Adapting Reinforcement Learning For Trust: Effective Modeling in Dynamic Environments

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

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

***Abstract*—In open multiagent systems, agents need to model their environments in order to identify trustworthy agents. Models of the environment should be accurate so that decisions about whom to interact with can be done soundly. Traditional trust models are based on modeling specific properties of agents, such as their expertise or reliability. Building those models requires too many prior interactions to be accurate. This paper proposes an approach that is based on keeping track of outcomes of agent's actions towards others rather than modeling other agents' performances explicitly. Contrary to existing modeling approaches that require domain knowledge to build models, our proposed approach can be effectively realized in multiagent systems when the agent's actions are clearly identified. Comparisons with other modeling approaches in various environments reveal that our proposed approach can create more precise models in short time and can adjust its behavior quickly when other agents' behaviors change.**

***Keywords*-trust; reinforcement learning; modeling;**

I. Introduction

Agents in a multiagent system interact with each other to carry out their tasks. In many systems, agents are not equally trustworthy. In such settings, agents need to identify trustworthy partners among other agents by modeling their environment accurately. Existing trust models in the literature are based on inspecting interactions of agents to build models [1], [2]. Traditional trust models as introduced above try to model specific properties of agents. These properties depend very much on the context that the agents execute in, thus making the modeling phase very sensitive to the environment. Also, in order to make an accurate evaluation of those properties, the agent has to know significant amount of domain knowledge. For example, in a domain with service providers and consumers, consumers seek to have a means of the expertise values for different providers. Using that knowledge, they can expect better services in the future. But, in order to model the expertise values accurately, agents representing consumers must know what a good service is (i.e., which expertise values lead to what quality of services). Further, it is not clear how these models adapt when the expertise of providers change over time.

However, a modeling approach should be as *generic* as possible. That is, the application of the modeling approach should depend as little as possible to the *particular domain* it is being applied to. For example, if the modeling approach is being applied in the ART appraisal domain [3] or Web services domain, the same principles should apply without much modification. The modeling approach should model others as *precisely* as possible. That is, if an agent interacts with a second agent, then the interacted agent should be as trustworthy as possible compared to other possible alternatives in the environment. If the environment changes, the model should be able to *adapt* to the changes by modifying itself. Hence, even if the environment is dynamic, the modeling approach should converge to new models in time.

The modeling approach we propose is generic in the sense that it only needs the actions of the agents to be identified. It then uses the consequences of those actions to make an accurate evaluation. Intuitively, if we follow the example above with the consumers trying to model service providers, the consumer agents can use the consequences of their actions in order to judge how well the providers perform in their professions. No explicit domain knowledge is required in order to compare the consequences of several actions. Thus, the agent representing the consumer may adapt this same strategy to another domain with different action sets much faster than it would be in the traditional model (i.e., when using heuristics and domain knowledge). In order to measure the precision and adaptability of our proposed approach in various settings, we use the *ART Testbed* simulation environment [3].

The rest of the paper is structured as follows: Section II describes the details of our approach. Section III introduces a typical agent-based modeling approach, the metrics, and evaluates our approach. Section IV concludes the paper.

II. Action-Based Modeling Approach

We propose agents to model their environments by modeling the outcomes of their *actions* explicitly. An action is any message that an agent can send to any other agent, such as requesting the opinion of a particular agent. Contrary to traditional approaches, we do not require agents to model others individually. We assume that the agent that models its environment has a list of actions that it can perform and it is aware of this list. This is a reasonable assumption and holds in many realistic settings, including ART. When an agent performs an action, it receives feedback from the environment that is helpful in judging the usefulness of the action that is performed. After an agent performs an action, it evaluates the quality of its action using this feedback. Accordingly, it tries to select the best possible action based on the consequences of its previous actions. To decide which action is better for it, it uses reinforcement learning. Typically, agents utilizing reinforcement learning model their environments through trial and error interactions [4]. The agent has an opportunity to select from a variety of actions which will either lead to rewards or punishments as consequences. The agent's primary goal is to maximize its total reinforcement value.

Blizzard is our concrete agent implementation that adopts action-based modeling. Blizzard participated in the second ART competition in 2007, and it ranked third. Here, we use that agent as the basis for our action-based modeling approach [5]. Blizzard uses an extension of reinforcement learning, called Q-learning, to model its actions [6]. The variation of the Q-learning embedded in the agent's architecture enables Blizzard to generate a reinforcement value for each action it performs. In order to act in an environment, Blizzard needs only to be aware of its set of actions that are available in that environment. Unlike traditional trust models, it does not require further heuristics on the operational properties of the environment since it will not aim on predicting how other agents are performing. The main type of action that Blizzard records for the ART domain is the *Opinion Request* action which we explain next.

Opinion Request Actions: An *Opinion Request* action in ART corresponds to asking the opinion of an agent about a painting in a given era. There are two properties associated with each *Opinion Request* action; the *agent* to ask for the opinion and the *era* associated with the painting. We assume that this action formulation is inherited by the agent once it is put into the ART domain to operate. The outcome of taking an opinion from an agent in a given era is determined only by these properties whenever it is performed. Thus, the learning is stateless in the ART domain.

After performing an opinion request action, if a reward is received, it shows that the agent can request opinion from the same agent in the same era in the future. Similarly, a punishment advises the agent not to take the same action again. For the ART context, Blizzard also records other actions such as *Reputation Requests*, *Opinion Responses*, and *Reputation Responses*. However, we do not deal with the details of those actions here, since an *Opinion Request* action is the main type of action required for modeling the environment of the agent.

Generation of Reinforcement Values: Since the goal of the ART game is to end up with maximum bank balance, purchasing good opinions must be rewarded even though it incurs certain costs. However, purchasing bad opinions should be punished as they lead to incorrect painting evaluations. Thus, the agent learns how to collect good opinions throughout the simulation via reinforcements. In order to see the outcome of an opinion request, the agent waits for the current timestep to be finished. Then, it has a chance to evaluate its actions since some useful information is revealed by the simulator. 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, t is the threshold for separating rewards and punishments, $opErr$ is the error associated with the opinion, and $maxR$ and $maxP$ are the maximum reward and punishment, 10 and -10, respectively.

Query Protocol: The query protocol of Blizzard determines which agents to request opinions from. At each timestep, Blizzard tries to find agents which it expects to get good opinions from. For each appraisal, Blizzard fetches the actions associated with the same era as the assigned painting. In the first few timesteps of the simulation, it is most possible that this retrieval will result in no actions to be fetched. When this is the case, Blizzard explores the environment by simply asking all agents for their opinions. Otherwise, it sorts the fetched actions so that actions with the highest rewards are ranked first. Finally, the agents associated with the top actions are selected as the candidates for opinion transactions.

III. Experiments

In this section, we compare our approach to an agent-based modeling approach, and we use the agent *Frost* [7] as the basis for this purpose. Frost ranked third in the first ART competition in 2006, and thus is a good benchmark for our comparisons. Similar to most other traditional approaches, Frost models agents it interacts with and uses these models to choose whom to interact with in the future. 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 represents how well the agent generates opinions about paintings. We use the following two metrics to evaluate our approach.

Precision: Precision measures how well an agent finds the useful opinion providers in the game [5].

Total Bank Balance: Bank balance evaluates the overall success of the agent [3]. It shows the total of the agent's all income minus its expenses.

A. Agents in the Static Expertise Environment

For the first set of experiments, we compare the agents Blizzard and Frost in settings where the expertise of the agents do not change throughout the simulation. Figure 1 plots the *precision* metric, comparing Blizzard with Frost in both honest and cheating environments. We do not provide the *bank balance* plots, but rather show numerical results (averages of two different settings) for two reasons; first due to space considerations, and second the bank balance of Blizzard is significantly higher than that of Frost in all simulations. The results are plotted as the averages of three simulations. The agent population in the environment consists of 3 agents from both Frost and Blizzard, 16 honest agents, 4 cheating agents, and 4 dummy agents that are used in the 2007 ART competition. For the cheating environment, we again use 3 agents from both Frost and Blizzard, but this time there are 4 honest agents, 16 cheating agents, and 4 dummy agents. The behavior of the dummy agents is rather unpredictable unlike the honest or cheating agents which act consistently throughout the simulation. Because we need to measure the performances of Frost and Blizzard due to their modeling differences, we keep other properties of the agents exactly the same (i.e., no reputation information is used, same weights are used in generating the final opinions, they respond to all other agents honestly).

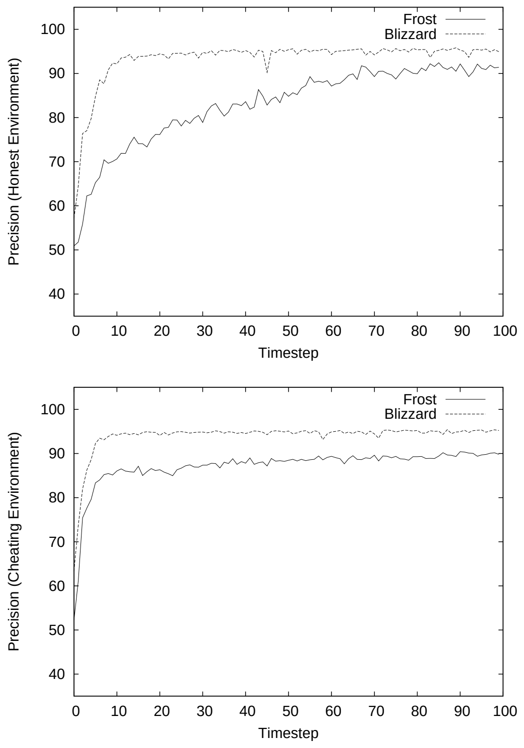

Figure 1. Agents in the Static Expertise Environment

Precision: Blizzard's precision in both the honest and cheating settings increases steeply and reaches the maximum value in the first few timesteps. This is very much related to the aggressive exploration strategy that Blizzard uses. Although it collects good opinions from other agents, it keeps on searching for better ones. Frost, on the other hand, has a significantly lower precision value in the first half of the simulation for the honest setting. Since Frost gathers enough data about its environment after the first half has passed, it progressively improves its precision and almost reaches Blizzard's precision value. For the cheating setting, Frost's precision improves faster, because it cannot find the good opinions at the beginning and explores its environment for better ones quickly.

Total Bank Balance: When we average the bank balances at the end of simulations for both settings, Frost ends up with 483743 and Blizzard ends up with 880043.

Note that the setting of the experiments also aim to compare the exploration strategies of the two agents. We have seen that in the trivial setting (where there are more honest experts for an agent to get opinions from), consistent exploration helps the agent gather more accurate opinions from the environment.

B. Self-Aware Agents in the Dynamic Expertise Environment

For the second set of experiments, we keep the agent populations same as in the first set of experiments. However, half of the honest agents in both honest and cheating settings now have dynamic expertise in the sense that their expertise values change after the first half of the simulation. If the agent's expertise is high prior to change, it decreases quickly. Similarly, if it's previously low, it increases quickly. Here, we improve the agents with a sense of self-awareness. That is, they respond to the drops sensed in their precision values (i.e., when their precision values drop for an amount greater than 10.0 from the maximum value ever reached, they initiate different actions). Frost increases the effect of its update operations from the point it senses the precision drop. So, it can increase the contribution of the newer findings while decreasing that of the older ones. Blizzard, on the other hand, discards half of its older actions from its action history in order to remove the effect that they have on selecting future actions. This is a previously experimented strategy in literature [4]. Blizzard also initiates a full exploration process on the environment. Figure 2 plots the two metrics, comparing Blizzard with Frost in both honest and cheating environments.

Precision: Interestingly in the honest setting, Frost's precision is not affected by the expertise changes at all. Since only half of the honest agents have their expertise values changed, it is possible that Frost has not explored enough to be aware of those agents. Therefore, it is not affected by the changes in their expertise values. Blizzard, on the other hand, is affected by the changes in the environment but

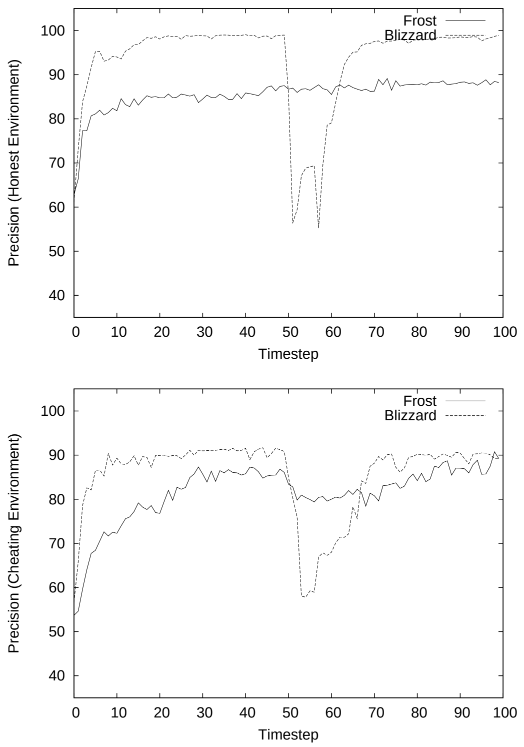

Figure 2. Self-Aware Agents in the Dynamic Expertise Environment

regains its highest precision values even after the expertise changes. In order to make this recovery faster, it explores the environment again when it senses the expertise changes.

Total Bank Balance: When we average the bank balances at the end of simulations for both settings, Frost ends up with 452904 and Blizzard ends up with 921978.

Note that the expertise changes that occur in these settings are sharp changes. That is, the expertise of an agent may decrease from its maximum value to the minimum value in five timesteps (or increase from its minimum value to the maximum value within the same time). We have also made other sets of experiments which we cannot show here due to space considerations. In those settings, we have started the expertise changes earlier and slower so that they continue almost throughout the whole simulation. We have obtained similar results where Blizzard recovers its highest precision value gradually after the expertise changes.

IV. Discussion

This paper studies an action-based approach for modeling an agent's environment. We have compared our proposed approach to a traditional trust seeking agent and showed that the action-based approach outperforms it in different settings. Such trust models have been created before [1], [2] using traditional approaches. Different dimensions of trust have been utilized in those approaches to build composite models of an agent's environment. We, on the other hand, relate those dimensions with the actions of the agent. The consequences of those actions are then evaluated to model the environment around the agent. Thus, Blizzard uses its own actions to act in the future. Since we consider trust as a service selection problem, the idea can be integrated to other problems where finding the useful service providers is the primary concern. The concepts *trustworthy* and *useful* can then be used interchangeably in such contexts. Because, if an agent has proved its usefulness previously by offering good services, than it will be considered as trustworthy when looking from the perspective of trust. So, we believe that our approach can be used in similar domains with ease. The precision metric we have built is a generic performance criterion that can be applied to different domains. It is a measure of how successful the agent is in finding the true service providers. However, some expert agents may not be willing to share their knowledge in some settings, thus lowering the precision values.

Acknowledgement

This research has been supported by Boğaziçi University Research Fund under grant BAP09A106P, The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073, and Turkish State Planning Organization under the project DPT 2007K120610.

References

[1] 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*, 2004, pp. 18–22.

[2] J. Sabater and C. Sierra, “Regret: Reputation in gregarious societies,” in *AGENTS ’01: Proceedings of the Fifth International Conference on Autonomous Agents*. ACM Press, 2001, pp. 194–195.

[3] 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.” The Workshop on Trust in Agent Societies at The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, 2005, pp. 50–62.

[4] L. P. Kaelbling, M. L. Littman, and A. P. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996. [Online]. Available: citeseer.ist.psu.edu/article/kaelbling96reinforcement.html

[5] Ö. Kafalı and P. Yolum, “Action-based environment modeling for maintaining trust,” in *Eleventh International Workshop on Trust in Agent Societies, AAMAS*, 2008, pp. 23–32.

[6] E. Wiewiora, G. W. Cottrell, and C. Elkan, “Principled methods for advising reinforcement learning agents,” in *International Conference on Machine Learning (ICML)*, 2003, pp. 792–799.

[7] Ö. Kafalı and P. Yolum, “Trust strategies for ART Testbed,” in *Ninth International Workshop on Trust in Agent Societies, AAMAS*, 2006, pp. 43–49.