A Trust Model for Service Selection, Using Learning Automata-Based Approach

Amir Khoshkbarchi
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
a.khoshkbarchi@aut.ac.ir

Hamid Reza Shahriari
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
shahriari@aut.ac.ir

Mehdi Amjadi
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
Mehdi.amjadi@it.uut.ac.ir

***Abstract*— Service-oriented environments comprise a number of interconnected service providers and service consumers and are filled by vast numbers of services of various functionalities with different qualities. Finding the desirable services among others is a major problem for a typical user, who can optimize its performance by utilizing services with good qualities. This problem is sometimes addressed by relying on votes and advices about qualities of services collected from other agents in the environment. However, there is no guarantee that all agents give fair advices about all services. The presence of unfair or malicious agents, who tend to misinform others about the quality of services, makes it necessary to develop methods for distinguishing fair and unfair agents from each other and providing users with reliable trust information. This should be done according to the previous behavior of agents that represents their reputation and trustworthiness. Learning automata is an abstract model capable of adapting itself with interactions from the environment in a way to perform a specific functionality in the form of optimal actions. This has been the motivation for designing specific learning automata for separating groups of users based on their previous behavior which can be extracted from their votes on services available in the environment. The user can utilize this classification for judging the trustworthiness of other users. Here we propose an improved learning automata-based trust model for partitioning fair and unfair agents in service-oriented, multi-agent environments with effective partitioning performance and reliable service selection efficiency.**

Keywords: trust, reputation, service-oriented environment, learning automata

I. INTRODUCTION

Nowadays, expansion of internet and interconnected networks of service providers and consumers has led to spreading of a vast amount of services with various functionalities and differing qualities on the web. This can be described well by the new emerging concept of service-oriented environments [1]. While services with great qualities can improve a user's performance, malicious error-prone services may decrease its performance and utilization. Thus, to achieve efficiency a user in such sophisticated environments needs to select the most optimal services available. A simple method for finding desirable services among others is to rely on the reports from other user-agents who have already tried various services and are aware of their qualities (Figure 1). The idea is to gather votes for a specific service from a set of agents in the environment and decide whether to use the service based on its average reported quality [2].

The major problem of this approach is how to trust other agents about their reports. Actually there is no guarantee that all agents report the quality of a service fairly. There exist many agents trying to misinform others about the services on the environment with various motivations and objectives. Blindly relying on advices from every unknown agent in such environments may lead to selecting unqualified services for use and losing desirable ones. Thus, deciding whether to trust a specific agent in such situations is a major problem in service oriented environments. Authors in [4] have offered three general definitions from existing research in order to point out some different aspects of trust. The first definition, refers to past encounters, and may be thought of by some as "reputation-based" trust "that Trust is a subjective expectation an agent has about another's future behavior based on the history of their encounters" [5].

The second definition [6] that have used the concept of "context" in its discussion of trust for first time, considers the "competence" of a party instead of her abilities: "Trust is the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context." The last definition, in [7], refers to actions: "Trust of a party A to party B for a service X is the measureable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X)."

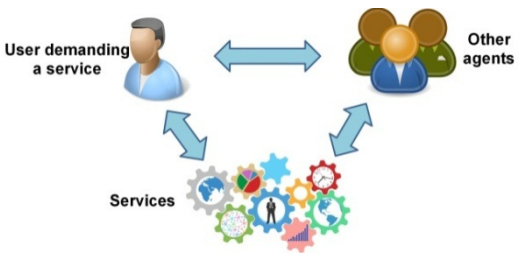

Figure 1. service selection based on reports from other agents

978-1-4799-5359-2/14/$31.00 ©2014 IEEE

Reputation systems help individuals in an environment to decide on a specific person's trustworthiness in undeterministic situations and with tolerance to probable negative results. This may lead to correction of users' behaviors, detection of fraudulent users, and improvement of the quality of user interactions [3].

Many different trust models have been proposed as methods for optimal service selection in service oriented environments [12]. The aim of such models is providing a user with methods for making decisions about services based on the behavior of other agents in the environment, whose degrees of trustworthiness is unknown. The user should be able to distinguish deceptive agents, who actively give the wrong answers about quality of services, from other fair agents. The great degree of sophistication in multi-agent, dynamic environments motivates the use of efficient learning methods in order to cope with the changes in the environmental situations. Learning automata is a learning model which could be used for developing methods for separating deceptive agents from others in stochastic dynamic environments.

Here we propose a LA-based method for partitioning fair and unfair agents. The automaton is organized in a way to be able to separate fair agents from unfair one and distribute them on a line according to their degree of fairness. The separation is based on the comparison of the votes given by agents on each queried service. The user then utilizes this partitioning for deciding on which group of agents to trust about the quality of services.

The rest of the paper is organized as follows: first, the theoretic backgrounds of LA-based Trust Models are demonstrated briefly. Then in the third section the new model for partitioning fair/unfair agents and evaluating their votes with a learning automaton is described in details, and then the performance of the new method is compared to that of another LA-based scheme. Finally, some concluding notes are discussed in the last section.

II. BACHGROUNG AND RELATED WORKS

The proposed method has two distinct aspects: the concept of trust in service-oriented environments as the context and learning automata as the method.

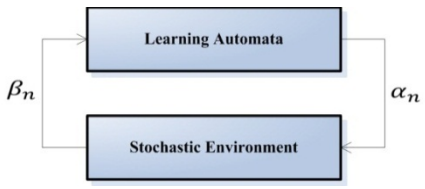

Figure 2. the relation between a learning automata and stochastic environment

A. Trust Models in service oriented environments

An algorithm for selecting the best service provider is proposed in [8]. The selection is based on provider ratings gathered from other uses. The goal of the algorithm is to select a subset of providers, with the least probabilistic guarantee for their trustworthiness as an accurate estimation of their reputation. The algorithm is sensitive to the ratio of deceptive users. In [9] proposes a heuristic method for reducing the complexity of calculating reputation values in a trust network, which is evaluated with data extracted from websites that use trust values. Authors in [12] proposed a modification to WMA algorithm for combining feedbacks from multiple trust evaluation sources. The main weakness of this method is its low convergence speed. Another algorithm based on Bayesian theory is proposed in [10] for filtering unfair votes using repetitive filters method, which is a machine learning approach. This scheme utilizes a reputation-based system proposed in [11]. demonstrates a method for distinguishing fair feedback from unfair ones. It takes an answer to be fair if it is equivalent to the majority of answers, and thus it is accurate only when fair users are the majority[13]. Authors in [14] propose an scheme for assessment of trustworthiness of users in peer-to-peer networks. Their basic assumption is that users in such networks are able of inferring the trustworthiness of other users. This is done by comparing a user's pre-assumption about its own trustworthiness to reports from other users. Although this approach is a feedback evaluation scheme, it's based on the fact that users act both as service providers and service consumers. In other words there is no difference between providers and consumers.

B. Learning Automata

Learning automata is a theoretical model for learning in stochastic environments which works with performing a limited number of actions. Each action is evaluated by the environment and responded by an answer which is then used by the automaton to update its state and decide on the next action to perform [15] [16]. Figure 2 illustrates the relation between a learning automata and the environment.

The environment is usually shown by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is a set of possible inputs to the environment which is actually the set of actions of the automata, $\beta = \{\beta_1, \beta_2, ..., \beta_m\}$ is the set of outputs of the environment (reactions to the automata), and $c = \{c_1, c_2, ..., c_r\}$ is the set of penalty probabilities. The environment is of type P, if the set β consists of only two members (0 and 1). In the P model, $\beta_1 = 1$ is known as penalty and $\beta_2 = 0$ as the reward. We consider the environment in this paper to be of type P. The environment with β of discrete values between 0 and 1 is of type Q, and with β consisting of any real value in [0,1] is of type S. C_i is the probability that the action α_i have not a desired consequence. In a static environment the values of C_i remain unchanged whereas in dynamic environment these values can vary over time. The concept of learning automata is divided into two broad categories, fixed-structure LA and dynamic-structure LA [15] [16] [17]. The former is the basis for the work in this paper.

A learning automaton is represented by a quintuple $LA \equiv \{\alpha, \beta, F, G, \varphi\}$ where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the set of possible actions of automaton, $\beta = \{\beta_1, \beta_2, ..., \beta_m\}$ is the set of inputs of automaton, F is the functions that maps the current state of the automaton to its future state based on the response

from the environment: $F \equiv \varphi \times \beta \to \varphi$, G is the output function which maps current state to an action (as the output): $G \equiv \varphi \to \alpha$, and $\varphi = \{\varphi_1, \varphi_2, \varphi_3, ..., \varphi_N\}$ is the set of automaton's internal states.

C. Trust Models based on Learning Automata

Yazidi et. al. describe in [2] a trust model for service oriented environments, based on object partitioning learning automata. They use a linear-structure automaton with limited states to separate the agents on its two sides as fair and unfair agents. The two sides themselves are separated by a border. The agents are initially put next to the border, which implies them belonging to one of the two groups. The votes taken from every pair of agents on each service are compared to each other. In the case of same answers, negative or positive, the two agents are rewarded if they belong to same group, and are penalized otherwise. The diverse penalty/reward policy is applied for differing answers. A reward moves an agent deeper in its current group, and a penalty moves it towards the other side of the automaton. The service selecting user also is initially located next to the border and with each service tried, it moves to the group which has given the true answer. Here we propose a modified LA-based agent partitioning scheme, with differing agent movement mechanisms.

Object migrating automata (OMA) are used for separating the agents. In such automata inputs from the environment does not result in a change in its state, but it causes transition of objects on the automata between its states. An OMA is represented by a senary $OMA \equiv \{V, \alpha, \beta, F, G, \varphi\}$ in which $V = \{V_1, V_2, ..., V_n\}$ is the set of objects distributed on it, $\varphi = \{\varphi_1, \varphi_2, \varphi_3, ..., \varphi_{KN}\}$ is the set of all possible states. These states can be separated into K subgroups as $\{\varphi_1, ..., \varphi_N\}, ..., \{\varphi_{(K-1)N+1}, ..., \varphi_{KN}\}$ that may be used for partitioning objects on the automaton into distinct groups based on their characteristics. This is what we refer to as an object partitioning automaton.

III. Comparison-Based Agent Partitioning

The automaton used here has a linear structure (Figure 3, 4 or 5) in which its different states are organized on a line. Each agent in the environment is located on one of these states and their transition is only possible between two adjacent states - an agent in the state φ_i can only move (normal form) to φ_{i+1} or to φ_{i-1} in a single transition. The automaton works in two phases: the partitioning phase and the service selection phase.

A. Partitioning Phase

The goal of this phase is to separate the agents who give the true answers in most of times from those who tend to misinform the user about quality of queried services. The former is the set of all agents with fairness values bigger than 0.5 and the latter is the set of the ones with fairness values less than 0.5. The partition is formed based on votes gathered from agents about services in the environment and the comparison between pairs of agents. Agents with similar answers will concentrate on the same side of the automaton which forms a partitioning of them in two distinct groups on these two sides.

The transitions of agents are organized by a penalty/reward policy with the aim of placing each agent in the right partition. There are three modes of such penalty/reward that may cause agent transitions:

1. If two agents in the same group give the same vote on a service, they will both be rewarded, which lets them move to an inner state on the current partition and increase their distance to the border between the two partitions (Figure 3).

2. If two agents from different groups give different votes on the same service, they will both be rewarded, which means each of them can move to an inner state on its corresponding partition and increase its distance to the border (Figure 4).

3. If two agents from the same group give different votes on a service, then they will both be penalized, and should get closer to the other side of the automaton (Figure 5).

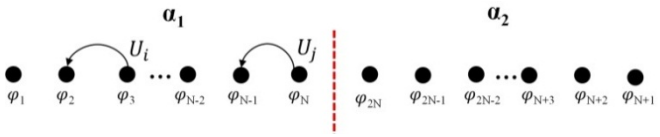

Figure 3. reward for agents in same groups and same vote

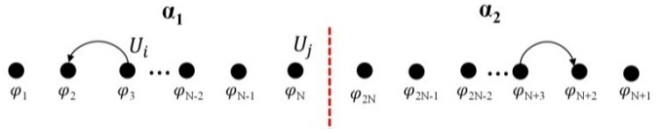

Figure 4. reward for agents in different groups and different vote

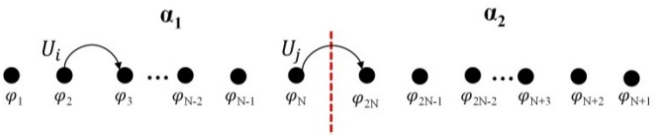

Figure 5. penalty for agents from same group with different votes

Note that there is another case in which two agents from different sides give the same vote for a service. The ATPP algorithm proposed in [2] takes the two agents to be penalized in this case, claiming that one of them is giving the wrong answer anyway; This also may reduce the performance of partitioning by misplacing the agent giving the true vote, and that is why we prefer ignoring this case and leaving the agents without any penalty and reward.

As the partitioning phase proceeds, fair agents concentrate on one side of the automaton and unfair ones on the other side. This results in changes in the average fairness of agents on each of the two sides, which can be an indicator of efficiency of the partitioning scheme. At a specific point in time each agent is located on one of two sides of the automaton. We denote the average fairness of right and left sides by $\overline{f}_R$ and $\overline{f}_L$ respectively. Figure 6 depicts such a situation. Giving a particular service to agents in the automaton and comparing their votes about it, makes them alter their places which may lead to changes in the values of $\overline{f}_R$ and $\overline{f}_L$. Take a typical

agent A located inside the right side whose distance to the border is equal to d and has the fairness value of f_A (Figure 6). The vote taken from A will be compared to those of all other agents.

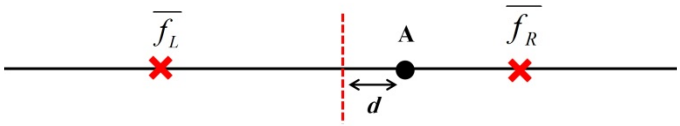

Figure 6. . a typical agent A and its position on the automaton

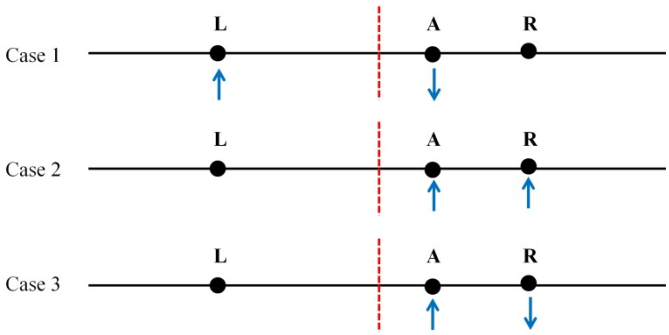

Figure 7. three diffeent penalty/reward cases for A

The probability that the vote taken from two agents A and B are equal is

$$P_{equal} = P(V_A = V_B) = f_A f_B + (1 - f_A)(1 - f_B) \quad (1)$$

Where f_A and f_B are the fairness values of agents A and B respectively. The first phrase of this equation represents the probability that both agents give the true answer, and the second is the probability that both give the unfair, wrong answer. Accordingly, the probability that the votes are different is

$$P_{different} = P(V_A \neq V_B) = f_A(1 - f_B) + f_B(1 - f_A) \quad (2)$$

The comparison may result in three different cases, which are derived from the three possible penalty/reward models described above. The first case is where the vote of agent A is different from the vote of an agent from the other side of the automaton (the left side) as illustrated in Figure 7.Case1. We can use an imaginary agent, L, with fairness value of $\overline{f}_L$ to represent all agents on the left side. The number of such comparison cases is given by

$$N_1 = n_L P_{different} = n_L\left(f_A(1 - \overline{f}_L) + \overline{f}_L(1 - f_A)\right) \quad (3)$$

Where n_L is the number of agents in the left side of the automaton. In this case agent A will be moved towards outmost states on the right side. The second case is the case that agents A gives a vote equal to that of an agent from the same side as Figure 7.Case2 shows. Again we use another imaginary agent R with fairness value $\overline{f}_R$ to represent agents on the right side. The number of occurrences of such comparisons is given by

$$N_2 = n_R P_{equal} = n_R\left(f_A \overline{f}_R + (1 - f_A)(1 - \overline{f}_R)\right) \quad (4)$$

Where n_R is the number of agents on the right side. This comparison will make agent A move towards inner positions of the right side and increase its distance to the border. The third case (Figure 7.Case3) is where two agents from the same side give two differing votes. Accordingly the average number of these comparisons is equal to

$$N_3 = n_R P_{different} = n_R\left(f_A(1 - \overline{f}_R) + \overline{f}_R(1 - f_A)\right) \quad (5)$$

Which will make the agent move towards the border. By combining these three equations, we can deduce the total movement of the agent

$$move = N_1 + N_2 - N_3 \quad (6)$$

This is the amount agent A travels on the automaton. This travel will result in the transfer of A from the right side of the automaton to its left side, if its magnitude is bigger than d, the initial distance of A to the border between two sides of the automaton.

$$P_{transfer} = P(move < -d) \quad (7)$$

This transfer will cause a change in the average fairness of agents on the two sides, which may be approximated by

$$\Delta\overline{f}_L \approx f_A / n_L \quad (8)$$

and

$$\Delta\overline{f}_R \approx -f_A / n_R \quad (9)$$

Taking the initial positions of agents, where they are randomly placed on one of the two sides of the automaton and next to the border, with applying this analysis to all agents and in a sequence of iterations for a given sequence of services, the approximate value of average fairness on each side of the automaton can be calculated on a particular moment of time.

B. Service Selection Phase

Agents in the environment can report their experience with a specific service in terms of a value of either 1 or -1, for good quality and bad quality respectively. A user who attempts to decide on using a service according to votes gathered from other agents, calculates sum of these values reported by other agents and decides to use the service only if this summation has a value greater than 0 and ignore it otherwise.

At the beginning of the algorithm, the user is located randomly on a boundary states of the automaton. The side in which the user is located is temporarily considered to be the fair side and the other side is taken as the unfair side (Figure 8). The user sits just next to the border between its two sides, because in this state it is not aware of the average fairness of agents on each side. He assumes that agents located on one side of the automaton are fair agents while the opposite side contains unfair agents. To improve reliability of the decision, in calculating the sum of reports the user multiplies the votes given by agents in the unfair side of automaton by -1, because they are believed to be wrong reports.

Here we model users that demand using of service or services. The user may be one of three possible types according to its behavioral model:

1. **Normal User**, who has no preferences in judging other agents' trustworthiness

2. **Optimistic User**, who believes most agents in the environment are fair agents; such uses easily trust others

3. **Pessimistic User**, who believes most agents are unfair and thus tends to trust them hardly

These differences are implemented in the form of specific penalty/reward and transition policies for each user type. A normal user alters its location by one hop on the case of either a penalty or a reward. An optimistic user however, completely trusts the automaton by any true answer and immediately jumps to the innermost state on the fair side as shown in Figure 9; the penalty for optimistic user is similar to normal form except, moving from one state to another is probabilistic. Here we assume this probability is 0.8 (Figure 10). On the other hand, a pessimistic user tends to lose its trust to agents on its current side, and jumps to the outermost position (or boundary states) by the first wrong answer (Figure 11). As well as reward for pessimistic user is similar to normal form except, moving from one state to another is probabilistic (we assume this probability is 0.8) (Figure 12).

Each time the user selects a service based on the votes from other agents, he compares the real quality of service experienced by himself with the one reported by other agents. This comparison will result in a transition of the user on the automaton. If qualities are of the same value, the user will reward the automaton and move deeper in the fair side, which we can interpret as an increase in his confidence in the fairness of this side. Note that the size of the automaton, N, is a limitation to the degree a user can remember the history of interactions, which is referred to as the fading factor in the context of trust models. On the other hand, if qualities don't have the same value, then the user will lose confidence in the fair side and penalize it by moving toward the other side, which may lead to conversion of fair and unfair sides, in the case that the user passes the border (Figure 13).

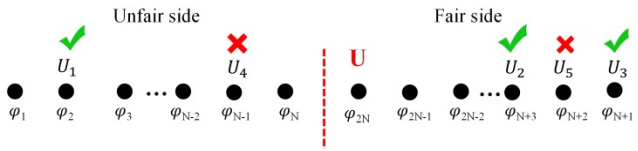

Figure 8. deciding for service selection based on votes

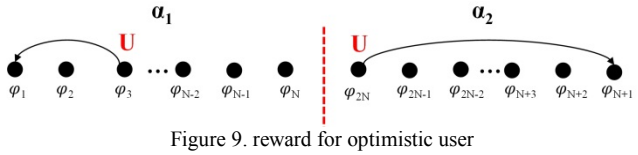

Figure 9. reward for optimistic user

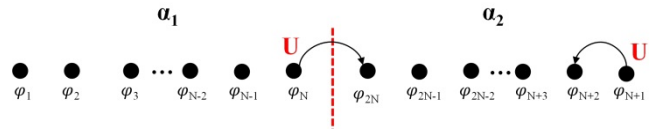

Figure 10. penalty for optimistic user

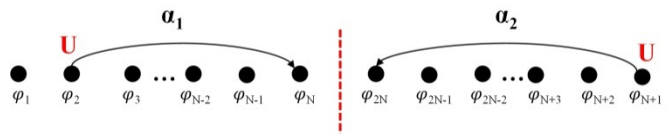

Figure 11. penalty for pessimistic user

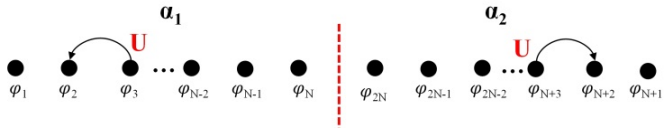

Figure 12. reward for pessimistic user

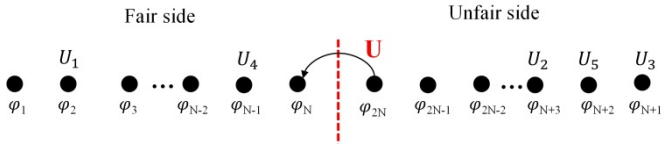

Figure 13. after experiencing service and penalty in event of dissatisfaction with the service

The iterations of partitioning phase separate the fair users from unfair ones, and the iterations of the service selection phase help the user to find the most reliable group of agents to judge the services based on their votes.

IV. Experimenal Results

In order to evaluate the performance of the proposed algorithm, we have implemented it in the form of a static-structure, deterministic automaton and applied it to an environment of 40 independent agents who give votes about the performance of available services. The batch of services consists of 100 services with randomly assigned qualities which have uniform distribution and are static over time. The fairness values for agents vary from 0 up to 1 with uniform distribution too and the agents are randomly located on the states next to the border of the two sides of the automaton at the beginning of the algorithm.

To illustrate the improvements made by the new algorithm its performance is compared to that of ATTP, a learning automaton based trust system described in [2]. Scenarios with different numbers of partitioning phase iterations are simulated and the values for performance measures are captured which represent the extent to which the selected services by the user were of desired quality. The services are the ones advised by other agents. Figure 14 demonstrates the relative performance of these two algorithms (the values in the Figure are the average of results captured from 10,000 runs)

As depicted by Figure 14, only with a few iterations of partitioning, the service selection performance of the proposed algorithm reaches more than 90 percent of the ideal performance, which means that the user is able to select 90 percent of services with desirable quality, while the performance of the other scheme increases with low inclination and needs a few more iterations to converge its best performance.

The size of the automaton and the number of states on each side of it, which is referred to as the size of memory, is another factor influencing the performance of the agent partitioning algorithm. Figure 15 depicts the performance of the two algorithms in automatons with different memory sizes. As

shown in the figure, the proposed algorithm reaches a performance of 90% only with a memory of 3 states, while the ATPP algorithm needs a memory size of at least 5.

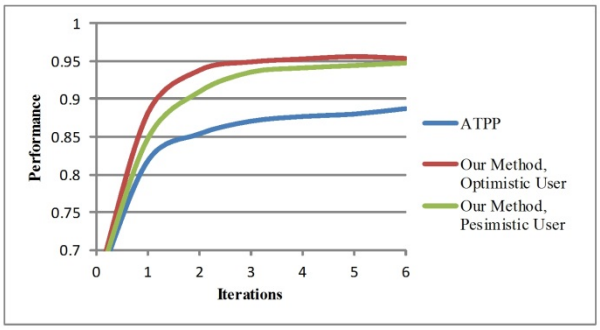

Figure 14. service selection performance for different numbers of iterations

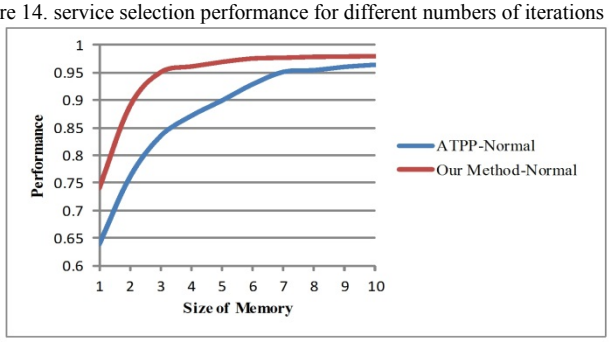

Figure 15. service selection performance for different numbers of partitioning

As another demonstrator for the partitioning performance of the two algorithms, the average fairness of agents located in two distinct sides of the automaton is captured in Figure 16 These values represent the purity of fair and unfair sides (the values in the Figure are the average of results captured from 10,000 runs).

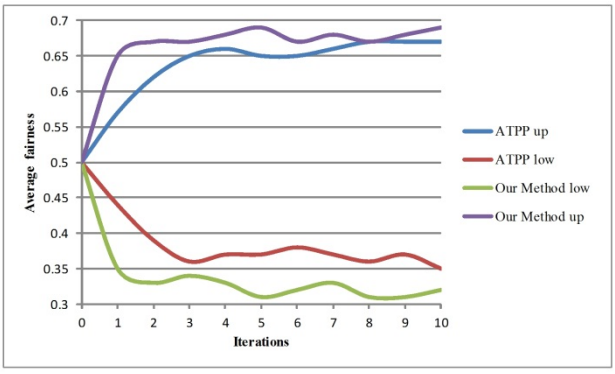

Figure 16. Average fairness for two side of the automaton in different iterations of partitioning in two algorithms

Each curve on the chart represents the values of average fairness on each side of the automaton for the two algorithms and in different iterations of partitioning. At the beginning of the algorithm - before the partitioning- when agents are uniformly distributed on the automaton, the average fairness value for both sides is equal to 0.5, which indicates that the average number of fair and unfair agents on both sides is almost the same. By beginning of partitioning, the average fairness of agents located on the fair side begins to increase, which implies the transfer of some fair agents from unfair side to the fair side and the movement of some unfair agents to the other side. Similarly the decrease of average fairness in unfair side implies the addition of new unfair agents and removal of fair ones. After a couple of iterations, the difference of average fairness values for the two sides reaches almost 0.35 in the proposed algorithm, while it is just about 0.1 for ATPP algorithm. This shows that the new method is faster and more successful in purifying the two fair and unfair sides.

V. CONCLUDING NOTES

Finding services with good quality from a pool of various similar services is a major problem in service oriented environments. Relying on the reports received from other agents in the environment is a common solution to this problem which should itself be based on a trust model between users. Utilizing learning automata (LA) as a means for distinguishing deceptive agents from others has shown to be of acceptable performance. A simple LA-based trust model for service oriented environments is demonstrated here which experimental results show its better performance compared to the other model.

REFERENCES

[1] M. Hafner, R. Breu, "Security Engineering for Service-Oriented Architectures", Springer, 2009.

[2] A. Yazidi, O. Granmo, BJ. Oommen, "Service selection in stochastic environment: learning automata based solution", Springer Science+Business Media, 2011.

[3] Y. Wang, and J. Vassileva, "Trust and reputation model in peer-to-peer networks," IEEE Conference on P2P Computing, Linkoeping, 2003.

[4] D. Artz, and Y. Gil, "A survey of trust in computer science and the semantic web," Journal of Web Semantics: Science, Services and Agents on the World Wide Web.5.2, pp. 58-71, 2007.

[5] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation," Proceedings of the 35th International Conference on System Science, pp. 280–287, 2002.

[6] T. Grandison, M. Sloman, "A survey of trust in internet applications," IEEE Commun. Surv. Tutorials 4, pp. 2–16, 2000.

[7] D. Olmedilla, O.Rana, B. Matthews, and W. Nejdl, "Security an trust issues in semantic grids," Proceedings of the Dagsthul Seminar, Semantic Grid: The Convergence of Technologies. vol. 05271, 1961.

[8] S. Sen, N. Sajja, "Robustness of reputation –based trust: boolean case", In: Proceedings of the first international joint conference on autonomous agents and multiagent systems, ACM, New York, 2002.

[9] W. Yuan, D. Guan, Y-K. Lee, S. Lee, "The small-world trust network", Appl Intell , Volume 35, pp 399-410, 2011.

[10] A. Whitby, A. Jøsang, J. Indulska, "Filtering out unfair ratings in bayesian reputation systems", J Manag Res 4(2):48–64, 2005.

[11] A. Jøsang, R. Ismail, C. Boyd, "A survey of trust and reputation systems for online service provision", Decis Support Syst 43(2):618–64, 2007.

[12] B. Yu, MP. Singh, "Detecting deception in reputation management ", In: Proceeding of the second international joint conference on autonomous agents and multiagent systems, ACM, New York, 2003.

[13] M. Chen, JP. Singh, "Computing and using reputation for internet ratings", In: proceedings of the 3rd ACM conference on electronic commerce, USA, ACM, NewYork, pp 154-162, 2001.

[14] Z. Despotovic, K. Aberer, "A probabilistic approach to predict peers performance in P2P networks", pp 62-76, 2004.

[15] P. Mars, J.R. Chen, Nambir, "Learning Algorithms: Theory and Applications in Signal Processing",Control and Communications, 1996.

[16] K.S. Narenrdra, M. A. L. Thathachar: "Learning Automata: An Introduction", Prentice Hall , 1989.

[17] S. Lakshmivarahan, "Learning Algorithms: Theory and Applications", New York, Springer Verlag, 1981.