A Translation Mechanism for Recommendations

Pierpaolo Dondio, Luca Longo and Stephen Barrett

Abstract An important class of distributed Trust-based solutions is based on the information sharing. A basic requirement of such systems is the ability of participating agents to effectively communicate, receiving and sending messages that can be interpreted correctly. Unfortunately, in open systems it is not possible to postulate a common agreement about the representation of a rating, its semantic meaning and cognitive and computational mechanisms behind a trust-rating formation. Social scientists agree to consider unqualified trust values not transferable, but a more pragmatic approach would conclude that qualified trust judgments are worth being transferred as far as decisions taken considering others' opinion are better than the ones taken in isolation. In this paper we investigate the problem of trust transferability in open distributed environments, proposing a translation mechanism able to make information exchanged from one agent to another more accurate and useful. Our strategy implies that the parties involved disclose some elements of their trust models in order to understand how compatible the two systems are. This degree of compatibility is used to weight exchanged trust judgements. If agents are not compatible enough, transmitted values can be discarded. We define a complete simulation environment where agents are modelled with characteristics that may differ. We show how agents' differences deteriorate the value of recommendations so that agents obtain better predictions on their own. We then show how different translation mechanisms based on the degree of compatibility improve drastically the quality of recommendations.

Distributed System Group
Department of Computer Science and Statistics
Trinity College Dublin, Dublin 2,
e-mail: dondiop@cs.tcd.ie e-mail: longol@cs.tcd.ie e-mail: stephen.barrett@cs.tcd.ie

1 Introduction

An important class of distributed Trust-based solutions is based on information sharing. This class encompasses Recommendation, Reputation systems and Social Networks. A basic requirement of such systems is the ability of participating agents to effectively communicate with others, receiving and sending messages that can be correctly interpreted. Unfortunately, in open systems, it is not possible to postulate a common agreement about the representation of a rating, its semantic meaning, the cognitive and computational mechanisms behind a trust-rating formation. The mentioned problem is logically precedent to any other: if agents cannot understand each other, the feasibility of these trust solutions is fatally undermined, even when all agents are acting honestly. Social scientists agree to consider unqualified trust values not transferable and computational trust researchers are aware of the not completely transitivity of trust, proposing conditions and uncertain models like the one by Josang [1]. A more pragmatic approach would conclude that qualified trust judgements are worth being transferred as far as decisions taken considering others' opinions are better than the ones taken in isolation.

Our research seeks to define a distributed translation mechanism able to make information, exchanged from one agent to another, more accurate and useful. We define the problem by analysing the potential differences that two trust systems may encounter. A generic trust systems may be depicted as a multi-layer architecture encompassing: a trust value representation, a trust metric used to compute such values, a trust ontology that defines concepts quantified by the trust metric and a function of satisfaction that represents how an agent consider the interaction performed. Moreover, each agent is interacting in a specific domain and therefore it is equipped with a domain representation (for instance an ontology) and a collection of past experiences. Each of these level may be affected by differences, as we describe in section 3. Referring to an eBay-like scenario, an agent may rate very high a seller for its low packaging time, while another may consider this information marginal as far as the item sold is of good quality. In section 4 we review the state-of-the-art solutions that focus mainly on the definition of a common ontology for reasoning about trust. Our complementary strategy implies that the parties involved disclose some elements of their trust model in order to understand how compatible the two systems are. This degree of compatibility is used to weight exchanged trust judgements. If agents are not compatible enough, transmitted values are discarded, while values received from highly compatible agents are strengthened. The strategy has an unsupervised nature that, under some conditions, skips or limits the ontology processing. We define a complete simulation environment where agents are modelled with characteristics that may differ. For instance, our simulator allows agents to have different trust metrics, thresholds and functions of satisfaction. Its distributed nature allows agents to have a personal database for pieces of evidences. In our evaluation we start considering some aspects of the problem, restricting the case where the agents share the same domain representation and they have trust metric exclusively dependant on a combination of direct past experience and recommendations. Further experiments

are left for future works. This paper is structured as follows: in section 2 we review how the trust transferability problem is discussed among social scientists, in section 3 we define the problem by analysing which differences may affect a generic model of a trust system, then in section 4 we review present and potential solutions. In section 5 we describe the simulator environment while in section 6 we describe our evaluation, defining goals and metrics used. Conclusions and future works end our contribution.

2 Trust Transferability in Computational Trust

Before asking how we can effectively transfer trust, a first question is whether trust has a degree of objectivity. Studies in social science seem to agree about the subjective nature of trust. In the classical definition by Gambetta [5], this is represented by the subjective probability that the trustor assigns to the trustee, that vary according to the trustee, the situation and the level of perceived risk. Any attempt at objective measurement can dangerously mislead agents into thinking that the value is transferable and be used by another trustor, which is not true for trust. In other words, trust is not transitive, which has also been formally shown in [7]. As Luhmann wrote [8]: *Trust is not transferable to other objects or to other people who trust*. To say that one trusts another without further qualification of that statement is meaningless. But, on the contrary, the success and the diffusion of systems like Social Networks or Ratings Systems make the problem worth to be investigated. Therefore, the problem is to qualify correctly trust judgements, and build a mechanism to translate values produced by two different systems to allow meaningful communications. Josang and Pope [1], investigated the transferability of trust by analysing under which formal condition trust may be considered transitive. Their conditional transitivity construct adds conditions for considering trust values, propagated by transitivity, more plausible. The concept is present also in Abdul-Rahman and Hailes distributed trust model [2]. The conditional transitivity requires that:

- A has direct knowledge of B
- B has direct knowledge of C
- A has knowledge of B as a recommender
- A can use B's trust value in C

Using Joasng words: "a transitive trust path therefore stops ... when there are no more outgoing referral trust", where referral trust is the trust in an agent as a recommender. These works clearly shows how trust transferability is not a valid concept, but a plausible one that deserves to be investigated. By respecting additional conditions and by adding explanation about the semantic meaning of its rating, an agent should consider transferred trust value still useful in its decision making process.

3 Defining the problem

In order to focus the problem, in figure 1 we depicted a high-view of the components of a trust-based system. At each level, differences among the systems may cause lack of trust transferability. In order to keep our discussion more realistic, we often refer to a scenario where agents have to select trustworthy open software. Each system is depicted as a multi-layer pile with the components described in the follow:

- *Domain Perception*. It includes all the information that an agent has available about the domain where it is interacting. It is generally composed by a domain ontology and a facts database (or evidence DB).
  - *Domain Ontology*: agent's representation and understanding of the domain it is interacting in. We model it as an ontology that describes terminology and relationships related to the domain under analysis.
  - *Evidence DB:* the facts the agent collected from its experience that are used to ground trust-based decisions. A possible evidence could be: "The component1 by developer XYZ crashed twice in the last two weeks".
- *Trust System*. It contains the agent notion of Trust, its computational model represented by a trust metric, a trust value representation, a decision making process and a function of satisfaction.
  - *Trust Value*: the quantification of the trust judgement.
  - *Trust Ontology (model of trust)*: agent's definition of trust in the context. It implies to define which are the elements composing its notion of trust. We can generalize a trust model as an ontology. An example of trust model, for open software, could be: trust is a combination of stability of the software, programming techniques used and experience of the authors or in absence of past evidences a recommendation from other users.
  - *Trust metric*: the actual computation performed over the available inputs to generate a trust value. The trust metric specifies how trust model's elements are converted into numerical values and how each of them concurs to the final aggregated trust value. Different metric has been defined, even starting from the same inputs.
  - *Function of Satisfaction*: each agent has a mechanism to evaluate the quality of an interaction just completed with the trustee agent. We model this mechanism as a function defined as follows: $S : O \rightarrow [0,1]$ that goes from the set O of possible outcomes to a value that represents agent's level of satisfaction associated with a specific outcome. The function models an essential concept in trust computation: the trustor should have a mechanism to understand if the trustee fulfilled its expectations, quantified by the function S. Using function S trustee's trust value can be automatically updated according to trustor level of satisfaction. For example, an agent may be satisfied if the software has clear comments and it does not crash in the first two months and therefore he can decide to give to that software a high rating.

– *Decision making process*: this component, also referred as trust management, describes how the agent exploits the computed trust values to support its decision making process. Generally, a threshold T defines the minimum trust value required to start an interaction. The decision making process could be more articulated and it is usually represented by a set of policies (see for instance the SECURE policy language [11]) or a set of (fuzzy) rules like in the REGRET trust model by Sabater [3].

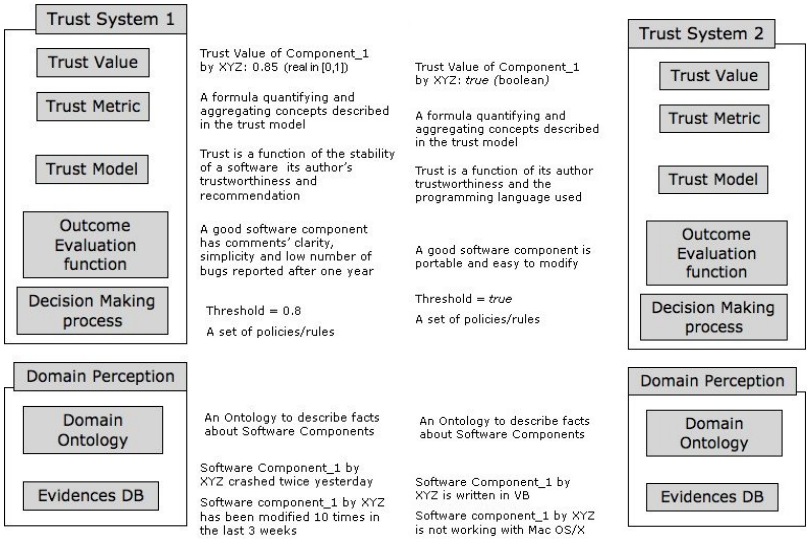

Fig. 1 Trust Transferability scenario

In any of the above levels, differences may reduce the transferability of trust.

- *Trust Value differences.* The two systems may have a different values representation. For example, a system may have trust value represented with a real number in the range [0, 1] while the other may represent values with five discrete levels from 0 to 4. A second problem is the identification of the agent whose trust values belong to: the two systems may have different names for the same agent or the same label could not be bounded to the same agent.
- *Trust metric differences.* Even when the two agents are using the same trust representation, i.e., they have the same trust ontology, they may compute trust in different ways. An entity may compute 90% of its trust value on the number of bugs reported, while another may assign to it a marginal role.
- *Trust models (ontology) differences*. Systems may have different representation and understanding of what trust is. An agent may define trust as a prediction of software quality based on its author, while another agent may include many other factors, like the author popularity, the stability and persistence of the software or the programming technique used.

- *Different Threshold.* An agent can be more optimistic about and have a lower threshold than another one, that may have more strict requirements in order to start cooperating. This implies that an exchanged trust value could be sufficient for A to start an interaction even if for B was not enough, leading to substantial differences in agents' behaviour.
- *Different function of Satisfactions.* Two agents may judge differently the same interaction because, for example, of different expectations. A user may not care about the clarity of the code or comments in a software component, as far as the component does not crash, while another entity may consider confusing code as an important negative factor, maybe because he wants to modify it.
- *Domain Representation differences.* Agents may have different knowledge of the domain structure in which they are interacting. An agent may have a limited knowledge, ignoring elements that are known to other agents. Two entities may have different names and representations for the same concept, define diverse relationships among domain elements and use concepts that are aggregation of elements, so that the same evidence using agent A's domain ontology can only partially described by agent B.
- *Evidence Database differences.* Two agents may have a dissimilar database of evidences. In a distributed scenario this is a common situation: each entity has a limited set of experience and its partial vision of the world.

4 State-of-the-art solutions

A general solution implies that the parties involved disclose some elements of their trust model in order to understand how similar the two systems can be considered. Referring again to figure 1, a matching between two systems can be performed at various levels: trust values, trust model, evidences.

Trust value matching. At the level of trust value representation, a conversion may be performed in order to reduce trust values to a common representation. Since during the conversion some information is lost, an estimation of this loss has to be part of the process. Pinyol et al. [3] performed an analytical study of trust value conversion. The authors consider four common trust value representations: boolean, bounded real, discrete sets and probabilistic distribution representation. The authors proposed a conversion between these four representations and they define a factor taking into account the uncertainty involved in the conversion. They propose to compute this factor considering the entropy of the trust value seen as a random variable. As it is easy to understand, high level of uncertainty are associated when a conversion is from a simpler to a richer representation, while no uncertainty is associated with the opposite conversion.

Trust model matching. ElMessery [4] proposes to enhance trust values by declaring the expectations that the trustee has. In this way, it is possible to understand

which elements may affect positively a trust value or not from the point of view of that particular trustee. If two agents share the same expectations, it is likely that they will judge situations in similar ways and consequently trust values could be transferred. The proposed solution has, as a condition, the fact that the two systems are able to understand the terminology (and the semantic) of each trust model. Methodology that are based on ontology matching have a great potential, but the problem that so far has not been investigated in trust studies and that represents a challenging task where some results has been achieved in a specific domain with semi-automatic tools. Another possible solution is the definition of a generic and basic ontology for trust representation that may act as a starting point where personalised ontology may be matched in order to be compared. This ontology does not want to force agents to a common trust model, but offer a generic and flexible mean of communications of trust information among agents. In Computational Trust this problem has been studied using high-level concept present in any trust-based decisions. The European project eRep [9] defines a set of terms about reputation concepts has been defined. The aim of this effort is to define an ontology that all partners participating in the project could use as a consensual starting point. This ontology is based on the cognitive theory of reputation defined by Conte and Paolucci [10]. Punyol at al. [3] propose a generic ontology as a possible common base for mapping different trust models. The authors describe a generic belief about an entity as a combination of SimpleBelief, a belief that the holding agent acknowledges as true, and MetaBelief, a belief about others' belief.

Evidence matching and unsupervised learning of similarity: extending collaborative filtering. If agents are acting in the same or similar domain, it may be likely that they encountered similar situations. A degree of similarity can be deduced by simply matching evidences and corresponding trust values. This could be performed in several ways as we describe in the next section, but the common idea is to compute a degree of compatibility based on common data or behaviours in recognized situations. Leaving details in the next section, a basic difference between these classes of solutions and the previous ones, is that it does not take in consideration the elements of a trust model, but it focuses on the comparison of common data in order to understand the similarity between the two systems. It is therefore an implicit and unsupervised solution that limits the need of a common ontology, but that still requires that, at least, the two agents speak a language partially understandable.

Privacy issues. The disclosure of extra information rather than a trust value implies, at least, an issue of privacy. One or both the parties may not want to share information about their trust judgements or their reasoning model. The problem was described by Seigneur [6] as the trade-off between trust and security: in order to trust an agent we need information that may reduce its privacy.

5 Our strategy

Our solution requires that two parties disclose some information in order to understand the degree of compatibility of their trust systems. Common to all the following solutions is the hypothesis that agents have been designed to interact with each other in a domain. We therefore assume that their domain representations partially overlap. Differences can arise in their preferences, trust models or their past experiences, but we postulate that they are partially able to communicate facts happening in the domain: this means that the domain ontology partially matches. Each strategy should be evaluated at least with the following criteria: impact on the quality of recommended trust values, privacy issues, risk and communication overload. The choice of adopting a specific strategy will be a trade-off dependant on situational factors like bandwidth, need for recommendations, privacy constraints. In this first paper we analyze three strategies: the sharing of trust values database or past interactions database, the direct comparisons of the function of satisfaction S or the trust metric T in presence of an accepted common domain ontology and the approximation of the function S or T using stereotypes situations without common ontology. In all the strategies the agents' goal is to get an idea of other agent's trust metric T or function of satisfaction S. Note how these goals are only partially linked: in the first case the agent is interested in understanding how the other agent assigns trust value, while in the second it is interested in other agent's preferences. Knowing T may not guarantee the expected S, exactly like knowing that a person is considered very trustworthy for another person does not guarantee that it will satisfy my expectations. Knowing the preferences of another agent (function S) does not guarantee on its ability to predict others trustworthiness. We are now ready to describe these three preliminary solutions, remembering that this paper, as the first on the topic, does not claim to be comprehensive.

- 1a. *Sharing of Trust Value DB*. In this strategy agents share a DB containing at least the couple $< agent\ name, trust\ value >$. The idea is that two agents check if they have some acquaintances in common, and they use common connections' trust value to compute a compatibility degree. A hypothesis is that agents are using the same ID for the same agent, hypothesis not always valid. Some statistical indicators like correlation, can be used, and supplemental information like number of accepted/rejected interactions with an agent can make the computation more plausible. If agents have different trust value representations, a conversion may be performed as described in Punyol at al. [3]. This strategy predicts the agents trust metric T, it does not require any knowledge of the Trust model and according to the number of agents in common could became an accurate indicator. On the contrary, privacy is very poor; communication overload can be heavy like the risk involved. Several systems have been implemented to add an extra-layer of security to guarantee the confidentiality of the information shared that relies on trusting computing and encryption keys policies.
- 1b. *Sharing of Evidences*. This solution is similar to the first one, but the data shared are single interactions and how each agent evaluated each interaction.

In this way agents predict the function S rather than T. Communication overload is even bigger, but privacy concerns are less relevant since evidences are anonymous in the sense that they describe situations rather than agents' personal information.

- 2. *Direct Comparison of function S*. When there is a common ontology describing facts that is accepted by all agents, each of them can easily map its function S over this common ontology and directly compare it with the others'. The simplest case, that we evaluate in section 6, is when function S has the same basic form (for example a linear combination of factors). An example of such an ontology is the recent evolution of the eBay feedback system, where four fixed criteria have been introduced to assess the validity of an item sold, representing a first common base for comparing feedbacks. By directly comparing the two functions, agents compute an accurate degree of compatibility, without disclosing sensitive information about other agents or personal experience, and with a few communication overload (unlike the previous two solutions). On the contrary, the hypothesis on which this solution relies can be hard to satisfy.
- 3. *Predicting S and T using stereotypes situations*. Solution 1 scales poor and suffers from privacy constraints. Solution 2 is better in any respects, but it requires the strong hypothesis of a common ontology for outcome evaluations. When there is no common ontology, but agents, at least, partially can understand each other, a solution can be build by using stereotypes situations. Here we describe the prediction of the function S, but the method can be applied to the prediction of the trust metric T using trust values instead of values of satisfaction and stereotypes agents instead of stereotype situation. Agents' goal is to accurately predict other agent's function S using the minimum number of messages. In the generic situation, each function S is any function defined from some domain concepts or elements to a value

$$S_1 : f(X_1, X_2, ..., X_n) \;\; S_2 : f(Y_1, Y_2, ..., Y_m) \tag{1}$$

We assume that, if the agents have different value representation, they translate it using the technique described in [3]. Each agent sends stereotypes situation that it considers meaningful to the other agent and wait for its evaluation of the situation. For instance, an agent considering the low shipping time, essential for being a good eBay seller, may propose two situations where this factor varies drastically. Agents can propose situations where only one key-factor changes, in order to understand the importance of that specific factors, with the drawback of not understanding the mutual dependence of the factors in the formula. In general, agents need a strategy to generate the appropriate next situation after having received the other agent's feedback. The strategy should indicate when the process should stop, i.e., enough information has been collected to understand other's agent model. In general, agents may employ an unsupervised learning system or adopt statistical tools like regression and correlation to understand other's agent reasoning model, performing an on-the-fly negotiation of their preferences. The solution appears a good trade-off between the previous ones: using stereotypes

situations sensitive data are not disclosed, communication overload is relatively small, varying from the perfect situation of solution 2 to the case where many messages have to be exchanged in order to understand other agents. Number of messages will in general depend on how close the two agents representations are, how many of the situations proposed are relevant and fully understood. It may happen that an agent may reply not with a value but with an "Unknown situation" message if it was not able to understand the specific situation proposed. The analysis of this issue requires further investigation beyond the scope of this paper.

6 Our simulator

In order to test the validity of our strategy, we designed a complete simulator where agents' community is divided in Buyers and Sellers. A Seller is modelled as a function that defines the quality of the items he can sell. Each item is described as a n-tuple $(f_1, f_2, .., f_n)$ of factors indicating item quality. In our evaluation, we propose an eBay-like scenario where each item is described by the five eBay factors: (1) *Item as described* (2) *Communication* (3) *Shipping Time* (4) *Shipping Cost* (5) *Pricing*. Each value is in the interval $[-1,1]$. Therefore the seller function S is defined as follows:

$$Q_R : \Re^n \rightarrow [-1,1] \tag{2}$$

When a buyer decides to buy from a seller, a seller will produce a n-tuple describing the quality of the item purchased. The buyer will then assess its satisfaction. A seller produces its n-tuple as follows. Each quality factors f_i is a uniformly distributed random variable in $[base\ value\ \pm\ service\ variance]$. Base value is a uniformed random variable in $[-0.9, 0.9]$. It is decided at the start of the simulation and it does not change, meaning that a seller does not change its average quality of service during the simulation. The service variance is a uniformed variable in $[0, 0.3]$ decided at start-up, that models how variable the service provided by a seller is, introducing a more realistic change variation. Values lower than -1 and value greater than 1 are rounded to -1 and 1. A seller is therefore modelled by an n-tuple $< base\ value, service\ variance >$, therefore 2^n value. A buyer is modelled with the following functions: a trust metric producing a trust value for a seller and a function of satisfaction that evaluates the degree of satisfaction of the buyer after purchasing an item. The buyer decides to buy only according to the seller trust value computed. Each buyer has a threshold T, the minimum seller's trust value needed to buy from it. After deciding to buy or not, the 5 quality factors of the item sold are disclosed to understand if the buyer's decision was correct. Each buyer has a local database of trust values shown in table 1. It also keeps a history DB of its transactions described in table 2 and a third database containing all the buyers known with a degree of similarity between it and other buyers, as depicted in table 3.

Seller ID	ID of the seller
Trust Value Local	a trust value for the seller
Num_Transaction	number of transactions engaged with that seller
Num_Transaction_OK	number of item purchased from that seller

Table 1 Local buyer's database

Seller ID	ID of the seller
Time	time of the transaction
n-tuple	description of the item engaged with that seller
Purchased	boolean value, true means item purchased from that seller
Level of satisfaction	a value in [0, 1], described as follows, representing the level of satisfaction of the buyer about the item

Table 2 Local buyer's history database

Seller ID	ID of the seller
Compatibility_Value	a value in [-1, 1] that represents how the two buyers are comparable (1: very similar, 0: non comparable, -1: opposite).

Table 3 Local buyer's similarity database

6.1 Computations

Computation of Trust Value. In this first evaluation, the trust metric will be dependant only on direct and indirect experience, i.e., recommendations. In details, the trust value is computed as follows:

$$T_{value} : i \cdot T_{value}^{local} + (1-i) \cdot T_{value}^{recommended} \tag{3}$$

where T_{value}^{local} is the value stored in the trust-value database of the buyer; $T_{value}^{recommended}$ is the value that the buyer collects from other buyers and i is a uniform random variable in $[0.1, 0.9]$, assigned at the start of the simulation to the buyer, that represents in which proportion recommended value and local trust value influence the final trust value. The local trust value is totally dependent on direct past experiences between the buyer and a specific seller. In absence of interactions, the buyer uses a dispositional trust value that is a property of each buyer. The dispositional trust value is again a uniformed random variable in [0.1, 1], representing buyer with a strong disposition to trust (value close to one) or diffident buyers (close to 0.1). After an interaction, the local trust value is updated like this:

$$T_{value}^{local} : m \cdot T_{value} + (m-i) \cdot V_{satisfaction} \tag{4}$$

where m is uniformly distributed in $[0.1, 0.9]$ representing the effect of memory, i.e. how much the last interaction count on the new local trust value. The value of satisfaction is a value in $[0, 1]$ computed as described in the next section. The recommended trust value is computed as follows: the buyer asks to all the buyers in

the community if they know the seller. All the buyers that know the seller will reply submitting their local trust value and the number of transactions used to compute that trust value. The recommended value is a weighted average of all of these trust values using the number of interactions completed.

Value of satisfaction computation. For each transaction a value of satisfaction of the buyer is computed in any case (item sold or not). Each buyer has a vector of preferences defined as a n-tuple of value between $[-1, 1]$, uniformly distributed random variable decided at the simulation start-up for each buyer. This vector is in correspondence with the seller's n-tuple of item quality factors. Each value of the buyer's n-tuple represents how important it considers each factor. A value of 1 means that the factor is essential for its satisfaction, a value of 0 means that the factor does not influence the buyer' s satisfaction and a value of -1 means that the buyer satisfaction increases when the correspondent factor is low (remind that quality factors are between 0 and 1). The negative values are inserted to model the situation where two buyers may have opposite view about a quality factors. For instance, the fact that in a movie there is a specific actor may be a strong negative factor for a person and a negative for another one. We can visualize the computation of the value of satisfaction using a vector representation. The n-tuple of vector preferences of the buyer and the n-tuple of the quality factors generated by the seller for an item are two vectors in an R^n space, P and Q_s. It seems appropriate to model the computation of the value of satisfaction as the scalar product between the two vectors. As displayed in figure 2, the scalar product is maximum when the two vectors have the same direction and versus, equal to zero when they are orthogonal and minimum when they are opposite. These three circumstances correspond to a situation where the quality of factors and the preferences are very similar (fig. 2 dx), not comparable (centre) and opposite (sx). By normalizing the vectors with then R^n norm, the value of satisfaction Vs is generally a value in $[-1, 1]$. In our case, it is a value in [0, 1] because we discarded negative numbers for the preferences. The level of satisfaction is used, in case the buyer decided to buy from that seller, to update the local trust value of the buyer associated to that seller. Finally, we summarise, in table 4, the random variable that defines a buyer: these variables are all local to each buyer.

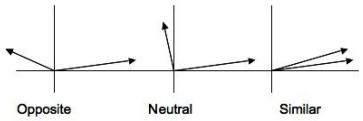

Fig. 2 Buyer dissimilarities in the preferences space

Similarity Value Computation. This value is computed using the strategy 2, by directly comparing the functions S of the two buyers, represented by a linear combination of the 5 quality factors. Thus the computation is a scalar product among two 5-dimension normalized vectors and values are properly uniformly re-distributed in $[-1, 1]$.

Tr	Agent's threshold [0,0.8]
I	proportion direct/indirect experience in [-0.7, 0.7]
Dispositional Tvalue	Dispositional trust value
M	buyer memory [0.1, 0.9]
n-tuple preferences	preferences coefficients in [-1, 1]

Table 4 Random variable for a buyer

6.2 How the simulation works

1. A couple buyer, seller $< B_{uyer}, S_{eller} >$ is selected
2. B_{uyer} computes the trust value of S_{eller}
   - a) B_{uyer} retrieves the local trust value
   - b) B_{uyer} collects the recommended trust values
   - c) B_{uyer} computes the degree of compatibility for each recommending Buyer
   - e) B_{uyer} computes the trust value
3. IF $T_{value} >$ threshold $\geq B_{uyer}$ THEN buys from S_{eller}
   - a) S_{eller} generates item quality factors n-tuple
   - b) B_{uyer} computes its level of satisfaction
   - c) B_{uyer} updates its local trust value for S_{eller}
   - d) B_{uyer} updates its interaction DB

   ELSE: e. b. d. (trust value not updated)

When the simulation is running without the degree of compatibility, step 2c is skipped. If agents don't use recommended values, step 2b and 2d are also skipped.

7 Evaluation

In this section we evaluate the benefit of recommendations enhanced by a degree of compatibility. We used our simulator with 20 sellers, 50 buyers and we simulated 1 000, 2 000, 10 000 and 20 000 transactions in the following cases:

- A) buyers don't use recommendations at all;
- B) buyers use recommendations without degree of compatibility ;
- C) buyers filter recommendations using a degree of compatibility: this degree is generated by directly comparing the two functions S as described in the previous section, strategy 2. The threshold of compatibility is set to 0.5, meaning that an agent discards all the values transmitted from an agent with degree of compatibility lower than 0.5.

Our first evaluation covers only the basic strategy 2 described in section 5, where all the agents have a function of satisfaction represented by a linear combination of fixed quality factors. The definition and evaluation of other's strategies are an interesting future development of this work. We assume that agents are not malicious: they always transmit their real values. The study of the robustness of our solutions is regarded as an interesting future works. Here our goal is to show that:

- When agents are different, the quality of recommendation is deteriorated, and the case A and B results became very close. It may happen that the case A performs better than B (i.e., an agent decides better on its own)
- By using the compatibility value, the quality of recommendations is better than case B and "even if not always predictable" results are better than case B.

The metrics used are:

- P_{ok}. (True Positive): the number of transactions completed by the buyer whose level of satisfaction was more than the buyer's threshold, i.e., the number of time it was a good idea to trust the seller;
- P_{no} (True Negative): the number of transactions correctly rejected by the buyer, i.e., the transactions whose level of satisfaction would have been smaller than buyer's expectations;
- N_{ok} (False Positive): the number of transactions accepted by the buyer whose level of satisfaction was smaller than buyer's threshold, i.e., it would have been better to decline the transaction;
- N_{no} (False Negative): the number of transactions erroneously rejected by the buyer, i.e., the transactions whose level of satisfaction would have been greater than buyer's expectations.

The metrics are computed locally and globally and their metrics represent the ability of the buyer of making good predictions in good and bad cases. The Case A is better than the case A if P_{ok} and N_{ok} are greater than P_{no} and N_{no}. We decided to assign (for good or bad) more importance to P_{ok} and N_{ok} than P_{no} and N_{no} , since the first two metrics represent a real benefit or damage, while the second ones are a potential benefit or damage. Therefore we defined a summarizing metric as in the following:

$$F = \frac{P_{ok}}{N_{ok}} \sqrt{\frac{P_{no}}{N_{no}}} \tag{5}$$

In the discussion we will also compare the value of P and N metrics in each case. Before describing the results obtained, we perform an analytical evaluation of the expected results. The case A, where agents do not use recommendations, is expected to perform very good when the number of transactions is very high, since each agent has sufficient number of past interactions to predict correctly sellers' behaviour. For small number of transactions, agents have not enough information on sellers and they follow they dispositional trust that, in our settings, encourages interactions. Therefore, we expect deteriorated values since the first interactions have a strong blind component. Recommendations should work better than the case A even for a small number of transactions, since agents share their information. In the long term, recommendations will perform like the case A. The above prediction is valid only if recommendations are meaningful for the agent. In case of recommendations from agents with strong differences, their effect will be reduced both in the short and in the long term. Therefore, we expect that the case C results, obtained with our compatibility degree, will perform better than the case B independently from the number of transactions. In the short term, the case C should perform better than the

case A, for the effect of meaningful recommendations, while in the long term the case C should tend to the case A's results, since agents receive recommendations only from other agents very compatible with them. The case B, even in the long run should be the worst case, since the effect of recommendations from different agents causes deterioration in the value of recommended values. Results are summarized in figure 3 and table 5 confirming many of our predictions. In particular:

1. 1 000 transactions. The case A performs poorly as expected, but the other two cases are worst. This means that in the case B and C recommendations are not yet effective. In the case C the degree of compatibility is not fully applicable yet, since it is hard to find compatible agents that interacted with the same seller.
2. 5 000 transactions. The case B and C close the gap with the case A. The case C is the best case showing the effect of recommendation based on the degree of similarity. The case B is still the worst case, but the gap with the other cases is the lowest.
3. 10 000 transactions. The case A is now the more effective, meaning that agents have gained enough direct past experience to predict sellers' trustworthiness correctly. The case C performs well, with metrics similar to the case A. The case B shows a growing inefficiency: recommendations without a compatibility degree are deteriorating the predictions
4. 20 000 or more transactions. After 20 000 transactions, the three cases reach an almost steady state (similar results were obtained with 100 000 transactions). The case A is the best case as predicted, the case C performs well, slightly less than A and the case B is very far from the other two. Looking at Table 5's results for 20 000 transactions, the case C, based on the degree of compatibility, appears the most precise in selecting a trustworthy patner, since the value of P_{ok} is even greater than the case A. Note that the case B has a high number of P_{ok}, but this is due to the fact that the system allows more transactions than in the other cases. It has the highest number of P_{ok} (7054 against 6638), but it has also a four time greater number of mistakes N_{ok} (2397 against 890 of the case A).

Transactions	1 000				5 000			
	P_{ok}	N_{ok}	P_{no}	N_{no}	P_{ok}	N_{ok}	P_{no}	N_{no}
Case A	364	223	400	13	1600	713	2559	128
Case B	297	208	424	71	1542	689	2583	186
Case C	348	281	351	20	1532	451	2821	196
Transactions	10 000				20 000			
	P_{ok}	N_{ok}	P_{no}	N_{no}	P_{ok}	N_{ok}	P_{no}	N_{no}
Case A	3565	700	5501	234	6563	890	11797	850
Case B	3678	1444	4757	121	7054	2397	10190	359
Case C	3524	667	5534	275	6638	1094	11518	750

Table 5 Metrics' Value in the four cases analyzed

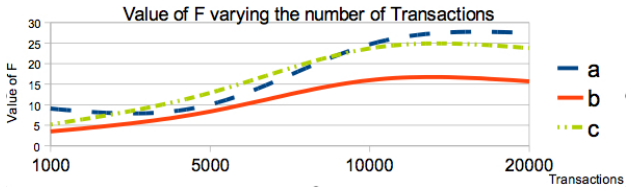

Fig. 3 Evaluation results

8 Open issues and future works

In this paper we analyze the problem of trust transferability. We defined the problem and we analysed the state-of-the-art of solutions. We showed how a degree of compatibility based on sharing common situations keeps the quality of recommendations even in presence of strong difference among agents. Future works are in the investigation of different solutions and their evaluation in the presence of malicious agents. In particular, our simulations show how, when the number of recommnedations are high, our strategy shows results comparable with a strategy purely based on past direct experience, while for an interval of interactions, our method works better than the others tested. We think that a study of trust ontology-matching will represent an important contribution and a complementary solution to our work. We think that an efficeint ontology matching could largely benefit from the support of unsupervised techniques to assist the matching process an viceversa, an macthing at the level of ontology can support a better computation of similarity.

References

1. A. Josang, S. Pope. *Semantic Constraints for Trust Transitivity.* Proceedings of the Second Asia-Pacific Conference on Conceptual Modelling, Newcastle, Australia, 2005
2. Abdul-Rahman 1997. *A distributed trust model.* In New Security Paradigms Workshop, UK.
3. I. Pinyol, M. Paolucci, J. Sabater-Mir *How to Talk About Reputation Using a Common Ontology: From Definition to Implementation.* 9th Workshop on Trust in Agent Societies, Hawaii.
4. A. Elmessery. *Expectations enhanced Trust value.* AAMAS 2006, Hawaii, USA.
5. D Gambetta. *Can we trust trust?* In Diego Gambetta, editor, Trust. Basil Blackwell, 1988.
6. Jm Seigneur, C. Jensen. *Trading Trust for Privacy,* Proceedings of iTrust'04 the International Conference on Trust Management, LNCS 2995, Springer-Verlag, 2004.
7. B. Christianson, William S. Harbison. *Why isn't trust transitive?* In Proceedings of the Security Protocols International Workshop, University of Cambridge, 1996.
8. N. Luhmann. *Familiarity, confidence, trust: Problems and alternatives*. Blackwell, 1988.
9. eRep. *eRep:Social Knowledge for e-Governance.* http://megatron.iiia.csic.es/eRep, 2006.
10. R. Conte, M. Paolucci. *Reputation in artificial societies: Social beliefs for social order*. Kluwer Academic Publishers, 2002.
11. Cahill, V. et al., 2003. *Using Trust for Secure Collaboration in Uncertain Environments*. IEEE Pervasive Computing Magazine, July-September 2003.
12. J. Sabater, C. Sierra. *Reputation and social network analysis in multi-agent systems.* In Proceedings of the 1st AAMAS conference, pages 475-482, New York, NY, USA, 2002. ACM.