Trust Strategies for ART Testbed

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

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

Abstract— **Recently several trust models have been developed. However, different trust models make different assumptions about agents' environments, making it difficult to evaluate and compare different models. An accurate examination requires different models to be tested on the same environment and evaluated using precise metrics. For this reason, we adopt well-known trust concepts and formulate them as agent strategies that can be applied in the Agent Reputation and Trust (ART) Testbed Simulation Environment. We devise trust strategies that can be applied at different stages of agents' interactions. We study different metrics for each stage and compare various strategies using these metrics.**

I. Introduction

Trust is a crucial element in business dealings. The characteristics of the business, the traits of the business participants, the interactions they are involved in influence how trust is engendered. In recent years, various trust models have been developed [13], [9], [2]. However, comparing them with each other have been difficult since each model makes different assumptions about the environment or allows different set of interactions.

Agent Reputation and Trust (ART) Testbed simulates a business environment as a game [6]. Agents that participate in the game are service providers that can offer information services. The information they offer can be based on their own knowledge or can be bought from other service providers. The game is structured so that buying information from others benefit a seller if the information sources are trustworthy but may harm the seller if they are not. This raises the question of whom to trust.

We study this question in three stages: modeling others, requesting information, and responding to questions. The modeling strategies study different ways to represent other agents and their interactions. They also study the type of information that should be maintained and how this information should be interpreted or updated. The requesting strategies study when and from whom the agent will request information. The main question here is when does an agent decide that the other party is trustworthy and request information? The responding strategies are used to decide under which circumstances and to whom the agent should sell information. The responding stage is indirectly related to finding trustworthy parties, since other agents may act in the agent's interest only if the requester has previously acted cooperatively.

To study different trust models systematically, we have designed and implemented strategies for the three mentioned stages of the ART game. At the end of each game, the simulation environment provides us information about the game such as the money earned, average error, and so on. We use these values to devise metrics that can be used to compare different strategies.

The rest of the paper is organized as follows: Section 2 is an overview of the game architecture together with the simulation environment and the agent model. Then, in Section 3 we propose strategies for the agent's trust model by using references to the implementation of the agent's code introduced in Section 2. Section 4 shows the statistics collected from the games simulated with the models proposed in Section 3 and makes some observations about the overall performances. Then we conclude the paper in Section 5 by comparisons to the existing trust models used in this area of research.

II. Background

The algorithms and methods proposed in this paper are implemented in the ART Testbed software, which is also a simulation platform for making test runs in order to compare different strategies. The ART (Agent Reputation and Trust) Testbed provides the researchers with an environment to model and run their agents to collect statistics about how successful their strategies are. Next, we explain the rules of the ART game [5], the characteristics of the simulation environment [7], and finally the implemented agent model.

A. 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., appraiser) that is selling its opinion when requested. Service consumers as well as other service providers may be willing to purchase opinions about a painting. For each painting, its era of expertise is known. Each service provider starts the game with high expertise in some eras. In addition, the service providers can query each other to find out the reputations of other agents for some eras. Throughout the simulation, there are no guarantees about the correctness of replies. That is, a service provider may provide wrong information about a provider or may provide inaccurate information about the reputation of another service provider. Similarly, since agents are heterogeneous (i.e., designed by different parties), they may be following different strategies for carrying out their tasks.

After a service provider consults whoever is necessary, it forms a final opinion about the requested painting. After submitting this final opinion, the true value of the painting is revealed. This allows 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. The service provider earns a certain amount of money for every answer it provides to the requesting agents. The more accurate answers a service provider generates, the more likely the service consumers buy opinions from that provider. The main aim of the game is to end with the maximum bank balance.

B. 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 manipulating the communication of agents through various types of messages and making the necessary computations. The database is used to store information about each game and view the statistics of a previously completed game. The simulation environment works as follows.

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. Client shares are determined for each time step and clients are assigned to the appraisers accordingly. After that, inter-agent communications begin (opinion requests and responses, reputation requests and responses). 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 and related researcher-coded methods is given in detail in the next section. After the transactions are complete, the simulator collects 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 gives the actual painting values to the agents as feedback.

C. 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 the abstract Agent class and filling up the methods that describe the agent's behavior and provide inter-agent communication.

The public parameters of the game are revealed to all agents that participate. 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 unknown).

The following list gives the order of interactions of each agent at each timestep. This sequence of actions is enforced by the simulation environment.

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 *Opinion-ReplyMsg* from the previous time step as additional information.
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 have 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 *Opinion-ReplyMsg* by finding the appropriate opinions that are already sent to the simulator.

III. The Proposed Strategies

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. We expect the response strategy of the agent to influence 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 on the modeling strategy of the environment. In this paper, we present several ways to keep track of the other agents' past behavior as far as the relations with the modeling agent are concerned. The following subsections provide in depth knowledge about each strategy and possible ways to implement them in the agent's algorithm.

A. 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 modelling strategies.

For the modeling of other agents, we capture the following information:

- 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, we record the expertise as multidimensional.
- Sociability: The *sociability* of an agent denotes how well the agent answers reputation requests [11]. The higher the sociability, the better the agent is knowing the reputation of others. Similar to the modelling of expertise, sociability of an agent can vary based on era. Hence, we again model the sociability of each agent for each era separately.
- References: The *references* of an agent is a list of agents that have been consulted to receive reputation information. 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. In our previous research, we have seen that the number of previous interactions with an agent affects the level of trust between agents [12].
- 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, we maintain this information in a complex data structure. As explained above, some information about an agent vary based on 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).

At each round of the simulation, certain events trigger update operations on some or all of the agents (again on some fields of the agent structure) in the list. The following is the list of these events and the types of update operations they lead to:

- After *Opinion Replies* are gathered: For each opinion reply message, the difference between the true painting value and the value gathered from the agent is calculated in order to update the agent's expertise in the given era. The agents that are consulted in the previous round to gather reputation information about that agent (the agent's references) are also updated according to the appraisal error. After all references of the agent are updated, they are removed from the agent's references list. The update operation is proceeded as follows:

$$\begin{aligned} Agent.Expertise &= Agent.Expertise \times \{1 + [(1 - \\ &\quad Agent.Expertise) \times UpdateMargin]\} \\ RefAgent.Sociability &= RefAgent.Sociability \times \{1 + [(1 - \\ &\quad Agent.Sociability) \times UpdateMargin]\} \end{aligned}$$

  where *UpdateMargin* is positive if the difference between the true painting value and the appraised value (appraisal error) is smaller than a threshold and it is negative otherwise. The above case shows the situation when the appraisal error is small enough to update the agent's expertise upwards. The closer the expertise of the agent to 1, the slower it progresses upwards after a successful evaluation. In the above equations, *Agent* represents the provider of the opinion and *RefAgent* represents a reputation provider which is previously consulted to get information about *Agent*.
  When the appraisal error is more than the threshold value (UpdateMargin is negative), the agent's expertise and the corresponding referencing agent's sociability are updated downwards as below:

$$\begin{aligned} Agent.Expertise &= Agent.Expertise \times \\ &\quad (1 + UpdateMargin) \\ RefAgent.Sociability &= RefAgent.Sociability \times \\ &\quad (1 + UpdateMargin) \end{aligned}$$

- After *Reputation Accept or Declines* are gathered: According to each reputation accept or decline message, the sociability of the agent (which sent the reply) is updated by a small multiplier which can be shown as follows:

$$\begin{aligned} Agent.Sociability &= Agent.Sociability \times \\ &\quad (1 + SocMultiplier) \end{aligned}$$

  where *SocMultiplier* is positive for accepted queries and it is negative for rejected queries.
- After *Reputation Replies* are gathered: For each reputation reply message, the expertise of the agent (about which the reputation information is gathered) is updated according to the sociability of the referencing agent and the referencing agent is added to the references of the updated agent. The formula for the update operation is as follows:

$$\begin{aligned} Agent.Expertise &= Agent.Expertise \times [1 + (Difference \\ &\quad \times RefAgent.Sociability)] \end{aligned}$$

  where *Difference* is obtained by subtracting the agent's current expertise value from the value gathered via the reputation reply message.

B. 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). We propose two requesting strategies.

- 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* request strategy is much simpler than the *Liberal* request strategy. The following paragraph describes the process of how agents are selected to request opinions and reputation information.

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, the agent checks the paintings assigned to it and makes an ordering of agents according to their expertise values stored in the agent structure for each era that it has to evaluate a painting. 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. We allow the agent to vary the value of n according to the agent's 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 opinion. However, if the agent has no idea about the era, 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 sociability values stored in the agent structure. The agents with the top sociability values are asked for reputation information. To answer the second question, i.e., to decide about which agents to gather information, an ordering based on available agent information is made. That is, the less we know about an agent, the more likely we would need reputation information. Hence, the agent has to figure out how much information it has about other agents. We measure the amount of information we have about an agent in terms of the number of previous interactions (asking opinions and asking reputations). This can be checked with the below condition:

$$OpinionsAsked + ReputationsAskedAbout \leq TimesAskedMax$$

where *TimesAskedMax* is a threshold value which determines whether enough information about the agent is gathered before.

Satisfaction of this condition ensures that we know little about that particular agent. However, there can be many agents in the system whom we know little about. Hence, it would be unfeasible to try to know everyone equally well. Conversely, we only want to know more about agents who have already shown their expertise by answering some queries correctly. For this reason, we also require that agents should have an expertise between a certain range. This can also be defined by the following condition:

$$AskingExpertiseMin \leq Agent.Expertise \leq AskingExpertiseMax$$

C. 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. The response strategy determines the reputation of the agent within its environment. We present two different response strategies of the agent: a *reciprocity* based response strategy and a *respond all* requesting type of strategy. In both strategies, the quality of the response is kept the same (reply all queries as accurate as the past knowledge allows), but the quantity of responses (the number of agents to respond) differs.

Respond All: An agent aiming a high reputation value will try to respond as accurate 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 to earn money since the agent's answer may help others to respond to customer requests correctly. A realistic exchange of opinions that take place in real life businesses are those of reciprocity where agents only respond to requests coming from the agents that respond to its requests correctly. This is kind of a reciprocity relation in which both agents are supposed to benefit from, if they're 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 a more sociable 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 which agents have answered its queries correctly.

IV. Metrics

The experimental simulations are run with the variety of ways explained in the previous section and the following metrics are used in order to compare them. For each separate strategy, we propose some statistical parameters to calculate:

In order to compare the distinction between an agent building up its model on reputation information gathered from the environment and a relatively unsociable agent which does not prepare its trust model to depend on other's evaluations, we generate the following metrics:

- Bank balance: Used to evaluate the overall success of the agent. It can be thought separately as income and expenses
- Average appraisal error: Another measure of success which is directly related to the client share

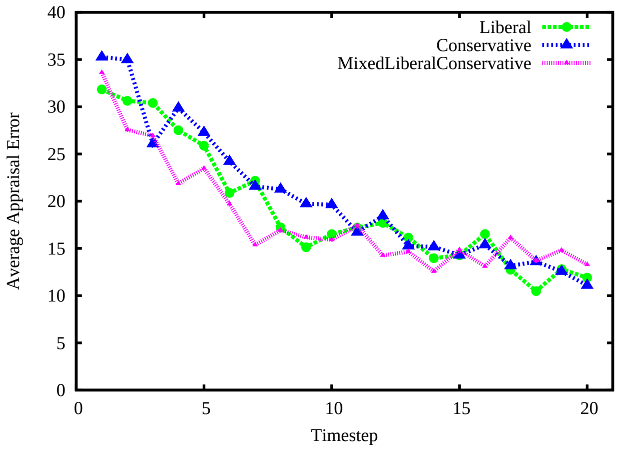

Fig. 1. Average Appraisal Error

Like the requesting strategy, we've proposed two methods on how the agent will manage the responses that it will give to the queries directed to it. The following metrics are used to compare the two methods:

- Bank balance: This time, it is mostly used to see how much income the agent makes out of the information it provides.
- Percentage of responses: Used to see how many responses are sent out of all queries directed to the agent.

Figures 1, 2, and 3 show the distribution of the above parameters with respect to the discrete timesteps of the simulation. Each simulation is run three times with 100 agents and the averages are reported. The simulations employ a variety of societies that differ in the strategies used.

First, we look at three societies that differ in their requesting strategy. As explained in Section III-B, we compare a society of *Liberals* with a society of *Conservatives* and a mixture of the two. All populations use the *Respond All* strategy for answering queries. We first study how these strategies affect the average appraisal error; that is the average of the differences between the paintings' true values and the agents' appraised values. Figure 1 plots this error for increasing timesteps. The *Liberal* strategy is more effective in the beginning of the simulation. The obvious explanation for this is that agents initially know few agents around and sharing information about each other helps them make fewer mistakes. However, with the *Conservative* strategy, agents ask others their opinions and thus learn how trustworthy other agents are on their own. As the simulation continues, agents with both strategies learn about the experts in the environment so that they start to make fewer mistakes. Meanwhile, the society with agents that employ *Conservative* as well as *Liberal* strategy generally make fewer mistakes than the other two strategies.

Next, we compare three populations that use *Respond All*, *Reciprocity*, and both strategies, respectively. All of these three populations use the *Liberal* strategy for requesting reputations. Figure 2 displays the *Percentage of Responses* versus the timesteps of the simulation. The results for the *RespondAll* and

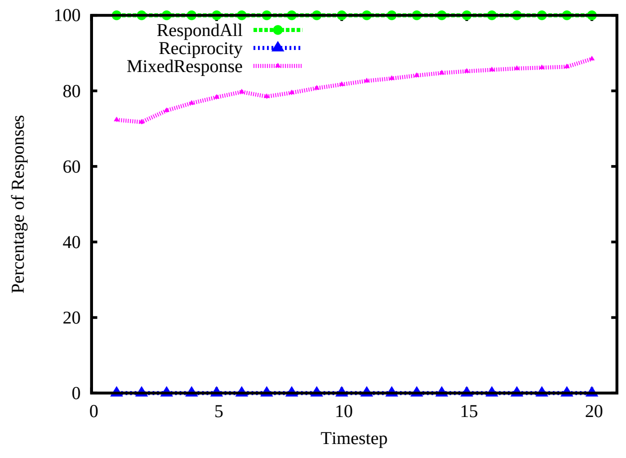

Fig. 2. Percentage of Responses

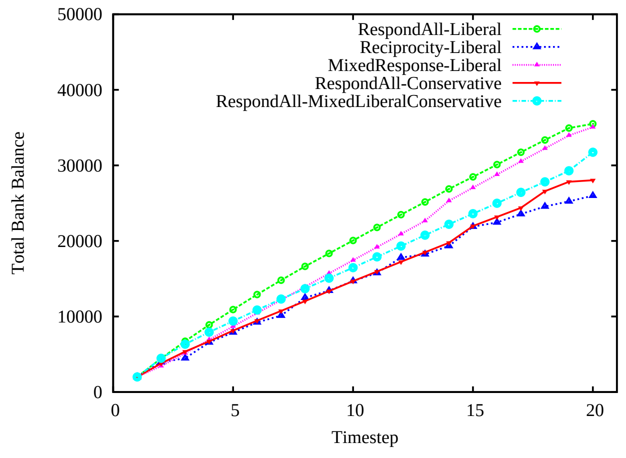

Fig. 3. Bank Balance

Reciprocity cases are trivial where in an environment with all agents using reciprocity, none of the agents reply each other (at least one starter response has to be triggered). With the *Respond All* strategy, we have the opposite effect, since agents respond to all queries the percentage of replies is fixed at 100%. For the *MixedResponse* case, the number of responses made out of all incoming requests increase in time which can be explained by the existence of agents using *RespondAll* strategy. At every timestep, there is an increasing probability that the *Response List* of the agents using *Reciprocity* grow wider, which leads to getting more responses from those. Note that, the agents in the system are telling the truth. When the agents deliberately hide information from each other, we expect the percentage of responses to drop for the *MixedResponse* strategy.

Finally, we examine how the combination of these strategies affect the total bank balance of the populations. Figure 3 plots *Total Bank Balance* distribution throughout the simulation. This plot is very much related with the *Appraisal Error* as fewer errors in evaluations lead to more customer share in the long run. When agents engage in more transactions by responding all requests and requesting reputation information (*RespondAll-Liberal*) they achieve the highest *Total Bank Balance*. On the other hand, when agents employ *Reciprocity-Liberal* strategy combination, they ask others for reputation information but do not themselves provide any answers. Hence, the society cannot share each others' findings. When this is the case, the society ends up with the smallest *Total Bank Balance*.

V. Related Work

Several methods and the ways how they can be combined into agent models are proposed in this paper after inspecting various trust models, comparing their advantages and disadvantages and determining their compatibility and integration to the ART testbed domain. These models vary from defining the social dimension and importance of trust [4] to the more applicable methods based on reputation in multi-agent systems [14]. The following describe some of these models.

While constructing the proposed model in this paper, some supplementary concepts are considered as well as taking into account existing trust models directly. One of these concepts is the idea of a PageRank system [3], which is used in search engines for ranking of Web sites. The idea can be integrated into the ART domain to keep the reputation of agents as PageRank values which will help manipulate the agent selection algorithm. Another approach is the concept of unsupervised learning which in some ways can be appropriate to utilize in the ART domain as the duration of the simulation runs is unknown. This prevents the agent from spending much time on training and modeling the environment.

Some existing trust models are not directly applicable to ART game. In FIRE [9], four types of trust and reputation are presented and the innovation comes from the concept of certified reputations in which the agent itself offers reputation information about its past behavior. However, such interactions are not expected in the ART game.

The SPORAS trust model treats different types of trust ratings (individual and social trust) equally as opposed to FIRE and gives more weight to more recent ratings (rather than experiences from the past) [15]. It also takes into account the deviation of rating values concerning an agent, which dramatically affects the reputation of the agent.

REGRET puts more emphasis on the social dimension of trust and presents the idea of an ontological structure to model the reputation of agents [10]. According to the model, reputation is a combination of different pieces of information and the overall value is computed after assigning weights to the individual components. This idea can be integrated into the ART domain up to some point as the reputation of an agent may be constructed from the ratings of the agent in different eras via certain weights which will be determined by the concerned agent, i.e., the agent that wants to learn about the other's reputation.

The existence of reciprocal agents in a multi-agent system brings some well-known challenges into the competition like zero-cost identity, free-riding, and collusion [1]. However, this is not the case for the ART domain. As it is not an open system, agents remain in the environment until the end of the simulation with the same identity. Also, free-riding is not an issue, because every request has a cost to be paid. Collusion in the ART domain is not allowed by the nature of the competition, but it can be created in an experimental environment where a group of agents knows each others' identity and behave accordingly throughout the game.

Collaboration in multi-agent systems is another interesting concept as experimented in the SPIRE system [8]. 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. The collaborative side of the system weighs more than the competitiveness encouraged in the ART testbed simulation environment.

VI. Conclusions and Future Work

In this paper, we've tried to implement well-known trust strategies and integrate them into the ART testbed simulation environment. By developing some useful metrics, we measured how successful they perform among each other. While constructing the agent societies that the simulations run on, we restricted the agents to respond honestly (as far as they're able to) to all queries.

In our future research, we're aiming to consider cases where the society consists of both honest, dishonest and totally random agents, in order to see how the parameters are affected. We're also eager to create environments within the ART testbed simulation where we can experiment the phenomenon of collusion in multi-agent systems and recommend suitable deterrences to the problem.

Acknowledgement

This research is supported by Boğaziçi University Research Fund under grant BAP05A104. We are indebted to the ART Development Team, especially Karen Fullam, Jordi Sabater and Tomas Klos for helping us with the testbed software and the anonymous referees for their comments.

References

[1] D. Banerjee, S. Saha, S. Sen, and P. Dasgupta. Reciprocal resource sharing in P2P environments. In *AAMAS*, pages 853–859, 2005.

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

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] C. Castelfranchi and R. Falcone. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. pages 72–79, Paris, France, 1998. Proceedings of Third International Conference on Multi-Agent Systems.

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

[6] K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. A specification of the agent reputation and trust (ART) testbed: experimentation and competition for trust in agent societies. In *AAMAS*, pages 512–518, 2005.

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

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

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

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

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

[12] P. Yolum and M. P. Singh. Service graphs for building trust. In *CoopIS/DOA/ODBASE (1)*, pages 509–525, 2004.

[13] P. Yolum and M. P. Singh. Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man, and Cybernetics. Part A*, 35(3):396–407, 2005.

[14] B. Yu and M. P. Singh. Detecting deception in reputation management. In *Proceedings of AAMAS '03*, July 2003.

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