Composing Trust: An Effective Trust Model for Multiagent Teamwork*

Feyza Merve Hafızoğlu[1] and Pınar Yolum[2]

[1] Department of Systems & Control Engineering
[2] Department of Computer Engineering
Boğaziçi University, TR-34342, Bebek, İstanbul, Turkey
{feyza.isik,pinar.yolum}@boun.edu.tr

Abstract. Composed services consist of interacting services. Generally each service in a composed service is brought out by a different service provider. The quality of the composed service depends not only on the individual capabilities of the providers but also on how well they work together. Existing trust models are geared towards identifying single services rather composed services. However, in many settings it is important to find a group of service providers that can be trusted for a composed service. To address this, we propose a trust model that captures how trustworthy a group of service providers is for a particular composed service. The approach is based on capturing relations between services. Our proposed approach is tested on an adaptation of ART Testbed. We compare our proposed model with an existing approach in the literature and show that capturing relations between services pays off in finding useful groups of service providers.

1 Introduction

In dynamic open systems, many agents interact with each other to achieve their goals. In such environments, a self-interested agent selects the most trusted and suitable partners to interact with from a pool of agents whose behaviors are not known. Ideally, an agent should interact with the agent who most probably fulfills the expectations of the requester agent. Trust model consists of opinions of an agent about other agents; it's formed by using its own experience with the related agent and other agents' opinions about the related agent. Each agent builds its own trust model and uses this model to decide on whom to trust.

Whereas there are various approaches for finding individual trustworthy services, most real life needs are satisfied by composite services, rather than single services. Such composite services are realized by groups of service providers. A service in a composite service cannot be performed without considering other

* This research has been partially supported by Boğaziçi University Research Fund under grant BAP09A106P and the Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073. We thank the anonymous reviewers for helpful comments.

services, because the services are dependent on each other. Consider a house owner that wants her house to be repaired in a short time. The repairing is a composite service that needs to be carried out by an electrician, a painter, and a plumber. These service providers have to work at the same time in the house due to the limited time of the housekeeper. It is obvious that the manner in which one provider works will affect the service of another provider in the group. Hence, the house owner needs to find a group of service providers that she can trust for the composed service, since even if the service providers work well individually, they may not have the same performance when they work together.

When an agent needs a team of agents rather than a single agent to fulfill its request, the agent will consider the trust to the team instead of the trust to each individual agent in this team. In this case, the agent needs a trust model to evaluate the trustworthiness of possible teams. Whereas a vast literature exists for modeling trustworthiness of individual service providers, there is not much work done in modeling groups of providers for a given composed service.

Developing a group model for trust requires the following questions to be answered. When a group of agents attempt to carry out a composite service but is unsuccessful, how can the blame be distributed among the participants? How can an unsuccessful group of agents be modified so that they become successful? Can addition of agents to a successful group be risky? We study these problems by developing a group model of trust.

Our model is based on the idea of the service graphs [2] to build composed services. Service graphs are used to represent the relationship between services, which have one or more common subtask. Intuitively, service graphs are helpful in reasoning about services that are related to each other. If a service provider is a good candidate in a service, it might be a good candidate in a related service as well. Further, to capture the trustworthiness of a group of agents for a particular composed service, a group trust model is introduced. The group trust model measures how a group of agents would be useful in performing a particular composed service. This group trust model is updated as the same group is used for the same composed service.

The remainder of this paper is organized as follows. First, our proposed group trust model and service graph model are presented in detail. Then, we give a brief information of ART Testbed and explain the adaptation of ART to handle composed services. Next, we give a step by step description of the strategies. Next, experimental results are provided. Last, we compare our approach to existing work in the literature.

2 Group Trust

When a group of providers are sought for a composed service, considering the providers' individual behavior is not enough because in carrying out the composed service providers will be participating in teamwork. Teamwork trust problem emerges with the following issue: the behavior of the agent in teamwork environment may differ from its behavior in single service environment. In team-

work, the behavior of the agent depends on the teamwork, other agents in the group, and so on.

One can naively think that whenever a group of agents are required for teamwork, for each subtask, we may select the most trusted agent for the corresponding service in the environment. But there is no guarantee that an agent has the same performance when it is taking place in teamwork and when it is acting individually.

Being in a collaboration may have a positive or negative influence on the performance of agents. For example an agent who is a successful painter works very well individually. However, it has a worse performance when it participates teamwork as a painter. As a conclusion, the idea behind the teamwork is totally different from a single service and it is more complicated in the sense of both representation and its reasoning. Hence, considering only the participant's individual trustworthiness is not going to be enough to understand the trustworthiness of a team.

Possible tendencies of agents who participate teamwork can be listed as the following:

- *Ideal Behavior:* The agent performs well both individually and in teamwork.
- *Group Antipathy:* The agent may dislike being in a team. Thus, whenever the agent participates in teamwork, its performance will be low.
- *Group Motivation:* The agent performs well in the teamwork even if it does not perform well individually, i.e. other agents may help the agent. Being in a group has a positive influence on the agent's behavior.
- *Colleague Effect:* The agent's behavior changes based on the other agents in the team. The agent may perform better with some agents but not with others.
- *Teamwork Effect:* The agent may have a bad performance due to the task characteristics. For example, a painter may work well with another plumber but not so well with an electrician.
- *Familiarity Effect:* The agents in the group improve their performance as the number of cooperations increase.

2.1 Representation

There are two important representations that we use to enable group trust. First representation is used to keep the instances of the teamwork, in other words the previous teamwork experiences of the agent are stored. The latter one is a service graph and used to represent the relation between the services of teamwork.

In group trust model, each agent classifies its experiences with respect to the requested composed service and the agents those participate in carrying it out. Each experience instance in composed trust model consists of a subtask list and the corresponding agents who are assigned these subtasks: that is, a list of agent-subtask pairs.

A group trust model has the information about the agents that exist in the group and the subtasks which agents in the group are assigned requested about.

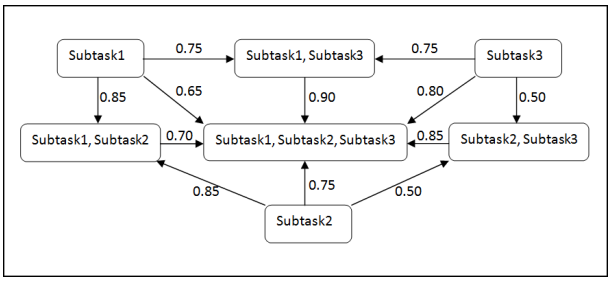

Fig. 1. Service Graph for Services including Subtask1, Subtask2, and Subtask3

Each model has an expertise weight, which is updated after each experience of the group for the same composed service, and the number of interactions which determines the accuracy of the expertise weight. As the number of interactions increase, the expertise weight would be more accurate as mentioned in several models in the literature. The expertise weight of a group trust model instance has a default value of 0.5 when it's created, and it's updated based on the overall performance of the group by using geometric update method.

2.2 Service Graphs

In order to characterize the tasks in teamwork, a graph-based representation of services [2] is used. A service graph is a weighted, directed graph including nodes for service and the edges for transitions between services. The weights on the edges show how likely providers that are successful in a source node are likely to be successful in the target node. Each composite service is a node in the graph. By using this relationship between different composite services, the agent composes a new group of agents for a given service.

In the service graph, only the nodes or services those have at least one common subtask are connected to each other. Otherwise, there is no relationship between two nodes. There is a weight related to each edge in the graph. Default weight, which has a value of 0.5, is assigned to each edge when it is created. The weights capture the likelihood of a group in the source node to be successful in the target node.

When it's the first time the agent is assigned a service which consists of Subtasks 1, 2, and 3, agent adds the directed edges to the service graph for this service, which becomes the destination node and the other services with combinations of the subtask set of this service become source nodes. In this case, assigned service becomes target node, and the service that contains Subtask1, the service that contains Subtask2, the service that contains Subtask3, the service that contains Subtasks 1 and 2, the service that contains Subtasks 1 and 3, and the service that contains Subtasks 2 and 3 are the source nodes. Service graph including these services and weighted edges is shown in Figure 1.

Using a service graph enables to look from the aspect of agent teamwork. As we asserted our motivation before, the agent may not behave the same way

when it is acting independently or when it is taking place in a team. The most important information for an agent, who is assigned a service including more than one subtask, is the edge weights of the graph. Each edge of the graph has an information of weight and number of usages. These edges of the target service are used to establish a group when this target service is assigned to the agent. The agent uses a set of edges, where the union of subtasks of these edges' source services is equal to the subtasks of the target service. Using edges means that the agent selects more useful edges with higher weights and composes groups which are experienced before for the source services of the selected edges to form a group for the currently assigned target service. The weights of edges which are used to build a composed service are updated based on the performance of the actual group whenever it's experienced, and the number of usages is increased by one for each selected edge.

Example 1 Let's say that an agent is assigned a composite service which consists of Subtasks 1, 2, and 3 and decides to use service graph to build a team. Related (services) nodes for this assigned service are all the subgroups of this composite service. The agent can follow the edges to obtain a group for the assigned service. Specifically, it selects a set of useful edges from the edges of the service graph, considering the weights on the edges. Average weights of three possible compositions are calculated by using the edge weights in Figure 1 as the following: (1) the composition of the service that contains Subtasks 1 and 2, and the service that contains Subtask3 is (0.70 * 2 + 0.80) / 3 = 0.73, (2) the composition of the service that contains Subtasks 1 and 3, and the service that contains Subtask2 is (0.90 * 2 + 0.75) / 3 = 0.85, (3) the composition of the service that contains Subtasks 2 and 3, and the service that contains Subtask1 is (0.85 * 2 + 0.65) / 3 = 0.78. Most probably, the agent would obtain a better service by composing the service that contains Subtasks 1 and 3, and the service that contains Subtask2 because this composition has the highest average weight which reflects the relationship between this composition and the required service. Actually, the agent composes the groups of agents experienced before for these selected services.

Example 2 The above is an example for composing nodes of the service graph to obtain requested composite service. Another alternative is separating nodes to obtain the requested service. Let's say an agent is assigned a service that contains Subtasks 1, 2, and 3. The agent may use previous experiences that belong to the service that contains Subtasks 1, 2, 3, and 4. This is done by removing the agent from the group that successfully performs the service with Subtasks 1, 2, 3, and 4. This is an example for the *separation of nodes*. In this study, we only use *composing nodes* method yet.

If the agent decides to use a service graph, it should carry out a two step procedure to determine the group of service providers. The agent begins with finding the most appropriate edges for the assigned composite service by the aid of composing nodes technique. The union of the subtasks of the selected nodes

should be equal to the subtasks of the service. Actually, edge is the transition between the different service types and the current service. Second step is the finding of suitable group instance for the selected nodes. If the selected services (nodes) are the service that contains Subtasks 1 and 3, and the service that contains Subtasks 6 and 8, then the agent remembers its experiences for these two services. When it finds successful experiences, it assigns subtasks of these services to corresponding agents with respect to previous experience.

Note that the service graph does not give any information about the expertise of individual agents or a group of agents. It just captures how likely groups that perform certain services are likely to perform other services.

2.3 Non-cooperative Behavior

In this paper, we consider agents whose behavior may change based on the particular composed service that it's taking part of. That is, even if an agent performs well independently, in certain types of composed services, its performance may be bad. Each agent has a finite list of composed services in which it is going to be non-cooperative. This is called the *noncooperativeness list* and may be different for each agent. *Noncooperativeness level* shows the extent of cooperation. If the noncooperativeness level of the agent is 0, then the agent cooperates with all assigned subtasks. If it is 1, then the agent never cooperates in any of the possible collaborations.

3 Experimental Setting

Proposed solution to teamwork trust problem is implemented within the Agent Reputation and Trust (ART) Testbed [1] which is a popular simulation platform to compare different trust strategies in the context of single tasks.

3.1 The ART Testbed

The ART Testbed domain consists of agents that appraise paintings. Each painting belongs to an era, and agents have varying expertise in these eras. An appraiser's expertise is its ability to generate an opinion about the value of a painting. At each timestep, agents are assigned a number of paintings to appraise. All agents are assigned the same number of paintings at the beginning of the game. Agents are paid a fee for each appraised painting. The goal of agents is to maximize their *bank balance* by minimizing appraisal error. As the accuracy of the appraisals of the assigned paintings increases, agents have more clients and consequently earn more money.

If an agent has low expertise value in an era in which assigned painting belongs to, the agent asks opinions of other agents to come up with more accurate appraisals. Intuitively, the agent should query the agents who have higher expertise values of corresponding era to increase the profit. However, the expertise values of other agents are not directly known by the agent. So each agent tries to

model and learn the behavior of other agents for existing eras by using its past experiences with other agents and reputation information of an agent, which is asked from other agents. Agents may ask certainty of agents, which is an information about the expertise of an agent about a particular era to decide from which agents to ask opinion. Agents are also paid a fixed fee for each opinion and reputation they provide, so agents may also increase their profit by selling opinions and reputation information to other agents. During the game, the correctness of the replies are not guaranteed, and the strategies of other agents are not known due to the heterogeneity of agents. In fact, it is quite likely that agents provide incorrect information in order to decrease the requester's client base in such a competition environment.

ART simulator is responsible for assigning paintings to the appraisal agents for evaluations, receiving the agents' answers about the painting, calculating the true value of the paintings, and informing the agents about this information. The agents, then, can calculate their errors and act accordingly.

3.2 Frost

The basic trust model includes the answers of the question: how can I trust agent x for era y? We use this basic model to support single tasks. If there are n agents and m eras in the environment, there exist $n*m$ model instances for each agent-era pair. A model instance contains the information of the past interactions with the agent for a particular era such as the expertise weight, number of interactions, and so on. At the initialization of the agent, a default model which has an expertise weight of value 0.5 is created for each agent-era pair. When an agent is assigned a painting belongs to one era and requests the opinion of an agent, the corresponding model instance for this agent-era pair is updated. The expertise weight is increased or decreased based on the appraisal error of the painting and the number of interactions is increased by one.

Geometric update is used for expertise weights as shown in Figure 2:

1: $error = |appraisedValue - trueValue|/trueValue$
2: **if** $(1.0 - error) < expertise$ **then**
3: $\quad expertise = penalty * (1 - error) + (1 - penalty) * expertise$
4: **else**
5: $\quad expertise = reward * (1 - error) + (1 - reward) * expertise$
6: **end if**

Fig. 2. Updating the Expertise Values

Reward and penalty weights can be monitored to obtain the most suitable strategy. In this study, we prefer to use high penalty weight and low reward weight. If an agent obtains a better result than its current expertise value which is defined in the basic trust model, then we increase its expertise weight by

a small amount. However, if it performs worse, we decrease its expertise by a higher amount to penalize this agent.

3.3 Modified ART

ART is originally designed and developed for comparing and evaluating different single dimensional trust models, since agents are expected to provide a single service, i.e., evaluating a single painting that belongs to a single era. However, to evaluate trust models for composed services, the environment needs to be modified so that an agent will be requested to evaluate a composed service. To facilitate this, we first modified ART Testbed simulator to provide teamwork environment and then developed an agent that can participate in this new platform.

Simulator Side Modification: The fundamental task in ART domain is appraising the value of a painting, which belongs to exactly one era. This can be viewed as a single service. However, a composed service consists of several services that act in combination and each service is fulfilled by a single provider. In order to achieve this, we extend the framework so that a composed service is represented as a painting that belongs to one or more eras. The true cost of a painting is determined at the creation time of the painting. Each era of the painting has an effect on the painting's cost. The effect of each era is represented with normalized weights whose sum is equal to 1.0. In order to evaluate a painting, opinions related to all the eras to which the painting belongs need to be collected. That is, the agent asks the opinion of exactly one agent for each era of the assigned painting, and then agents, whose opinion are requested for any era of the painting, become a group and offer a composed service. Each agent in the group appraises a value for the corresponding era part of the painting.

In the modified ART, the creation process of a painting has three fundamental steps: determining the number of eras, determining the eras, and determining the weights of eras, respectively. Our assumption is that a painting may belong to at least one and at most four eras (out of 10 eras) and the actual number is picked randomly. Finally, the weights of the eras, whose sum is 1.0, are generated.

In ART domain, each composed service (i.e., painting) is characterized by the weights of its subtasks. Hence, the weights of corresponding eras for two different paintings, which belong to the same combination of eras, should be similar to each other. According to this characterization, the first time a combination of eras is created, the weights for this combination are determined randomly and registered in the *weights table*, which stores the weights of eras for each combination has been so far. After registration, whenever a painting belonging to an existing combination of eras is created, the weights of eras for this combination are taken from the *weights table* and slightly perturbed (between 0 and 0.05). This perturbation is decided randomly for the weight of each era. Note that these weights are only known by the simulator. That is, agents are not aware of the weights of the eras for a painting.

These weights have important roles in calculating true values of paintings, appraised values, and thus error rates in the following ways. After determining the weights of a painting at the creation time, the simulator generates a true value for each era part of the painting, and the overall true value of the painting becomes the weighted sum of these partial true values, by using the weights of eras. Remember that each agent in the group appraises a value for the corresponding era part of the painting. Overall appraised value is the weighted sum of the appraised values of agents in the group. Appraisal error of each agent in the group is calculated for the corresponding era by using the appraised value of this era part of the painting and the true value of this era part of the painting. Overall appraisal error is again the weighted sum of these partial appraisal errors.

Example 3 A painting is created and randomly decided to have three eras: Era1, Era3 and Era7. Let's say this is the first time a painting, which belongs to Era1, Era3 and Era7, is created. In this case, normalized weights of these eras are generated randomly as the following: 0.45, 0.25, and 0.30, for Era1, Era3, and Era7 respectively. These weight-era pairs are registered for the combination of Era1, Era3 and Era7 in the *weights table*. The next step is the generation of true values: true values of each era part of the painting are generated with the same formula with the formula used to generate true value of a painting in the original ART Testbed. Let's say true values for the era parts become: 1000, 2000, and 1500 for Era1, Era3, and Era7 respectively. The overall true value is the weighted sum of partial true values:
$(0.45 * 1000) + (0.25 * 2000) + (0.30 * 1500) = 1400$

Agents are only informed about the overall appraisal error, overall true value, and overall appraised value of the assigned painting. That is, an agent learns the error it made in evaluating a painting, but it does not learn the individual errors in different era(s). This corresponds to the case that a consumer does not like a composed service but does not give feedback about which individual parts have failed. Whereas in original ART Testbed, agents try to find the most suitable provider for each painting, in modified ART Testbed, agents need to find the most suitable group of agents for each painting.

Example 4 Continuing from the above example, an agent requests opinions of a group of agents such that one agent will be responsible from one era of the painting. Let's say, appraised values of the agents in the group are 1100, 1980, and 1560 for Era1, Era3, and Era7 respectively. Then the overall relative appraisal error is weighted sum of partial appraisal errors, and calculated by weights, partial true values, and partial appraised values:
$(0.45 * ((1100 - 1000)/1000)) + (0.25 * ((2000 - 1980)/2000)) + (0.30 * ((1560 - 1500)/1500)) = 0.0545$

After some time, if the simulator creates a painting with the same era group Era1, Era3, Era7, then it considers the weights of this combination from the weights table and determines the new weights for the current painting: 0.40,

0.27, and 0.33 for Era1, Era3 and Era7. The weight of Era1 is decreased by 0.05, the weight of Era3 is increased by 0.02, and the last weight is also increased by 0.03 randomly one by one. The sum of these weights is again 1 and the absolute rate of difference of the weights of an era in different paintings with the same era group is at most 0.05. Simulator behaves this way whenever a painting, which is belong to Era1, Era3 and Era7, is created.

Agent Side Modification: Agents are assigned paintings which belong to one or more eras in modified ART Testbed. Appraising a value for any era part of the painting can be thought as a subtask in a composed service. Agents ask the opinion of one agent for each era of assigned paintings in modified ART. The total number of opinions asked for a painting is equal to the number of eras which this painting is belong to.

In the original testbed, when requesting an opinion, an agent asks the opinion of other agents by sending the era name for which an opinion is requested. In modified testbed, when requesting an opinion, in addition to the era name for which an opinion is requested, the agent is informed that it would be involved in a composed service consisting of a particular era group with a particular group of agents. All information is included in the opinion request message of the opinion protocol.

In ART domain, *noncooperativeness property* is defined as the following: when the agent is requested its opinion about an era of the painting which is in the noncooperativeness list of this agent, then this agent will send the worst opinion creation order to mimic the fact that the agent cannot do well in this composed service. Actually, non-cooperative behavior emerges due to *teamwork effect* mentioned before.

3.4 Opinion Request Strategy

Opinion request strategy of the agent for a particular painting depends on the number of eras of the painting. If the number of eras is one, agent uses basic trust model to find the most suitable agent to request opinion. Otherwise, it uses the group trust tools to decide the group of agents.

Remember that the basic trust model includes all agent-era pairs. When the agent is assigned a painting, it selects the most trusted agent for the era of this painting. The most trusted is measured with a weighted sum of certainty of agents and the expertise level of agents for the era. The weights of the certainty and expertise values changes during the game and their sum is 1.0. Actually, these weights are adaptable parameters and their value depends on the current timestep. The weight of the expertise is increased by a small amount in certain timesteps, while the weight of certainty decreases by the same amount directly. Since, the importance of the agent's own experiences increases as the number of experiences increases, the agent with the highest combination of certainty and expertise value is selected to request opinion.

Finding a trusted group of agents for a painting which belongs to more than one era is more complicated. There are four alternative strategies: (1) using the

exact experience which is higher than a certain threshold from the group trust model instances, (2) using a set of experienced edges of the service graph if edges with high weights exist in the graph, (3) using inexperienced edges whose group instances have an expertise value higher than a certain threshold, and the last alternative is (4) using the basic trust model. The agent pursues these alternatives one by one in this order. Once it finds a suitable group of agents in any of the steps, it finalizes the opinion request procedure. In order an agent to use the first and second strategies, it should have successful past experiences with the corresponding painting. We explain these strategies next.

First strategy is similar to the strategy used for paintings with one era, but in this case there is no certainty information of a group. Instead, the agent uses the expertise values of the group trust models with the same painting type. If there is an appropriate experience, which is higher than a certain threshold, for exactly the same painting type, the same group can be used.

Second strategy uses the service graph information, namely the weights of the edges. If there exists a set of edges whose average is higher than a certain threshold value, the agent selects these edges, where the union of eras of these edges' source nodes is equal to the eras of the assigned painting. Then, the agent looks at its group trust models to find an appropriate group instance for each selected edges' source node. Finding suitable group instances is similar to first strategy.

Third strategy uses the service graph information, namely the edges of the graph rather than the weights of the edges. The agent finds all possible edge sets by using these edges to obtain a final group for the current painting, and then finds the best groups instances for each possible edge set by looking at its group trust model instances, that is its past experiences. The edge set, which has the highest average expertise based on the expertise weights of the group instances, is selected. If the selected highest expertise is higher than a certain threshold value, these group instances of the selected edges' source nodes are used.

If the agent cannot find a group of agents at the end of first three strategies, it uses the last strategy. In this case, it selects agents one by one for each era of the painting by using basic trust models without considering any threshold value, since no alternative strategy remains in this step. This strategy works well with the agents those have *ideal behavior* mentioned before.

Note that if the agent uses second or third strategies in the current timestep, it keeps the selected edges for the corresponding painting to update the edge weights of service graph at the beginning of the next step, where the appraisal errors are received from the simulator. Additionally, the agent keeps the group and painting pairs to update the group trust model at the beginning of the next step.

3.5 Agent Strategies

We explain the representation of three models, namely basic trust model, group trust model, service graph model, geometric update procedure, non-cooperative

behavior of agents, and opinion request strategies so far. In this section, we present how the game evolves based on the ART Testbed messages:

1. *prepareReputationRequests():* The agent doesn't use reputation information. Hence, it doesn't send any reputation request message. Opinion replies are received in this step, the agent updates basic trust models, group trust models and the service graph based on the replies.
2. *prepareReputationAcceptsandDeclines():* The agent accepts all reputation requests.
3. *prepareReputationReplies():* The agent generates reply messages for the agents that request reputation information based on how the agent modeled the agent whose reputation information is being asked.
4. *prepareCertaintyRequests():* We set a high value for the maximum number of certainty messages. Though, the agent sends a certainty request for each era of each assigned painting. The agent prefers to ask certainty especially from agents whose certainty value is unknown about the related era.
5. *prepareCertaintyReplies():* The agent replies the certainty messages according to its real expertise value of the related era.
6. *prepareOpinionRequests():* The agent uses basic trust model for paintings with one era. On the other hand, the agent uses group trust models and service graph for paintings with more than one era.
7. *prepareOpinionCreationOrders():* Noncooperativeness property of the agent emerges in this step. When the agent's opinion is asked about an era of a painting which exists in its noncooperativeness list, then agent order an opinion value of 1 from the simulator via sending a message of type OpinionOrderMsg. If the non-cooperative list doesn't contain this painting type, then the agent orders an opinion value of 10.
8. *prepareOpinionProviderWeights():* Weights don't have any effect in group model setting, since the agent asks one opinion from only one agent for an era of a painting. This is the only opinion that effects the appraisal of the painting about particular era. Formally, the agent sends 1 as a weight to the simulator via WeightMsg.
9. *prepareOpinionReplies():* The agent sends messages of type OpinionReplyMsg by finding the appropriate opinions that are already sent to the simulator.

4 Experimental Results

So far, we have explained how an agent can decide on the trustworthiness of teams using group trust model. Now, through experiments, we evaluate how well such an agent can indeed model a team. To understand this, we compare the performance of the group trust model with the basic trust model. Our agents are named as GMA (Group Modeling Agents) and the other agents are called SMA (Single Modeling Agents), respectively.

SMA agents are modeled by excluding the group trust model and service graph of the GMA agents. SMA agents use exactly the same basic trust model,

the same opinion request strategy for paintings with one era, same adaptable parameters, same reward and penalty weights in the update procedure with the GMA agents. The only difference is that when requesting an opinion, SMA agents select agents to request opinion one by one for each era of the painting by using basic trust model.

In addition to SMA agent, we use honest agents, which randomly select the agents to request opinion from. Our experimental setup contains 9 agents: 3 GMA agents, 3 SMA agents and 3 honest agents in the environment. All agents have noncooperativeness behavior and the same noncooperativeness level is used for these agents. The game continues 100 timesteps. We have repeated our experiments with larger populations and our results are similar.

Threshold parameter values used in strategy (1), (2) and (3) are adapted by the agents during the game based on the current timestep. Different threshold parameters are used for these strategies. The expertise weights of group trust models and the weights of the service graph are expected to increase for better groups during the game. Thus, an agent increases the threshold values by a small amount in certain periods.

Example 5 For group trust models, threshold values are adapted as shown in Figure 3.

```
1: if currentTimestep < 10 then
2:     threshold = 0.50
3: else if currentTimestep < 20 then
4:     threshold = 0.55
5: else if currentTimestep < 30 then
6:     threshold = 0.60
7: else if currentTimestep < 40 then
8:     threshold = 0.65
9: else if currentTimestep < 50 then
10:    threshold = 0.70
11: else if currentTimestep < 60 then
12:    threshold = 0.75
13: else
14:    threshold = 0.80
15: end if
```

Fig. 3. Adapting the Threshold Values

We compare GMA and SMA agents based on their bank balances and the behavior of their final bank balances with respect to changing noncooperativeness level. GMA agents considerably outperform SMA agents by using group trust modeling tools. Note that honest agents are not represented in the graphs because they behave randomly and have no strategy.

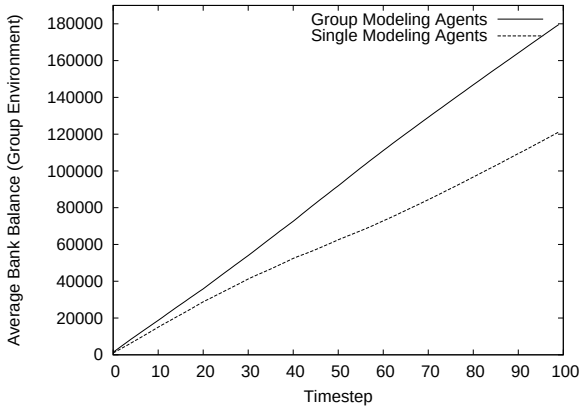

Fig. 4. Bank Balances of GMA vs. SMA

Figure 4 depicts the bank balances of GMA and SMA agents with noncooperativeness value of 0.3. This setting is run 50 times and the average value of bank balance are used in the graph. Accordingly, for each run average bank balances of GMA agents and SMA agents for each timestep are used.

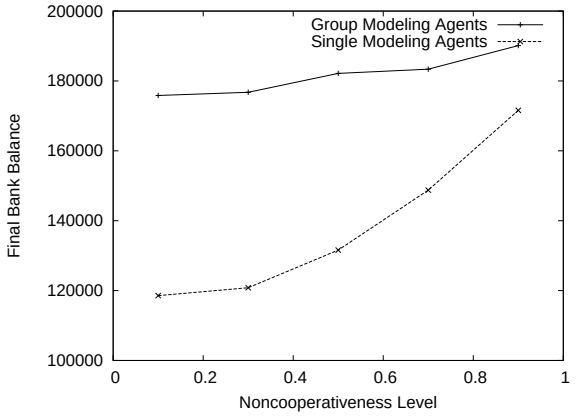

Fig. 5. Noncooperativeness level vs. Final Bank Balances

In Figure 5, final bank balances of GMA and SMA agents with respect to different noncooperativeness levels are shown. For small values of noncooperativeness level, the difference between the bank balances of GMA and SMA agents has the largest values. This difference decreases as the noncooperativeness level increases and the cooperativeness level of the agents decreases. Decreasing cooperativeness means that agents, from which opinions are requested, send opinions

properly for very limited set of paintings and they choose not to cooperate for most of the paintings in the environment. Hence, GMA agents start to misclassify groups as noncooperativeness level increases.

On the other hand, SMA agents cannot handle even the smaller noncooperativeness levels. Since they just use basic trust models, they assume that agent behave the same way for any environment and for any painting. However, since agents behave noncooperatively with different paintings, projection of their behavior to teams is not always successful. Another important result is that higher noncooperativeness levels produce higher bank balances. Remember that noncooperative behavior is paying the smallest amount to the simulator via opinion creation order in ART domain. Hence, as the noncooperativeness level increases, opinion costs (the amount paid by the opinion provider to generate the opinion) decrease, and provider agents save their money.

5 Discussion

Most approaches to trust model consider a single agent and predict its trustworthiness accordingly. However, in many real-life settings, an agent has to interact with a group of agents to receive a composed service. This paper proposes a group trust model to understand the behavior of such teams that carry out a composed service.

Barber [7] presents a trust-based mechanism for team formation problem where agents selectively pursue partners of varying trustworthiness in a market-based environment, where a job consists of multiple subtasks and agents have different skills which correspond to subtasks. A certain percentage of the agents are randomly selected as contractors at each round and they decide to continue their current job or a new job, which turns to establish teams to work on their new job, by using a greedy heuristic. Candidate members of the team have different tendencies towards completing an assigned task. Results show that an agent may utilize better by selecting less trustworthy partners with comparison to more trustworthy partners. In contrast to our group trust model, this study proposed a trust model with the aspect of the participants of teamwork, the agents are modeled individually based on the tendency to complete a subtask and considers subtasks requiring different number of rounds to complete, and maximizing the profit. The behavior of the agents in the team doesn't differ based on the team or teamwork, instead they have certain characteristics to continue or leave their current job based on maximizing their profit.

TRAVOS [5] is a probabilistic trust model that considers both trust and reputation in order to handle the possibility of inaccurate reputation information. Self-interested agents may betray the trust by not performing the requested action as required. In TRAVOS, trust is calculated using probability theory between agents considering the past interactions. Whenever there is little or no interaction with an agent, the agent uses the reputation information gathered from third parties. This study especially handles the possibility of inaccurate reputation information based on the interactions with the agent whom requests

the reputation information. However, TRAVOS does not provide a modeling mechanism to evaluate teamwork.

Another solution [6] is developed by using Bayesian approach and deals with the sequential decision making problem of agents operating in computational economies. It allows agents to incorporate different trust priors and explore optimally with respect to their beliefs when choosing potential service or information providers. The trustworthiness of the agents in the environment is uncertain. A generic Bayesian Reinforcement Learning algorithm is applied to the exploration-exploitation problem where agents decide whether to keep interacting with the same "trusted" agents or keep experimenting by trying other agents with whom they haven't had much interaction so far. This algorithm considers the expected value of perfect information of an agent's actions to take optimal sequential decisions; it's applied to the ART Testbed scenario.

The proposed solution in Blizzard [4] is an action-based approach for modeling the environment; and it is also developed in ART Testbed. Blizzard differs from traditional agent-based trust models by modeling actions of the agent and their effect on the environment instead of models all agents individually. Q-learning method which originally deals with actions and states is used by removing state mapping since there is no state info in ART. Three versions of the Blizzard is developed and compared with Frost agent which is an agent-based trust model in the evaluation part, and it dramatically outperforms the agent-based approaches during evaluations.

References

1. 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. In *The Workshop on Trust in Agent Societies at The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 50–62. (2005)
2. P. Yolum, and M. P. Singh. Service Graphs for Building Trust. In *International Conference on Cooperative Information Systems (CoopIS)*, 509–525. (2004)
3. O. Kafalı, and P. Yolum. Trust strategies for ART Testbed. In *In Ninth International Workshop on Trust in Agent Societies, AAMAS*, 43–49. (2006)
4. O. Kafalı, and P. Yolum. Action-Based Environment Modeling for Maintaining Trust. In *Eleventh International Workshop on Trust in Agent Societies, AAMAS*, 23–32. (2008)
5. W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. TRAVOS: Trust and reputation in the context of inaccurate information sources. In *JAAMAS*, 183–198. (2006)
6. W. T. L. Teacy, G. Chalkiadakis, A. Rogers and N. R. Jennings. Sequential Decision Making with Untrustworthy Service Providers. In *AAMAS*, 755–762. (2008)
7. C. L. D. Jones, K. K. Fullam, S. Barber Exploiting Untrustworthy Agents in Team Formation. In *International Conference on Intelligent Agent Technology*, 299-302. (2007)