Knots Maintenance for Optimal Management of Trust Relations

Libi Gur, Nurit Gal-Oz, Ehud Gudes

Knots Maintenance for Optimal Management of Trust Relations

Libi Gur[1], Nurit Gal-Oz[2] and Ehud Gudes[1]

[1] Ben-Gurion University, Beer-Sheva, 84105, Israel, libigu@cs.bgu.ac.il, ehud@cs.bgu.ac.il
[2] Sapir Academic College, D.N. Hof Ashkelon 79165, Israel galoz@sapir.ac.il

Abstract. The knot model is aimed at obtaining a trust-based reputation in communities of strangers. It identifies groups of trustees, denoted as knots and among whom overall trust is strong, and is thus considered the most capable solution for providing reputation information to other members within the same knot. The problem of identifying knots in a trust network is modeled as a graph clustering problem. When considering dynamic and large-scale communities, the task of keeping the clustering correct over time is a great challenge. This paper introduces a clustering maintenance algorithm based on the properties of knots of trust. A maintenance strategy is defined that addresses violations of knot properties due to changes in trust relations that occur with time in response to the dynamic nature of the community. Based on this strategy, a reputation management procedure is implemented in two phases: the first identifies the essence of change and makes a decision regarding the need to improve knot clustering. The second phase locally modifies the clustering to preserve a stable network structure while keeping the network correctly clustered with respect to the knot utility function. We demonstrate by simulation the efficiency of the maintenance algorithm in preserving knots quality, for cases in which only local changes have occurred, to ensure the reliability of the reputation system.

Keywords: trust, reputation, maintenance, model, clustering

1 Introduction

The fast growth of the internet encouraged the creation of user-cooperative applications called virtual communities. In these communities, users may choose to make their identities known or to remain anonymous. Therefore, reputation and trust play major roles in such virtual communities by enabling users to interact with other virtual users (total strangers) and to establish interactions that are based on mutual benefit. Reputation allows members to build trust or a level of confidence in other members within the context of decision making or other objectives. The method of choice for providing the means through which reputation and ultimately trust can be quantified and disseminated is a reputation system. A reputation system computes and publishes reputation scores for a set of entities (e.g., services or experts) within a community, and those scores are inferred from a collection of ratings supplied by another group of entities (e.g., members of the community). The ratings are typically transferred to a reputation engine that plugs them into a specific reputation algorithm to dynamically compute the reputation scores.

Several trust-based reputation strategies and models were developed to produce reputation metrics for specific communities [1] Most of these strategies treat a community as a single, homogeneous entity and do not explicitly address the issue of community diversity. The knot-aware trust-based reputation model for virtual communities introduced by Gal-Oz et al. [2] refers to a community as a collection of knots (sub-communities). A knot is defined as a group of community members having overall "strong" trust relations between them. As was shown in [2] defining such knots enables reputation to be more accurately computed, which, in turn, results in the derivation of more reliable trust measures. Naturally, it also helps protect members from fraud and manipulation by other virtual members. The knot-aware clustering algorithm presented in [3] partitions the community into knots of members who have strong trust relations between them, while the trust relations of members who are not in the same knot are much weaker. The main goal of the knot-aware management system is to maintain knots attributes and encourage honesty among members by identifying and subsequently excluding members with dishonest or biased recommendations. Whenever reputation changes and trust relations are being modified, the reputation management algorithm must examine the accumulated trust relations of members and exclude members from their knot accordingly. A re-clustering algorithm is then applied to cluster the excluded members to gain maximum utility for the whole community.

Over time, the reputation system evolves and changes occur. These changes incude the participation of community members in new transactions, the creation of new trust relations among members, and the modification of existing trust relations, which together cause the existing knot structure to become sub-optimal. Keeping the knots model consistent is not straightforward. The reputation system must be able to collect the information pertaining to its members' new experiences and, according to some predefined criteria, invoke a maintenance algorithm that is based on this information to maintain knot properties. Successful maintenance may detect users who try to manipulate the reputation system to their own benefit, thereby causing an attack on the system. The detection of such dishonest behavior may result in their removal from their original knot, which will reduce their inuence considerably. A successful reputation system and especially its maintenance strategy should be evaluated by the quality both of its reputation computation and of its defense against attacks.

In this paper we investigate the problem of maintenance strategy in the knots model, in which knots are constructed using a graph clustering algorithm. The maintenance algorithm must consider the existing clusters and make as few changes as possible to restore their quality. Such maintenance algorithm must be compared to the complete re-clustering using all available information, since such re-clustering may be problematic for two reasons: first, for a large community it may be very computationally heavy, and second, it may completely change the structure of the knots, which may cause instability in the computation of some users reputations. A good maintenance algorithm, therefore, which should avoid these two problems, is presented in this paper. Although the algorithm is specic to the knots model, the overall strategy is of a general nature, and as such, it can be applied to other trust-based reputation models.

This paper makes three major contributions. First, we propose a knot-aware reputation management algorithm by which the knot-aware system can maintain its knots properties based on the different viewpoints and opinions of of all of its knots. We evaluate the resulting knots based on objective clustering quality measures, compare them to the results of a complete reclustering algorithm, and analyze and evaluate the possible design choices of the knot management algorithm to determine which ones are optimal. Next, we investigate the problem of when to apply the knot-aware management algorithm since executing such an algorithm is computationally intensive, and therefore, it should probably not be invoked every time there is a minor reputation change or for every new rating performed in the system. Finally, we conduct an evaluation based on a large-scale simulation of a virtual community to demonstrate the effectiveness of our algorithm, including its ability to detect attacks.

The rest of the paper is organized as follows. The next section provides the necessary background on the knots model and knots clustering and presents an overview of related work. Section 3 discusses the strategy of knots maintenance and the main parameters of the algorithm. In section 4 the knot management algorithm is formally defined and all states of reputation and trust modifications are tabulated. Section 5 present the experimental evaluation. It uses a simulation of a large-scale virtual community that is based on an existing knots structure and then simulate thousands of new ratings ratings, causing the trust relations between members to change, thereby necessitating invocation of the maintenance algorithm. This section contains analyses of parameters of the knot-aware management algorithm for different graphs and different frequencies of changes. The results are then used to determine knot qualities. Finally, we simulate some attacks against the reputation system and show knot maintenance algorithm effectiveness in the detection of those attacks. We conclude in section 6 and suggest some directions for future research.

2 Related Work & Background

In this section we first review related work on reputation-based clustering algorithms and their maintenance and then review the Knots clustering algorithm of [3] and define its main parameters, which will also be used by the knots maintenance algorithm in the rest of the paper.

2.1 Related work

Two recent papers present cluster computing management based on Scheduling [4, 5]. After the clusters are established, the trust relations of the community network may change due to members activities. A perfect clustering is not limited to the initial clustering of network. Rather, It should respond to the natural dynamics of the community network. Cluster initialization, therefore, should only be executed once, while cluster maintenance needs to be performed repeatedly. An example of such application-based scheduling is the P2P scheduling system [4], which includes trust, incentives, fairness, security, and new criteria for evaluating performance. It encompasses the activities involved in the management of network applications. The maintenance of reputation is discussed in [5], where two constraints are emphasized. The rst is that the maintenance operation will not be continuously monitoring and checking the clustering at any given moment, a setup that would require too many resources and cause overhead in the system, which together would result in overall poor maintenance performance efciency. The second constraint is related to the number of ratings needed to perform each update, as the implementation of maintenance and reputation adjustment operations based on only a few ratings at a time will also lead to system inefficiency. Considering these two constraints [5] introduced a scheduling algorithm that decides when, during a certain time interval, the behavior of members should be checked. Such recurrent maintenance based on periodic checks is essential to the proper evaluation of trust and reputation and a major objective of our work.

The vast majority of the papers in the literature handle cluster maintenance using a node-centric approach [6] and work in the presence of node mobility. These cluster maintenance algorithms handle situations of change that include a node moving away from a cluster, a new node joining a cluster, a cluster splitting due to its excessive num- ber of nodes, and the merging of clusters. In contrast to the node-centric approach, Wang et. al [7] presented a cluster-centric maintenance algorithm that is based on a number of interesting properties of diameter-2 graphs. Rather than requiring complete cluster topology information to be maintained at each node, this algorithm depends on a spanning tree maintained at some specic node that functions as a maintenance leader, makes maintenance decisions, and informs all the other nodes in the original cluster. Unlike these algorithms, our knot maintenance algorithm is edge-centric. An edge between two nodes, indicating a trust relationship, belongs to one cluster (intra-knot) or is located between two clusters (inter-knot). Changes in edge values due to newly formed trust relations and the modification of existing trust relations are handled by the knot maintenance algorithm.

Most of protocols handle cluster maintenance by periodic re-clustering [8, 9] and re-cluster the nodes from time to time to satisfy specic cluster characteristics, which results in the consumption of excessive network resources. The knot maintenance algorithm separates the clustering into two phases, cluster initialization and cluster maintenance. During the latter phase, initial cluster congurations may be modied, depending on members behavior. Cluster initialization should only be executed once, while cluster maintenance must be performed repeatedly. As such, our algorithm aims to minimize overhead and enhance knots stability.

2.2 Review of the Knots Clustering algorithm

In this section we review the knot clustering algorithm of Gal-Oz et al. [3] and present some basic terms that will be used later. The virtual community is described, without loss of generality, as a community in which experts in specic elds offer their advice and consulting services to community members who seek such services. A community consists of individual members, all of who may participate in community ac- tivities, such as searching for an expert, interacting with an expert, and sharing opinions about experts with other members. Although experts are a subclass of members, they are considered as two disjoint sets for simplicity. The trust that two members have in the same expert (Trust Expert) is used to infer their implicit trust in each other (Trust Member). Gal-Oz et al. [3]] discussed the problem of partitioning the members of the community into knots and introduced a knots clustering algorithm. The problem of partitioning the community into knots is similar to the optimization problem known as Correlation Clustering (CC) [10]. The community is being represented as a graph $G = (V, E)$(called a community graph) in which vertices correspond to members and edge weights describe direct trust relations between the members. The knot clustering problem is very close to the CC optimization problem, since the latter is dened on a graph in which the edge label indicates whether two nodes are similar (+) or different (-) and the task is to cluster the vertices such that similar vertices are grouped together. Unlike other clustering algorithms, the CC algorithm does not require that the number of clusters be specified in advance. The solution of the CC optimization problem is known to be NP-hard. Bansal et al. [10] discussed the NP- completeness proof and also presented both a constant factor approximation algorithm and a polynomial-time approximation scheme to nd the clusters in this setting. Ailon et al. [11] propose a randomized 3-approximation algorithm for the same problem.

The knot clustering solution presented in [3] differs from the classical CC problem in that it attempts to satisfy several major objectives derived from the virtual communities domain and trust knots. First, the algorithm considers weighted edges and not just (+,-) edges. The goal is to create strong knots, having as many high weighted edges indicating strong trust relations and as few low weighted edges within a knot. The weight of an edge is based on the value of the direct trust (based on first-hand interaction) between the pair of vertices at its end-points. The weight of an edge is based on the value of the direct trust (based on rst-hand interaction) between the pair of vertices at its end-points. These values are used to compute the similarity between vertices, which is referred to as the Mutual Trust in Member (MTM) relation and corresponds to the minimum trust either member has for the other [2]. In this way, the edge weight is used as the input for the clustering algorithm, which must decide whether or not its two end-vertices should reside in the same cluster.

Secondly, is the use of the trust indirectness property [12]. Based on a more general trust, indirect trust is derived from the ratings of other members, which are known as transitive trust-chains [13]. The knot clustering uses transitive trust to provide knots with modified level of distributed trust among knot's members. This is done by clustering together two nodes that have a neighbor node(s) with high trust between them even if there is no strong trust relation between these two original nodes. Finally, the algorithm ensures that the indirect trust relations between any pair of members in any knot possess reliability, which depends on limiting the knot diameters [14]. Thus, the knots clustering algorithm requires that the length of the path between each pair of vertices be limited. This limitation, denoted the Trust Chain Length (TCL), denes the length of the longest trust chain connecting any two vertices within any knot in the cluster.

In the knot clustering algorithm, the edge weight is not exactly the MTM. The edge weight must reflect the community perception of how strong a trust relation should be between two members of the same knot. Therefore, the MTM is normalized by a weighting function called the WF, which uses a Trust Threshold Level (TTL) that is dened in the range of [0.5,1] since trust in our model is in the ranges of [0;1], such that 1 represents complete trust and 0 represents complete distrust. WF output that has a positive sign signifies that the two members should be assigned to the same knot; otherwise, they should not. The value of the WF reects the extent to which the decision is believed to be true, i.e., the condence in the decision [3].

The problem of partitioning the community into knots is solved by a heuristic algorithm that uses the hierarchical approach [15]. First the community graph denoted by $CC =< V, E_{cc} >$ is generated by assigning each edge $e_{ij} \in E_{cc}$ a label and a weight w_{ij}, based on $MTM(e_{ij})$ and the weighting function. Next, the hierarchical clustering algorithm is applied on the CC graph, calculating the connectivity components of the graph based on the positive edges. In the initial state all vertices form singleton clusters. Then pairs of clusters are iteratively merged based on their merging utility, denoted MCC. In each iteration the two clusters for which the MCC is highest are merged into a single cluster. The result of the clustering is defined by a clustering matrix $M^c = \{x_{ij} | i, j = 1, \ldots, |V|\}$ where $x_{ij} = 1$ if vertices v_i and v_j belong to the same cluster or $x_{ij} = 0$ if they are in different clusters.

The quality of the clustering is measured by two desired properties of knots. First, the **Strength** of the clustering (Strength(C)) is measured by the total strength of its knots where the strength of a knot is computed by the average of its vertices degree. Second, the **Stability** of a knot represents the minimal amount of trust loss that would justify splitting the knot into two sub-knots. More specifically, we search for a minimum cut (MinCut) of the knot, i.e., the cut having the smallest sum of MTM values of edges. Intuitively, if the MinCut value of a knot is high, many changes (e.g., decrease or increase of MTM value on intra-knot or inter-knot edges respectively) must occur to justify a split.

3 Applying Clustering Maintenance to maintain Knots

The community graph represents a dynamic trust network that is continuously changing, and therefore, it is time-dependent. At the initialization stage, a clustering algorithm is invoked and the community graph is partitioned into individual clusters, each of which satisfies the required knot properties. Over time, members of the community take part in new transactions and generate new evaluations that lead to the creation of new trust relations, e.g., between members who had no comparable experience before. These changes can be summarized as changes in MTM between vertices. These changes can be summarized as changes in the MTM values between vertices, and they can, in turn, lead to possible violations of the strength and stability properties of the desired knot. Any violation of these properties should elicit the running of a cluster maintenance algorithm.

The term maintenance in this paper refers to the update activity performed by the clustering maintenance algorithm, an activity that results in the modification of knots structures. An important part of preserving updated values of trust and reputation, is periodic maintenance of knot structure that helps both to prevent collusion and to discourage members from acting maliciously. The maintenance operation updates the trust value of each member in the community based on the members recent behavior, thereby increasing the reliability of the reputation mechanism. In addition, the role of clustering maintenance is to rene knots clustering whenever community behavior changes. In this sense, renement corresponds to restoring the strength and stability of a clustering in which the values for those two parameters have decreased. In other words, the refinement process improves the quality of a clustering to obtain more precise reputation values.

3.1 Clustering Maintenance Prerequisites

A maintenance clustering algorithm should operate only when there is a high probability that a better clustering exists and in accordance with a community policy. The goal is to carry out maintenance actions with minimum overhead, in the process allowing members to join and leave a knot without perturbing the trust relations of the knot and preserving the current knot structure as much as possible. Our maintenance policy is motivated by several objectives. The first objective is to minimize the number of invocations of the algorithm since executing the maintenance algorithm is computationally intensive, and it should only be performed when signicant changes have occurred in MTM values. The maintenance module schedules periodic evaluations of the extent of changes in trust that have occurred in the system. Increases above a certain threshold in the amount of new information invokes the maintenance algorithm. The second objective is to maintain knots that are stable and strong by collecting and preserving a large amount of aggregated MTM. The maintenance procedure thus aims to track changes in the MTM values of edges, in the process identifying within knot edges whose MTM values have decreased or increased beyond a predened trust threshold level. To that end, we analyze the nature of changes in the community graph with respect to the elapsed time and the number of new members ratings that have been accumulated.

Definition 1. A Maintenance Interval(MI) *a period of time during which the members ratings are monitored and analyzed by the maintenance function, is dened as:*

$$MI^{t_i} = [t_i - \beta; t_i] \tag{1}$$

where t_i is the scheduled time for the maintenance action and β represents the length of the time slot in which members' interactions are taken into account for analysis. β represents the sensitivity of the community to changes. A large (small) β represents how much (little) a community is concerned with preserving the structure of its knots inspite of changes in its members' ratings.

Definition 2. *An* R-level *of the maintenance mechanism is the total number of new ratings collected during a maintenance interval above which the maintenance algorithm is invoked. Let N_r be the number of new ratings obtained during MI^{t_i}. A maintenance action will be invoked if N_r is greater than the R-level.*

Definition 3. *A Trust Expert List of member A at maintenance interval MI^t, denoted $TEL^t(A, x_1, x_2, ..., x_n)$ stores the updated information of direct trust that member A has in each expert $x_i, i = 1..n$. The trust in expert $TE(A, x)$ is calculated based on all ratings provided by A upon each transaction with x , while taking into account the time at which the transaction, (old transactions weigh less than new ones).*

Consider the following observations regarding the effects of changes in the trust values between two members caused by new ratings. Assume the edge weights are calculated using the weighting function discussed in section 2.2. The term *inter-cluster* edge refers an edge connecting two nodes hosted in two different knots, while *an intra-cluster* edge refer to an edge connecting two nodes of the same knot.

Observation 1 *Decreases in the strength and stability of a clustering due to inter-cluster edges can only be caused in two cases:*

1. *by negative edges whose weights increase and they become positive.*
2. *by positive edges whose weights increase.*

Negative inter-cluster edges whose weights decrease, or positive edges whose weights decrease can only improve cluster correctness.

Observation 2 *Decreases in the strength and stability of a clustering due to intra-cluster edges can only be caused in two cases:*

1. *by negative edges whose weights decrease, or positive edges whose weights decrease can only improve cluster correctness. whose weight decreases.*
2. *by positive edges whose weights decrease and become negative.*

Positive intra-cluster edges whose weights increase or negative edges whose weights increase can only improve cluster quality.

Based on these observations we now define two maintenance policies.

Correctness Policy: Maintenance should be conducted in the case of either observation 1 or observation 2 carried out . This policy is not tolerant to a decrease in clustering correctness.

Relaxed Policy: Maintenance should be conducted only in the case of observation 2. This policy is tolerant to a decrease in clustering correctness due to observation 1,since it does not affect the quality of the connections within the clusters,and therefore, the contribution of intra-knot members to each other remains basically about the same (the strength of the knot). so that basically the contribution of intra-knot members to each other remains basically about the same (the strength of the knot).

Under the assumption of honest members we can be more biased toward stability (low sensitivity). For example - if an honest member's rating of an expert differs from that of overall knot opinion, that members rating should be considered a contribution to knot opinion rather than dishonesty, and that member should be allowed to remain in the knot . Under the assumption of malicious or dishonest participants, we should be more biased toward correctness (high sensitivity). For example if an attackers rating of an expert differs from that of overall knot opinion, that attackers rating is strictly considered to be damaging to knot opinion, and the attacker should be removed from the knot.

4 The Knot Maintenance Algorithm

In this section, we present our knot-aware management policy, which uses a scheduled reputation maintenance algorithm to evaluate the changes that have taken place in the trust relations of community members. The algorithm identies decreases in knot correctness and acts to exclude the members that have caused this decrease (i.e., they are evaluated as unsuitable for the knot). It then applies the hierarchical approach for re-clustering the semi-clustered graph in which the excluded nodes are singleton clusters.

Before invoking the maintenance procedure (see Algorithm 1), it is important to describe the process of updating the weights on the graph edges. During a maintenance interval MI, new ratings are accumulated and the TEL is modified accordingly. The MTM values between a vertex and its neighbors are then calculated in accordance with the updated TEL. Finally the weighting function and the threshold value TTL are applied and the new edge weights are calculated. For existing edge weights, the results may comprise an increase, a decrease, or no change, and for a newly created edge, the results will be its initial weight.

The maintenance algorithm is executed in four phases. In the first phase, during the MI the new ratings are accumulated. At the scheduled time of maintenance, the information that has been accumulated is analyzed. If the number

Algorithm 1 Maintenance Algorithm

Input

$G =< V, E >$: Community graph, V: the set of vertices, E: the set of MTM edges

C_G: A clustering of G.

WF: Weight function.

MTM: Mutual Trust Member relation function.

κ: Maximum allowed TCL.

α: Community TTL.

Output

C: An updated clustering of G.

```
 1: /* Second Phase */
 2: $CalculateTEL(RatingsList, V)$
 3: $E^{'} = WF(MTM(E^{'}, \alpha))$
 4: $G_{new} =< V, E^{'} >$
 5: $C = C_G$
 6: /* Third phase */
 7: $clustersList = \{\}$ /*The list of clusters for the re-clustering phase*/
 8: for all $e = (v_i, v_j) \in E^{'}$ do
 9:   /* process intra-cluster edges */
10:   if $cluster(v_i) = cluster(v_j)$ then
11:     if $(WF(v_i, v_j)_{old} \leq 0 \wedge WF(v_i, v_j)_{new} \leq 0 \wedge WF(v_i, v_j)_{new} \leq WF(v_i, v_j)_{old}) \vee (WF(v_i, v_j)_{old} \geq 0 \wedge WF(v_i, v_j)_{new} \leq 0)$ then
12:       $clustersList.addVertexAsSingletonCluster(v_i)$
13:       $clustersList.addVertexAsSingletonCluster(v_j)$
14:       $C \leftarrow reconstructCluster(C, v_i)$
15:   /*process inter-cluster edges */
16:   if $cluster(v_i) \neq cluster(v_j)$ then
17:     if $((WF(v_i, v_j)_{old} \leq 0) \wedge (WF(v_i, v_j)_{new} \geq 0)) \vee (WF(v_i(WF(v_i, v_j)_{old} \geq 0 \wedge WF(v_i, v_j)_{new} \geq 0 \wedge WF(v_i, v_j)_{new} \geq WF(v_i, v_j)_{old})$ then
18:       $clustersList.addVertexAsSingletonCluster(v_i)$
19:       $clustersList.addVertexAsSingletonCluster(v_j)$
20:       $C \leftarrow reconstructCluster(C, v_i)$
21:       $C \leftarrow reconstructCluster(C, v_j)$
22: /* Fourth phase - re-clustering */
23: $candidatePairs \leftarrow allPositiveMCCPairs(clustersList, C)$
24: while $candidatePairs \neq \emptyset$ do
25:   Extract the cluster $(c_i, c_j) \in candidatePairs$ whose MCC is maximal;
26:   if $\forall u, v \in (c_i \cup c_j)|TCL_{c_{ij}}(u, v)| \leq \kappa$ then
27:     $\forall c \in \{c_i, c_j\}$
28:     if $c \in C$ then
29:       $C.remove(c)$
30:     else
31:       $clustersList \leftarrow clustersList - \{c\}$
32:     $C \leftarrow C \cup merge(c_i, c_j)$
33:     $candidatePairs \leftarrow getPositiveMCCPairs(clustersList, C)$
```

Split action	Member position	Old WF(i,j)	New WF(i,j)
Ignore	Intra-knot	$WF(i,j)_{old} \leq 0$	$WF(i,j)_{new} \leq 0$ $WF(i,j)_{new} \geq WF(i,j)_{old}$
Ignore	Intra-knot	$WF(i,j)_{old} \leq 0$	$WF(i,j)_{new} \geq 0$ $WF(i,j)_{new} \geq WF(i,j)_{old}$
perform	Intra-knot	$WF(i,j)_{old} \leq 0$	$WF(i,j)_{new} \leq 0$ $WF(i,j)_{new} \leq WF(i,j)_{old}$
Ignore	Intra-knot	$WF(i,j)_{old} \geq 0$	$WF(i,j)_{new} \geq 0$ $WF(i,j)_{new} \geq WF(i,j)_{old}$
Ignore	Intra-knot	$WF(i,j)_{old} \geq 0$	$WF(i,j)_{new} \geq 0$ $WF(i,j)_{new} \leq WF(i,j)_{old}$
perform	Intra-knot	$WF(i,j)_{old} \geq 0$	$WF(i,j)_{new} \leq 0$ $WF(i,j)_{new} \leq WF(i,j)_{old}$
Ignore	Inter-knot	$WF(i,j)_{old} \leq 0$	$WF(i,j)_{new} \leq 0$ $WF(i,j)_{new} \geq WF(i,j)_{old}$
perform	Inter-knot	$WF(i,j)_{old} \leq 0$	$WF(i,j)_{new} \geq 0$ $WF(i,j)_{new} \geq WF(i,j)_{old}$
Ignore	Inter-knot	$WF(i,j)_{old} \leq 0$	$WF(i,j)_{new} \leq 0$ $WF(i,j)_{new} \leq WF(i,j)_{old}$
perform	Inter-knot	$WF(i,j)_{old} \geq 0$	$WF(i,j)_{new} \geq 0$ $WF(i,j)_{new} \geq WF(i,j)_{old}$
Ignore	Inter-knot	$WF(i,j)_{old} \geq 0$	$WF(i,j)_{new} \geq 0$ $WF(i,j)_{new} \leq WF(i,j)_{old}$
Ignore	Inter-knot	$WF(i,j)_{old} \geq 0$	$WF(i,j)_{new} \leq 0$ $WF(i,j)_{new} \leq WF(i,j)_{old}$

Table 1. Summary of Maintenance Policy

of total community ratings is above R-level, we continue to the next step of the maintenance algorithm; otherwise, there is not enough information to justify maintenance.

The second phase updates the MTM values and the edge weights using the weighting function. Calculating new MTM values based on the updated TEL is performed using an edge matrix $M_{new}^{MTM} = \{x_{ij} | i,j = 1,\ldots,|V|\}$ which holds the MTM value of each pair of vertices of the community graph, where $x_{i,j} = 0$ for $MTM(v_i, v_j) = 0$ when v_i and v_j have no comparable ratings and $x_{i,j} \neq 0$ for $MTM(v_i, v_j) \neq 0$ when v_i have a number of comparable ratings with v_j.

After the nal edge weights are determined, we apply WF to each pair of vertices in the M_{new}^{MTM} matrix to convert it to the M_{new}^{WF} matrix.

The third phase entails analyses of changes in members' behavior and of changes in edge weights that have occurred in existing trust relations, or it evaluates new relations that have been created. The changes can be tracked by

Algorithm 2 reconstructCluster

```
1: function $reconstructCluster(clustering\ C, vertex\ v)$
2:     $cluster \leftarrow cluster(v)$
3:     $cluster.RemoveVertex(v)$
4:     if $\exists u, w \in cluster$ s.t. $|TCL(u, w)| \geq \kappa$ then
5:         $clustersList.addAllVerticesOfCluster(cluster)$
6:         $C.remove(cluster)$
7:     return $C$
```

comparing values of the M^{WF}_{new} matrix with the values of the previous community graph weight matrix M^{WF}. The algorithm behaves differently for inter-cluster vs. for intra-cluster edges. The exact operations of this phase are detailed in Table 1. Each entry in the table corresponds to the changes that occur in the weights between any two nodes. The decision may be either to extract the node from its current node, or to leave it as is. The purpose of this strategy is to exclude from a knot members whose opinions are different from those of the rest of the knot members and to perform a re-clustering action in predened cases.

The fourth phase of the algorithm is the re-clustering of the singleton clusters using the original hierarchical clustering of [3], wherEach entry in the table corresponds to the changes that occur in the weights between any two nodes. The decision may be either to extract the node from its current node, or to leave it as is. The purpose of this strategy is to exclude (from a knot) members whose opinions are is different from those of the rest of the knot members and to perform a re-clustering action in predened cases.e the pair of clustered to be merged is the pair with the highest MCC value.

The algorithm is depicted in detail in Algorithm 1. Only the third phase, which is the central focus of this paper, is described in detail and follows the rules described in Table 1. If the exclusion of a node from a cluster result in trust chains of length higher then κ the cluster is split according to Algorithm 2. The clustering that result from the third phase is the input for the fourth phase.

5 Evaluation results

We evaluate the maintenance algorithm in light of our goal to reduce the need to perform complete re-clustering operations after changes have occurred in the community trust graph by instead conducting local modications to the knot clustering of the graph. Therefore, we evaluate the performance of our maintenance algorithm in terms of knot strength and stability and in terms of the quality of the clustering as a classication algorithm.

For the purposes of evaluation, we use a set of community graphs, each constructed using the same set of members and a different set of members ratings. For each simulation, we randomly generated over 5000 ratings meant to simulate the ground true knowledge regarding of an existing clustered structure. For each experiment, ratings were constructed with a different level of noise ranging from

10% to 30% (noise is a signicant change in members' rating proles that may cause deviations from the original knot structure).

Each simulation begins by applying the knot clustering, the results of which can vary from clearly identfied knots to an arbitrary partitioning of the knots in accordance with the existing trust levels. Next, at each scheduled maintenance time ti, the maintenance algorithm is invoked if the total number of ratings collected during the maintenance interval is above the R-level. New ratings can result either in new edges in the graph or in modifications of the trust values (edge weights) between existing members represented by nodes.

The goal of the maintenance algorithm is to track signicant changes in edge values that violate clustering structure, remove the corresponding vertices from existing knots, and then apply partial re-clustering to obtain an optimal set of knots. The resulting set of knots is compared with the alternative of repeatedly conducting knot clustering. Therefore, each time a maintenance event is performed, we execute knot clustering on the full data set of ratings. For input, maintenance re- clustering requires the last clustering and the set of ratings collected by the last maintenance interval. For the knot clustering operation, the input is the community graph, including the ratings collected by all maintenance intervals. The experiments are repeated for graphs with different levels of clear cluster structures and for different levels of noise.

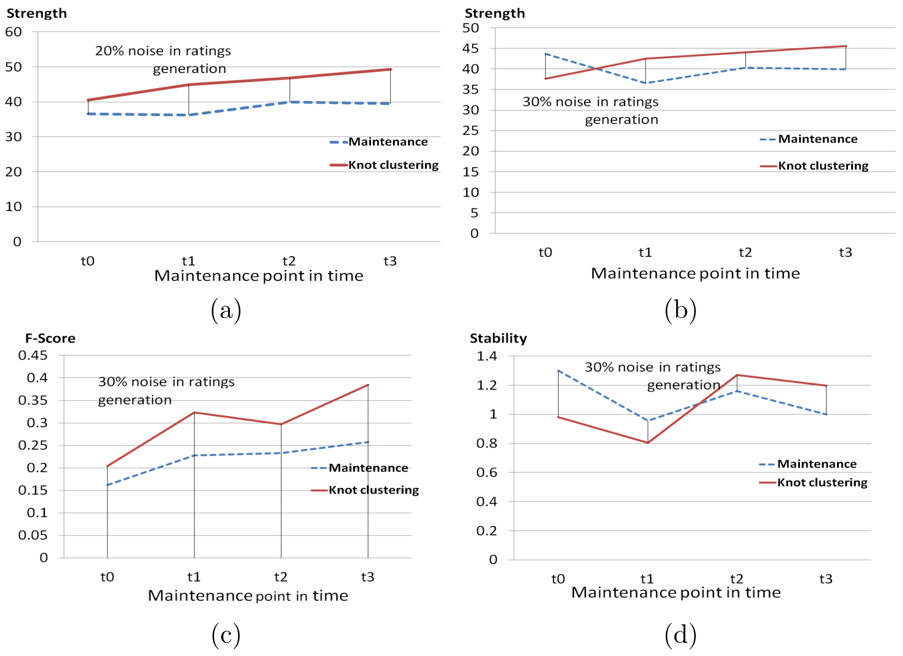

Fig. 1. Performances of maintenance algorithm in different noise levels.

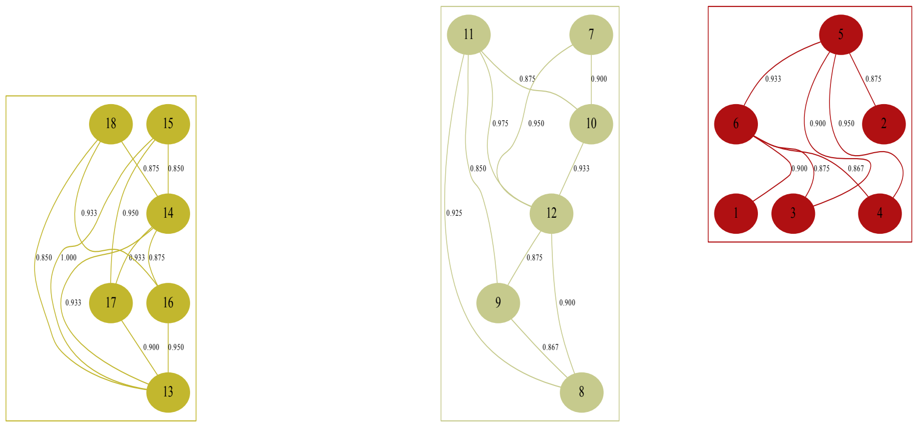
Fig. 2. Knot structure before a slander attack

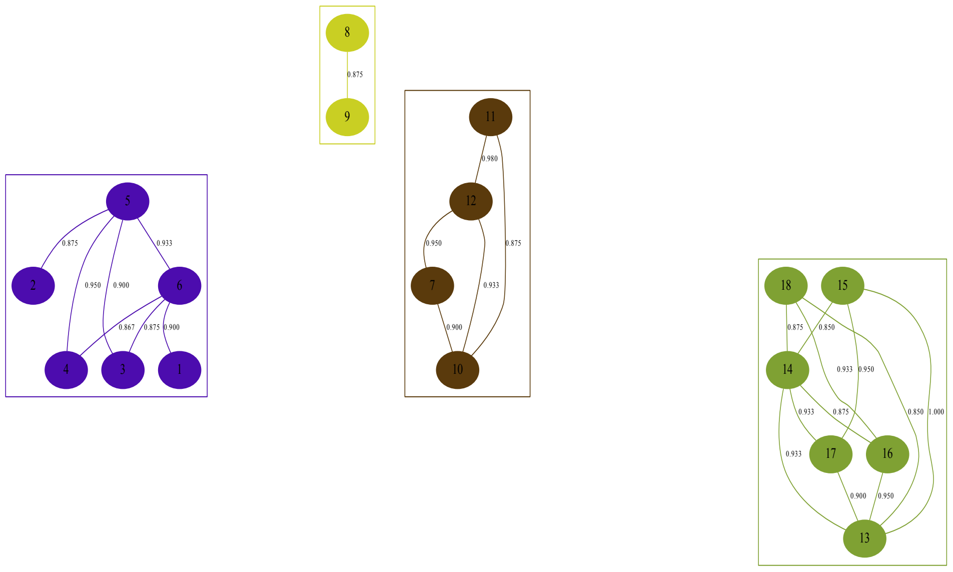
Fig. 3. Knot structure after a slander attack and a maintenance operation

Fig. 1 demonstrates the results of the experiment by comparing the performances of the maintenance algorithm and the knot clustering algorithm at different phases of maintenance time and for different noise levels. In each panel of Fig. 1, the x-axis represents the points in time at which maintenance operation was conducted. The two lines represent the quality of the result clustering as obtained by the maintenance algorithm and by the knot algorithm in terms of strength, stability, and F-score. Fig. 1(a) depicts strength results under conditions of 20% noise. As expected, the strength of the clustering by the maintenance algorithm is lower than that by knot clustering. However, the difference between the two algorithms in the first maintenance event is relatively small. Moreover, the results indicate that the difference becomes much smaller for higher levels of noise, as shown by the better performance of the maintenance algorithm in the first maintenance event at a 30% noise level (Fig. 1(b)). This can be explained by the nature of the first maintenance event, in which the effects of changes are less global. From the second maintenance phase, the strength of the clustering knots created by the maintenance algorithm is, as expected, lower than that of the clustering created by knot clustering. Fig. 1(d), presents the stability results of the same experiment with 30% noise. Although the stability of the clustering created by the maintenance algorithm is better in the first and second maintenance phases than that of the clustering created by knot clustering, that changes over time. The reason for that outcome is also explained by the local changes that are being applied by the maintenance algorithm to preserve stability. The F-Score, which is defined as the harmonic mean of precision and recall, examines the quality of the resulting clustering as a classier (and where the error is defined as a node belonging to a different cluster than its original cluster.) All the experiments show that the results of the knot algorithm are better than those of the maintenance algorithm, but a comparison of error shows that relative to the knot algorithm, that of the maintenance algorithm is less than 0.15. These results show that the maintenance algorithm, which is more efficient in terms of computational overhead, could replace the knot clustering algorithm for early maintenance events, especially to preserve the original knot structure .

Our next experiment simulates the attack scenario. We evaluate algorithm performance in the event of a slander- attack aimed to decrease the reputation of an expert in a virtual community network. In slander attacks, one or more members falsely give low ratings to an expert. The effect of a single slandering member is minor, especially if the system limits the rate at which negative ratings can be produced. However, slander attacks may become serious if they involve the collusion of several members. Our goal was to demonstrate that the maintenance algorithm can be used as a defense technique that identies malicious members in an effort to prevent the false ratings of a slander- attack. To simulate the attack scenario, we rst apply the initial clustering, and for the next phase of maintenance we use a set of low ratings of an expert given by fraudulent members of the same knot vs. other the opinions of other members, who gave the same expert high ratings. Our maintenance mechanism demonstrates a high sensitivity to false ratings and reacts accordingly by removing the fraudulent attackers from the knot. This is clearly shown in 2 and 3, in which slandering users are moved from their original knot and inserted into their own, new knot.

6 Concluding remarks

We have introduced a clustering maintenance algorithm for a dynamic trust network based on the properties of knot clustering. The algorithm is edge-centric and works in response to changes that occur in the trust relations within the community over time. The key motivation for presenting this algorithm is to provide a good alternative to the expensive process of knot clustering that is run in response to new trust information obtained within a community. The simulation based experiment confirms our expectations and demonstrates the effectiveness of the algorithm. In future work we intend to further investigate the optimal points in time for triggering the maintenance procedure. In addition, we intend to extend our evaluation and further examine different and more sophisticated attack scenarios.

References

1. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems **43**(2) (2007) 618–644
2. Gal-Oz, N., Gudes, E., Hendler, D.: A robust and knot-aware trust-based reputation model. In: Proceedings of the 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08), Trondheim, Norway (June 2008) 167–182
3. Gal-Oz, N., Yahalom, R., Gudes, E.: Identifying knots of trust in virtual communities. In Wakeman, I., Gudes, E., Jensen, C.D., Crampton, J., eds.: IFIPTM. Volume 358 of IFIP Advances in Information and Communication Technology., Springer (2011) 67–81
4. Lo, V.M., Zappala, D., Zhou, D., Liu, Y., Zhao, S.: Cluster computing on the fly: P2p scheduling of idle cycles in the internet. In Voelker, G.M., Shenker, S., eds.: IPTPS. Volume 3279 of Lecture Notes in Computer Science., Springer (2004) 227–236
5. M'hamdi, M.A., Bentahar, J.: Scheduling reputation maintenance in agent-based communities using game theory. JSW **7**(7) (2012) 1514–1523
6. Gopalan, S., Venkataraman, G., Emmanuel, S.: Fpga implementation and analyses of cluster maintenance algorithms in mobile ad-hoc networks. In Srikanthan, T., Xue, J., Chang, C.H., eds.: Asia-Pacific Computer Systems Architecture Conference. Volume 3740 of Lecture Notes in Computer Science., Springer (2005) 714–727
7. Wang, L., Olariu, S.: Cluster maintenance in mobile ad-hoc networks. Cluster Computing **8**(2-3) (2005) 111–118
8. Li, F., Zhang, S., Wang, X., Xue, X., Shen, H.: Vote-based clustering algorithm in mobile ad hoc networks. In: ICOIN. (2004) 13–23
9. Lin, C.R., Gerla, M.: Adaptive clustering for mobile wireless networks. IEEE Journal on Selected Areas in Communications **15**(7) (1997) 1265–1275
10. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. In: Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS'02), Washington, DC, USA, IEEE Computer Society (2002) 238–247
11. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. Journal of the ACM (JACM) **55**(5) (2008) 1–27
12. Kinateder, M., Baschny, E., Rothermel, K.: Towards a generic trust model - comparison of various trust update algorithms. In: iTrust. Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2005) 177–192
13. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: Proceedings of the 13th international conference on World Wide Web (WWW'04), ACM (May 2004) 403–412
14. Edachery, J., Sen, A., Brandenburg, F.J.: Graph clustering using distance-k cliques. In: Graph Drawing. (1999) 98–106
15. Ward, J., Hook, M. In: Application of an Hierarchical Grouping Procedure to a Problem of Grouping Profiles, vol. 23, no. 1. (1963) 69–82