SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks **00**: 1–16 (0000)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.0000

Semiring-based Frameworks for Trust Propagation in Small-World Networks and Coalition Formation Criteria

S. Bistarelli[1], S.N. Foley[2], B. O'Sullivan[3], F. Santini[4]*

[1]*Dipartimento di Matematica e Informatica, Università di Perugia and Istituto di Informatica e Telematica (CNR) and Dipartimento di Scienze, Università "G. d'Annunzio" Chieti Pescara, Italy*
[2]*Department of Computer Science University College Cork, Cork, Ireland*
[3]*Department of Computer Science University College Cork and Cork Constraint Computation Centre, Cork, Ireland*
[4]*Dipartimento di Scienze, Università "G. d'Annunzio" Pescara and Istituto di Informatica e Telematica (CNR) and IMT Scuola di Studi Avanzati Lucca, Italy*

Summary

Multitrust provides a flexible approach to encoding trust metrics whereby definitions for trust propagation and aggregation are specified in terms of a semiring. Determining the degree of trust between principals across a trust network is, in turn, programmed as a (semiring based) soft-constraint satisfaction problem. In this paper we consider the use of semiring-based metrics in reasoning about trust between coalition-forming principals. The configurable nature of multitrust makes it well-suited to modeling trust within coalitions: whether adding more principals to a coalition increases trust or decreases trust is captured by the definition of trust aggregation within the semiring. Copyright © 0000 John Wiley & Sons, Ltd.

KEY WORDS: Soft Constraint Logic Programming, And-or Graphs, Trust Propagation, Trust Network

1. Introduction

Dynamic coalitions [1] range from simple spaces used by participants to exchange information, to complex processes in which services operate and are governed according to local and/or global regulation and contract. These coalitions may spawn further coalitions and coalitions may come-together and/or merge. Coalitions may span multiple administrative domains and without a centralized and/or globally trusted third party available to supervise relationships. These trust relationships between coalitions and their members are typically characterized in terms of trust networks (TN) or the web of trust [2]. We are interested in computational models of these trust relationships.

The paper mainly collects different but strongly linked methods to compute trust evaluation-scores towards coalitions of entities/individuals. We propose a computational framework that can be used with already existing trust metrics, e.g. $\langle trust, confidence \rangle$ scores as proposed in [3]. This paper provides four contributions. Firstly, we propose the concept of *multitrust* [4], which correlates trust between a trustor and multiple trustees. The intuition for the term comes from the *multicast* delivery scheme in networks. Multitrust can be used to model and evaluate a coalition of multiple trustees from the

*Correspondence to: Istituto di Informatica e Telematica, Via Moruzzi 1, IT-56124 Pisa, Italy. E-mail: francesco.santini@iit.cnr.it

subjective point of view of the trustor. For example, when downloading a file from multiple sources in a peer-to-peer network, multitrust provides a indicator of trust for the entire download process.

The second contribution of the paper is a model of trust propagation in TNs. We represent TNs using weighted *and-or* graphs [5] (i.e., hypergraphs), mapping individuals to nodes and their relationships to directed connectors. The *and* connectors (i.e., hyperarcs) represent the event of simultaneously trusting a group of individuals. The degree of trust between individuals is modeled in terms of the costs of the connectors. In this paper, the costs of the connectors symbolize how trustworthy the source estimates the destination nodes, that is a *trust value*, and how accurate is this trust opinion, i.e. a *confidence value*.

Soft Constraint Logic Programming (SCLP) [6, 7] provides a convenient declarative programming environment in which to solve the trust propagation problem for multitrust. In SCLP programs, classical logic programming is used in conjunction with *soft constraints* [8, 6]. In particular, we show how to translate the *and-or* graph obtained in the first step into a SCLP program, and how the semantics of such a program computes the best trust propagation tree in the corresponding weighted *and-or* graph. SCLP is based on the general structure of a *c-semiring* [6] (or simply, semiring) with two operations $\times$ and +. The $\times$ is used to combine the preferences, while the partial order resulting from the $+$ operator (see Section 2) is used to compare them.

We take advantage of the semiring structure in order to model and compose different trust metrics. SCLP is parametric with respect to the chosen semiring: the same program deals with different metrics by only choosing the proper semiring structure. A similar model has been proposed for routing [9]. *CIAO Prolog* [10] has been used to implement the SCLP-based trust model described in this paper.

As already stated in literature [11, 12, 13], trust networks exhibit small-world characteristics and the third contribution of this paper is an evaluation of a proposed coalition model as a random small-world network generated using the *Java Universal Network/Graph Framework* (JUNG) [14]. The small-world phenomenon describes the tendency for each entity in a large system to be separated from any other entity by only a few hops. Moreover, these networks show a high clustering coefficient, which quantifies how close a vertex and its neighbors are from being a complete graph. The small average distance in small-world networks [15] allows to cut the solution search after a small threshold, thus improving the search even in wide networks. The path usually consists in few hops of the network.

The fourth and last result provided in the paper uses soft constraints [8, 6] to partition entities into coalitions according to different trust criteria. A coalition can be defined as a temporary alliance among agents, during which they cooperate in joint action for a common task [16]; we use trust scores in order to evaluate the relationships among these entities. Thus, in this case we are no longer finding a path, but sets of elements: trust is no longer computed according to the point of view of a single entity, i.e. the source node of the path, but it is a global score. The obtained set of coalitions is the most trustworthy with respect to the trust metrics used (represented in terms of semirings) [17, 18, 19]. In addition to this optimization, we introduce a condition of stability between an entity and a coalition, which can be used as a more "local" need of an entity and can be adopted to represent further complex interactions.

Self-interested, autonomous software agents on the Internet may negotiate rationally to gain and share benefits in stable (temporary) coalitions [1]. We chose to not consider a coalition of elements as predefined entity, but we decided to work on a graph of elements and to (possibly) aggregate them in order to create coalitions and to find trustworthy groups of individuals that can accomplish a given complex task. The reason is that we consider flexible environments, where new users can register or cancel themselves in a dynamic way [1] (e.g. forums and social networks [20]): therefore, nodes can appear and disappear and thus we do not want to treat them as a fixed "consortium". In general, we consider trust networks as dynamic information, and to find a new solution for the modified graph we only need to update the model and compute again the solution with our semiring-based frameworks (presented in the paper).

Due to its nature dynamic coalition formation methods promise to be particularly well suited for applications of ubiquitous and mobile computing, including mobile commerce (i.e. *M-commerce*). M-commerce as it may be supported by personalized, rational information agents residing, for example, on WAP-enabled access devices such as pagers, organizers, (sub)notebooks, or UMTS cell phones, currently still remains to be an appealing vision for the common Internet user [1].

DOI: 10.1002/sec

This paper presents formal and sound results that extend the preliminary ones in [17, 18, 21]: here we collect and link together information on how coalitions can be represented with soft constraints and we provide an uniform view of multitrust towards and inside them. The paper is is organized as follows. In Section 2 we present some background information on trust metrics, the small-world phenomenon in social networks and the SCLP and soft constraints frameworks. Section 3 describes the related work and, Section 4 introduces multitrust and its interpretation in coalitions. Section 5 shows how to represent a TN with an *and-or* graph, while in Section 6 we describe how to translate *and-or* graphs to SCLP programs and demonstrate that the program computes the best trust propagation for the corresponding *and-or* graph. In Section 7 we evaluate the performance of multitrust over a trust network with small-world properties, since social networks are proved to show this feature [11, 12, 13]. Section 8 considers coalition forming when constrained not just by the trust-metrics but also by other rules that govern coalition membership. Finally, Section 9 draws the final conclusions and outlines intentions on future works.

2. Background

Trust Metrics and Small World Networks. A range of definitions for trust and reputation exist in the literature [20]. In this paper, we adopt the following definitions. Trust describes one node's belief in another node's capabilities with respect to honesty and reliability based on its own direct experiences. Reputation is based on recommendations received also from other nodes. While closely related, a primary difference between trust and reputation is that trust is a score that reflects the relying party's subjective view of an entity's trustworthiness, whereas reputation systems produce the entity's (public) reputation score as seen by the whole community.

Trust and reputation ranking metrics have been used for public key certification, rating and reputation schemes for online communities, peer-to-peer networks, semantic web and mobile computing [20, 22, 2]. These different applications use their own distinct trust metrics. Trust metrics are used to infer trust scores of users by exploiting transitive relationships: if two nodes, for instance node n_1 and node n_4 in Figure 1, are not directly connected then any indirect transitive connection in the TN, for instance, $n_1 \rightarrow n_2 \rightarrow n_4$, can be used to generate an inferred trust rating. A TN represents all the direct trust relationship

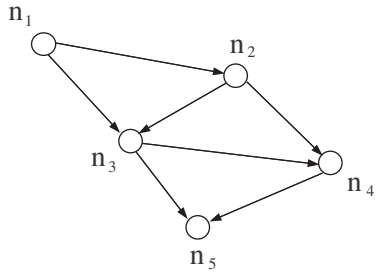


Fig. 1. A classical trust network.

in a community. An example of a classical TN is provided in Figure 1, where we can see that trust is usually interpreted as a 1-to-1 relationship between two individuals: the edges are directed from the trustor to the trustee.

If node n_1 trusts node n_2, and node n_2 trusts node n_3, then n_1 can use the path to compose the inferred rating for n_3, assuming that trust is transitive in this case. This process is called *trust propagation* by concatenation, and it is a desirable requirement since in most settings a user has a direct opinion only about a very small portion of nodes in the TN. Therefore, trust needs to be granted also by basing on third-party recommendations: if n_1 trusts n_2, she/he can use the recommendation about n_3 provided by n_2 [20]. How to compose this information depends on the trust metrics of the links, i.e. it specifically depends on the problem [20] (e.g. by multiplying together the trust scores of the links n_1-n_2 and n_2-n_3).

A trust network, where nodes represent individuals and edges represent their relationships, exhibits the small-world phenomenon if any two individuals in the network are likely to be connected through a short sequence of intermediate acquaintances. In [15] the authors observe that such graphs have a high clustering coefficient (like regular graphs) and short paths between the nodes (like random graphs). According to this definition, small-world networks have sub-networks that are characterized by the presence of connections between almost any two nodes within them. Many empirical graphs are suitably modeled by small-world networks, for example, social networks, the connectivity of the Internet, and gene networks all exhibit small-world network characteristics. Moreover, the advantages to small-world networking for social movement groups are their resistance to change due to the filtering apparatus of using highly connected nodes, and its

DOI: 10.1002/sec

better effectiveness in relaying information while keeping the number of links required to connect a network to a minimum [13].

These networks are divided in sub-communities (i.e., clusters) where few individuals, called the *pivots* [23], represent the bridges towards different groups. These connections are termed *weak ties* in the sociology literature [23], as opposed to *strong ties* that connect a vertex to others in its own sub-community. Weak ties are important because the individuals inside other communities will bring in greater value due to different knowledge and perspectives, while people in the same group would generally tend to have the same knowledge. An example of a small-world network is shown in Section 7.

Social networks have, in general, small-world characteristics [11, 12, 13], and give its relationship to trust, also trust networks exhibit small world characteristics [11, 12]. This has been demonstrated for *Pretty Good Privacy* (PGP) certificate networks [24] as a consequence of the self-organization of users, and in [11] which explores trust-based schemes from the perspective of small worlds. For this reason, in Section 7 we test our computational framework with small-world networks.

C-semirings and Soft Constraints. A c-semiring [8, 6, 25] S (or simply semiring in the following) is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where A is a set with two special elements ($\mathbf{0}, \mathbf{1} \in A$) and with two binary operations $+$ and $\times$ that satisfy certain properties: $+$ is defined over the elements in A and is commutative, associative, idempotent, it is closed and $\mathbf{0}$ is its unit element and $\mathbf{1}$ is its absorbing element; $\times$ is closed, associative, commutative, distributes over $+$, $\mathbf{1}$ is its unit element, and $\mathbf{0}$ is its absorbing element (for the exhaustive definition, please refer to [8]). The $+$ operation defines a partial order $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$ [8, 6, 25]; we say that $a \leq_S b$ if b represents a value *better* than a. A short survey on how this view of valuation structures (and the presented partial order definition) can be linked to different ones is presented in [26]. Other properties related to the two operations are that $+$ and $\times$ are monotone on $\leq_S$, $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum, $\langle A, \leq_S\rangle$ is a complete lattice and $+$ is its lub. Finally, if $\times$ is idempotent, then $+$ distributes over $\times$ and $\langle A, \leq_S\rangle$ is a complete distributive lattice.

A *soft constraint* [8, 6] may be seen as a constraint where each instantiation of its variables has an associated preference. Given $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ and a set of variables V over a finite domain D, a soft constraint is a function which, given an assignment $\eta : V \to D$ of the variables, returns a value of the semiring. Using this notation $\mathcal{C} = \eta \to A$ is the set of all possible constraints that can be built starting from S, D and V.

Any function in $\mathcal{C}$ involves all the variables in V, but we require that it depends on the assignment of only a finite subset of them. Thus, for instance, a binary constraint $c_{x,y}$ over variables x and y, is a function $c_{x,y} : V \to D \to A$, but it depends only on the assignment of variables $\{x, y\} \subseteq V$ (the *support* of the constraint, or *scope*). Note that $c\eta[v := d_1]$ means $c\eta'$ where η' is η modified with the assignment $v := d_1$. Note also that $c\eta$ is the application of a constraint function $c : V \to D \to A$ to a function $\eta : V \to D$; what we obtain, is a semiring value $c\eta = a$. $\bar{\mathbf{0}}$ and $\bar{\mathbf{1}}$ respectively represent the constraint functions associating $\mathbf{0}$ and $\mathbf{1}$ to all assignments of domain values; in general, the $\bar{a}$ function returns the semiring value a.

Given the set $\mathcal{C}$, the combination function $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times c_2\eta$ (see also [8, 6]). Informally, performing the $\otimes$ or between two constraints means building a new constraint whose support involves all the variables of the original ones, and which associates with each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate sub-tuples. The partial order $\leq_S$ over $\mathcal{C}$ can be easily extended among constraints by defining $c_1 \sqsubseteq c_2 \iff c_1\eta \leq c_2\eta$. Consider the set $\mathcal{C}$ and the partial order $\sqsubseteq$. Then an entailment relation $\vdash \subseteq \wp(\mathcal{C}) \times \mathcal{C}$ is defined s.t. for each $C \in \wp(\mathcal{C})$ and $c \in \mathcal{C}$, we have $C \vdash c \iff \bigotimes C \sqsubseteq c$ (see also [6]).

Given a constraint $c \in \mathcal{C}$ and a variable $v \in V$, the *projection* [8, 6] of c over $V - \{v\}$, written $c \Downarrow_{(V\setminus\{v\})}$ is the constraint c' s.t. $c'\eta = \sum_{d\in D} c\eta[v := d]$. Informally, projecting means eliminating some variables from the support.

A SCSP [6] defined as $P = \langle C, con\rangle$ (C is the set of constraints and $con \subseteq V$, i.e. a subset the problem variables). The *best level of consistency* notion defined as $blevel(P) = Sol(P) \Downarrow_\emptyset$, where $Sol(P) = (\bigotimes C) \Downarrow_{con}$ [6]. A problem P is α-consistent if $blevel(P) = \alpha$ [6]; P is instead simply "consistent" iff there exists $\alpha >_S \mathbf{0}$ such that P is α-consistent [6]. P is inconsistent if it is not consistent.

Soft Constraint Logic Programming. The SCLP framework [6, 7, 27], is based on the notion of *c-semiring* introduced in [28, 8]. *Constraint Logic*

DOI: 10.1002/sec

```
s(X)   :- p(X,Y).
p(a,b) :- q(a).
p(a,c) :- r(a).
q(a)   :- t(a).
t(a)   :- 2.
r(a)   :- 3.
```

Table I. A simple example of an SCLP program.

Programming (CLP) [29] extends Logic Programming by replacing term equalities with constraints and unification with constraint solving. In classical Prolog, the program logic is expressed in terms of relations, and execution is triggered by running queries over these relations. Relations and queries are constructed using Prolog's single data type, the term. Relations are defined by clauses. Given a query, the Prolog engine attempts to find a resolution refutation of the negated query.

The SCLP framework extends the classical CLP formalism in order to be able to handle also SCSP [28, 8] problems. In passing from CLP to SCLP languages, we replace classical constraints with the more general SCSP constraints where we are able to assign a *level of preference* to each instantiated constraint (i.e. a ground atom). To do this, we also modify the notions of interpretation, model, model intersection, and others, since we have to take into account the semiring operations and not the usual CLP operations. The fact that we have to combine several refutation paths (a refutation is a finite derivation and the corresponding semiring value a [6]) when we have a partial order among the elements of the semiring (instead of a total one), can be fruitfully used in the context of this paper when we have an graph/hypergraph problems with incomparable costs associated to the edges/connectors. In fact, in the case of a partial order, the solution of the problem of finding the best path/tree should consist of all those paths/trees whose cost is not "dominated" by others.

A simple example of a SCLP program over the semiring $\langle \{N \cup \infty\}, min, +, +\infty, 0 \rangle$, where N is the set of non-negative integers and $D = \{a, b, c\}$, is represented in Tab. I. The intuitive meaning of a semiring value like 3 associated to the atom $r(a)$ (in Tab. I) is that $r(a)$ costs 3 units. Thus the set N contains all possible costs, and the choice of the two operations min and $+$ implies that we intend to minimize the sum of the costs. This gives us the possibility to select the atom instantiation which gives the minimum cost overall. Given a goal like $s(x)$ to this program, the operational semantics collects both a substitution for x (in this case, $x = a$) and also a semiring value (in this case, 2) which represents the minimum cost among the costs for all derivations for $s(x)$. To find one of these solutions, it starts from the goal and uses the clauses as usual in logic programming, except that at each step two items are accumulated and combined with the current state: a substitution and a semiring value (both provided by the used clause). The combination of these two items with what is contained in the current goal is done via the usual combination of substitutions (for the substitution part) and via the multiplicative operation of the semiring (for the semiring value part), which in this example is the arithmetic $+$. Thus, in the example of goal $s(X)$, we get two possible solutions, both with substitution $X = a$ but with two different semiring values: 2 and 3. Then, the combination of such two solutions via the min operation give us the semiring value 2.

3. Related Work

PGP allows the user to assign four levels of trustworthiness to a public-key. These levels correspond to how much the user thinks the owner of that public-key can be trusted to be an "introducer" to another trustworthy public-key certificate. To compensate for the ambiguity of the proposed trust levels, PGP allows its users to tune PGP's "skepticism". This is done by adjusting two parameters, $COMPLETES\_NEEDED$ and $MARGINALS\_NEEDED$. The former defines the number of completely trusted signatures required to make a certificate completely valid, and the latter defines the number of marginally trusted signatures to achieve the same outcome [30]. The formal framework presented in Section 5 can be used also to aggregate all these values together toward a given public-key in the graph, e.g. belonging to Alice, in order to be sure of the Alice/public-key association. In words, the best tree, which must be seen in the inverse order (i.e. the root represents the public-key we would like to trust), could maximize for example the number of marginally trusted signatures.

Our approach is different from the PGP marginal trust approach because, *i)* the idea of multitrust explicitly considers trees and trust toward a group of users (trees are not considered in PGP), and *ii)* we use semiring operators to combine the trust scores along the path/tree, while in PGP no idea of different metrics is considered. The multitrust approach is also more flexible than the recent extension of GNU PGP which

DOI: 10.1002/sec

allows trust values to be associated with certificates whereby trust-propagation corresponds to a two-terminal network reliability problem in a probabilistic (trust) graph.

Other approaches to trust propose the aggregation of multiple trust paths in order to have feedback from different individuals and to enforce the final evaluation [22, 2]. The main reason is to improve the attack-resistance of the metrics when an attacker wishes to introduce a false name-key binding. *Global* trust metric values [2] are assigned to an individual based upon complete trust graph information; many of these metrics borrow their ideas from the PageRank algorithm [2]. Essentially, in these metrics trust begins at the source node and "flows" to all the other nodes in the network. We can instead classify a trust metric as *local* if the values it computes are based on local estimates of trust in the graph. For example, the Rahman-Hailes metric computes the trust value of a path as a product of the trust values of edges on the path, and takes the average of all path values [31].

However, even if it is possible to compute this aggregation of paths within our framework, our goal is quite different and, has not to our knowledge been considered in literature. A multitrust connector represents a collaboration among the n trustees and, in some sense, a coalition of the reached entities. It represents the act of "simultaneously" trusting a collection of individuals, which behave in a coordinated way. The trust values associated with this hyperarc represents how well the collaboration works. Trust paths need not be independent of on another: the trust level of each and-connector is not given by summing up the levels of the edges, but using a $\odot$ operator that explicitly calculates a weight for the connector.

A related and parallel study on the combination of trust values has been proposed in [32, 33]. In these works, we present a variant of the Datalog language (we call it Datalog^W) to deal with weights on ground facts and to consequently compute a feedback result for the goal satisfaction. The weights are chosen from a proper c-semiring. In this context, our goal is to use this language as a semantic foundation for languages expressing trust relationships, as the *Role-based Trust-management Markup Language* (RTML). The final trust score is obtained by aggregating the trust scores associated with the basic role definition. For example, $StateU.highMarks \longrightarrow \langle Alice, 0.8\rangle$ certifies that *Alice* has obtained a good number of high marks (since the value is 0.8) for the exams completed at the *StateU* university (the credential is issued by *StateU*). These papers consider the concept from the point of view of trust management languages and credentials, and, unlike this paper, do not consider trust propagation and coalition formation.

4. Multitrust and Coalitions

We introduce the concept of multitrust [4], which extends the usual trust relationship from pairs of individuals to one trustor and multiple trustees in a "correlated" way (e.g. time-correlated):

Definition 1 *Given a set of entities E in the considered trust domain, multitrust is defined as a relationship R_{mt} between a trustor $t \in E$ and a set of trustees $T \subset E$, where $t \notin T$ and $|T| \geq 1$. R_{mt} can be described in terms of time (e.g. at the same time), modalities (e.g. with the same behavior) or collaboration among the trustees in T w.r.t. t.*

For example if we consider time, the trustor could simultaneously trust multiple trustees, or, considering instead a modality example, the trustor could contact the trustees with the same communication device, e.g. by phone. Consequently, this trust relation R_{mt} is 1-to-n, unlike the 1-to-1 relationship in conventional trust systems [2]. One interpretation of multitrust is *team effectiveness* [34]. For example, suppose we have a decentralized community of open-source programmers and we want to know if a subset of them can be reliably assigned to a new project.

Multitrust can be used to represent trust within a coalition of entities. For example, by cooperating and sharing their respective expertise, a team of 3 programmers, could significantly enhance the quality of a software product than working as individuals on the project.Cooperating groups, have been thoroughly investigated in Artificial Intelligence and Game Theory and has proved to be a useful strategy in both real-world economic scenarios and multi-agent systems [16]. For instance, coalition formation can be seen as a co-operative game [16] determined by a set A of agents and a real-valued characteristic function assigning each coalition its maximum gain (the so-called coalition value, e.g. money). A solution is represented by a partition of the agents and an efficient payoff distribution: the payoff distribution assigns each agent its utility out of the value of the coalition it is member of in a given coalition structure. Stability conditions consider individually rational distributions, which are assigning each agent at least the gain it may get without collaborating within any coalition.

DOI: 10.1002/sec

Also AI can be used to partition autonomous agents by considering trust relationships [35, 36, 37].

Coalitions are generally considered to be task-directed and short-lived, but last longer than team organization [16]. However, in some cases, coalitions may have a long lifetime created [35]. For the purposes of this paper, given a population of entities E, then the problem of coalition formation consists of selecting the appropriate partition of E, $P = \{C_1, \ldots, C_n\}$ ($|P| = |E|$ if each entity forms a coalition on its own), s.t. $\forall C_i \in P$, $C_i \subseteq E$ and $C_i \cap C_j = \emptyset$, if $i \neq j$. P maximizes the utility (utility against costs) that each coalition can achieve in the environment. Therefore, agents group together because an utility can be gained by working in groups, but this growth is somewhat limited by the costs associated with forming and maintaining such a structure.

Notice that the multitrust problem is intrinsically different from the aggregation of disjoint trust paths examined in other works [38, 22]. Our goal is different, since with a 1-to-n relationship we model a collaboration (in a coalition sense) among the n trustees, and not a mathematical aggregation (e.g. the average) of the trust values on the disjoint paths rooted in them. This characterizes the difference between multitrust and the conventional definition of trust in literature.

5. From Trust Networks to *and-or* Graph

An *and-or* graph [5] is defined as a special type of hypergraph. In particular, instead of arcs connecting pairs of nodes there are hyperarcs connecting an n-tuple of nodes ($n = 1, 2, 3, \ldots$). The arcs are called *connectors* and they must be considered as directed from their first node to all the other nodes in the n-tuple. Formally an *and-or* graph is a pair $G = (N, C)$, where N is a set of *nodes* and C is a set of connectors defined as $C \subseteq N \times \bigcup_{i=0}^{k} N^i$.

When $k > 1$ we have an *and* connector since it reaches multiple destinations at the same time; all the different connectors rooted in the same n_i node can be singly chosen, i.e. *or* connectors. Note that the definition allows 0-connectors, i.e. connectors with one input and no output node. In the following explanation we will also use the concept of *and* tree [5]: given an *and-or* graph G, an *and* tree H is a *solution tree of G with start node n_r*, if there is a function g mapping nodes of H into nodes of G such that: *i)* the root of H is mapped in n_r, and *ii)* if $(n_{i_0}, n_{i_1}, \ldots, n_{i_k})$ is a connector of H, then $(g(n_{i_0}), g(n_{i_1}), \ldots, \ldots, g(n_{i_k}))$ is a connector of G.

Informally, a solution tree of an *and-or* graph is analogous to a path of an ordinary graph: it can be obtained by selecting exactly one outgoing connector for each node. If all the chosen connectors are 1-connectors, then we obtain a plain path and not a tree.

In Figure 2 we represent a TN for multitrust as a weighted *and-or* graph: this graph can be built starting from the TN in Figure 1. Each of the individuals can be easily cast in a corresponding node of the *and-or* graph. In Figure 2 we represent our trustor as a black node (i.e. n_1) and the target trustees as two concentric circles (i.e. n_4 and n_5). Nodes n_2 and n_3 can be used to propagate trust.

To model the trust relationship between two nodes we use 1-connectors, which correspond to usual TN arcs: the 1-connectors in Figure 2 are (n_1, n_2), (n_1, n_3), (n_2, n_3), (n_2, n_4), (n_3, n_4), (n_3, n_5), (n_4, n_5). We note that the connectors are directed, and thus, for example the connector (n_4, n_5) means that the input node n_4 trusts the individual represented by n_5. Moreover, since we are now dealing with multitrust, we need to represent the event of trusting more individuals at the same time. To attain this, in Figure 2 we can see the three 2-connectors (n_1, n_2, n_3), (n_2, n_3, n_4) and (n_3, n_4, n_5): for example, the first of these hyperconnectors defines the possibility for n_1 to trust both n_2 and n_3 in a correlated way. In Figure 2 we draw these n-connectors (with $n > 1$) as curved oriented arcs where the set of their output nodes corresponds to the output nodes of the 1-connectors traversed by the curved arc. Considering the ordering of the nodes in the tuple describing the connector, the input node is at the first position and the output nodes (when more than one) follow the orientation of the related arc in the graph (in Figure 2 this orientation is lexicographic). Notice that in the example we decided to use connectors with dimension at most equal to 2 (i.e. 2-connectors) for sake of simplicity. However it is possible to represent whatever cardinality of trust relationship, that is among a trustor and n trustees (i.e. with a n-connector).

We represent the TN using a weighted *and-or* graph. However an algebraic framework is needed in order to model our preferences for the connectors for use during trust propagation. For the purpose of this paper we use (but are not limited to) a semiring structure based on trust and confidence attributes.

Each of the connectors in Figure 2 is labeled with a pair of values $\langle t, c \rangle$: the first component represents a trust value in the range $[0, 1]$, while the second component represents the accuracy of the trust value

DOI: 10.1002/sec

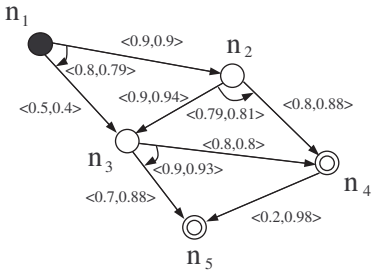


Fig. 2. An *and-or* graph representing multitrust: the weights on the connectors represent trust and confidence values (i.e. $\langle t, c\rangle$).

assignment (i.e. a *confidence* value), and it is still in the range $[0, 1]$. This parameter can be assumed as a quality of the opinion represented instead by the trust value; for example, a high confidence could mean that the trustor has interacted with the target for a long time and then the correlated trust value is estimated with precision. A trust value close to 1 indicates that the output nodes of the connector have gained a good feedback in terms of their past performance and thus are more trustworthy, whereas a low trust value means the nodes showed relatively poor trustworthiness in the past and are rated with low score. In general, we could have trust expressed with a k-dimensional vector representing k different metrics; in this example we have 2-dimensional vectors. This kind of trust/condfidence metric, that is the semiring we use to propagate trust in the network, corresponds to the *path semiring* [3]: $S_{trust} = \langle\langle[0, 1], [0, 1]\rangle, +_p, \times_p, \langle 0, 0\rangle, \langle 1, 1\rangle\rangle$, where

$$\langle t_i, c_i\rangle +_p \langle t_j, c_j\rangle = \begin{cases} \langle t_i, c_i\rangle & \text{if } c_i > c_j, \\ \langle t_j, c_j\rangle & \text{if } c_i < c_j, \\ \langle max(t_i, t_j), c_i\rangle & \text{if } c_i = c_j. \end{cases}$$

$$\langle t_i, c_i\rangle \times_p \langle t_j, c_j\rangle = \langle t_i t_j, c_i c_j\rangle$$

Along the same path, the $\times_p$ computes the scalar product of both trust and confidence values, and since the considered interval is $[0, 1]$, they both decrease when aggregated along a path. When paths are instead compared, $+_p$ chooses the one with the highest confidence. If the two opinions have equal confidences but different trust values, $+_p$ picks the one with the highest trust value[†]. In this way, the precision of the information is more important than the information itself. If the k-dimensional costs of the connectors are not elements of a totally ordered set (therefore, not in our trust/confidence example), it may be possible to obtain several Pareto-optimal solutions.

Different semirings can be used to model other trust metrics: for example, the *Fuzzy Semiring* $\langle[0, 1], \max, \min, 0, 1\rangle$ can be used if we decide that the score of a trust chain corresponds to the weakest of its links. Or we can select the *Weighted Semiring*, i.e. $\langle\mathcal{R}^+, \min, +, \infty, 0\rangle$, to count negative referrals in reputation systems as in e-Bay [20]. Other works proposing trust metrics as semiring structures are [19, 39]. A metric that cannot be represented with semiring is, for example, the *arithmetic mean* of the trust values.

Collecting the trust values to assign to the labels of the connectors is beyond of the scope of this paper, but they can be described in terms of specificity/generality dimensions (if we relay on one or more aspects) and subjective/objective dimensions (respectively personal, as e-Bay, or formal criteria, as credit rating) [20]. In this section, for n-connectors with $n \geq 2$, we suppose objective ratings and, therefore, the use of a composition operation $\circ$ which takes n k-dimensional trust metric vectors (e.g. $tvalue_1, \ldots, tvalue_n$) as operands and returns the estimated trust value for the considered n-connector ($tvalue_{nc}$): $\circ\,(tvalue_1, tvalue_2, \ldots, tvalue_n) \longrightarrow tvalue_{nc}$.

Notice that we can also suppose to have a different $\circ_i$ operator for each entity i of the TN, in order to model different subjective ratings instead of a global objective rating as used in the example of this section. As already noted, the $\circ$ operation is not necessarily an arithmetic "addition" of single trust values, but it must take into account also the "added value" (or "subtracted value") derived from the effect of combination of ratings. For example, the cost $\langle 0.9, 0.93\rangle$ of connector (n_3, n_4, n_5) in Figure 2 significantly benefits from simultaneously trusting n_4 and n_5, since both the trust/confidence values of (n_3, n_4) and (n_3, n_5) are sensibly lower (i.e., respectively $\langle 0.8, 0.8\rangle$ and $\langle 0.7, 0.88\rangle$). The reason could be that n_3 has frequently observed fruitful collaboration between n_4 and n_5 reflecting the high confidence value. On the other hand, n_2 does not consider n_3 and n_4 to be so "collaborative" since the trust label of (n_2, n_3, n_4), i.e. $\langle 0.8, 0.81\rangle$, is worse than the costs of (n_2, n_3) and (n_2, n_4) (i.e. $\langle 0.9, 0.94\rangle$ and $\langle 0.8, 0.88\rangle$). In the example in Figure 2 we use subjective ratings, and therefore the trust values for 2-connectors do not follow any specific $\circ$ function. Some

[†]Notice that in the *path semiring*, the $+$ operator defines a total order on the couples; $\langle 0.3, 0.89\rangle \leq_S \langle 0.3, 0.9\rangle$ and $\langle 0.31, 0.9\rangle \leq_S \langle 0.3, 0.9\rangle$.

DOI: 10.1002/sec

examples of functions for computing an objective rating are instead shown in Section 8.

Notice that trust is sometimes computed by considering all the paths between two individuals and then by applying a function in order to find a single result [22] (e.g. the mean of the trust scores for all the paths). This could be accomplished by using the *expectation* semiring [40], where the $+$ operation of the semiring is used to aggregate the trust values across paths, as proposed in [3]. In this paper, we decide to keep $+$ as a "preference" operator for distinct paths (as proposed for classical SCLP, see Section 2) in order to choose the best one, since in Section 7.1 we suggest how to reduce the complexity of the framework by visiting less paths as possible. Thus, aggregating the trust values of the paths is not so meaningful when trying to reduce the number of visited paths at the same time.

Our intention in this paper is to leave the definition of the $\circ$ function as more general as possible: its aim is to compute the trust for a coalition of trustees and it clearly depends on the criteria chosen to evaluate a coalition and the internal dynamics inside it. For example, if the trustor n_i is aware of the trust score between n_j and n_k (but in general this could not be possible), the cost of the 2-connector (n_i, n_k, n_j) could be computed with $\circ$ by using the costs of (n_i, n_k), (n_i, n_j) but also (n_k, n_j) and (n_j, n_k). The method we instead adopt to compute trust, e.g. using only (n_i, n_k) and (n_i, n_j) as described before, can be autonomously computed by each single node/agent, since it only needs the trust evaluation of a trustor w.r.t. all his trustees; this can be useful in decentralized trust architectures. To provide more possible scenarios, the $\circ$ function can be also expressed with a table having 2^n rows (where n is the number of agents) reporting all the possible coalitions and a column which associates the trust score with the corresponding coalition. This method can be use with few agents and when the environment is closed and well-known (thus not for highly dynamic coalition formation [1]): the advantages are that the calculation can be faster (i.e. reading a valued from the table) or that it may not be a real computation, but an estimation: in this case we do not need to find a mathematical function (it could be a difficult task).

As a further possibility, the $\circ$ function could be implemented by the same $\times$ operator of the semiring used to propagate trust (or by a different semiring) in order to simplify the implementation of the framework. In this case the algebraic properties of the $\circ$, that is $\times$, are reported in Section 2. More complex $\circ$ function can be created, for example not taking values as input but portions of (or the entire) TN. Such functions can check also its topology, that is how much the subgraph representing the coalition is connected: a clique means a lot of already established relationships among its members, which can be an advantage. In literature there are different proposals for computing trust of coalitions, and therefore we prefer to not rigidly define the $\circ$ function.

6. *And-or* Graphs Using SCLP

In this Section, we explain how to represent *and-or* graphs with a program in SCLP. Our approach is motivated by two important features of this programming framework: *i)* SCLP is a declarative programming environment and, thus, is relatively easy to implement different problems; *ii)* the c-semiring structure is can be encoded to represent a variety of different trust metrics. For the purposes of illustration, we translate the *and-or* graph in Figure 2; by simply changing the facts in the program, it is possible to translate any other tree.

Using this framework, we can easily find the best trust propagation over the hypergraph built in Section 5. Our aim is to find the best path/tree simultaneously reaching all the desired trustees, which is only one of the possible choices when computing trust [22]. According to *multipath propagation*, when multiple propagation paths (in this case, trees) exist between A and C (in this case, several trustees at the same time), all their relative trust scores can be composed together in order to have a single result balanced with every opportunity. To attain multipath propagation we need to use the *expectation* semiring [40] as explained in Section 5.

In SCLP a clause such as $c(n_i, [n_j, n_k])$:- $tvalue$, means that the graph has connector from n_i to nodes n_j and n_k with $tvalue$ cost. Other SCLP clauses can describe the structure of the path/tree we desire to search over the graph. Notice that possible cycles in the graph are automatically avoided by SCLP, since the $\times$ of the semiring is a monotonic operation.

We use CIAO Prolog [10] as the system to solve the problem. CIAO Prolog has also a fuzzy extension, but it does not completely conform to the semantics of SCLP defined in [7] (due to interpolation in the interval of the fuzzy set). For this reason, we inserted the cost of the connector in the head of the clauses, unlike the SCLP clauses which have the cost in the body of the clause.

DOI: 10.1002/sec

```
:- module(trust,_,_).
:- use_module(library(lists)).
:- use_module(library(aggregates)).
:- use_module(library(sort)).

times
times([T1, C1], [T2, C2], [T, C]) :-
    T is (T1 * T2),
    C is (C1 * C2).

plus
plus([], MaxSoFar, MaxSoFar).

plus([[T,C]|Rest], [MT,MC], Max):-
    C > MC, plus(Rest, [T,C], Max).

plus([[T,C]|Rest], [MT,MC], Max):-
    C = MC, T > MT,
    plus(Rest, [T,C], Max).

plus([[T,C]|Rest], [MT,MC], Max):-
    C < MC,
    plus(Rest, [MT,MC], Max).

plus([[T,C]|Rest], [MT,MC], Max):-
    C = MC,
    T < MT,
    plus(Rest, [MT,MC], Max).

trust
trust(X, Y, Max):-
    findall([T,C], trustrel(X, Y, [T,C]), L1),
    plus(L1,[0,0],Max).

Leaves
leaf([n1], [1,1]).
leaf([n2], [1,1]).
leaf([n3], [1,1]).
leaf([n4], [1,1]).
leaf([n5], [1,1]).

Connectors
connector(n1,[n2], [0.9,0.9]).
connector(n1,[n3], [0.5,0.4]).
connector(n1,[n2,n3], [0.8,0.79]).
connector(n2,[n3], [0.9,0.94]).
connector(n2,[n4], [0.8,0.88]).
connector(n2,[n3,n4], [0.8,0.82]).
connector(n3,[n4], [0.8,0.8]).
connector(n3,[n5], [0.7,0.88]).
connector(n3,[n4,n5], [0.9,0.93]).
connector(n4,[n5], [0.2,0.98]).

1)
trustrel(X,[X], [T,C]):-
    leaf([X], [T,C]).

2)
trustrel(X, Z, [T,C]):-
    connector(X,W, [T1,C1]),
    trustrelList(W, Z, [T2,C2]),
    times([T1,C1], [T2,C2], [T,C]).

3)
trustrelList([],[], [1,1]).

4)
trustrelList([X|Xs],Z, [T,C]):-
    trustrel(X, Z1, [T1,C1]),
    append(Z1, Z2, Z),
    trustrelList(Xs, Z2, [T2,C2]),
    times([T1,C1], [T2,C2], [T,C]).
```

Table II. The CIAO program representing the *and-or* graph in Figure 2

From the *and-or* graph in Figure 2 we build the corresponding CIAO program of Tab. II as follows. First, we describe the connectors of the graph with facts such as:

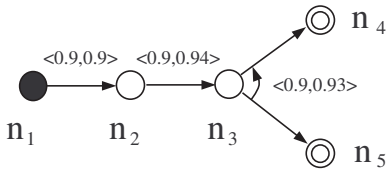
$$connector(trustor, [trustees\_list],$$
$$[trust\_value, condifence\_value])$$

The fact $connector(n_1, [n_2, n_3], [0.8, 0.79])$ represents the connector of the graph (n_1, n_2, n_3) with a trust/confidence value of $\langle 0.8, 0.79\rangle$ (n_i represents the name of the node). The set of connector facts is highlighted as *Connectors* in Tab. II, and represents all the trust relationships of the community. The *Leaves* facts of Tab. II represent the terminations for the Prolog rules. Their cost must not influence the final trust score, and then it is equal to the unit element of the $\times$ operator of the S_{trust} semiring presented in Section 5, i.e. $\langle 1, 1\rangle$. The *times* and *plus* clauses in Tab. II respectively mimic the $\times$ and $+$ operation of $S_{trust} = \langle\langle[0, 1], [0, 1]\rangle, +_p, \times_p, \langle 0, 0\rangle, \langle 1, 1\rangle\rangle$ explained in Section 5. The *trust* clause is used as the query to compute trust in the network: it collects all the results for the given source and destinations, and then finds the best trust/confidence couple by using the *plus* clauses.

Lastly, the rules 1-2-3-4 in Tab. II describe the structure of the relationships to be found over the social network: with these rules it is possible to find both 1-to-1 relationships (i.e. for classical trust propagation) or 1-to-n relationships (i.e. for multitrust propagation, described in Section 2). *Rule 1* represents a relationship made of only one leaf node, *Rule 2* outlines a relationship made of a connector plus a list of sub-relationships with root nodes in the list of the destination nodes of the connector, *Rule 3* is the termination for *Rule 4*, and *Rule 4* is needed to manage the junction of the disjoint sub-relationships with roots in the list $[X|Xs]$. When we compose connectors and tree-shaped relationships (*Rule 2* and *Rule 4*), we use the *times* clause to compose their trust/confidence values together.

To solve the search over the *and-or* graph problem it is enough to perform a query in Prolog language: for example, if we want to compute the cost of the best relationship rooted at n_1 (i.e. n_1 is the starting trustor) and having as leaves the nodes representing the trustees (i.e. n_4 and n_5), we have to perform the query $trust(n_1, [n_4, n_5], [T, C])$, where T and C will be respectively instantiated with the trust and confidence values of the found relationship. The output for this query corresponds to the cost of the tree in Figure 3, i.e. $\langle 0.72, 0.78\rangle$. Otherwise, if we are interested in knowing the best trust relationship between one trustor (e.g. n_1) and only one trustee

DOI: 10.1002/sec



Trust result:	Confidence result:
0.9 x	0.9 x
0.9 x	0.94 x
0.9 =	0.93 =
0.72	0.78

Fig. 3. The best trust relationship that can be found with the query $trust(n_1, [n_4, n_5], [T, C])$ for the program in Tab. II.

(e.g. n_4), as in classical trust propagation, we should perform the query $trust(n_1, [n_4], [T, C])$.

Notice that if the ratings of our trust relationships are objective (see Section 5) then it is possible to program the $\circ$ operator in CIAO Prolog. In this case the n-connectors with $n > 1$ are constructed in the program by applying the $\circ$ operator on the relevant 1-connectors. In the program in Tab. II all the n-connectors are, instead, directly expressed as facts, and not automatically built with clauses. In this sense, an extension to the program in Tab. II is provided in Tab. III: the connectors are created from the edges in a dynamic way and the computation of their cost is obtained by using the $\circ$ function directly in the *Connector* clauses. In this case, $\circ \equiv max$ for both the trust/confidence values. The *sort* predicate is used to reduce the number of the created connectors, i.e. only (n_1, n_2) and not also (n_2, n_1).

7. An Implementation of the Model

To develop and test a practical implementation of our model, we adopt the *Java Universal Network/Graph Framework* [14], a software library for the modeling, analysis, and visualization of data that can be represented as a graph or network. The *WattsBetaSmallWorldGenerator* included in the library is a graph generator that produces a random small world network using the beta-model as proposed in [41]. We use this small-world generator because social and trust networks usually show small-world properties [11, 12, 13]. The basic idea is to start with a one-dimensional ring lattice in which each vertex has k-neighbors and then randomly rewire the edges,

```
use_module(library(sort)).

max
max([X, Y], X) :- X >= Y.
max([X, Y], Y) :- X < Y.

Edges
edge(n1,[n2], [0.9, 0.9]).
edge(n1,[n3], [0.5, 0.4]).
edge(n2,[n3], [0.9, 0.94]).
edge(n2,[n4], [0.8, 0.88]).
edge(n3,[n4], [0.8, 0.8]).
edge(n3,[n5], [0.7, 0.88]).
edge(n4,[n5], [0.2, 0.98]).

Connector
connector(X, [Y], L, [T,C]):-
   edge(X, [Y], [T,C]),
   nocontainsx(L, Y),
   insert_last(L, Y, Z),
   sort(Z,Z).

connector(X, [Y|Ys], L, [T,C]):-
   edge(X, [Y], [T1,C1]),
   nocontainsx(L,Y),
   insert_last(L, Y, Z),
   sort(Z,Z),
   connector(X, Ys, Z, [T2,C2]),
   max([T1,T2], T),
   max([C1,C2], C).
```

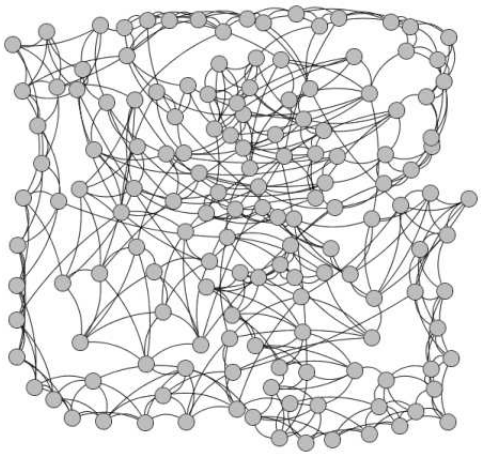
Table III. These clauses show how 1-to-1 relationships (i.e. the edges) can be composed with the $\circ$ operator to form 1-to-n relationships (i.e. the connectors).

with probability β, in such a way that a small-world networks can be created for certain values of β and k that exhibit low characteristic path lengths and high clustering coefficient.

We generated the small-world network in Figure 4 (with undirected edges) and then we automatically produced the corresponding program in CIAO (considering the edges as directed), as in Section 6. The results reported in Figure 4 suggest the small-world nature of our test network: a quite high clustering coefficient and a low average shortest path.

With respect to the program in Tab. II we added the $Trust\_Hops < \lceil 2 \cdot Avg\_Shortest\_Path \rceil$ constraint: in this case, $Trust\_Hops < 9$, which is also the diameter of the network (see Figure 4). This constraint limits the search space (the depth of possible found paths) and provides a good approximation at the same time: in small-world networks, the average distance between two nodes is logarithmic in the number of nodes [15], i.e. every two nodes are close to each other. Therefore, this constraint limits the number of found paths (by using

DOI: 10.1002/sec



Nodes	Edges	Clustering	Avg. SP	Min Deg
150	450	0.44	4.26	4

Max Deg.	Avg. Deg	Diameter
8	6	9

Fig. 4. The test small-world network generated with JUNG [14] and the corresponding statistics.

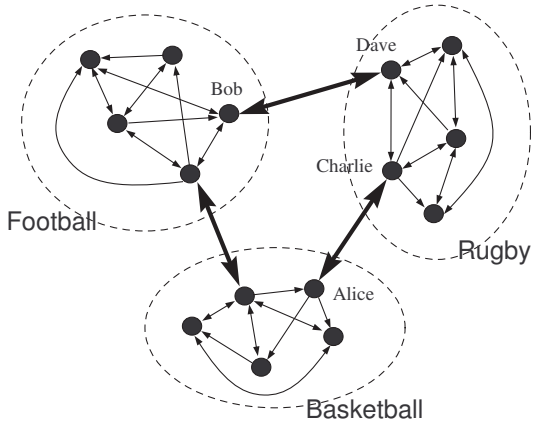


Fig. 5. The small-world of sports.

their depth), but however a large number of alternative routes fall within short distance from the source. Performance clearly benefit from this cut on the search space.

We performed 50 tests on the graph in Figure 4, and in every case the propagation between two nodes was computed within 5 minutes. Clearly, even if the results are promising and the small-world nature allows them to be repeated also on larger graph due to the logarithmic increase of average shortest path statistics, we need some improvements to further relax the $Trust\_Hops$ constraint. These improvements are suggested in Section 7.1.

7.1. Complexity Considerations

The representation of TN given in Section 5 can lead to an exponential time solution because of the degree of the nodes: for each of the individuals, we have a connector towards each of the subsets of individuals in their social neighborhood, whose number is 2^d, where d is the out-degree of the node. The complexity of the tree search can be reduced by using *Tabled Constraint Logic Programming* (TCLP), i.e. with *tabling* (or *memoing*) techniques. Tabling efficiency, w.r.t. not-tabled queries is shown in [42]: the improvement strongly depends on the considered problem instance, for example the time is reduced by a factor 100 in [43]. The calls to *tabled* predicates are stored in a searchable structure together with their proven instances, and subsequent identical calls can use the stored answers without repeating the computation: a partial solution is already present in a table.

The work in [44] explains how to port *Constraint Handling Rules* (CHR) to XSB (acronym of *eXtended Stony Brook*), and in particular its focus is on technical issues related to the integration of CHR with tabled resolution. CHR is a high-level natural formalism to specify constraint solvers and propagation algorithm. At present time, from the XSB system it is possible to load a CHR package and to use its solving functionalities combined with tabling. SCSPs have already been successfully implemented in the CHR system [45].

The procedure of finding such a goal table for each single sub-community is much less time consuming than finding it for a whole not-partitioned social network. For this reason we can take advantage from the highly clustered nature of small-worlds. In Figure 5 it is represented the community of people practising sports; the community is clustered into three sub-groups: Football, Basketball and Rugby. The individuals that represent the bridges among these groups are people practising two different sports, and are called *pivots*; their very important relationships are instead called *weak ties* (as we explained in Section 2), and can be used to widen the knowledge from a sub-group towards the rest of the small-world. If *Alice* (a pivot in the Basketball cluster) wants to retrieve a trust score about *Bob* (a pivot in the Football cluster), she could ask to *Charlie* and *Charlie* to *Dave* (pivots in the Rugby cluster). Therefore, the pivots should store a "tabled vision" of their community to improve the performances for intra-community relationships.

DOI: 10.1002/sec

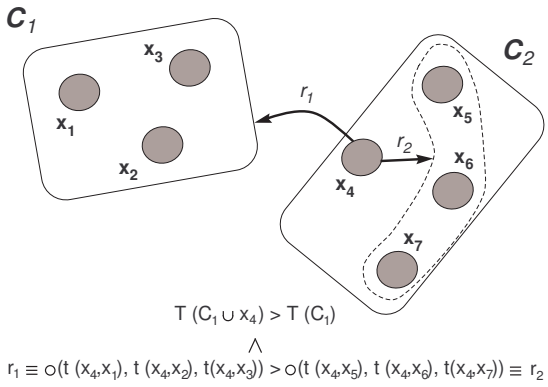


Fig. 6. A graphical intuition of two blocking coalitions.

Therefore, our idea is to solve the problem for each distinct medium-size cluster of elements, and then to compose the results by benefiting from the clustered nature of social networks, in order to solve the problem over wider network.

8. Using Soft Constraints to Represent Different Coalition Formation Criteria

In this Section we consider the partitioning of the set of entities into different coalitions, whereby a single entity can appear in only one coalition at time. With multitrust, even the trust score of a coalition is related to the view of the evaluating trustor entity, while in this Section the score is objective. Notice that in both these frameworks, the trust score of a coalition is obtained with the $\circ$ operator (see Section 5 and Def. 2).

Cooperation involves a degree of risk arising from the uncertainties of interacting with autonomous self-interested agents. Trust [20] describes node's belief in another node's capabilities, honesty and reliability based on its own direct experiences. Trust metrics have been already adopted to perceive this risk, for example, by estimating how likely other agents are to fulfill their cooperative commitments [35, 36]. Since trust is usually associated with a specific scope [20], we suppose that this scope concerns the task that the coalition must face after its formation; for example, in electronic marketplaces the agents in the same coalition agree with a specific discount for each transaction executed [36, 37]. Clearly, an entity can also trust itself in achieving the task, and can form a singleton coalition.

In the individually-oriented approach an agent prefers to be in the same coalition with the agent with whom it has the best relationship [36]. In the socially-oriented approach the agent, instead, prefers the coalition in which it has most summative trust [36]. Alternatively, in this Section we model different optimization criteria based on different semirings (see Section 2). To do so, in Def. 2 we formalize how to compute the trustworthiness of a whole coalition:

Definition 2 *Given a coalition C of agents defined by the set $\{x_1, \ldots, x_n\}$ and a trust function t defined on ordered couples (i.e. $t(x_i, x_j)$ is the trust score that x_i has collected on x_j), the trustworthiness of C (i.e. $T(C)$) is defined as the composition (i.e. $\circ$) of the 1-to-1 trust relationships, i.e. $\forall x_i, x_j \in C. \circ t(x_i, x_j)$ (notice that i can be equal to j, modeling the trust in itself). Therefore, the multitrust (see Section 4) score among all the elements inside a coalition is computed to define its trustworthiness.*

The $\circ$ function has already been defined in Section 5; it models the composition of the 1-to-1 trust relationships. For the sake of simplicity, in this paper we consider objective ratings [20] in order to easily represent and compute trust with a mathematical operator: therefore, we use a $\circ$ function common to every node in the graph, i.e. a single operator. For instance, some practical instantiations of the $\circ$ function can be the *arithmetic mean* or the *max* operator (as in Tab. III): $\forall x_i, x_j \in C. avg\, t(x_i, x_j)$ or $\forall x_i, x_j \in C. \max t(x_i, x_j)$. Notice that, differently from previous sections, we now consider more than one coalition at once as the result of the partition, and thus we need to compute a trust score internal to each coalition. This is accomplished by using the $\circ$ function. With respect to Section 5, this step simply consists in aggregating all the 1-to-1 hyperarcs internal to the same coalition with the same $\circ$ operator (where n is the number of other elements in the coalition). Therefore, it represents the aggregation of the costs of all the individuals in the coalition, that is the multitrust score w.r.t. each entity inside it.

As described in Section 5, changing the semiring structure allows different trust-propagation/trust metrics [17, 19] to be constructed. Organization of principals into optimal coalitions can be based on different principals, such as minimizing a general cost of the aggregation or maximizing "consistency" evaluation of the included entities, i.e. how much their interests are alike. In order to represent more complex interactions among the entities and the coalitions, we propose also a mandatory stability condition that is able to model a local requirement of an entity, w.r.t.

DOI: 10.1002/sec

to the global optimization considering all the formed coalitions. In this paper, this condition is inspired by the stability condition of classical *Stable Marriage* problem [21, 46], but other conditions can be designed and cast in the framework by using different soft constraints. *Blocking* coalitions are defined in Def. 3:

Definition 3 *Two coalitions C_u and C_v are defined as* blocking *if, an individual $x_k \in C_v$ exists such that, $\forall x_i \in C_u, x_j \in C_v$ with $j \neq k$, $\circ_{x_i \in C_u} t(x_k, x_i) > \circ_{x_j \in C_v} t(x_k, x_j)$ and $T(C_u \cup x_k) > T(C_u)$ at the same time.*

A set $\{C_1, C_2, \ldots, C_n\}$ of coalitions is *stable*, i.e. is a valid solution, if no blocking coalitions exist in the partitioning of the agents. An example of two blocking coalitions is sketched in Figure 6: if x_4 prefers the coalition C_1 (i.e. relationship r_1 in Figure 6) to the elements in its coalitions C_2 (i.e. r_2 in Figure 6), i.e. $\circ(t(x_4, x_1), t(x_4, x_2), t(x_4, x_3)) > \circ(t(x_4, x_5), t(x_4, x_6), t(x_4, x_7))$, and C_1 increases its trust value by having x_4 inside itself, i.e. $T(C_1 \cup x_4) > T(C_1)$, then C_1 and C_2 are two blocking coalitions and the partitioning $\{C_1, C_2\}$ is not stable and thus, it is not a feasible solution of our problem.

We therefore require the stability condition to be satisfied, but at the same time we want also to optimize the trustworthiness of the partitioning given by aggregating together all the trustworthiness scores of the obtained coalitions.

The results of this Section can be applied also to model the formation and the consequent behaviour of the other organizational paradigms, e.g. *Holoarchies*, *Federations* or *Teams* [16]. To do so, we need to represent the different grouping relationships among the entities with soft constraints.

8.1. A Formalization of the Problem

In this Section we define the soft constraints needed to represent the coalition-extension problem. We adopt the *Fuzzy* semiring $\langle [0,1], max, min, 0, 1\rangle$ in order to maximize the minimum trustworthiness of all the obtained coalitions (as proposed also in [17, 18]). The following definition takes the general $\circ$ operator (presented in Section 5 and Section 8) as one of its parameters: it can be considered in some sense as a "lower level" operator with respect to the other two semiring operators (i.e. $+$ and $\times$).

The variables V of this SCSP (see Section 2) problem are represented by the maximum number of possible coalitions: $\{co_1, co_2, \ldots, co_n\}$ if we have to partition a set $\{x_1, x_2, \ldots, x_n\}$ of n elements. The domain D for each of the variables is the powerset of the element identifiers, i.e. $\mathcal{P}\{1, 2, \ldots, n\}$; for instance, if $\eta(co_1) = \{1, 3, 5\}$ it means the the coalition co_1 groups the elements x_1, x_2, x_5 together ($\eta : V \rightarrow D$ is the variable assignment function shown in Section 2). Clearly, $\eta(co_i) = \emptyset$ if the framework finds less than n coalitions.

1. *Trust* constraints. As an example from this class of constraint, the soft constraint $c_t(co_i = \{1, 3, 5\}) = a$ quantifies the trustworthiness of the coalition formed by $\{x_1, x_3, x_5\}$ into the semiring value represented by a. According to Def. 2, this value is obtained by using the $\circ$ operator and composing all the 1-to-1 trust relationships inside the coalition. In this way we can find the best set of coalitions according to the semiring operators.
2. *Partition* constraints. This set of constraints is used to enforce that an element belongs only to one single coalition. For this goal we can use a binary crisp constraint between any two coalition, as $c_p(co_i, co_j) = \mathbf{0}$ if $\eta(co_i) \cap \eta(co_j) \neq \emptyset$, and $c_p(co_i, co_j) = \mathbf{1}$ otherwise (with $i \neq j$). Moreover, we need to add one crisp constraint more, in order to check that all the elements are assigned to one coalition at least: $c_p(co_1, co_2, \ldots, co_n) = \mathbf{0}$ if $|\eta(co_1) \cup \eta(co_2) \cup \cdots \cup \eta(co_n)| \neq n$, and $c_p(co_1, co_2, \ldots, co_n) = \mathbf{1}$ if $|\eta(co_1) \cup \eta(co_2) \cup \cdots \cup \eta(co_n)| = n$.
3. *Stability* constraints. These crisp constraints model the stability condition as proposed in Def. 3. We have several ternary constraints for this goal: $c_s(co_v, co_u, x_k) = \mathbf{0}$ if $k \in \eta(co_v)$ (i.e. x_k belongs to the co_v coalition), $\circ_{i \in \eta(co_u)} t(x_k, x_i) > \circ_{j \in \eta(co_v)} t(x_k, x_j)$ and $c_t(\eta(co_u) \cup k) > c_t(co_u)$. Otherwise, $c_s(co_v, co_u, x_k) = \mathbf{1}$.

The above constraints provide one example of how formation constraints can be imposed on coalitions; many other formation constraints are possible so long as they can be encoded as constraints within our model. For example, the Partition constraints in 2 above could be weakened to support Chinese Wall style partitioning. In this case, a coalition is a group of market analysts working together for some company. Market analysts can be assigned to a number of different coalitions so long as it does not violate the conflict of interest partition constraint. Assuming that the company is also an individual (the trustor) in the coalition, then we'd like to maximize the trust (with the analysts) within each coalition.

DOI: 10.1002/sec

9. Conclusions and Future Work

We have defined the concept of multitrust and provided a method to represent and solve the trust propagation problem using a combination of *and-or* graphs and SCLP programming. Our framework allows different trust metrics to be encoded as a semiring; in this paper we used trust and confidence, providing an estimation of the trust observation. We believe that multitrust can be used in many real-world cases: trusting a group of individuals at the same time can lead to different conclusions with respect to simply aggregating together the trust values of the single individuals in the group.

A small-world interpretation of the trust model has been evaluated, whereby all the individuals are reachable within few hops. Exploiting the small-world properties show that the framework can be effectively used with small/medium networks containing a few hundreds of nodes.

Different coalition formation criteria based on distinct trust metrics were presented. These criteria can be enriched with particular stability conditions that represent the local requirements of the entities (w.r.t. the global optimization of coalitions). The model proposed in Section 8, is inspired by the stability condition of classical *Stable Marriage* problem [21, 46]. This framework and the one based on multitrust represent two possible ways to use semiring-based frameworks to expressively organize entities into coalitions by using trust scores.

Future work on improving the performance of multitrust evaluation includes the use of memoization/tabling techniques, to filter out redundant computations and the use of *branch and bound* techniques to prune the non promising partial solutions. Some preliminary results obtained with *ECLiPSe* [47] and its branch and bound pruning confirm this observation.

A further goal for future research is to find a structure able to aggregate distinct trust paths in a single trust value, i.e. to compute *multipath* propagation (e.g. an average cost of the independent paths). A solution could be represented by the expectation semiring [40], which is however somehow in contrast with pruning algorithms. In addition, we would like to introduce the notion of "distrust" in the model and to propagate it by using the inverse of the semiring $\times$ operator [26]. At last, we would like to study operators similar to those proposed by in [48], in order to embed them in our semiring-based framework.

References

1. Klusch M, Gerber A. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems* 2002; **17**:42–47, doi: http://doi.ieeecomputersociety.org/10.1109/MIS.2002.1005630.
2. Ziegler CN, Lausen G. Propagation models for trust and distrust in social networks. *Information Systems Frontiers* 2005; **7**(4-5):337–358, doi:http://dx.doi.org/10.1007/s10796-005-4807-3.
3. Theodorakopoulos G, Baras JS. Trust evaluation in ad-hoc networks. *WiSe '04: Workshop on Wireless security*, ACM, 2004; 1–10, doi:http://doi.acm.org/10.1145/1023646.1023648.
4. Bistarelli S, Santini F. Propagating multitrust within trust networks. *SAC*, Wainwright RL, Haddad H (eds.), ACM, 2008; 1990–1994.
5. Martelli A, Montanari U. Optimizing decision trees through heuristically guided search. *Commun. ACM* 1978; **21**(12):1025–1039, doi: http://doi.acm.org/10.1145/359657.359664.
6. Bistarelli S. *Semirings for Soft Constraint Solving and Programming*, *LNCS*, vol. 2962. Springer, 2004.
7. Bistarelli S, Montanari U, Rossi F. Semiring-based constraint logic programming. *Proc. IJCAI97*, Morgan Kaufman, 1997; 352–357.
8. Bistarelli S, Montanari U, Rossi F. Semiring-based constraint solving and optimization. *Journal of the ACM* 1997; **44**(2):201–236.
9. Bistarelli S, Montanari U, Rossi F, Santini F. Modelling multicast QoS routing by using best-tree search in and-or graphs and soft constraint logic programming. *Electr. Notes Theor. Comput. Sci.* 2007; **190**(3):111–127.
10. Bueno F, Cabeza D, Carro M, Hermenegildo M, López-García P, Puebla G. The CIAO prolog system: reference manual. *Technical Report CLIP3/97.1*, School of Computer Science, Technical University of Madrid (UPM) 1997.
11. Gray E, Seigneur JM, Chen Y, Jensen CD. Trust propagation in small worlds. *iTrust*, Springer-Verlag, 2003; 239–254.
12. Venkatraman M, Yu B, Singh MP. Trust and reputation management in a small-world network. *In Proceedings of Fourth International Conference on MultiAgent Systems*, 2000; 449–450.
13. Buchanan M. *Nexus: Small Worlds and the Groundbreaking Theory of Networks*. W. W. Norton & Co., Inc.: New York, NY, USA, 2003.
14. O'Madadhain J, Fisher D, White S, Boey Y. The JUNG (Java Universal Network/Graph) framework. *Technical Report*, UC Irvine 2003.
15. Watts DJ, Strogatz SH. Collective dynamics of small-world networks. *Nature* 1998; **393**:440, doi:10.1038/30918.
16. Horling B, Lesser V. A survey of multi-agent organizational paradigms. *Knowl. Eng. Rev.* 2004; **19**(4):281–316, doi: http://dx.doi.org/10.1017/S0269888905000317.
17. Bistarelli S, Santini F. Propagating multitrust within trust networks. *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, ACM: New York, NY, USA, 2008; 1990–1994, doi:http://doi.acm.org/10.1145/1363686.1364170.
18. Bistarelli S, Santini F. SCLP for trust propagation in small-world networks. *CSCLP*, *Lecture Notes in Computer Science*, vol. 5129, Fages F, Rossi F, Soliman S (eds.), Springer, 2007; 32–46.
19. Theodorakopoulos G, Baras JS. Trust evaluation in ad-hoc networks. *WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security*, ACM: New York, NY, USA, 2004; 1–10, doi:http://doi.acm.org/10.1145/1023646.1023648.
20. Jøsang A, Ismail R, Boyd C. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.* 2007; **43**(2):618–644, doi: http://dx.doi.org/10.1016/j.dss.2005.05.019.

21. Bistarelli S, Foley S, O'Sullivan B, Santini F. From marriages to coalitions: A soft CSP approach. *Recent Advances in Constraints 13th Annual ERCIM International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2008, Revised Selected Papers*, LNAI, Springer, 2008.
22. Twigg A, Dimmock N. Attack-resistance of computational trust models. *WETICE '03*, IEEE Computer Society, 2003; 275–280.
23. Granovetter MS. The Strength of Weak Ties. *The American Journal of Sociology* 1973; **78**(6):1360–1380.
24. Čapkun S, Buttyán L, Hubaux JP. Small worlds in security systems: an analysis of the PGP certificate graph. *NSPW '02: Proceedings of the 2002 workshop on New security paradigms*, ACM: New York, NY, USA, 2002; 28–35, doi: http://doi.acm.org/10.1145/844102.844108.
25. Allenby R. *Rings, Fields and Groups*. Butterworth-Heinemann, 1992.
26. Bistarelli S, Gadducci F. Enhancing constraints manipulation in semiring-based formalisms. *ECAI 2006*, IOS Press, 2006; 63–67.
27. Georget Y, Codognet P. Compiling semiring-based constraints with clp (FD, S). *CP '98: Proc. of Principles and Practice of Constraint Programming*, Springer-Verlag: London, UK, 1998; 205–219.
28. Bistarelli S, Montanari U, Rossi F. Constraint solving over semirings. *Proc. IJCAI95 (Morgan Kaufman)*, Morgan Kaufman, 1995; 624–630.
29. Jaffar J, Maher M. Constraint logic programming: a survey. *Journal of Logic Programming* 1994; **19/20**:503–581.
30. Abdul-Rahman A. The pgp trust model. edi-forum. *Journal of Electronic Commerce* 1997; .
31. Abdul-Rahman A, Hailes S. A distributed trust model. *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, ACM: New York, NY, USA, 1997; 48–60, doi: http://doi.acm.org/10.1145/283699.283739.
32. Bistarelli S, Martinelli F, Santini F. A semantic foundation for trust management languages with weights: An application to the rtfamily. *ATC*, *Lecture Notes in Computer Science*, vol. 5060, Rong C, Jaatun MG, Sandnes FE, Yang LT, Ma J (eds.), Springer, 2008; 481–495.
33. Bistarelli S, Martinelli F, Santini F. Weighted datalog and levels of trust. *ARES*, IEEE Computer Society, 2008; 1128–1134.
34. Costa AC, Roe RA, Taillieu T. Trust within teams: the relation with performance effectiveness. *European Journal of Work and Organizational Psychology* 2001; **10**(3):225–244.
35. Griffiths N, Luck M. Coalition formation through motivation and trust. *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, ACM: New York, NY, USA, 2003; 17–24, doi: http://doi.acm.org/10.1145/860575.860579.
36. Breban S, Vassileva J. A coalition formation mechanism based on inter-agent trust relationships. *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, ACM: New York, NY, USA, 2002; 306–307, doi:http://doi.acm.org/10.1145/544741.544812.
37. Lerman K, OShehory. Coalition formation for large-scale electronic markets. *ICMAS*, IEEE Computer Society, 2000; 167–174.
38. Reiter MK, Stubblebine SG. Path independence for authentication in large-scale systems. *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, ACM: New York, NY, USA, 1997; 57–66, doi: http://doi.acm.org/10.1145/266420.266435.
39. Mingwu Z, Bo Y, Wenzheng Z. A semiring privacy protect model. *NPC '07: Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops*, IEEE Computer Society: Washington, DC, USA, 2007; 255–262.
40. Eisner J. Parameter estimation for probabilistic finite-state transducers. *ACL '02*, Association for Computational Linguistics, 2001; 1–8, doi: http://dx.doi.org/10.3115/1073083.1073085.
41. Watts DJ. *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press: Princeton, NJ, USA, 1999.
42. Ramakrishnan IV, Rao P, Sagonas KF, Swift T, Warren DS. Efficient tabling mechanisms for logic programs. *International Conference on Logic Programming*, The MIT Press, 1995; 697–711.
43. Wunderwald JE. Memoing evaluation by source-to-source transformation. *LOPSTR '95: Proceedings of the 5th International Workshop on Logic Programming Synthesis and Transformation*, Springer-Verlag: London, UK, 1996; 17–32.
44. Schrijvers T, Warren DS. Constraint handling rules and tabled execution. *ICLP*, *LNCS*, vol. 3132, Springer, 2004; 120–136.
45. Bistarelli S, Frühwirth T, Marte M. Soft constraint propagation and solving in chrs. *SAC '02: Proc. of the 2002 ACM symposium on Applied computing*, ACM Press: New York, NY, USA, 2002; 1–5, doi:http://doi.acm.org/10.1145/508791.508793.
46. Iwama K, Miyazaki S. A survey of the stable marriage problem and its variants. *International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008)*, IEEE Computer Society, 2008; 131–136.
47. Apt KR, Wallace M. *Constraint Logic Programming using Eclipse*. Cambridge University Press: New York, NY, USA, 2007.
48. Jøsang A, Hayward R, Pope S. Trust network analysis with subjective logic. *ACSC*, *CRPIT*, vol. 48, Estivill-Castro V, Dobbie G (eds.), Australian Computer Society, 2006; 85–94.