MODELING, LEARNING, AND UTILIZING TRUST IN MULTIAGENT SYSTEMS

by

Remzi Özgür Kafalı

B.S., Computer Engineering, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems & Control Engineering

Boğaziçi University

2007

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Assist. Prof. Pınar Yolum for her encouraging support and guidance on making this thesis come through from the very beginning. I would also like to thank my committee members Prof. Levent Akın and Assoc. Prof. Necati Aras.

I would like to send my regards to the ART Development Team, especially Karen Fullam, Jordi Sabater and Tomas Klos for helping us with the testbed software. I also thank the reviewers for their comments on our AAMAS 2006 Trust Workshop paper.

I would also like to thank many others for keeping their support on me. My fellow friends and colleagues were always there when I was unable to think anymore. It is my pleasure to have friends like Utku Birdal and Türker Şahin, I could not survive the excitement of the ART Competition without them. I also thank my former IT Manager Mr. Hüseyin Akay for his faith in my research.

Everyone does not have the chance to have a brother so harmonic with himself. I thank him and his wife for cheering me up all the time I needed, with their help in our FRP sessions.

At last, I would like to thank my dear parents who always believe in where I'm going in life. They always supported what I wanted to do and made sure I was doing fine with their consistent calls.

This research has been partially supported by Boğaziçi University Research Fund under grant BAP06A103 and The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073.

ABSTRACT

MODELING, LEARNING, AND UTILIZING TRUST IN MULTIAGENT SYSTEMS

In open multiagent systems with autonomous and heterogeneous agents involved, achieving cooperation is necessary but difficult. Since every agent has limited and different capabilities, they need to exchange some of their tasks to get the maximum efficiency. For this purpose, they need to find out whom they can trust and delegate their tasks to be done. An important issue in building trust is to model other agents based on the previous interactions with them and decide on future relations by reviewing this useful information. However, building accurate models of others and updating them with recent findings is difficult since the agents do not always behave as expected regarding their autonomous behavior.

This thesis studies trust in the context of a service selection problem where the agents try to find the best service provider. Two learning algorithms that can be used to model the agent's environment are considered. The learning algorithms vary in terms of the models that they generate as well as their update behavior based on interactions. The learning algorithms have been evaluated using the Agent Reputation and Trust (ART) Testbed simulation environment. This platform is chosen since it fits the best to the experiments done in the thesis. The results of the simulations compare the two algorithms in terms of the accuracy of models, the effectiveness in finding trustworthy agents as well as the effort needed to build accurate models. Further, the algorithms are compared in terms of their robustness when some agents cheat or respond erratically.

ÖZET

ÇOK ETMENLİ SİSTEMLERDE GÜVEN MODELLEMESİ, ÖĞRENİMİ VE KULLANIMI

Özerk ve heterojen etmenlerin yeraldığı çok etmenli sistemlerde, etmenler arasında işbirliğinin sağlanması gerekli ama zordur. Her etmenin değişik konularda ve bir limit dahilinde becerisi olduğu düşünülürse, aralarında bazı işleri paylaşmaları her birinin etkinliğini arttıracaktır. Bu nedenle, kendilerinin yapamayacağı işler için güvenebilecekleri diğer etmenler bulmaları gerekir. Heterojen bir ortamda güven inşa etmenin önemli bir kısmı diğer etmenleri önceki davranışlarına göre modelleyip, bu veriyi daha sonraki etkileşimlerde kullanmaktan geçer. Fakat, özerk etmenler her zaman kendilerinden beklenildiği gibi davranmadıklarından, bu modellemeyi yapmak ve yeni verilerle geliştirmek zor olabilir.

Bu tezde, güven kavramını etmenlerin işlerini en iyi yaptırabilecekleri bir diğer etmen bulmak için uğraştıkları bir servis seçme problemi olarak görüp, bu problemi çözmeye yönelik iki öğrenme algoritması üzerinde duracağız. Bu iki algoritma, hem diğer etmenleri modelleme hem de bu modeli güncel verilerle geliştirme aşamasında birbirinden ayrılıyor. Bu algoritmalar Agent Reputation and Trust (ART) Testbed simulasyon ortamında değerlendirilmiştir. Bu platformun seçilmesinin nedeni, yapılmak istenen deneylere son derece uyum göstermesidir. Simulasyon sonuçları kurulan modellerin doğruluğu, güvenilir etmenleri bulmadaki etkinlik ve modeli oluşturmak için harcanan efor açısından değerlendirilmiştir. Ayrıca, algoritmalar bazı etmenlerin hile yaptığı veya değişken yanıtlar verdiği durumlara dayanıklılıklarına göre de karşılaştırılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

e_{ij}	actual expertise of agent i in era j
h_j	the most expert agent in era j
p_{ij}	percentage of queries directed to agent i in the jth era
ART	Agent Reputation and Trust
Blizzard	Action-Based Modeling Agent
Frost	Agent-Based Modeling Agent

1. INTRODUCTION

An agent is an entity (i.e., a software process) that perceives, reasons, and acts within an environment [1]. It may interact directly with its environment, it may interact with other agents in the same environment, or it may do both. In a multiagent system, several agents interact with each other. The type of interactions may change according to the context of the environment. The agents may cooperate, compete, or do both in order to achieve individual or group tasks. These agents may be autonomous and heterogeneous. Since they are autonomous, they can decide and change whom to interact with or may vary the quality of their interactions [2]. Since they are heterogeneous, it is difficult to make assumptions about their present or future behavior.

A multiagent system may be a closed multiagent system or an open multiagent system. An open multiagent system differs from a closed multiagent system since it allows arbitrary external agents to join and leave the system at any time. For a closed multiagent system, the set of agents is predefined by the controller of the system.

Trust is a crucial element in open multiagent systems. We can view the agent interactions in multiagent systems as business dealings. The characteristics of the business, the traits of the business participants, and the interactions they are involved in influence how trust is engendered. A difficult problem in open multiagent systems is finding out whom to trust [3]. Trust can be studied in two different view points: self confidence which the agent can build by using the knowledge it has about itself, and 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.

In this thesis, 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 [4, 5, 6].

In the above setting, we study the problem of modeling the environment around the agent accurately and in a reasonable time. Modeling means that an agent will predict how well a service provider can really offer a service and will revise its prediction over interactions [7]. Together with learning, the agent will update its current knowledge of the environment with the recent data it receives from others. Thus, accurate modeling is crucial to identify trustworthy agents in the following interactions by reviewing the received data. Based on the built models, the agent can decide whether a particular provider can be trusted for a particular service according to its previous performance. 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 discriminate the agent from others in the environment.

Accordingly, two agent models and two different learning algorithms to learn these models are studied in this thesis: agent-based modeling and action-based modeling, for predicting the actual expertise of service providers in a multiagent system. To compare the algorithms fairly, agents are developed that follow these strategies in the ART Testbed Environment [8]. 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 learning algorithms). The simulations are compared with four metrics that denote the average error each agent makes in providing its service to the customers, success in finding useful service providers, the number of service providers that are needed to be 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 remove probabilistic behavior that may be caused by the randomness of the simulation environment. The algorithms are also tested in cases where the behavior of the agents change considerably like cheating consistently or responding erratically.

The rest of this thesis is organized as follows:

Chapter 2 describes the ART Testbed environment, the concepts of modeling and learning, and reviews the relevant literature about trust. Section 2.1 introduces the ART Game, its simulation environment and the agent model to be implemented. Then Section 2.2 describes why modeling will be needed for the agents participating in the ART Testbed competition. Section 2.3 introduces the two learning algorithms that will be compared in this thesis, agent-based modeling and action-based modeling. Finally, Section 2.4 reviews the past research about trust in agent societies.

Chapter 3 mainly describes the experimental setup that will be used in the simulations and the metrics to be used for comparing different models. Section 3.1 lists the parameter of the games to be simulated and their corresponding values. Section 3.2 explains the four metrics and how they will be utilized to compare the implemented agent versions.

Chapter 4 gives an insight to the architecture of the agents to be implemented. The agent architectures will consist of three parts: Section 4.1 describes how the agent models the world around itself by using the information it gathers form other agents, Section 4.2 tells how the agent determines the number of queries it will send out and to whom it will direct those, and finally Section 4.3 describes how the agent will manage

its responses (i.e., the quality of the response to be generated as well as the decision to respond or not).

Chapter 5 introduces Frost; an agent that models other agents individually. Section 5.1 gives the details of how Frost models other agents in the environment. Section 5.2 describes how Frost decides which queries to be prepared and sent out. Then Section 5.3 introduces the four different versions of Frost, and finally Section 5.4 shows the results of the simulations done using the different versions of Frost.

Chapter 6 introduces Blizzard; an agent that models actions rather than agents. Section 6.1 gives the details of how Blizzards keeps its previous interactions with other agents in the environment. Section 6.2 describes how Blizzard generates the reinforcement values to be associated with previous actions. Section 6.3 decides which queries to be prepared by Blizzard and sent out. Section 6.4 tells how Blizzard will respond to the incoming requests. Then Section 6.5 introduces the four different versions of Blizzard, and finally Section 6.6 shows the results of the simulations done using the different versions of Blizzard.

Chapter 7 compares the four versions of Frost and Blizzard with each other (i.e., base version of Frost vs. base version of Blizzard). It shows the results of each experiment with the related metrics and observations.

Finally, Chapter 8 concludes the thesis by reviewing the results of our experiments, showing our observations, and explaining the possible ways that we can improve the agents for better performance.

2. BACKGROUND

The algorithms and methods proposed in this thesis are implemented in the Agent Reputation and Trust (ART) Testbed software, which is also a simulation platform for making test runs in order to compare different strategies. The ART Testbed provides the researchers with an environment to model and run their agents to collect statistics about how successful their strategies are. Section 2.1 explains the rules of the ART game [9], the characteristics of the simulation environment [8], and finally the agent model to be implemented. Sections 2.2 and 2.3 review the concepts of modeling and learning; respectively. Section 2.4 reviews the relevant literature about trust that was inspiring for our research and discusses the ways how the ideas can be integrated into the ART Testbed environment.

2.1. ART Testbed

2.1.1. The Game Overview

The game acts as a business environment where customers come to buy opinions about paintings. Each agent participating in the game is a service provider (i.e., an appraiser) that is selling its opinion about a painting when requested. For each painting, the era that it belongs to is known. Each service provider starts the game with a predefined expertise value for all painting eras.

When the simulation starts, each provider is assigned a number of clients, who have paintings belonging to different eras and who are interested in receiving an evaluation about these paintings. The agent can use its own expertise as well as contact other service providers and use their expertise to form an evaluation about a painting. After submitting this final opinion, the true value of the painting is revealed. This allows the service provider to compare its opinion with the true painting value. The level of accuracy of an opinion is determined by finding the difference between the appraised and true values. This is called the appraisal error and with this feedback, the agent

has a means to evaluate other agents in the game. For example, if the quality of the received opinion is good (i.e., the appraisal error is low), the provider of that opinion is more likely to be an expert in the era that the painting belongs to.

The service providers also earn a certain amount of money for every answer they provide to the requesters (both from clients and other service providers). The more accurate answers a service provider generates, the more likely the service consumers buy opinions from that provider. The aim of the game is to end with the maximum bank balance. Throughout the simulation, there are no guarantees about the correctness of replies. That is, a service provider may provide wrong information about a painting 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. The complete execution flow of the simulation is described in Section 2.1.2 with examples.

There are a number of parameters to be set in the ART game. The game may exhibit different behaviors depending on how these parameters are set at the beginning. The public parameters of the game are revealed to all agents that participate throughout the simulation. The static parameters are: (1) *Client Fee* (the money earned for evaluating a painting), (2) *Opinion Cost* (the cost to request an opinion from another agent), (3) *Reputation Cost* (the cost to request a reputation information from another agent), and (4) *Expertise Values* (agent's own expertise values for all painting eras). The agent also has access to some of the dynamic variables of the simulation such as (5) *Bank Balance* (agent's current money) and (6) *Current Timestep* (the round that the game is currently simulating is revealed although the total number of timesteps is not known by the agent).

Table 2.1 summarizes the parameters used in the game together with their descriptions. Some of these parameters are more important than others in a competition setting where each individual agent tries to win the game by reaching the maximum bank balance. For example, the ratio of the *Client Fee* to the *Opinion Cost* determines the main source of income for an agent. If this ratio is high, the agent may earn more

Table 2.1. ART Game Parameters

Parameter Name	Description
Number of Agents	The number of agents that participate in the game
Average Clients per Agent	The number of clients an agent will have at the beginning, this number may vary among the agents, but the total number of clients in the game is kept the same
Timesteps per Session	The number of rounds that the game will run for
Number of Sessions	The number of times a game is repeated with the same parameters
Client Fee	The fee that the agent receives by evaluating a painting for the client
Opinion Cost	The cost of gathering the opinion of another agent
Reputation Cost	The cost of gathering reputation information about another agent
Expertise Values	The starting expertise values of the agents remain the same throughout the simulation
Bank Balance	The total bank account of the agent
Current Timestep	The round that the simulation is currently running

by evaluating its painting assignments better and it does not have to respond to the incoming opinion queries since the fee it receives will be low. But, if the cost of gathering opinions is high enough to compete with the fee of evaluating a painting for a client, an agent may choose to sell its opinions for the eras in which it is an expert.

2.1.2. The Simulation Environment

The game architecture consists of a game simulation engine, a game server and a database. The game simulation engine runs the game by managing the communication of agents through various types of messages and making the necessary computations (i.e., final appraisals). The database is used to record game specific data for later viewing of the statistics of a previously completed game.

The game can be thought as a series of discrete time steps controlled by the simulation engine. The game parameters are set at the beginning of each simulation run and some of them are updated at each time step. Some parameters like the expertise values of appraisers for different eras are assigned randomly by the simulator and most of the other parameters have predefined values which remain the same throughout the game.

There are several actions that take place every time step. First, the client shares are determined for each time step and clients are assigned to the appraisers accordingly. The clients are willing to get their paintings evaluated by our intelligent agents (i.e., service providers). After the paintings are assigned, the agents try to evaluate their own assignments by either generating opinions themselves or by purchasing others' opinions with associated costs. Every inter-agent communication is based on a sequence of message transfers. The opinion and reputation transactions are handled by the simulator as a sequence of incoming and outgoing messages. Each type of transaction, together with the message types associated is given in detail in Section 2.1.3. After the transactions are complete, the simulator collects the data from each agent (opinion values, weights, messages, and so on) and makes the necessary final computations. At the end of the time step, it reveals the actual painting values to the agents as feedback.

The following examples describe the execution flow of the simulation for different sections of the game:

Example 1 *Client shares:* Let agents A and B participate in the same environment. At the beginning of the simulation, they are assigned the same number of paintings to evaluate. Now let us suppose that agent A has evaluated its paintings better than agent B. In other words, agent A made fewer errors in its appraisals than agent B. In the following round, agent A will have more clients, thus more paintings to evaluate, than agent B. ▮

Example 2 *Reputation transactions:* Let agents A, B, and C participate in the same game. If agent A is assigned a painting that belongs to era 1 but it does not have enough expertise to evaluate it itself, it may need to find others to get opinions from. Now let us suppose agent A has previously interacted with agent B and knows that it is trustworthy. In addition to that, it knows that agent B is not an expert in era 1. In that case, it may query agent B about agent C to find out how trustworthy agent C is in era 1. According to agent B's strategy, agent B may respond to the reputation query or not. ▮

Example 3 *Opinion transactions:* Let agent A have low expertise in era 1, and it is assigned a painting to evaluate that belongs to era 1. Since its expertise does not allow it to generate a good opinion about the painting's true value, it has to buy opinions from other agents in the environment. Now, let us suppose it has selected agents B and C to get opinions from. It first queries these agents to find out how well they think they can evaluate the painting. That is, it receives certainty values from these agents telling how expert they are in era 1. After agent A evaluates the certainty values, it decides to confirm the opinion of agent B, and reject that of agent C. Even if the opinion transaction is confirmed by agent A, agent B may not send any opinions at all. For that case, agent A decides to generate an opinion itself, too. Now that it has two opinions about the painting, it has to assign weights to those opinions and

send them to the simulator to initiate the final calculations. At the end of the round, the simulator calculates the agent's final evaluation about the painting by getting a weighted average of the opinions. ▌

Example 4 *Opinion evaluations:* Assume that agent A has evaluated a painting belonging to era 1 in the previous round by taking the opinion of agent B. Since the appraised values and the true painting values are revealed at the beginnning of the next round, agent A now has a means of evaluating agent B's performance in era 1. Now, let us suppose the appraisal error for the opinion that agent B has generated is very low. Following this finding, agent A may record agent B as an expert in era 1. ▌

2.1.3. ART Agent Model

The agent model in the ART Testbed has been fixed in terms of the operations that will be performed. Each agent's model is created by extending an abstract Agent class and filling up the methods that describe the agent's behavior. The following list gives the order of interactions of each agent for each timestep. This sequence of actions is enforced by the simulation engine.

1. *prepareReputationRequests()*: The agent determines which agents to request reputation information about other agents in this method and sends messages of type *ReputationRequestMsg* for each request to the simulator. The agent may use the messages of type *OpinionReplyMsg* from the previous time step as additional information to evaluate the opinions it receives from other agents.
2. *prepareReputationAcceptsandDeclines()*: The agent determines whether to respond to reputation requests for that round. It first empties its inbox of messages of type *ReputationRequestMsg* and sends an accept or decline message for each of them.
3. *prepareReputationReplies()*: According to the accept or decline messages, the agent generates reply messages for the agents that request reputation information.

4. *prepareOpinionRequests()*: Like the reputation transaction case, the agent determines which agents to ask for opinions about the paintings it has to evaluate in this method. It may use the reputation information gathered from the previous method.
5. *prepareOpinionCreationOrders()*: For each opinion request and the agent's own appraisal assignments, the agent has to order an opinion value from the simulator via sending a message of type *OpinionOrderMsg*.
6. *prepareOpinionCertainties()*: After collecting the messages of type *OpinionRequestMsg* from its inbox, the agent prepares certainties of its opinions if it wants to respond to the requesting agent.
7. *prepareOpinionRequestConfirmations()*: According to the certainty values that the responding agent sends, the requesting agent sends a confirmation message if it really wants to purchase its opinion about the already made request.
8. *prepareOpinionProviderWeights()*: For each confirmed opinion request, the agent sends the corresponding weight of that opinion to the simulator as a message of type *WeightMsg* (as the simulator has both the opinion values and the weights, it can make the final calculations).
9. *prepareOpinionReplies()*: According to the confirmation messages, the agent creates messages of type *OpinionReplyMsg* by finding the appropriate opinions that are already sent to the simulator.

2.2. Modeling

In a multiagent competition environment, modeling others is a crucial element to successs. Since the agents participating in the ART Testbed have limited capabilities to complete all of their tasks, they have to receive help from each other. For example, when an agent cannot evaluate a painting, it will ask others. If the agent has modeled others perfectly, it can ask the expert in that era. The quality of the help they receive will increase with how well they model the environment around them. Also the agent has to know its own capabilities to decide on which situations it will need others' help. The best combination of such a strategy will lead the agent to success in ART game.

2.2.1. Modeling Self

For an agent competing in the ART Testbed, modeling self is not a hard problem. Since the simulation engine allows the agents to know their exact expertise values on all the eras, the agent has a means of building trust on itself precisely. If it is the case that the agent is also unaware of the capabilities it has, the situation will be more complicated. In order to model itself, the agent will also need to go through its own history of actions (like it does in order to model other agents in the environment). But this time the information it receives will be more trustworthy.

2.2.2. Modeling Others

This is the most complicated part to be implemented in the agent's architecture. Since the agents are autonomous and heterogeneous in an open environment, it is nearly impossible to predict their future behavior [10]. But the agent may have a means to model the behavior of others via processing the information from their past interactions. In the ART Testbed, there are several ways to get clues about the behavior of an agent. Every opinion the agent provides has an accuracy which will be revealed to the querying agent after the round ends. By processing this accuracy information, the agent can model the other agent's expertise and keep it updated when more recent findings are gathered. But since the game is a competition environment, the agent does not always have to behave as expected. That is where the complication starts for the modeling agent. It also has to adapt to the changes that can occur in other agents' behavior.

2.3. Learning

In order to model its environment better and keep it updated, the agent has to have some sort of learning mechanism it can use to update its models of others with recent data. In this thesis, we will be considering two such learning methods, agent-based modeling and action-based modeling. Next, we will briefly explain what these methods are. The details are given separately when we are talking about the agent

architectures.

2.3.1. Agent-Based Modeling

In the agent-based modeling case, the agent tries to build update multipliers according to some built-in heuristics. The agent then learns its environment (i.e., the expertise of other agents) iteratively by applying update operations based on the pre-built multipliers. These multipliers are calculated via comparing the accuracy of opinion providers (i.e., how well they generated opinions) to the actual values of the paintings. A similar approach may be utilised to the reputation transactions case in which the agent tries to model the sociability of others in the environment (i.e., how well they know others). That way, the agent tries to model every other agent in its society.

The agent-based modeling agents utilize both the individual dimension of trust (i.e., taking opinions from others and evaluating the results) and the social dimension (i.e., gathering reputation information from the society). After some time passes in the simulation, they begin to prepare reputation queries and update their models according to the reputation information gained. Each interaction with another agent (either opinion or reputation query) will give out some useful information about the ability of that agent. By updating the current information it has about that agent with the recent findings, the agent learns its environment iteratively. After it has learned the expertise of others in the environment, it can prepare opinion queries via sorting the agents according to their expertise values in the eras that the agent's assigned paintings belong to. This way the agent can have a means of deciding which agents to direct its queries to.

2.3.2. Action-Based Modeling

The action-based modeling agent utilizes reinforcement learning to model its environment. In its simplest definition, reinforcement learning can be thought as learning through rewards and punishment. Each action the agent takes has an associated re-

inforcement value. Maximizing this overall reinforcement value will lead the agent to better understanding of its environment and help it succeed in the game.

The idea of reinforcement learning can be integrated into the agents in the ART Testbed domain in the following sense. Agents in the ART Testbed involve in certain transactions like requesting opinions from others or creating opinions for others on demand. Each action leads to a consequence (i.e., the agent receives a bad answer and loses client share in the next round) and these consequences can be ranked (i.e., given reinforcement values) by taking into account different parameters of the game. For example, an action to request an opinion from another agent has a positive reinforcement value (i.e., a reward) if the received appraisal is accurate enough. Of course, the accuracy level of the answer will determine the actual reinforcement value. Another type of transaction in the game is the sharing of reputation values which can be thought of as advice coming from others. By collecting reputation information about others in the environment, the agent has a means of finding more promising agents and directing its requests to those agents only. This is a superiority over the individual dimension of trust, because the agent has a clue of selecting the action that will lead to rewards rather than punishment. The quality of the reputation information can be used as rewards or punishment. The complete list of actions and how they are assigned reinforcement values will be explained in detail when introducing the architecture of the action-based modeling agent.

2.4. Utilizing Trust in Agent Societies

In this thesis, we have designed two types of agents that differ mainly on the learning algorithms that they use in modeling others. They are compared in terms of how well they model others, and how fast they build their models. While constructing the learning algorithms proposed, we have inspected existing trust models, determining their compatibility with the ART testbed domain.

According to the FIRE trust model [11], trust arises from two levels, the individual (direct experiences from past interactions) level and the society (observations

by the society of agents past behavior, reputation) level. Since the agents are self-interested and unreliable, and there is no central authority or contol 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 [5] 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 thesis, we have preferred in our models 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 [4]. 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 cannot always refer to this information while preparing its queries. Consider the following example:

Let agent A be in a competition environment with two other agents B and C, and let there be three areas of expertise 1, 2, and 3. Agent A has previously learned that agent B is an expert in areas 1, and 2, but poor at area 3. So the overall reputation of agent B is good. On the other hand, it has modeled agent C to be poor at areas 1, and 2, but to be expert in area 3. Agent C's overall reputation is average in that case. Now, when agent A receives a task to be done from area 3, the two approaches differ in finding the right agent. When agent A tries to choose according to the overall reputation of the agents, it has to select agent B, but it is clear that agent C is more capable of doing that specific task. The situation in ART Testbed is similar since the agents vary in their expertise values for different eras. So for the ART Testbed environment, we have chosen to model the reputation of agents separately by era.

Reinforcement learning is very much 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 [12]. 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.

Trust in a competition environment like ART Testbed is mainly built upon observation and opponent modeling as studied by a recent research [13]. Since the information an agent has about its opponents is limited to the interactions it has previously (i.e., either opinion transactions or reputation transactions), it may use an argumentation based decision making to decide on its future interactions. It may be studied in three steps; in the first step the agent searches through its transaction database to recall its previous actions about its opponent, the second step tries to model the opponent by performing rule induction on the recovered information from the database (i.e., perform predefined rules on the knowledge base). At the last step, the agent feeds this information with the possible actions it may take and the goals it has. The argumentation protocol then selects the best action for the agent to take. The opponent model also tries to predict the consequences of the action regarding the whole society the agent resides.

The existence of reciprocal agents in a multiagent system brings some well-known challenges into the competition environment like zero-cost identity, free-riding, and collusion [14]. Zero-cost identity is a common problem in open multiagent systems where an agent may enter or leave the system at any time of the execution with no cost. This flexibility of the system allows harmful agents to continue their activities without revealing their identity. Since they may rejoin the system at any time with

different identities, it is nearly impossible to identify them. However, ART Testbed execution environment is a partially-open system. That is, the agents that enter the system are not known upfront. But, when they do enter at the beginning, they cannot leave and enter again. Thus, they hold their identity throughout the execution of the simulation. So, zero-cost identity does not bring about a huge problem.

Free-riding is another issue related to reciprocal resource sharing environments. In its simplest definition, free-riding is the selfishness of an agent to use the resources in the environment without sharing any itself. If we try to find similarities between a resource sharing environment and ART Testbed environment, we can think of the opinion and reputation informations as the shared resources in the environment. However, since they have costs associated with them, free-riding is out of the context for the agents in ART Testbed.

The last and the most important challenge in multiagent systems is collusion among the agents. When there is no central authority in the system (i.e., no control over the agents), a group of agents may act together to perform collusion. There are two ways to realize it; the first one is to exaggerate the expertise of a group member (this is relatively the harmless approach), and the second one is to distort the bahavior of a good agent in the environment (this is the harmful approach since the group is trying to show that a non-group member is poor in expertise). Both approaches may be applicable in the ART Testbed domain when the number of participants increase. The following two examples describe how agents may collude in the ART Testbed:

Let 10 agents participate in the same competition with five painting eras and five of them decide to form a group. Now, in the setting that ART Testbed allows, they may collude through reputation transactions. So, if the intention of the collusion group is harmless, they may exaggerate the reputation of a single agent in the group every time a query comes about it. When a query is directed about other agents in the group, they have to lower the reputation values they give out in order to make that agent the most reputable one. In this setting, only one agent benefits in the group. Taking this to one step further, they may choose to promote one agent for each era

so that every agent in the group has a chance to benefit. Apart from the reputation transactions, they may also force the non-group members to buy opinions from a single agent in the group by not responding to the opinion requests themselves and letting that agent to reply to the opinion queries.

Let the same setting exist as in the previous example, but this time suppose that the group is a more harmful coalition. They first have to identify a target opponent from the non-group members, preferably an expert one. Once they select the target agent, they try to demote it (i.e., telling that its performance is poor) by their reputation responses.

Even if the groups in the above two examples try to collude by the reputation transactions approach, they may not be successful in societies where the majority of the agents model their environment via the individual dimension of trust (i.e., by evaluating the opinion transactions). So, we can say that ART Testbed environment has some sort of built-in resistance to collusion.

Collaboration in multiagent systems is another interesting concept as experimented in the SPIRE system [15]. 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 en-

couraged 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).

3. EXPERIMENTAL SETUP & METRICS

The simulations have been conducted using the ART Testbed Environment. First, we will explain certain parameters of the environment and then introduce the metrics that are used for evaluating the agents.

3.1. Simulation Environment

This environment has been used to hold a competition among participating agents in 2006. It can be configured in numerous ways. To establish a common ground with existing usage of the environment, we have configured it to follow the rules of the 2006 ART competition [9]. Further, some parameters are identical to the 2006 Competition parameters except for four parameters. Instead of having 10 painting eras, the experiments are conducted with 5 eras and instead of running the simulations for 60 timesteps, we ran them for 20 timesteps. The main motivation for the changes for these parameters was the computational difficulty. The cost of gathering reputation information is also set to zero in our experiments to encourage the flow of information in the society. Finally, instead of having five agents as in the competition, we put six agents competing against each other. The reason for this is that since we are comparing two types of learning and thus two types of agents, only including equal number of both types would have been fair. We have repeated each simulation five times to remove the probabilistic behavior that may be caused by the distribution of era expertise values to the agents.

Table 3.1 summarizes the values of the parameters as they are used in the experiments.

3.2. 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

Table 3.1. Simulation Parameters

Parameter Name	ART Competition Value	Experiment Value
Number of Agents	5	6
Average Clients per Agent	20	20
Timesteps per Session	60	20
Number of Sessions	1	5
Client Fee	100	100
Opinion Cost	10	10
Reputation Cost	1	0

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 3.1 measures the precision for a particular era j.

$$Precision(j) = \frac{\sum_{i=0}^{numAgents} p_{ij} * e_{ij}}{h_j} \tag{3.1}$$

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.

4. AGENT ARCHITECTURE

The agent's overall strategy is studied in three categories: the modeling strategy of other agents (the environment), the requesting strategy, and the response strategy. The requesting strategy handles which and how many agents to ask for opinions and reputation information, whereas the response strategy deals with whom to give answers and in which way to prepare the answers. It is expected that the response strategy of the agent influences its reputation. That is, if the agent responds to queries correctly about a particular era, it will have a strong reputation in that era. Both the requesting strategies and the response strategies are directly related to and built upon the modeling strategy of the agent. The following sections provide in depth knowledge about each strategy and possible ways to implement them in the agent's algorithm.

4.1. The Modeling Strategy of Agents

An agent's model of its environment is the collection of its models of other agents as well as its observations about the game. The observations about the game are already available to each agent without further probing. This leaves modeling of other agents as the main challenge for modeling strategies.

For the modeling of other agents, the following information is captured:

- Name: The name of the agent, which will be used to address the agent during interactions.
- Expertise: The *expertise* value keeps track of how well the agent answers queries about paintings. Since the actual expertise of agents differs based on eras, the expertise is recorded separately for each era.
- Sociability: The *sociability* of an agent denotes how well the agent answers reputation requests [6]. The higher the sociability, the better the agent knows the reputation of others. Similar to the modeling of expertise, sociability of an agent can vary based on era. Hence, again the sociability of each agent is modeled for

each era separately.

- References: The *references* of an agent is a list of agents that have been consulted to receive reputation information about that agent. The references are important, especially for distributing credit after agent's answers are evaluated.
- Number of Opinions Asked: This value keeps the number of times the agent is consulted for opinion transactions. This information is useful to compute the confidence of information that is gained from the agents. For example, if the number of opinions asked is higher than a certain number, our confidence in the information about the agent may be stronger. It is previously shown that the number of previous interactions with an agent affects the level of trust between agents [16].
- Number of Reputations Asked: This value keeps the number of times the agent is consulted for reputation transactions. This value can be used in various ways. In some cases it may be desirable to get reputation information from agents whose opinions have not been asked before. On the other hand, an agent may prefer to only receive reputation information from those that have been successful several times before.
- Number of Reputations Asked About: This value keeps the number of times reputation information is gathered about the agent. Again, this value may be used in different ways. For example, if we have gathered the reputation of an agent many times before, it may not be necessary to query others about this agent again.

For each previously contacted agent, this information is stored in a complex data structure. As explained above, some information about an agent depends on the era (i.e., expertise and sociability), whereas some values about the agent are kept independent of the era (number of opinions asked, number of reputations asked, and so on).

4.2. The Requesting Strategy

The requesting strategy determines the quantity and quality of queries to be prepared. The term quantity stands for the number of agents to be consulted both for opinion and reputation requests, and the term quality is used to represent the reputation of the agents to request from (from whom to request data from). Two most common requesting strategies are the following:

- Liberal: The agent gathers reputation information about other agents from the environment via preparing reputation request queries. To whom the agent will ask for reputation information and how it will interpret the incoming information is discussed below.
- Conservative: The agent does not get any reputation information from the environment and builds a trust model only on its past experiences with the agents in contact.

Both strategies use the data structures built by the modeling strategy of the agent, but the update process of the data structures for the *Conservative* requesting strategy is much simpler than the *Liberal* requesting strategy. The following paragraph briefly describes the process of how agents are selected to request opinions and reputation information. The details are explained in separate learning strategies.

The opinion transactions are used by both strategies and the procedure of finding agents to get opinions is simpler than the one to prepare reputation queries. In each round of the simulation, each agent checks the paintings assigned to it and makes an ordering of agents according to their modeled expertise values. After the sorting operation is complete, the agent selects the top n agents from the ordered list and directs the opinion request queries to them. According to the strategy chosen, the agent may be allowed to vary the value of n according to its own expertise value in that era. That is, if the agent itself is already an expert in the era in which it needs to evaluate a painting, then it may only ask a few other agents for their opinions. However, if the agent has no idea about the painting, it may choose to ask a large

number of agents for their opinions.

There are two ends to prepare a reputation request: one is to find the set of agents to get information from and the other is to find the set of agents to gather information about. The first question is related to the sociability of the agents. The question can be rephrased as this: Who would know others best to provide reputation information about a particular agent. The second question is more subtle. When does an agent decide that it needs to gather reputation information about a particular other agent?

To answer the first question (i.e., to decide which agents to consult), an ordering of agents is made according to their modeled sociability values. The agents with the top sociability values are asked for reputation information (they are supposed to know the environment best). To answer the second question (i.e., to decide about which agents to gather information from), an ordering based on available agent information is made. That is, the less we know about an agent, the more likely we will need reputation information. Hence, the agent has to figure out how much information it has about other agents. The amount of information we have about an agent can be measured in terms of the number of previous interactions (opinions asked and reputations asked).

4.3. The Response Strategy

The response strategy of the agent decides to whom and in what way the agent responds to the queries directed to it. Two well-known response strategies can be proposed for the agent: a *reciprocity-based* response strategy and a *respond all requesting* type of strategy. In both strategies, also the quality of the responses may vary (i.e., the agent may exaggerate its answer or generate a totally wrong one).

- Respond All: An agent aiming a high reputation value will try to respond as accurately as possible to all requests without any elimination of some sort. After all, if the agent answers more questions correctly, there is a higher chance of being identified as an expert and thus be trusted. To realize this strategy, whenever a query comes in, first the main model of the environment is consulted for each

incoming query, then the response is generated and finally the response is sent according to the information gathered from the model.

- Reciprocity: Note that the game does not offer any explicit advantages to those agents that are considered reputable by others. Hence, there may be times when an agent may not care to have a good reputation. Moreover, when an agent answers queries of others, it is helping other agents earn money since the agent's answer may help others respond to customer requests correctly. A realistic exchange of opinions that take place in real life businesses is that of reciprocity where agents only respond to requests coming from the agents that respond to their requests correctly. This is kind of a reciprocity relation in which both agents are supposed to benefit from, if they are correct in their responses.

A reciprocity-based method will allow the agent to have a high reputation value among the agents that the agent itself requests queries from, but the overall reputation of the agent may be noticeably lower than that of another agent who answers every query directed to it. In order to implement a reciprocity relation, the agent has to keep track of (1) the agents that respond to its queries and (2) how well the agents have answered the queries. Using this data, the agent can decide on which agents to respond back.

5. AGENT-BASED MODELING

The agent-based modeling agent (Frost) models other agents in the system individually, and updates the models incrementally. Frost can be studied in three main parts: modeling of the agent's environment, the query protocol and the response protocol.

5.1. Modeling Strategy

In order to model its environment, Frost uses the data structure explained in the previous chapter. The main part of the model is the estimations about the modeled agent. There are two primary aspects here. The first one is the expertise of the agent, which denotes how well the agent answers queries about paintings (opinion requests). The second one is the sociability of the agent, which denotes how well the agent answers queries about others' reputation (reputation requests). Learning comes into play when these two aspects are considered. We will specifically consider how the agent learns the expertise of another agent.

Frost updates models of others to learn expertise values. The agent builds update multipliers according to some built-in heuristics. The agent thus learns the expertise of other agents iteratively by applying update operations based on the pre-built multipliers. These multipliers are calculated via comparing the accuracy of opinion providers (i.e., how well they generated opinions) to the actual values of the paintings. Hence, the update of agent models takes place at the end of each round, when the actual painting values are revealed to the agents as feedback. Only then, the agent has a means of calculating the error in each appraisal it received from others.

Figure 5.1 shows how Frost updates the expertise values of other agents that it is modeling. In the very beginning of the simulation, all expertise values are initialized to 0.5. To put the algorithm simpler, the agent's modeled expertise is increased when it provides one more correct answer (the value of its answer is above a threshold);

1: initialize the expertise values
2: **for all** $receivedOpinions$ **do**
3: $error = |appraisedValue - trueValue|/trueValue$
4: **if** $error < threshold$ **then**
5: $expertMargin = 1 - expertise$
6: $updateMargin = expertMargin * (threshold - error)$
7: $expertise = expertise * (1 + updateMargin)$
8: **else**
9: $updateMargin = threshold - error$
10: $expertise = expertise * (1 + updateMargin)$
11: **end if**
12: **end for**

Figure 5.1. Updating the Expertise Values

otherwise it is decreased. However, the amount to be increased and decreased are important. In Figure 5.1, 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. Therefore, when an agent provides a wrong answer, its modeled expertise is significantly decreased. The value of *threshold* varies according to the modeled expertise of the agent. For example, if the agent is modeled to be an expert in the era, the threshold value will be low because a good opinion is expected from it.

Example 5 Let the actual value of the painting be $10,000$, the opinion received from the agent be $11,000$, the modeled expertise of the agent be 0.8, and the predefined threshold value be 0.3. Now, the calculated error value is 0.1, and since it is less than the threshold, we have to increase the expertise of the agent by an amount equal to $(1 + ((1 - 0.8) * (0.3 - 0.1))) = 1.04$. So, the agent's new expertise will be $(0.8 * 1.04) = 0.832$. ∎

```
1: initialize the era properties
2: for timeStep = 1 to maxRound do
3:    for era = 1 to maxEra do
4:       sort the modeled expertise of agents
5:       query top n agents according to their modeled expertise
6:       if expertiseofAgent > thresholdExpertise then
7:          ask the agent for its opinion
8:       end if
9:    end for
10: end for
```

Figure 5.2. Finding the Experts

Example 6 Let the actual value of the painting be $10,000$, the opinion received from the agent be $6,500$, and the predefined threshold value be 0.3. Now, the calculated error value is 0.35, and since it is greater than the threshold this time, we have to decrease the expertise of the agent. The multiplier to be used is $(1 + (0.3 - 0.35)) = 0.95$. So, if the agent's previous expertise is 0.8, the new expertise will be $(0.8 * 0.95) = 0.76$. ▮

5.2. 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. After it has a means of whom to direct the queries to, it can prepare its opinion requests.

Figure 5.2 shows how Frost decides whom to ask for opinions. First, the properties for each era are set according to the agent's own expertise distribution (line 1). This includes setting the maximum number of agents to query and calculating the threshold value for the expertise of agents to be queried. For example, if Frost's own expertise is high in a given era, it may ask for fewer opinions and set the threshold value slightly higher than its own expertise to get better opinions than it can generate. After the era properties are set for each round, Frost orders the agents based on their models and

selects the top n agents matching the criteria defined in the era properties. The value of n may change according to different implementations of Frost. When the round is over, it updates its model with the recent findings.

Example 7 Let the agent need to buy an opinion about a painting that belongs to era 1. It has previously modeled the expertise of agents A, B, and C to be 0.7, 0.4, and 0.9; respectively. Now let us suppose the agent has set the value of expertise threshold to be 0.7 and the maximum number of agents to 2 for era 1. So, even if agent A is the second one in the ordering according to the modeled expertise values, it is not asked for its opinion since its expertise does not pass the required threshold value. ▌

We will explain in detail how requesting and response strategies can be varied when we are inspecting the different implementations of Frost.

5.3. Implementations of Frost

In the following subsections, we will inspect four versions of Frost varying in their trust strategies. Both implementations use the same learning algorithm to model their environments and they use the *Liberal* type of requesting strategy to manage reputation queries as told in Section 4.2, but the first two differ in their requesting strategies and the last two differ in the way they respond to others.

5.3.1. Frost-v1

In its basic form, Frost-v1 answers all requests that it receives as accurately as possible. The difference it has from other versions is that it always requests two opinions per painting it has to evaluate even if it is also an expert in that era. That is, the value of n in Figure 5.2 is always two for all assignments. This is a naive agent that does not adjust its actions based on the environment.

5.3.2. Frost-v2

Frost-v2 also answers all requests that it receives as accurately as possible. The difference from Frost-v1 is again in the requesting strategy. Frost-v2 has the flexibility to decide on how many queries to generate per painting. It does so according to the algorithm in Figure 5.2 and selects the number n by taking into consideration its own expertise in the era that the painting belongs to. For example, if Frost-v2's expertise is high in an era, it will probably choose to get only one more opinion from another agent, since a second opinion will have an extra cost and possibly it will not be as good as the first one. On the other hand, if its expertise is low or it has not previously identified an expert in the era, it may need more than two opinions to justify its evaluation.

5.3.3. Frost-v3

Frost-v3 uses the same requesting strategy as Frost-v2 does. But since ART Testbed is a competition environment, the agent sometimes 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-v3 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). We call this a consistent cheating behavior. On the other hand, it tries to be as accurate as possible when it has enough expertise to keep its clients satisfied.

5.3.4. Frost-v4

Keeping the cheating behavior from the previous version, Frost-v4 also uses a reciprocity-based response strategy. There are two motivations for this. One, the testbed does not offer explicit advantages to more reputable agents. Hence, there is no immediate benefit for being reputable. Two, by providing useful information to others, the agent would help others to earn money, which is unintuitive in a competition

setting. For these reasons, Frost-v4 only responds to requests coming from the agents that respond to its requests correctly. Frost responds to all incoming requests until the third round, then it switches to reciprocal behavior. This kind of a reciprocal relation [14] may lead each participating agent to success in their relations. Sen *et al.* [17] showed that a group of agents making a community and helping each other will lead to the least error in their appraisals. While implementing the reciprocity relations in Frost-v4, a supplementary data structure is built for (1) keeping track of the agents that respond to its queries and (2) determining how well the agents have answered the queries. By processing this information, the agent can determine which agents to send responses. Frost-v4 is the agent that is ranked third in the 2006 ART competition [18].

5.4. Simulations

Here, we compare the four implementations of Frost. In each simulation, a version of Frost is compared with the next version, and in the last simulation Frost-v4 is experimented with different initial expertise values.

5.4.1. Frost-v1 vs. Frost-v2

Figure 5.3 plots the four metrics, comparing Frost-v1 with Frost-v2.

Average Appraisal Error: Beginning with the appraisal error, both agents show an improving behavior in evaluating their paintings. But Frost-v2 makes slightly fewer errors. This is because when necessary, Frost-v2 consults more than two agents (or when it is enough, it consults less than two agents) whereas Frost-v1 always consults two others.

Opinions Purchased: It is easily seen from the figures that Frost-v2 lowers its opinion requests after the first quarter of the simulation, while Frost-v1 keeps asking the same amount of others every round. The oscillations are caused by the change in

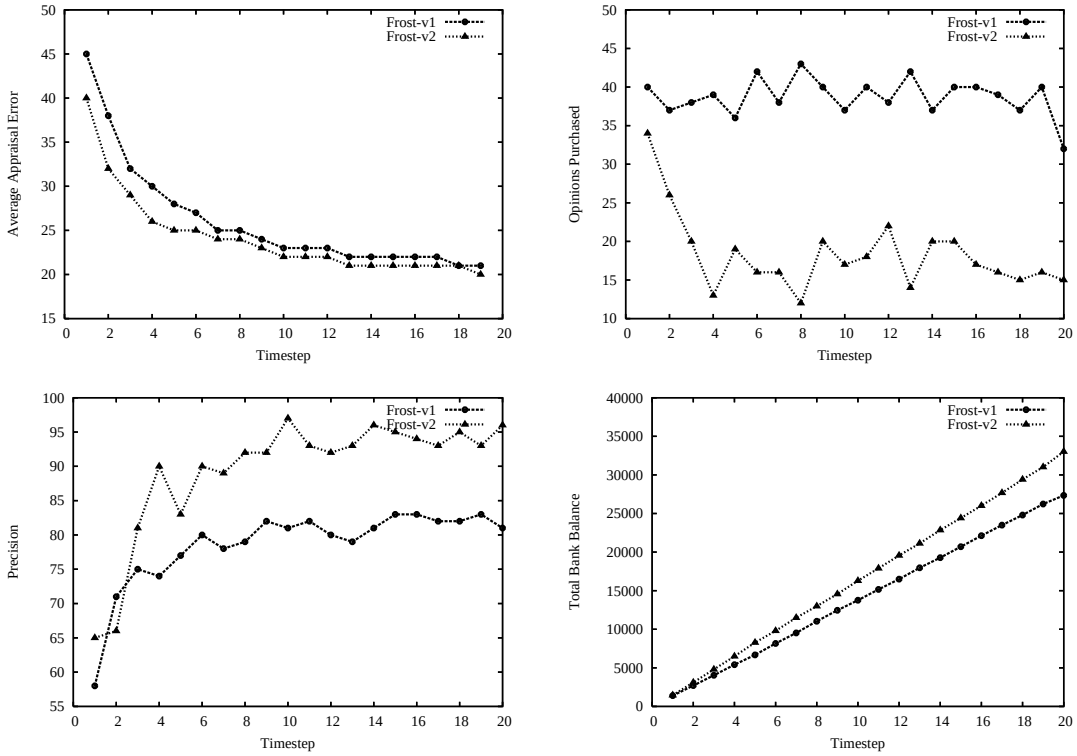


Figure 5.3. Frost-v1 vs. Frost-v2

client shares (i.e., when fewer paintings are assigned, fewer opinions are purchased).

Precision: Since Frost-v2 has a flexibility to choose how many opinions to purchase about a painting, it can reach a better precision with less help from others in the environment. The first quarter is the period where both agents almost complete modeling their environment, but since Frost-v1 has a built-in behavior to request a fixed number of opinions every round, its precision stays lower (i.e., it also asks agents with below average expertise to complete its total request size).

Total Bank Balance: In terms of bank balance, Frost-v2 earns significantly more than Frost-v1 does. This is mostly because Frost-v2 buys fewer opinions and hence spends less money while it makes fewer errors than Frost-v1.

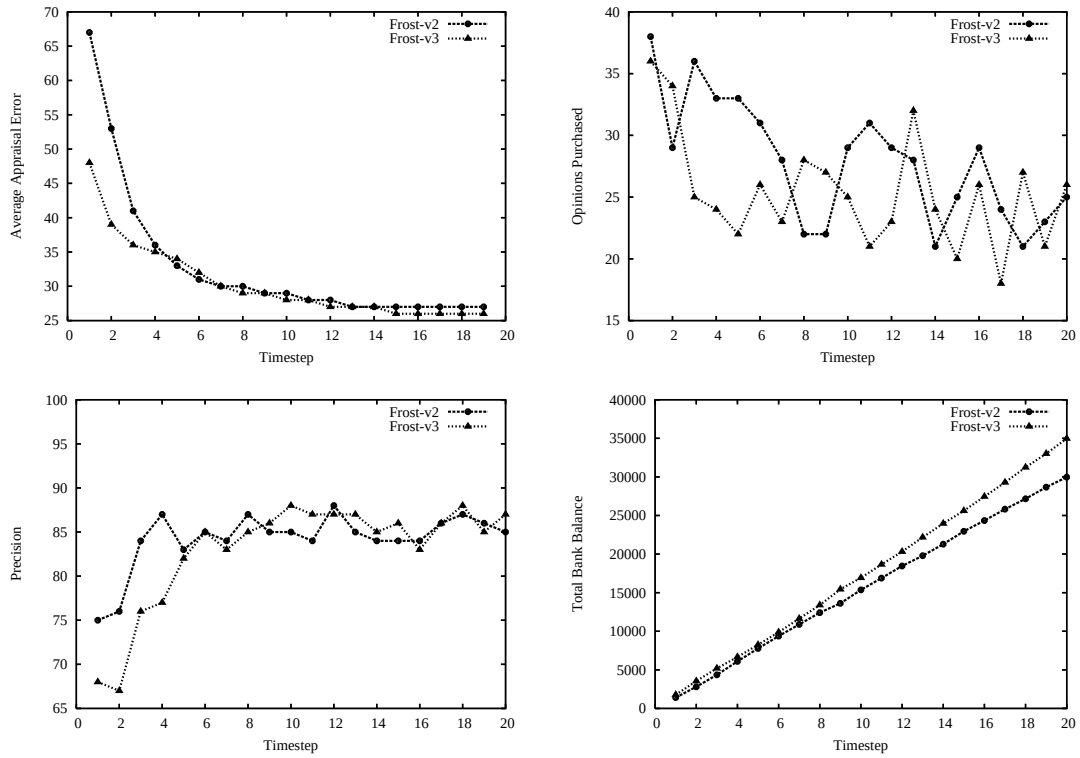


Figure 5.4. Frost-v2 vs. Frost-v3

5.4.2. Frost-v2 vs. Frost-v3

Figure 5.4 plots the four metrics, comparing Frost-v2 with Frost-v3.

Average Appraisal Error: When cheating is injected into the environment, neither agents seem to be affected much. Since the cheating behavior of Frost-v3 is consistent (i.e., generate the worst opinion when not able to generate a precise one), it is not hard for the agents to adapt to that. Further, they use it to their advantage as they can eliminate the agents with below average expertise. When we look at the appraisal errors, the cheating agent takes advantage of the first few rounds while Frost-v2 still tries to adapt to the cheating behavior.

Opinions Purchased: Another interesting behavior is found in the agents' requesting strategies. The unstability in the number of opinions purchased is caused by the change in distribution of client shares (i.e., when client share is high in one round, the number of opinions is also high, then in the next round client share is higher for the

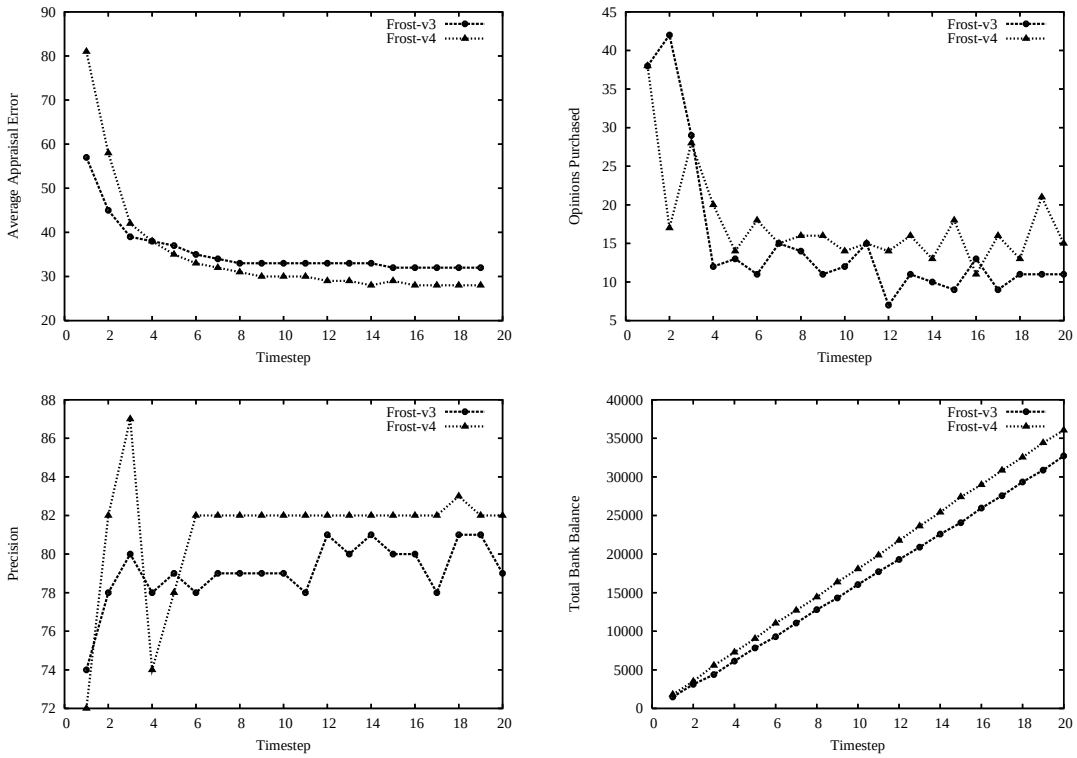


Figure 5.5. Frost-v3 vs. Frost-v4

other agent increasing its purchase of opinions that round).

Precision: In terms of precision, Frost-v2 seems more stable whereas Frost-v3 exhibits an increasing trend. But both agents stay almost at the same levels of precision while modeling their environment.

Total Bank Balance: Since Frost-v3 makes slightly fewer errors than Frost-v2 and it buys noticably fewer opinions, Frost-v3 begins to earn more than Frost-v2 after the first half of the simulation.

5.4.3. Frost-v3 vs. Frost-v4

Figure 5.5 plots the four metrics, comparing Frost-v3 with Frost-v4.

Average Appraisal Error: This experiment shows that the response strategy of the agent has the most positive effect on success in a competition environment. In the

first quarter of the simulation, the appraisal errors of both agents seem similar, but then Frost-v4 continues to lower its error while Frost-v3 cannot improve the quality of its appraisals further. This is mainly because Frost-v3 is not able to buy as much opinions as it needs from other experts anymore.

Opinions Purchased: Both agents purchase fewer opinions compared to the previous experiments since they cannot get the precise opinions they expect from the experts in the environment anymore.

Precision: Again after the third round when Frost-v4 applies reciprocity on responses, Frost-v3 cannot improve its precision further. Although the precision of Frost-v4 also drops for a while, it is able to redirect its queries to other experts and keep its precision slightly higher even after the first quarter. The main reason for this kind of behavior is that Frost-v3 does not show adaptivity to not responding agents as Frost-v4 does. It keeps on asking the same agents and still get no answer because of the reciprocal behavior.

Total Bank Balance: It is clear from the precision metric that Frost-v4 models the experts around it better than Frost-v3 does. Its lower appraisal error is also a proof of it. Although the number of opinions Frost-v4 buys seem greater than Frost-v3, that's because it has more paintings assigned to it. As a result of these, Frost-v4 earns significantly more than Frost-v3 and wins the game.

5.4.4. Frost-v4 with Varying Expertise

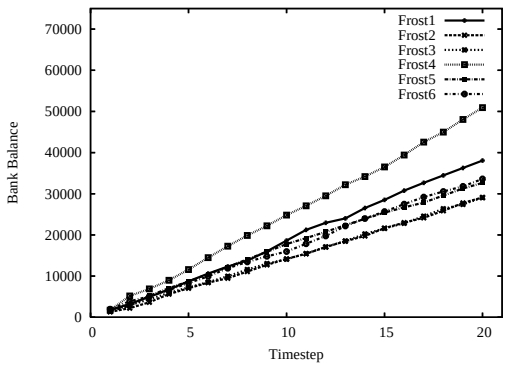
Figure 5.6 plots the three metrics, comparing Frost-v4 with varying initial expertise values. Here since we are comparing the reciprocal versions of Frost among themselves, the only difference they have is their initial expertise distributions. So Table 5.6(a) shows the expertise values of each agent for every painting era. Table 5.6(b) shows the separate appraisal errors each agent makes when evaluating the paintings for each era. Since the agents are very strict in their responses (i.e., do not sell their

Agent	Expertise				
	Era 1	Era 2	Era 3	Era 4	Era 5
Frost1	0.8	1.0	0.8	0.4	0.2
Frost2	0.7	0.3	0.7	0.6	0.4
Frost3	0.3	0.3	0.9	1.0	0.1
Frost4	0.6	1.0	0.9	0.9	0.7
Frost5	0.4	0.7	0.1	1.0	0.7
Frost6	0.3	0.9	0.3	0.5	1.0

(a) Initial Expertise Distribution

Agent	Appraisal Error				
	Era 1	Era 2	Era 3	Era 4	Era 5
Frost1	29.1	9.3	25.1	52.6	92.0
Frost2	28.5	76.0	42.8	40.2	70.6
Frost3	78.8	66.7	22.7	11.6	104.5
Frost4	46.4	12.0	18.1	21.6	37.4
Frost5	50.9	32.9	88.4	11.5	39.7
Frost6	67.1	23.3	69.0	53.7	11.8

(b) Appraisal Error



(c) Total Bank Balance

Figure 5.6. Frost-v4 with Varying Expertise

opinions), the number of opinion transactions will be low in the society as we have seen in the previous experiment. Following that finding, we believe that the appraisal errors are highly correlated with the initial expertise values of the agents. Figure 5.6(c) shows the bank balance of the agents for increasing timesteps. Since the aim of this experiment is to see the effect of initial expertise distributions on agents' performance, we make a single run of the simulation.

Expertise vs. Appraisal Error: Since the tables for expertise values and appraisal errors are expected to be highly correlated, we evaluate them together. As expected, we can see from the tables that when an agent has a high expertise value for an era, its error value is less for that era. For example, the expertise values of Frost1 and Frost4 are both 1.0 for era 2 as shown in Table 5.6(a), and their appraisal errors for era 2 are the least among other agents which can be seen from Table 5.6(b). This high level of correlation is caused by the reciprocal behavior in the society, because the agents cannot find other service providers to get opinions from and they only rely on their own opinions when evaluating the paintings.

Bank Balance: Following the finding from the tables, we can safely assume that the agent with the highest expertise values and thus the lowest appraisal errors will have the maximum bank balnce in the end. Actually, this is the case for this experiment and Frost4 wins the game since it has a relatively high expertise value for almost each painting era.

6. ACTION-BASED MODELING

The second type of modeling we consider is the action-based modeling. Blizzard models its environment based on actions. This differs from Frost mainly in terms of how it builds and updates models of others. Rather than keeping separate information about each agent it interacts with, Blizzard evaluates the quality of its previous actions based on 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 others.

Typically, agents that employ reinforcement learning learn their environment through trial and error interactions [19]. 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 [20]. The variation of the Q-learning embedded in the agent's architecture enables Blizzard to build reinforcement values, called Q-values, for each action it takes. A Q-value function maps the state-action pairs of the agent to the corresponding Q-values [21]. Since we assume the agent in the ART Testbed to be stateless (i.e., taking an action does not bring the agent to a new state), the Q-value function utilized by Blizzard only maps its actions to the reinforcement values. The details of the reinforcement value generating function will be considered in Section 6.2.

6.1. 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 like in the opinion request case. 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

```
1: for all responseActionsTakenPreviousRound do
2:    ratio = requestsForThatEra/allRequests
3:    if ratio > threshold then
4:       difference = ratio − threshold
5:       assign the reward according to difference
6:    else
7:       difference = threshold − ratio
8:       assign the punishment according to difference
9:    end if
10: end for
```

Figure 6.1. Generation of Reinforcement Values for Response Actions

percentage of requests belonging 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.

The action-based modeling agent, Blizzard, also divides its strategy into parts. Although it does not store a complete data structure for modeling its environment, it keeps the history of actions it has done in the past. It then builds the requesting and response strategies on top of the recorded actions and their corresponding reinforcement values.

6.2. Generation of Reinforcement Values

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). Figure 6.1 shows how the agent's response actions are related with reinforcement values:

According to the algorithm in Figure 6.1, the agent tries to evaluate its past

response actions by calculating the rate at which they supply more incoming requests. In order to do so, it calculates the ratio of the incoming requests to all requests, and if that ratio is higher than the threshold value, a reward is assigned to the response action. Otherwise, the reinforcement value is a punishment. Here, the predefined threshold value is set to 0.1, that is if the percentage of received opinion requests for that era exceeds ten percent, rewards are assigned for the opinion response actions belonging to that era. The amount of the reward is increased by how high the ratio is from the threshold value. In other words, higher rewards are assigned if the ratio is large.

Example 8 Let us suppose that the agent has responded to an incoming opinion query about a painting that belongs to era 1. Now, in the next round the agent checks the incoming opinion requests and finds out that a total of 40 requests have been received in which 12 of them belongs to era 1. So, the ratio is $(12/40) = 0.3$, which is significantly higher than the threshold value (0.1). According to the algorithm in Figure 6.1, the opinion response action is assigned a reward. ▮

Next, we will look at how reinforcement values are generated for the agent's opinion request actions:

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 (Figure 6.2). If the appraisal error is less than the overall error for the era, 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 9 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,

```
1: for all opinionRequestActionsTakenPreviousRound do
2:    opErr = findAppraisalErrorForThatOpinion()
3:    eraErr = findAppraisalErrorForThatEra()
4:    difference = opError − eraError
5:    if difference < 0 then
6:       if difference < thresholdReward then
7:          reinforcement = maxReward
8:       else
9:          reinforcement = maxReward ∗ difference/thresholdReward
10:      end if
11:   else
12:      if difference > thresholdPunishment then
13:         reinforcement = maxPunishment
14:      else
15:         reinforcement = maxPunishment ∗ difference/thresholdPunishment
16:      end if
17:   end if
18: end for
```

Figure 6.2. Generation of Reinforcement Values for Opinion Request Actions

```
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
```

Figure 6.3. Preparing Opinion Request Actions

the action will be assigned a reward, and the amount will be $(10 * (-0.25/-0.5)) = 5$. ∎

Example 10 Let the received opinion has 0.6 error value associated with it and the average error for that era be 0.2. Since the received error is greater than the average value, the action will be assigned a punishment, and the amount will be the maximum punishment value -10 since the difference 0.4 is greater than the threshold value. ∎

6.3. Requesting Strategy

At each round, Blizzard tries to predict the experts in the environment for its assigned paintings and queries them to get their opinions. Figure 6.3 shows how Blizzard decides whom to ask for opinions.

According to the algorithm in Figure 6.3, the agent first checks its assigned paintings for the given round of the 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

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: $reputation = repAvg * repWeight + opAvg * opWeight$

Figure 6.4. Finding Agent's Reputation

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 6.4 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.

6.4. Response Strategy

Since opinion responses generated by Blizzard may vary among the different implementations, we explain the details of it in the next section. For the reputation responses, it sends the reputation value calculated by the algorithm in Figure 6.4. Again, this may vary when the agent wishes to exaggerate its answer, send out a totally wrong answer or do not want to respond at all.

6.5. Implementations of Blizzard

In the following subsections, we will inspect four versions of Blizzard varying in their trust strategies. These variations are similar to those variations made for Frost. Both implementations use the same learning algorithm to model their environments as in the agent-based modeling case and they utilize a *Liberal* type of requesting strategy as explained in Section 4.2, but the first two differ in the number of requests they make and the next two vary in how they respond to others.

6.5.1. Blizzard-v1

In its base form, Blizzard-v1 responds to all queries that it receives from others. It generates as accurate answers as its expertise allows and gives out reputation information according to its findings by referring to the algorithm in Figure 6.4. It also takes two opinions per painting as its agent-based modeling counter-part does. That is, the value of n in Figure 6.3 is always two for all assignments of Blizzard-v1.

6.5.2. Blizzard-v2

Blizzard-v2 also uses the same response strategy as Blizzard-v1 does, but it also has a built-in functionality to determine how many agents to request opinions from. It utilizes the algorithm in Figure 6.3 and directs its queries to the top n agents in the list. According to the algorithm in Figure 6.3, it first gathers the history of previously recorded opinion request actions. Then 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 ask for their opinions.

6.5.3. Blizzard-v3

While using the same requesting strategy as Blizzard-v2 does, Blizzard-v3 also has the ability to cheat. Since the cheating behavior is consistent as in the case with agent-based modeling agent, Blizzard-v3 always provides its requester with the worst

opinion if it lacks the expertise to generate a better opinion. But it also provides its customer with the best avaliable service when it has the talent to do so (i.e., being an expert in that era).

6.5.4. Blizzard-v4

Blizzard-v4 adds the reciprocal behavior in its responses while keeping the consistent cheating from Blizzard-v3. The reciprocal behavior is again applied after the third round as in the agent-based modeling case. In the first three rounds, Blizzard-v4 responds to all incoming queries. In order to implement the reciprocity relations in Blizzard-v4, another data structure is not needed. By processing the reputation information it calculates from the action history, the agent can determine whether to respond to that agent or not.

6.6. Simulations

Here, we compare the four implementations of Blizzard. In each simulation, a version of Blizzard is compared with the next version, and in the last simulation Blizzard-v4 is experimented with different initial expertise values.

6.6.1. Blizzard-v1 vs. Blizzard-v2

Figure 6.5 plots the four metrics, comparing Blizzard-v1 with Blizzard-v2.

Average Appraisal Error: When we look at the appraisal errors, it is clear that Blizzard-v2 lowers its error rate consistently while the average appraisal error of Blizzard-v1 is stable after the fifth round. This shows that Blizzard-v2 improves its evaluations every round. This is partially due to the second figure, where we see that Blizzard-v2 purchases more opinions than Blizzard-v1. With more opinions, Blizzard-

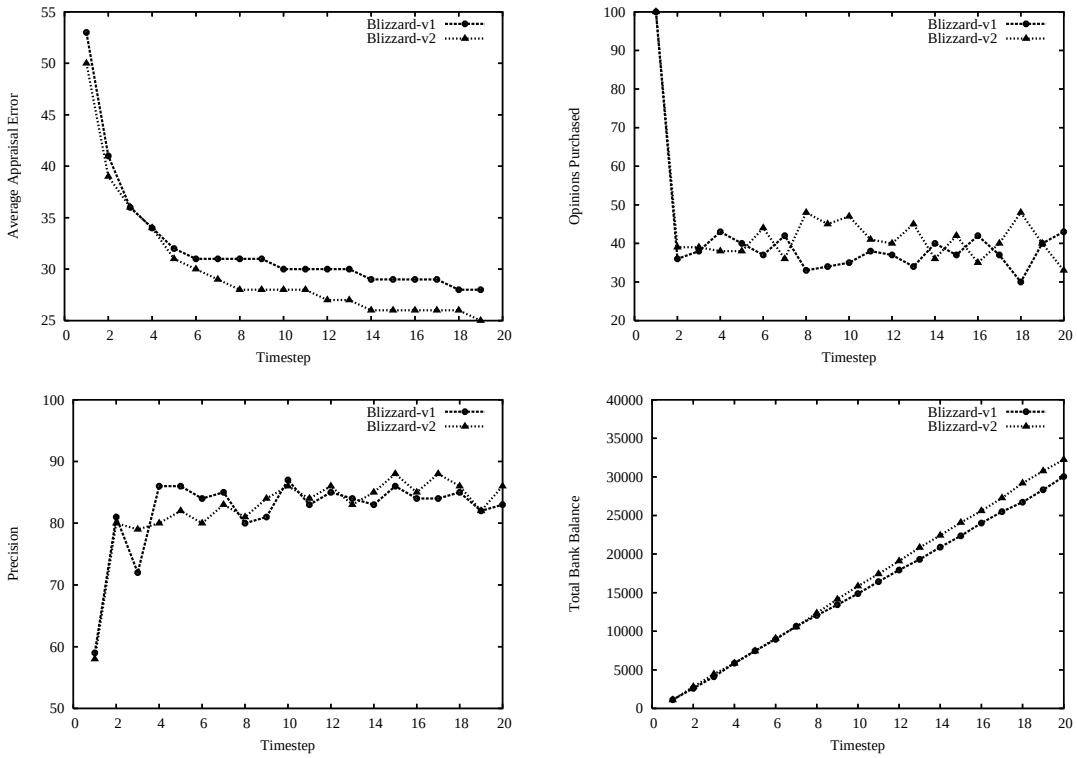


Figure 6.5. Blizzard-v1 vs. Blizzard-v2

v2 makes fewer errors.

Opinions Purchased: The first thing that differs from the agent-based modeling case is that Blizzard-v2 purchases more opinions than Blizzard-v1, which was the opposite in Frost-v1 vs. Frost-v2.

Precision: In terms of precision, they both find the experts well in the environment. Blizzard-v2 has a more increasing trend, while Blizzard-v1 is more stable throughout the simulation since it always asks the same experts every round. This also proves that Blizzard-v2 improves its models of others throughout the simulation.

Total Bank Balance: Although it spends more on other's opinions, Blizzard-v2's bank balance is still higher since it makes far fewer errors than its opponent. This means that it attracts more clients at each round, leading to more income.

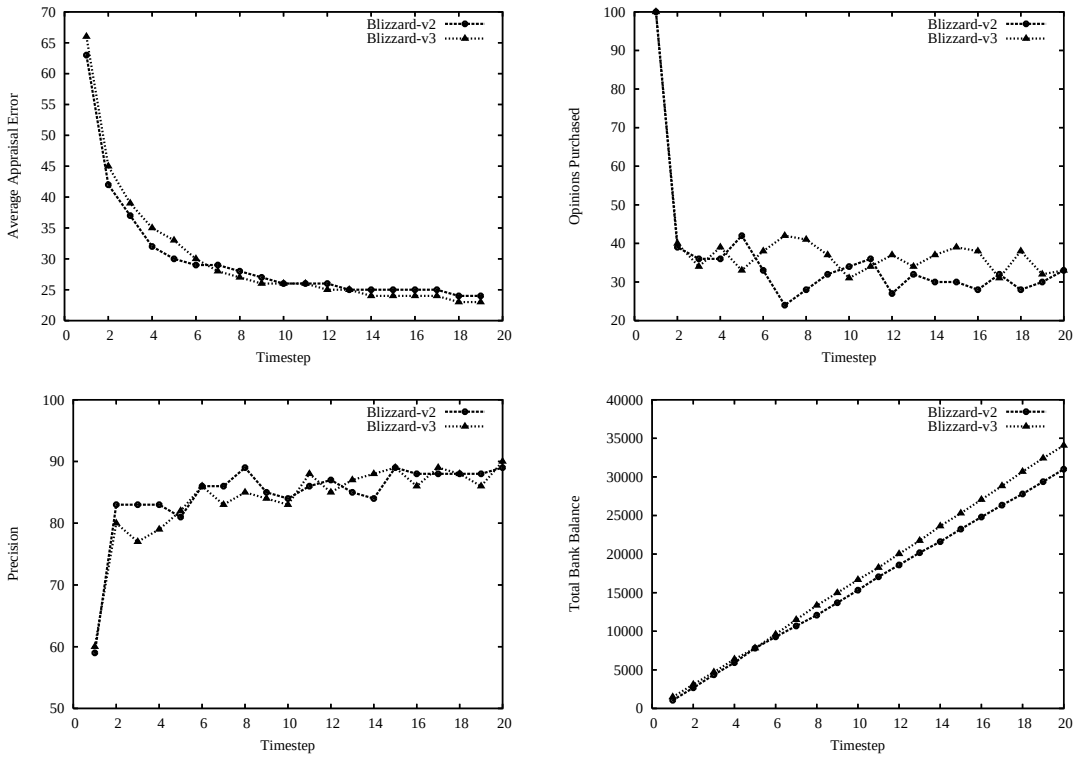


Figure 6.6. Blizzard-v2 vs. Blizzard-v3

6.6.2. Blizzard-v2 vs. Blizzard-v3

Figure 6.6 plots the four metrics, comparing Blizzard-v2 with Blizzard-v3.

Average Appraisal Error: Both agents exhibit a decreasing trend for the appraisal error. Blizzard-v3's error is slightly less especially after the first half of the simulation. The difference is small since the difference between two agents is mainly in their responding strategies rather than their requesting behavior.

Opinions Purchased: When we look at the number of opinions purchased, we see that Blizzard-v3 buys more opinions than Blizzard-v2 on average. Although they follow the same strategy for requesting opinions, the reason that Blizzard-v3 buys more opinions is the fact that it is assigned more paintings each round.

Precision: Previously, we have seen that consistent cheating helps agent-based modeling agents better identify agents with below average expertise. This does not have

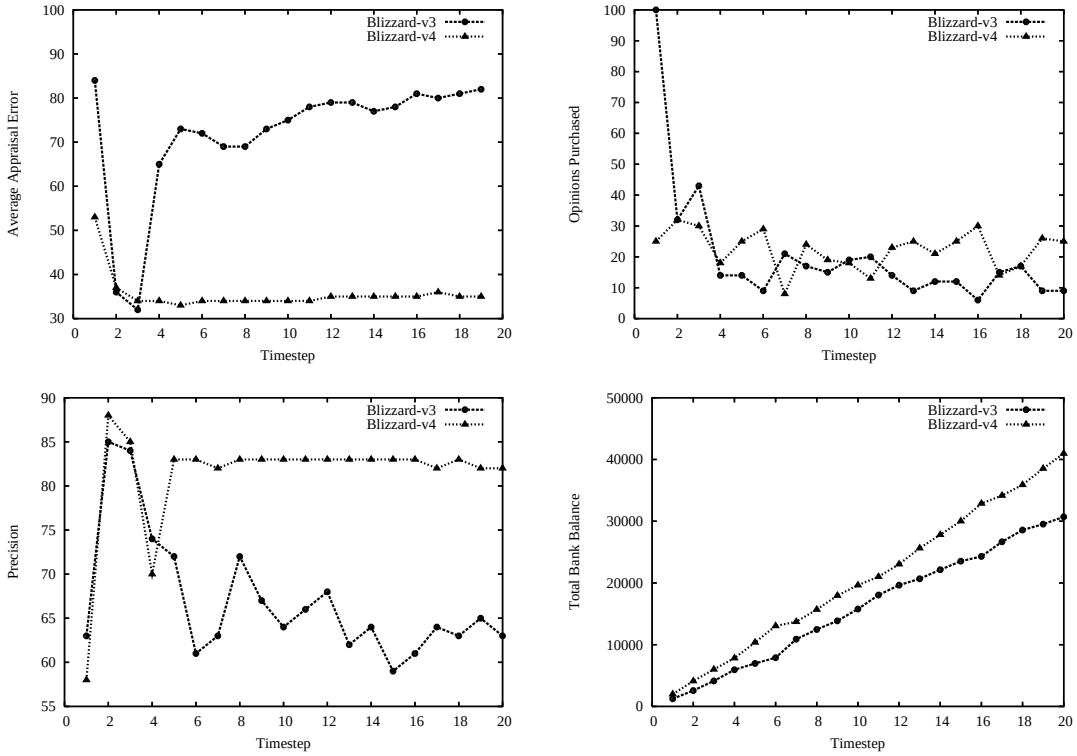


Figure 6.7. Blizzard-v3 vs. Blizzard-v4

the same positive effect on the action-based modeling agents. Since Frost builds up its agent models by incremental updates, a negative feedback coming from a cheating agent may have a huge effect on the ordering of agents (i.e., modeled expertise of others). But for Blizzard, every other agent is tested at the beginning of the simulation and further cheating behavior may not have any effect (neither negative nor positive) on better recognition.

Total Bank Balance: Although Blizzard-v3 purchases slightly more opinions than Blizzard-v2, it earns more money since its precision is better and its average appraisal error is lower.

6.6.3. Blizzard-v3 vs. Blizzard-v4

Figure 6.7 plots the four metrics, comparing Blizzard-v3 with Blizzard-v4.

Average Appraisal Error: When reciprocal behavior is added to the action-based

modeling agent Blizzard-v4, the results are very interesting especially for its opponent. Again as in the agent-based modeling case, the reciprocal behavior is applied after the third round. So when we observe the behavior after third round, the appraisal error of Blizzard-v3 begins to rise again to huge numbers.

Opinions Purchased: The number of opinion transactions in the environment is low compared to the experiments where agents respond to all queries. This is an expected behavior since experts in the environment do not wish to sell their opinions as before.

Precision: The precision of Blizzard-v3 also drops significantly after the third round since it struggles to find new experts to get opinions from. The precision of Blizzard-v4 also drops for a short period, but then it is able to regain its precision by finding new experts.

Total Bank Balance: Since Blizzard-v4 performs better on all metrics, the results lead to a huge difference in bank balance (this heavy defeat is not observed in the agent-based modeling case).

6.6.4. Blizzard-v4 with Varying Expertise

Figure 6.8 plots the three metrics, comparing Blizzard-v4 with varying initial expertise values. Again, we are using the same metrics as we did in the experiment while comparing the Frost-v4 agents among themselves, and we have run the simulation once in order to clearly show the effect of expertise distributions on success. Since their expected behavior is the same, the initial expertise values of the agents determine their level of success in the game.

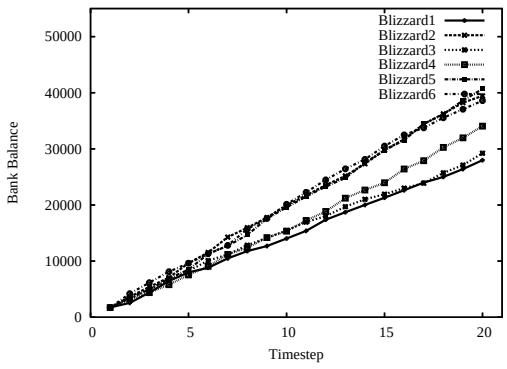
Expertise vs. Appraisal Error: We expect the results to be similar to the case with the experiment that involves Frost-v4 version agents only. We again observe high correlation between the expertise values and the appraisal errors. For example, the error in the appraisals of Blizzard4 in era 3 is the lowest in the society as shown in

Agent	Expertise				
	Era 1	Era 2	Era 3	Era 4	Era 5
Blizzard1	0.1	0.7	0.9	0.3	0.1
Blizzard2	1.0	1.0	0.4	0.3	1.0
Blizzard3	0.5	0.6	0.1	1.0	0.4
Blizzard4	0.3	0.8	1.0	0.8	0.2
Blizzard5	0.5	1.0	0.9	0.6	0.2
Blizzard6	0.8	1.0	0.4	1.0	0.4

(a) Initial Expertise Distribution

Agent	Appraisal Error				
	Era 1	Era 2	Era 3	Era 4	Era 5
Blizzard1	69.6	16.7	14.1	69.2	27.3
Blizzard2	13.4	12.2	40.8	73.0	13.0
Blizzard3	22.9	42.4	76.1	11.6	19.2
Blizzard4	70.2	21.3	13.1	25.8	21.7
Blizzard5	52.1	11.5	17.4	22.5	17.9
Blizzard6	26.5	11.1	61.7	10.4	18.9

(b) Appraisal Error



(c) Total Bank Balance

Figure 6.8. Blizzard-v4 with Varying Expertise

Table 6.8(b), since its expertise has the maximum possible value which can be seen from Table 6.8(a).

Bank Balance: The bank balance metric is a refleciton of the appraisal errors the agents make when evaluating their paintings. Since the average appraisal error of Blizzard5 is the least among other agents, it wins the game with the maximum bank balance. The bank balance disributions shown in Figure 6.8(c) are highly related to the apprisal errors in Table 6.8(b).

7. COMPARATIVE SIMULATIONS

The following sections compare the four versions of Frost and Blizzard with the same metrics we used in the previous experiments.

7.1. Frost-v1 vs. Blizzard-v1

Figure 7.1 plots the four metrics, comparing Frost-v1 with Blizzard-v1.

Average Appraisal Error: This is the comparison of the base cases of the two agent types. They both seem to lower their appraisal errors throughout the simulation.

Opinions Purchased: Since both take the same number of opinions per painting, their purchase of opinions exhibit the same behavior. The oscillations are again caused by the changes in the client shares.

Precision: The precision metric determines which learning behavior models its environment better since they both ask the same number of agents per painting. We see that Blizzard's precision rises noticably in the second half of the simulation and gets past that of Frost's.

Total Bank Balance: Since Blizzard-v1's precision is better than that of Frost-v1's, it earns slightly more and keeps its bank balance higher.

Conclusion: We can easily conclude from this experiment that Blizzard models its environment better than Frost given the same resources.

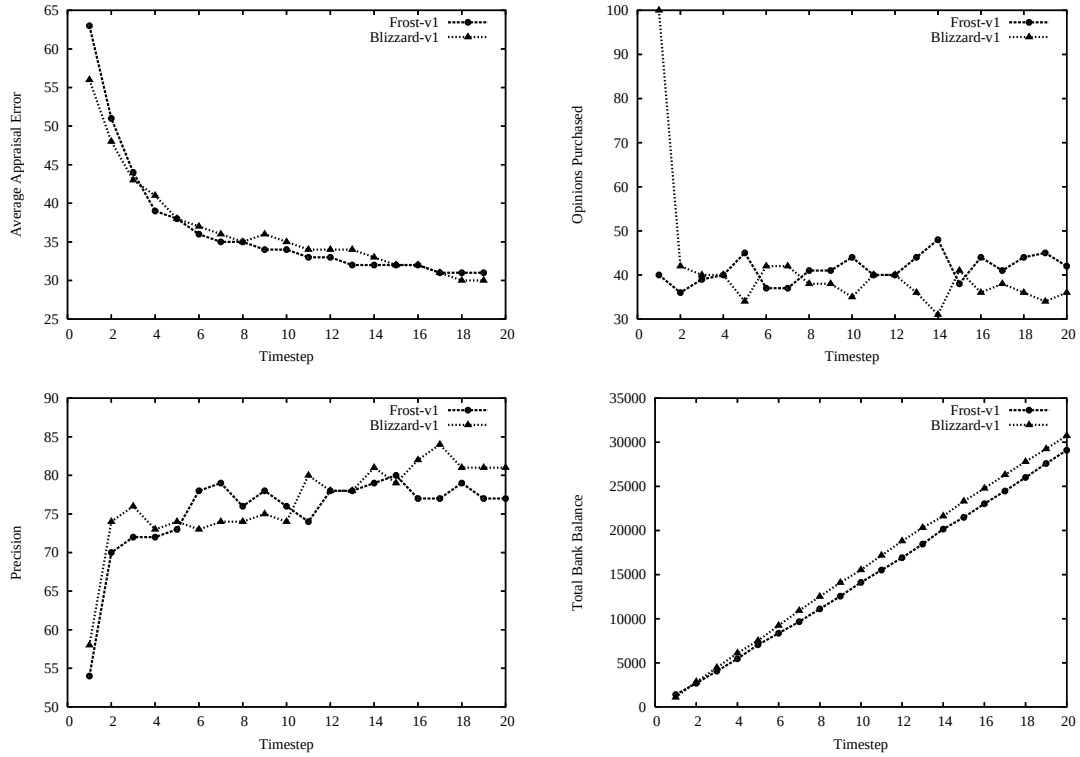


Figure 7.1. Frost-v1 vs. Blizzard-v1

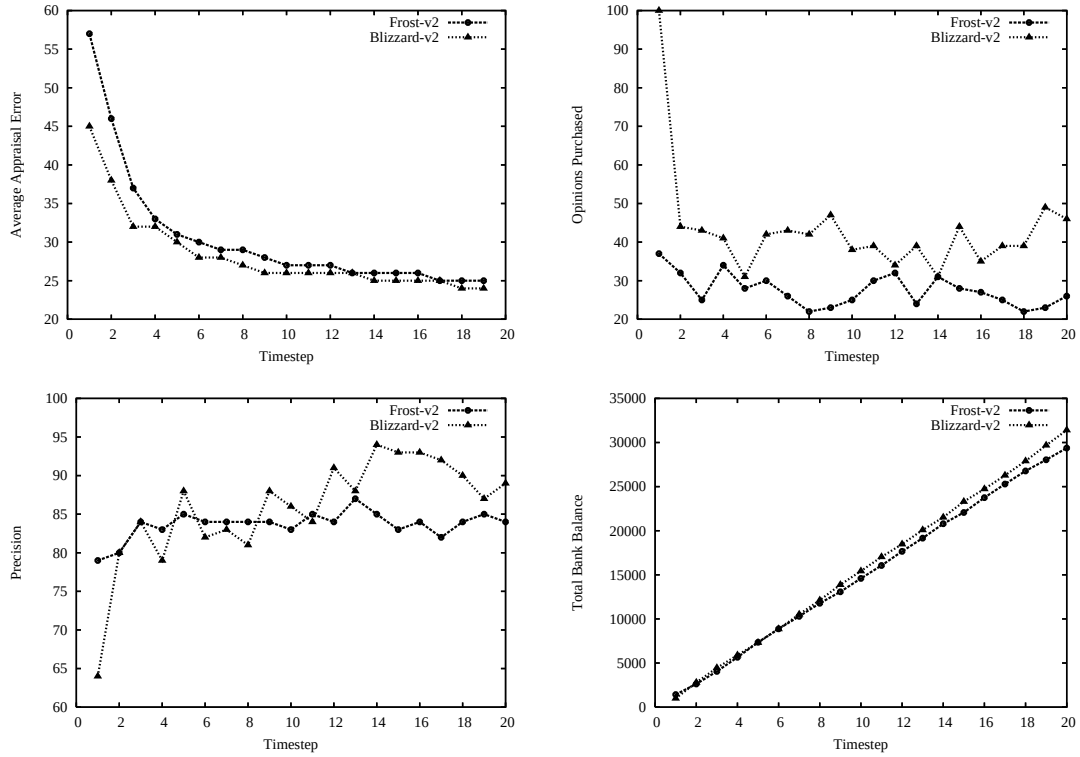


Figure 7.2. Frost-v2 vs. Blizzard-v2

7.2. Frost-v2 vs. Blizzard-v2

Figure 7.2 plots the four metrics, comparing Frost-v2 with Blizzard-v2.

Average Appraisal Error: When both agents have the flexibility to choose how many opinions to purchase per painting (i.e., according to their own expertise in that era), they seem to decrease their apraisal errors compared to the previous simulation.

Opinions Purchased: When we look at the opinions purchased, Blizzard-v2 nearly doubles the number of times Frost-v2 consults another agent for its opinion.

Precision: Here, the extra opinions bought pay off. After the first half of the simulation, Blizzard-v2's precision gets past that of Frost-v2's by a significant amount, while Frost-v2 stays almost stable in terms of precision. This shows that Blizzard-v2 learns its environment better than Frost-v2 does.

Total Bank Balance: Although Blizzard-v2 takes more help from other agents (i.e., spend more on opinions), it maintains a superiority over Frost-v2 in terms of bank balance. Blizzard-v2 begins to make more money than Frost-v2 in the second half of the simulation where its precision is far better towards the end.

Conclusion: The main conclusion from this simulation is that more flexibility in determining the request size of the agents (i.e., how many opinions to take) leads to better evaluations and precision compared to the fixed requests case.

7.3. Frost-v3 vs. Blizzard-v3

Figure 7.3 plots the four metrics, comparing Frost-v3 with Blizzard-v3.

Average Appraisal Error: When we have experimented the effects of consistent cheating separately in different learning behaviors, we have observed that Frost is

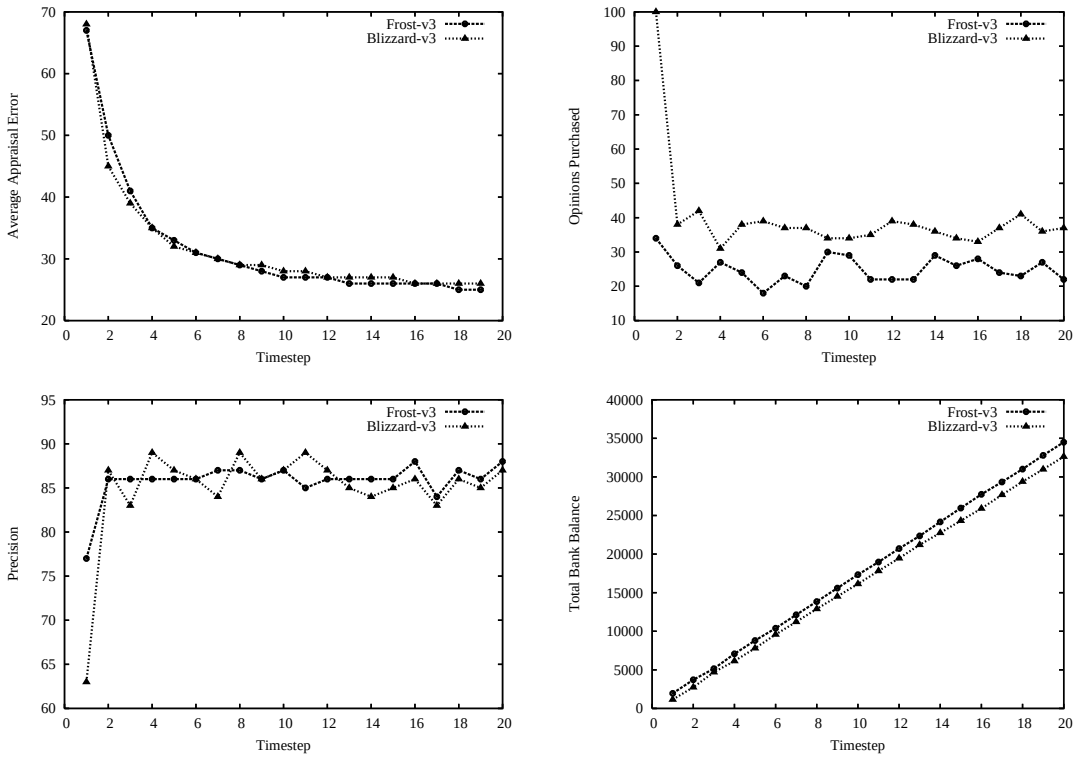


Figure 7.3. Frost-v3 vs. Blizzard-v3

positively affected while cheating does not have a noticable effect on Blizzard. This comparative experiment also justifies this finding since Frost does slightly better on average when all the metrics are considered. The appraisal errors seem almost the same, but Frost-v3 does slightly fewer errors than Blizzard-v3 does.

Opinions Purchased: Blizzard-v3 again tries to model its environment by purchasing more opinions than Frost-v3 does.

Precision: Although Blizzard-v3 gets more help from other agents in the environment, its precision stays lower than that of Frost-v3's.

Total Bank Balance: When we add the lower precision of Blizzard-v3 to the fact that it spends more on others' opinions, the bank balance stays lower for Blizzard-v3.

Conclusion: We conclude from this experiment that cheating negatively affects Blizzard more than Frost in terms of precision. This is because action-based modeling

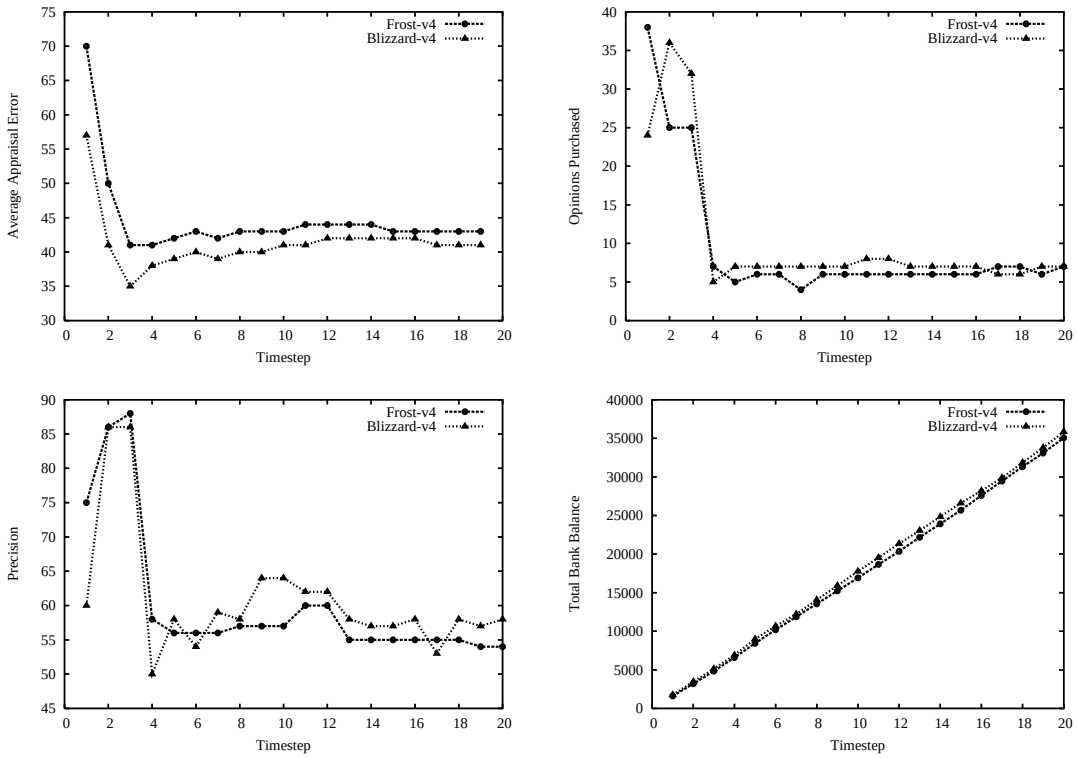


Figure 7.4. Frost-v4 vs. Blizzard-v4

agents cannot adjust to conflicting information easily whereas the agent-based modeling agents adjusts more easily.

7.4. Frost-v4 vs. Blizzard-v4

Figure 7.4 plots the four metrics, comparing Frost-v4 with Blizzard-v4.

Average Appraisal Error: We have previously seen that when there are reciprocal agents in the environment, their opponents struggle in finding the right experts. In this experiment every agent in the environment applies reciprocity after the third round and the appraisal errors do not decrease further for both agents. However, Blizzard-v4 does significantly less errors.

Opinions Purchased: When we look at the opinions purchased, we see that this experiment exhibits the lowest number of opinion transactions between the agents since

the agents do not wish to sell their opinions.

Precision: Since the agents have a rigid reciprocal behavior (i.e., expect much to sell opinions), the precision is almost the half of that in the previous experiments for both agents. Note that the precision is far better until the third round.

Total Bank Balance: When we look at the winning metric, Blizzard-v4 has a higher bank balance at the end in which the difference may seem ignorable. Although Blizzard-v4 models its environment better, and thus makes slightly fewer errors than Frost-v4, it maintains its models by buying more opinions than Frost-v4. As a result, they are tied in terms of bank balance.

Conclusion: This experiment shows the results of reciprocal behavior that can be observed in an heterogeneous environment. When all agents apply reciprocity, they struggle to find others for purchasing opinions.

7.5. Frost-v4 vs. Blizzard-v4 (4 agents each)

Figure 7.5 plots the four metrics, comparing Frost-v4 with Blizzard-v4. Here, we use the same agent society that we have used in the previous experiment. However, there are eight agents in this experiment instead of six, four for each agent type. We aim to see the effect of increasing the environment size by a small amount.

Average Appraisal Error: According to the appraisal errors, Blizzard is again better on painting evaluations. However, the difference is not much and both agents cannot perform well on appraisals compared to the societies with no reciprocal agents even if the society grows bigger in size (i.e., more opportunities on selecting a suitable service provider). This is because the behavior of the agents play an important role on the sharing of information in the society rather than the number of agents in the society. That is, since the agents have a reciprocal response strategy, the sharing of opinions is

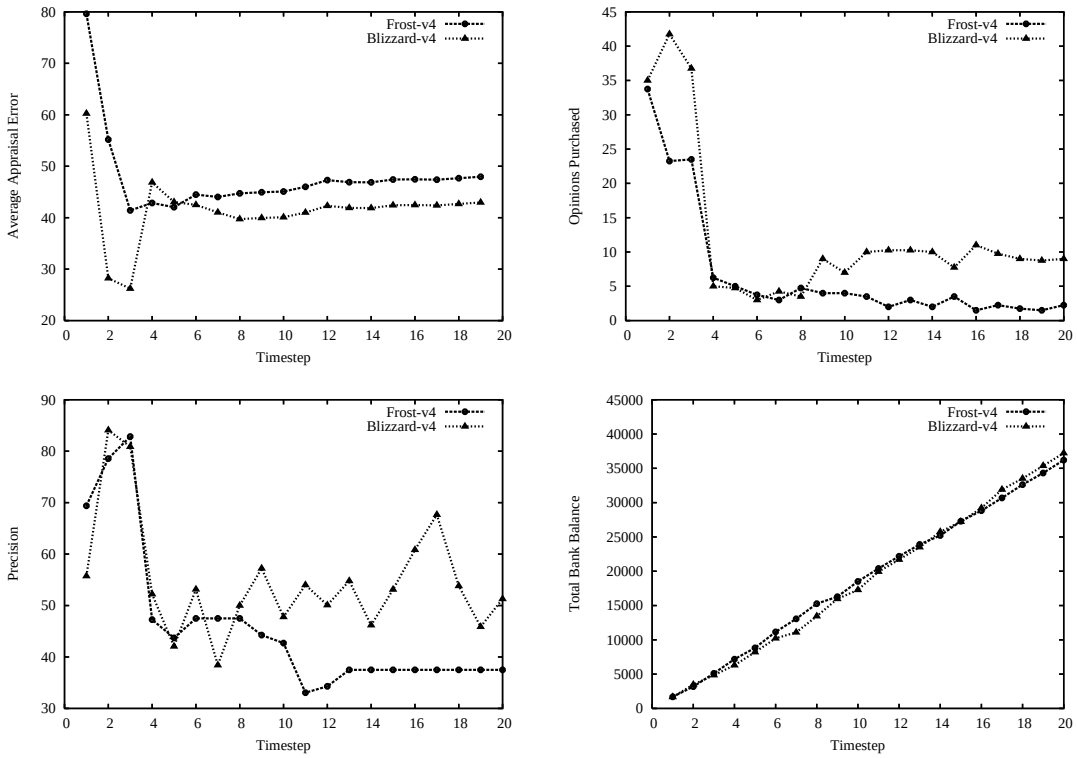


Figure 7.5. Frost-v4 vs. Blizzard-v4 (4 agents each)

limited in the environment preventing the agents from evaluating their paintings well.

Opinions Purchased: The number of opinion transactions is still low compared to the previous experiment, but Blizzard buys slightly more opinions which decreases the rate it makes errors on its evaluations.

Precision: As we can expect from the appraisal errors, Blizzard is more precise on finding the suitable service providers although they are very hard to be found because of their reciprocal behavior. The basic reason behind this is that Blizzard tries to know all agents in the environment by populating its actions from the very beginning of the simulation giving it a wider range to choose service providers from.

Total Bank Balance: Although Blizzard buys more opinions than Frost and thus spends more money on evaluations, it still manages to compete with its opponent Frost

in terms of bank balance, because it is more precise and makes fewer errors.

Conclusion: The metrics show that the society used in this experiment exhibits almost the same behavior with the society used in the previous experiment. In general terms, the society as a whole is expected to perform better when its size increases because it is more probable that there are more experts for each painting era (which may help others in evaluating their paintings). But since the agents in this experiment do not wish to sell their opinions to all requesting agents, increasing the size of the society does not affect the quality of evaluations at all.

7.6. All Versions

Figure 7.6 plots the three metrics, comparing all agent versions. In this last experiment, we are comparing all the agent versions in the same environment to see how successful they are against each other. Here, each agent is competing against every other strategy we have built, and we have a chance to evaluate them in terms of their bank balances. In order to see the effect of the initial expertise distributions, we have shown the expertise values of each agent for every era in Table 7.6(a). The corresponding appraisal errors are shown in Table 7.6(b). Finally Figure 7.6(c), plots the bank balances of the agents for increasing timesteps.

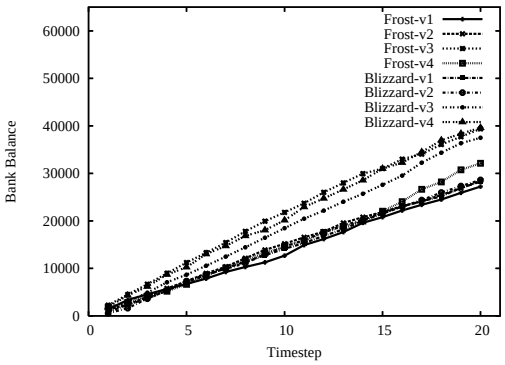
Expertise vs. Appraisal Error: In this experiment, we do not expect the appraisal errors of the agents to be highly correlated to the initial expertise values. Since there are only two agents in the society with reciprocal response strategy, the opinion flow in the environment is expected to be high. Following this, agents with low initial expertise values may also gather opinions from other agents and evaluate their paintings better. For example, Blizzard-v4 has relatively low expertise values compared to Frost-v3, but it still manages to evaluate its paintings slightly better than Frost-v3. Although the first two versions of each agent types may have high expertise distributions, they cannot

Agent	Expertise				
	Era 1	Era 2	Era 3	Era 4	Era 5
Frost-v1	0.4	0.5	0.9	0.1	0.7
Frost-v2	0.5	0.1	0.5	0.8	0.2
Frost-v3	0.2	1.0	0.6	0.8	1.0
Frost-v4	0.2	0.3	0.5	1.0	0.3
Blizzard-v1	0.7	0.4	0.7	0.7	0.7
Blizzard-v2	0.6	0.6	0.6	0.7	0.6
Blizzard-v3	0.9	0.7	0.8	0.3	1.0
Blizzard-v4	0.7	0.3	1.0	0.2	0.8

(a) Initial Expertise Distribution

Agent	Appraisal Error				
	Era 1	Era 2	Era 3	Era 4	Era 5
Frost-v1	23.0	25.3	21.5	25.6	18.9
Frost-v2	23.0	28.3	25.1	25.1	22.4
Frost-v3	22.5	11.9	24.9	25.7	13.8
Frost-v4	23.9	29.9	33.4	12.7	23.1
Blizzard-v1	27.6	40.0	19.6	25.9	14.88
Blizzard-v2	26.0	22.0	21.6	25.9	24.6
Blizzard-v3	26.6	21.7	18.2	27.0	14.4
Blizzard-v4	21.1	20.0	12.6	26.8	12.1

(b) Appraisal Error



(c) Total Bank Balance

Figure 7.6. All Versions

maintain their evaluations as well as the more improved versions as expected.

Bank Balance: The bank balances are again highly correlated with the appraisal errors. In the previous experiments, we have seen that low appraisal errors together with high precision lead to the maximum bank balance in the society. Here, we are adding another dimension to the simulation, the initial expertise values. The agents with the minimum appraisal errors (possibly the agents with high expertise values) end the game with the maximum bank balances. The first two versions of each agent types have the lowest bank balance in the environment as expected from the previous experiments. The remaining versions are ranked according to their initial expertise values. The most outstanding example is Frost-v4. Since the distribution of its expertise values is relatively low, its bank balance stays lower than its opponents.

7.7. Overall Performance

We have tried to compare the performances of Frost and Blizzard by matching their different versions. Except the third experiment (where cheating was involved), Blizzard was slightly better on most metrics. Generally, it maintains its precision on finding the experts in the environment via sending out more queries in the beginning. Modeling others better and faster has a cost, but since it also leads to more client share in the long run, Blizzard also does well in bank balance.

When there is cheating involved in the environment, Blizzard is not affected much but since Frost can use it to its advantage, it improves its metrics above those of Blizzard's.

In the environment where reciprocal agents exist, the important thing is to maintain the reciprocal relationships since they empower both participants during their duration (i.e., both agents make better evaluations). But since the type of reciprocal behavior used in our experiments is more rigid, the agents are not able to maintain their relations further. Changing the reciprocal behavior may exhibit different results in the metrics. For example, if the responding agent's tolerance is high when choosing

whom to respond (i.e., expect less to sell its opinions), the number of opinion interactions will certainly increase in the society which will lead to better coalitions among the agents.

8. CONCLUSIONS

In this thesis, we have implemented four different versions of agents for both learning behaviors. In each improved version, we have tried to add one more functionality to the existing ones and see the effect of it on the metrics.

In the basic versions, agents have the intention to get a constant number of opinions from their environment. When we enable them to adaptively decide on the number of opinions they will purchase, we see that they utilize their trust models better. Because, they no longer need to get the opinions of an agent with average expertise when they identify an expert earlier, or try to get more opinions when they are not sure of the current findings they got.

The next improvement has been added in the third version which changes the response behavior of agents. With the consistent cheating behavior implemented, the agents provide their requesters with totally wrong opinions when they do not have the enough expertise to generate better opinions. We have seen that this also helps agents better identify the others with poor expertise. Thus, they improve their precision and decrease the rate they make errors in evaluating their paintings.

The last change we have added is again related to the response behavior. But this time, we have given the agents the capability to choose the quantity of their responses rather than the quality of the generated responses. Regarding the new fuctionality, agents have a reciprocal behavior in which an agent only responds to the agents that previously responded to its queries well. This behavior has a huge effect on the society since after a certain time in the simulatios, agents struggle to find others to get opinions from. This is directly related with the type of reciprocal behavior implemented in the agent architecture. If the agents have more tolerance when deciding whom to respond, the opinion transactions will increase in the environment.

In our previous research, we have shown that the response strategy of the agents

determine the flow of opinions in the society. The existence of more reciprocal agents in the environment increases the percentage that the opinion requests are left unanswered [22]. The decrease in sharing opinions in the society also decreases the quality of the painting evaluations and thus increase appraisal errors.

Beginning from the earlier versions to the most complex agent architecture, the experiments show that action-based modeling agents can model others more accurately and faster, but they need substantially higher number of interactions to build these models. When consistent cheating is enabled, it is seen that action-based modeling agents do not get affected from consistent cheating (neither negative nor positive). However, when agents start to answer erratically (as a result of reciprocity), both agent types struggle to maintain accurate models and they make more errors. But again, Blizzard models its environment slightly better than Frost does even when there is not much information available because of the response behavior.

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 [17]. 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. But if the competition is reformed to allow competition among groups of agents, the results of the group performances may be interesting.

In fact, these coalitions may not always be explicitly formed. During the games, a significant number of small sized coalitions are formed and destroyed between the agents. These implicitly signed contracts empower the participants for the duration that they exist. In the experiments where reciprocal agents exist, it is shown that although the action-based modeling agents model their environment better previously,

they also struggle against their agent-based modeling opponents as they cannot maintain reciprocal relations further. Following this finding, the agents may further be improved by adding more adaptivity to certain situations like the one when they cannot manage to purchase opinions from other agents. By adding this functionality, we expect the agents to be more resistant on situations when there is less information flow in the society.

REFERENCES

1. Huhns, M. N. and M. P. Singh, Agents and multiagent systems: Themes, approaches, and challenges. In *Michael N. Huhns and Munindar P. Singh, editors. Readings in Agents. Morgan Kaufmann, San Francisco, 1998*, chapter 1, pages 1–23. 1998.

2. Fisher, M. and M. Wooldridge, On the formal specification and verification of multi-agent systems. In *International Journal of Cooperative Information Systems*, 1994.

3. Castelfranchi, C. and R. Falcone, The socio-cognitive dynamics of trust: Does trust create trust? In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber-societies*, LNAI 2246, pages 55–72. Springer-Verlag, 2001.

4. Sabater, J. 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.

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

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

7. Barber, K. S. 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.

8. Barber, K. S., K. Fullam, T. B. Klos, G. Muller, J. S. Rosenschein, J. Sabater, Z. Topol, and L. Vercouter, The agent reputation and trust (ART) testbed archi-

tecture. In *The Workshop on Trust in Agent Societies at The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 50–62, July 2005.

9. Barber, K. S., K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, and Z. Topol, The agent reputation and trust (ART) testbed competition game rules. Laboratory for Intelligent Processes and Systems Technical Report TR2004-UT-LIPS-028, 2004.

10. Conte, R., N. Gilbert, and J. S. Sichman, MAS and social simulation: A suitable commitment. *Lecture Notes in Computer Science*, 1534:1–9, 1998.

11. Huynh, T. D., 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.

12. Barber, K. S. 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, New York, NY, USA, 2006. ACM Press.

13. Stranders, R., Argumentation based decision making for trust in multi-agent systems. Master's Thesis, Delft University of Technology, 2006.

14. Banerjee, D., P. Dasgupta, S. Saha, and S. Sen, Reciprocal resource sharing in P2P environments. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 853–859, 2005.

15. Glass, A. and B. J. Grosz, Socially conscious decision-making. *Autonomous Agents and Multi-Agent Systems*, 6(3):317–339, 2003.

16. Singh, M. P. and P. Yolum, Service graphs for building trust. In Robert Meersman and Zahir Tari, editors, *CoopIS/DOA/ODBASE (1)*, volume 3290 of *Lecture Notes*

in Computer Science, pages 509–525. Springer, 2004.

17. Airiau, S., 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.

18. ART Testbed Competition, Hakodate, Japan. http://www.lips.utexas.edu/art-testbed/competition_results_final.htm, 2006.

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

20. Harmon, M., Reinforcement learning: a tutorial, 1996.

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

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