Knowledge-Based Systems 52 (2013) 147–164

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Decision making matters: A better way to evaluate trust models

David Jelenc [a,*], Ramón Hermoso [b,c], Jordi Sabater-Mir [d], Denis Trček [a]

[a] University of Ljubljana, Faculty of Computer and Information Science, Laboratory for e-Media, Tržaška 25, 1000 Ljubljana, Slovenia
[b] University of Essex, School of Computer Science and Electronic Engineering, Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
[c] University Rey Juan Carlos, CETINIA, Tulipán s/n, 28937 Mostoles, Madrid, Spain
[d] Artificial Intelligence Research Institute - IIIA, Spanish National Research Council - CSIC, Campus UAB, 08193 Bellaterra, Catalonia, Spain

ARTICLE INFO

Article history:
Received 14 November 2012
Received in revised form 15 July 2013
Accepted 23 July 2013
Available online 31 July 2013

Keywords:
Trust
Reputation
Testbed
Evaluation
Multi-agent system

ABSTRACT

Trust models are mechanisms that predict behavior of potential interaction partners. They have been proposed in several domains and many advances in trust formation have been made recently. The question of comparing trust models, however, is still without a clear answer. Traditionally, authors set up ad hoc experiments and present evaluation results that are difficult to compare – sometimes even interpret – in the context of other trust models. As a solution, the community came up with common evaluation platforms, called trust testbeds. In this paper we expose shortcomings of evaluation models that existing testbeds use; they evaluate trust models by combining them with some ad hoc decision making mechanism and then evaluate the quality of trust-based decisions. They assume that if all trust models use the same decision making mechanism, the mechanism itself becomes irrelevant for the evaluation. We hypothesized that the choice of decision making mechanism is in fact relevant. To test our claim we built a testbed, called Alpha testbed, that can evaluate trust models either with or without decision making mechanism. With it we evaluated five well-known trust models using two different decision making mechanisms. The results confirm our hypothesis; the choice of decision making mechanisms influences the performance of trust models. Based on our findings, we recommend to evaluate trust models independently of the decision making mechanism – and we also provide a method (and a tool) to do so.

1. Introduction

In the last decade, the internet has grown rapidly. This growth spurred the development of collaborative applications, in which different parties interact to fulfill their goals. Such cooperation is often hard to establish, because participants have no assurance that their interaction counterparts will honor agreements. To mitigate such risks, trust and reputation systems have shown promising results in many domains.

Trust models are mechanisms that enable parties to build trust; a concept usually understood as a degree to which one party has confidence in another within the context of a given purpose or decision [1]. Trust models compute trust on behalf of their users by using various information, such as opinions from other participants, their users' own experiences, social-network information and others. Trust models predict the quality of interactions by estimating the future behavior of potential interaction partners. Thus a good trust model should help its user to avoid participants that do not honor agreements and advise her to select interaction partners that honor them.

Computational trust modeling research began in the middle of nineties. Since then, the research community has proposed many solutions and a compilation of them can be found in surveys such as [2–5]. These papers classify trust models into various categories ranging from the type of information that they use to the techniques that they deploy.

In this abundance of proposals a natural question emerged: how does one evaluate and compare performances of trust models? Many researchers addressed this question by creating ad hoc testing environments. In them, they compare their trust models against the existing ones [6,7]. Such evaluation has almost become a *de facto* standard when introducing new trust models. However, it was soon realized that ad hoc evaluations are biased, since they always cover a limited number of trust model's aspects. In ad hoc evaluations, authors' proposals often come out as the best ones. This is expected, since ad hoc evaluations are designed to emphasize the features of new proposals. But for an evaluation to be complete, it also has to include scenarios that violate assumptions behind the model. A good evaluation has to show not only the virtues of the model but also its faults.

* Corresponding author. Tel.: +386 (0)1 47 68 259.
E-mail addresses: david.jelenc@fri.uni-lj.si (D. Jelenc), rhermoso@essex.ac.uk (R. Hermoso), jsabater@iiia.csic.es (J. Sabater-Mir), denis.trcek@fri.uni-lj.si (D. Trček).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.07.016

An improvement over ad hoc evaluations comes in the form of trust testbeds. A trust testbed is an independent and easy-to-use platform for benchmarking trust models. The underlying idea is that a neutral party provides the testbed that other researchers can use to evaluate their own proposals against a set of well-defined scenarios and metrics. The most known example of a trust testbed is ART [8], but there are a few others that we describe in the related work section.

Existing testbeds evaluate trust models with a similar evaluation model. They provide an environment of agents that are of different quality as interaction partners. Within this environment they configure one agent (in some cases several agents) to use the selected trust model. The chosen agent then interacts with other agents in the environment and in so doing it uses the tested trust model to select interaction partners. The testbed evaluates the trust model by measuring the *utility* that was obtained by the agent that used the trust model. The actual meaning of this utility differs between testbeds; it can be the amount of earnings in the art appraisal contest [8] or the accuracy of routing packets in a wireless sensor network [9]. The reasoning behind such evaluation is that a good trust model should advise its agent better than a bad one and, therefore, agents with good trust models should obtain higher utilities than those that use bad ones.

The problem of such evaluation lies in the measured object. Existing testbeds do not measure the output of trust models directly. Instead, they measure it indirectly by evaluating the quality of decisions that agents make using those trust models (usually they measure the quality of partner selections). The decisions, however, are seldom based only on the computed trust; agents often incorporate other factors in their decision making processes. But even in cases when trust is the only relevant factor, deciding with whom to interact next may affect trust models, since it determines the exploration vs. exploitation policy. We refer to the component of an agent responsible for making such decisions as the decision making mechanism. We claim that the indirect evaluation entails an important concern; a good working trust model can be hindered by a bad decision making mechanism or the other way around. This would not be a problem if the trust models were presented together with the decision making mechanism as a whole and what was being evaluated was the pair (trust model, decision making mechanism). However, that is not the case. The trust model is usually presented as an isolated oracle that is queried by the decision making mechanism and no hints are provided about how the latter uses the answers to those queries. Therefore, because evaluating trust models with utility requires trust models to have a decision making mechanism and because most trust model proposals lack it, existing testbeds have to combine trust models with their own ad hoc versions of decision making mechanisms.

As we will show in this paper, such ad hoc pairing can be problematic, because *some decision making mechanisms may be more suitable for certain trust models and may not work well with others*. For instance, it can happen that using one trust model in combination with a decision making mechanism *A* could produce different results than using the same trust model in conjunction with a decision making mechanism *B*. Something that looks as a minor implementation detail may completely swing the evaluation results.

The purpose of this paper is to demonstrate that in order to evaluate trust models it is not enough to attach them to arbitrary decision making mechanisms and measure the utility that their owners obtain. We show that the choice of the decision making mechanism will favor some models while being detrimental to others and, therefore, the initial goal of comparing models in a neutral environment becomes compromised. Given that, we claim that it is improper to evaluate trust models by attaching them to arbitrary decision making mechanisms as it happens in current trust testbeds.

To support the aforementioned claims, and as a second contribution, we have developed a testbed platform, called the Alpha Testbed (ATB). Its purpose is twofold: first, to create a measurement apparatus to test our hypothesis, and second, to overcome the problems of existing testbeds that pair trust models with decision making mechanisms for the sole purpose of evaluation. In contrast, ATB can evaluate trust models with or without decision making mechanisms. The second contribution is an evolved and extended version of the work that was first presented in [10].

The paper is structured as follows. Section 2 describes the elements of trust models and defines several terms used throughout the article. Section 3 presents the Alpha Testbed (ATB). Empirical results are provided and discussed in Section 4, and ATB's design and extendability are depicted in Section 5. We compare our approach with others in Section 6 and conclude with Section 7.

2. Trust models

Trust is a complex concept that can be analyzed from several perspectives. So before we discuss how to evaluate trust models, we have to describe our understanding of trust models and define several concepts that we use in the rest of the article.

Using common conceptions from existing surveys [2–5] and following a couple of standardization attempts [11,12], we treat trust models as computational mechanisms that assign *trust values* to third-party agents. Trust values are used to make different decisions such as selecting partners to interact with in a commercial exchange, changing a negotiation strategy or deciding who to ask for opinions.

There is a great diversity of trust models and they can be classified considering different features. However, one of the aspects that takes more relevance, specially when one talks about testbeds, is the *type of information* from which they compute trust. Some use experiences from previous interactions, some opinions from other agents in the system, some analyze the underlying social network of agents or study the information about the virtual organization to which agents belong, and even more complex examples exist [5]. Many combine several types of information to achieve better estimations. In this section, we describe the *information context* of a trust model and provide some semi-formal definitions that we use throughout the paper.

2.1. Information context of a trust model

Information context denotes the sources of information and the flow of information from which a trust model computes trust. To graphically depict an information context of a general-purpose trust model, we build upon a schema that we partly borrow from [11,13]. The schema – shown on Fig. 1 – is centered around the agent that uses the trust model, called **agent** α. It shows three information sources from which α's trust model computes trust. The agent can obtain information by **interacting with agents**, by **asking for opinions**, or by using information from the **environment**. Because the first two information sources are the most common in current trust models, we highlighted them and encapsulated other possible sources for trust computation in a special component called environment; examples of such include the analysis of social networks, information about the virtual organizations, etc.

Furthermore, agent α consists of the *interpretation*, the *trust model* and the *decision making mechanism* sub-components. The interpretation converts obtained information to a representation that is compatible with the trust model (in the schema this corresponds to converting interaction outcomes to experiences, obtained opinions to opinions, and environmental information to

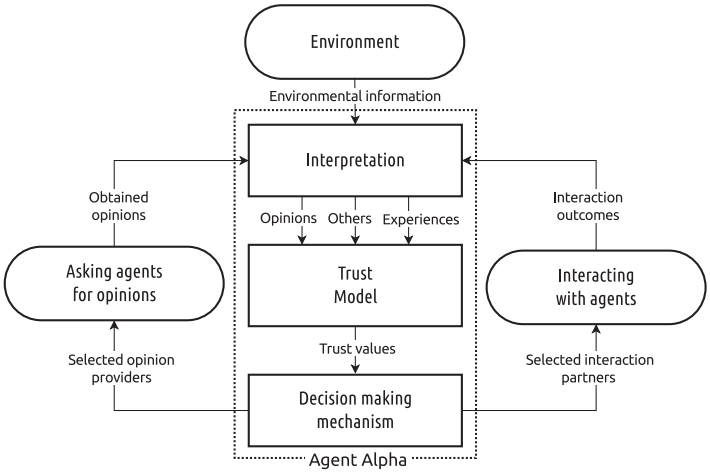

Fig. 1. Information context of a trust model. Agent Alpha uses a trust model that obtains information by (i) interacting with agents, by (ii) asking agents for opinions, and by using other information from the (iii) environment. Agent then conveys the computed trust values to its decision making mechanism where they are used in various decision making processes, such as deciding with whom to interact or who to ask for opinions.

others). The trust model then uses this information to compute trust values. These are then conveyed to the decision making mechanism to (i) select interaction partners and to (ii) select opinion providers (and in some cases offer opinions to other agents). The decision making mechanism is usually very complex and while trust values can be an important part of its input, the decision making mechanism also considers other factors. They are, however, domain specific and often independent of the trust model, which is why the majority of trust models do not provide any guidance on how to use the computed values in the decision making process [5].

As pointed out in the existing surveys, most trust models only use experiences and opinions to compute trust. Because of this, we shall restrict ourselves and in this paper investigate only such models. We plan to extend our framework to accommodate trust models that use other environmental information in the future.

2.2. Definitions

This section presents definitions of the most relevant concepts that we use in the rest of the article.

Time and services. We represent time with a totally ordered discrete set $\mathcal{T}$ of time values (ticks) $t_i \in \mathcal{T},\ i \in \mathbb{N},\ t_i < t_{i+1}$. Additionally, we denote the available set of service types with $\mathcal{S}$.
Agents. The agent that uses the trust model is called agent α. We denote the rest of the agents in the system with set $\mathcal{A}$. This set can change, because agents may leave and join the system. To denote the set of agents at a given time $t \in \mathcal{T}$, we use $\mathcal{A}(t)$.
Experiences. An experience $\varepsilon \in \mathcal{E}$ is a record of an interaction between α and some other agent. We denote an experience as a tuple $\varepsilon = \langle a, s, t, \lambda \rangle$, where $\lambda \in \Lambda$ represents the assessment with which α evaluates the performance of the service provider $a \in \mathcal{A}$ for performing service $s \in \mathcal{S}$ at time $t \in \mathcal{T}$. We denote the set of all experiences with $\mathcal{E}$.
Trust values. A trust value $\tau \in \Theta$ represents α's trust towards a particular agent. We denote a trust value as a tuple $\tau = \langle a, s, \omega \rangle$, where $\omega \in \Omega$ represents α's trust degree towards agent $a \in \mathcal{A}$ for service $s \in \mathcal{S}$. A trust degree is a value with which α expresses the level of trust towards other agents. While we do not assume any particular type of trust degrees, we require that the set Ω be totally ordered. This means that any two trust degrees have to be mutually comparable. Analogously, we denote the set of all possible trust values with Θ.

Opinions. An opinion $o \in \mathcal{O}$ is a statement about trust that someone gave to α about a third-party. We denote an opinion as a tuple $o = \langle a_o, a_p, s, t, \omega, u \rangle$, where $\omega \in \Omega$ represents the trust degree that agent $a_o \in \mathcal{A}$ told α to have towards agent $a_p \in \mathcal{A}$ for service $s \in \mathcal{S}$. Symbol $t \in \mathcal{T}$ denotes the time at which the opinion was given, while $u \in [0, 1]$ denotes the level of uncertainty of agent a_o about this opinion. Similarly, $\mathcal{O}$ denotes the set of all possible opinions.
Trust model. A trust model is computational device that computes α's trust towards other agents. We define a trust model as a function $\mathit{TrustModel} : \mathcal{T} \times \mathcal{P}(\mathcal{E}) \times \mathcal{P}(\mathcal{O}) \to \mathcal{P}(\Theta)$ that maps a point in time $t \in \mathcal{T}$, a set of experiences $\varepsilon_{\text{set}} \in \mathcal{P}(\mathcal{E})$ and a set of opinions $o_{set} \in \mathcal{P}(\mathcal{O})$ to a set of trust values $\tau_{\text{set}} \in \mathcal{P}(\Theta)$, thus $\tau_{\text{set}} = \mathit{TrustModel}(t, \varepsilon_{\text{set}}, o_{\text{set}})$.

3. The Alpha testbed

This section proposes a simulation-based testbed, which we named Alpha Testbed (ATB, http://atb.fri.uni-lj.si). In ATB, one agent, named agent Alpha (α), uses the tested trust model while other agents are simulated. During evaluation, ATB generates data that α conveys to its trust model. The data consists of experiences from past interactions and opinions from other simulated agents in the system. ATB can evaluate both the output of α's trust model and the decisions that agent α makes. In the following subsections, we describe how ATB operates during the evaluation, how it generates experiences and opinions and how it evaluates trust models.

3.1. Evaluation protocol

ATB can evaluate two types of trust models; those that have a decision making mechanism and those that do not. The type of trust model thus determines the protocol and the metrics that will be used in the evaluation. In this context, the term evaluation protocol describes the sequence of steps that are carried out at each tick of the simulation run.

3.1.1. Trust models without decision making mechanism

Trust models without decision making mechanism are trust models that provide rules, formulas or algorithms describing how to compute trust, but at the same time they provide no guidance on how to use that information in decision making processes. Such trust models are the most common in the literature. To evaluate them, we have to give them experiences, and for creating those,

someone has to select interaction partners. Existing testbeds solve this by pairing trust models with ad hoc decision making mechanisms. A common example is to select the agent to whom the trust model assigned the highest trust degree. As we show later (see Section 4), such pairing can be problematic, because some decision making mechanisms may be more suitable for certain trust models and may not work well with others.

When ATB evaluates trust models without decision making mechanisms, the interaction partners are not determined by an ad hoc decision making mechanism. Instead they are determined by the parameters of the evaluation run. Such parametrization has two notable benefits: first it assures that trust models are evaluated isolated from the decision making mechanism, and second, it assures that all trust models are evaluated with the same input. The latter is essential if evaluation is to be fair. When a testbed pairs a trust model with an ad hoc decision making mechanism, it violates this principle. For instance, two trust models may compute different trust values and although they apply the same decision making to those values, the selection of interaction partner can be different. If this happens, the two trust models will obtain different experiences and from then on, they will compute trust from different data. The results of such evaluation thus become ambiguous. But if the interaction partners (and thus experiences) are determined externally, every trust model receives the same input, which means that we can easily compare performances.

In this protocol, ATB evaluates trust models by evaluating the accuracy of computed trust values (we discuss accuracy in depth in Section 3.5.1). The evaluation protocol thus consists of the following steps:

1. ATB notifies the trust model of a new time tick.
2. ATB generates opinion tuples and conveys them to the trust model.
3. ATB selects an agent as the current interaction partner, generates an experience tuple, and conveys it to the trust model.
4. The trust model computes trust values and submits them to ATB.
5. ATB evaluates computed trust values.

This sequence of steps is repeated for every tick. Notice that the testbed conveys the same opinion and experience tuples to every trust model.

3.1.2. Trust models with decision making mechanism

Contrary to previous case, trust models with decision making mechanism are trust models that provide both (i) rules, formulas or algorithms describing how to compute trust, and also (ii) hints on how to use that information in the decision making processes. The evaluation protocol and the used metrics differ, depending on what the decision making mechanism does. When it only selects interaction partners, we have mode *A*, and when the decision making mechanism also suggests who to ask for opinions, we have mode *B*. The evaluation protocol for mode *A* is the following.

1. ATB notifies the trust model of a new time tick.
2. ATB generates opinion tuples and conveys them to the trust model.
3. The trust model selects an agent as the interaction partner and announces it to ATB.
4. ATB generates an experience tuple for the selected agent, and conveys the tuple to the trust model.
5. The trust model computes trust values and submits them to ATB.
6. ATB evaluates the computed trust values.
7. ATB evaluates the utility that was obtained in the interaction.

This sequence is repeated at every tick. Because here the selection of interaction partners is determined by the decision making mechanism (and not by the testbed as in the previous protocol), ATB also evaluates the quality of partner selections with utility metric (for a comprehensive explanation of utility see Section 3.5.2). The evaluation protocol for mode *B* is similar, however, it includes two additional steps (shown in italics):

1. ATB notifies the trust model of a new time tick.
2. *The trust model selects which agents to ask for opinions and announces them to ATB.*
3. ATB generates opinion tuples and conveys them to the trust model.
4. The trust model selects an agent as the interaction partner and announces it to ATB.
5. ATB generates an experience tuple for the selected agent, and conveys the tuple to the trust model.
6. The trust model computes trust values and submits them to ATB.
7. ATB evaluates the computed trust values.
8. ATB evaluates the utility that was obtained in the interaction.
9. *ATB evaluates the endured cost when fetching requested opinions.*

These steps are repeated at every tick. Because in mode *B* trust models also decide who to ask for opinions, ATB additionally evaluates the costs associated with obtaining the required opinions (for detailed description of opinion cost metric see Section 3.5.3). The distinction between modes *A* and *B* allows us to evaluate trust models whose decision making mechanism only selects interaction partners separately from those trust models whose decision making mechanism also provides guidance on who to ask for opinions. We made this distinction, because few trust models define how to select interaction partners and even fewer define who to ask for opinions. Modes allow ATB to cover both cases.

3.2. Generating experiences

Having defined the evaluation protocols, we continue with describing the procedure that ATB uses to create experiences. Agent α gets experiences when it interacts with other agents. When interacting, agents provide services and the quality of these provided services reflects the quality of the agents. So to describe the creation of experiences, we must explain what we mean by the quality of agents and explain precisely how interactions reflect this quality.

3.2.1. Capabilities of agents for providing services

We model the quality of agents as service providers using the notion of *capability*.

Definition 1 (*Capability*). Capability of agent $a \in \mathcal{A}$ for service $s \in \mathcal{S}$ at time $t \in \mathcal{T}$ represents that agent's ability and willingness to provide a quality service to agent α. We express capability as a real number from [0, 1], where 0 and 1 represent the lowest and highest values, respectively. The capability is thus a mapping $\mathit{Capability} : \mathcal{A} \times \mathcal{S} \times \mathcal{T} \to [0, 1]$.

Capabilities represent the actual quality of agents in interactions. Interacting with an agent whose capability is high yields more utility than interacting with an agent whose capability is low. We use capabilities as a basis (i) to model interactions (to construct α's experience) and (ii) to simulate opinions that agents give to α. Moreover, the actual capabilities are only known by the testbed and are never revealed to α. Agent α can only infer them from interactions or when agents provide opinions about other agents. Therefore we use capabilities as an integral part of the evaluation.

3.2.2. Modeling interactions

We model interactions between α and the other agents by generating *experience* tuples. An experience $\varepsilon = \langle a, s, t, \lambda \rangle$ is a record of a past interaction, where a denotes the providing agent, and symbols s, t and λ denote the type of service, time of interaction and the assessment of the interaction, respectively. Trust models represent assessments with different formats; some use binary, some discrete, some continuous scales and even other approaches exist. In trying to be general, ATB internally represents assessments as real numbers from [0, 1], where 0 and 1 represent the worst and best assessment, respectively. We call these internal assessments interaction outcomes. If a trust model represents assessments in a way that is different from interaction outcomes, it has to convert interaction outcomes to its own domain of assessments [14].

To create an experience tuple, ATB computes the interaction outcome from the capability of the agent that provides the service. But to simulate a realistic setting, a particular interaction outcome could be different than the actual capability of the providing agent. The difference, however, should not be substantial and in the long run, interaction outcomes should resemble the actual capabilities of agents; the difference should amount only to the noise. Therefore ATB generates interaction outcomes with a pseudo-random generator. The generator uses a probability density function (PDF) that is parametrized with two parameters: (i) the capability of the agent that provides the service and (ii) the parameter that controls the noise. For this purpose we use the PDF of a normal distribution that is truncated on [0, 1].

$$p(x;\mu,\sigma) = \begin{cases} \dfrac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\int_0^1 e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt} & 0 \leqslant x \leqslant 1 \\ 0 & \text{else} \end{cases}$$

The shape of this PDF is determined by the mean μ and the standard deviation σ. To model an interaction outcome between α and agent a for service s at time t, we invoke a pseudo-random generator and set the parameter μ to the capability of the providing agent, thus $\mu = \mathit{Capability}(a, s, t)$, while we can set σ to an arbitrary value that is fixed for all agents. For instance, Fig. 2 shows PDFs for three agents with capabilities 0.1, 0.5 and 0.9, respectively. The σ is set to 0.15 in all three cases.

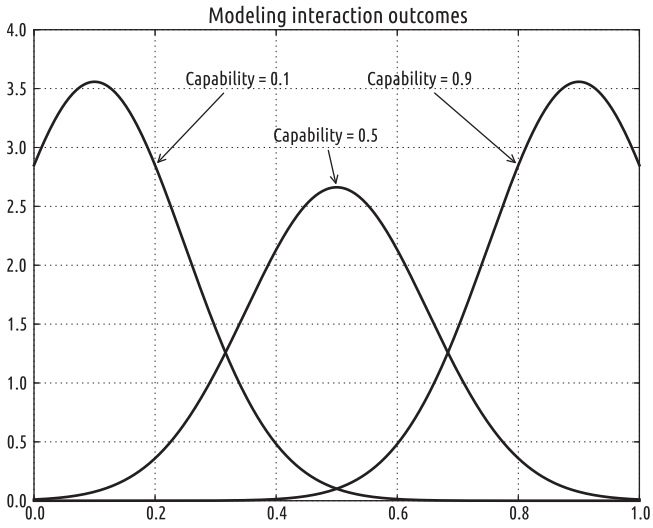

Fig. 2. ATB uses truncated normal distribution to generate interaction outcomes. The mean, μ, is set to the capability of an agent, while the standard deviation, σ, is set to an arbitrary value that controls the noise. Here we show agents with capabilities of 0.1, 0.5 and 0.9, while σ is set to 0.15 in all three cases.

3.3. Generating opinions

Other agents in the system can provide opinions about third-parties to agent α. In fact, those opinions would be a combination of two factors; the trust models that other agents use and their honesty towards agent α. Because we wish to investigate the performance of α's trust model and minimize the impact of other factors, we simulate the trust models of other agents. To model the honesty of other agents we use deception models.

3.3.1. Modeling trust between other agents

We model trust between other agents in the system with opinions. An opinion is defined as a tuple $o = \langle a_i, a_j, s, t, \omega, u \rangle$, where ω denotes the trust degree of a_i towards a_j for service s at time t. Variable u denotes how uncertain is agent a_i about the given opinion; values 0 and 1 denote complete certainty and uncertainty, respectively. Because trust models represent trust degrees, like assessments, with different formats, ATB internally represents trust degrees with real values from [0, 1], where 0 and 1 represent the lowest and the highest degrees, respectively. If a trust model uses a different representation of trust degrees, it has to convert internal trust degrees to its own domain (recall Fig. 1).

ATB creates opinions by generating internal trust degrees with a pseudo-random generator. The generator uses a PDF of a truncated normal distribution that is parameterized with the capability of the agent who is the object of the opinion. So to compute the internal trust degree of agent a_i towards a_j for service s and at time t, we invoke the pseudo-random generator and set the mean to $\mu = \mathit{Capability}(a_j, s, t)$, while we use the standard deviation, σ, to control the uncertainty of the opinion; we use small values when agent a_i knows agent a_j very well, and larger values when contrary is the case.

This way of generating opinions assures three things. First, it assures that agents provide opinions with varying accuracy; some agents may know specific agents better than others. Second, it assures that opinions vary even when they are given by agents that know the chosen agent in the same amount. And third, it assures that the evaluated trust model does not interfere with the creation of opinions; the same opinions are generated in every evaluation regardless of the evaluated trust model.

3.3.2. Modeling deception

It is reasonable to expect that in an open environment at least some agents will provide false opinions. To model such agents, ATB uses deception models [15]. Deception models determine which opinions will be reported to α and in what form. They can alter opinions in three ways; they can change internal trust degrees, they can delete opinions or they can leave them unchanged.

A deception model is a mapping d: [0, 1] → [0, 1] that transforms the internal trust degree into a trust degree that will be given to α. In their proposal, Yu and Sing [15] used four deception models; truthful, complementary, exaggerated positive and exaggerated negative. The truthful model does not alter trust degrees; it accounts for agents that are honest at providing opinions. In the complementary model, agents report opinions with the complementary trust degrees, while in the exaggeration models agents either overestimate (positive exaggeration) or underestimate (negative exaggeration). If a deception model is not assigned (that is, $\mathcal{D}(a_i, a_j, s, t) = \emptyset$) this means that at time t and for service s agent a_i will not provide an opinion about agent a_j. This covers the examples where agents are either unwilling to provide opinions or are unaware of that agent's existence. A graphic representation of models is given in Fig. 3.

We are adding a *random* model to the set of existing deception models. In random model, agents provide opinions with trust degrees that are selected at random from a uniform distribution.

Set $\mathcal{D}$ represents an example set of deception models that can be used in an evaluation. In our case we have $\mathcal{D} = \{d_t, d_c, d_{pe}, d_{ne}, d_r\}$, where every model is defined as follows (parameter $0 \leqslant \kappa \leqslant 1$ denotes the exaggeration coefficient; Fig. 3 uses $\kappa = 0.25$):

- truthful model $d_t(x) = x$;
- complementary model $d_c(x) = 1 - x$;
- positive exaggeration model $d_{pe}(x) = x \cdot (1 - \kappa) + \kappa$;
- negative exaggeration model $d_{ne}(x) = x \cdot (1 - \kappa)$;
- random model $d_r = \mathcal{U}(0, 1)$;

Deception models in $\mathcal{D}$ constitute the most common ways of providing opinions in an open multi-agent environment: there are some agents that provide honest opinions, some that lie extensively [15] and some that are biased in either positive or negative direction [16]. Random model simulates the most unpredictable agents whose opinions are the least reliable, while the lack of deception model assignments accounts for the scarcity of opinions.

ATB assigns a deception model to every agent in the system. The assignment depends on which agent the opinion is about, the service, and the time, thus $Deception : \mathcal{A} \times \mathcal{A} \times \mathcal{S} \times \mathcal{T} \to \mathcal{D}$. This way a particular agent can vary its honestly in time, between services and even between agents.

Finally, we also use deception models to simulate discriminating behavior of agents as service providers. When an agent behaves differently towards α than it behaves towards other agents, then the opinions of other agents about this particular provider become lies from α's standpoint. For α, there is no difference between an agent providing a false opinion about a third-party, and an agent providing a truthful opinion about the same-third party, if the third party discriminates. In either case, the opinion is of no use to α.

3.4. Evaluation scenario

While the preceding sections describe capabilities, deception models, and procedures that use them to generate experiences and opinions, this section describes a component of the testbed that ties all these concepts together – evaluation scenarios. An evaluation scenario represents the entire surroundings of agent α; it describes other agents and their behavior. More specifically, a scenario determines the following:

Agent population. A scenario determines the set of agents in the system, $\mathcal{A}$, and it also determines how the set changes as the evaluation progresses; the set may change because some agents may leave and new may enter the system.
Assignment of capabilities. A scenario assigns capabilities and manages their changes. This includes defining and managing the set of available services, $\mathcal{S}$.
Assignment of deception models. A scenario assigns deception models and manages their changes.
Selection of interaction partners. When evaluating trust models without decision making mechanisms, the scenario determines interaction partners of agent α. When the contrary is the case, this task is delegated to α's decision making mechanism.
Selection of opinion providers. When evaluating trust models without decision making mechanisms or when the decision making mechanisms do not suggest who to ask for opinions (mode *A*), the scenario determines the opinions that are given to agent α. When the contrary is the case (mode *B*), this task is delegated to α's decision making mechanism.

Scenarios provide flexibility for constructing evaluation environments; they can be used to reproduce various attacks on trust models. For instance, in a *white-washing attack* [1] an agent, called attacker, abuses the system by letting its trust degrade and then escapes the consequences by re-entering the system with a fresh identity. In ATB, one would represent this attack by carefully manipulating the set of agents and assignment of capabilities. For instance, in the beginning the attacker would have a high capability, but as the attack progresses, the attacker's capability would be lowered. Once α would no longer select the attacker as the interaction partner, the scenario would remove the attacker and create a new agent with a different identity but with the same behavior. A more sophisticated scenario might create an additional cast of colluding attackers that would falsely promote the attacker to maximize the damage.

Another way of looking at scenarios is to consider the entire evaluation as a game. The game is played between the scenario and agent α. The scenario controls the system and knows everything, while agent α knows nothing at the beginning, but senses the environment (via experiences and opinions) and uses the tested trust model to infer trust.

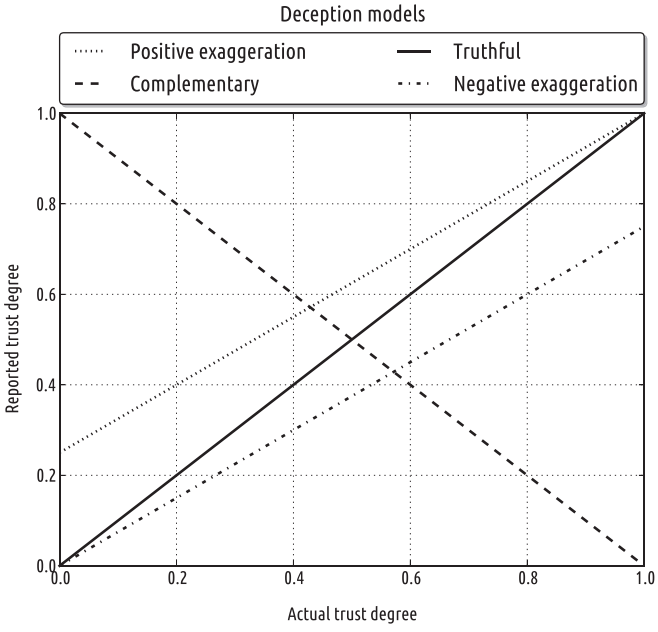

Fig. 3. Modeling deception with deception models. If an agent's actual trust degree towards some agent is 0.20, a truthful agent will report 0.20, while agents with complementary, positive exaggeration and negative exaggeration deception models will report 0.80, 0.40 and 0.15, respectively.

3.5. Metrics

ATB provides several metrics and their use depends on the evaluated trust model. The following subsections explain metrics and their use.

3.5.1. Accuracy

When evaluating trust models without decision making mechanism, ATB evaluates the correctness of the computed trust values. Because capabilities define the quality of agents, we can evaluate trust values by measuring the similarity between them and the actual capabilities of agents. One way of doing it would be to compute a standard forecast error metric, such as the mean absolute error, between the estimated trust degrees and the capabilities of agents. However, this would require that all trust models represent trust degrees as real values from [0, 1]. While some models represent trust degrees in this format, others use different

representations. In Section 2 we noted that the set of trust degrees should be totally ordered. This means that any trust model – regardless of its domain of trust degrees – should be able to rank agents by their trust degrees. Therefore, we propose to compute accuracy by evaluating the rankings of agents that emerge from trust estimations. The idea is to compare rankings that emerge from trust degrees with the ranking that emerges from capabilities. For this purpose, we can use any metric that computes distances between rankings.

Definition 2 (*Accuracy*). Let $Trust(a, s, t)$ denote the trust degree that agent α has towards agent $a \in \mathcal{A}$ for service $s \in \mathcal{S}$ at time $t \in \mathcal{T}$. We define $Concordant : \mathcal{S} \times \mathcal{T} \to \mathbb{N}_0$ as a function that returns the number of pairwise comparisons of agents by their trust degrees that are concordant with the pairwise comparisons of agents by their capabilities.

$$Concordant(s,t) = \sum_{\substack{a_i, a_j \in \mathcal{A}(t) \\ Capability(a_i,s,t) > Capability(a_j,s,t)}} [Trust(a_i,s,t) > Trust(a_j,s,t)]$$

Similarly, we define a function $Discordant : \mathcal{S} \times \mathcal{T} \to \mathbb{N}_0$ that returns the number of pairwise comparisons of agents by their trust degrees that are discordant with the pairwise comparisons of agents by their capabilities.

$$Discordant(s,t) = \sum_{\substack{a_i, a_j \in \mathcal{A}(t) \\ Capability(a_i,s,t) > Capability(a_j,s,t)}} [Trust(a_i,s,t) < Trust(a_j,s,t)]$$

Finally, we define $Accuracy : \mathcal{S} \times \mathcal{T} \to [0, 1]$ as a metric that returns the level of similarity between trust degrees and capabilities of agents. We evaluate accuracy at every tick $t \in \mathcal{T}$ and for every type of service $s \in \mathcal{S}$.

$$Accuracy(s,t) = \frac{1}{2} + \frac{Concordant(s,t) - Discordant(s,t)}{|\mathcal{A}(t)| \cdot (|\mathcal{A}(t)| - 1)}$$

The accuracy from above is a slight variation of Kendall's Tau-A metric. Our modification only scales the result from its original domain of $[-1, 1]$ to $[0, 1]$. Score 1 accounts for perfect estimations, where the ranking of agents by trust degrees completely agrees with the ranking of agents by their capabilities. This means that the trust model works flawlessly. Score 0 means that trust estimations are the exact opposite of capabilities. In such case, the trust model is harmful. It suggests its user to interact with agents that have low capabilities and dissuades her from interacting with those whose capabilities are high. When the result is 0.5, the trust model bears no information; it is not beneficial, but it is also not harmful. Result 0.5 means that trust estimations and capabilities are uncorrelated. This result is obtained if, for instance, the trust model assigns the same trust degree to all agents.

Example. Say we have a set of agents, $\mathcal{A} = \{a_1, a_2, a_3\}$, with the following capabilities: $Capability(a_1) = 0.1$, $Capability(a_2) = 0.5$ and $Capability(a_3) = 0.9$. (For simplicity, assume constant capabilities and only one type of service, thus we omit the time and service parameters.) Additionally, say we test a trust model that uses qualitative trust degrees $\Omega = \{\text{bad} < \text{average} < \text{good}\}$. Now, imagine that the trust model computes the following: $Trust(a_1) = \text{average}$, $Trust(a_2) = \text{bad}$ and $Trust(a_3) = \text{good}$. In this case, the trust model computes trust degrees that impose a different ranking of agents than that imposed by capabilities:

- Agent ranking by trust degrees: $a_2 < a_1 < a_3$;
- Agent ranking by capabilities: $a_1 < a_2 < a_3$.

We compute the accuracy of the trust model in three steps. First, we compute the number of concordant trust degrees.

$$Concordant = 1_{[Trust(a_1) < Trust(a_3)]} + 1_{[Trust(a_2) < Trust(a_3)]} = 2$$

Second, we compute the number of discordant trust degrees.

$$Discordant = 1_{[Trust(a_1) > Trust(a_2)]} = 1$$

Finally, we compute the accuracy of the computed trust values.

$$Accuracy = \frac{1}{2} + \frac{2 - 1}{3 \cdot (3 - 1)} = \frac{2}{3}$$

Kendall's Tau-A from Definition 2 is only one possible metric to measure the accuracy of rankings. We could easily use Spearman's footrule or any other suitable replacement [17]. In fact, ATB has several accuracy metrics already implemented and new ones can be easily added as plug-ins. But in this paper, we shall only use Kendall's Tau-A.

3.5.2. Utility

When a trust model provides a decision making mechanism that selects interaction partners, we can evaluate the combined performance of the pair (trust model, decision making mechanism). The standard way of evaluating this combination is to measure the utility that agent α collects in interactions.

Definition 3 (*Utility*). Let $Partner(s, t)$ denote the agent that α selected as interaction partner for service $s \in \mathcal{S}$ at time $t \in \mathcal{T}$. We define $Utility : \mathcal{S} \times \mathcal{T} \to [0, 1]$ as a metric that for a given service $s \in \mathcal{S}$ returns the relative amount of utility that α obtained up to time $t \in \mathcal{T}$.

$$Utility(s,t) = \frac{\sum_{t_i \leqslant t} Capability(Partner(s,t_i), s, t_i)}{\sum_{t_i \leqslant t} \max_{a \in \mathcal{A}(t_i)} Capability(a, s, t_i)}$$

The utility measures α's relative performance in selecting quality interaction partners. In the numerator, we sum the capabilities of α's interaction partners from the beginning of the evaluation to the most recent tick. In the denominator, we sum the capabilities of the most capable agents for the same period. If α constantly selects the most capable agents as interaction partners the *Utility* is constantly 1. Notice that the utility has a 'memory'; the most recent value encapsulates all previous partner selections.

Example Let us reuse the set of agents and assignment of capabilities from previous example. (Again, assume static capabilities and only one service type, thus we can omit time and service from $Capability(a, s, t)$ notation and service from $Partner(s, t)$ notation.) If α selects the following agents as interaction partners $Partner(1) = a_1$, $Partner(2) = a_2$ and $Partner(3) = a_3$, we compute α's utility at time 3.

$$Utility = \frac{Capability(a_1) + Capability(a_2) + Capability(a_3)}{Capability(a_3) + Capability(a_3) + Capability(a_3)} = \frac{0.10 + 0.50 + 0.90}{0.90 + 0.90 + 0.90} = 0.55$$

Like accuracy, utility from Definition 3 is only one possible metric to measure the quality of decisions. We could have used any other implementation.

3.5.3. Opinion cost

When the decision making mechanism not only selects the interaction partners but also determines who to ask for opinions, ATB can also measure the cost of obtaining opinions.

Definition 4 (*Opinion cost*). Let $\mathcal{OR}(s, t)$ denote the set of opinion requests that α generated for service s at time t. Each opinion request is represented as a tuple $\langle a_o, a_p, s, t \rangle$, where a_o denotes the

agent that provides the opinion, a_p the agent that the opinion is about and symbols s and t denote the service and time, respectively. We define the *OpinionCost* : $\mathcal{S} \times \mathcal{T} \to \mathbb{N}_0$ as a metric that for a given service $s \in \mathcal{S}$ returns the number of opinions that α requested at time $t \in \mathcal{T}$.

$$\mathit{OpinionCost}(s,t) = |\mathcal{OR}(s,t)|$$

The *OpinionCost* above simply returns the number of opinions that agent α requests from other agents. Such number can be interpreted in various ways, for instance as the level of congestion in the communication network that the trust model causes with its opinion requests.

3.5.4. The use of metrics

Accuracy, utility and opinion cost allow us to benchmark different aspects of trust models. Accuracy measure the quality of trust values, utility measures the quality of trust-based decisions, and opinion cost measures the consumption of resources when obtaining opinions (like the communication overhead). Sometimes these metrics may contradict each other. For instance, to compute more accurate trust values an agent may have to interact with low quality agents. But interacting with low quality agents will result in lower utility gains. Alternatively, the agent could ask for more opinions, but in a highly deceitful environment this would not help. Specifically, accuracy and utility measure α's tendencies between exploration and exploitation. To obtain higher accuracy on one hand, agent α has to interact with unknown agents; it has to explore the space of interaction partners. But on the other hand, using the existing knowledge and exploiting it by interacting with highly capable agents will lead α to higher utility gains. A good decision making mechanism should balance the actions of an agent to get good results for every metric.

3.6. A walk-through example

The following example demonstrates how an evaluation run looks in practice. We have an environment of five agents $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5\}$ with corresponding capabilities *Capability*(a_1) = 0.10, *Capability*(a_2) = 0.25, *Capability*(a_3) = 0.50, *Capability*(a_4) = 0.75 and *Capability*(a_5) = 1.00 (for simplicity, agents have static capabilities and only one service exists, which is why we omit time and service parameters from *Capability*(a, s, t) notation and service from *Partner*(s, t) notation). Agent α uses a trust model that has a decision making mechanism that selects interaction partners, but does not determine who to ask for opinions. This example thus demonstrates ATB's mode *A*. The trust model computes trust degrees on scale [0, 1], while the decision making mechanism determines the following sequence of partner selections: *Partner*(1) = a_1, *Partner*(2) = a_2, *Partner*(3) = a_3, *Partner*(4) = a_4, *Partner*(5) = a_5, *Partner*(6) = a_5, *Partner*(7) = a_5 and *Partner*(8) = a_5. Table 1 shows the evolution of trust model estimations and the changes in accuracy and utility metrics.

In the beginning, the trust model assigns a default value, 0.50, to all agents, which results in *Accuracy*(0) = 0.50. Because α has not interacted with anyone yet, the utility remains *Utility*(0) = 0.00. At t = 1 agent α interacts with a_1. Since that agent is of poor quality as interaction partner (*Capability*(a_1) = 0.10), agent α lowers its trust degree from 0.50 to 0.37 (we are not concerned with *how exactly* α updates trust degrees, only that they *change*; we italicized changes in trust degrees in Table 1). This change increases both, the accuracy and the utility; the first rises to 0.70 and second to 0.10. Similarly, at t = 2 agent α interacts with a_2 and changes its trust degree from 0.50 to 0.42. This correction lifts accuracy to 0.85, while the utility rises to 0.18. Similar procedure repeats at every tick.

Two observations can be made from this example. First, the accuracy depends on the relative order of agents and not on the absolute similarity between trust degrees and capabilities. This is evident from t = 5 onward, where as trust degree toward agent a_5 becomes more and more similar to a_5's capability, the accuracy remains the same. Second, once α begins selecting the most capable agent as the interaction partner (a_5), the utility starts to converge towards 1.

Finally, when we present results in Section 4.5, we plot both metrics; Fig. 4 shows accuracy and utility for this walk-through example. The figure is a prototype of how we present results later. To ease comparison between different trust models, we compute a representative number for each metric. For accuracy, we compute an average value, 0.86, while for utility, we take the most recent result, *Utility*(8) = 0.69. This is because of the utility's nature, as given in Definition 3, that already includes all past partner selections, whereas the accuracy evaluates rankings for a particular tick. The representative numbers are shown in the legend.

4. Empirical evaluation

In this section, we describe how we used ATB to evaluate a set of five well-known trust models. The purpose of the evaluation is twofold, firstly to test the hypothesis claiming that attaching arbitrary decision making mechanisms to trust models can produce

Table 1
The computation of trust and the changes in accuracy and utility for the walk-through example. The italic values show changes in trust estimations.

Time (t)	Trust(a_x, t)					Accuracy (t)	Utility (t)
	a_1	a_2	a_3	a_4	a_5		
0	0.50	0.50	0.50	0.50	0.50	0.50	0.00
1	*0.37*	0.50	0.50	0.50	0.50	0.70	0.10
2	0.37	*0.42*	0.50	0.50	0.50	0.85	0.18
3	0.37	0.42	*0.50*	0.50	0.50	0.85	0.28
4	0.37	0.42	0.50	*0.58*	0.50	0.85	0.39
5	0.37	0.42	0.50	0.58	*0.66*	1.00	0.51
6	0.37	0.42	0.50	0.58	*0.75*	1.00	0.59
7	0.37	0.42	0.50	0.58	*0.80*	1.00	0.65
8	0.37	0.42	0.50	0.58	*0.83*	1.00	0.69

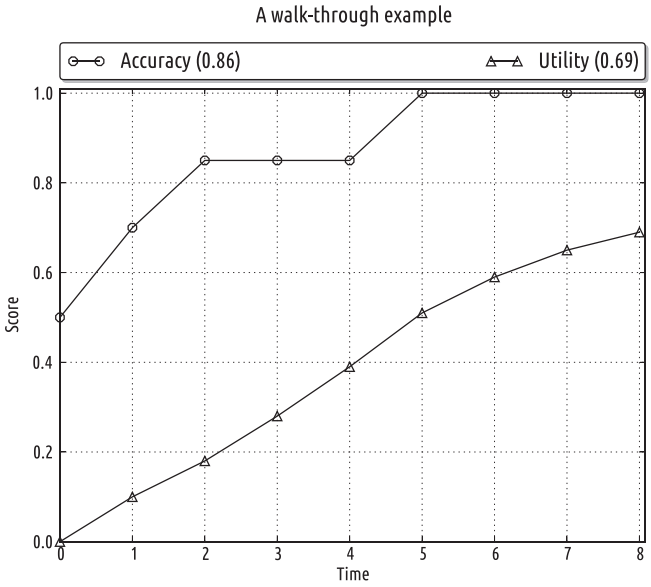

Fig. 4. Accuracy and utility for the trust model in the walk-through example. The values in the legend represent the average accuracy and the most recent utility.

ambiguous results and secondly, to demonstrate that ATB can be used with a wide spectrum of existing trust models. In this section we first state the hypothesis and propose a plan to test it, then we successively describe the used trust models, the used decision making mechanisms and the evaluation scenario. Finally, we present the results and discuss them.

4.1. Hypothesis and testing plan

We claim that *some decision making mechanisms are more suitable for certain trust models and may not work well with others*. To test this hypothesis, we have devised two experiments; in the first, we paired trust models with one decision making mechanism, and in the second with a different one. We reasoned that if our hypothesis was incorrect, we would get the same results in every experiment regardless of the decision making mechanism used. By the same results, we refer to the same ranking of trust models by their performances. However, if the results between experiments differed, our hypothesis must hold.

The decision making mechanisms were used only to select interaction partners. This meant using ATB in mode *A*, thus measuring accuracy and utility. More specifically, we compared accuracy and utility between trust models when using the first decision making mechanism, against accuracy and utility when using the other one.

4.2. Trust models

We chose the following trust models: Beta Reputation System [18], Travos [6], EigenTrust [19], and trust models of Abdul-Rahman and Hailes' [20], and Yu, Singh and Sycara's [21]. This selection represents enough diversity to cover both centralized and decentralized approaches, both qualitative and quantitative information representations and both types of trust models; those that have and those that do not have a decision making mechanism attached. In the next sections, we briefly describe these trust models. We do not focus on their details – for the details see the references to the publications associated to each model – but on their implementation within ATB. We describe the adapters for converting generated experiences and opinions to the formats that are compatible with the selected trust models; we describe the implementation of the "Interpretation" component from Fig. 1. Moreover, we have implemented trust models using recommended parameter values wherever given in the original publications or trying to find a sensible value for those that were omitted.

4.2.1. Beta reputation system (BRS)

Beta reputation system (BRS) [18] uses the expected value of the beta distribution to represent trust. Because of this, its trust degrees are real numbers from [0, 1]. BRS computes trust from agent's own experiences and from opinions from third-parties. Such information comes in the form of 2-tuples $\langle r, s\rangle$ that represent the amount of positive and negative feedback, respectively. In ATB, we compute these tuples from experience (opinion) tuples by setting r to the actual interaction outcome (trust degree), and setting s to its complement. For instance, we convert an experience tuple $\langle a, s, t, 0.8\rangle$ into $\langle a, s, t, \langle 0.8, 0.2\rangle\rangle$.

BRS uses a simple discounting procedure for handling false opinions. The discounting is based on the level of trust the BRS places in the agents that provide opinions. For instance, if BRS considers an agent to be very untrustworthy as a service provider, it heavily discounts its opinions. Such assumption is sometimes called *trust transitivity*, because it states that if an agent is trustworthy to provide a certain service it can also be trusted to provide good (honest) opinions.

4.2.2. Abdul-Rahman, Hailes (ARH)

The trust model proposed by Abdul-Rahman and Hailes (ARH) [20] uses qualitative information for computing and representing trust. In ARH, domains of trust degrees and assessments are the same: $\Omega = \Lambda = \{\text{vb} < \text{b} < \text{g} < \text{vg}\}$, where elements denote 'very bad', 'bad', 'good' and 'very good' degrees (assessments), respectively. We compute these values by dividing the interval [0, 1] into four sub-intervals. We tried three mappings: ARH_L: [0.00, 0.10) → 'vb', [0.10, 0.25) → 'b', [0.25, 0.50) → 'g', [0.50, 1.00] → 'vg'; ARH_M: [0.00, 0.25) → 'vb', [0.25, 0.50) → 'b', [0.50, 0.75) → 'g', [0.75, 1.00] → 'vg'; and ARH_H: [0.00, 0.50) → 'vb', [0.50, 0.75) → 'b', [0.75, 0.90) → 'g', [0.90, 1.00] → 'vg'. For instance, the interpretation components of ARH_L, ARH_M, and ARH_H, convert an interaction outcome 0.50 into 'vg', 'g', and 'b', respectively. The letters L, M, and H stand for mapping names; low, middle, and high.

ARH copes with liars by using a mechanism capable of correcting opinions. For instance, ARH can learn if an agent consistently badmouths other agents and adjusts its opinions accordingly. Additionally, ARH is the only tested trust model that separates trust by service types. This feature, however, is not highlighted in our experiments, because we used a scenario with only one type of service.

4.2.3. Yu, Singh, Sycara (YSS)

Yu, Singh and Sycara (YSS) [21] proposed a trust model for large-scale P2P systems. Since their trust model represents trust degrees and assessments as real values from [0, 1], we did not have to make any conversion.

YSS computes trust from experiences and opinions. In so doing, it computes confidence in its experiences, and if that confidence is sufficient, opinions are discarded. If confidence is not sufficient, YSS computes trust by combining experiences with opinions. The latter are discounted by using a variation of the weighted majority algorithm. This algorithm simply halves the credibility of agents, whose opinions turn out to be incorrect. This way liars lose credibility quickly. When computing trust from experiences, YSS provides two methods, simple averaging and exponential averaging. In our experiments, we used simple averaging.

4.2.4. Travos (TRA)

Travos (TRA) is a trust and reputation model for agent-based virtual organizations [6]. Similar to BRS it is based on the beta distribution and represents trust degrees as its expected value. Moreover, feedback in Travos is also represented in the form of 2-tuples $\langle m, n\rangle$, but contrary to BRS, Travos uses binary interaction outcomes. Thus $\langle 1, 0\rangle$ represents a satisfactory and $\langle 0, 1\rangle$ an unsatisfactory interaction. The interpretation component computes these tuples by thresholding the interaction outcomes; if the outcome reaches the threshold, we get $\langle 1, 0\rangle$, if not, $\langle 0, 1\rangle$. Like ARH, we present three thresholds; TRA_L thresholds at 0.25, TRA_M at 0.50, and TRA_H at 0.75.

Travos expects opinions as tuples $\langle r, s\rangle$ that contain the number of positive, r, and negative, s, past interactions. When α receives an opinion, say $\langle a_i, a_j, s, t, 0.60, 0.05\rangle$, the interpretation component *simulates* a number of interactions, 10 in our case,[1] of a_i with a_j by using truncated normal distribution. It sets the mean to the opinion's internal trust degree, 0.60, and the standard deviation to the same value that is used for generating experiences, 0.10. Each sampled number is then compared against the threshold to determine whether the interaction is satisfactory. This procedure assures that α obtains the same tuple – adjusted for the correctness of the given

[1] Alternatively, we could devise this number from the opinion's uncertainty, 0.05, where low uncertainties would indicate a high number, and high uncertainties a low number of interactions. But since we generate all opinions with the same level of uncertainty, this would add unnecessary complexity.

opinion – that would have been obtained if agent a_i had interacted with a_j 10 times and then reported the number of positive and negative interactions. For instance, with threshold 0.50, the opinion above would most likely be transformed into $\langle a_i, a_j, s, t, \langle 8, 2\rangle, 0.05\rangle$.

Like YSS, Travos computes confidence in its experiences and if confidence is not sufficient, it combines experiences with opinions. Additionally, it also uses a complex mechanism to reduce the effect of false opinions. If an opinion provider is deemed as a liar, Travos reduces the weight of its opinions. This principle is very similar to the one that is used by YSS, however, its implementation is quite different. While YSS uses a variant of the weighted majority algorithm, Travos manipulates parameters of the beta distribution.

4.2.5. EigenTrust (ET)

EigenTrust (ET) [19] is a trust model for P2P networks. It computes global trust values based on opinions from all peers in the system. An important aspect of EigenTrust is the notion of special peers that are pre-trusted. The trust in those peers has to be accurate, otherwise EigenTrust's computation method does not converge. Since EigenTrust paper does not specify how to determine such peers, we used the approach from Personalized EigenTrust [22], where each peer computes its own set of pre-trusted peers based on its own experiences.

EigenTrust uses binary interaction outcomes and computes local trust values in the form of net difference between the number of positive and negative interactions. If the difference is negative – more negative than positive interactions – EigenTrust assigns a local trust value of 0 to such peer. Because of this, it is said that EigenTrust does not measure negative trust [23], since it cannot differentiate between peers with whom it has had bad experiences from those with whom it has not interacted. In ATB, we convert information for EigenTrust the same as for Travos; we create a binary experiences by thresholding, while we use the same procedure to create opinions – EigenTrust also exchanges opinions in the form of tuples that contain the number of positive and negative past interactions. As with Travos, we present results for thresholds 0.25, 0.50, 0.75 by denoting them with ET_L, ET_M, ET_H, respectively. EigenTrust does not have any special mechanism to deal with false opinions. Similar to BRS, it considers trust to be *transitive*, and simply discounts opinions based on the level of trust it has in agents as service providers.

4.3. Decision making mechanisms

Of the aforementioned trust models, only EigenTrust provides a decision making mechanism. Moreover, it provides not one, but two mechanisms: a maximal selection and a probabilistic selection. The maximal selection selects the most trusted agent as the interaction partner, whereas the probabilistic selection selects randomly, but the probability of selecting a particular agent coincides with its trust level. This allows every agent to be selected, while assuring that highly trusted agents are selected more frequently. Maximal selection thus ensures maximum utility gains, while the probabilistic selection balances utility gains and exploration.

We used these two decision making mechanisms to test the hypothesis; we simply paired them with the tested trust models. The pairing was straightforward, except in the case of ARH. Because this model uses qualitative trust degrees, we had to create their numerical counterparts when making probabilistic selections. For this purpose, we mapped qualitative values to the lower bounds of the intervals to which they belong; for instance 'vg' maps to 0.50 in ARH_L, to 0.75 in ARH_M, and to 0.90 in ARH_H.

4.4. Evaluation scenario

We have evaluated trust models in a *Simple scenario*, which we describe below. Note that this is only one scenario and that the results reflect the performance of models under these specific conditions; to thoroughly evaluate all aspects of trust models, one should evaluate models in several scenarios, probing their resilience against various attacks and similar. However, this is not the purpose of this paper; we merely wish to show that decision making mechanisms can influence the performance of trust models and we use a minimal example to demonstrate the claim.

- The scenario contains 100 agents whose capabilities are assigned randomly using a uniform distribution. The set of agents and their capabilities remains unchanged during the entire evaluation run. Additionally, the scenario allows only one type of service.
- When assigning deception models, the scenario follows a transitive notion of trust. This means that the scenario assigns truthful deception models to agents whose capabilities are high and complementary deception models to agents whose capabilities are low. More precisely, the probability of assigning a truthful deception model is directly proportional to the capability of the agent that provides the opinion. The scenario assigns deception models at the beginning of the experiment and the assignments remain unchanged throughout the evaluation run.
- The scenario assumes full connectivity; every agent has an opinion about every other agent in the system.
- Each run lasts for 500 ticks. This gives models enough time to converge, and our experiments show that results are stable even if we prolong runs.
- The scenario generates experiences with a standard deviation of 0.10. When generating opinions, however, the standard deviation is set to 0.05 and, for simplicity, the same standard deviation is used for generating all opinions. We generate opinions with a lower deviation, to model that opinions, when truthful, are more accurate than a single experience.

4.5. Results

We have repeated each evaluation 30 times with different random seeds. We averaged the results and created *average runs* that are shown on Figs. 5–9. Each figure consists of part (a) and part (b). The first shows the results when using the maximal selection, while the second shows the results when using the probabilistic selection. Figures show the evolution of accuracy and utility, while the legends contain the representative values for each metric (in brackets). The results are summed up in Table 2.

When computing *average runs* we also computed standard deviations. They are drawn in light grey and they appear in lower parts of figures. To get an overall sense of a deviation, we computed a simple average of values on the deviation curves. They are also shown in the legends.

4.6. Discussion

The results supports the hypothesis that the decision making mechanism can influence the performance of trust models. If we rank trust models by their performances measured with utility, we get different results depending on which decision making mechanism trust models use. This can be seen by comparing the third and the fifth column of Table 2. For instance, when models use the middle thresholds (M) and we measure the utility, the best trust model with maximal selection is YSS, while the best model with probabilistic selection is EigenTrust. Moreover, we can compute correlation between results; Kendall tau rank correlation

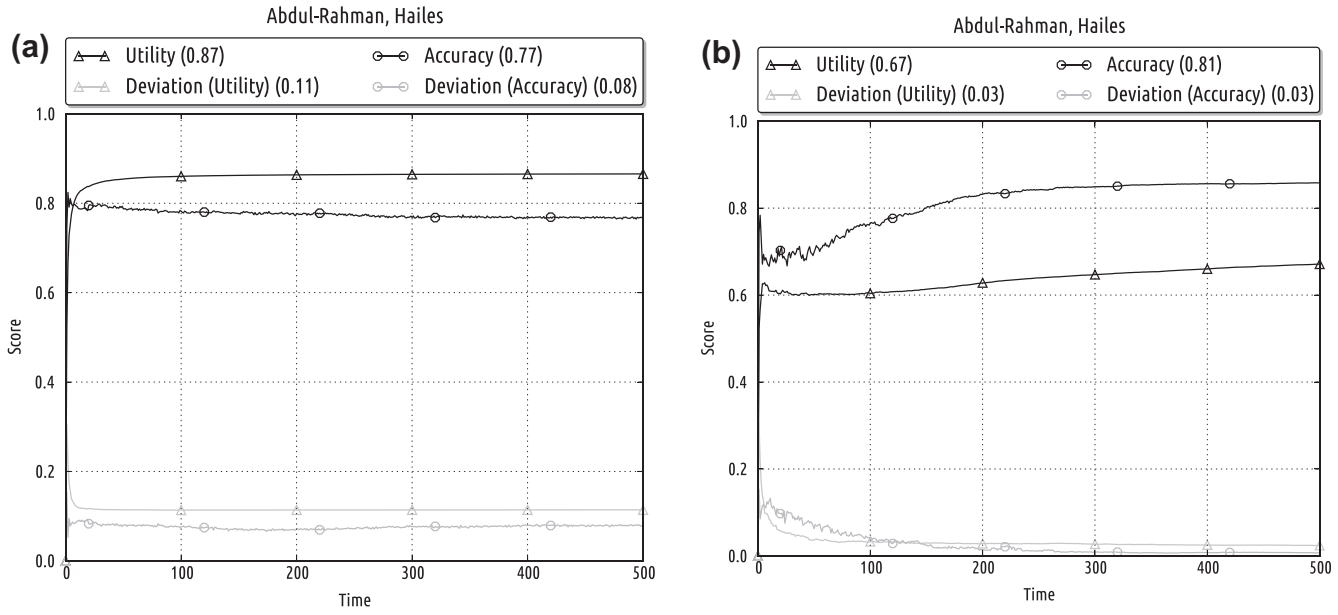

Fig. 5. Abdul-Rahman, Hailes with mappings of ARH_M. (a) Maximal selection. (b) Probabilistic selection.

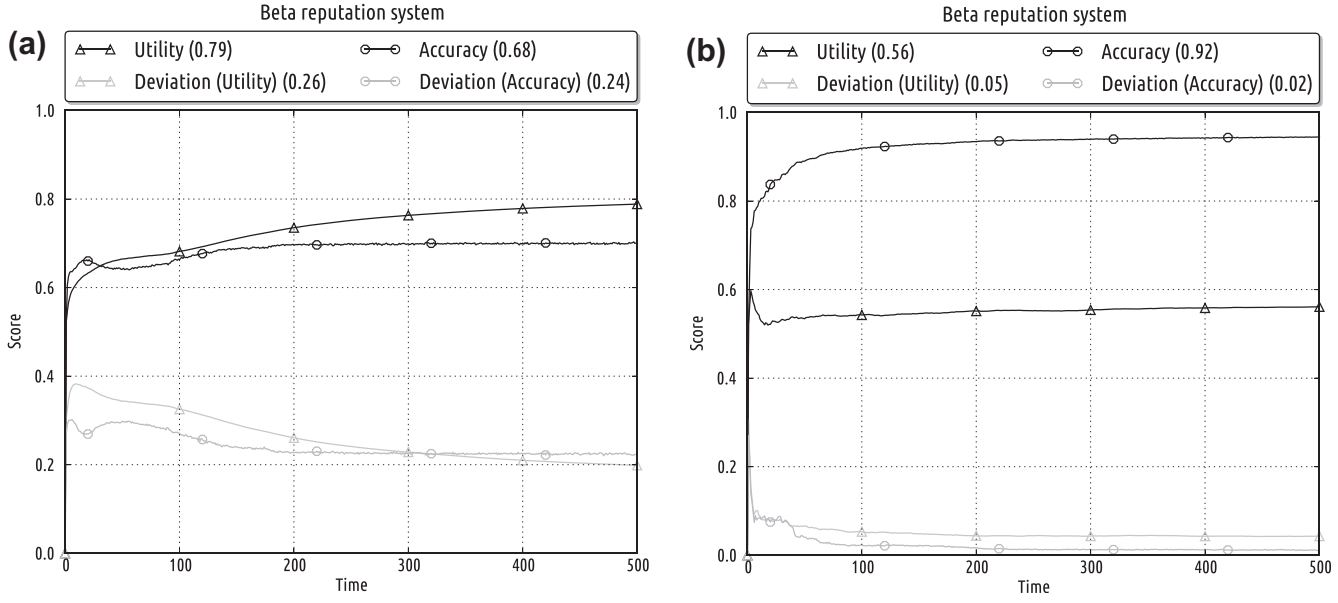

Fig. 6. Beta reputation system. (a) Maximal selection. (b) Probabilistic selection.

coefficient yields −0.2, which means that results actually show week inverse correlation. The hypothesis also holds when we compare performances with different thresholds; the correlations are 0.5 and −0.1 when thresholds are low (L) and high (H), respectively. If we compare performances measured with accuracy (the second and the fourth column in Table 2) the correlations for low, medium and high thresholds are 0.6, 0.4, and 0.6, respectively. This is a bit better, but the decision making mechanism still affects the results.

Models with thresholds, ARH, ET, and TRA, achieve the highest accuracy with medium (M) thresholds, while their utility rises as thresholds increase. Since medium thresholds divide the interval [0, 1] equidistantly, and since capabilities are assigned uniformly over [0, 1], equidistant sub-intervals provide models with the best data for estimations. For instance, if threshold is 0.75 then 75% of agents provide unsatisfactory interactions; while such agents have different capabilities, models hardly spot the difference between them. However, high thresholds help models estimate highly capable agents more accurately, which increases utility; it is easier to differentiate between highly capable agents if thresholds are high. This shows how models can be tweaked to achieve higher utility while lowering their accuracy. If we measured only utility, this observation would remain hidden.

When α uses the maximal selection it achieves higher utility. This is expected since probabilistic selection inevitably sometimes selects agents whose capabilities are low. However, when α uses probabilistic selection, trust models achieve substantially higher accuracy. This is also expected since probabilistic selection implies more exploring.

The results variate more when using maximal selection. This is especially evident with BRS and EigenTrust (Figs. 6 and 7). High variance comes from not exploring enough and not having enough

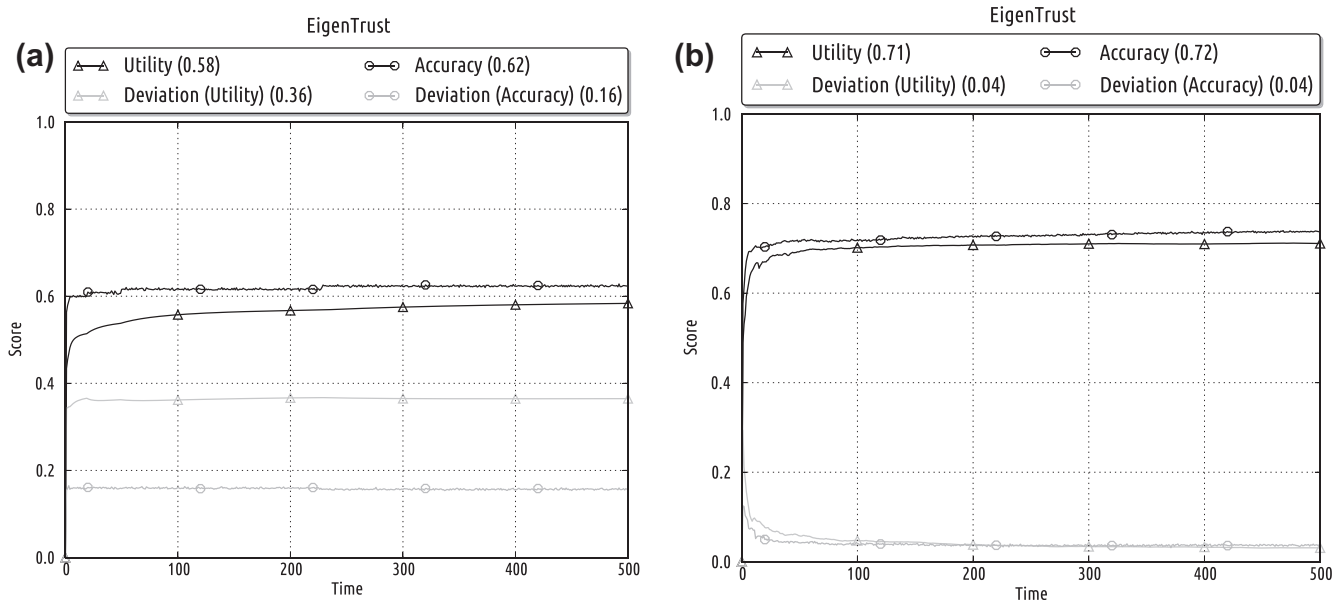

Fig. 7. EigenTrust with threshold 0.50 (ET_M). (a) Maximal selection. (b) Probabilistic selection.

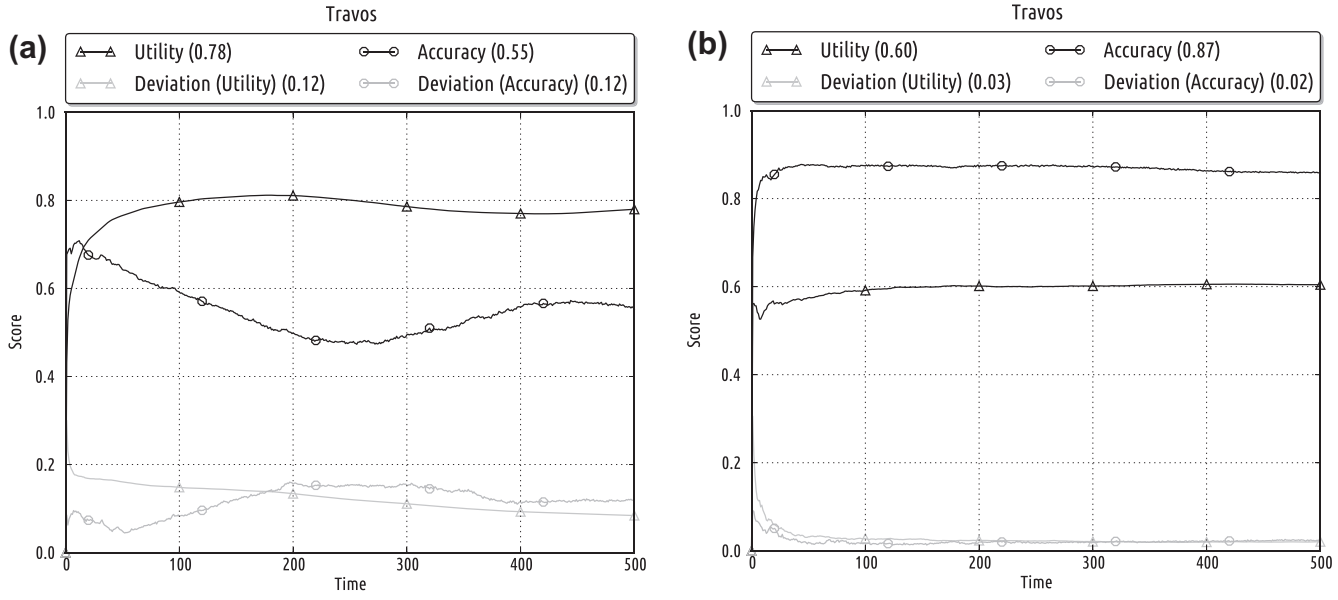

Fig. 8. Travos with threshold 0.50 (TRA_M). (a) Maximal selection. (b) Probabilistic selection.

local estimations. This has a domino effect in the aforementioned trust models. Because they use the transitive notion of trust, they weigh opinions based on the level of trust they put in opinion providers. If trust in the latter is incorrect – more likely, if agent does not explore enough – the mistakes propagate to opinions, which makes final trust estimations even more inaccurate.

Abdul-Rahman, Hailes and Yu, Singh, Sycara perform steadily with both decision making mechanisms. Both models achieve higher utility when using maximal selection and higher accuracy when using probabilistic selection. YSS's mechanism for dealing with false opinions seems to be particularly effective even with maximal selection. ARH uses qualitative values and its accuracy therefore limits around 0.86. This is expected since ARH labels agents that have similar capabilities with the same qualitative trust degree. For instance, ARH_M considers all agents whose capabilities fall inside [0.50, 0.75) equally trustworthy.

The decision making mechanism strongly influences the performance of Beta reputation system (BRS); BRS's accuracy rises from 0.70 to 0.92 when we switch from maximal to probabilistic selection. The reason for such difference lies in its opinion discount mechanism that requires some exploring in the beginning. Since maximal selection does not allow it, the combination of BRS and maximal selection is prone to errors.

Interestingly, when paired with probabilistic selection, BRS achieves high accuracy, but very low utility. This is counter-intuitive; good estimations should intuitively lead to good selections. To explain this, we have to look into BRS's computation method. In gross simplification, BRS assigns a default trust value of 0.5 to every agent. When the feedback is positive, BRS increments the trust value, but if the feedback is negative, BRS decrements it. In other words, every trust value starts with 0.5 and then moves either up or down. Because of this, all trust values are clustered

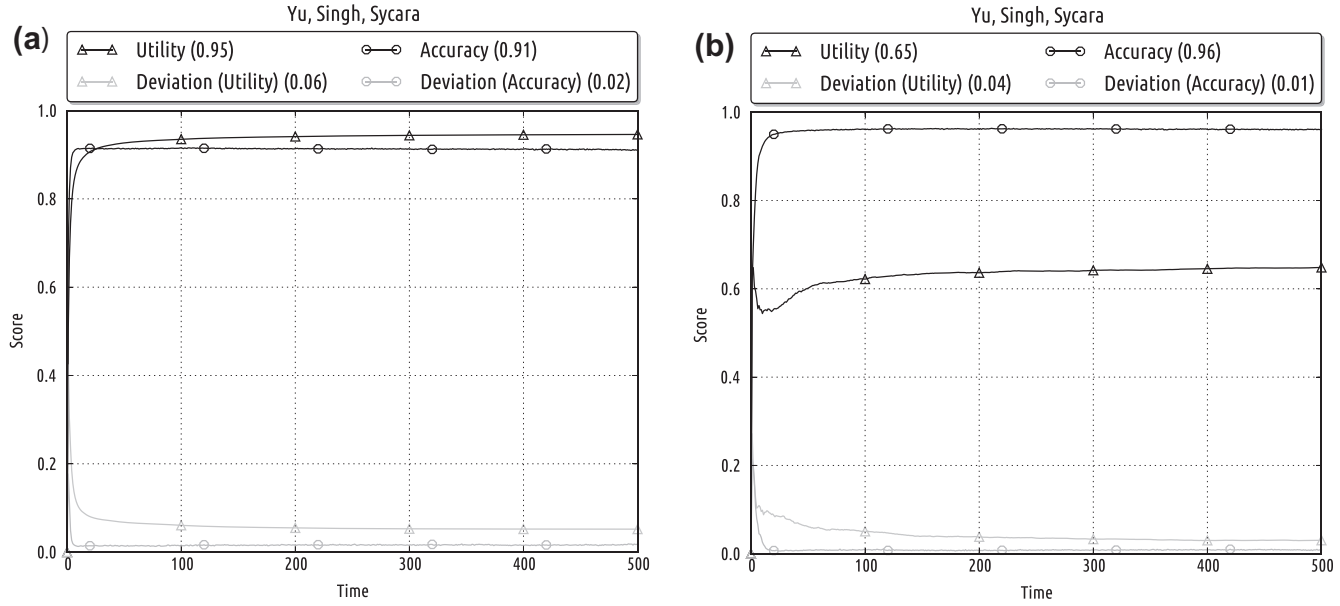

Fig. 9. Yu, Singh, Sycara. (a) Maximal selection. (b) Probabilistic selection.

Table 2
Summary of results. The choice of decision making mechanism influences the performance of trust models. The choice of mappings and thresholds is also influential; as trust models increase thresholds, they increase utility, but accuracy peaks when thresholds are in the middle (M); the numbers in parenthesis therefore show rankings with middle thresholds.

Trust model	Maximal selection		Probabilistic selection	
	Accuracy	Utility	Accuracy	Utility
ARH_L	0.62	0.79	0.75	0.62
ARH_M	0.77 (2)	0.87 (2)	0.81 (4)	0.67 (2)
ARH_H	0.73	0.94	0.78	0.74
BRS	0.68 (3)	0.79 (3)	0.92 (2)	0.56 (5)
ET_L	0.49	0.50	0.65	0.58
ET_M	0.62 (4)	0.58 (5)	0.72 (5)	0.71 (1)
ET_H	0.57	0.79	0.59	0.85
TRA_L	0.48	0.65	0.83	0.53
TRA_M	0.55 (5)	0.78 (4)	0.87 (3)	0.60 (4)
TRA_H	0.54	0.96	0.67	0.68
YSS	0.91 (1)	0.95 (1)	0.96 (1)	0.65 (3)

around 0.5, which means that numerical difference between the most trusted and the least trusted agent is rather small. When selections are probabilistic, highly trusted agents have only a slightly better chance of being selected than the highly distrusted ones. This means that highly capable agents are not selected as often as they would have been if trust values were more dispersed. This is the reason why BRS achieves low utility despite having high accuracy.

Travos varies in performance; with maximal selection its accuracy is poor and while probabilistic selections do increase it, Travos achieves good accuracy only with medium thresholds. But when thresholds are high, Travos collects the most utility with maximal selections (0.96), although YSS (0.95) and ARH_H (0.94) trail closely. In this case, Travos is able to select good partners in spite having low overall accuracy; its trust estimations towards highly capable agents – capabilities from [0.80, 1.00] – are accurate, while estimations towards remaining agents are not. This results from having a high threshold and from Travos' opinion discount mechanisms; only highly capable agents produce satisfactory interactions and, consequently, Travos learns to correctly discount only highly positive opinions.

Travos also suffers from using binary values that carry less information than real ones. Because of this, Travos may sometimes fall in a *local trap*. A local trap occurs when a trust model estimates a certain agent is the most trustworthy and it keeps selecting that agent as the interaction partner, although that agent may not be the most capable one. Say an agent has a capability 0.75 and Travos uses threshold 0.50. If at the beginning – when Travos does not know how to discount opinions – opinions suggest that this agent is the most trustworthy, Travos will select it as the interaction partner. The interaction will most likely be positive and Travos will actually increase the trust value towards that agent. In the next tick, the situation will repeat. And again in the successive ticks. Eventually this agent will have the maximal possible trust value; even though there are other agents with higher capabilities. But if the threshold is high, only highly capable agents will produce positive interactions, so such cases are less likely to evolve. Alternatively, the local trap can be avoided if the model explores enough (like it happens with probabilistic selection).

The decision making mechanism most notably influences the results of EigenTrust. When using maximal selection EigenTrust collects the least utility, but when paired with probabilistic selection EigenTrust achieves the best score. This is because EigenTrust requires some exploration, which is not available when using maximal selection. In this case, EigenTrust produces extreme results; they are either very good or they are very bad (few are in between). However, the results on average are worse than from any other model. But when we combine it with probabilistic selection, EigenTrust achieves the best utility.

Like BRS, EigenTrust's good result with probabilistic selection is counter-intuitive. While having good utility, EigenTrust achieves very poor accuracy; how can a trust model make good selections if its computations are off? Again, the explanation is hidden in the trust model's computation procedure. EigenTrust is known for not measuring *negative* trust. In ATB, this translates into giving local trust score 0 to every agent whose capability is below 0.5. In other words, EigenTrust considers agents as completely untrustworthy if their capabilities are below 0.5. Once EigenTrust labels such agents with local trust value 0, they get a near 0% chance of being selected when selections are probabilistic. This means that EigenTrust can only select agents, who have above average capabilities, which inherently leads to higher utility gains. But at the same

time EigenTrust cannot accurately rank agents whose capabilities are below 0.5, which leads to low accuracy. This duality is shown in Fig. 10.

This is a telling example how measuring utility gives only partial results; utility only evaluates the quality of selections, while the accuracy evaluates the quality of all trust estimations.

We highlight the results in the following points.

1. Decision making mechanism plays an important role in the evaluation of trust models, because it determines the ratio between exploration and exploitation. This can influence the performance of trust models.
2. Some decision making mechanisms are more suitable for certain trust models and do not work well with others. Forcing arbitrary selection procedures on trust models can thus yield ambiguous results.
3. Utility metrics only evaluate the quality of partner selections and ignore the remaining trust estimations. We advocate for metrics like accuracy that evaluate the entire output of trust models.

5. Implementation

This section shows an architectural overview of ATB and describes how ATB could be extended to accommodate trust models that compute trust from other information sources.

5.1. Architecture

ATB consists of four basic components: an evaluation protocol, a scenario, a trust model, and a set of metrics. In brief, the scenario generates information from which the trust model computes trust that is then evaluated by various metrics. The evaluation protocol binds these components together by defining when and how they communicate. The evaluation protocol is instantiated and ran by an infrastructure facility. The architecture of ATB is shown on Fig. 11.

As the infrastructure we use Repast Simphony framework [24]. It provides basic facilities needed for a simulation, such as time management, graph plotting, batch-run support, and similar. Note that the infrastructure only invokes the evaluation protocol and it is otherwise completely detached from other components.

Evaluation protocols, recall Section 3.1, determine the steps of the evaluation run. They also serve as an intermediary between trust models, scenarios and metrics. Evaluation protocols therefore define the types of scenarios, trust models and metrics that can be used together. For instance, the evaluation protocol for mode *B* requires the type of scenario that generates experiences for selected interaction partners and opinions from selected opinion providers, the type of a trust model that selects both interaction partners and opinion provides, and the types of metrics that evaluate accuracy, utility and opinion cost. In ATB, defining the types of components means defining their Java interfaces; specifying their methods. Being an intermediary, the evaluation protocol also defines data structures that are exchanged between components.

Trust models are represented as implementations of interfaces that are specified in evaluation protocols. Typical methods of trust models include `processExperiences()`, `processOpinions()`, `getTrust()` and similar. If a trust model does more than compute trust (if it selects interaction partners for instance) it should have methods like `getNextInteractionPartner()`.

Scenarios are also implementations of interfaces specified in evaluation protocols. A scenario would have methods like `generateExperiences()`, `generateOpinions()`, `getCapabilities()` and similar. Scenarios should generate experiences and opinions as described in Sections 3.2 and 3.3, however, a developer is free to use different approaches – the only limitation is the expressiveness of the Java language. Internally, scenarios must handle the population of agents, assignment of their capabilities and deception models as well as other aspects of the agents.

Similarly, metrics are implementations of interfaces specified in evaluation protocols. Often a metric will take a piece of information from the scenario (like capabilities) and a piece of information from the trust model (like trust) and then evaluate those pieces in some form. For instance, the accuracy in Definition 2 computes correlation between capabilities and trust.

Since implementations of trust models, scenarios and metrics are simple Java classes, ATB can load them dynamically as plugins. All a developer has to do is (i) implement a component's interface, (ii) annotate the implementation with a standard Service Pro-

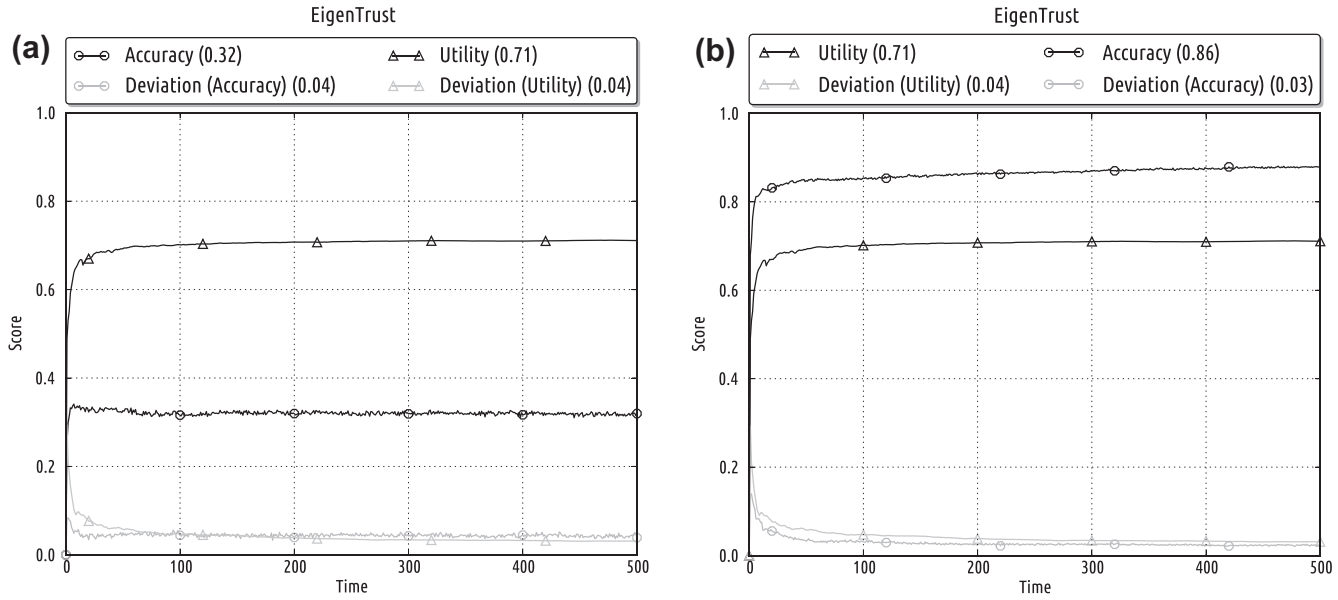

Fig. 10. EigenTrust is unable to evaluate *negative trust*. Estimations towards agents with low capabilities (left) are very inaccurate, while remaining estimations (right) are quite accurate. (a) Maximal selection. (b) Probabilistic selection.

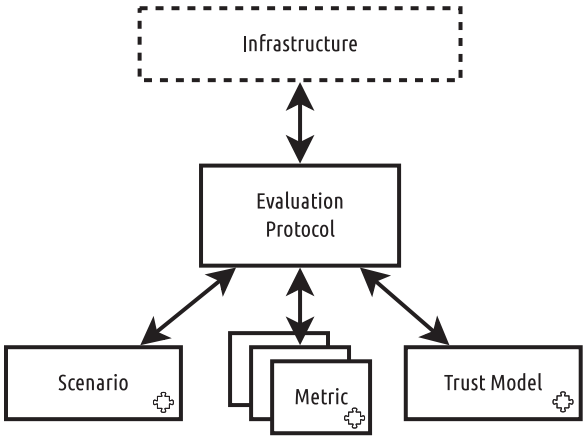

Fig. 11. The high-level components of ATB: an evaluation protocol, a scenario, a trust model and a set of metrics. The scenario generates information from which the trust model computes trust, while the metrics evaluate it. The evaluation protocol links these components together and the infrastructure instantiates and runs the evaluation protocol.

vider configuration file [25] and package it in a JAR, and (iii) place the JAR inside ATB's classpath.

5.2. Extendability or how to evaluate different kinds of trust models

ATB can cover a variety of trust models; besides those presented in Section 4 even more recent proposals such as HABIT from Teacy et al. [26] or TRR from Rosaci et al. [27] seem to be testable with ATB's design. However, such models need to be compatible with existing evaluation protocols. But if a researcher wishes to evaluate a trust model that is not compatible with those protocols, she needs to devise a new one. Here we describe how one would devise an evaluation protocol – as well as other components – to evaluate trust models that use information from social networks.

We motivate the design on the case of Regret by Sabater and Sierra [28]. Regret is a trust model that estimates trust from experiences, opinions, and social relations between agents. The latter are represented as directed and weighted graphs called sociograms. Regret uses sociograms to decide who to ask for opinions and how to assign weights to opinion providers.

First, a researcher would have to define a new data type to represent sociograms. Next, she would have to define a new type of scenario that generates sociograms; let us call it sociogram-enabled scenario. Its interface might extend the interface for scenario in mode B by adding a method named `generateSociograms()`. Analogously, the researcher would define an interface for a sociogram-enabled trust model; it would contain a method likely named `processSociograms()`. Finally, the researcher would create a new evaluation protocol that queries the sociogram-enabled scenario for opinions, experiences, and sociograms and then conveys them to the sociogram-enabled trust model. As metrics, existing solutions would all be feasible.

Once the new interfaces and new evaluation protocol have been defined, the researcher would have to implement Regret as a plug-in. Finally, she would have to implement (at least) one sociogram-enabled scenario. Naturally, the implementation would have to be aware of sociograms; the scenario would have to generate capabilities and deception models appropriately, so that the information encoded in sociograms would be meaningful.

Note that the work described above is more involved than implementing new trust models, scenarios or metrics. However, it can be done without modifying existing code (except for a few configuration settings).

6. Related work

In this section we first survey the existing testbed proposals. Then we compare ATB's main characteristics with existing solutions. Finally, we discuss ATB's limitations.

6.1. An overview of existing testbeds

Many researchers, for instance [6,7], built their own ad hoc testing environment to show how effective their proposals are. Such tools can demonstrate the effectiveness of the proposed models, however, they cannot be used as general purpose testbeds, since they are model specific; they evaluate specific aspects of trust models. Moreover, most ad hoc implementations are not publicly available, which makes it even more difficult to compare results.

Few researchers have addressed the problem of building a general purpose testbed. The Agent Reputation and Trust (ART) testbed [8] has been a courageous and well-known initiative to fill this gap. Authors proposed a testbed, in which agents enact the role of art appraisers. Agents estimate the worth of paintings by either using their own judgment (determined by the testbed) or by asking other agents for help. Agents can use trust models to i) evaluate how other agents can help them in estimating prices of paintings and, to ii) decide how opinions from other agents can help them in finding additional appraisers. During evaluations the testbed pays agents for their appraisals in proportion to how accurate their appraisals are. Moreover, agents can use earnings to purchase opinions or appraisals from other agents and thus improve the quality of their own appraisals. At the end of the competition, the agent with the most earnings wins.

The setting of ART testbed is a good example of a competitive multi-agent environment, in which agents need complex reasoning to perform well. During the evaluation, agents have to appraise paintings, query other agents for appraisals, and buy and sell reputation information. They can use (or not) the trust model in any or every task, which blurs trust model's effect in the final evaluation. The ART authors themselves pointed out [29] that the most successful competitors did not concentrate much on developing trust models, but more on deciding how to invest their earnings. Because the evaluation domain was complex, the competitors of ART competition shifted their focus from working on trust models to trying to boost the overall performance of their agents. Thus the results did not reflect the quality of the trust model but the overall reasoning capabilities of the agent that used it.

The success of ART testbed inspired Kerr and Cohen to create a more general and focused testbed: Trust and Reputation Experimentation and Evaluation Testbed (TREET) [30]. TREET simulates a marketplace where sellers and buyers exchange items. Buyers are motivated by the value of items, while sellers are motivated by the profit they make from sales. Sellers can cheat by not shipping the item and so increasing their profits. While TREET defines items that each seller can sell at the beginning, buyers get a different *shopping list* in each tick of the simulation. Buyers then use trust models to select the most trustworthy sellers. TREET also models the information lag by not letting buyers know whether they have been cheated until some time has passed. Based on this feedback, buyers update trust calculations. These are then used for selecting sellers in the future. The main metric in TREET is the ratio of sales (profits) between honest and cheating sellers.

According to its authors, TREET resolves many limitations of ART; it supports both centralized and decentralized systems, allows collusion attacks to be easily implemented and does not impose the format of trust values. However, like ART, TREET is very domain specific and, more importantly, its metrics do not evaluate the quality of computed trust, but the quality of trust-based deci-

sions; the quality of selecting honest buyers. In so doing, the authors used the maximal partner selection procedure.

TRMSim-WSN [9,31] is a dedicated testbed for benchmarking trust and reputation models intended for wireless-sensor networks (WSNs). The testbed creates a WSN, equips nodes with trust models and defines which sink-nodes will behave benevolently and which maliciously. Then it simulates WSN traffic. The nodes have to send packets to benevolent sink-nodes and avoid malicious ones. Testbed's main metric is accuracy, which tells the percentage of nodes that send packets to benevolent sinks.

Metrics in TRMSim-WSN are inherently based on decision making. However, in comparison to ART testbed, calculated trust is the only considered factor. While it is not specifically mentioned (although it can be inferred from inspecting TRMSim-WSN's source code) the testbed allows trust models to use their own selection procedures. TRMSim-WSN has a similar limitation as TREET; it is highly domain oriented and it evaluates the quality of trust-based decisions.

In the domain of P2P networks, West et al. introduced a P2P simulator (P2P-Sim) [23]. P2P-Sim is a trust management simulator that simulates a P2P network. In these networks, each peer has a trust management system that helps it to avoid peers that share corrupt files. The P2P-Sim is based on the trace/simulator paradigm. This means that the scenario of the evaluation is defined in a trace file that is generated before the simulation. Traces contain information about the number of network nodes, their behavior models, the files they currently have and the files that they wish to obtain. Once a trace is generated, the simulator loads it and assigns the tested trust model to every node in the P2P network. During simulation, nodes download files while using their trust models to find the most trustworthy peers. The performance of a trust model is measured with the effectiveness metric, which is defined as the proportion of valid downloads over all attempted downloads between peers that the testbed defines as *good nodes*.

P2P-Sim is well-documented, comes with good examples and has a very clear and straightforward metric. Its authors acknowledge that one of its limitations is that the simulated networks are closed meaning that nodes cannot dynamically leave or enter the network. Additionally, trust models need a partner selection procedure in order to select a peer to download from. Hence P2P-Sim measures the quality of trust-based decisions and not the quality of trust estimations. Moreover, we think that P2P-Sim models malicious nodes inappropriately. While *good nodes* download from the most trustworthy peers, *malicious nodes* – in their quest to further pollute the network – download from the least trustworthy ones. This is inappropriate, because the actual level of maliciousness depends on the performance of the trust model. If a trust model is bad at measuring negative trust (such as EigenTrust), then malicious nodes are not as effective as they would be if they used a trust model without such a limitation. This means that trust models are not tested under the same conditions.

Recently, Chandrasekaran and Esfandiari [32] proposed a novel testbed evaluation model that is based on graph theory. Their idea is to construct three directed graphs successively, namely, a feedback graph, a reputation graph and a trust graph. In all graphs, vertexes represent agents, while the interpretation of edges varies between graphs. In the feedback graph, edges represent feedback from past interactions. Using a reputation algorithm, this graph is converted to the reputation graph. The reputation graph is a directed weighted graph, where edges represent local trust values. Finally, using a trust algorithm the reputation graph is converted to a trust graph. The trust graph is a normal directed graph, where edges between agents represent (binary) trust. An edge from agent a to agent b means that agent a fully trusts agent b, an absence of an edge implies that a totally distrusts b. Once the trust graph is constructed, they plan to launch an attack and then recalculate the three graphs. As a metric, they propose to evaluate changes in the trust graph, after the attack has been launched.

Their proposal is interesting because it is general. However, at the time of this writing, the model is purely theoretical and no experimental results exist yet. Thus its usefulness remains to be proven. Additionally, the authors acknowledge that constructing a trust graph is something that few proposals provide. To remedy that, they propose their own implementations of trust algorithms that would be applicable to any existing trust model. While this is not the same as ad hoc implemented partner selection procedures, ad hoc implemented trust algorithms – that are not part of the original proposals – have the potential to influence results.

Very recently, Salehi-Abari and White [33] introduced a framework for a distributed analysis of reputation and trust (DART). They base their approach on games of iterated and generalized prisoner dilemma. DART arranges agents into a dynamic and undirected graph, where nodes represent agents and edges between agents indicate that agents can interact and communicate. Every agent thus maintains a list of connections to other agents, known as its neighborhood. An agent then plays rounds of iterated prisoner dilemma games with agents that are part of its neighborhood. The outcomes of these games represent direct interactions. Additionally, every agent can ask its neighbors for opinions about third-parties or it can ask its neighbors about direct interactions that they had with third-parties. Finally, an agent can suggest its neighbor to include another agent into its neighborhood or, analogously, an agent can accept such an invitation from a neighbor. For making such decisions agents use perception models (trust or reputation models) and various policies. The main metrics are the utility that the agents obtain in direct interactions and the number of agents that the agents remove from their neighborhoods. On a system level, DART offers a view of the society in the form of a graph. This can be used to see whether the tested agents allow bad agents in their neighborhoods and similar.

DART provides a flexible, multi-faced and general environment for evaluating performance of agents. However, its evaluation model concentrates around trust-based decisions. In DART, every agent has a perception model and a set of policies. Agents thus combine information from perception models with policies to reach various decisions. And there are many decisions to be made: how to play in PD games, who to ask for opinions, when to add or remove an agent from a neighborhood, when to lie, when to tell the truth and similar. Such tight coupling between perception models and policies blurs the effect of the perception model on the final agents performance; it is hard to tell whether the measured performance is due to the well functioning of the perception model, to the chosen policies, or to both. Because certain policies may favor some trust models, as we have shown in this paper, such evaluation can yield ambiguous results. However, due to its generality, DART is a good framework for evaluating the combined performance of perception models and policies. If a trust model comes with a set of such policies, DART would be a good tool to test it.

The overview above reveals that testbeds use different metrics. We can classify them into two groups, individual metrics and system metrics. Individual metrics measure the performance of one entity using a trust model, whereas system metrics benchmark the performance of the entire system, where trust model is used by several entities. An example of the individual metric are the earnings of an agent in the ART testbed, while the accuracy of routing in a WSN or the ratio of valid file downloads in a P2P network are instances of system metrics in TRMSim-WSN and P2P-Sim, respectively.

6.2. Comparing ATB against existing testbeds

ATB most differs from existing testbeds in three ways: how it evaluates trust models, how it controls the environment of other

agents, and how it operates in a general domain. We discuss these differences below.

Existing testbeds use a similar evaluation model; they measure the quality of trust-based decisions that agents with trust models make – they do not evaluate the calculated trust directly. The problem with evaluating decisions is that they can only be taken if the trust model has a decision making mechanism that determines how to incorporate calculated trust in decision making processes. This is rare and therefore testbeds combine trust models with ad hoc decision making mechanisms to overcome this problem. As we have shown, the election of decision making mechanism does matter and can influence the evaluation results. ATB differs because it does not require a decision making mechanism for evaluation. However, if a trust model does have a decision making mechanism, ATB can evaluate the combined effect of the trust model and the decision making mechanism.

ATB also differs in managing *other agents* – agents that only provide data to the agents that are being evaluated. Existing testbeds usually provide a multi-agent system or an agent-based simulation, where other agents reason, sense, and communicate. Sometimes other agents even use tested trust models to strengthen their benevolence or malevolence; for instance in P2P-Sim all agents use trust models, because malicious agents purposely download content from – according to their trust models – the least trustworthy peers to further pollute the network. In such environments, the agent being evaluated is part of the system; it receives information from other agents and also provides them with information. Other agents can then adjust their behavior accordingly. The point here is that the tested trust model – or the agent that is using it – *influences* the behavior of other agents. Different trust models will influence other agents differently, and consequently, different trust models will be evaluated under different conditions. In ATB, the only *real* agent is agent Alpha, other agents are simulated. They behave according to their capabilities, deception models and other scenario parameters; other agents do not reason, sense or communicate. In effect, they are puppets obeying their master, the scenario. Notice that in experiments described in Section 4, all trust models received exactly the same opinions – the experiences differed, however, because α was allowed to select interaction partners. Had we evaluated trust model with evaluation protocol designed for trust models without decision making mechanism (described in Section 3.1.1), agent α would receive the same opinions and experiences, regardless of its trust model.

Finally, ATB also does not enforce any particular domain, such as an electronic commerce or a P2P network. Instead, it provides general and abstract information that can be fed to various trust models, even if they were intended for different domains. While ATB is not the only general testbed, for instance the testbed from Chandrasekaran and Esfandiari [32] and DART [33] are also general, many other proposals are domain specific.

6.3. Potentially untestable trust models and other limitations of ATB

While ATB has notable benefits, it is also limited in certain aspects. This section thus discusses ATB's limitations. First, being general, ATB cannot accommodate domain specific peculiarities of trust models. For instance, in Regret [28] the transaction outcomes consists of seller's tendencies to overcharge, to deliver late and to deliver low-quality items. Each of this aspects has a certain value and Regret defines ways how to combine them in the final assessment; ATB does not support such sub-assessments.

Second, because only one agent, α, uses the tested trust model and all other agents are simulated, there is no sense in using system-level metrics – that is metrics that benchmark the performance of the entire system. Introducing such metrics is sensible only if all agents use the tested trust model. However, as described in previous section, if *other agents* use the tested trust model (or any other signal from the evaluated trust model), this means that the trust model interferes with the evaluation. Different trust models will interfere differently, which means that different trust models will be evaluated in different conditions. Moreover, to benchmark the entire system, agents need more than trust models; they have to be full-blown agents with reasoning capabilities that go beyond trust estimations.

Finally, ATB is unfit to evaluate trust models that either (i) represent information (opinions, experiences) in formats that substantially differ from the ones used in ATB, or trust models that (ii) exchange or compute opinions with special protocols that cannot be straightforwardly recreated by interpretation components. An example of the first limitation are trust models that represent opinions with various ontological structures, while an example of the second limitation, are trust models that expect opinions to be computed with certain – usually trust model specific – procedures; such trust models usually expect that all agents use the same trust model. For instance, in our evaluation we converted opinions for models YSS, ARH and BRS straightforwardly, while opinions for Travos and EigenTrust required additional work in the interpretation component.

7. Conclusion

In this paper we exposed several limitations in the existing trust evaluation techniques. We argued that their main drawbacks are their evaluation methods that evaluate trust models indirectly by evaluating the quality of trust-based decisions. To reach those, testbeds pair trust models with ad hoc decision making mechanisms. We hypothesized that results of such evaluations can be ambiguous, because some decision making mechanisms might be more suitable for certain trust models and might not work well with others.

We created a new trust testbed, the Alpha testbed, that can evaluate trust models in three ways: when a trust model is without a decision making mechanism, ATB evaluates the quality of computed trust values; when a trust model has a decision making mechanism that only selects interaction partners, ATB additionally evaluates the quality of trust-based decisions, and finally; when a trust model has a decision making mechanism that besides selecting interaction partners also determines who to ask for opinions, ATB also evaluates – besides the quality of computed trust and the quality of trust-based decisions – the cost of fetching requested opinions. Using ATB, we tested our hypothesis on five trust models. We made two separate evaluations, in which we changed the decision making mechanism that trust models used. We found out that the performance of trust models depended on the election of the decision making mechanism. We attributed this mostly to the fact that decision making mechanism determines the ratio between exploration and exploitation of trust models, which is essential for their well-functioning. These results confirmed our hypothesis and given that, we claim that it is only valid to evaluate the pair of (trust model, decision making mechanism) if they are proposed together as a whole. If that is not the case, trust models should not be evaluated by coupling them with arbitrary decision making mechanisms. Instead, they should be evaluated independently of the decision making mechanism as it is possible with ATB.

To the best of our knowledge, existing solutions evaluate trust models by measuring the quality of trust based decisions and not the quality of the calculated trust. In this regard, ATB is the first that evaluates calculated trust directly while not enforcing any specific trust degree representation (except that trust degrees have to be mutually comparable). Moreover, we are also the first to

empirically support the claim that evaluating trust models by evaluating decision making mechanisms can lead to ambiguous results.

Currently, the testbed can evaluate trust models that compute trust only from experiences and opinions. Because some models also use additional information, such as the information from the social networks or virtual organizations, we plan to extend ATB to accommodate such models. Additionally, as we opensource the implementation, we are setting up a web page that shall hopefully serve as a forum for other potential ATB users. This page could connect researchers and enable them to share their own implementations of trust models, metrics and evaluation scenarios. An idealized goal is to create a corpus of testing artifacts that may serve as a bench-marking suite for trust models; something similar to the publicly available data sets in the pattern recognition community.

Acknowledgments

The authors have been supported by the following institutions: David Jelenc by the Slovenian Research Agency (Grant 1000-09-310289) and the Agreement Technologies COST Action IC0801 (STSM Grant COST-STSM-ECOST-STSM-IC0801-160511-003839); Ramón Hermoso by the Spanish Ministry of Science and Innovation (Projects OVAMAH, Grant TIN2009-13839-C03-02; co-funded by Plan E) and the Spanish Ministry of Economy and Competitiveness through the Project iHAS (Grant TIN2012-36586-C03-02); Jordi Sabater-Mir by the CBIT Project (TIN2010-16306), the Agreement Technologies Project (CONSOLIDER CSD2007-0022, INGENIO 2010), the SINTELNET coordinated action, and the Generalitat de Catalunya (grant 2009-SGR-1434); and Denis Trček by the Slovenian Research Agency (research program Pervasive computing P2-0359).

References

[1] K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attack and defense techniques for reputation systems, ACM Computing Surveys (2009).
[2] S.D. Ramchurn, D. Huynh, N.R. Jennings, Trust in multi-agent systems, Knowledge Engineering Review (2004).
[3] J. Sabater, C. Sierra, Review on computational trust and reputation models, Artificial Intelligence Review (2005).
[4] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems (2007).
[5] I. Pinyol, J. Sabater-Mir, Computational trust and reputation models for open multi-agent systems: a review, Artificial Intelligence Review (2011).
[6] W.T.L. Teacy, J. Patel, N.R. Jennings, M. Luck, Travos: trust and reputation in the context of inaccurate information sources, Autonomous Agents and Multi-Agent Systems (2006).
[7] R. Hermoso, H. Billhardt, R. Centeno, S. Ossowski, Effective use of organisational abstractions for confidence models, in: Proceedings of the 4th European Workshop on Multi-Agent Systems EUMAS '06, 2006.
[8] K.K. Fullam, T.B. Klos, G. Muller, J. Sabater, A. Schlosser, K.S. Barber, J.S. Rosenschein, L. Vercouter, M. Voss, A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies, in: Proceedings of the 4th International Joint Conference on Autonomous Agents and MultiAgent Systems, 2005.
[9] F.G. Mármol, G.M. Pérez, Trmsim-wsn, trust and reputation models simulator for wireless sensor networks, in: Proceedings of the 2009 IEEE International Conference on Communications, 2009.
[10] D. Jelenc, R. Hermoso, S. Ossowski, D. Trček, Alpha test-bed: A new approach for evaluating trust models, in: Proceedings of The Third International Workshop on Infrastructures and Tools For Multiagent Systems, 2012.
[11] D. Trček, Towards trust management standardization, Computer Standards & Interfaces (2004).
[12] F.G. Mármol, G.M. Pérez, Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems, Computer Standards & Interfaces (2010).
[13] D. Trček, An integrative architecture for a sensor-supported trust management system, Sensors (2012).
[14] I. Pinyol, J. Sabater-Mir, G. Cuni, How to talk about reputation using a common ontology: from definition to implementation, in: Ninth Workshop on Trust in Agent Societies, 2007.
[15] B. Yu, M.P. Singh, Detecting deception in reputation management, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, 2003.
[16] P. Resnick, R. Zeckhauser, Trust among strangers in Internet transactions: empirical analysis of eBay's reputation system, in: The Economics of the Internet and E-Commerce, Emerald Group Publishing Limited, 2002.
[17] R. Kumar, S. Vassilvitskii, Generalized distances between rankings, in: Proceedings of the 19th International Conference on World Wide Web, 2010.
[18] A. Jøsang, R. Ismail, The beta reputation system, in: Proceedings of the 15th Bled Electronic Commerce Conference, 2002.
[19] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proceedings of the 12th International Conference on World Wide Web, 2003.
[20] A. Abdul-Rahman, S. Hailes, Supporting trust in virtual communities, in: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.
[21] B. Yu, M.P. Singh, K. Sycara, Developing trust in large-scale peer-to-peer systems, in: IEEE First Symposium on Multi-Agent Security and Survivability, 2004.
[22] N. Chiluka, N. Andrade, D. Gkorou, J. Pouwelse, Personalizing eigentrust in the face of communities and centrality attack, in: IEEE 26th International Conference on Advanced Information Networking and Applications (AINA), 2012.
[23] A.G. West, S. Kannan, I. Lee, O. Sokolsky, An evaluation framework for reputation management systems, in: Z. Yan (Ed.), Trust Modeling and Management in Digital Environments, IGI Global, 2010.
[24] M.J. North, T.R. Howe, N.T. Collier, J.R. Vos, A declarative model assembly infrastructure for verification and validation, in: Advancing Social Simulation: The First World Congress, 2007.
[25] Oracle, Jar file specification – service provider, Java SE 6 Documentation, 2011.
[26] W.T.L. Teacy, M. Luck, A. Rogers, N.R. Jennings, An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling, Artificial Intelligence (2012).
[27] D. Rosaci, G.M.L. Sarné, S. Garruzzo, Integrating trust measures in multiagent systems, International Journal of Intelligent Systems (2012).
[28] J. Sabater, C. Sierra, Reputation and social network analysis in multi-agent systems, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, 2002.
[29] M. Gómez, J. Sabater-Mir, J. Carbó, G. Muller, Analysis of the agent reputation and trust testbed Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial (2008).
[30] R. Kerr, R. Cohen, Treet: the trust and reputation experimentation and evaluation testbed, Electronic Commerce Research (2010).
[31] F.G. Mármol, G.M. Pérez, Trust and reputation models comparison, Internet Research (2011).
[32] P. Chandrasekaran, B. Esfandiari, A model for a testbed for evaluating reputation systems, in: IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2011.
[33] A. Salehi-Abari, T. White, Dart: a distributed analysis of reputation and trust framework, Computational Intelligence (2012).