Action-Based Environment Modeling for Maintaining Trust

Özgür Kafalı
Department of Computer Engineering
Boğaziçi University,
TR-34342, Bebek, İstanbul, Turkey
ozgurkafali@gmail.com

Pınar Yolum
Department of Computer Engineering
Boğaziçi University,
TR-34342, Bebek, İstanbul, Turkey
pinar.yolum@boun.edu.tr

***Abstract*— In open multiagent systems, agents need to select whom to trust. Traditionally, this selection is done based on the models that are built for the agents in the system. After each interaction with others, agents update their models of others based on the outcome. However, building and maintaining accurate models is difficult especially before many interactions take place. Contrary to traditional modeling approaches, we propose to model the environment in terms of agents' actions and their effects, rather than building individual models for each agent. Based on the effects of its actions, each agent can modify its behavior appropriately. We evaluate our proposed approach in comparison to a traditional approach in the Agent Reputation and Trust (ART) Testbed simulation environment. The simulations compare the two approaches in terms of the accuracy of models, the effectiveness in finding trustworthy agents as well as the effort needed to build accurate models.**

I. Introduction

Open multiagent systems are composed of autonomous and heterogeneous agents. Because of their autonomy and heterogeneity, these agents request services with varying characteristics as well as provide services with different qualities. Hence, not all agents can provide the requested services with the expected accuracy or detail. This creates the problem of trust. That is, each agent needs to identify whom to trust in different contexts [4].

Trust about other agents which can be achieved via three ways: (1) by using the individual dimension of trust, that is the agent tries to build trust about an agent using its previous interactions with it, (2) by using the social dimension of trust in which the agent builds trust about the agent relying on the opinions of other agents about that agent (i.e., the agent's reputation in the society), (3) by using a combination of the individual and social dimensions [15], [6].

In this paper, trust is studied in the context of service selection. More specifically, we consider a setting where the agents are both service providers and service consumers depending on the context of the interactions they are involved in. Since the agents may not be proficient in giving all the services properly, they may need to get help from others. So when the agent is in need of a service (i.e., the role of service consumer) from another agent (i.e., the role of service provider), it needs to identify a useful service provider. In order to do so, it may use its previous interaction history with the service providers (i.e., individual dimension of trust) as well as the experiences of other service consumers that it trusts (i.e., social dimension of trust). This is a common setting that has been used in other previous research on trust [12], [16], [14].

In the above setting, we study the problem of modeling the environment around the agent accurately and in a reasonable time. The model can be revised over interactions [3]. Together with learning, the agent can update its current knowledge of the environment with the recent data it collects (possibly from others). Based on the built models, the agent can decide whether a particular provider can be trusted for a particular service. If the agent has modeled its environment well, it is more probable for it to find the best service providers. But even if the model is accurate enough, the service providers may respond erratically since their behavior is not perfectly predictable. At this point, the agent has to have a means of recording this kind of behavior and learn it. A successful combination of modeling and learning will provide a significant advantage to the agent over others in the environment.

Traditionally, the environment is modeled using an *agent-based modeling* approach in which each agent models others individually. However, this has some shortcomings. First, to build an accurate model of an individual, one needs significant amount of training data. That is, an agent must interact with another agent several times to build a good model of the other agent. Second, it is difficult to decide what percent of the individuals should be modeled. If an agent models everyone else, then there will be a blow up in the amount of modeling that has to be done. On the other hand, if the agent models only a few of the other agents, then these agents may not be adequate to serve the agent well.

Accordingly, we propose an *action-based modeling* approach that is based on modeling the actions of the agent and their effects on the environment, rather than modeling the agents in the environment individually. The agent improves its model of the environment by considering the effects of the actions through a learning algorithm. That is, if an action has been successful in a particular setting, then the agent has an incentive to repeat that action in the same setting. An agent can quantify how successful the action has been and rank its actions based on their outcomes. This gives the agent a

principled method to choose among its actions.

We compare the action-based and agent-based modeling approaches using the ART Testbed Environment [5]. ART Testbed has been chosen since it simulates the setting described above closely. Several experiments are run with different agent societies (i.e., environments consisting of heterogeneous agents that are implemented with either modeling algorithms). The simulations compare the approaches with different metrics such as the success in finding useful service providers, the number of service providers that are contacted to receive a service, and finally the profit of the agent when the costs for receiving service information and the service itself is considered. The experiments are repeated for several times to decrease the effect of the randomness in the simulation environment. Our results show that an action-based modeling approach makes fewer errors and achieves higher precision compared to an agent-based modeling approach.

The rest of the paper is organized as follows: First, we explain the technical background, including the ART testbed and Frost, an agent that uses agent-based modeling approach for trust in Section II. Then, we introduce Blizzard in Section III, an agent that models the environment through its actions rather than the agents it interacts with. Next, we introduce our metrics in Section IV, and compare the two agents with the main focus on their modeling in Section V. Then, we describe different versions of Blizzard that differ in update, and request behavior, in Sections VI, VII, respectively. Lastly, we discuss relevant literature and conclude the paper in Section VIII.

II. Technical Background

In order to evaluate our work, we have used the ART Testbed simulation environment [5]. We will first describe the game environment in detail and then explain Frost, an agent that uses agent-based modeling to manage trust in the ART Testbed.

A. The ART Testbed

The ART game simulates a business environment in an art appraisal domain where customers try to get their paintings evaluated by the participating service provider agents. Each agent that participates in the game is a service provider (i.e., appraiser) that is selling its opinion when requested. Service consumers as well as other service providers may be willing to purchase opinions about a painting. Each painting belongs to an era. Every agent has a predefined expertise value in each era, which remains constant throughout the game. The expertise values are distributed among the agents randomly before the simulation begins.

The game is a series of rounds, and at each round agents try to evaluate the paintings assigned to them by the customers. When the round ends, all evaluations are compared to the actual values of the paintings and the relative appraisal errors are determined. In the next round, the number of assigned paintings are determined accordingly (i.e., less error leads to more client share). When an agent needs to evaluate a painting but lacks expertise in a given era, it may query other agents to see if they have a better evaluation of the painting. In addition, the service providers can query each other to find out the reputations of other agents for some eras. Throughout the simulation, there are no guarantees about the correctness of replies. That is, a service provider may provide wrong information about a provider or may provide inaccurate information about the reputation of another service provider. Similarly, since agents are heterogeneous (i.e., designed by different parties), they may be following different strategies for carrying out their tasks.

After a service provider consults whoever is necessary, it forms a final opinion about the requested painting. After submitting this final opinion, the true value of the painting is revealed. This allows a service provider to compare its opinion with the correct value. The opinions that are similar to the correct value of the painting are assumed to be correct. The mentioned activities, such as requesting opinions from others have costs. Similarly, answering a query has monetary benefits. The main aim of the game is to end with the maximum money (termed bank balance in the simulations).

B. Agent-Based Modeling Agent: Frost

In order to test the performance of Blizzard, we have compared it to another agent (Frost) through simulations. Frost has performed successfully previously; i.e., ranked third in the 2006 ART Competition. Frost uses an agent-based modeling approach and models every other agent in the environment. Its overall strategy can be studied in three main parts: the modeling of the agent's environment, the requesting strategy, and the response strategy [11].

The Modeling of the Agent's Environment: In order to model its environment, Frost keeps an estimation of every other agent's expertise values for each era in the game. The modeled expertise value denotes how well the agent answers queries about paintings. It also keeps other useful information about the agents such as how many times the agent is contacted for its opinion about a specific era.

Frost learns the expertise of other agents iteratively by applying update operations on its models. The update of agent models takes place at the end of each round, when the actual painting values are revealed. Only then, the agent has a means of calculating the error in each appraisal it received from others.

The algorithm in Figure 1 shows how the agent updates the expertise values of other agents that it is modeling. To put it simply, the agent's modeled expertise is increased when it provides one more correct answer (the value of its answer is above a threshold value); otherwise it is decreased. However, the amount to be increased and decreased are important. In the algorithm, the variable *expertMargin* ensures that when the agent's modeled expertise is closer to the maximum value (i.e. 1 in this case), its expertise is incremented slowly. The same reasoning does not apply for the case in which the expertise is decremented, since being cautious in understanding that an agent is not useful can bring along many wrong answers and cause loss of money.

```
initialize the expertise values
for all receivedOpinions do
    error = |appraisedValue − trueValue|/trueValue
    if error < threshold then
        expertMargin = 1 − expertise
        updateMargin = expertMargin ∗ (threshold − error)
        expertise = expertise ∗ (1 + UpdateMargin)
    else
        updateMargin = threshold − error
        expertise = expertise ∗ (1 + UpdateMargin)
    end if
end for
```

Fig. 1. Updating the Expertise Values

Example 1: Let the actual value of the painting be $10,000$ and the opinion received from the agent be $6,500$. If the predefined threshold value is 0.3 and the error value is 0.35, the error is greater than the threshold. So we decrease the expertise of the agent by an amount of $0.95(= 1+(0.3-0.35))$.

The Requesting Strategy: The requesting strategy helps the agent decide whom to query for opinions and reputation information. When the agent needs an opinion about a painting, it may first query the environment to gather reputation information about others. However, Frost builds its trust model solely on the opinion queries. The reason for this is purely pragmatic. Frost has participated in the 2006 ART Competition, where five agents competed against each other at a time. When the number of agents in the environment is small, collecting reputation information becomes redundant. Hence, Frost only uses opinion requests.

The motivation behind the requesting behavior of Frost is simple. First, it sets the maximum number of agents to query and calculates the threshold value for the expertise of agents to be queried for each era. For example, if the expertise of Frost is high in a given era, it asks for fewer opinions and sets the threshold value slightly higher than its own expertise to get better opinions than it can generate. Then, it makes an ordering of agents based on the modeled expertise values and selects the top agents passing the threshold value.

The Response Strategy: Frost uses its response strategy to decide to whom and in what way it will respond to the queries directed to it. The response strategy also determines how reputable it will be within its environment. But since the agents in the environment are competitive, there is no immediate benefit for being reputable. Also, by providing useful information to others, the agent would help others to earn money, which is unintuitive in a competition setting.

According to its response strategy, Frost may choose to mislead others by its answers. This can be done in one of the two ways; (1) by sending out wrong or exaggerated opinions or (2) by giving inaccurate information about one's reputation in an era. For the case of opinion responses, Frost chooses to generate wrong appraisals when its expertise lacks in the given era. The main reason for doing so is the belief that it does not expect any income by selling its opinions for those painting eras (i.e., an intelligent opponent will not buy further opinions when it receives a bad answer). On the other hand, it tries to be as accurate as possible when it has enough expertise to keep its clients satisfied. This may increase its reputation in those eras, thus leading to more income from opinions.

III. Action-Based Modeling Agent: Blizzard

Contrary to Frost, Blizzard does not maintain models for each agent in the environment. Blizzard models its environment based on the actions it performs. It evaluates the quality of its previous actions using the feedback it receives from the environment. Thus, it tries to select the best possible action at any time of the simulation based on the results of its previous actions. Blizzard utilizes reinforcement learning to maintain its models of the environment.

Typically, agents that employ reinforcement learning learn their environment through trial and error interactions [10]. The agent has an opportunity to select from a variety of actions which will either lead to rewards or punishments as consequences (each action has a reinforcement value in return). The agent's primary goal is to maximize this reinforcement value in the long run. Instead of building a separate model for each agent in the environment, Blizzard records the history of actions it performs throughout the simulation. Each action has a corresponding reinforcement value calculated differently according to the complexity of the action performed.

Blizzard uses an extension of reinforcement learning, called Q-learning, to model its actions [8]. The variation of the Q-learning embedded in the agent's architecture enables Blizzard to build reinforcement values for each action it takes. In its original form, a Q-value function maps the state-action pairs of the agent to the corresponding Q-values [13]. However, in the ART Testbed, the agent does not have an explicit state (i.e., a specific location in the game, or current situation among the others). So, we can safely assume that the Q-learning scheme utilized by Blizzard only maps its actions to the corresponding reinforcement values, no matter what state the agent is in. Next, we will define the possible actions that Blizzard can take at each round in the game.

A. History of Actions

Blizzard records the following four actions:

Opinion Requests: An opinion request action is recorded when the agent sends out a query that specifies its opinion request from another agent in a given era. For the first few rounds of the simulation, Blizzard tries to ask as many agents as possible about their opinions in different eras to populate its opinion request actions. This is done since the associated reinforcement values will be used to prepare better queries in the following rounds.

The reinforcement value of an opinion request can be calculated after the true values and the appraised values are revealed at the end of the round. The reinforcement values are distributed among the actions according to the appraisal errors

relative to the average appraisal error computed for the subject era in that round. If the specific action's appraisal error is less than the average appraisal error for that era, it is certain that the reinforcement value will be positive (i.e., a reward), but the amount is calculated according to the distance from the average appraisal error.

Reputation Requests: Once a reputation request action is recorded after an actual reputation query is sent out, it is harder to calculate its reinforcement value as there is no clear measure to evaluate it like the case for opinion requests. But the agent may make use of the previously recorded opinion request actions to compare the received value with the internally modeled value and compute a reinforcement value for the action performed.

Opinion Responses: For the response actions, there is no accurate way of predicting the quality of the response generated. One way of evaluating the reinforcement value of the action may be to check the incoming opinion requests in the following round for the same painting era. If the incoming requests for that era increase relative to all incoming requests, the agent evaluates the action as a reward. Likewise, if the percentage of requests belonging to that era decrease, the action is assigned a punishment value according to the level of decrease.

Reputation Responses: Like in the opinion response case, the agent may update the reinforcement value of its reputation response actions by checking the incoming reputation requests in the same era for the next round.

B. Generation of Reinforcement Values

After a timestep is finished in the game, each agent has a chance to evaluate its actions since some useful information is revealed by the simulator (i.e., separate appraisal errors the agent receives from other agents). For each recorded action, there is a reinforcement value that shows how rational it was to take that action (and possibly how rational it will be in the future).

When generating the reinforcement values for the opinion request actions, the agent first calculates the appraisal error for the opinion it purchased and then compares it with the overall error for that era. Then, it generates the reinforcement value as follows:

$$r = \begin{cases} maxR & , d \leq 0 \ \& \ d \leq tRew \\ maxR * d/tRew & , d \leq 0 \ \& \ d > tRew \\ maxP & , d > 0 \ \& \ d > tPun \\ maxP * d/tPun & , otherwise \end{cases}$$

, where r is the reinforcement value, d is the difference between error for the opinion and the average error for that era, $tRew$ is the threshold value for rewards, $tPun$ is the threshold value for punishments, and $maxR$ and $maxP$ are the maximum reward and punishment values, respectively.

If the appraisal error is less than the overall error for the era (i.e., $difference \leq 0$), then it calculates the difference between the two error values and assigns a reward for the opinion request action by the amount of difference. The greater the difference is, the greater the reward will be. If the received opinion has more error than the era average, it is given a punishment value as reinforcement. Here, we set the values of *thresholdReward*, *thresholdPunishment*, *maxReward*, and *maxPunishment* to be -0.5, 0.3, 10, and -10; respectively.

Example 2: Let the received opinion have 0.1 error value associated with it and the average error for that era be 0.35. Since the received error is less than the average value, the action will be assigned a reward, and the amount will be $(10 * (-0.25/-0.5)) = 5$.

C. Overall Strategy

Based on its action history, Blizzard divides its strategy into some parts. The overall strategy consists of the generation of reinforcement values, the requesting behavior, and the response behavior. The details of the first two strategies will be given separately in the following two sections. We will propose different approaches for each strategy, and compare them in the experiments. Since our main concern here is on modeling, we will not focus on the details of the response behavior.

IV. Evaluation Metrics

The results of the simulations are evaluated using the following metrics. Since modeling is expected to improve over time, we study the values of the following metrics over time.

1) *Average appraisal error:* Average appraisal error measures the appraisal error agents make when evaluating their paintings. Recall that, after an agent generates an opinion for a painting, the true evaluation of the painting is revealed. This allows the agent to calculate its overall error from the previous round.
2) *Number of opinions purchased:* This metric counts the number of opinions that an agent buys. This is important to know since from the perspective of the competition buying too many opinions is costly.
3) *Precision:* Informally, this metric measures how well an agent finds the useful service providers in the environment. The metric is designed to measure the percentage of contacted service providers that are actually experts in their eras. Equation 1 measures the precision for a particular era j.

$$Precision(j) = \frac{\sum_{i=0}^{numAgents} p_{ij} * e_{ij}}{h_j} \quad (1)$$

Here, p_{ij} is the percentage of queries directed to agent i in the jth era. In measuring this, we have recorded the number of times each agent has been directed queries and calculated the percentages accordingly. e_{ij} is the actual expertise of agent i in era j. The actual expertise values can be gathered from the simulation engine. The value of this multiplication increases when more queries are directed to the relative experts in the environment. However, there may not always be the same level experts for each era. Hence, we need to normalize the metric. To do so, for each era we divide it by the expertise of the most expert agent (h_j) in that era. Then we average

this value over all eras to get an average metric value between 0 and 100.

4) *Bank balance:* Bank balance is used to evaluate the overall success of the agent. It shows the total of the agent's income minus the agent's expenses over the game rounds. Recall that the agent's income mainly stems from correct opinions it generates (either using its own expertise or using opinions from others) and it's expenses stem from buying opinions from others. So, it is for the agent's benefit to lower its average appraisal error since low error will lead to more customers. It is also good for the agent to lower its purchase of opinions. Because, it not only is an expense for the agent, but also is a source of income for other agents.

In the ART competition, the bank balance is the definitive metric for winning the competition. However, here since our main focus is on modeling, the other three metrics are also essential. The precision signals how accurately an agent actually models the experts around it. Notice that precision does not measure if all the experts have been identified or not. The precision will still give a high value even when one good provider has been identified. This is acceptable for us since finding one expert is generally enough to generate good opinions. It might seem as high precision should guarantee low average appraisal error: when right agents are found, fewer mistakes should be made. However, identifying the right service providers do not always mean that these service providers will answer requests or will answer them correctly. Thus, average appraisal error measures another dimension of modeling. Finally, the number of agents contacted is important to assign a cost to modeling. An agent may model another accurately but if this takes too many interactions then the modeling approach may not be applicable in some settings.

V. Comparison of Blizzard with Frost

We begin our evaluations through the comparison of Blizzard with Frost. The version of Blizzard we have used in this experiment behaves as described in the previous sections. It may be considered as the base version of Blizzard and is accordingly called Blizzard1.

A. Experiments

For the first experiment, we compare Blizzard1 with Frost in order to see their performances, mainly on how they model their environment.

Figure 2 plots the four metrics, comparing Blizzard1 with Frost. The results are plotted as the averages of five games. Each game consists of 5 dummy agents, and 5 agents from either Frost or Blizzard1.

Average Appraisal Error: As the figure for the appraisal errors shows, it is obvious that Blizzard is evaluating its paintings better than Frost after a few timesteps have passed. This is related with the way it models its environment (i.e., whom to get opinions from) as well as how it prepares the final appraisals (i.e., opinion weights). Both affect the quality of the appraisals, but modeling the environment well has a wider effect, since the agent may not always be capable of preparing the appraisals depending only on its own opinions.

Opinions Purchased: When we consider the number of opinion transactions in the game, it seems that Blizzard tries to learn about its environment starting from the very first timestep. Although this may be very costly at the beginning, it helps better understand its surrounding, and select future actions accordingly. Also, the opinions purchased in the first rounds do not affect Blizzard's painting evaluations as it does not have to include all of them in making the final appraisal. Frost, on the other hand, seems not to get much opinions from others. This is related with its response strategy as well as its requesting strategy. Because, having cheated in some opinions, it becomes harder for Frost to get the opinions from others. So, it has to rely on its own opinions most of the time.

Precision: Blizzard is also more precise than Frost in finding other expert agents in the environment. As we have explained when talking about the opinion purchases, Frost's precision also drops after the third timestep noticeably since it cannot get good opinions from the actual experts in the environment.

Total Bank Balance: When the other metrics are taken into account, it is not surprising that Blizzard ends the game with noticeably higher bank balance than Frost. This is because, it makes fewer errors in painting appraisals, thus gets more client share each round, although its opinion costs are higher than that of Frost's. The high number of opinion purchases of Blizzard1 is the reason for Frost to keep the game ahead in the first half of the simulation.

VI. Effect of Reinforcement Value Generation

From this section on, we concentrate on other versions of Blizzard and compare their performances. Although Frost will exist in the simulations, we will not evaluate its performance further. We have previously shown that the base version of Blizzard (Blizzard1) already performs much better than Frost. Here, we consider two versions of Blizzard that differ in how they generate the reinforcement values for their previously taken opinion request actions.

Blizzard1: This is the base version of Blizzard as we have discussed before, and compared with Frost.

Blizzard2: This agent differs from Blizzard1 in terms of the way it evaluates the previous appraisals received form other agents. For each opinion requested, the agent first calculates the associated error, and then generates the reinforcement value as follows:

$$r = \begin{cases} maxR * (t - opErr) & , opErr \leq t \\ maxP * (opError - t) & , otherwise \end{cases}$$

, where r is the reinforcement value, t is the threshold value, $opErr$ is the error associated with the opinion, and $maxR$ and $maxP$ are the maximum reward and punishment values, respectively.

Here, the part that differs form Blizzard1 is that the agent does not consider the average error value for the era that the painting belongs to. But rather, it considers the error associated with the opinion separately. If the appraisal error is less than the error threshold for the era (i.e., if the agent is also an expert

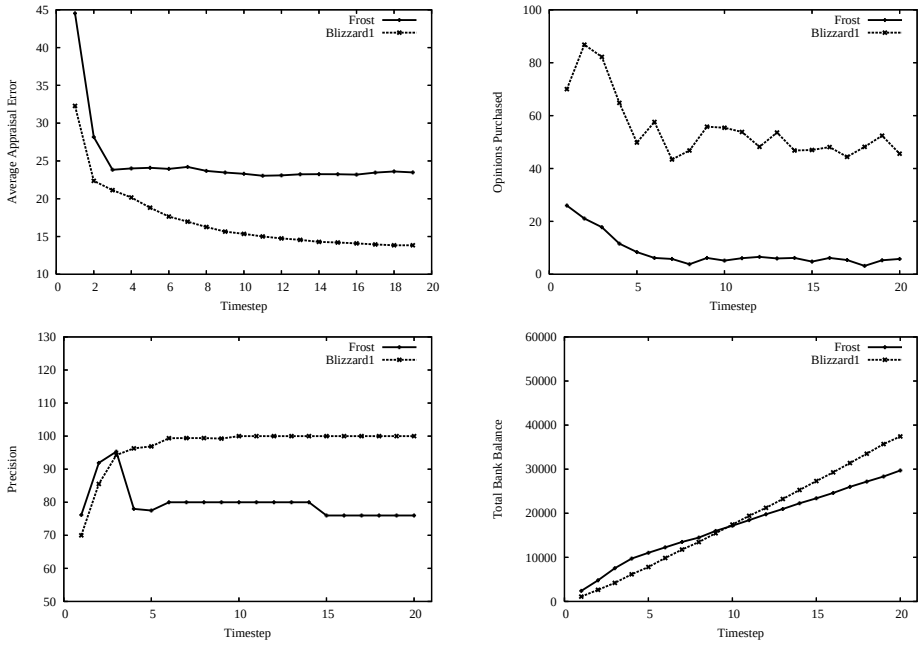

Fig. 2. Blizzard vs. Frost

for that era, the threshold will be small), then it calculates the difference between them and assigns a reward for the opinion request action according to the amount of difference. The better the opinion is, the greater the reward will be. If the received opinion has more error than the threshold value, it is given a punishment value as reinforcement. Here, the values for *maxReward*, and *maxPunishment* are again 10, and -10; respectively as in the previous case.

Example 3: Let the received opinion have 0.1 error value associated with it and the error threshold for that era be 0.3. Since the received error is less than the threshold value, the action will be assigned a reward, and the amount will be $(10 * (0.3 - 0.1)) = 2$.

A. Experiments

In this experiment, we try to see the how different reinforcement value generation approaches effects the agent's performance in the game. The competitors of this experiment are the two versions of Blizzard (Blizzard1 and Blizzard2) that are explained above.

Figure 3 plots the four metrics, comparing the different approaches that Blizzard employ on generating the reinforcement values. The results are plotted as the averages of five games. Each game consists of 10 dummy agents, 5 Frost agents, and 5 agents from either Blizzard1 or Blizzard2.

Average Appraisal Error: The appraisal evaluations show that Blizzard2 makes fewer errors than Blizzard1 starting from the third timestep. Since the difference between the two versions is how they evaluate their opponents, the reason for that error difference is the way they generate the reinforcement values. As Blizzard2 evaluates an appraisal based on its deviation from the average error for that era, it may not always make a true evaluation (i.e., an opinion may seem good compared to the average value, but if the average itself is as worse as a bad opinion, than the evaluation is wrong).

Opinions Purchased: Because of the same fact we argued above, Blizzard1 gets much more opinions than its opponent (since it may have misjudged a majority of its appraisals). The variations in the number of opinions purchased also justifies this fact, because the same appraisal may be evaluated as a good opinion in some timestep, but it may be evaluated as a bad one for another timestep.

Precision: When the agents are compared according to their precision, both seem to be precise in terms of the metric values. But Blizzard2 is more precise, since it evaluates its opponents better by separately considering their error values in their appraisals.

Total Bank Balance: The difference between the bank balances of the two agents are huge, since Blizzard2 makes fewer errors in its evaluations and is able to find the experts in the environment by asking fewer agents (thus, spending less on opinion purchases).

VII. Effect of Requesting Behavior

The requesting strategy of the agent determines which agents to request opinions and reputation information from. If we consider the opinion requests case, at each round Blizzard first tries to predict the experts in the environment for its assigned paintings (i.e., according to the previously assigned reinforcement values) and queries them to get their opinions. Figure 4 shows how Blizzard decides whom to ask for opinions.

According to the algorithm in Figure 4, the agent first checks its assigned paintings for the given round of the

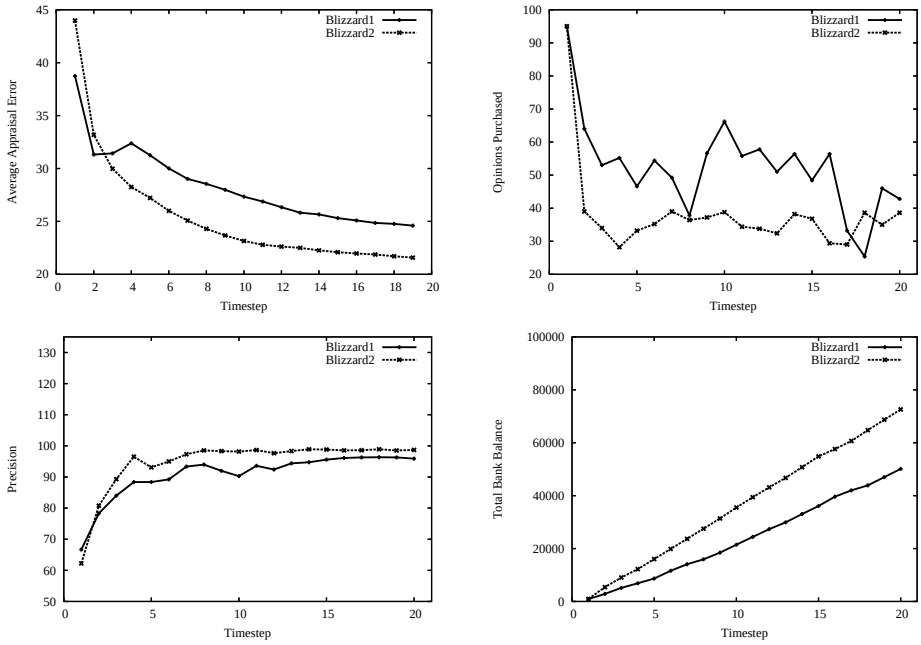

Fig. 3. Generation of Reinforcement Values

1: **for all** $assignedPaintings$ **do**
2: get past actions in the corresponding painting's era
3: **if** $noPastActionsRecordedForThatEra$ **then**
4: ask all agents for their opinions
5: **else**
6: list candidate agents by reputations
7: sort the list for decreasing reputation
8: query the top *n* agents
9: **end if**
10: **end for**

Fig. 4. Preparing Opinion Request Actions for Blizzard2

1: **for all** $reputationRequestActions$ **do**
2: get the reinforcement value for the recorded action
3: **end for**
4: $repAvg = calculateTheReputationAverage()$
5: **for all** $opinionRequestActions$ **do**
6: get the reinforcement value for the recorded action
7: **end for**
8: $opAvg = calculateTheOpinionAverage()$
9: $reputation = repAvg * repWeight + opAvg * opWeight$

Fig. 5. Finding Agent's Reputation

simulation, and gathers the history of previously recorded opinion request actions that belong to the same era with the assigned painting. If it is the case that it does not have any previously recorded actions for that era, it simply asks all agents about their opinions for the painting. Otherwise, it sorts the agents by their reputations according to the findings from the previous actions and selects the top *n* agents from the ordering to query for opinions.

In order to find an agent's reputation in a given era, the algorithm in Figure 5 uses a weighted average based calculation. First, it traces through the previous reputation request actions and gets their average, then it checks the opinion request actions and gets their average. After the averages are calculated, the final reputation is found by applying the weights. Here, we set the values of *repWeight*, and *opWeight* to be 0.1, and 0.9; respectively. The weights represent what proportion of the agent's model depend on the individual dimension of trust (i.e., opinion queries), and what proportion it depends on the social dimension (i.e., reputation queries).

Next, we introduce the agent types that will be used in the experiments to compare the requesting strategies.

Blizzard2: This is the agent that we have described in the previous section. It evaluates each opinion received separately and generates the reinforcement value accordingly.

Blizzard3: This version behaves differently when deciding whom to ask opinions from. It always keeps a classification of agents in its model which gives some clues on the behaviors of them. There are simply four classes of agents; neutral (no information gained about this agent yet), potential cheater (the agent is very likely to cheat), cheater (the agent is caught cheating in an era), and definite cheater (the agent cheats in more than one era).

In order for an agent to be considered as a potential cheater, it has to take at least one malicious act (i.e., responding with a bad opinion although it promised a good one) previously. If it continues its dishonesty in its responses, it will be considered a cheater. And a definite cheater is the one which cheats in any era, no matter when.

The algorithm in Figure 6 defines this behavior. When deciding to purchase an opinion, Blizzard now checks another

```
for all assignedPaintings do
    get past actions in the corresponding painting's era
    if noPastActionsRecordedForThatEra then
        ask all agents for their opinions
    else
        list candidate agents by reputations
        sort the list for decreasing reputation
        if agentNotClassifiedAsCheater then
            ask the agent for its opinion
        end if
    end if
end for
```

Fig. 6. Preparing Opinion Request Actions for Blizzard3

constraint about the agent, i.e., the agent's classification. Although the agent may be expert in an era, Blizzard may not get an opinion since if the agent has showed cheating behavior previously in other eras. This way, it may reduce the risk of receiving bad opinions.

A. Experiments

In the next experiment, we see the differences on performances of the two versions of Blizzard (Blizzard2 and Blizzard3) that employ different requesting strategies as told above.

Figure 7 plots the four metrics, comparing the different requesting behaviors of Blizzard. The results are plotted as the averages of five games. Each game consists of 10 dummy agents, 5 Frost agents, and 5 agents from either Blizzard2 or Blizzard3.

Average Appraisal Error: Blizzard3 improves its appraisals against Blizzard2 in terms of appraisal error by a noticeable amount. This is simply because it decreases the risk of getting a bad opinion by classifying its opponents according to their response behaviors (i.e., honest, cheater, etc.).

Opinions Purchased: The number of opinions purchased metric shows that Blizzard3 buys more opinions than Blizzard2. Although it lowers its requests because of the agent classifications it has (i.e., do not ask cheaters), it may also increase the number of purchases if its appraisal error for an era goes above a certain threshold value. As a result Blizzard3 buys more opinions than its opponent to maintain better appraisals.

Precision: According to the precision metric, Blizzard2 seems to be more precise than its opponent. The reason for Blizzard3 to be less precise is that, although it finds the similar experts as Blizzard2, it does not direct it queries to them because of their previous harmful behavior (i.e., cheat in other eras). Also, the need to buy opinions from agents with average expertise in order to maintain better appraisals decreases Blizzard3's precision.

Total Bank Balance: As a result of the overall performance of the agents, Blizzard3 ends the game with higher bank balance as its appraisal errors are remarkably less than Blizzard2.

VIII. Discussion

This paper studies an action-based approach for modeling the environment. We have compared this proposed approach to an agent-based modeling approach and showed that the action-based approach outperforms the agent-based approach. Further, we have shown different versions of the agent that employs the proposed approach that differ in the generations of rewards and responding.

According to the FIRE trust model [9], trust arises from two levels, the individual (direct experiences from past interactions) level and the society (observations by the society of agent's past behavior, reputation) level. Since the agents are self-interested and unreliable, and there is no central authority or control over the agents in open multiagent systems, each agent has to model trust for itself with incomplete knowledge about the environment. As the model describes, there are four ways to achieve trust, interaction trust and role-based trust that result from direct interactions, witness reputation telling the reports of others about an agent, and the concept of certified reputations which is the most interesting one. By a certified reputation, it is meant that the agent tries to certify its past behavior to the requesting agent. Since the multiagent systems differ on the environments they are settled, each of these approaches are not applicable to all platforms. For example, in the ART Testbed environment, there is no way for the agent to certify its past behavior to another one. So it is not possible to integrate this idea directly into the agent architectures we have implemented.

In the experimental setup that FIRE uses, there are the sociable consumers with no expertise. That is, they have to find others to get their tasks done, and there are the unsociable but expert providers who only serve when they are asked to. However, the concepts of consumers and providers are interchangeable according to the context in the ART Testbed. When we are talking about the real consumers in ART Testbed, they are just dummy clients who have no sociability or expertise. On the other hand, the agents that participate in the game may take the roles of consumers or producers according to the interaction type they are involved in.

SPORAS [16] also approaches trust from different view points as FIRE does. The agent in SPORAS also tries to build its trust model via evaluating the different sources of information it gathers from the environment. But, the main difference is that it gives emphasis on more recent findings while considering these various sources of information. This may be an interesting approach to model the environment. While processing the recent data, the agent may forget some of the information from its long term history. Again, this may have advantages or disadvantages according to the setting of the environment. In a cooperative environment where agents' expertise may vary throughout the simulation, considering recent data and ignoring the past information may lead to better modeling of the environment. However, in a competition setting any piece of information about an opponent may be useful and may have clues about its behavior. In this paper, we

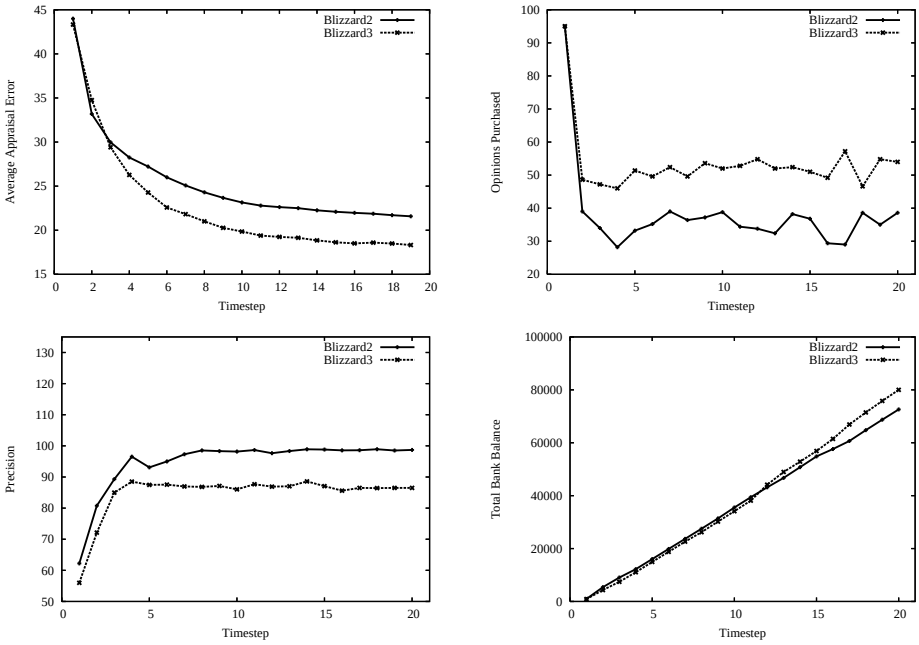

Fig. 7. Requesting Behavior

have preferred to treat each rating gathered from other agents equally, since the expertise of agents do not change throughout the game. Thus, each older response is as important as a more recent one.

While considering the social dimension of agents, REGRET tries to build an ontological structure to model the reputation of an agent [12]. By adding an ontological dimension, the model allows to combine reputations of different aspects. The idea shows some similarities to the trust models we have built for ART Testbed. An agent in ART Testbed may be judged in two ways, the reputation it has because of the opinions it gives about paintings (we call this the expertise of the agent in our models), and the reputation of the agent resulting from its precision in evaluating other agents (this is called the sociability of the agent). The combination of these two aspects is a good measure of the trustworthiness of the agent. But the agent may not always have accurate information about these while preparing its queries.

Reinforcement learning is very suitable for the context in ART Testbed. In every single step of the simulation, an agent is subject to make a decision which will either lead it to success or failure. A successful combination of these actions determines the agent's overall performance in the game. The correlation between the actions and their corresponding rewards is also studied previously [2]. Each action an agent takes, not only influences its own future in the game, but also affects the whole society or a part of it. ART Testbed is a two-sided game. In one side, the agent has to choose which information to obtain from others. On the other side, it has to decide on which information it will reveal to others and how (i.e., honestly or dishonestly). The former determines the agent's own future in the environment, but the latter may affect the whole society since the information revealed by the agent may be consumed by a majority of other agents.

Collaboration in multiagent systems is another interesting concept as experimented in the SPIRE system [7]. As opposed to the ART domain, agents in the SPIRE framework are more group-oriented (working for the sake of the whole society) than being self-interested. According to the setup, there is a team of agents that try to accomplish group tasks. A group task is a combination of small activities which can be carried out by individual agents according to their capabilities. Once a group task is complete, all the agents that participate benefit from it. But, to make the environment more interesting, there are also individual tasks presented by third parties, called the outside offers. The immediate gain from completing an individual task may be greater than accomplishing a group task, but when an agent accepts an outside offer that conflicts with a group task, the group task is left unfinished and the whole society is affected negatively by the incident. So, the agent has to have a mechanism to get the best utility out of the combination of tasks it completes. The collaborative side of the system weighs more than the competitiveness encouraged in the ART testbed simulation environment. Since all agents have individual assignments to complete, they only help others in cases where they also benefit from (i.e., selling their opinions in order to earn money).

Every agent participating in ART Testbed has to cooperate with others independent of the way it models the environment. As the agents may not always have high expertise in all the eras, making a group of cooperative agents (i.e., sharing opinions honestly) will benefit each member in the coalition [1]. This will both lower the costs of modeling (i.e., less opinions transactions since each agent in the coalition knows

whom to get help from about a specific assignment) the whole environment and help the agents make more accurate evaluations as the coalition will have an expert for each era (that is the aim of making a coalition). But this approach may not always be applicable in a competition environment, especially when the number of participants is low. However, if the competition is configured to allow competition among groups of agents, then the results of the group performances may be interesting to investigate.

ACKNOWLEDGEMENT

This research has been supported by Boğaziçi University Research Fund under grant BAP07A102 and The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073. We are indebted to the ART Development Team, especially Karen Fullam and Jordi Sabater for helping us with the testbed software and the anonymous referees for their comments on the paper.

REFERENCES

[1] S. Airiau, I. Goswami, and S. Sen. Expertise and trust-based formation of effective coalitions: an evaluation of the art testbed. In *Ninth International Workshop on Trust in Agent Societies, AAMAS*, pages 71–78, 2006.

[2] K. S. Barber and K. Fullam. Learning trust strategies in reputation exchange networks. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1241–1248, 2006.

[3] K. S. Barber and J. Kim. Belief revision process based on trust: Agents evaluating reputation of information sources. In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber-societies*, LNAI 2246, pages 73–82. Springer-Verlag, 2001.

[4] R. Falcone and C. Castelfranchi. The socio-cognitive dynamics of trust: Does trust create trust? In *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference*, pages 55–72, London, UK, 2001. Springer-Verlag.

[5] K. Fullam, T. B. Klos, G. Muller, J. Sabater, Z. Topol, K. S. Barber, J. S. Rosenschein, and L. Vercouter. The agent reputation and trust (ART) testbed architecture. pages 50–62. The Workshop on Trust in Agent Societies at The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, July 2005.

[6] K. K. Fullam and K. S. Barber. Dynamically learning sources of trust information: experience vs. reputation. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[7] A. Glass and B. Grosz. Socially conscious decision-making. In C. Sierra, M. Gini, and J. S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 217–224, Barcelona, Catalonia, Spain, 2000. ACM Press.

[8] M. Harmon. Reinforcement learning: a tutorial, 1996.

[9] T. D. Huynh, N. R. Jennings, and N. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *Proceedings of 16th European Conference on Artificial Intelligence*, pages 18–22, 2004.

[10] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[11] Ö. Kafalı and P. Yolum. Trust strategies for ART Testbed. In *Ninth International Workshop on Trust in Agent Societies, AAMAS*, pages 43–49, 2006.

[12] J. Sabater and C. Sierra. Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195, New York, NY, USA, 2001. ACM Press.

[13] E. Wiewiora, G. W. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. In *ICML*, pages 792–799, 2003.

[14] P. Yolum and M. P. Singh. Emergent properties of referral systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 592–599, 2003.

[15] P. Yolum and M. P. Singh. Service graphs for building trust. In *Proceedings of the 12th International Conference on Cooperative Information Systems (CoopIS)*, volume 3290 of *Lecture Notes in Computer Science*, pages 509–525. Springer-Verlag, 2004.

[16] G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14:881–907, 2000.