Identifying Knots of Trust in Virtual Communities

Nurit Gal-Oz, Ran Yahalom, Ehud Gudes

Identifying Knots of Trust in Virtual Communities

Nurit Gal-Oz Ran Yahalom and Ehud Gudes

Deutsche Telekom Laboratories at Ben-Gurion University, Beer-Sheva, 84105, Israel,
galoz@cs.bgu.ac.il, yahalomr@cs.bgu.ac.il, ehud@cs.bgu.ac.il

Abstract. Knots of trust are groups of community members having overall "strong" trust relations between them. In previous work we introduced the knot aware trust based reputation model. According to this model, in order to provide a member with reputation information relative to her viewpoint, the system must identify the knot to which that member belongs and interpret its reputation data correctly. In the current paper we present the problem of identifying knots which is modeled as a graph clustering problem, where vertices correspond to individuals and edges describe trust relationships between them. We propose a new perspective for clustering that reflects the subjective idea of trust and the nature of the community. A class of weight functions is suggested for assigning edge weights and their impact on the stability and strength of knots is demonstrated. Finally we show the efficiency of knots of high quality for providing their members with relevant reputation information.

1 Introduction

Trust is itself a term for clustering of perceptions. (White, 1992)

Trust and reputation systems are considered key enablers of virtual communities, especially communities of strangers, where users are not required to reveal their real identities and use pseudonyms instead. These systems support the accumulation of member reputation information and leverage this information to increase the likelihood of successful member interactions and to better protect the community from fraudulent members.

As the scale of virtual communities continues to increase, they become more and more heterogeneous. This implies that, rather than being a single, homogeneous community, they become a collection of loosely-coupled *knots* (i.e. subcommunities) of users. A *knot* is defined as a group of community members having overall "strong" trust relations between themselves. Typically, members belonging to the same knot are more likely to have similar viewpoints and preferences as compared to members that belong to different knots.

The knot-aware trust-based reputation model, introduced in previous work [12], models virtual communities of strangers where members seek services or expert advice from other members. Two key examples of such communities are eBay [1] and Experts-Exchange [2]. The assumption underlying our knot-aware model is that "less is more": the use of relatively small, but carefully selected, subsets of the overall community's reputation data yields better results than those represented by the full data set.

Since members are primarily influenced by members that shared their preferences in the past, a useful feature of the knot model is that it naturally prevents malicious attempts to bias community members' decisions. Another advantage is that

smaller sub-communities, whose viewpoints differ from the overall community average, can maintain their distinctive preferences without having their opinions "diluted' by those of the majority of users outside their knot.

In this paper we focus on the task of partitioning the community into knots. We model the community as a graph where vertices correspond to members and edges describe direct trust relations between them and refer to this task as graph clustering. Specifically, we find the knot clustering task very close to the optimization problem known as correlation clustering [5] which aims at obtaining clusters based on pairwise node relations without specifying the number of clusters in advance. However, unlike the general problem of graph clustering, knot clustering is also motivated by several objectives which arise from the domain of virtual communities and the essence of trust knots.

First, a desirable goal is to group together vertices that are connected with high weighted edges while simultaneously avoiding the inclusion of low weighted edges within the same group. The inherent difficulty we have with this goal is that edge weights are derived from trust relations and are not a distance metric; therefore a person may have a great deal of trust in two other members who have very little trust between themselves.

Second, the length of the path between each pair of vertices should be restricted. A path length greater than one indicates a transitive trust chain that represents an indirect trust relation. The longer a chain is, the lower is the trust between its endpoint vertices. Transitive trust chains [15] are a means to overcome the sparsity problem from which community graphs representing trust relations may suffer. However, allowing a long trust chain may result in very large knots. As such, we may prefer to divide a big cluster into several smaller clusters (i.e., "less is more") in which the path between each pair of vertices is shorter.

Third, clusters should be stable. Intuitively a cluster is considered to be more stable as more modifications to its edges' weights are required to justify splitting it. Weight functions that were mentioned in the literature [11] refer to the same notion of correlation for all input graphs. Our research regards trust as correlation and we assert that the extent to which two individuals are correlated is relative to a required level of mutual trust and subject to its existence in the community. Thus we support different notions of correlation by using different weight functions for different community graphs.

Finally although clustering is a common technique in AI and data mining, in most clustering applications, the graph representation of the problem is an obvious step. In knot clustering this step is very significant. The graph representation, the weights on the edges and the distance functions all reflect the subjective idea of trust and the nature of the community to which our clustering is sensitive. This is shown in our experimental evaluation and is a major contribution of our paper.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work. In section 3, we describe the knots-aware clustering problem and in section 4 we provide the knot clustering algorithm. Evaluation results are presented in Section 5. We conclude by discussing future research directions in Section 6.

2 Related Work

One of the basic properties of trust is directness [17]. Direct trust refers to trust based on first hand experience. Indirect trust is based on the opinion of one's

trustees by transitivity. Several studies use transitive trust-chains to propagate trust. Instead of using trust propagation as in [6, 16, 14] we use clustering that leans on the transitivity property to make sure there exists a predefined level of propagated trust among a knot's members.

Given a data set in the form of a graph, the goal of graph clustering is to divide the set of vertices into clusters such that the vertices assigned to a particular cluster are similar or connected in some predefined sense. Commonly used clustering algorithms such as k-means, k-sum and k-center require prior knowledge of the number of clusters that we wish to divide the data into. However in some applications this information is unknown. The Correlation Clustering (CC) problem introduced by Bansal et al. [5] is a method for clustering a graph into the optimal number of clusters without knowing that number in advance. This problem is defined on a complete graph of n vertices (items), where each edge is labeled $< + >$ if its end vertices are considered similar or $< - >$ if they are considered different. The objective of CC is to produce a clustering that agrees as much as possible with the edge labels. This corresponds to the optimization problem of maximizing agreements or to its equivalent problem of minimizing the number of disagreements. The solution of the CC optimization problem is known to be NP-hard [5]. Integer linear programming (ILP) can be used to solve the general problem optimally, for a relatively small number of vertices. According to [11], beyond a few hundred vertices, the only available solutions are heuristic or approximate. Several effective approximation algorithms were proposed for CC with worst-case theoretical guarantees (e.g. [5, 9, 7]). Bansal et al. [5] provide an approximation algorithm for clustering by minimizing disagreements in complete graphs. They show that the number of disagreements in the solution found by the algorithm is bounded by a constant factor of the optimal solution. Demaine et al. [9] present an $O(logn)$ approximation algorithm for minimizing disagreements in general weighted graphs. This algorithm first solves a linear program and then uses the resulting fractional values to determine the distance between two vertices. They use a region-growing technique to group close vertices together and round the fractional values. Swamy [23] shows that the maximization problem is solvable within a factor of 0.7666 approximation. Correlation Clustering has applications in data mining and natural language processing [8], consensus clustering [13], co-reference resolution [22, 21, 8]. Elsner and Schudy [11] have examined four greedy algorithms First [22], Best [21] Vote and Pivot [4]. They used an implementation of the semidefinite programming (SDP) relaxation to provide lower bounds on the optimal solution and show that the heuristic algorithms are quite close to optimal.

The goal of limiting the cluster diameters is discussed in [10]. The authors present a heuristic algorithm for graph clustering using distance-k cliques. A sub-graph is a distance-k clique if any two vertices in it are connected by a path of length k or less. This study also considers the goal of limiting the clusters' diameter as well as other goals which in-

volve using the edge weights as additional criteria for assigning vertices to clusters.

3 Applying Clustering for Identifying Knots

A *knot* [12] is a subset of community members identified as having overall strong trust relations among them. A trust member i has in member j is derived from a trust computation model(e.g., the knot model [12]) or from directly assigned trust values [18]. Two members i and j should belong to the same knot if i has high enough direct trust in j denoted $TM(i,j)$, or if i has high enough transitive trust in j (e.g., if i trusts k and k trusts j we conclude that i trusts j), and vice versa. Knots are groups of members that can rely on each other's recommendations even if they did not rate the same experts. Different knots typically represent different view points and preferences. It is therefore plausible that the reputation of the same expert may differ significantly between different knots. Using the knot-aware approach, we can deal with heterogeneous communities where an experts reputation may be distributed in a multi-modal manner. As discussed in [12], knots have the ability of reducing the risk of relying on dishonest or biased recommendations, since the members that provide them can be identified and excluded from the knot.
A community is modeled as a directed graph $G = (V,E)$ (called the community graph), in which vertices represent members and edges represent the trust relations between the members at their end-point vertices. The weight on a directed edge from vertex i to vertex j is the level of direct trust i has in j at time t and is computed by $TM^t(i,j)$. Since we deal with the state of the graph at time t, for simplicity, we omit the time indicator. We refer to the task of identifying knots as graph clustering. More specifically, we aim to find a partition of the community graph based on the direct trust between pairs of members. For this purpose, we replace the trust relations between any two members $TM(i,j)$ and $TM(j,i)$ with a weaker relation named Mutual Trust in Member (MTM). Thus, the directed edges (i,j) and (j,i), whose weights were $TM(i,j)$ and $TM(j,i)$, are replaced by a single, undirected edge whose weight is $MTM(i,j) = MTM(j,i) = min\{TM(i,j), TM(j,i)\}$. This way we can use the edge relation as the input for the clustering algorithm, which must decide if its two end-vertices should reside in the same cluster or not. Intuitively, the new relation is more stringent in the sense that it takes into account the minimum level of mutual trust between any two members as the representing value of trust between them.

3.1 Correlation Clustering

We consider the problem of clustering a community of members based on the mutual trust they have in each other. Each cluster in the resulting clustering will constitute a different knot. For this purpose, we adopt

the correlation clustering (CC) approach defined by [5]. Given a graph $G_{CC} =< V, E_{CC} >$, each edge $e_{ij} \in E_{CC}$ is either labeled $< + >$ if we believe that i and j should belong to the same cluster or $< - >$ if we believe that they should not. In addition, the edge is assigned a weight w_{ij} that quantifies our belief. The process of assigning of the label and weight to e_{ij} is based on $MTM(i, j)$ and controlled by the weight function discussed in the next sections.

A partition of V is defined by a set of variables $x_{ij} \in \{0, 1\}$ corresponding to the set of edges $e_{ij} \in E_{CC}$. The assignment of vertices i and j to the same cluster is expressed by assigning $x_{ij} = 1$, and assigning them to different clusters is expressed by assigning $x_{ij} = 0$.

We search for a partition that agrees as much as possible with the edge labels. An agreement with a label refers to either assigning a positive edge within a cluster or assigning a negative edge between clusters. Our goal is therefore to maximize the amount of agreement (known as the maximization version of CC in incomplete graphs). Following [7], we thus define the objective function as follows:

$$Maximize(\sum_{e_{ij} \in E+} w_{ij} \cdot x_{ij} + \sum_{e_{ij} \in E-} w_{ij} \cdot (1 - x_{ij})) \quad (1)$$

where $E+$ and $E-$ denote all positive and negative labeled edges, respectively, and subject to: $x_{ij} \in \{0, 1\}$; $x_{ij} = x_{ji}$; $x_{ii} = 1$ and $x_{ij} = x_{jk} = 1$ implies $x_{ik} = 1$. This objective function is referred to as the MaxAgree objective.

3.2 Clustering Criteria

A clustering algorithm aimed at achieving the MaxAgree objective attempts to assign edges with high values of MTM within knots while keeping edges with low MTM values outside knots. However, we also require our clustering to meet three other objectives related to the essence of knots, which we to address by fine tuning the weight function and clustering algorithm. The first objective is to create strong knots, or in other words, to construct clusters having a large aggregated amount of MTM. Although this may seem to derive from the MaxAgree objective, it emphasizes the need to have as many high MTM edges and as few low MTM edges within knots as possible. Our second objective is ensuring that the indirect trust relations between any two members in any knot meets some minimal level of reliability, thereby increasing knot efficiency. This reliability depends highly on the trust chain of the clustering (definition 1). The longest trust chain that exists between any two vertices in a knot, known as the diameter of the subgraph denoted by the knot, characterizes the connectivity of the knot.

Definition 1. *A* Trust Chain Length *(TCL) of a clustering C, denoted by* κ*, is the length of the longest trust chain connecting any two vertices*

within any knot in the clustering. Formally:

$$\kappa = \max_{K \in C} \max_{i,j \in V_K} TC_K(i,j) \tag{2}$$

where $|TC_K(i,j)|$ is the length of the trust chain between nodes i and j in knot K.

A path of length greater than one is a transitive trust chain which represents an indirect trust relation. The longer a chain is, the lower we rely on the trust between its endpoint vertices, regardless of the actual trust level assigned to each edge on the path. Assuming that the reliability of indirect trust between members in the same knot decreases as the trust chain between them becomes longer, limiting the TCL of a clustering can be used as a mechanism of ensuring that the indirect trust relations between any two members in any knot will meet some minimal level of reliability. Finally, we want to generate stable knots. Since trust relations are constantly updated, we need the clustering to be firm enough so that no single trust modification will turn it incorrect, which is important for practically maintaining knots. Next we formally define trust chain length and the measures of strength and stability.
The strength of a clustering is defined in terms of the strength of its clusters (see definition 2 for knot strength). For consistency with the clustering graph, instead of using $TM(i,j)+TM(j,i)$ as in definition 2, we use $2 \cdot MTM(i,j)$.

Definition 2. Strength *of a clustering C is the sum of the strength of all the knots in the clustering, $K =< V_K, E_K > \in C$.*

$$Strength(C) = \sum_{K \in C} \frac{\sum_{i \in V_K} deg_i}{|V_K|} = \sum_{K \in C} \frac{2 \cdot \sum_{e_{ij} \in E_K} MTM(i,j)}{|V_K|} \tag{3}$$

Intuitively, as the average node degree increases and the knot becomes stronger, it has a better edges-to-vertices ratio and more paths between vertices. This indicates that the members of the knot have a lot of mutual "trustees," and therefore, they are more likely to trust each other.
Stability of a clustering *C* is calculated as the average stability of its knots. The stability of a knot $K =< V_K, E_K >$ 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 of intra-knot or increase of inter-knot edge MTM values) must occur to justify a split. Furthermore, we require that knot stability indicate the consequence of the *MinCut* split. The closer the sizes of the two sub-knots, the greater the affect on the knot's structure, and therefore, the knot is considered less stable. Thus we define stability as follows:

Definition 3. Stability *of a knot is the weight of the minimum cut on edges relative to the ratio between the size of the sub-knots derived from this cut.*

$$Stability(K) = \frac{MinCut_K \cdot \frac{|K'|}{|K''|}}{|V_K| - 1} \tag{4}$$

where K' and K'' are the two sub-knots induced by the removal of the minimum cut edges and $|V_{K'}| \geq |V_{K''}|$.

Stability of a clustering is calculated accordingly:

$$Stability(C) = \frac{\sum_{K \in C} Stability(K)}{|C|} \tag{5}$$

3.3 Different Weight Functions

An integral element of the CC graph generation is the weight function (WF). The WF provides a pairwise decision of whether or not two members should be assigned to the same knot. If the sign of the WF output is positive it means the two members should be assigned to the same knot, otherwise they should not. The WF also provides the extent to which the decision is believed to be true (the confidence in the decision), and corresponds to the weight of the edge in the CC graph. The WF output is calculated from the MTM between the two members while taking into consideration a community dependent trust threshold level (definition 4):

Definition 4. Trust Threshold Level *(TTL) is a value in* $[0.5, 1]$*,denoted by* α*, which represents the minimum level of MTM required for an edge to be labeled* $< + >$ *. The TTL is a community dependent parameter. It is defined in the range of* $[0.5, 1]$ *since trust in our model ranges in* $[0, 1]$ *where complete trust is set to 1, and complete distrust is set to 0.*

The WF is formally defined in definition 5:

Definition 5. $WF : [0,1] \times [0.5,1] \rightarrow \Re$ *is a function that assigns the weight* $w_{ij} = |WF(MTM(i,j),\alpha)|$ *for edge* e_{ij} *and labels it with* $sign(WF(MTM(i,j),\alpha))$.

A WF is required to have the following two properties:

1. It must be be monotonically non-decreasing to give higher MTM valued pairs a higher tendency of being assigned to the same knot.
2. The MTM value for which the WF switches its sign, denoted as $MTM_{boundary}$, must be in the range $[0.5, \alpha]$, where $\alpha > 0.5$. This is necessary to reflect our assumptions that, for any given pair of members, if $MTM < 0.5$, they **do not** trust each other enough to be assigned to the same knot whereas if $MTM > \alpha$, they **do**.

A key aspect of the WF is its slope, which controls controlling how sensitive its labeling is to changes in MTM values. Defining different slopes for different intervals of MTM results in different levels of labeling sensitivity between those intervals. Another key aspect is whether its output weight values are symmetric with respect to $MTM_{boundary}$. One may

choose to define a symmetric WF for which the MTM values at equal distances from $MTM_{boundary}$ derive the same weight but with opposite signs. However, if the WF is asymmetric, a value of $MTM = MTM_{boundary} + \varepsilon$ corresponds to a positive output whose magnitude differs from the negative output corresponding to a value of $MTM = MTM_{boundary} - \varepsilon$. An asymmetric WF allows us to distinguish between the significance of positive and negative edges (and therefore, to detect the effect that they have on the clustering algorithm). For example, by defining the slope after $MTM_{boundary}$ to be steeper than the slope before it, one can express that $MTM = MTM_{boundary} + \varepsilon$ corresponds to a heavier positive edge when compared to the weight of the negative edge corresponding to $MTM = MTM_{boundary} - \varepsilon$. A basic weight function simply considers the difference between an edge's MTM value and α:

$$WF_{basic}(MTM(i,j),\alpha) = MTM(i,j) - \alpha \tag{6}$$

This WF is symmetric and satisfies $MTM_{boundary} = \alpha$: an MTM value of $\alpha + \varepsilon$ for $0 < \varepsilon < min(\alpha, 1-\alpha)$ is "good" to the same extent that an MTM value of $\alpha - \varepsilon$ is considered "bad". An asymmetric growth WF involves a parameter $\lambda \geq 0$ which allows us to regulate both the value of $MTM_{boundary}$ and the slope:

$$WF_{asymGrowth}(MTM(i,j),\alpha) = \frac{\lambda}{1 + e^{(\alpha - MTM(i,j)))\cdot 10}} - (\alpha - MTM(i,j)) \tag{7}$$

This WF is identical to WF_{basic} for $\lambda = 0$. Notice that $0 \leq MTM_{boundary} \leq \alpha$. As λ increases, $MTM_{boundary}$ becomes smaller and positive edges receive much higher weights than negative ones.

The ability to use different values of TTL and/or different WFs is cardinal for knot identification. This ability is essential for accommodating different views of how it is best to determine whether or not two members should be in the same knot. This is important not only for working with different communities but also for dynamic communities where the perception of trust may change over time as more rating data is accumulated.

Figure 1 presents an example for this. The central graph represents a toy community of whose knots were identified using WF_{basic} with $\alpha = 0.7$, representing the perceived TTL at time t_0. The strength of this clustering is 4 and its stability is 2. Now assume that a maintenance reclustering is done every T days. Consider a scenario s_1 in which members 3 and 4 rated the same experts similarly during the T days after t_0 causing $MTM(3,4)$ to increase by 0.3 (upper part of Figure 1). Reclustering the community with the same WF and α, results in a weaker (strength = 2.27) and less stable (stability = 0.16) clustering (upper-left clustering). However, if we realize that the community has changed in a way that requires us to raise the TTL (e.g. there is no point in having a TTL

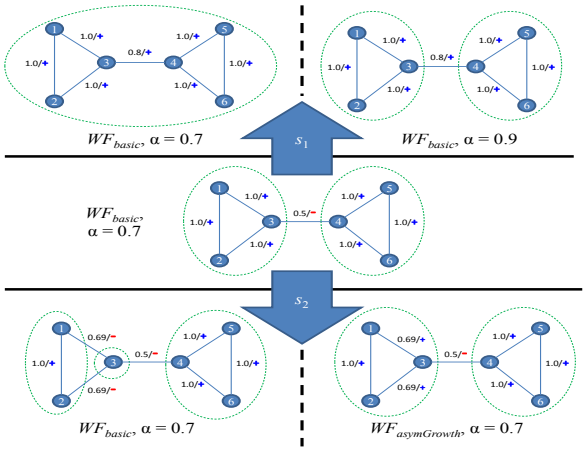

Fig. 1. Motivation for TTL/WF adjustment in dynamic communities. All graphs represent the same toy community where the central graph represents the community at time t_0 and all other graphs at time $t_0 + T$. The upper two graphs depict the changes in the graph due to new ratings that were introduced according to scenario s_1 and the lower two due to a different scenario, s_2.

that is lower than all MTM values in the community), we could recluster using WF_{basic}, with say $\alpha = 0.9$, and get the stronger and more stable clustering we had at t_0. In a different scenario s_2, members 1 and 2 rated the same experts as member 3 only differently (again, between t_0 and $t_0 + T$), causing both $MTM(1,3)$ and $MTM(2,3)$ to decrease by 0.31 (lower part of Figure 1). Reclustering the community with the same WF and α will result in a weaker (strength = 3) and less stable (stability = 1) clustering (lower-left clustering). This may be acceptable in communities where we would be willing to sacrifice stability on account of gaining accuracy. However, if 0.7 adequately represents the TTL and the community is more interested in stability, we could make this threshold tolerant to MTM values in its vicinity. One way to do this is by switching to an appropriate $WF_{asymGrowth}$. Reclustering would then result in a stronger (strength = 3.59) and more stable (stability = 1.69) clustering (lower-right clustering).

4 The Knot Clustering Algorithm

The knot clustering algorithm uses the hierarchical approach [24] as a feasible solution to the CC problem defined in section 3. First, the CC graph $G_{CC} =< V, E_{CC} >$ is derived from the community graph G using the given WF and TTL. Next, we calculate the connectivity components of the graph induced by the positive edges, denoted by G_C^+, instead of G_C. For each connectivity component with n vertices, we initialize its

clustering to n singleton clusters, and iteratively merge pairs of clusters until a stopping criterion is met, and the clustering is final. The stopping criterion is derived from the TCL requirement and the MaxAgree objective, the latter of which is used to guide the merging process. More specifically, at each step, the pair of clusters whose merging leads to the highest increase in the value of the MaxAgree objective function, without violating the TCL requirement, are merged.

A clustering C can be defined by its corresponding 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. For a given clustering C, the assignment of its clustering matrix in the MaxAgree objective function (eq. 1)can be written as:

$$Agreement(C) = \sum_{e_{ij} \in E^+, x_{ij}=1} w_{ij} \quad + \sum_{e_{ij} \in E^-, x_{ij}=0} w_{ij} \tag{8}$$

where $x_{ij} \in M^C$. The Agreement function expresses the amount of weighted agreement associated with a clustering by summing the weights of all positive intra-cluster edges and negative inter-cluster (bridge) edges. To quantify the contribution of merging clusters c_1 and c_2, we define the utility of merging as the increase in the Agreement function resulting from the merging denoted $MergeUtil$: $MergeUtil(c_1, c_2) = Agreement(C') - Agreement(C)$, where C' is the clustering derived from clustering C by merging clusters $c_1, c_2 \in C$ into a single new cluster $c'_{12} \in C'$. Inserting equation 8 into this definition gives:

$$MergeUtil(c_1, c_2) = \sum_{e_{ij} \in Bridge^+_{c_1,c_2}} w_{ij} \quad - \sum_{e_{ij} \in Bridge^-_{c_1,c_2}} w_{ij} \tag{9}$$

where $Bridge^+_{c_1,c_2}$ and $Bridge^-_{c_1,c_2}$ are respectively the sets of positive and negative bridge edges between clusters c_1 and c_2. Intuitively, the only edges that can affect the value of the Agreement function due to a merging of c_1 and c_2 are the bridge edges between them, which become intra-cluster edges in c'_1. Any such positive edge adds to the overall agreement in the clustering, and therefore increases the value of the Agreement function. On the other hand, any such negative edge no longer contributes what it contributed when it was a bridge edge, thereby decreasing the value of the Agreement function. Thus, the MergeUtil can be computed by iterating over all bridge edges between c_1 and c_2 while adding the weights of the positive ones and subtracting the weights of negative ones. Algorithm 1 outlines the knot clustering algorithm. Lines 1-3 construct the clustering graph with the appropriate weights and separate this graph into connectivity components. Lines 5-14 perform the clustering for each connectivity component. In line 5, we initialize the clustering for the current connectivity component by creating a singleton cluster for each vertex in the graph. In line 6, we calculate the MergeUtil values of all cluster

Algorithm 1 ClusterGraph(G, κ, WF, α)

1: $G_{CC} \leftarrow < V, E_{CC} >$ s.t. $E_{CC} = WF(E, \alpha)$;
2: $C \leftarrow \emptyset$
3: $ConComps \leftarrow \{C_{comp} | C_{comp} \subset G_{cc} \wedge \forall e_{ij}$ s.t. $i \in C_{comp}, j \notin C_{comp} \rightarrow eij \in E^{-}_{CC}\}$;
4: **for all** $comp \in ConComps$ **do**
5: $C_{comp} \leftarrow \{c_i | c_i = \{i\}, \forall i \in V_{comp}\}$;
6: $S \leftarrow \{(c_i, c_j) | MergeUtil(c_i, c_j) > 0; c_i, c_j \in C_{comp}\}$;
7: **while** $S \neq \emptyset$ **do**
8: $c_{ij} \leftarrow c_i \cup c_j$ s.t. $Max_{(c_i,c_j) \in S} MergeUtil(c_i, c_j)$;
9: **if** $\forall u, v \in V_{c_{ij}} : |TC_{c_{ij}}(u, v)| \leq \kappa$ **then**
10: $C_{comp} \leftarrow C_{comp} - \{c_i, c_j\}$;
11: $C_{comp} \leftarrow C_{comp} \cup c_{ij}$;
12: $S \leftarrow \{(c_r, c_s) | MergeUtil(c_r, c_s) > 0; c_r, c_s \in C_{comp}\}$;
13: **else**
14: $S \leftarrow S - \{(c_i, c_j)\}$;
15: **end if**
16: **end while**
17: $C \leftarrow C \cup C_{comp}$
18: **end for**

pairs and generate the list S. This list contains references to the cluster pairs (c_i, c_j) whose merging into cluster c_{ij} can increase the value of the Agreement function, i.e., the MergeUtil value of this pair is positive. In lines 8-9, the best candidate pair of clusters is checked for TCL-compliance. If found compliant, then each of the clusters in the pair is removed from the clustering and the candidate merged cluster is added to the clustering in lines 10-11. Any MergeUtil values that may have consequently changed are recalculated and S is repopulated with the positive MergeUtil valued cluster pairs. If the candidate pair for merge is not TCL-compliant, the pair is removed from the list S (line14). The algorithm terminates when the list S is empty. Termination is guaranteed since the size of S is reduced in each step of the loop.

5 Clustering Evaluation

We evaluate the knot clustering in light of the objectives discussed in section 3. The evaluation is divided into two parts. First, we evaluate the quality of the clustering on a synthetic dataset produced by a simulation program in different settings. Then we evaluate the quality of the knot as a group of trustees by testing the reputation computation based on the knots we identify in the MovieLens [3] community.

The evaluation of the clustering requires a set of community graphs as input. The structure of a community graph depends on the existence of trust relations between the members and the level of trust they represent. In the early days of a community, its graph is necessarily very sparse, increasing in density as more experience is gained within the community. Moreover, the extent of member nodes partitioning in a graph may

vary from a clearly structured graph to a completely uniform one with the edges evenly distributed over the set of vertices. In the latter case, the clustering computed by any algorithm will be rather arbitrary. If the graph is clearly structured and a clear clustering based on the trust levels exists, our task is to identify it. We may further divide this structure by restricting the TCL if the graph is highly connected. However, a more challenging task is to identify the best set of knots, when the data set is noisy. We refer to two types of noise – graph sparsity and a lack of a clear structure of groups of members that trust each other. We constructed graphs that simulate communities with different levels of density and cluster-structure clarity. Assuming that each community is characterized by some TTL value, referred to as the characteristic TTL and denoted by $\widehat{\alpha}$, we also generate graphs with different values of $\widehat{\alpha}$. The cluster-structure was created by first defining groups of members that correspond to knots and then generating their MTM values accordingly, with respect to $\widehat{\alpha}$. Different levels of cluster-structure clarity were introduced by generating MTM values that agree with the pre-defined knots with different values of probability p (i.e. there is a $p\%$ chance that the MTM will be $\geq \widehat{\alpha}$). The purpose of this experiment is to demonstrate how we overcome the two types of noise by using our approach. The second measure is the Mathews Correlation Coefficient [19] which is generally regarded as a balanced measure that can be used for comparing clusterings of very different sizes. The third measure we used is variation of information (VI), suggested by Meila [20], which is based on using entropy and measures the amount of information lost and gained in changing from one clustering to another.

5.1 Evaluation Results

Our experiment includes 1200 tests in which we tested values of TTL ranging from 0.5 to 0.95 (with a 0.05 increment); TCL values ranging from 1 to 6; density levels of $5-15\%, 16-25\%$, and $26-35\%$; levels of cluster-structure clarity represented by probabilities ranging from 0.6 to 1 (with increments of 0.1); and $\widehat{\alpha}$ values of 0.6, 0.7, 0.8, and 0.9.

As expected, the best results were obtained when we used a weight function with $\widehat{\alpha}$. This was right even when the probability for a clear structure was decreased to 0.6 and as the graph density was as low as 9%. Table 1 shows the average improvement in clustering gained by using WF_{basic} with $TTL = \widehat{\alpha}$ in our algorithm instead of using other TTL values. The first column represents the $\widehat{\alpha}$ examined and each of the other three columns depicts the average improvement (in percents) according to one of the three measures F-score, Matthews-CC, or VI, respectively, when using the $\widehat{\alpha}$ as the TTL instead of the rest 3 TTL values.

We show that in dense communities, different values of TCL have little benefit on the quality of clustering whereas in sparse communities the

difference is significant and a higher TCL is required to overcome sparsity. Figure 2 demonstrates that a TCL of 4 significantly improves the quality of clustering when the density of the community is low compared to lower TCL values. As the density increases, the improvement becomes less significant. The clustering quality is measured here by VI (VI is best as it tends to 0) but similar results were also obtained for the F-score and Mathews Correlation Coefficient measures.

$\widehat{\alpha}$	F-Score	Matthews-CC	VI
0.6	75.6	55.1	3.3
0.7	49.2	49.0	2.6
0.8	25.5	40.9	3.5
0.9	11.3	40.8	4.9

Table 1. Clustering improvement (%) when using $\widehat{\alpha}$, as measured by F-score, Matthews-CC or VI.

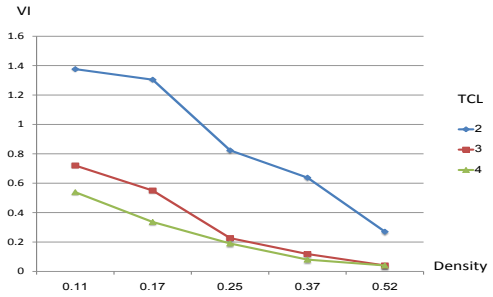

Fig. 2. Quality of clustering measured by VI in different levels of sparsity of the same graph.

Next we show the tradeoff between strength and stability. We compared the results from using $WF_{AsymGrowth}$ with $\lambda = 1$ to using WF_{Basic} to demonstrate their impact on stability and strength. When comparing different levels of TTL (Figure 3(a) and (b)) we can see that $WF_{AsymGrowth}$ yields better stability for all levels of TTL. However, this advantage is significantly reduced for high levels of TTL where the knots obtained where relatively small for both WFs and the clustering tends to contain many singleton clusters. WF_{Basic} yields better strength in general with the exception of high levels of TTL in which the clusters obtained by the $WF_{AsymGrowth}$ consisted of more edges whose MTM was less than the TTL. Figure 3(c), shows that the $WF_{AsymGrowth}$ yields better stability for all values of TCL, although for low TCL values the difference between the two WFs is relatively minor due to high connectivity of the knots. WF_{Basic} yields better

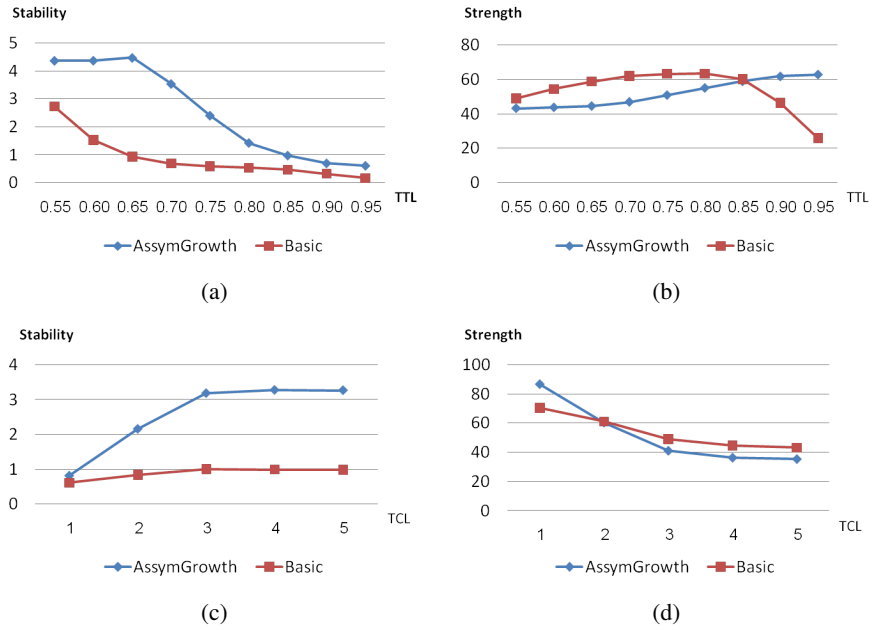

Fig. 3. Average Strength and Stability for different WFs: (a) Stability by TTL, (b) Strength by TTL, (c) Stability by TCL, (d) Strength by TCL.

strength in general but for a TCL value of 1, which assigns only connected members to the same knot, the $WF_{AsymGrowth}$ is stronger. This is because $WF_{AsymGrowth}$ is more tolerant to edges with an MTM value lower than α [Figure 3(d)].

In the second experiment we used a MovieLens [3] dataset to evaluate the quality of the knots. This dataset consists of 100,000 movie ratings submitted from 943 users on 1682 movies. Movies play the role of experts and the reputation of an expert is replaced by a movie score. In this case our criteria was how well a movie's reputation within a knot represents what knot members may think of it (in terms of predicting how they may rate the movie).

We divided the complete set of ratings into a training set, from which MTM values were derived and the community graph was constructed, and a test set which we kept aside for later evaluation. In this experiment we used a 4-fold cross validation. After identifying the knots in the community graph, we calculated the mean absolute error (MAE) between the knot based reputation scores and their corresponding test set rating scores. The majority of ratings in the MovieLens (over 60%) are of ratings 3 and 4 and therefore predicting these ratings according to any

popularity measure such as average, will produce good results. Since low ratings (1,2) and high rating (5) are relatively rare in the dataset, our goal was to show that by using the knots as a group of trustees we can provide a good prediction for these ratings where using the popularity measure would be less precise. We compared two different configurations of the knot clustering algorithm, both conducted with the $WF_{AsymGrowth}$ which is more suitable for a movies rating community. We used a TTL value of 0.9 and TCL values of 2 and 5 respectively. The results showed that in general the reputation scores provided to members by their knots were better than the global reputation [12] computed based on all knots which are not singletons. Table 2 presents the advantage of using knot based reputation over the global reputation for each score separately. It presents the improvement in percentage of the MAE of the global reputation. As shown, the improvement was stressed for the extreme scores while for values of 3 and 4 knots had no advantage. This can be attributed to the distribution of ratings as noted above. Clustering with TCL of 2 and 5 showed no significant advantage of one over the other.

Rating	TCL=2 TTL=0.9	TCL=5 TTL=0.9
1	5.16	4.46
2	3.1	3.12
3	-0.32	0.74
4	-1.13	-0.89
5	2.17	1.7

Table 2. The improvement (%) in MAE of reputation scores provided by knots compared to global reputation

6 Conclusions and Future Work

The community graph represents a dynamic trust network that changes all the time. We define the problem of identifying knots of members in the graph and propose a new perspective for clustering the community graph that refers to the underlying levels of trust existing in the community at a given time. The knot clustering problem is close to the correlation clustering problem with an additional limitation on the trust chain length. We suggest a heuristic algorithm related to the agglomerative hierarchical clustering that uses different weight functions to cluster different graphs. We show that the best solution that can be achieved with our approach, strongly depends on understanding the state of the community. This understanding allows us to adjust the TTL and weight function to meet the required objectives such as strength and stability of knots.

In future work we intend to further explore the subject of knot stability for the purpose of maintaining knots. *Knots Maintenance* is an action

taken in order to refine the clustering upon changes in the community. Refinement corresponds to either restoring the quality of a clustering which has decreased or improving the quality of a clustering when possible. Our goal is to carry out maintenance actions only when there is high probability that a better clustering exists and/or in accordance with a community policy.

References

1. eBay, http://www.ebay.com/.
2. Experts-exchange, http://www.experts-exchange.com/.
3. Grouplens, http://www.grouplens.org/.
4. N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008.
5. N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS'02)*, pages 238–247, Washington, DC, USA, 2002. IEEE Computer Society.
6. S. Chakraborty and I. Ray. Trustbac: integrating trust relationships into the rbac model for access control in open systems. In *Proceedings of the 11th ACM symposium on Access control models and technologies (SACMAT '06)*, pages 49–58, New York, NY, USA, 2006. ACM.
7. M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.
8. W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480, 2002.
9. E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2–3):172–187, September 2006. Special issue on approximation and online algorithms.
10. J. Edachery, A. Sen, and F. J. Brandenburg. Graph clustering using distance-k cliques. In *Graph Drawing*, pages 98–106, 1999.
11. M. Elsner and W. Schudy. Bounding and comparing methods for correlation clustering beyond ilp. In *NAACL-HLT Workshop on Integer Linear Programming for Natural Language Processing (ILPNLP'09)*, pages 19–27, 2009.
12. N. Gal-Oz, E. Gudes, and D. Hendler. 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)*, pages 167–182, Trondheim, Norway, June 2008.
13. A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):4, 2007.
14. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, pages 403–412. ACM, May 2004.
15. A. Jøsang, E. Gray, and M. Kinateder. Analysing topologies of transitive trust. In *Proceedings of the 1st International Workshop on Formal Aspects in Security and Trust (FAST'03)*, pages 9–22, 2003.

16. S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web(WWW2003)*, pages 640–651. ACM, 2003.
17. M. Kinateder, E. Baschny, and K. Rothermel. Towards a generic trust model - comparison of various trust update algorithms. In *iTrust*, Lecture Notes in Computer Science, pages 177–192. Springer Berlin / Heidelberg, 2005.
18. P. Massa and P. Avesani. Controversial users demand local trust metrics: An experimental study on epinions. com community. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'05)*, volume 20, page 121, 2005.
19. B.W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.
20. M. Meilă. Comparing clusterings–an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.
21. Vincent Ng and Claire Gardent. Improving machine learning approaches to coreference resolution. In *ACL*, pages 104–111, 2002.
22. W.M. Soon, H.T. Ng, and D.C.Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.
23. C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 526–527. Society for Industrial and Applied Mathematics, 2004.
24. J.H. Ward and M.E. Hook. *Application of an Hierarchical Grouping Procedure to a Problem of Grouping Profiles, vol. 23, no. 1*, pages 69–82. 1963.