Auton Agent Multi-Agent Sys (2006) 13: 119–154
DOI 10.1007/s10458-005-6825-4

An integrated trust and reputation model for open multi-agent systems

Trung Dong Huynh · Nicholas R. Jennings · Nigel R. Shadbolt

Published online: 10 March 2006

Abstract Trust and reputation are central to effective interactions in open multi-agent systems (MAS) in which agents, that are owned by a variety of stakeholders, continuously enter and leave the system. This openness means existing trust and reputation models cannot readily be used since their performance suffers when there are various (unforseen) changes in the environment. To this end, this paper presents FIRE, a trust and reputation model that integrates a number of information sources to produce a comprehensive assessment of an agent's likely performance in open systems. Specifically, FIRE incorporates interaction trust, role-based trust, witness reputation, and certified reputation to provide trust metrics in most circumstances. FIRE is empirically evaluated and is shown to help agents gain better utility (by effectively selecting appropriate interaction partners) than our benchmarks in a variety of agent populations. It is also shown that FIRE is able to effectively respond to changes that occur in an agent's environment.

Keywords Trust · Reputation · Multi-agent systems

1. Introduction

A wide variety of networked computer systems (such as the Grid [11], the Semantic Web [3], pervasive computing systems [28], and peer-to-peer systems [33]) can be viewed as

T. D. Huynh (✉)· N. R. Jennings · N. R. Shadbolt
School of Electronics and Computer Science,
University of Southampton,
Southampton SO17 1BJ,
UK

T. D. Huynh
e-mail: tdh02r@ecs.soton.ac.uk

N. R. Jennings
e-mail: nrj@ecs.soton.ac.uk

N. R. Shadbolt
e-mail: nrs@ecs.soton.ac.uk

multi-agent systems (MAS) in which the individual components act in an autonomous and flexible manner in order to achieve their objectives [16]. An important class of these systems are those that are *open*; here defined as systems in which agents can freely join and leave at any time and where the agents are owned by various stakeholders with different aims and objectives. From these two features, it can be assumed that in open MAS:

1. because of different ownership, the agents are likely to be self-interested and may be unreliable;
2. no agent can know everything about its environment because in such environments it is impossible or too costly to obtain such a global perspective; and
3. because of different ownership, no central authority can control all the agents.

Despite these many uncertainties, a key component of such systems is the interactions that necessarily have to take place between the agents. Moreover, as the individuals only have incomplete knowledge about their environment and their peers, *trust* plays a central role in facilitating these interactions [12, 24]. Here, trust can be viewed as the expectation or the belief that a party will act benignly and cooperatively with the trusting party [8, 12]. Evaluating this expectation before making interactions is important because it can help an agent to estimate the trustworthiness of each potential partner and thus to decide whether the partner is reliable enough to interact with. Specifically, trust is here defined as the subjective probability with which an agent a assesses that another agent b will perform a particular action, both before a can monitor such action and in a context in which it affects its own action (adapted from [12]). Generally speaking, trust can arise from two views: the individual and the societal. The former consists of agent a's direct experiences from interactions with agent b and the various relationships that may exist between them (e.g. owned by the same organisation, relationships derived from links between the agents' owners in real life such as friendship or kinship, or relationships between a service provider agent and its registered consumer agents). The latter consists of observations by a society of agent b's past behaviour (here termed its *reputation*) that are then made available to agents who themselves have not interacted with b. These indirect observations are aggregated in some way to define agent b's past behaviour based on the experiences of participants in the system.

Given its importance, a number of computational models of trust and reputation have been developed (see Section 2 for a more detailed review). However, as they stand, none of them are well suited to open MAS. Typically, they are centralised models, designed with the assumption that they are accepted and trusted by all the individuals that join the system. However, in an open MAS, agents (of various ownerships) may well question the trustworthiness of a centralised reputation service and may not use it (see Property 2 below). On the other hand, in order to work as intended, some of the current models require particular knowledge that is, in general, not readily available. Given the above characteristics of an open MAS, a trust model for such environments needs to possess the following properties:

1. It should take into account a variety of sources of trust information in order to have a more robust trust measure (by cross correlating several perspectives) and to cope with the situation that some of the sources may not be available.
2. Each agent should be able to evaluate trust for itself. Given the "no central authority" nature of an open MAS, agents will typically be unwilling to rely solely on a single centralised reputation service.
3. It should be robust against possible lying from agents (since the agents are self-interested).

To deal with these requirements, we have developed a new trust and reputation model called FIRE[1] (preliminary versions of which are described in [14,15]). In so doing, we advance the state of the art in a number of ways. First, we have developed a modular model that integrates four different types of trust and reputation:

– *interaction trust* resulting from past experience of direct interactions,
– *role-based trust* defined by various role-based relationships between the agents,
– *witness reputation* built from reports of witnesses about an agent's behaviour, and
– *certified reputation* built from third-party references provided by the agent itself.

This breadth is important in our domain because it enables an agent to combine a variety of alternative sources of information (to cope with the inherent uncertainties) and because, in various circumstances, not all of these sources will be readily available (but a measure of trust is nevertheless needed to interact). Second, we empirically evaluate our model in a simulated open MAS where provider agents offer services of various quality levels to consumer agents. In this case, trust models are used by consumer agents to select good providers (i.e. those that yield high utility gain to their consumers) for interactions. In our evaluation, it is shown that the integration of these sources of trust/reputation results in better performance of our model (in terms of the level of utility gain of agents) as against only using one or two of these sources. Third, and of particular relevance, is the introduction of a novel type of reputation—certified reputation. The other, more traditional, ways of building a trust measure (i.e. interaction trust, role-based trust and witness reputation) have certain limitations. For example, if agent a has not interacted with b before, it has no information to calculate its interaction trust. In the case of witness reputation, a may not be able to find relevant witness ratings about b, or the search process may take too long. Finally, there may be no role-base relationships with b. If all these things happen at the same time (e.g. agent a has just joined the environment), agent a will not be able to assess agent b's trustworthiness. In such situations, if agent b can present certified information about its past performance to a (in the form of references from other agents who have interacted with it), agent a will then be able to make some assessment of its trustworthiness. Not only does the addition of certified reputation make our model more applicable than without it, but our evaluation also shows that it significantly increases our model's robustness.

The remainder of the paper is organised as follows. In the next section, we review a number of trust/reputation models against our requirements for trust models in open MAS. In Section 3 we present the FIRE model and its components. Then the testbed and the methodology described in Section 4 will be used to empirically evaluate the model in Section 5. Finally, Section 6 concludes this paper and outlines future work.

2. Related work

A wide variety of trust and reputation models have been developed in the last few years (e.g. [1], [4] [5], [17], [19], [20], [21], [22], [23], [26], [30], [31], [34], [38], [40]). This section reviews a selection of notable models and shows how computational trust models have evolved in recent years with a particular emphasis on their applicability in open MAS (see [24] for a more complete review). Specifically, we divide our discussion into three areas. First, Section 2.1 presents mechanisms that derive trust using certificates, rules, and policies. Second, Section 2.2 surveys the popular trust models that follow the centralised approach in

[1] FIRE is from "**fi**des" (Latin for "trust") and "**re**putation". In the Ramayana legend of India, Sita proved the purity of her character by passing through the raging flames.

which witness observations are reported to a central authority. Finally, Section 2.3 presents notable models that follow the decentralised approach in which no central authority is needed for trust evaluations. From now on, for the convenience in referring agents, we call the agent evaluating the trustworthiness of another the *evaluator*, or agent a; and the agent being evaluated by a the *target* agent, or agent b.

2.1. Trust policy management

In general, security systems are designed to protect resources from harmful accesses. They do so by restricting access to the resources to authorised users only. After having been authenticated by the system (e.g. logging in), an authorised user is granted a clearly defined set of rights, which allows it to access a certain set of resources [13]. In traditional security systems, the set of rights for each user is set manually by system administrators and, therefore, which user and how much it is trusted is determined by them. In order to automate this process, several trust policy management systems have been developed (such as PolicyMaker [13], Trust-Serv [32], and KAoS [35]). In these systems, an agent or user may present information to seek the trust of the system. An agent/user is granted rights based on its certificates of its identity according to predefined policies (i.e. rules such as "if a is a registered user and it possesses a valid credit card then it can book flights"). These systems regard an agent as trusted when it can present sufficient certificates about its identity. This is called *access trust* [13] since the agent is trusted to access the system's resources. However, the situation is reversed in MAS. A typical scenario in MAS is that an agent usually needs to use services provided by others to achieve its goals. It frequently has to decide on which agents' services it should rely. In such situations, it does not need to protect itself from unauthorised accesses, but, rather, needs to protect itself from relying on unreliable service providers. In security or trust policy management systems, the certificates an agent presents cannot provide further information on whether it will act in a trustworthy/reliable manner. Therefore, in scenarios in which an agent needs to protect itself from unreliable providers, such systems are not useful.

In this respect, Maximilien and Singh introduce the concept of endorsements [21]—certificates endorsing that a service (provider) is trusted and preferred by their issuers. However, such endorsements can only let an agent know that the service may be trusted because it is preferred by other agents. This information is typically somewhat vague and does not reveal the expected or achievable performance of that service. Moreover, a service provider may serve different consumers differently. Therefore, endorsements have little utility in identifying unreliable providers. Although endorsements may appear somewhat similar to certified reputation, they are different. Certified reputation are references valuing how good an agent is with respect to an aspect of its service. Therefore, certified reputation provides more useful information for estimating an agent's performance than endorsements. The idea of certified reputation is also similar to the RCertPX protocol [23] in storing ratings at the ratee. However, the work in [23] does not provide a trust model based on these ratings, but focuses instead on designing a sophisticated protocol for peer-to-peer systems that ensures ratings are protected from tampering and can be verified through communication with the raters.

Certificates are also used by Mass and Shehory [20] for trust establishment in open MAS. In their system, an agent presents certificates about itself given by third-parties to gain the trust of another agent. The content of such certificates can be freely defined to certify virtually any facts (e.g. "agent X was developed by IBM", "agent X has a good service quality", "agent X is recommended with a recommendation value 9"). Having been presented with a set of such certificates, an agent uses rules to assign the certified agent to predefined roles (with predefined trust levels) based on these certificates. In contrast, certificates in our approach

provide quantified information about an agent's past performance which are used to computationally estimate that agent's future performance and then the estimated performance will determine the trustworthiness of that agent.

2.2. Centralised reputation mechanisms

Reputation mechanisms have been widely used in online electronic commerce systems (e.g. eBay, Amazon) which typically manage the reputation of all its users in a centralised manner. This section surveys the reputation model of EBay and SPORAS, which are the most popular in this approach.

2.2.1. EBay reputation model

Since traditional security mechanisms cannot protect an agent from unreliable service providers, novel models have been developed to model *service provision trust*—the trust that a service provider is competent and will provide a service in a reliable manner [13]. The main building block of these models is information about an agent's past behaviours. This information is used to deduce the trustworthiness of that agent in terms of its competency and reliability. Online reputation mechanisms (e.g. those on eBay [10, 25] and Amazon Auctions [2]) are probably the most widely used such models. They are implemented as a centralised rating system so that their users can report about the behaviour of one another in past transactions via rating and leaving textual comments. In so doing, users in their communities can learn about the past behaviour of a given user to decide whether it is trustworthy to do business with. For example, an eBay user, after an interaction, can rate its partner on the scale of -1, 0, or $+1$, which means positive, neutral and negative rating respectively. The ratings are stored centrally and the reputation value is computed as the sum of those ratings over six months. Thus, reputation in these models is a global single value representing a user's overall trustworthiness. However, this is too simple for applications in MAS since they only consider the trustworthiness of an agent as one dimension.[2] Since the ratings are aggregated equally, the mechanism cannot adapt well to changes in a user's performance (e.g. a user may cheat in a few interactions after obtaining a high reputation value, but still retains a positive reputation).

In summary, the reputation values in these systems contain very little information, and users of these systems need to look for textual comments providing more information. Therefore, such mechanisms are not well suited to computational agents, which must usually make decisions autonomously. In addition, since there is no central authority that can control all the agents in an open MAS, an agent may well question the credibility of those centralised reputation models and decide not to use them.

2.2.2. SPORAS

SPORAS [40] extends the online reputation models mentioned above by introducing a new method for rating aggregation. Specifically, instead of storing all the ratings, each time a rat-

[2] Decisions associated with task delegation in MAS usually involve considering numerous factors. Therefore, a single overall trustworthiness value is likely be too abstract to be used to differentiate between similar service providers. For example, a news provider who may not always provide the latest news first, but always has correct news, may or may not be preferable over the one who can have all the latest news but cannot guarantee its accuracy.

ing is received it updates the reputation of the involved party using an algorithm that satisfies the following principles:

1. New users start with a minimum reputation value and they build up reputation during their activity on the system.
2. The reputation value of a user never falls below the reputation of a new user.
3. After each transaction, the reputation values of the involved users are updated according to the feedback provided by other parties, which reflect their trustworthiness in the latest transaction.
4. Users with very high reputation values experience much smaller rating changes after each update.
5. Ratings must be discounted over time so that the most recent ratings have more weight in the evaluation of a users's reputation.

In general, SPORAS is a centralised reputation model with more sophisticated characteristics to model the trust dynamics than the simple models in the previous section. For example, Principles 1 and 2 above are to prevent a user with a bad reputation leaving the community and entering with a fresh reputation (since the reputation of a new user is the lowest reputation possible). However, at the same time, this penalises newcomers and may discourage them from participating in the community. In addition, SPORAS also introduces a reliability measure based on the deviation of rating values. This is an indication of the predictive power of SPORAS for that user's reputation. For instance, a high deviation value can mean either that the user has not been active enough to be able to generate a more accurate prediction for his/her reputation, or that the user's behaviour has a high degree of variation. Hence, each user has a reputation value and a reliability value globally available to other users. In SPORAS, the reputation value of a user and its reliability are discounted over time as a new rating is received. Therefore, SPORAS can adapt to changes in a user's behaviour according to the latest rating.

In summary, SPORAS provides a trust measure that has more desirable features than that of similar online models such as eBay's, or Amazon's. However, as discussed in the previous section, its centralised design is not suitable for applications in an open MAS. Moreover, as shown by our experiments in Section 5, SPORAS is very susceptible to rating noise resulted from agents' subjective views that are commonplace in open MAS.

2.3. Decentralised trust models

As more and more computational systems of all kinds move toward large-scale, open and dynamic architectures, more and more trust models are designed such that each agent can carry out trust evaluation themselves without a central trust authority. This section surveys several models that follow this approach.

2.3.1. Jurca and Faltings

Jurca and Faltings [17] introduce a reputation mechanism where agents are incentivised to report truthfully about their interactions' results. They define a set of broker agents (called R-agents) whose tasks are buying and aggregating reports from other agents and selling back reputation information to them when they need it. All reports about an agent are simply aggregated using the averaging method to produce the reputation value for that agent. Although the R-agents are distributed in the system, each of them collects and aggregates reputation reports centrally. Hence this approach still possesses the inherent shortcoming of centralised models

above (i.e. the questionable objectiveness of R-agents in open MAS). In order to incentivise agents to share their reports truthfully, Jurca and Faltings propose a payment scheme for reputation reports. This scheme guarantees that agents who report incorrectly will gradually lose money (during the process of selling reports and buying reputation information), while honest agents will not. Therefore, this mechanism makes it rational for an agent to report its observations honestly and this is the main contribution of their work. However, reputation reports are limited to the values 0 and 1 (0 for cheating agents and 1 for cooperating agents in an iterated Prisoner's Dilemma environment [7]), and the rational property may not hold if an application requires reports represented by more than these particular values (e.g. 0.1, 0.75).

2.3.2. Regret

Regret [27, 26] is a reputation model in which the trust evaluation process is completely decentralised. Employing Regret, each agent is able to evaluate the reputation of others by itself. In order to do so, each agent rates its partner's performance after every interaction and records its ratings in a local database. The relevant ratings will be queried from this database when trust evaluation is needed. The trust value derived from those ratings is termed *direct trust* and is calculated as the weighed means of all ratings. Each rating is weighed according to its recency. Intuitively, a more recent rating is deemed to be more current and is weighted more than those that are less recent. However, the method Regret uses to calculate the weights for each rating has a shortcoming regarding time granularity control and does not actually reflect a rating's recency (see Appendix A.) Like SPORAS, Regret also provides a reliability value for each trust value to represent its predictive power. The reliability value is calculated from two reliability measures: the number of ratings taken into account in producing the trust values and the deviation of these ratings.

In addition, agents are assumed to be willing to share their opinions about one another. Based on this, Regret develops a witness reputation component along with a sophisticated method for aggregating witness reports taking into account the possibility of dishonest reports. The operation of this component depends on the social network[3] built up by each agent. In particular, Regret uses the social network to find witnesses, to decide which witnesses will be consulted, and how to weight those witnesses' opinions. However, Regret does not specify how such social networks are to be built, and, thus, this component is of limited use.

Besides direct trust and witness reputation, Regret also introduces the concepts of neighbourhood reputation and system reputation. The former is calculated from the reputation of the target's neighbour agents based on fuzzy rules. However, this again requires a social network to work. The system reputation is a mechanism to assign default trust values to the target agent based on its social role in an interaction (e.g. buyer, seller). However, this is only useful if additional domain-specific information is available.

In summary, the decentralising approach of Regret allows agents to evaluate trust by themselves without relying on a centralised mechanism. It also takes various sources of trust information into account and consider the possibility of disinformation. Therefore, the approach Regret adopts is compatible with the requirements for a trust model in open MAS. However, apart from the direct trust component, the rest of the model is not readily applicable.

[3] A social network is a graph whose nodes represent agents in a society and whose edges represent social relationships between them (see [36] for more detail). It can be viewed as an agent's (limited) knowledge about the social structure of the world in which it is situated.

The main reason is that Regret does not show how each agent can build the social network on which Regret heavily depends.

2.3.3. Referral System

In building a reputation system based on witness information, Yu and Singh [38, 39] develop a mechanism to locate information sources (i.e. witnesses) based on individual agents' knowledge and help (through each agent's contacts) without relying on a centralised service. Hence, this approach is well suited for applications in an open MAS which is distributed by nature. In particular, in this system, agents cooperate by giving, pursuing, and evaluating referrals (a recommendation to contact another agent). Each agent in the system maintains a list of acquaintances (other agents that it knows) and their expertise. Thus, when looking for a certain piece of information, an agent can send the query to a number of its acquaintances who will try to answer the query if possible or, if they cannot, they will send back referrals pointing to other agents that they believe are likely to have the desired information (based on those agent's expertise). Yu and Singh's referral system uses a vector space model (VSM) [29] to model agents' expertise. An agent's expertise is then used to determine how likely it is to have interaction with or to know witnesses of the target agent.

2.3.4. TRAVOS

TRAVOS [34] is a trust model that is built upon probability theory and based on observations of past interaction between agents. In this model, the outcome of an interaction is simplified into a binary rating (i.e. 1 for a successful interaction, 0 for a unsuccessful one). Using binary ratings allows TRAVOS to make use of the beta family of *probability density functions* (PDF) [9] to model the probability of having a successful interaction with a particular given agent. This probability is then used as that agent's trust value. In addition, using PDFs, TRAVOS also calculates the confidence of its trust values given an acceptable level of error. If the confidence level of a trust value is below a predetermined minimum level, TRAVOS will seek witness information about the target agent's past performance. Witness information is shared in the form of frequencies of successful and unsuccessful interactions that the witness has had with the target agent. After interacting with the target agent itself, the evaluator compares the received witness report with its own observations. By this means, the evaluator calculates the probability that the witness's information supports the true behaviour of the target agent within a reasonable margin of error, and uses this probability to weight the impact of the witness' opinions on future decisions made be the evaluator. However, as discussed in Section 2.2, TRAVOS's simplified representation of interaction ratings is rather limited and not suitable for a wide range of applications in open MAS.

3. The FIRE model

This section presents our model of trust and reputation for applications in open MAS. First, we present the sources of trust information that FIRE uses for trust evaluation (Section 3.1). Then Section 3.2 shows how a trust value is calculated. The individual components of FIRE are subsequently described in Sections 3.3–3.6. Finally, Section 3.7 describes how an overall trust value is produced by combining these components.

3.1. Sources of trust information

As can be seen in the previous section, trust can come from a number of information sources: direct experience, witness information, rules or policies. However, due to the openness of a MAS, the level of knowledge of an agent about its environment and its peers may vary greatly during its life cycle. Therefore, at any given time, some information sources may not be available, or adequate, for deducing trust. For example, the following situations may (independently) happen:

- An agent may never have interacted with a given target agent and, hence, its experience cannot be used to deduce how trustworthy the target agent is.
- An agent may not be able to locate a witness for the target agent (because of a lack of knowledge about the target agent's society) and, therefore, it cannot obtain witness information about that agent's behaviours.
- The current set of rules to determine the level of trust is not applicable for the target agent.

In such scenarios, trust models that use only one source of information will fail to provide a trust value of the target agent. For that reason, FIRE adopts a broader base of information than has hitherto been used for providing trust-related information. Although the number of sources that provide trust-related information can be greatly varied from application to application, we consider that most of them can be categorised into the four main sources as follows:

- *Direct experience*: The evaluator uses its previous experiences in interacting with the target agent to determine its trustworthiness. This type of trust is called *Interaction Trust*.
- *Witness information*: Assuming that agents are willing to share their direct experiences, the evaluator can collect experiences of other agents that interacted with the target agent. Such information will be used to derive the trustworthiness of the target agent based on the views of its witnesses. Hence this type of trust is called *Witness Reputation*.
- *Role-based rules*: Besides an agent's past behaviours (which is used in the two previous types of trust), there are certain types of information that can be used to deduce trust. These can be the various relationships between the evaluator and the target agent or its knowledge about its domain (e.g. norms, or the legal system in effect). For example, an agent may be preset to trust any other agent that is owned, or certified, by its owner; it may trust that any authorised dealer will sell products complying to their company's standards; or it may trust another agent if it is a member of a trustworthy group.[4] Such settings or beliefs (which are mostly domain-specific) can be captured by rules based on the roles of the evaluator and the target agent to assign a predetermined trustworthiness to the target agent. Hence this type of trust is called *Role-based Trust*.
- *Third-party references provided by the target agents*: In the previous cases, the evaluator needs to collect the required information itself. However, the target agent can also actively seek the trust of the evaluator by presenting arguments about its trustworthiness. In this paper, such arguments are references produced by the agents that have interacted with the target agents certifying its behaviours.[5] However, in contrast to witness information

[4] This belief is similar to the neighbourhood reputation in Regret, which calculates the reputation of an agent from the reputation of the agents that it is connected to.

[5] The arguments can also be the target agent's identity, its certifications (e.g. "authorised dealer", performance awards), its sources of products (to guarantee their quality), and so on. However, deducing trust (or the expected performance) of the target agent from such information requires knowledge about the application domain.

which needs to be collected by the evaluator, the target agent stores and provides such certified references on request to gain the trust of the evaluator. Those references can be obtained by the target agent (assuming the cooperation of its partners) from only a few interactions, thus, they are usually readily available. This type of trust is called *Certified Reputation*.

FIRE integrates all four sources of information and is able to provide trust metrics in a wide variety of situations. Certified Reputation, in particular, greatly enhances FIRE in this respect since the evaluator does not have to obtain this type of information itself (as is the case with other types of trust). Hence, the addition of certified reputation decreases the possibility that the evaluator fails to evaluate the trustworthiness of the target agent due to a lack of information. Our hypothesis is that integrating these various sources will also enhance the precision of the trust model. This will be verified subsequently in our empirical evaluation. Here, each type of trust information is processed by a particular component of FIRE: interaction trust (IT), witness reputation (WR), role-based trust (RT), and certified reputation (CR) components.

It should be noted that the WR and CR components depend on third-party information (witness experiences and references) and, therefore, they are susceptible to any inaccuracy of such information. Since agents in an open MAS are self-interested, they may provide false ratings to gain unwarranted trust for their partners. However, in this paper, we have not considered the problem of lying and inaccuracy. Therefore, we currently assume that all agents are honest in exchanging information. Although this is unrealistic for an open MAS, our aim now is to ascertain that our philosophy and trust components are actually effective. The problem of various sorts of disinformation in reporting ratings will be considered in future work.

In addition to the characteristics of an open MAS, we have made a number of assumptions about the agents and their environment. Before going on to discuss FIRE, we state these assumptions:

Assumption 1. Agents are willing to share their experiences with others (as witnesses or as referees).

Assumption 2. Agents are honest in exchanging information with one another.

In FIRE, except the RT component, which deduces trust based on rules, the other components deduce trust from information about the target agent's behaviour. We use *ratings* to capture this type of information. Here, a rating is the evaluation about an agent's performance given by its partner in an interaction between them. For instance, consider an example where agent a subscribes to a news service provided by agent b. Each time a receives a piece of news from b, it can evaluate the news provided in terms of topicality, quality, and honesty. From its evaluation, agent a may give ratings about agent b's service in those terms for that particular interaction. Ratings are thus tuples in the following form: $r = (a, b, c, i, v)$, where a and b are the agents that participated in the interaction i, and v is the rating value a gave b for the term c (e.g. topicality, quality, honesty). The range of v is $[-1, +1]$, where -1 means absolutely negative, $+1$ means absolutely positive, and 0 means neutral.

Each time agent a gives a rating, it will be stored in the agent's local rating database. Ratings in this database will be retrieved when needed for trust evaluation or for sharing with

This is dealt with in Role-based Trust based on rules encoding an agent's beliefs. Therefore, we only consider third-party references here because they can be quantified and computationally aggregated in a standardised way as we show later in this section.

other agents. However, an agent does not need to store all ratings it makes. As the environment of an open MAS is dynamic, old ratings usually become out-of-date due to changes in the environment. In addition, since each agent has limited resource (i.e. memory), storing all ratings about various agents is not necessarily an option. Therefore, each agent will only store at maximum the H latest ratings given to another agent. Here H is called the *rating history size*. This parameter is adjustable according to a particular agent's situation.

3.2. Trust formula

In order to calculate the trust value of a target agent, the components of FIRE will have to collect relevant ratings about that agent's past behaviour. The subsequent sections will define how and which ratings are collected by each component. This section describes how the set of ratings each component collects is used to estimate the target agent's future behaviour, or more specifically, the expected rating value that agent is likely to receive in a future interaction. It is also viewed as the target agent's trust value. A common way to estimate that value is to calculate it as the arithmetic mean of all the rating values in the set. However, these ratings are usually not equally relevant when estimating the expected rating value. For example, some ratings may be older than others and, thus, are deemed to be out-of-date; some may come from a more reliable source that suggests a higher level of credibility compared to others. Therefore, we devise a *rating weight function* ω_{K} for each component of FIRE[6] which calculates the relevance of each given rating. K is thus one of I, R, W, and C standing for interaction trust, role-based trust, witness reputation, and certified reputation respectively. Then instead of considering all ratings equally, the trust value is calculated as the weighted mean of all the ratings available[7], whose weights are given by the corresponding weight function:

$$\mathcal{T}_{\mathsf{K}}(a, b, c) = \frac{\sum_{r_i \in \mathcal{R}_{\mathsf{K}}(a,b,c)} \omega_{\mathsf{K}}(r_i) \cdot v_i}{\sum_{r_i \in \mathcal{R}_{\mathsf{K}}(a,b,c)} \omega_{\mathsf{K}}(r_i)} \tag{1}$$

where $\mathcal{T}_{\mathsf{K}}(a, b, c)$ is the trust value that agent a has in agent b with respect to term c, which is calculated by the component K; $\mathcal{R}_{\mathsf{K}}(a, b, c)$ is the set of ratings collected by component K for calculating $\mathcal{T}_{\mathsf{K}}(a, b, c)$; $\omega_{\mathsf{K}}(r_i)$ is the rating weight function that calculates the relevance or the reliability of the rating r_i ($\omega_{\mathsf{K}}(r_i) \geq 0$); and v_i is the value of the rating r_i. In short, the trust value is calculated as the sum of all the available ratings weighted by the rating relevance and normalised to the range of $[-1, 1]$ (by dividing the sum by the sum of all the weights). The rating weight function $\omega_{\mathsf{K}}(r_i)$ is later defined for each component.

As we have discussed, the trust value given above lets an agent know the expected performance of the target agent. However, the trust value alone is not enough. For example, a trust value of $+1$ calculated from only 1 rating or from 10 ratings may have different effects on an agent's decision. Therefore, an agent usually also needs to know how likely it is that the target agent will perform at that expected performance (similar to the expected value and deviation measures in statistics). In other words, apart from the trust value, its reliability should also be provided by a trust model. Here, we define a reliability measure that reflects the confidence of the trust model in producing each trust value given the data it took into account. This is given in the form of a reliability value that ranges in [0, 1], where 0 is for

[6] Since each component of FIRE collects ratings from a different source, it also needs a different way to calculate the relevancy of ratings. For example, the WR component may have information about witness credibility to take into account when weighing ratings, while this information is not relevant to the IT or RT components.

[7] We choose the weighted mean method here because it allows us to take the relevance of each rating into account. Other aggregation methods could equally well be used if desired.

complete uncertainty and 1 for total confidence. The reliability value is given based on the two following measures:

– *Rating reliability*: Since the rating weight function ω_{K} gives us the relevancy — in other words, the quality or the reliability — of each rating taken into account, the sum of all rating weights reflects the reliability of the rating set taken into account in computing $\mathcal{T}_{\mathsf{K}}(a, b, c)$ in (1) above. Therefore, we devise a rating reliability measure based on this sum:

$$\rho_{\mathsf{RK}}(a, b, c) = 1 - e^{-\gamma_{\mathsf{K}} \cdot \left(\sum_{r_i \in \mathcal{R}_{\mathsf{K}}(a,b,c)} \omega_{\mathsf{K}}(r_i)\right)} \tag{2}$$

where $\rho_{\mathsf{RK}}(a, b, c)$ is the reliability value of the rating set $\mathcal{R}_{\mathsf{K}}(a, b, c)$, γ_{K} is a parameter used to adjust the slope of the reliability function to suit the rating weight function of each component (see Fig. 1). Since each component has its own rating weight function, it also has a rating reliability function of its own—ρ_{RK}. As above, K is one of I, R, W, and C. R in ρ_{RK} stands for "rating reliability". Intuitively, the rating reliability should increase proportionally to the sum of the rating weights. However, since this sum is not limited, we choose the (increasing) function $1 - e^{-x}$ in order that the resulted reliability value is limited in [0, 1]. This normalisation is required because the trust and reliability values of FIRE's components will be combined later on in Section 3.7. Moreover, since each rating weight function is defined differently and may have an inconsistent range compared to one another's, the parameter γ_{K} is introduced in order to adjust the rate of the rating reliability in (2) according to each rating weight function's range. In sum, the rating reliability function $\rho_{\mathsf{RK}}(a, b, c)$ gradually increases from 0 (the lowest reliability) to 1 (the highest reliability) when the sum of rating weights increases from 0 to $+\infty$.

– *Deviation reliability*: The greater the variability in the rating values, the more volatile the other agent is likely to be in fulfilling its agreements. Therefore, the deviation in the ratings' values is also a metric that reflects a trust value's reliability:

$$\rho_{\mathsf{DK}}(a, b, c) = 1 - \frac{1}{2} \cdot \frac{\sum_{r_i \in \mathcal{R}_{\mathsf{K}}(a,b,c)} \omega_{\mathsf{K}}(r_i) \cdot |v_i - \mathcal{T}_{\mathsf{K}}(a, b, c)|}{\sum_{r_i \in \mathcal{R}_{\mathsf{K}}(a,b,c)} \omega_{\mathsf{K}}(r_i)}, \tag{3}$$

where $\rho_{\mathsf{DK}}(a, b, c)$ is the deviation reliability value of the trust value $\mathcal{T}_{\mathsf{K}}(a, b, c)$. Here, D in ρ_{DK} stands for "deviation". Basically, (3) calculates the deviation of ratings' values in the set of ratings $\mathcal{R}_{\mathsf{K}}(a, b, c)$ around the "expected" value (i.e. the trust value); the calculated deviation is then normalised to [0,1]. Intuitively, when there is no deviation in the rating's value (i.e. the target agent performs consistently), the deviation reliability is 1 (i.e. the most reliable); and it decreases proportionally to 0 (i.e. the least reliable) when the deviation increases.

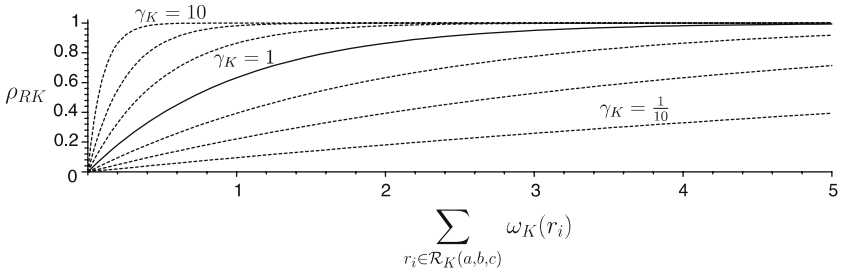

Fig. 1 Rating reliability function $\rho_{RK}(a, b, c)$

In order to take both of these reliability factors above into account, the reliability value of the produced trust value, denoted by $\rho_{\mathsf{K}}(a, b, c)$, is defined as the combination of the rating and the deviation reliabilities:

$$\rho_{\mathsf{K}}(a, b, c) = \rho_{\mathsf{RK}}(a, b, c) \cdot \rho_{\mathsf{DK}}(a, b, c) \tag{4}$$

3.3. Interaction trust

As introduced in Section 3.1, interaction trust is built from the direct experience of an agent. It models the trust that ensues from the direct interactions between two agents. Here we simply exploit the direct trust component of Regret (see Section 2.3) since this meets all our requirements for dealing with direct experiences. In more detail, each agent rates its partner's performance after every transaction and stores its ratings in a local rating database. When calculating the IT value for agent b with respect to term c, agent a has to query its database for all the ratings that have the form $(a, b, c, \_, \_)$, where the "_" symbol can be replaced by any value. We call the set of those ratings $\mathcal{R}_{\mathsf{I}}(a, b, c)$.

Since older ratings may become out-of-date quickly, we use the recency of the ratings as a rating weight function to give recent, and likely more updated, ratings more weights than older ratings in IT evaluation. However, as pointed out in Section 2.3, Regret's method of calculating rating recency has several unfavourable aspects. Therefore, we devise a new rating recency function based on the time difference between current time and the rating time as this metrics reflects precisely how old (i.e. how recent) a rating is. In order to make our rating recency function adjustable to suit the time granularity in different applications, the parameter λ, called the *recency scaling factor*, is introduced in the function (to scale time values). Our rating recency function, which is also used as the rating weight function for IT, is given by the following formula:

$$\omega_{\mathsf{I}}(r_i) = e^{-\frac{\Delta t(r_i)}{\lambda}} \tag{5}$$

where $\omega_{\mathsf{I}}(r_i)$ is the weight for the rating r_i (used in (1)), $\Delta t(r_i)$ is the time difference between the current time and the time when the rating r_i is recorded. In our model, analogously to human perception, we view the time difference of two recent events as more significant than the same one of two older events (see Appendix A for an example). Hence, the exponential function above is chosen for rating recency because its shape over time fits our view on how the recency of ratings should affect an agent's decision about trust (see Fig. 2). Our intuition is that new ratings are deemed to reflect the target agent's current performance more appropriately than old ratings, and our recency function here is to help FIRE adapt quickly to any changes in that agent's performance. In (5), the parameter λ is hand-picked for a particular application depending on the time unit used. For instance, if the time unit used is *day* and we want a rating obtained *five days* earlier to only have half the effect of a new rating obtained *today* (i.e. rating weights of 0.5 and 1, respectively; $\Delta t(r_i) = 5$) then $\lambda = -\frac{5}{\ln(0.5)}$.

Given the rating set $\mathcal{R}_{\mathsf{I}}(a, b, c)$ and the rating weight function $\omega_{\mathsf{I}}(r_i)$ as specified above, the IT value $\mathcal{T}_{\mathsf{I}}(a, b, c)$ and its reliability $\rho_{\mathsf{I}}(a, b, c)$ are calculated as defined in (1) and (4) (Section 3.2).

3.4. Role-based trust

Role-based trust models the trust resulting from the role-based relationships between two agents (e.g. owned by the same company, a service provider and its registered user, or a

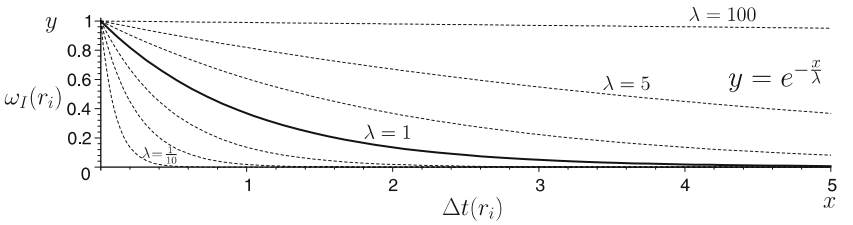

Fig. 2 Rating weight function of interaction trust component

friendship relationship between their owners). Since there is no general method for computationally quantifying trust based on this type of relationship, we use rules to assign RT values. As previously discussed, those rules are used to encode knowledge about the trust dynamics in the application domain. Therefore, they are usually domain-specific and must be specified by an agent's designer or its owner. In other words, this component provides the means of adapting FIRE to a particular environment and, thus, making it perform better in that environment. Here, rules are tuples of the following form: $rul = (\text{role}_a, \text{role}_b, c, e, v)$, which describes a rule that if role_a and role_b are the roles of agent a and b, respectively, then the expected performance of b with respect to the term c in an interaction with a is v ($v \in [-1, 1]$); $e \in [0, 1]$ is the level of influence of this rule on the resulting RT value or the belief strength of agent a on the rule. For example, possible rules may be:

$rul_1 =$ (buyer, seller, quality, 0.3, −0.2),
$rul_2 =$ (_, government-seller, quality, 0.8, 0.0),
$rul_3 =$ (_, team-mate, honesty, 1.0, 1.0).

Here rul_1 expresses an agent's belief that an ordinary seller will usually sell a product of slightly lower quality than agreed, but the reliability of this belief is low (0.3); rul_2 expresses a stronger belief that an agent can expect a governmental seller to do what is agreed in terms of product quality; rul_3 tells an agent to expect total honesty from its team mate (e.g. agents of the same owner). Here, rul_1 and rul_2 encode norms of the environment, while rul_3 is the belief based on an arrangement between agents. Such rules are given to the agent by its owner. Additional rules can naturally be added during an agent's life cycle.

Each agent has its own set of rules which are stored in a (local) rule database. In order to determine the RT of agent b with respect to term c, agent a looks up the relevant rules from its rule database. We call the set of those rules $\mathcal{R}_{\mathsf{R}}(a, b, c)$. Since the form of a rule is very analogous to that of a rating, the general trust formula in (1) can be used to calculate the RT of b, which is denoted by $\mathcal{T}_{\mathsf{R}}(a, b, c)$, from this set. Here, the level of influence of each rule is used as the weight for that rule: $\omega_{\mathsf{R}}(r_i) = e_i$. Therefore, it should be noted that in case there exist conflicts in the applicable rules (i.e. contradicting expected performance values), all these rules will be taken into account but the deviation measure reliability (ρ_{DK}) of the resulted trust value will be low (because of the high deviation of the rules used). This, in turn, will result in a low reliability of the RT trust value, which shows that the RT trust value has a low predictive power and so it will be weighted accordingly in calculating the overall trust value (see Section 3.7).

3.5. Witness reputation

The witness reputation of a target agent b is built on observations about its behaviour by other agents (witnesses). In order to evaluate the WR of b, an agent a needs to find the

witnesses that have interacted with b. Here, it is assumed that agents in open MAS are willing to share ratings that they made and to help others search witnesses. In order to find relevant witnesses, we implement a variant of Yu and Singh's referral system (Section 2.3) without using the VSM model for modelling expertise.[8] Instead, our system assumes that each agent has a measure of the degree of likeliness with which an agent can fulfil an information query about witness information and witness locating. This measure needs to be defined in an application specific manner. For example, in our testbed (described in Section 4.1), an agent is assumed to know local agents (those that are near to it) better and, therefore, we use the distance between an acquaintance and the target agent as the knowledge measure. Thus the nearer to the target agent, the more likely the acquaintance is to know it. This measure is used in the referral process to help locate witnesses. However, it should be noted that the resources available to each agent are limited (in terms of its memory, its communication cost, or some other measure) and the evaluator (agent a) usually has limited time for trust evaluation (before it has to initiate an interaction). Thus, the process of locating witnesses should be limited according to an agent's time constraints, though this may result in no witnesses being located (even though appropriate agents are available in the system). Here, the parameters n_{BF} (called the *branching factor* [39]) and n_{RL} (called the *referral length threshold*, or the depth of referral graphs in [39]) are introduced for that purpose. Specifically, n_{BF} is used to limit the number of acquaintances to which a query is forwarded, and n_{RL} to limit the length of referral chains. Besides restricting the search range of agent a due to time constraints, the referral length threshold also helps an agent not to waste its effort querying agents that are too distant because the further the witness is from a (in terms of the length of the referral chain to the witness from a), the less reliable or relevant its information. At present, n_{BF} and n_{RL} need to be hand-picked according to an agent's resources constraints and its environment's acquaintance networks.

Specifically, the process of evaluating WR is as follows:

1. When agent a assesses the WR of agent b with respect to term c, denoted by $\mathcal{T}_{\mathsf{W}}(a, b, c)$, it sends out a query for ratings of the form $(\_, b, c, \_, \_)$ to n_{BF} acquaintances that are likely to have relevant ratings on agent b and term c (see Fig. 3, where $n_{\mathsf{BF}} = 2$).
2. These acquaintances, upon receiving the query, try to match it to their own (local) rating databases. If they find matching ratings, it means they have had interactions with b, and they will return the ratings found to a.
3. If they cannot find the requested information, they will return referrals identifying their n_{BF} acquaintances that they believe are most likely to have the relevant ratings to the query (based on the knowledge measure) so that a can look further.
4. This process continues until a finds sufficient witnesses or the length of its referral chains reach the defined threshold n_{RL}.

It should be noted here that in this process we implicitly assume that agents in a's referral network are willing to help a finding the required witness ratings. This is not a trivial assumption and needs to be guaranteed for this referral process (as for any mechanism based on third-party information) to work, especially in open MAS where agents are self-interested. However, we do not consider how such a guarantee can be obtained in this paper because that task would very much depend on the particular application domain being considered. Thus, end-users who wish to make use of WR need to provide necessary measures for this

[8] As pointed out in [37], the VSM model does not support hierarchy in expertise types, which can be better represented by service graphs. In this respect (i.e. modelling expertise), there is no universal model for all applications. Therefore, we leave the choice of expertise model to end users as they can evaluate which method is best suited to their particular applications.

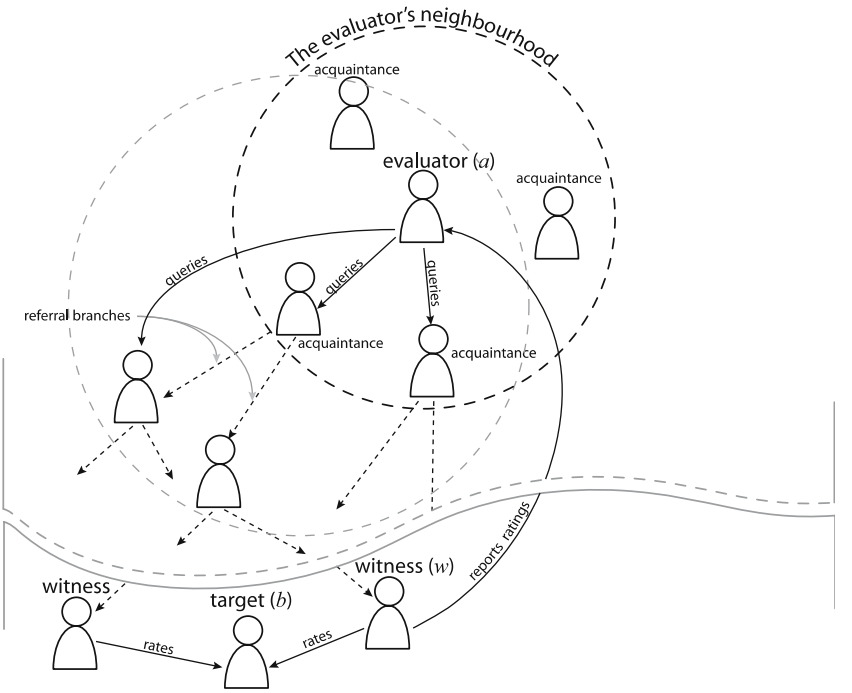

Fig. 3 Referral process

willingness assumption to hold (e.g. obtaining an agreement between agents on sharing witness information or paying for any information request).

The set of ratings collected from the referral process, denoted by $\mathcal{R}_{\mathsf{W}}(a, b, c)$, is then used to calculate the WR of agent b (i.e. $\mathcal{T}_{\mathsf{W}}(a, b, c)$) following (1). Here, the rating weight function for WR, $\omega_{\mathsf{W}}(r_i)$, is intended to reflect the quality of a witness rating which also includes the rating's credibility (since in realistic environments agents may give false or inaccurate ratings). However, as we currently assume all agents are honest, only the recency of ratings is used as per Section 3.3 (i.e. $\omega_{\mathsf{W}}(r_i) = \omega_{\mathsf{I}}(r_i)$, see (5)). A model of witness credibility will be developed and incorporated into FIRE in future work.

3.6. Certified reputation

Certified reputation of a target agent b consists of a number of certified references[9] about its behaviour on particular tasks that are provided by third-party agents. Such information is obtained and stored by the target agent itself and is then made available to any other agent that wishes to evaluate its trustworthiness for further interactions (somewhat like a reference when a person is applying for a job). The agents giving references are called the *referees*. Here, references are in the form of ratings given by b's partners about its performance in (past) interactions between them. These ratings allow b to prove its (achievable)

[9] It is assumed that some form of security mechanism (such as a public-key infrastructure) is employed to ensure that the provided references cannot be tampered with. For instance, all references could be accompanied by digital signatures from the issuers using their private keys [41]. By so doing, any change to a reference will be easily detected. Digital signatures are also a means to verify the references' origins.

performance as viewed by its previous interaction partners and then to gain the trust of its potential partners. However, since b can choose which ratings to put forward, a rational agent will only present its best ones. Therefore, it should be assumed that CR information probably overestimates an agent's expected behaviour. Thus, although it cannot guarantee b's minimal performance in future interactions, the CR information does reveal a partial perspective on agent b's capabilities (which is certainly useful for trust evaluation in the absence of other sources of information).

Though CR may have lower predictive power than the other types of trust (where all bad and good ratings can be collected), it is useful because of its wide applicability. With the cooperation of its partners, agent b can obtain their references from just a small number of interactions.[10] From our evaluation, for instance, in a society where 100 agents provide a service to 500 others, agents using direct experience to evaluate trust require more than 100 interactions to achieve a reasonable level of utility gain, which is still less than that achieved by agents using CR after five interactions (see Section 5.2 for more detail). In addition to its high availability, since references are stored by the target agent and provided directly to the evaluator, CR has a very low running cost (i.e. time, communication, processing cost) compared to sources like witness reputation. Since CR information comes from the target agent, the CR component complements the other components of FIRE, which use information collected by the evaluator, reducing the chances that they may fail to calculate trust due to lack of input (see Section 3.1). Thus, incorporating the CR component makes FIRE able to provide a trust value in most circumstances.

In more detail, the process of CR is as follows:

- After every transaction, agent b asks its partners to provide their certified ratings about its performance from which it can choose the best ratings to store in its (local) rating database.
- When agent a contacts b to express its interest in using b's service, it asks b to provide references about its past performance with respect to an interested term c.
- Agent a receives the set of certified ratings of b from b, which we call $\mathcal{R}_{\mathsf{C}}(a, b, c)$ (C to denote this set is obtained via the CR mechanism), and calculates the CR of b based on this set.

In this process, since agent b relies on its interaction partners' cooperation to get references, agents may refuse to give out their ratings (as in the case of witness reputation). However, this is a much smaller problem than that in witness reputation because this information is requested far less frequently (each referee is requested to give its rating only once). Moreover, giving such information could be made a standard part of any agreement for task allocation and so agents could be forced to give it. The most notable point in this process is that when agent a makes the trust evaluation, it only involves agents a and b. Since the certified ratings about b are stored by b itself, they are immediately available to a as in the case when a uses its own experience. It should also be noted that when a referee provides references to an interaction partner, it surrenders its privacy with respect to how it values that partner's performance. This may lead to various possible reactions of that partner (e.g. it may retaliate against the referee for a bad reference, it may treat the referee differently the next time to get a better reference). However, due to the vast number of possibilities in the reactions of both

[10] In many scenarios, such as those in the Internet, established service providers (e.g. news service, online merchants) usually have high volumes of interactions (at any time). Therefore, if they adopt the CR process outlined here, we can reasonably expect that such providers will have an abundance of performance ratings readily available.

agents (i.e. the referee and the referred agent), we do not consider the effects of giving up privacy in CR here and defer it to future work.

Having obtained the references of b, a can calculate the CR value of b using the formula in (1). However, since there is no guarantee about the honesty of agents in an open MAS, we need measures to prevent, or to minimise the adverse effects of, lying (e.g. collusion between the target agent and its referees in producing falsely inflated references). Here, we use the rating weight function $\omega_{\mathsf{C}}(r_i)$ to reflect the credibility of a reference (i.e. rating). Again, since we are not considering the problem of lying in this paper, the rating weight function for CR is defined based only on the recency of ratings as per Section 3.3 (i.e. $\omega_{\mathsf{C}}(r_i) = \omega_{\mathsf{I}}(r_i)$). The value of CR, $\mathcal{T}_{\mathsf{C}}(a, b, c)$, and its reliability, $\rho_{\mathsf{C}}(a, b, c)$, are calculated as defined in Section 3.2.

3.7. An overall value

When using FIRE to evaluate trust, an agent can decide which components it will use for trust evaluation according to its needs and situation. However, as each component produces trust values from a separate source of information, we believe that combining the four components, and effectively the four information sources, will in most cases yield a higher level of precision (as confirmed by the empirical evaluation in Section 5.2). Thus, we recommend combining all the aforementioned trust values into a single composite measure to give an overall picture of an agent's likely performance. As all trust values in FIRE come with reliability values, instead of averaging the trust values from the four components, we again use the weighted mean method to calculate the composite trust value, denoted by $\mathcal{T}(a, b, c)$, to take each trust value's reliability into account:

$$\mathcal{T}(a, b, c) = \frac{\sum_{\mathsf{K}\in\{\mathsf{I},\mathsf{R},\mathsf{W},\mathsf{C}\}} w_{\mathsf{K}} \cdot \mathcal{T}_{\mathsf{K}}(a, b, c)}{\sum_{\mathsf{K}\in\{\mathsf{I},\mathsf{R},\mathsf{W},\mathsf{C}\}} w_{\mathsf{K}}} \tag{6}$$

where $w_{\mathsf{K}} = W_{\mathsf{K}} \cdot \rho_{\mathsf{K}}(a, b, c)$, and W_{I}, W_{R}, W_{W}, W_{C} are the coefficients corresponding to the IT, RT, WR, and CR components. These coefficients are set by end users to reflect the importance of each component in a particular application. The reliability of the composite trust value, denoted by $\rho_{\mathcal{T}}(a, b, c)$, is calculated from the components' reliability values in a similar manner:

$$\rho_{\mathcal{T}}(a, b, c) = \frac{\sum_{\mathsf{K}\in\{\mathsf{I},\mathsf{R},\mathsf{W},\mathsf{C}\}} w_{\mathsf{K}}}{\sum_{\mathsf{K}\in\{\mathsf{I},\mathsf{R},\mathsf{W},\mathsf{C}\}} W_{\mathsf{K}}} \tag{7}$$

4. Experimental setup and methodology

In order to empirically evaluate FIRE, we designed a testbed that simulates the relationships and interactions between agents in which trust models are used for selecting interaction partners (Section 4.1). The testbed's environment characterises that of an open MAS. The methodology used for FIRE's evaluation is described in Section 4.2.

4.1. The testbed

The testbed environment for evaluating FIRE is a multi-agent system consisting of agents providing services (called *providers*) and agents using those services (called *consumers*). We assume that the performance of a provider (and effectively its trustworthiness) in a particular service it provides (e.g. news services) is generally independent from that in another service

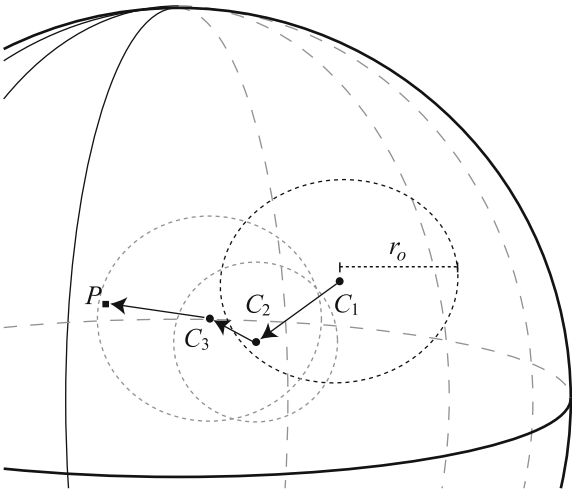

Fig. 4 The spherical world and an example referral chain from consumer C_1 (through C_2 and C_3) to provider P via acquaintances

(e.g. weather services or banking services). Therefore, without loss of generality, and in order to reduce the complexity of the testbed's environment, it is assumed that there is only one type of service in the testbed. Hence, all the provider agents offer the same service. However, their performance (i.e. the quality of the service) differs and determines the utility that a consumer gains from each interaction (called UG). The agents are situated randomly on a spherical world whose radius is 1.0 (see Fig. 4). Each agent has a *radius of operation* (r_o—depicted by a dotted circle around an agent in Fig. 4) that models the agent's capability in interacting with others (e.g. the available bandwidth or the agent's infrastructure) and any agents situated in that range are the agent's acquaintances. In the case of a provider, its radius of operation serves as the normal operational range in which it can provide its service at its full capability without loss of quality. For consumers outside that provider's normal operational range, the quality of service they receive from it gradually degrades. This simulates the phenomenon that each agent usually has particular circumstances (here its location) which affect service delivery. For example, two distant agents may experience significant network latency during their interactions, or a seller agent in the UK may charge another agent extra for shipping goods abroad and the goods may arrive much later than usual.

Simulations are run in the testbed in rounds (of agent interactions). Events that take place in the same round are considered simultaneous. The round number is used as the time value for events. In each round, if a consumer agent needs to use the service it can contact the environment to locate nearby provider agents[11] (in terms of the distance between the agents on the spherical world). The consumer agent will then select one provider from the list to use its service. The selection process relies on the agent's trust model to decide which provider is likely to be the most reliable. Consumer agents without a trust model randomly select a provider from the list. On the other hand, an agent with a trust model selects a provider as follows:

[11] This is to simulate a situation in which only a portion of the provider population is available to a given agent. For example, a retail banking agent can only serve customers in its country. In addition, as the degradation of service quality is proportional to the distance between a provider and its consumer, providers that are too distant may not be useful.

1. It evaluates the trustworthiness of all the providers in the list. Providers whose trustworthiness cannot be determined (due to no available rating) are placed in the set *NoTrustValue*. The rest, whose trustworthiness has been determined, are placed in the set *HasTrustValue*.
2. There can be up to two options available to the agent:
   (a_1) select the provider with the highest trust value in the set *HasTrustValue*, which according to the trust model is likely to yield the highest UG; and
   (a_2) select a random provider from the set *NoTrustValue*, allowing it to learn about the performance of an unknown provider (i.e. exploring the provider population).
3. Obviously, if the set *HasTrustValue* is empty, it can only choose (a_2); if the set *NoTrustValue* is empty, it can only chose (a_1).
4. Otherwise, it needs to determine which action it should take. Choosing (a_2) allows it to explore more about the provider population although it may risk losing utility if it encounters a bad provider. In contrast, choosing (a_1) is likely to give a more predictable value for the expected UG. However, it may not be the optimal performance the agent can get because it has not learnt enough about the provider population. This *exploit-vs-explore* dilemma is addressed in this work by using the Boltzmann exploration strategy [18]. Using this strategy, an agent tends to explore its environment first and then gradually move its stance towards exploitation when it learns more about the environment. Thus, the agent chooses an action a_k with the probability of

$$P(a_k) = \frac{e^{\frac{ER(a_k)}{T}}}{\sum_{a_i} e^{\frac{ER(a_i)}{T}}} \tag{8}$$

   where $ER(a_i)$ is the expected return from choosing action a_i, T is the temperature parameter which is set to decrease over time to decrease exploration. In brief, the probability that an action a_k is selected is biased by the expected return of that action. Moreover, when an agent's level of exploration is decreased (by decreasing T over time) the action with the highest expected return will be more likely to be selected (i.e. the agent is more likely to exploit the knowledge it has learnt about the performance of provider agents). Here, the expected return for (a_1) is the expected UG of the highest trusted provider as calculated from its trust value, and that for (a_2) is the average UG of the provider population that has been observed by the consumer agent.

Having selected a provider, the consumer agent then uses the service of the selected provider and gains some utility from the interaction (UG). The value of UG is in $[-10, 10]$ and depends on the level of performance of the provider in that interaction. A provider agent can serve many users at a time. As in realistic situations, a consumer agent, however, does not always use the service in every round. The probability it needs and requests the service, called its *activity level* and denoted by α, is selected randomly when the consumer is created. In other words, the activity level of a consumer determines how frequently it uses the service.

After an interaction, the consumer agent rates the service of the provider based on the level of performance, or the quality of the service, it received. It records the rating for future trust evaluations and also informs the provider about the rating it made. The provider may record the rating as evidence about its performance to be presented to potential consumers. As we have previously discussed, it is assumed that all agents exchange their information honestly in this testbed. This means an agent (as a witness or as a referee) provides its true ratings as they are without any modification.

In our testbed the only difference in each situation is the performance of the provider agents. We consider four types of provider agents: good, ordinary, bad, and intermittent.

Each of them, except the last, has a mean level of performance, denoted by μ_{P}. Its actual performance follows a normal distribution around this mean. The values of μ_{P} and the associated standard deviation of these types of providers, denoted by σ_{P}, are given in Table 2. Intermittent providers, on the other hand, yield unpredictable (random) performance levels in the range [PL_BAD, PL_GOOD]. In addition, if a consumer agent is situated outside of the provider's normal operational range (i.e. r_o) the service quality of that provider is also degraded linearly in proportion to the distance between the provider and the consumer.

Since agents can freely join and leave an open MAS, the agent population can be very dynamic. Moreover, since agents are owned and controlled by various stakeholders, the performance of an agent may not be consistent over time (i.e. its performance may change according to its own situation). Therefore, in order to simulate such dynamism, we introduce the following factors in our testbed:

– *The population of agents*: In an open MAS, agents can come and leave the system at anytime. This is simulated by removing a number of randomly selected agents from the testbed and adding new ones into it. The numbers of agents added and removed after each round vary, but have an upper limit of some predefined percentage of the whole population. The population change limits for the consumer and the provider populations are denoted, respectively, by p_{CPC} and p_{PPC}. Since providers are usually more established than consumers, p_{PPC} is set to be lower than p_{CPC} in our simulations. The characteristics of the newly added agents are set randomly but they are uniformly distributed over the initial agent populations (i.e. the proportions of providers of different profiles and that of consumers in different groups are maintained).
– *The locations of agents*: During their life cycle, agents break old relationships and make new ones (reflecting the notion of continual change that is inherent in open MAS). In our testbed, this type of change is reflected by the change in an agent's location on the spherical world. When a consumer changes its location, it will have a new set of acquaintances according to its r_o. In addition, the location of an agent in the testbed also reflects its individual situation covering things such as its knowledge about other local agents (see Section 3.5) and the service delivery between providers and consumers (see above). Therefore, changing an agent's location changes its relationships with others, as well as its individual situation. Specifically, we use polar coordinations (r, φ, θ) for agent locations on the spherical world. Then in order to change an agent's location, amounts of angular changes $\Delta\varphi$ and $\Delta\theta$ are added to φ and θ, respectively. In this case, $\Delta\varphi$ and

Table 1 Performance level constants

Performance level	Utility gained
PL_PERFECT	10
PL_GOOD	5
PL_OK	0
PL_BAD	−5
PL_WORST	−10

Table 2 Profiles of provider agents (performance constants defined in Table 1)

Profile	Range of μ_{P}	σ_{P}
Good	[PL_GOOD, PL_PERFECT]	1.0
Ordinary	[PL_OK, PL_GOOD]	2.0
Bad	[PL_WORST, PL_OK]	2.0

$\Delta\theta$ are selected randomly in $[-\Delta\phi, +\Delta\phi]$. Thus, $\Delta\phi$ limits the variability of agents' locations. Not every agent changes its locations every round and, in particular, p_{CLC} and p_{PLC} are used to denote the probabilities that a given consumer or provider, respectively, changes its location in a round.

– *The behaviour of the providers*: In many environments, provider performance may alter (for better or worse) over time. A provider may even change its behaviour completely (e.g. a provider may take advantage of its good reputation and decide to perform selfishly to obtain better utility). In our testbed, the average performance of a provider (μ) can be changed by an amount of $\Delta\mu$ randomly selected in $[-\mathsf{M}, +\mathsf{M}]$, and this happens in each round with the probability of $p_{\mu\mathsf{C}}$. Moreover, after each round, a provider can switch to a completely new provider profile with a probability of $p_{\mathsf{ProfileSwitch}}$.

The changes above to the testbed's environment (if in effect) are applied only after each round of interactions finishes. The nature and degree of dynamism are specified in each experiment.

4.2. Experimental methodology

In each experiment, the testbed is populated with provider and consumer agents. Each consumer agent is equipped with a particular trust model, which helps it select a provider when it needs to use a service. Since the only difference among consumer agents is the trust models that they use, the utility gained by each agent through simulations will reflect the performance of its trust model in selecting reliable providers for interactions. Therefore, the testbed records the UG (Section 4.1) of each interaction along with the trust model used. In order to obtain an accurate result for performance comparisons between trust models, each one will be employed by a large number of consumer agents ($\mathsf{N_C}$). In addition, the average UG of agents employing the same trust models (called consumer groups) are compared with each other's using the two-sample t-test (for means comparison) [6] with the confidence level of 95%[12]. The result of an experiment is then presented in a graph with two y-axes; the first plots the UG means of consumer groups in each interaction and the second plots the corresponding performance rankings obtained from the UG means comparisons using the t-test. The plot for the performance ranking of a group is also denoted by the group's name but prefixed by 'R.' Here, we specify that the group of a higher rank outperforms the one of a lower rank and groups of equal rank have insignificantly different performance. For example, in Fig. 5, at the 5th interaction (on the x-axis), agents in group FIRE obtain an average UG of 5.06 (reading on the left y-axis) and, according to the t-test ranking, the rank of FIRE (as shown by the plot R.FIRE) is 3 (reading on the right y-axis).

The experimental variables are presented in Table 3 and their values will be used in all cases unless otherwise specified. Although a "typical" provider population may differ in various applications, the space of possibilities is vast and exploring it completely would be impossible. Therefore, we choose provider populations which we believe are more common than others for our experiments. Here, we consider a typical provider population to consist of about half profitable providers (i.e. yielding positive UG) and half exploiting providers (i.e. yielding negative UG, including intermittent providers). However, good and intermittent

[12] In analysing (experimental) data about two populations, say their income levels, the fact that the means of the two sample groups' incomes is different does not always mean that the two populations have different levels of income if randomness can affect sample selection. It is possible that the means of these two particular samples are different, but the means of the two populations' incomes are not. Thus the t-test is a statistical method that allows us to confirm within a predefined confidence level whether the difference of the two means actually means one group has higher income than the other, eliminating the random factor in selecting the samples (see [6] for more detail).

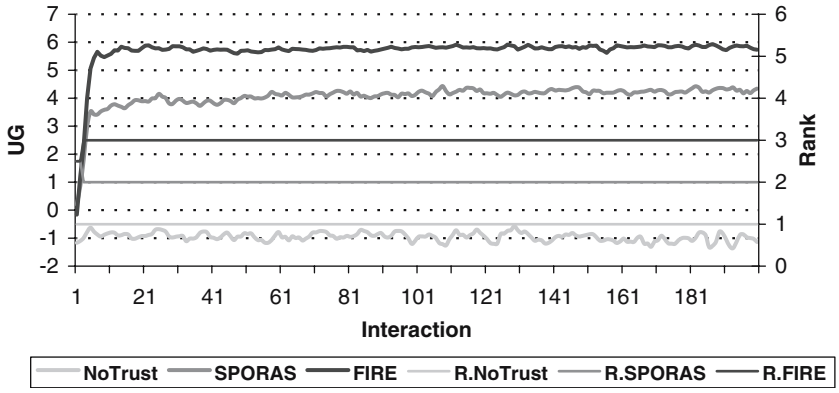

Fig. 5 Overall performance of FIRE in the typical provider population

Table 3 Experimental variables

Simulation variable	Symbol	Value
Number of simulation rounds	N	500
Total number of provider agents:	N_P	100
+ Good providers	N_{PG}	10
+ Ordinary providers	N_{PO}	40
+ Intermittent providers	N_{PI}	5
+ Bad providers	N_{PB}	45
Number of consumer agents in each group	N_C	500
Range of consumer activity level	α	[0.25, 1.00]

providers are usually exceptional cases and, thus, they take only a small portion of each half. Except in the experiments where we evaluate FIRE with different provider populations, this typical provider population is used throughout.

Here, we also show the default parameters of FIRE for the experiments in Table 4. The component coefficients are set to reflect our view on the reliability of the information source used by each component. The IT component deduces trust from ratings in which the agent does the rating itself and, thus, is more reliable than the WR and CR components, which use information from third-parties.[13] We expect the CR information to exaggerate an agent's true performance, hence, the CR component has the lowest reliability. The RT component provides rules encoding knowledge and beliefs about the agent's environment to customise the trust model, so it should also have a high reliability. Given the time unit used in the test bed (round of interactions), we set the IT recency scaling factor such that a 5-round old rating has half (0.5) the effect of a new rating (1.0).

5. Empirical evaluation

Having presented the testbed and the methodology for FIRE's evaluation, we now turn to the experiments themselves. In particular, we concentrate on the benefit of using FIRE for

[13] Because the situation of each agent can differ from that of another, different agents can have different observations about the performance of a particular provider. For example, in our testbed, from the same provider, the quality of the service a consumer receives depends on the distance between it and the provider. Therefore, because of such differences, even when witnesses are honest (as is assumed in this paper), their reports are not always as useful as an agent's own observations.

Table 4 FIRE's default parameters

Parameters	Symbol	Value
Local rating history size	H	10
IT recency scaling factor	λ	$-(5/\ln(0.5))$
Branching factor	n_{BF}	2
Referral length threshold	n_{RL}	5
Component coefficients		
+ Interaction trust	W_{I}	2.0
+ Role-base trust	W_{R}	2.0
+ Witness reputation	W_{W}	1.0
+ Certified reputation	W_{C}	0.5
Reliability function parameters		
+ Interaction trust	γ_{I}	$-\ln(0.5)$
+ Role-base trust	γ_{R}	$-\ln(0.5)$
+ Witness reputation	γ_{W}	$-\ln(0.5)$
+ Certified reputation	γ_{C}	$-\ln(0.5)$

selecting interaction partners with different provider populations (Section 5.1) and the contributing values of its individual components (Section 5.2). We also explore the impact of various dynamic factors on the performance of FIRE (Section 5.3).

5.1. Overall performance of FIRE

In order to evaluate the overall performance of FIRE, we compare it with the SPORAS model[14] (whose operation is described in Section 2.2) and a group of agents with no trust model. Hence, there are three groups of consumer agents: FIRE, SPORAS, and NoTrust. The first thing to test is whether FIRE helps consumer agents select profitable providers (i.e. those yielding positive UG) from the population and, by so doing, helps them gain better utility than without FIRE (i.e. the NoTrust group). In this section, the testbed's environment is static, which means that no dynamic factors are in effect.

Here, Fig. 5 shows that the NoTrust group, selecting providers randomly without any trust evaluation, performs consistently the lowest (as we would expect). On the other hand, both SPORAS and FIRE prove to be beneficial to consumer agents, helping them to obtain significantly higher UG. This shows that the tested trust models can learn about the provider population, and allow their agents to select profitable providers for interactions. However, the chart, as well as the *t*-test ranking, also shows that FIRE outperforms SPORAS, the second rank, throughout the interactions by about 2 UG units. This is despite the fact that SPORAS, being a centralised model, gathers much more information than FIRE (a decentralised model).[15] The performance difference of FIRE and SPORAS is accounted for by the fact that FIRE separates direct experiences from others' experiences (i.e. ratings) in trust

[14] SPORAS is chosen as the control benchmark for two reasons. First, it is a successful independently developed trust model that several other researchers have used for benchmarking (e.g. [4], [27]). Second, other than SPORAS, most other notable trust models make assumptions that are incompatible with open MAS, or require additional knowledge, and, thus, they will not operate as intended in our testbed.

[15] After every interaction, the consumer reports its rating about the provider's service in that interaction to SPORAS. Therefore, as a centralised service, SPORAS collects all the available ratings from its users. In contrast, consumers employing FIRE have only ratings from a limited set of witnesses (from the WR component) and those presented by providers (from the CR component) in addition to their own ratings. Typically in our experiments, after the first 10 rounds the average number of ratings taken into account in each trust evaluation request to FIRE is 3.28, and that to SPORAS is 15.55. After 20 rounds the corresponding numbers

Table 5 The performance of SPORAS and FIRE in the first 10 interactions

Interaction:	1	2	3	4	5	6	7	8	9	10
SPORAS	0.20	0.85	1.80	2.96	3.53	3.42	3.42	3.52	3.58	3.62
FIRE	−0.16	1.20	2.30	4.00	5.06	5.44	5.66	5.52	5.47	5.53

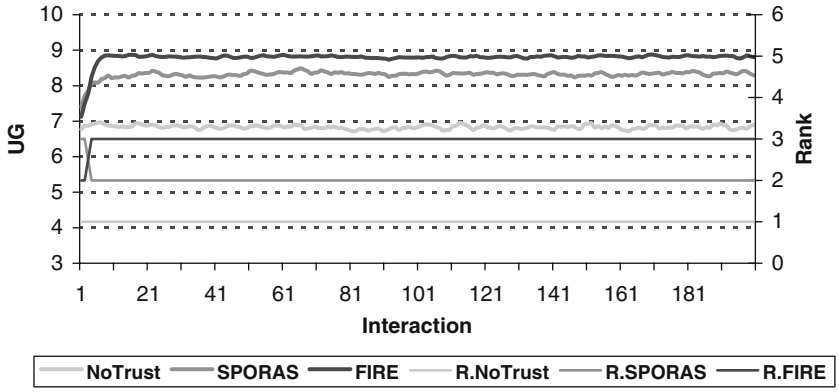

Fig. 6 Overall performance of FIRE—100% good providers

evaluation, while SPORAS treats all types of ratings equally. Therefore, SPORAS suffers from noise in ratings (resulting from different degrees of degradation of service quality due to different provider-consumer distances). In contrast, FIRE reduces rating noise by giving more weight to direct experiences (see Table 4), which are more relevant to an individual agent's situation. Another noticeable point is that in the first few interactions, FIRE can learn about the providers quicker than SPORAS as the FIRE group achieves its superiority from the first interaction (see Table 5) despite much less information being available to it. As we show in Section 5.2, this is achieved thanks to the WR and, in particular, the CR components.

We reran the same experiment but with provider populations consisting of providers of only one profile (e.g. good, ordinary, bad, and intermittent) to see how different types of providers may affect FIRE's performance. Specifically, the experiment is run with 100 good providers, then with 100 ordinary providers, 100 bad providers, and 100 intermittent providers. The result in the case of intermittent providers is not shown here because the performance of all three groups is indistinguishable; fluctuating randomly in $[-1, 0]$ (this is expected because of the random behaviour of intermittent providers). In the rest of the experiments, we observe similar results (see Figs. 6–8) to that in our first experiment with the typical provider population. FIRE maintains its superiority in all the three types of provider population (good, ordinary, and bad). This suggests that FIRE can work well in a wide range of provider population.

In sum, through the experiments on FIRE's overall performance, we confirm that FIRE is beneficial to agents in selecting interaction partners in the various provider populations. In addition, despite being decentralised and having less information than SPORAS, FIRE outperforms the model in all the cases thanks to its differential treatment of each source of trust information.

are 4.05 and 28.47, respectively. This suggests that FIRE may be advantageous in environments in which rating information costs some premium.

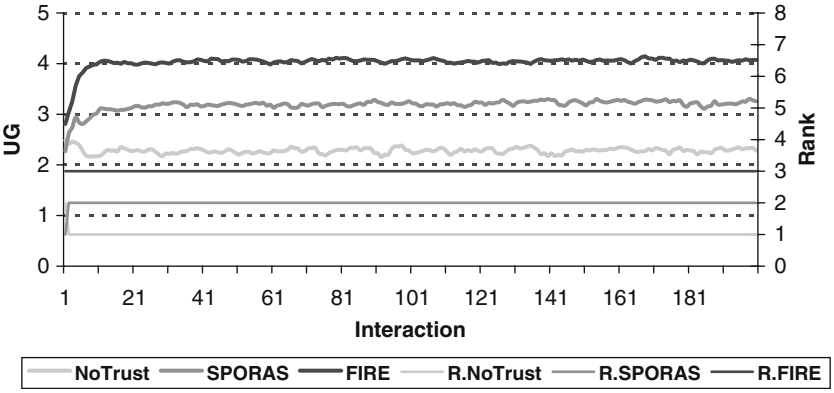

Fig. 7 Overall performance of FIRE—100% ordinary providers

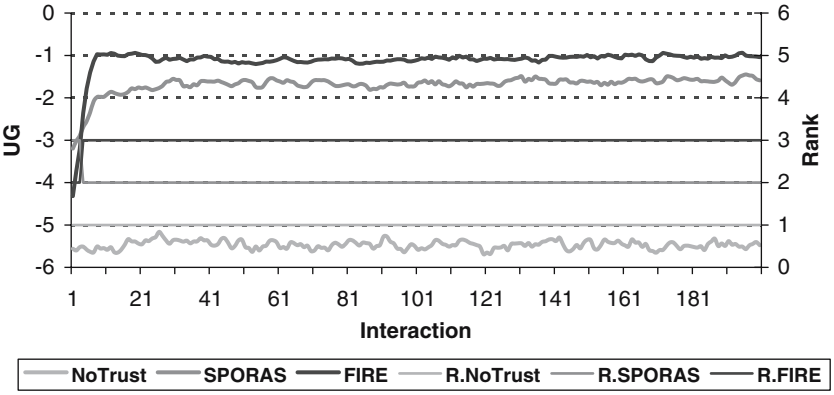

Fig. 8 Overall performance of FIRE—100% bad providers

5.2. Performance of FIRE's components

We argued that each component of FIRE plays an important role in exploiting trust information from a particular source and this, in turn, contributes to the effectiveness of the overall model. In order to confirm this, we benchmark FIRE with and without various components to evaluate the contribution of that component to the whole model. However, since the IT component is mostly reused from Regret, we will only focus on evaluating the novel components (i.e. WR and CR). Role-based trust is not considered here because it is typically highly domain specific. Experiments in this section evaluate the WR and CR components with the typical provider population in a static environment.

First, we benchmark the WR component. In this experiment, there are two groups of consumer agents. The first one uses only the IT component[16] (called the Control group). The

[16] It should be noted that if Regret is employed in our testbed, its performance will be similar to that of FIRE's IT component only. This is due to the fact that there is no information about a social network of the agents in the testbed available for Regret. Therefore, other than the direct trust component, the other components of Regret will not be able to work due to a lack of supporting information (e.g. its witness reputation component cannot locate witnesses, and the neighbourhood reputation component cannot locate neighbouring agents of the target agent).

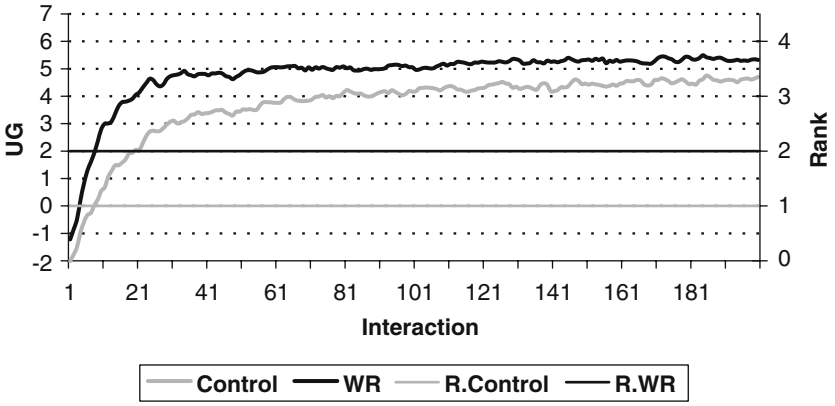

Fig. 9 Performance of the WR component

second makes use of the WR component in addition to the IT component (called the WR group). This experiment's hypothesis is that the performance of the WR should be higher than that of the Control group.

The result of the experiment, presented in Fig. 9, shows that the WR component substantially improves the performance of agents in the WR group compared to that of those in the Control group. The t-test ranking confirms this by showing that agents using the WR component outperform agents using only the IT component in all interactions, and, effectively, also verifies our hypothesis. More importantly, the WR group achieves its higher performance quicker than the Control group. This means that WR speeds up an agent's learning about its environment by propagating trust in the agent's community (here the community of consumer agents is connected to one another via acquaintances).

In the next experiment we evaluate the CR component (using a similar setting). Here, there are two groups of consumer agents. The Control group employs the IT component, and the other employs the CR component in addition (called the CR group). The hypothesis in this experiment is that the CR group should outperform the Control group. The result presented in Fig. 10 shows a similar result to that of the previous experiment. Hence, all of the claims made above for the WR component are still valid for the CR component. However, the most noticeable thing about this experiment is its execution time. Without having to look for witnesses as in the process of WR, the process of CR is more direct, resulting in an execution time for this experiment being about 15 times faster than that of the previous one. This confirms our intuition about the high serviceability of the CR component. A subtler point shown in this experiment is the quick learning time of the CR group. Comparing the first interactions of the WR group in Fig. 9 with those of the CR group, Fig. 10 shows that the CR group starts off better than the WR group right from the first few interactions. In order to verify this observation, eliminating the random factor that may affect the results of the two independent experiments above, we ran another experiment to compare the performance of FIRE with and without the CR component. In this experiment, there are also two groups of consumer agents: the WR group employing the IT and WR components and the FIRE group employing the IT, WR, and CR component. Our hypothesis is that the addition of the CR component to FIRE is beneficial, or, equivalently, the FIRE group should outperform the WR group. The result in Fig. 11 shows that the performance of FIRE is indeed always higher than that of the WR group. In more detail, presented in Table 6, the FIRE group's performance

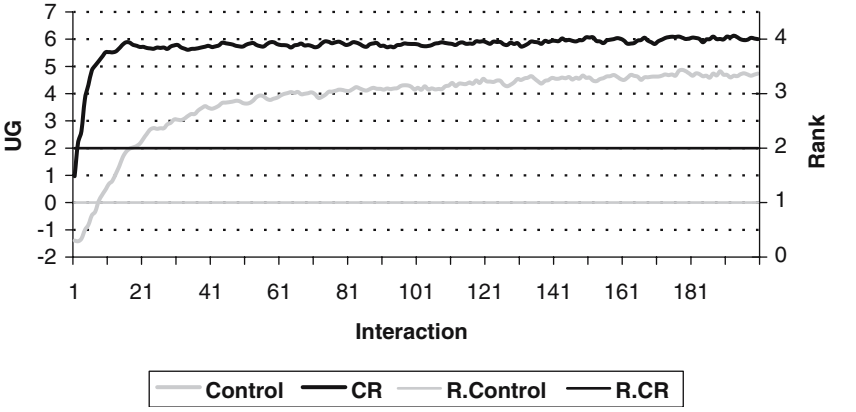

Fig. 10 Performance of the CR component

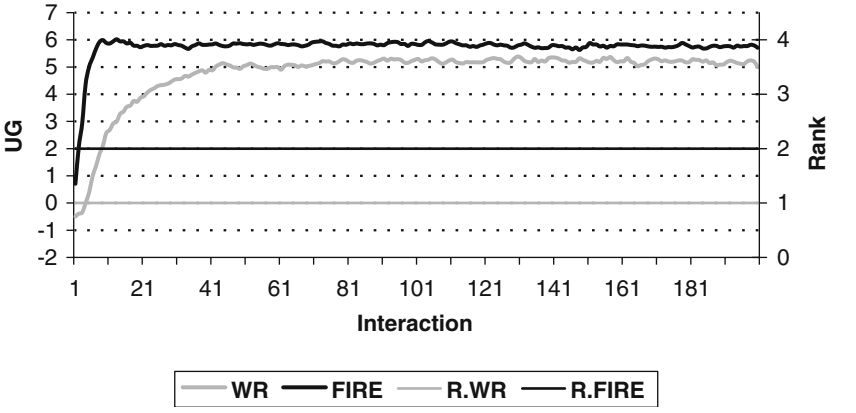

Fig. 11 Performance of FIRE with and without the CR component

Table 6 The performance of WR and FIRE in the first 10 interactions

Interaction:	1	2	3	4	5	6	7	8	9	10
WR	−0.50	−0.39	−0.35	0.01	0.40	0.99	1.36	1.80	2.10	2.57
CR	0.71	2.08	2.94	4.38	5.02	5.34	5.67	5.92	5.99	5.89

reaches its stable level of around 6.0 in only eight interactions, while that of the WR group only reaches 2.57 after 10 interactions. This shows that the CR process propagates trust in an agent community more quickly than the WR process. Taking into consideration its very quick execution time, compared to that of the WR component in the previous experiments, the CR component is clearly useful in situations where an agent needs to have a quick trust evaluation in order to expedite decisions, or when WR information is scarce and difficult to locate. We conclude that the addition of the CR component to FIRE is beneficial both in terms of its robustness (reflected by its higher level of performance) and its serviceability.

In summary, it has been shown that taking various sources of trust information into account not only helps FIRE be able to make trust evaluations in a wide variety of situations, but also increases its precision; and that all the components contribute a significant amount to its overall performance.

5.3. The effects of dynamism

The environment of a realistic open MAS is always changing because of its openness. Hence, a trust model designed for open MAS should be able to function properly in such a dynamic environment. This section concentrates on testing the hypothesis that FIRE still maintains its properties (i.e. being beneficial to agents in selecting interaction partners) in a changing environment. Similarly to the experiments in Section 5.1, there are three groups of consumer agents in the experiments: NoTrust, SPORAS, and FIRE. The provider population is the typical one. Each experiment will test the hypothesis with only one dynamic factor in effect (see Section 4.1). Specifically, the same experiments will be run but with each of the following conditions:[17]

1. The provider population changes at maximum 2% every round ($p_{\mathsf{PPC}} = 2\%$).
2. The consumer population changes at maximum 5% every round ($p_{\mathsf{CPC}} = 5\%$).
3. A provider may alter its average level of performance at maximum 1.0 UG unit with a probability of 0.10 each round ($p_{\mu\mathsf{C}} = 0.10$, $\mathsf{M} = 1.0$).
4. A provider may switch into a different (performance) profile with a probability of 2 each round ($p_{\mathsf{ProfileSwitch}} = 0.02$).
5. A provider may move to a new location on the spherical world at a maximum angular distance of $\frac{\pi}{20}$ with a probability of 0.10 each round ($p_{\mathsf{PLC}} = 0.10$, $\Delta\phi = \frac{\pi}{20}$).
6. A consumer may move to a new location on the spherical world at a maximum angular distance of $\frac{\pi}{20}$ with a probability of 0.10 each round ($p_{\mathsf{CLC}} = 0.10$, $\Delta\phi = \frac{\pi}{20}$).

These experiments are named Experiment 1–6, respectively and their results are shown in Figs. 12–17. Since the NoTrust group still has the lowest performance, we omit its results from the charts for the sake of simplicity. A general observation from all the results is that both FIRE and SPORAS still maintain positive UG from about 3.0–6.0 (except SPORAS in Experiment 4, Fig. 15). However, dynamism, as it introduces noise to the environments, adversely affects the performance of both of them in all the experiments reported here. Specifically, and as we would expect, their performance is lower than that in the static environment (Fig. 5) and the performance plots also evolve differently over time. Nevertheless, although having lower levels of performance than in a static environment, the shape of FIRE's performance plots are generally maintained after they reach their stable level in the first few interactions in all the experiments. This shows that FIRE is able to maintain a stable performance regardless of the various types of changes in the environment. In other words, FIRE can learn about the changes and adapt quickly to them.

In more detail, the experiments here can be put into two categories: dynamism on the consumer side (Experiments 2 and 6), and dynamism on the provider side (Experiments 1, 3–5). In the first group, the results (Figs. 13 and 17) show that SPORAS can cope well with these types of changes. This is because SPORAS collects ratings centrally from all consumers and, thus, small changes on the consumer side do not have a significant impact on its performance as its learned knowledge about the provider population is still useful. Particularly in Experiment 2, where new consumers are added to the testbed, newly joined agents using

[17] These are what we consider to be reasonable values of variation. We have conducted similar experiments with both greater and lesser degrees of dynamism and we see the same broad trends as we report here.

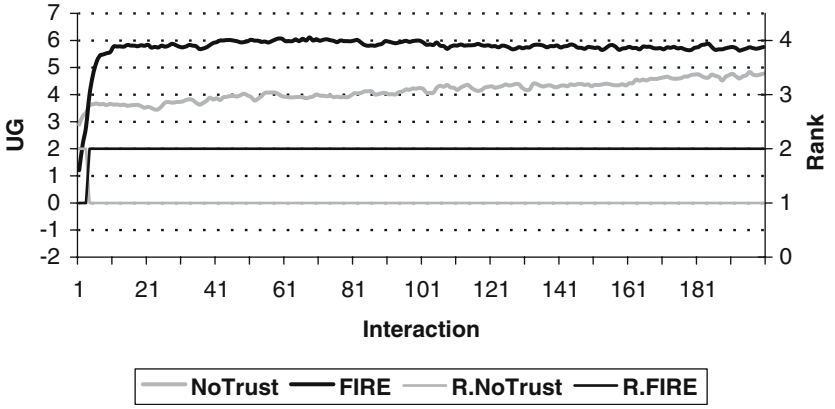

Fig. 12 Experiment 1: Provider population change: $p_{\mathsf{PPC}} = 2\%$

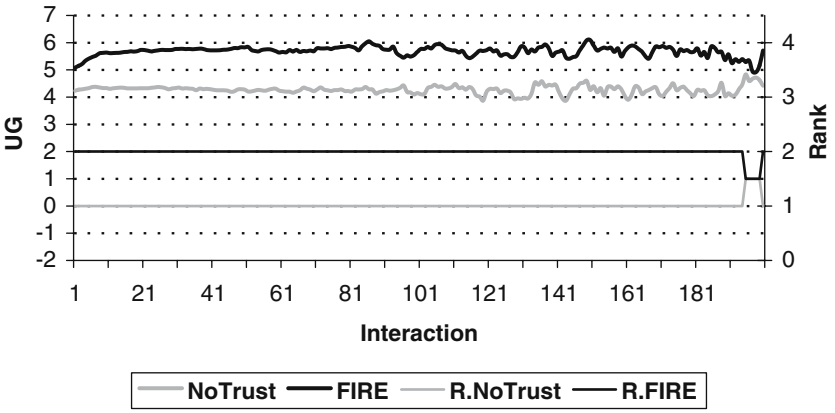

Fig. 13 Experiment 2: Consumer population change: $p_{\mathsf{CPC}} = 5\%$

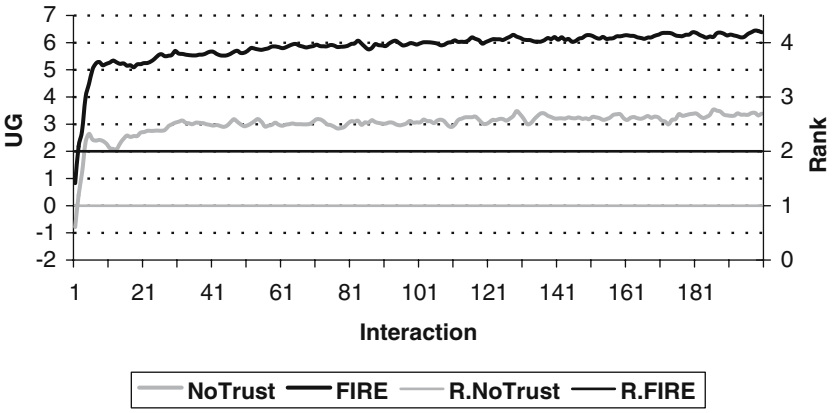

Fig. 14 Experiment 3: Providers change their performance: $p_{\mu\mathsf{C}} = 10\%, M = 1.0$

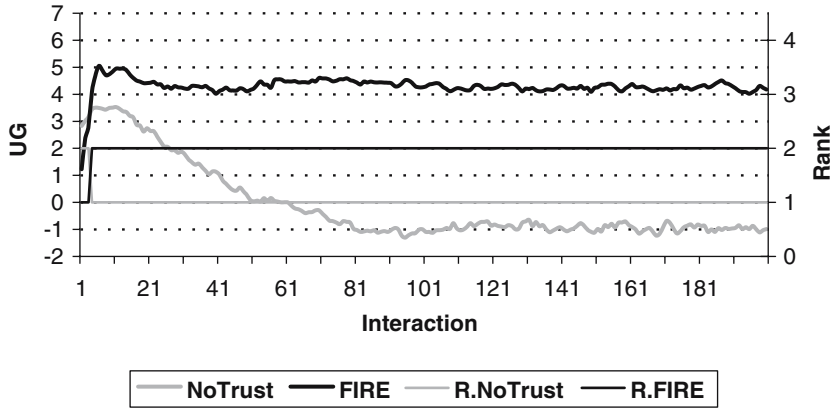

Fig. 15 Experiment 4: Providers switch their profiles: $p_{\mathsf{ProfileSwitch}} = 2\%$

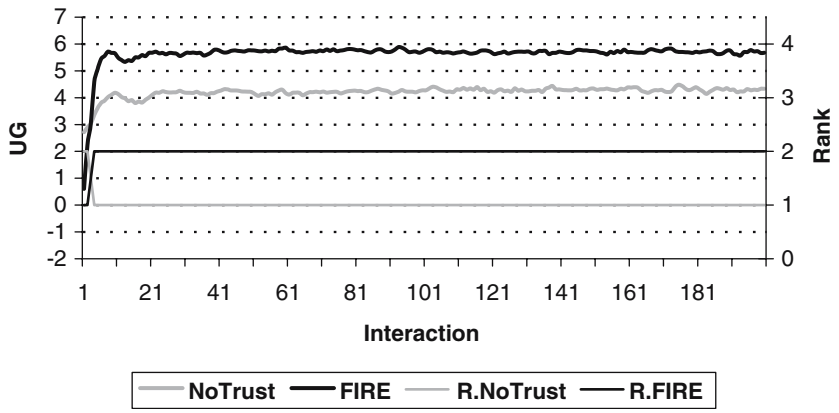

Fig. 16 Experiment 5: Providers change their locations: $p_{\mathsf{PLC}} = 10\%$, $\Delta\phi = \pi/20$

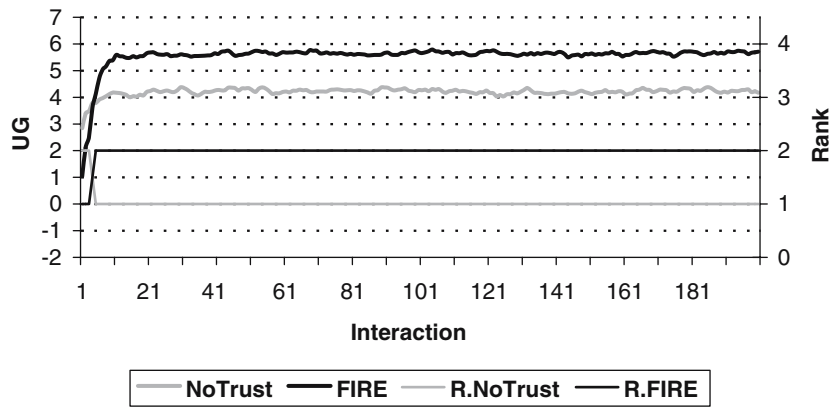

Fig. 17 Experiment 6: Consumers change their locations: $p_{\mathsf{CLC}} = 10\%$, $\Delta\phi = \pi/20$

SPORAS take advantage of the existing knowledge of the centralised model and perform highly right from the start. In contrast, FIRE relies on the consumer community for witness reputation and, thus, has a slightly lower performance than that in the static environment. Nevertheless, it still outperforms SPORAS in Experiments 2 and 6.

The situation is different in the experiments of the second category. SPORAS's performance is significantly affected when providers change their performance levels (Experiments 3 and 4, whose results are shown in Figs. 14 and 15), most noticeably in Experiment 4, where providers switch their performance profiles completely. In this experiment, although FIRE is also affected greatly by the steep changes in the provider population, it still maintains a generally high and stable performance, while SPORAS's performance degrades disproportionately to that of the NoTrust group. It should be noted that the trust models' duty here is to learn and predict the behaviour (i.e. the performance) of providers and, therefore, their performance in an environment where there are changes on the provider side reflects their ability to adapt to such changes. Hence, the results suggest that SPORAS can quickly learn about the environment (because of its centralised nature) but has difficulty adapting to the continual changes of the providers. In Experiments 1 and 5 (Figs. 12 and 16), where the provider population changes and where the providers move around, respectively, the performance of FIRE and SPORAS seem to be only slightly affected. In general, FIRE performs consistently in all these experiments. Its average UG is stable around 6.0 in Experiments 1, 3, and 5; and around 4.0 in Experiment 4 (which has most abrupt changes). This confirms our intuition that the recency function of FIRE helps it adapt quickly to changes in the environment.

Since a realistic open MAS usually has a combination of all the dynamic factors considered here, we also want to test how FIRE performs in such situations. Therefore, we ran an additional experiment with all the dynamic factors active at the same time. The result, in Fig. 18, shows that although both FIRE and SPORAS suffer from the continual changes of various types in the testbed's environment, FIRE manages to maintain a rather stable performance in the range [4.0, 5.0]. SPORAS, however, cannot cope with all the types of changes at the same time and its performance degrades dramatically and is at the same level as that of the NoTrust group through all interactions. This again confirms the adaptability of FIRE in a complex dynamic environment.

In summary, the experiments in this section show that FIRE is able to perform consistently in various dynamic environments, maintaining a high level of utility gain for its agents (in all the experiments).

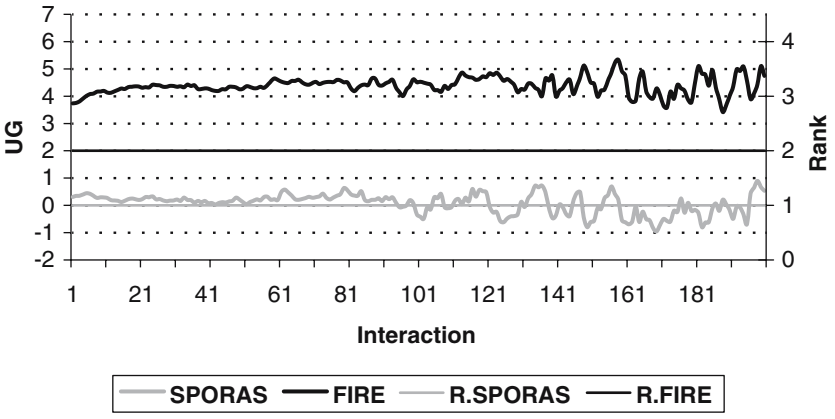

Fig. 18 Experiment 7: Performance of FIRE in an environment where all dynamic factors are in effect

6. Conclusions and future work

This paper has presented FIRE, a novel decentralised model for trust evaluation that is specifically designed for general applications in open MAS. By so doing, we also introduce a generic framework which allows a variety of sources of trust information to be integrated to provide a collective and precise trust measure (i.e. able to predict closely the behaviour of an agent). These sources include: direct experiences of an agent from its interactions, witness reports, third-party references, and rules provided by end users encoding beliefs or knowledge about the environment. In particular, we introduce a novel type of reputation based on third-party references called Certified Reputation. The addition of this new type of reputation greatly enhances the serviceability of our trust model, allowing a trust measure to be available in most circumstances. Moreover, our proposed framework can easily be customised via its various parameters to suit a particular application domain. In short, we believe FIRE can provide a trust measure that is sufficiently precise to be used in a wide range of situations and applications in open MAS.[18]

In more detail, through empirical evaluation, we have shown that:

- Agents using the trust measure provided by FIRE are able to select reliable partners for interactions and, thus, obtain better utility gain compared to those using no trust measure. This result was reconfirmed with various types of provider population.
- Each component of FIRE plays an important role in its operation and significantly contributes to its overall performance.
- FIRE is able to cope well with the various types of changes in an open MAS and can maintain its properties despite the dynamism possible in an environment.
- Although decentralised, to suit the requirements of a trust model in open MAS, FIRE still outperforms or maintains a comparable performance level with SPORAS, a centralised trust model.

In sum, FIRE satisfies the first two requirements for a trust model in open MAS as specified in Section 1. However, at present, FIRE assumes the agents report their trust information truthfully. As noted in the requirements, this is not suitable for our target domain. As the WR and CR components are based on third-party reports, they are susceptible to false or inaccurate information, whose existence is unavoidable in open MAS. For this reason, we plan to devise reliability measures for witness ratings and certified ratings (i.e. $\omega_{\mathsf{W}}(r_i)$ and $\omega_{\mathsf{C}}(r_i)$) that determine and take into account the credibility of third-party information. These will allow the general trust formula in (1) to weight ratings by their credibility and to filter out those that deem to be completely inaccurate. This will make FIRE robust against strategic disinformation behaviours in reporting trust information as well as inherent inaccuracy. In addition, as noted in Section 3.6, giving references in our CR component also means giving up an agent's privacy of how it values others' services and the effects of this can be very complex. Therefore, we plan to investigate this problem thoroughly in order to avoid undesirable effects on the performance of CR. After having been able to deal with such obstacles, FIRE will be ready to be used in real open MAS applications.

[18] Obviously, there are still cases when FIRE cannot produce a trust value. Specifically, those are when a service provider newly joins the system. Hence, it does not have references about its performance and other agents do not have past experience with it. However, in a realistic scenario, in order to promote its service, that provider can join a (popular) scheme/organisation that provides quality assurance about its members' service. For example, a car dealer can obtain the title "authorised dealer" from a car manufacturer. Such (popular) membership (and inherently its quality assurance) can be recognised by other agents (via rules in FIRE's RT component) and thus helps the provider to sell its service.

In terms of improving FIRE's overall performance, we also plan to incorporate learning abilities. At present, FIRE is a static parametric model (i.e. all of its parameters need to be set by users in order to suit a particular application domain). This is clearly limiting and so we aim to study which of FIRE's parameters can be adjusted dynamically to adapt it to changes in an agent environment. For instance, if the number of lying agents in its environment increases, an agent may reduce the component coefficient of witness and certified reputation (W_{W} and W_{C}, see Section 3.7); or if its environment changes too quickly (e.g. the agents alter their behaviours frequently), it can reduce the local rating history size H to discard older (and likely out-of-date) ratings.

Appendix A

A Regret's direct trust

In Regret's direct trust component, each agent rates its partner's performance after every interaction and records its ratings in a local database. The relevant ratings will be queried from this database when trust evaluation is needed. In order to calculate the direct trust of agent b, agent a retrieves its past ratings about b's performance. The set of ratings is called R[19]. Then the direct trust of a to b, denoted by $DT_{a\to b}$, is calculated as follows:

$$DT_{a\to b} = \sum_{r_i \in R} \rho(t, t_i) \cdot r_i \tag{A.1}$$

where r_i is a rating value in the set $\mathcal{R}$, and $\rho(t, t_i)$ is a normalised weight value that gives higher values to ratings of more recent interactions:

$$\rho(t, t_i) = \frac{f(t_i, t)}{\sum_{r_i \in R} f(t_i, t)} \tag{A.2}$$

$$f(t_i, t) = \frac{t_i}{t} \tag{A.3}$$

where t is the current time and t_i is the time the rating r_i is recorded. However, this weight function has some shortcomings on time granularity control. Actually, after being factored, (A.2) is equivalent to:

$$\rho(t, t_i) = \frac{t_i}{\sum_{r_i \in \mathcal{R}} t_i} \tag{A.4}$$

Therefore, the weight function depends only on the time values of a particular set of ratings, rather than the recency of those ratings in comparison with the current time t. For example, assume that r_1 and r_2 are ratings about the interactions that took place at time $t_1 = 1$ and $t_2 = 2$, respectively, and that $R = \{r_1, r_2\}$. Equation A.4 will give $\rho(t, t_1) = 1/3$ and $\rho(t, t_2) = 2/3$. Hence the difference of the weights for r_1 and r_2 is $1/3$ in the interaction trust calculation. Suppose that r_1' and r_2' are completely the same ratings to r_1 and r_2 except that $t_1' = 100$ and $t_2' = 101$. The same calculations will give a weight difference that is now $1/10100$. This vast change in weight differences is not desirable given that the rating time difference in both cases is the same (1 time unit, e.g. a second). Moreover, given the same

[19] Due to the limited scope of this paper, Regret and its notation are simplified here. However, the main ideas of the model are still retained.

set of ratings, the weight function produces the same value regardless of the current time t[20] (i.e. the time of calculating the interaction trust).

Every trust value in Regret comes with a reliability value that reflects the confidence of Regret in that trust value. This reliability value is calculated from a combination of two measures that are based on the number of ratings in the set R and the deviation of those ratings. Regret defines an intimate level of interactions, denoted by itm, that represents the minimum number of ratings needed for a close relationship. The reliability degree increases until $|R|$ reaches this number. After that, more interactions will not increase reliability. The measures (No for number of ratings and Dv for deviation of ratings) and the reliability for direct trust, denoted by $DTRL_{a\to b}$ are specified in the following formula:

$$No(R) = \begin{cases} \sin\left(\frac{|R|\cdot\pi}{2\cdot itm}\right) & |R| \le itm \\ 1 & |R| > itm \end{cases} \tag{A.5}$$

$$Dv(R) = \sum_{r_i \in R} \rho(t, t_i) \cdot |r_i - DT_{a\to b}| \tag{A.6}$$

$$DTRL_{a\to b} = No(R) \cdot (1 - Dv(R)) \tag{A.7}$$

Acknowledgements We would like to acknowledge all three reviewers of this article for their excellent comments that enabled us to significantly improve this paper. This paper is a significantly extended version of [14] and [15]. This work is partly supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01.

References

1. Abdul-Rahman, A., & Hailes, S. (2000). Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii international conference on system sciences*. Vol. 6., IEEE Computer Society Press.
2. Amazon Site. (http://www.amazon.com) World Wide Web.
3. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*.
4. Carbo, J., Molina, J.M., & Davila, J. (2003). Trust management through fuzzy reputation. *International Journal of Cooperative Information Systems*, *12*(1), 135–155.
5. Castelfranchi, C., & Falcone, R. (2001). Social trust: A cognitive approach. In C. Castelfranchi, & Y. Tan, (Eds.), *Trust and deception in virtual societies*. (pp. 55–90). Kluwer Academic Publishers.
6. Cohen, P. R. (1995). *Empirical methods for artificial intelligence*. The MIT Press.
7. Conte, R., & Paolucci, M. (2002). *Reputation in artificial societies*. Kluwer Academic Publishers.
8. Dasgupta, P. (2000). Trust as a Commodity. In D. Gambetta (Ed.), *Trust: Making and breaking cooperative relations*. Electronic edn. Department of Sociology, University of Oxford 49–72.
9. DeGroot, M. H., & Schervish, M. J. (2002). *Probability and statistics*. Addison-Wesley.
10. eBay Site. (http://www.ebay.com) World Wide Web.
11. Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid: enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, *15*(3), 200–222.
12. Gambetta, D. (2000). Trust: Making and breaking cooperative relations. Electronic edn. Department of Sociology, University of Oxford.
13. Grandison, T., & Sloman, M. (2000). A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, *3*(4).
14. Huynh, T. D., Jennings, N. R., & Shadbolt, N. R. (2004a). Developing an integrated trust and reputation model for open multi-agent systems. In R. Falcone, S. Barber, J. Sabater, & M. Singh, (Eds.), *Proceedings of the 7th international workshop on trust in agent societies* (pp. 65–74).

[20] Suppose that r_1 and r_2 are experiences of some person and the time unit used is a *year*. If the current time is year 2, r_2 should have much more influence on the trust decision of that person than r_1 since it is much more recent. If the current time is year 20, both r_1 and r_2 are too old, and they should have similarly low levels of influence on his current trust decision.

15. Huynh, T. D., Jennings, N. R., & Shadbolt, N. R. (2004b). FIRE: An integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 16th European conference on artificial intelligence (ECAI)* (pp. 18–22).
16. Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, *44*(4), 35–41.
17. Jurca, R., & Faltings, B. (2003). Towards incentive-compatible reputation management. In R. Falcone, S. Barber, L. Korba, & M. Singh, (Eds.), *Trust, reputation and security: theories and practice*. Vol. 2631 of Lecture Notes in AI (pp. 138–147). Springer-Verlag, Berlin, Heidelberg.
18. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.
19. Marsh, S. P. (1994). *Formalising trust as a computational concept*, PhD thesis, University of Stirling.
20. Mass, Y., & Shehory, O. (2001). Distributed trust in open multi-agent systems. In R. Falcone, M. Singh, & Y. Tan, (Eds.), *Trust in cyber-societies* (pp. 159–173). Springer-Verlag, Berlin, Heidelberg.
21. Maximilien, E. M., & Singh, M. P. (2002). Reputation and endorsement for web services. *ACM SIGEcom Exchanges*, *3*(1), 24–31.
22. Mui, L., Mohtashemi, M., & Halberstadt, A. (2002). A computational model of trust and reputation. In *Proceedings of the 35th Hawaii international conference on system science*. (pp. 280–287).
23. Ooi, B. C., Liau, C. Y., & Tan, K. L. (2003). Managing trust in peer-to-peer systems using reputation-based techniques. In *Proceedings of the 4th international conference in advances in web-age information management (WAIM 2003)*. Vol. 2762 of Lecture Notes in Computer Science. 2–12 Chengdu, China, August 17–19.
24. Ramchurn, S. D., Huynh, T. D., & Jennings, N. R. (2004). Trust in multi-agent systems. *The Knowledge Engineering Review*, *19*(1), 1–25.
25. Resnick, P., & Zeckhauser, R. (2002). Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. In M. R. Baye, (Ed.), *The economics of the internet and e-commerce*. Vol. 11 of Advances in Applied Microeconomics. Elsevier Science.
26. Sabater, J. (2003). *Trust and Reputation for Agent Societies*. PhD thesis, Universitat Autònoma de Barcelona.
27. Sabater, J., & Sierra, C. (2001). REGRET: A reputation model for gregarious societies. In *Fourth workshop on deception fraud and trust in agent societies* (pp. 61–70). Montreal, Canada.
28. Saha, D., & Mukherjee, A. (2003). Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, *36*(3), 25–31.
29. Salton, G., & McGill, M. (1983). *An introduction to modern information retrieval*. McGraw-Hill, New York.
30. Schillo, M., Rovatsos, M., & Funk, P. (2000). Using trust for detecting deceitful agents in artificial societies. *Special Issue of the Applied Artificial Intelligence Journal on "Deception, Fraud and Trust in Agent Societies"*, *14*(8), 825–848.
31. Sen, S., & Sajja, N. (2002). Robustness of reputation-based trust: Boolean case. In *Proceedings of the first intenational joint conference on autonomous agents and multiagent systems*. Vol. 1. (pp. 288–293).
32. Skogsrud, H., Benatallah, B., & Casati, F. (2003). Model-driven trust negotiation for web services. *IEEE Internet Computing*, *7*(6), 45–52.
33. Steinmetz, R., & Wehrle, K., (Eds.) (2005). Peer-to-peer systems and applications. Vol. 3485 of *Lecture Notes on Computer Science*, Springer Publishing.
34. Teacy, W. T. L., Patel, J., Jennings, N. R., & Luck, M. (2005). Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *proceedings of fourth international joint conference on autonomous agents and multiagent systems* (pp. 997–1004).
35. Uszok, A., Bradshaw, J. M., & Jeffers, R. (2004). KAoS: A policy and domain services framework for grid computing and semantic web services. In C. Jensen, S. Poslad, & T. Dimitrakos, (Eds.), Trust management: Second international conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings. Vol. 2995 of Lecture Notes in Computer Science., (pp. 16–26). Springer-Verlag, Berlin, Heidelberg.
36. Wasserman, S., & Faust, K. (1994). *Social network analysis : Methods and applications*. Volume 8 of Structural Analysis in the Social Sciences. Cambridge University Press.
37. Yolum, P., & Singh, M. P. (2004). Service graphs for building trust. In *International conference on cooperative information systems (CoopIS)*. Vol. 3290 of Lecture Notes in Computer Science (pp. 509–525).
38. Yu, B., & Singh, M. P. (2002). An evidential model of distributed reputation management. In *Proceedings of first international joint conference on autonomous agents and multi-agent systems*. Vol. 1. (pp. 294–301). ACM Press.
39. Yu, B., & Singh, M. P. (2003). Searching social networks. In *Proceedings of the second international joint conference on autonomous agents and multiAgent systems (AAMAS)* (pp. 65–72). ACM Press.
40. Zacharia, G., & Maes, P. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence*, *14*(9), 881–908.
41. Zimmermann, P. R. (1995). *The official PGP user's guide*. Cambridge, MA: MIT Press.