycara, K. 1998. Towards modeling other agents: A simulation-based study. In Sichman, J.; Conte, R.; and Gilbert, N., eds., *Multi-Agent Systems and Agent-Based Simulation*, volume 1534 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag. 210–225.

Gmytrasiewicz, P. J., and Durfee, E. H. 1995. A rigorous, operational formalization of recursive modeling. In Lesser, V., and Gasser, L., eds., *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, CA, USA: AAAI Press.

Gmytrasiewicz, P.; Noh, S.; and Kellogg, T. 1998. Bayesian update of recursive agents models. *Journal of User Modeling and User-Adapted Interaction* 8(1/2):49–69.

Gmytrasiewicz, P. 1996. On reasoning about other agents. In *Intelligent Agents II*, Lecture Notes in Artificial Intelligence (LNAI 1037). Springer Verlag.

Han, K., and Veloso, M. 1999. Automated robot behavior recognition applied to robotics soccer. In *IJCAI-99 Workshop on Team Behavior and Plan Recognition*.

Mor, Y.; Goldman, C.; and Rosenschein, J. 1996. Learn your opponent's strategy (in polynomial time)! In Weiss, G., and Sen, S., eds., *Lecture note in AI, 1042: Adaptation and Learning in Multi-agent Systems*, Lecture Notes in Artificial Intelligence. Springer-Verlag.

Nadella, R., and Sen, S. 1997. Correlating internal parameters and external performance: Learning soccer agents. In Weiss, G., ed., *Distributed Artificial Intelligence Meets Machine Learning - Learning in Multiagent Environments*, Lecture Notes in Artificial Intelligence. Springer-Verlag. 137–150.

Sen, S., and Arora, N. 1997. Learning to take risks. *AAAI-97 Workshop on Learning Agents*.

Singh, S.; Norvig, P.; and Cohn, D. 1997. Agents and reinforcement learning. *Dr. Dobb's Journal of Software Tools* 22(3):28–32.

Stone, P.; Veloso, M.; and Riley, P. 1999. The cmunited-98 champion simulator team. In Asada, and Kitano., eds., *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag. 61–76.

Suryadi, D., and Gmytrasiewicz, P. 1999. Learning models of other agents using influence diagrams. In *IJCAI-99 Workshop on Agents Learning About, From, and With other Agents*. John Wiley and Sons.

Tambe, M., and Rosenbloom, P. 1996. Architectures for agents that track other agents in multi-agent worlds. In *Intelligent Agents II*, Lecture Notes in Artificial Intelligence (LNAI 1037). Springer Verlag.

Vidal, J., and Durfee, E. 1996. Using recursive agent models effectively. In *Intelligent Agents II*, Lecture Notes in Artificial Intelligence (LNAI 1037). Springer Verlag.

Vidal, J., and Durfee, E. 1997a. Agents learning about agents: A framework and analysis. In *AAAI-97 Workshop on Multiagent Learning*.

Vidal, J. M., and Durfee, E. H. 1997b. Analyzing agents that learn about agents. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, 849–849. Menlo Park: AAAI Press.

Wooldridge, M., and Jennings, N. R. 1995. Intelligent agents: Theory and practice. *Knowledge Engineering Review* 10(2):115–152.

Zeng, D., and Sycara, K. 1998. Bayesian learning in negotiation. *Internation Journal in Human-Computer Systems* 48:125–141.