Noname manuscript No.
(will be inserted by the editor)

Generalized Framework for Personalized Recommendations in Agent Networks

Chung-Wei Hang · Munindar P. Singh

Received: date / Accepted: date

Abstract An agent network can be modeled as a directed weighted graph whose vertices represent agents and edges represent a trust relationship between the agents. This paper proposes a new recommendation approach, dubbed LocPat, which can recommend trustworthy agents to a requester in an agent network. We relate the recommendation problem to the graph similarity problem, and define the similarity measurement as a mutually reinforcing relation. We understand an agent as querying an agent network to which it belongs to generate personalized recommendations. We formulate a query into an agent network as a *structure graph* applied in a personalized manner that reflects the pattern of relationships centered on the requesting agent. We use this pattern as a basis for recommending an agent or object (a vertex in the graph). By calculating the vertex similarity between the agent network and a structure graph, we can produce a recommendation based on similarity scores that reflect both the link structure and the trust values on the edges. Our resulting approach is generic in that it can capture existing network-based approaches merely through the introduction of appropriate structure graphs. We evaluate different structure graphs with respect to two main kinds of settings, namely, social networks and ratings networks. Our experimental results show that our approach provides personalized and flexible recommendations effectively and efficiently based on local information.

Keywords Agent mining, Personalized recommendation, Social networks, ratings networks, Trust

A preliminary version of this manuscript, "Trust-Based Recommendation Based on Graph Similarity," was presented at the *AAMAS 2010 Workshop on Trust in Agent Societies* and appears in the unpublished workshop notes. Sections 1, 2, and 3 of this manuscript are based on that paper. This manuscript incorporates substantial revisions and extensions to the formal models and techniques. The evaluation and results are new.

Chung-Wei Hang · Munindar P. Singh
North Carolina State University
Tel.: +1-919-515-5677
Fax: +1-919-515-7896
E-mail: {chang,singh}@ncsu.edu

1 Introduction

An agent network is a directed weighted graph whose vertices represent agents, edges represent trust relations, and weights represent trust values from the source agent to the target agent. The resulting network in such a case is a social network. We generalize the above further to settings where some of the vertices are agents and others are objects such as products or services. In such a setting, an agent would trust an agent or an object but an object does not trust anything. (Note that a service that is sufficiently sophisticated to trust others is better treated as an agent rather than an object.) The resulting network in such a case is a ratings network.

We treat *trust* in the above (generalized) agent networks in a generic way here to indicate any positive relationship conducive to trust in the real world. For example, friendship, kinship, and business transactions among agents are appropriate for our approach; similarly, an agent liking a product or service is also appropriate for our approach.

An edge from vertex u to vertex v with trust value t means u trusts v to the extent of t. A similar concept is commonly seen in the real world. Examples include Facebook[1] where vertices are people and edges mean friendship; citation networks where vertices are papers (or their authors) and edges mean citations; web graphs where vertices are webpages and edges are hyperlinks; FilmTrust [19] where vertices are people and edges are movie taste ratings; and, Epinions [10] and Advogato [21,34], where the edges are trust relations.

There are two main challenges in agent networks: (a) *trust propagation* and (b) *personalized recommendation*. Trust propagation is about predicting the trustworthiness of nonadjacent agents by combining trust values through distinct indirect paths. To be more specific, trust propagation defines how trust values are aggregated and propagated through an agent network. It can help agents estimate a stranger's trustworthiness without assuming previous experience with the stranger. The other problem in agent networks is personalized recommendation. Given an agent network and a requester agent v (a vertex), how may we recommend a trustworthy agent for v to interact with, personalized to v? Both the problem of trust propagation [9,12,21,28,29,32–34] and personalized recommendation [6,10,14,31] have drawn much attention from the research community.

A possible solution to personalized recommendation is to apply trust propagation to estimate the trustworthiness of all agents that are not adjacent to the requester v, and to recommend the agents that obtain high trust estimates. However, in order to do so, we need to calculate the propagated trust value for each recommendation candidate. Thus, the above solution is not scalable because the complexity of the trust propagation grows quickly as the number of agents increases. Besides, trust propagation fails when there exist no indirect paths to the recommendation candidates, for example, in bipartite datasets (Section 4).

Another possible solution is to recommend agents who share a number of common neighbors. For example, Facebook recommends friends based on the number of mutual friends between people. More general solutions apply graph mining techniques on the whole agent network to predict trust reputation values of nodes and edges [10,14]. However, these approaches may suffer in a distributed setting. For

[1] http://www.facebook.com/

example, due to the scale of the environment or the privacy constraints, the agent network may be implicitly distributed over heterogeneous agents. Processing such information in a centralized manner is infeasible.

Agent mining provides a tempting idea to address above issues by integrating agents and data mining [5]. Agent mining seeks to either use agents to mine large data (agent-driven data mining), or to make agents more efficient based on the knowledge extracted via data mining (data mining for agents) [7,17,26]. In this paper, we propose an agent mining approach for agent-centered trust-based recommendation, where agents apply graph mining techniques on their local agent networks to explore potential trustworthy interaction partners in a distributed, personalized manner.

Our approach, dubbed LocPat, aims to provide personalized recommendations, i.e., to recommend trustworthy relationships by considering the link structure (e.g., the number of common neighbors). Instead of considering each potential neighbor separately, LocPat processes the agent network around the requester (i.e., the agent that requests a recommendation), thereby providing recommendations more efficiently. LocPat is based on graph similarity [3]. We show that by calculating vertex similarity between the agent network and a specified *structure graph*, the personalized recommendation problem can be translated into a graph similarity problem, and the similarity scores can be viewed as a measurement of how many good connections (i.e., with high trust values) the agents shares with the requester. Besides, instead of predicting how the agent network evolves from a *network*-level perspective, LocPat adopts a personalized or *vertex*-level perspective, providing personalized recommendations from the perspective of a requester. The benefits of LocPat include (1) flexibility, (2) decentralized design, and (3) efficiency. First, LocPat provides customized recommendations, meaning that we can customize it with respect to any of a variety of heuristics. Second, LocPat only requires a local subgraph of the whole network. Third, LocPat is based on graph similarity, which has been shown to be computationally efficient [3].

Note that we can customize personalized recommendation by using different structure graphs. In this paper, we study six basic structure graphs and eight variants of LocPat that facilitate making recommendations based on natural patterns such as friend-of-a-friend (three variants), coupling, cocitation, and introduction. However, LocPat is not limited to these structure graphs. It can be extended to produce recommendations based on other criteria that can be expressed via weighted directed graphs.

To summarize, our personalized recommendation takes an agent network and a requester (who requests a recommendation in the agent network) as inputs, and outputs a list of trustworthy agents for the requester to interact with as recommendation associated with estimated trust values. This paper makes four key contributions to agent mining. First, LocPat provides personalized recommendations for a particular agent. Second, LocPat produces a recommendation based not only on the network topology, but also on the trust values associated with the edges. Third, LocPat is efficient. Besides, it considers only a small subgraph of the agent network (selected in a personalized manner), and processes potential candidates all at once. Fourth, LocPat enables customizing recommendations based on a variety of criteria.

The rest of this paper is organized as follows. Section 2 surveys the state of the art in related research areas. Section 3 presents the LocPat approach by first

introducing the graph similarity measurement used in LocPat, and then demonstrating how the personalized recommendation problem can be solved via graph similarity. Section 4 evaluates our approach over four real, preexisting datasets. Section 5 concludes with a discussion of the ramifications of our approach and identifies some important future directions.

2 Related Work

We categorize the relevant literature into four areas: *graph similarity*, *link prediction*, *trust propagation*, and *recommender systems*.

Graph Similarity

Most common vertex similarity measurements seek to estimate the *structural equivalence* between two nodes within a graph—two nodes are similar if they share many common neighbors [23]. Structural equivalence can be measured by, for example, the Jaccard index, or Çosine similarity. Leicht et al. [20] generalize the idea of vertex similarity that two nodes are similar if they occupy similar structural positions without sharing common neighbors. Vertex similarity can be further extended to measuring the structural similarity between vertices in different graphs. LocPat is built on such a vertex similarity measurement that measures structural similarity between a node in one graph and a node in another graph [3]. This measurement considers not only the structural positions of nodes in two graphs but also the weights on the edges. If we take a structural pattern as the first graph and an agent network as the second graph, by looking at the vertex similarity scores, we can construct an algorithm to locate two nodes in the agent network that are connected via the most structural patterns.

There are other domain-specific graph similarity has been applied in various settings. For example, Melnik et al. [24] present a graph similarity approach, called *similarity flooding*, for database schema matching. Their approach takes two graphs representing schemas as inputs, measures the vertex similarity between the inputs, and outputs a mapping between the schemas as a subgraph consisting of similar vertices. This work differs from ours in many ways except that both apply vertex similarity between two graphs. First, Melnik et al.'s input graphs have no weights, whereas LocPat takes edge weights into consideration. Second, their approach takes two graphs as input, and calculates the similarity between them. LocPat takes only one graph as input. Given that graph, LocPat calculates the similarity between the graph and a separate *structure graph*, which reflects the features we care about in the desired recommendation. Third, Melnik et al.'s approach requires adjustment by humans, which LocPat does not.

Jeh and Widom [13] propose a domain-independent similarity measurement, *SimRank*. SimRank measures the similarity between objects. It follows the intuition that "two objects are similar if they are related to similar objects." Jeh and Widom first convert the graph into a *vertex-pair* graph, where each vertex represents a vertex-pair in the original graph. The vertex-pair (a, b) is connected to the vertex-pair (c, d) if a connects to c and b connects to d in the original graph. Then Jeh and Widom calculate and propagate similarity score in the converted graph iteratively

until convergence. Again, they only consider graphs with no edge weights, whereas edge weights (i.e., trust values) play an important role in our approach.

Link Prediction

Link prediction for large networks studies how, given the current snapshot of a network, to predict the edges that may be added in the future or those that are already present in principle but happen to be unknown in the model. Liben-Nowell and Kleinberg [22] survey various link prediction methods from graph theory and social-network analysis. These methods measure the similarity between vertices with respect to the network topology, assign a weight to each pair of vertices, and generate a list sorted in decreasing order in terms of weights. Liben-Nowell and Kleinberg evaluate these link prediction methods in five collaboration networks, where the edges connect authors who have coauthored papers. They indicate that the link prediction approaches can provide a network evolution model learned from the observed data. This learned network evolution model can help predict how the network is expected to evolve based on the network features. Unfortunately, Liben-Nowell and Kleinberg's approach considers only undirected graphs without edge weights.

Kunegis and Lommatzsch [18] propose a general link prediction approach that applies machine learning techniques to reduce the learning parameters for link prediction, and then uses a curve-fitting method to estimate the parameters. Their approach can be applied to undirected, weighted, or bipartite graphs. Kunegis and Lommatzsch evaluate the approach on web graphs, trust networks, social networks, citation networks, and collaboration networks. Note that, in general, the link prediction methods provide *network*-level prediction, whereas LocPat focuses on *vertex*-level recommendation. In other words, link prediction recommends links for the whole network, but LocPat recommends trustworthy others for a particular vertex.

Trust Propagation

Trust models have been widely studied in computer science [1]. Trust propagation provides an alternative solution to personalized recommendation from a vertex-level perspective. Recommendations can be made by first estimating the trustworthiness of all nonadjacent agents, and then listing the agents that obtain high trust estimates. Hang et al. [12] model trust as a binary event. Their approach may be considered a traditional friend-of-a-friend propagation. They define three operators for concatenating trust along a path, aggregating trust from distinct paths from the same witness, and selecting the most trustworthy path among all witnesses, respectively. Advogato's own approach, as specified by Levien [21], is to adopt a network flow algorithm where the flow capacity of edges is determined by their depth along the path. Appleseed, due to Ziegler and Lausen [34], applies spreading activation, where *trust energy* is spread across the agent network. The energy is divided when the agent has more than one successor.

Tavakolifard [31] studies trust management based on similarity. The similarity here is referred to how similar two agents are in an agent network. Tavakolifard argues traditional friend-of-a-friend propagation is insufficient to predict accurate trust values. Her approach also considers the similarity between agents with respect

to being either trustees or trusters. Tavakolifard's intuition, which we agree with, is that two agents are similar either if they trust the same agents or if they are trusted by the same agents. Further, we agree that traditional friend-of-a-friend propagation is merely one aspect behind general personalized recommendation. LocPat is a general approach which can be customized by using any graphical patterns, including patterns for friend-of-a-friend and similarity between agents.

Guha et al. [10] proposes trust propagation that considers structures including transitive trust (friend-of-a-friend), trust coupling, cocitation, and transpose trust. Their approach is more general than traditional trust propagation that only incorporates friend-of-a-friend. Their approach is more naturally applicable in settings where we cannot rely on forward edges alone, for example, in bipartite networks. However, their approach is limited to four predefined patterns, namely, direct propagation (friend-of-a-friend), co-citation, transpose trust, and trust coupling. The predictions made by Guha et al.'s approach rely on a heuristic they call *majority rounding*, which is vaguely defined, and may not be applicable in a larger number of situations (i.e., when no majority exists). LocPat also provides recommendation with trust value predictions. LocPat enables an agent to use any structure graphs it chooses to customize recommendations. We show the structure graphs that capture Guha et al.'s direct propagation, trust coupling, and cocitation patterns in Section 4.

EigenTrust [14] calculates reputation of all peers in peer-to-peer networks. EigenTrust propagates trust values through friend-of-a-friend relations. For each peer, it aggregates trust values from all the other peers into a global reputation values. LocPat predicts a trust value of a peer from a personalized perspective. That is, LocPat predicts trust values on nonexisting edges, whereas EigenTrust predicts reputation values on all nodes. Another difference is that LocPat considers not only friend-of-a-friend pattern but also any patterns defined by the structure graphs.

All these trust propagation methods (except EigenTrust) provide recommendations for a particular pair of agents. Besides, these methods lack flexibility by predefining the recommendation patterns. By contrast, LocPat does not require agents to know a particular target to estimate its trustworthiness. Also, LocPat provides a flexible, general framework that enables agents to customize recommendation with proper recommendation patterns.

Recommender Systems

Now we discuss some related work of recommender systems. In general, recommender systems suggest *items* to *users*. There are two main categories of recommender systems: *content-based* and *collaborative filtering* systems [30]. Content-based approaches produce recommendations based on the similarity between items. Collaborative filtering approaches recommend the items chosen by the users with similar tastes. Of these, our problem setting is closer to collaborative filtering because (1) we do not examine the contents of the objects involved and (2) some of the collaborative filtering approaches construct an agent network wherein vertices are users and each edge represents how much similarity the source user expects between his or her tasks and the tastes of the target user. For example, FilmTrust [19] is a social network where edges represent the similarity of movie taste.

Ben-Shimon et al. [2] propose a recommendation approach that is quite similar to [12]. They construct a *personal* social network containing friend-of-a-friend (up to six levels) of a user who needs recommendation. Ben-Shimon et al. then find the sum of all the ratings of a particular item, discounted by the distance from the rater to the user. They recommend the item if the sum is high. In contrast, no items are involved in our approach. Thus, rather than computing the sum of all ratings, LocPat considers the link structure and the trust values on the edges. Fouss et al. [6] present a recommender system based on a similarity measurement between the vertices of a directed weighted graph. They compute similarity based on a Markov-chain random walk model, which assigns a transition probability to each edge. The distance required for a random walker to travel from one vertex to another defines the similarity between these two vertices. Although both Fouss et al.'s approach and LocPat measure vertex similarity, these two approaches are quite different. LocPat measures similarity between a node in the agent network and a node in the structure graph, whereas Fouss et al. measure similarity between two nodes within the given network. And, LocPat uses the similarity measurement defined by a *mutually reinforcing relation* rather than the Markov-chain random walk model.

3 Our Approach: LocPat

Now we introduce our approach, LocPat. LocPat is an agent mining approach that provides personalized and customized recommendations based on local trust relations in an agent network. LocPat recommends potential trustworthy links by measuring the vertex similarity between nodes in the local agent network and nodes in a structure graph. The structure graph customizes the recommendations by defining how trustworthy links are located. LocPat also provides trust value predictions of each recommendation based on a local rounding method. In Section 3.1, we briefly review the graph similarity measurement applied in LocPat and show an application of graph similarity based on a structure graph. In Section 3.2, we begin by defining an agent network. Next, in Section 3.3, we formalize LocPat by customizing the graph similarity measurement by devising a structure graph that satisfies our claim for suggesting recommendations in an agent network. Then we show examples of how trust values are considered, and how they affect the recommendations produced. Section 3.4 explains how the similarity scores obtained from LocPat are translated into trust value predictions.

3.1 Background: Vertex Similarity between Graphs

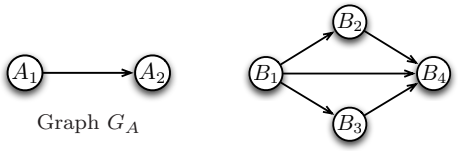
Blondel et al. [3] propose a vertex similarity measurement between graphs. Given two directed graph G_A with n_A vertices, and G_B with n_B vertices, a similarity matrix $\mathbf{S}$ is an $n_B \times n_A$ matrix where s_{ij} is the similarity score between vertex i in G_B and vertex j in G_A. $\mathbf{S}$ can be calculated by a convergent iterative process, where each $\mathbf{S}_i$ is a matrix of the same dimensions as $\mathbf{S}$:

$$\mathbf{S}_{k+1} = \frac{B\mathbf{S}_k A^T + B^T \mathbf{S}_k A}{\|B\mathbf{S}_k A^T + B^T \mathbf{S}_k A\|_F}, \tag{1}$$

where A and B are the adjacency matrices of G_A and G_B, respectively; $\mathbf{S}_0$ has all entries equal to 1; and $\|.\|_F$ is the Frobenius norm of a matrix (i.e., the square root of the sum of the squares of all matrix entries) [8]. The denominator normalizes each entry in $\mathbf{S}_{k+1}$ to $[0, 1]$. The limit of this convergent process is $\mathbf{S}$. The convergence to error tolerance ϵ can be determined by

$$\|\mathbf{S}_{k+1} - \mathbf{S}_k\|_F < \epsilon. \tag{2}$$

For example, Figure 1 shows the similarity matrix between two graphs: G_A and G_B. G_A contains two vertices: A_1 has out-degree of one, and A_2 has in-degree of one. After measuring the vertex similarity with G_B, one can observe that B_1, which has the largest out-degree, is the most similar vertex to A_1. B_4, which has the largest in-degree, has the highest similarity score to A_2. Notice that the greater the out-degree a vertex has the higher is its similarity to A_1. An analogous observation applies to the in-degree and A_2. Hence, we conclude that by comparing the similarity score to G_A, we can find the vertex that connects to the most others, and to which most others connect.



	A_1	A_2
B_1	0.5774	0
B_2	0.2887	0.2887
B_3	0.2887	0.2887
B_4	0	0.5774

Fig. 1 Example of two graphs, G_A and G_B, and their similarity matrix. The most similar vertex to A_1 is B_1 (largest out-degree); the most similar vertex to A_2 is B_4 (largest in-degree).

The idea behind the similarity measurement is the *mutually reinforcing relation*, which is widely applied in web search [4, 16], and reputation management in peer-to-peer systems [14]. To illustrate the mutually reinforcing relation, we take G_A and G_B in Figure 1 as an example. For each vertex B_i in G_B, we associate two similarity scores, say s_{i1} (for A_1) and s_{i2} (for A_2), each of which corresponds to the similarity between one vertex in G_A and B_i. Both scores are initialized to one. Then the scores are updated iteratively according to the mutually reinforcing relation:

$$\begin{cases} s_{i1} = \sum_{j:(i,j)\in E_B} s_{j2} \\ s_{i2} = \sum_{j:(j,i)\in E_B} s_{j1}, \end{cases} \tag{3}$$

where E_B is the set of edges of G_B. This mutually reinforcing relation says a vertex is similar to A_1 if it connects to many vertices that are similar to A_2, whereas a vertex is similar to A_2 if many vertices similar to A_1 connect to it. The update process is iterated. The scores s_{i1} and s_{i2} mutually reinforce each other. Blondel et al. show that this update process converges to a state that corresponds to the similarity scores between A_1 and A_2, and B_i.

Now let us extend the similarity scores to all vertices in G_B. Suppose $\mathbf{s}_1$ and $\mathbf{s}_2$ are the similarity scores to A_1 and A_2, respectively, for all B_i in G_B. We can rewrite Equation 3 as a recursive relation:

$$\mathbf{S}_{k+1} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix}_k = \mathbf{S}_k, \tag{4}$$

where B is the adjacency matrix of G_B and $k = 0, 1, \ldots$ is the number of iterations. Blondel et al. further extend G_A to an arbitrary graph and generalize Equation 4 to Equation 1, where A is the adjacency matrix of G_A.

Notice that we can view G_A as a query because it serves as the basis for selecting appropriate vertices from G_B. For this reason, G_A is termed the *structure graph*. By using different structure graphs, we can apply vertex similarity to solve different problems and applications. Blondel et al. show that the web search algorithm, *HITS* [16], which searches for web pages based on a query, is an application of vertex similarity. HITS ranks web pages based on an *authority score* and a *hit score*. A web page is a good authority if there are many hits that link to it. In contrast, a good hit is a web page that points to many good authorities. The HITS algorithm is a special case of the above that computes the vertex similarity between the graph induced from the web and the graph G_A of Figure 1.

3.2 Personalized Recommendation

An agent network is a graph whose vertices represent entities (agents or objects) and edges represent trust relations between agents and entities. This is a generalization of the model of Yu and Singh [33]. A trust relation from agent u to entity v indicates how much trust u places in v. Thus, an edge in an agent network is associated with a trust value as its weight. Depending on the trust models, a trust value can be a single scalar, a Beta distribution, or follow another representation. The trust relations can be obtained from direct interaction or from a referral via trust propagation [12]. For example, a social network such as Facebook is an agent network where all edges are modeled as having the same trust values. For simplicity, here we treat a trust value as a single scalar. Other trust representations can be translated into a scalar, for example, a probability.

Definition 1 An agent network is a directed weighted graph $G(V, E)$, where V is a finite set of agents $\{v_1, \ldots, v_n\}$, and E is a set of trust relations $\{e_1, \ldots, e_m\}$. A weighted adjacency matrix $\mathbf{A}$ of an agent network $G(V, E)$ of n agents is an $n \times n$ matrix where the entry a_{ij} is the trust value v_i places in v_j, where $v_i, v_j \in V$.

Each of the edges e_m in G is associated with a trust value. Instead of using a traditional adjacency matrix, whose entries are 0 or 1, we define a *weighted adjacency matrix*, in which the entries are real numbers. In this manner, a weighted adjacency matrix generalizes over an adjacency matrix for multigraphs (permitted to have multiple edges between the same vertices), whose entries are nonnegative integers. The entries in the weighted adjacency matrix represent the trust values associated with the corresponding edges.

We now formalize our personalized recommendation approach. Given an agent network $G(V, E)$, to find recommendations for agent v, we construct a *neighborhood network* for v as $G'(V', E')$ where $v \in V'$, $V' \subseteq V$, and $E' \subseteq E$ are all the trust relations involving the vertices in V'. The reason we take a (typically, proper) subgraph of G including the neighborhood of v is that were we to consider the whole G, the result would not necessarily be a recommendation for the agent v but represent an amorphous notion of a global recommendation. Instead, the agents with high similarity scores would simply be those similar to w in G_S. That is,

these agents are connected to by many other agents that are connected to by many others. We revisit the choice of V' below.

We define a structure graph G_S with a query vertex w. The structure graph G_S defines the pattern that matters in the recommendation. The query vertex w defines the vertex we want to recommend in the structure graph. For example, as mentioned in Section 3.1, the HITS algorithm [16] uses the G_A of Figure 1 as the structure graph, which is interpreted as hubs (A_1) connecting to authorities (A_2). To recommend hubs, G_1 is used as the structure graph and A_1 is the query vertex. To recommend authorities, G_1 and A_2 are used as the structure graph and the query vertex, respectively. Then we calculate the similarity matrix between the structure graph G_S and the weighted adjacency matrix $\mathbf{A}'$ of G'. The vertices that are not neighbors of v and have high similarity scores to w are recommended.

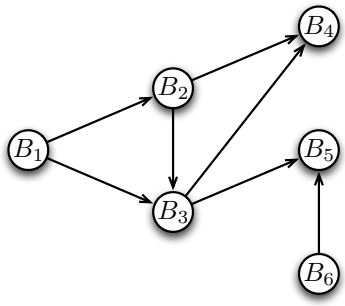
The choice of G_S (with w) depends on the context. Once G_S is chosen, V' can be constructed accordingly. We show some examples in Sections 3.3 and 4. We can summarize the main steps of the LocPat approach as follows:

1. Let an agent be v in an agent network $G(V, E)$ ($v \in V$).
2. Define the structure graph G_S and a query vertex w in G_S.
3. Construct $G'(V', E')$, where $V' \subseteq V$ contains v; $E' \subseteq E$ contains all trust relations between $(u', v') \in V'$. The choice of V' is determined by the context.
4. Calculate the similarity matrix $\mathbf{S}$ between G_S and the adjacency matrix $\mathbf{A}'$ of G' using Equation 1.
5. Recommend the vertices that are not neighbors of v but obtain high similarity scores to vertex w in G_S.

3.3 A Simple Example on a Social Network

Consider the recommendation problem in agent networks: Given a snapshot of an agent network, how can we recommend (i.e., predict) a trustworthy agent to an agent v? Based on the obvious intuition, we claim a good recommendation for v is an agent to whom many of v's neighbors connect. Let us start with a simple case where all edges have the same trust values 1 (i.e., no weights). For example, Figure 2 (left) shows an agent network G, which contains the neighbors of the neighbors of agent B_1. Among all the agents except B_1's neighbors B_2 and B_3, B_4 is the most natural candidate, because two neighbors of B_1 connect to it. Consider the structure graph G_S in Figure 3, which illustrates our claim of producing good recommendations: a friend (A_3) of A_1's friend (A_2) is potentially A_1's friend (i.e., a good recommendation for v is an agent to whom many of v's neighbors connect). Figure 2 (right) shows the similarity matrix between G_S and G. The similarity score between A_3 and vertices indicates how the link structure of the vertices is similar to the link structure of A_3.

Now we consider the general case where each of the edges in G is associated with a trust value. Figure 4 shows an example of an agent network G', and the similarity matrix between the structure graph G_S in Figure 3 and G'. G' shares the same topology as G in Figure 2, except G' has trust values as its edge weights (rather than 1). Unlike the outcome in Figure 2, although B_5 has fewer connections with B_1's neighbors than B_4, B_5 has the highest similarity score because the trust value of its only connection is much stronger than the trust values of B_4's connections. Note that B_3 (not considered as a recommendation because it is already a neighbor



	A_1	A_2	A_3
B_1	0.43	0.09	0
B_2	0.31	0.36	0.05
B_3	0.08	0.48	0.23
B_4	0	0.13	0.43
B_5	0	0.03	0.29
B_6	0.02	0.10	0

Fig. 2 Example of an agent network with no edge weights specified (assume each edge weight is 1), and its similarity matrix with the structure graph G_S in Figure 3. Among friends of B_1's friends (i.e., B_4 and B_5), B_4 is the best recommendation with the highest similarity score with A_3.

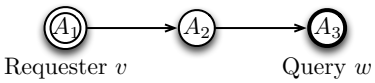
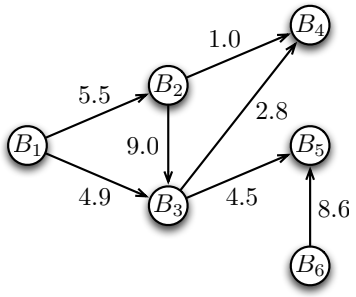


Fig. 3 Structure graph G_S. Double circled A_1 indicates the requesting vertex v and bold A_3 indicates the query vertex w.

of B_1) also has a high similarity score because B_2 (B_1's neighbor) connects to it with a high trust value.



	A_1	A_2	A_3
B_1	0.40	0.13	0
B_2	0.40	0.34	0.07
B_3	0.01	0.48	0.35
B_4	0	0.03	0.13
B_5	0	0	0.36
B_6	0	0.19	0

Fig. 4 Example of an agent network where edge weights represent trust values, and its similarity matrix with the structure graph G_S in Figure 3. Unlike the result in Figure 2, although both of B_1's friends connect B_4, B_5 is the best recommendation because of its strong connection with B_3.

3.4 Personalized Scaling and Rounding

THe graph similarity measurement produces relative scores. We compare these scores to distinguish good recommendations from bad ones. However, these scores cannot be used as predictions because they are not in the same scale as the trust values in the agent network. For example, in FilmTrust [19], trust values are integers from 1 to 10, whereas the similarity scores are normalized numbers from 0 to

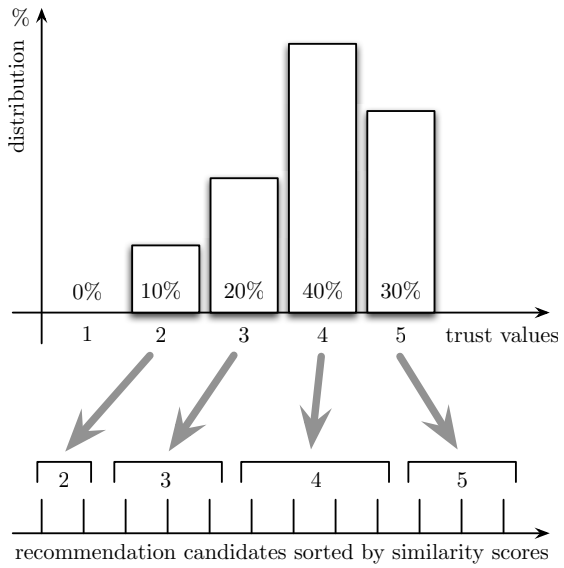


Fig. 5 Similarity scores are rounded to trust values based on the requester's trust value distribution.

1. Without properly scaling to provide trust value prediction, the applicability of this approach would be limited.

To solve this problem, we provide a rounding approach to *locally* map the similarity scores to trust values. Following the decentralized and personalized nature of our approach, here, by locally we mean rounding from a requester's perspective. Given a requester v, a structure graph, and its corresponding neighborhood network, our approach suggests a list of candidates with similarity scores. Our assumption is that the distribution of the trust values that v places in the candidates would naturally follow the distribution of v's trust values placed in its known neighbors. One benefit of doing so is the rounded trust values reflect the requester's preference. For example, a picky critic would hardly ever provide a high rating, whereas an easy-going reviewer may always give the highest possible rating. Another benefit is this approach only requires local information.

Figure 5 illustrates our rounding approach. As an example, consider the trust values the requester places in its neighbors. Suppose there are five integral trust values 1, 2, 3, 4, and 5 with the distribution of 0%, 10%, 20%, 40%, and 30%, respectively. We take the recommendation candidates located by LocPat and sort them from low to high based on the similarity scores. Then we assign trust values based on the distribution. That is, the first 10% candidates are assigned to trust value 2. The following 20% are assigned to trust value 3, and so on. Specifically, we make the computed trust values follow the same distribution as the original edge data. Note that the rounding process is a classification problem. We can plug in other supervised classification methods into our approach if necessary.

Table 1 Summary of the datasets used in our experiments.

Category	Social network		Ratings Networks	
Name	FilmTrust[2]	Advogato[3]	MovieLens[4]	Jester[5]
Graph	Directed		Directed, Bipartite	
Type	People $\mapsto$ People		People $\mapsto$ Items	
Goal	Recommend people		Recommend items	
Vertices	528	2,703	943	2,493
Edges	1,233	26,653	1,682	61,743
Weights	$[1, 10]$	$\{1, 2, 3\}$	$[1, 5]$	$[-10, 10]$

4 Experimental Evaluation

As Table 1 summarizes, we evaluate LocPat via experiments with two kinds of datasets: *social networks* and *ratings networks*. Note that we treat edge weights as trust values in a linear manner, i.e., higher ratings indicate greater trust. Other, more sophisticated translation approaches can be also used. For example, Hang et al. [12] introduced the Weber-Fechner transformation for trust propagation, and it would be interesting to consider it here.

Social Network Datasets

We consider two social network datasets. FilmTrust [19] is a social network where edges represent trust relations in terms of movie taste similarity. Advogato [21] is an agent network where the vertices are people and the edges are trust relations. In these datasets, the goal is to recommend potential friends from strangers, as described in Section 3.3.

Ratings Network Datasets

In ratings network datasets, there are two sets of vertices, representing people and items, respectively. A ratings network dataset forms a bipartite graph. More precisely, there are only edges from people to items. An edge from a person to an item indicates that the person likes or dislikes the item. Our goal is to recommend items for a particular vertex v (a person).

We consider two ratings network datasets. MovieLens [25] is a movie recommendation dataset where the vertices are users and movies, and the edges from users to movies indicate how a user likes a movie. Jester [27] is a joke recommendation dataset where the vertices are people and jokes, and the edges from people to jokes indicate how a person rates a joke. If a person has not read a joke, there is no edge between them.

[2] http://trust.mindswap.org
[3] http://www.advogato.org/
[4] http://www.grouplens.org/node/73
[5] http://eigentaste.berkeley.edu/dataset/

4.1 Experimental Setup

We compare LocPat with Guha [10], EigenTrust [14], and Fouss [6]. The differences between these approaches are described in Section 2. In our comparison, all the scores produces by these three approaches are translated into trust value predictions using our rounding method (Section 3.4).

In our experiment, we devise eight variants of our approach with six structure graphs. Table 2 enumerates the six structure graphs, where a double circle indicates the requesting vertex and a bold circle indicates the query vertex w. We collectively term the transitive trust structure graphs Foaf for *friend-of-a-friend*. Foaf2, Foaf3, and Foaf4 capture transitive trust with depths of two, three, and four, respectively. We dig no deeper than four levels because it is quite well-known (and has been confirmed by Katz and Golbeck [15] among others) that people tend to place more trust in shorter paths, even though considering deeper paths can take more candidates into account. We return to this point in Section 4.2. Coupling considers the situation where a party who trusts the same items can be trustworthy. A real-world example is that people tend to trust a reviewer who recommends products that they like very much. Cocite is similar to the scenario of Coupling except Cocite recommends the products the reviewer rates very high but which the agent has never used. Coupling recommends a reviewer whereas Cocite recommends a product. Intro represents the case where the parties trusted by my truster can be trustworthy. For example, one of your colleagues may introduce you to another professional colleague, because the introducer knows both you and the other colleague well. FoafAll is a hybrid variant that adds the weighted similarity scores from Foaf2 (0.6 weight), Foaf3 (0.3 weight), and Foaf4 (0.1 weight). The LocPat variant we compare with Guha, EigenTrust, and Fouss is a variant that sums the scores from FoafAll, Coupling, Cocite, and Intro with equal weights.

For each user in a dataset, we apply all approaches to find recommendation candidates and predict their trust values based on the training set. This leads us to two evaluation criteria.

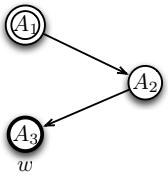
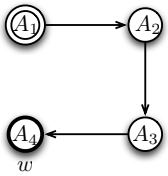
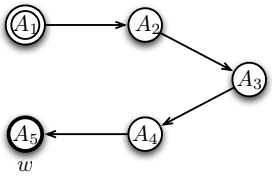
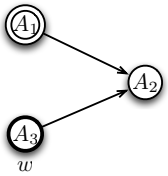
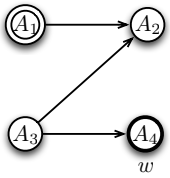
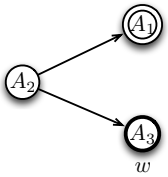
Error. The *error* is the difference between the predicted trust value and the actual value in the testing set. We summarize the errors as the root mean square error (RMSE), and calculate the mean and standard deviation of RMSE over all users.

F-measure. A user is *covered* in a recommendation approach if a recommendation candidate for this user exists in the testing set, the recommendation locates it. And, the *recall* of an approach is simply the fraction of users in the test set who are covered by that approach. Given the recommendation list and the testing set of a user, the average precision is calculated by the average of the rank in testing set divided by the rank in the recommendation list. The *precision* of an approach is the mean average precision among all users. Then the *F-measure* is calculated by

$$\frac{2 \times recall \times precision}{recall + precision}.$$

We evaluate our approach by three-fold cross validation. (In MovieLens, we use the five-fold cross validation data provided by default.) For each dataset, we divide the edges into three disjoint sets of equal size. We iterate over each edge

Table 2 Our approach captures existing approaches merely through the specification of suitable structure graphs. This table shows the structure graphs used in our evaluation.

Name	Structure	Description
FOAF2		A friend of a friend is trustworthy (2 levels)
FOAF3		A friend of a friend of a friend is trustworthy (3 levels)
FOAF4		A friend of a friend of a friend of a friend is trustworthy (4 levels)
COUPLING		A party who trusts the same target as I do is trustworthy
COCITE		The target that is trusted by a party who likes the same things as I like is trustworthy
INTRO		A party who is trusted by a party who trusts me is trustworthy

set. At each iteration, we take each set as a testing set, and the remaining two sets as a training set. The training and testing sets are 67% and 33% of all edges, respectively. We apply all the approaches to the agent network of the training set, locate the recommendations, predict trust values of each recommendation, and validate results on the testing set. We calculate the error and F-measure for each iteration, and average over the three iterations.

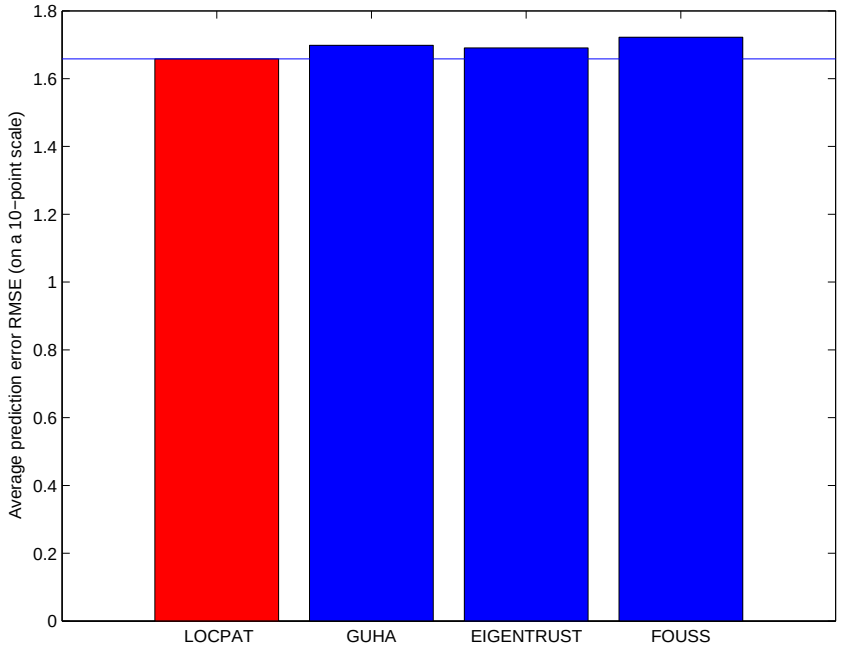


Fig. 6 Average prediction error RMSE in FilmTrust.

4.2 Results and Observations

Now we show our experimental results in social networks (Section 4.2.1) and ratings networks (Section 4.2.2), and discuss the size of the neighborhood networks considered by LocPat and local rounding compared with global rounding (Section 4.2.3). The efficiency of all approaches are studied in Section 4.2.4.

4.2.1 Social Networks

Figure 6 and 7 show the error and the F-measure of all approaches in FilmTrust. All approaches provide competitive predictions. LocPat yields the most accurate prediction and the best F-measure. We observe that LocPat produces the best precision but the worst recall because it only considers local neighborhood networks. The considered local neighborhood networks cover around 75% of the testing set. Guha, EigenTrust, and Fouss all takes the whole network into account, yielding recall around 85%, 80%, and 100%, respectively. LocPat yields the best precision, followed by Guha, EigenTrust, and Fouss.

Figure 8 and 9 show the error and the F-measure of all LocPat variants in FilmTrust. Most variants yield similar accuracy as LocPat. Besides the Foaf family of patterns, Coupling is quite effective in terms of low error. Regarding recall and precision, Foaf2, Coupling, and Intro have the highest precision, although their recall is lower than others. The rest of the Foaf family achieves better recall by considering larger neighborhood networks, though with the pre-

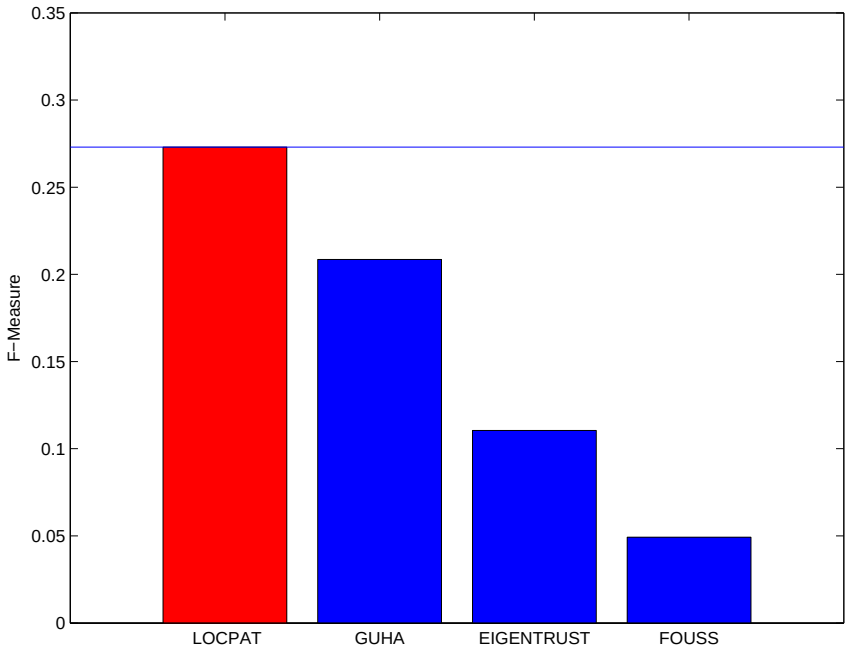


Fig. 7 F-Measure in FilmTrust.

cision compromised. Coupling and Intro, which are not captured by traditional trust propagation, provide effective recommendations as well.

Figure 10 and 11 compare the error and the F-measure of all approaches in Advogato. All approaches yield almost the same prediction accuracy. All approaches have near perfect recall. LocPat produces the best F-measure with the best precision. Fouss yields the worst precision, followed by Guha and EigenTrust.

Consider the error and the F-measure of LocPat variants in Advogato. Figure 12 shows that almost all of the variants are effective in terms of prediction accuracy. Figure 13 shows the F-measure of the LocPat variants in Advogato. Similarly to the case for FilmTrust, Foaf2, Coupling, and Intro yield the highest F-measure with the highest precision. Other variants have higher recall, though with their precision compromised. Differently from the case for FilmTrust, Cocite is quite effective in Advogato.

4.2.2 Ratings Networks

In ratings networks, edges only exist from people to items. Our goal is to recommend items to people. The friend-of-a-friend type of approaches (e.g., EigenTrust and Foaf family) are not applicable in this case because there are no paths of length two. Note that Coupling and Intro fail to make a contribution either, because Coupling is used to recommend people to people, and Intro is used when there exist edges connecting to people. Cocite is the only applicable pattern in ratings networks, because it captures the idea that the items trusted by the person who trusts the same items as you are also trustworthy. In other words, Cocite

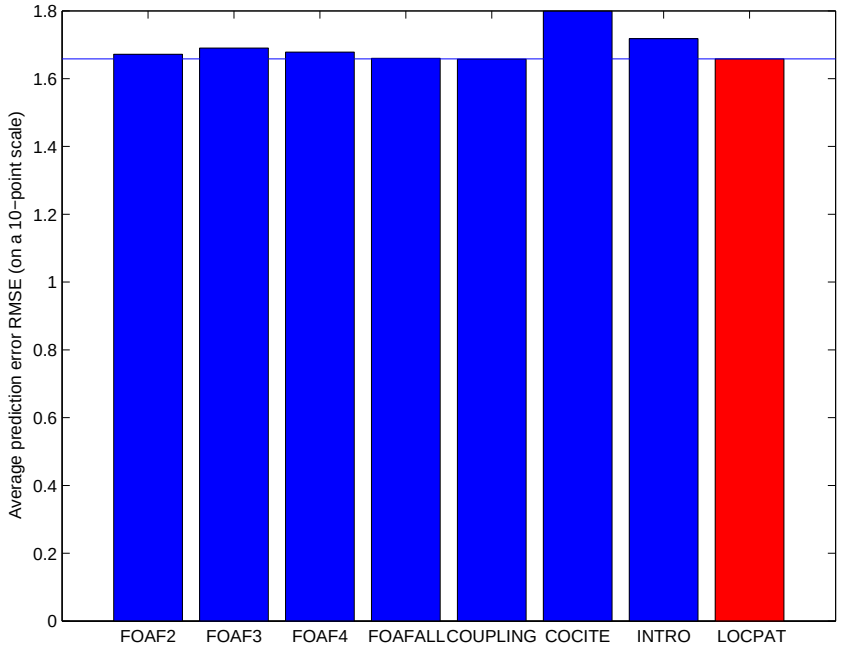


Fig. 8 Average prediction error RMSE of LocPat variants in FilmTrust.

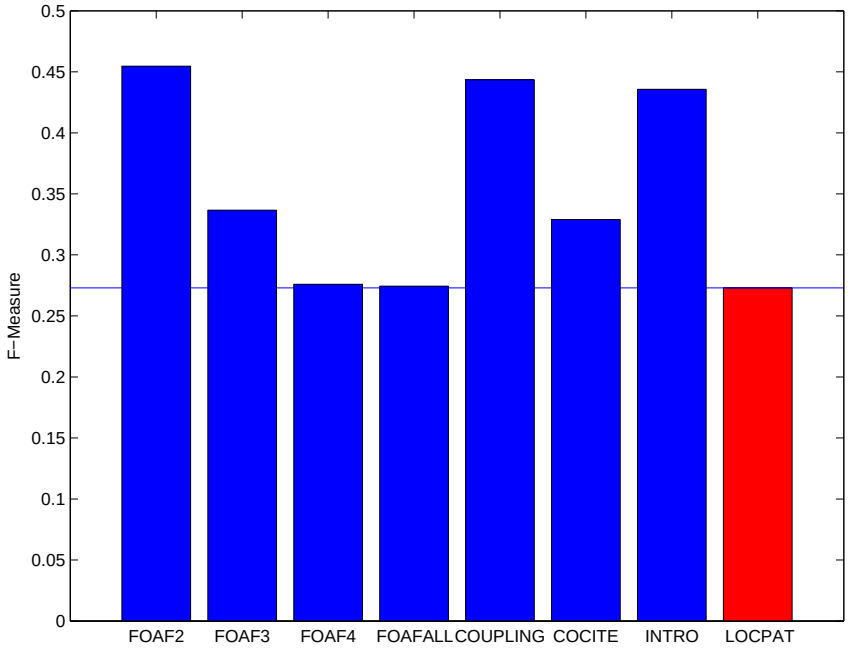


Fig. 9 F-Measure of LocPat variants in FilmTrust.

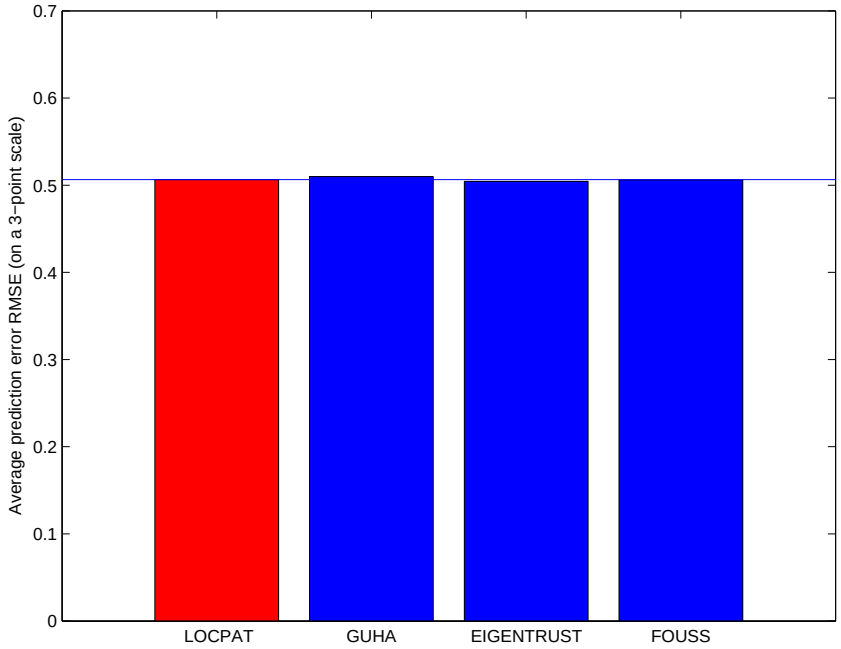


Fig. 10 Average prediction error in Advogato.

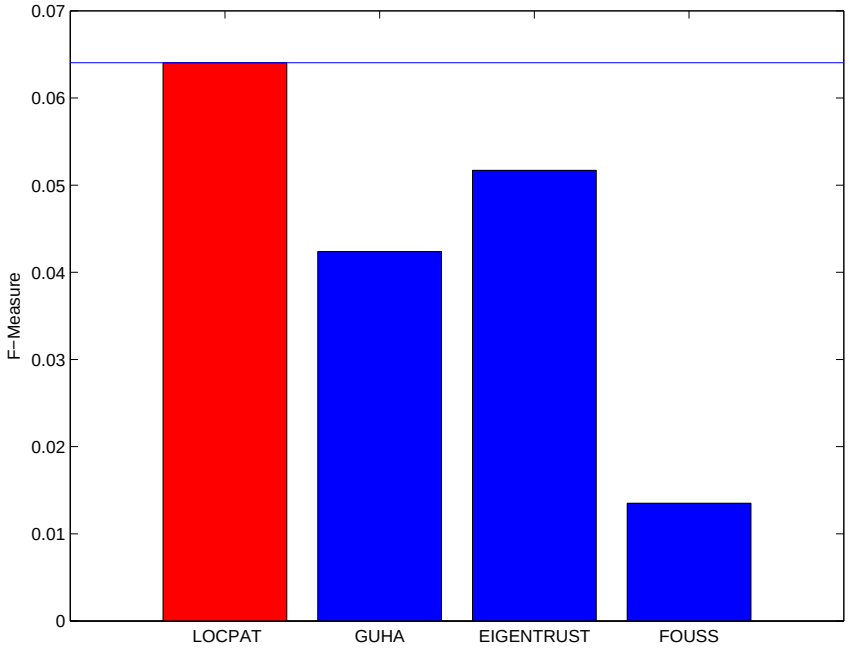


Fig. 11 F-Measure in Advogato.

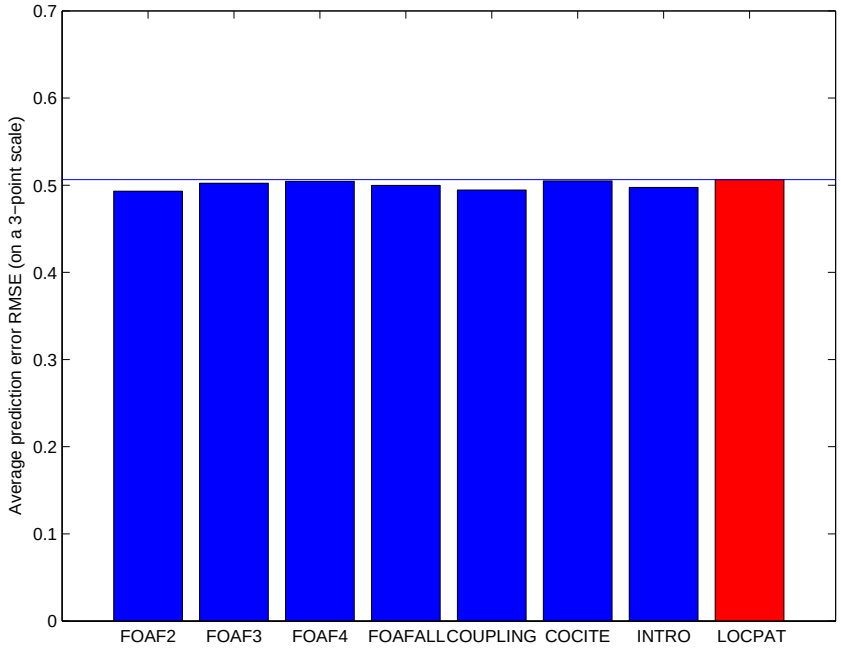


Fig. 12 Average prediction error of LocPat variants in Advogato.

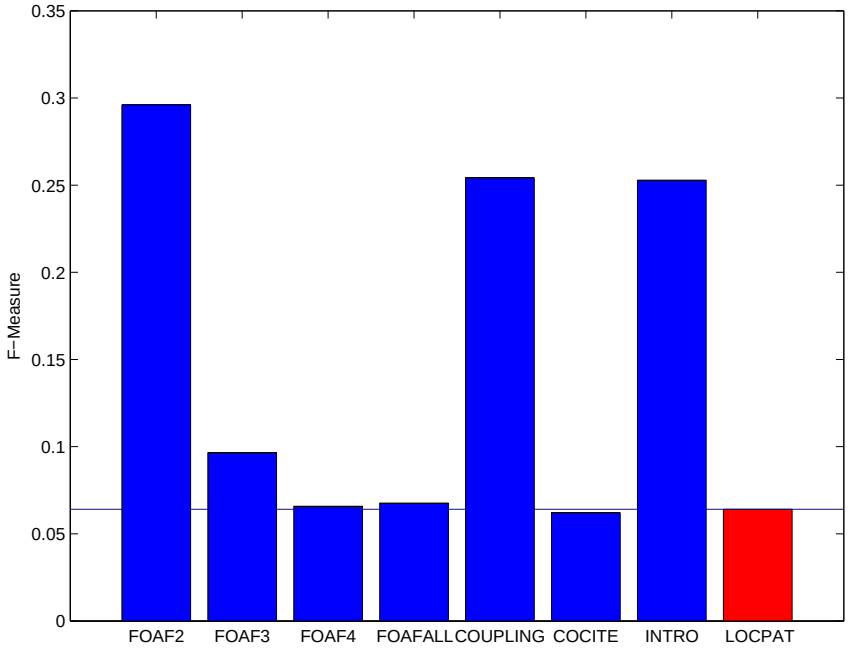


Fig. 13 F-Measure of LocPat variants in Advogato.

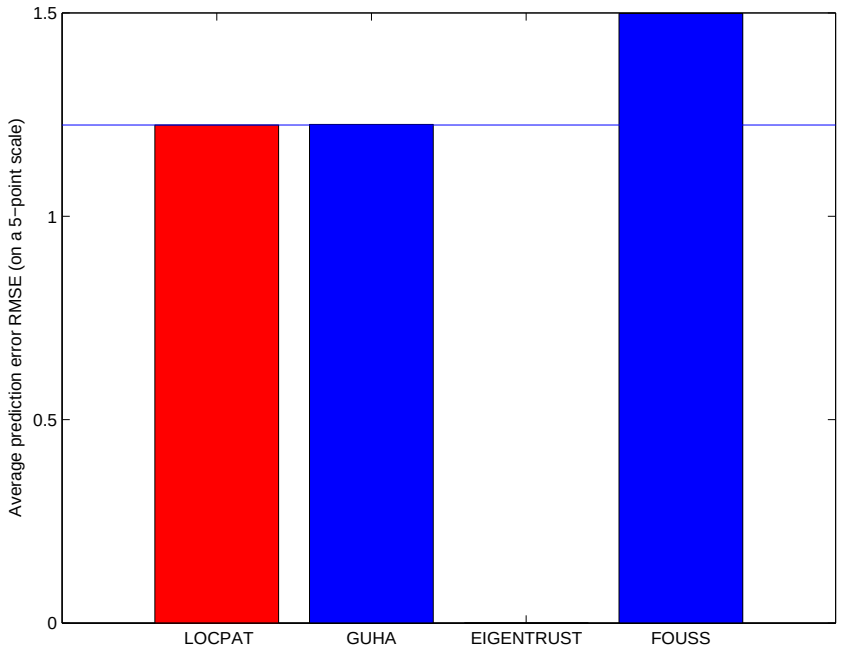


Fig. 14 Average prediction error in MovieLens.

locates people who have similar interests as you, and then recommends the items those people trust to you.

Figure 14 and 15 show the error and the F-measure of all approaches in MovieLens. Guha is competitive against LocPat in terms of prediction error. Fouss produces the worst prediction. In terms of F-measure, all approaches have perfect recall. LocPat yields the best precision, leading to the best F-measure. EigenTrust is not applicable in ratings networks. Figure 16 and 17 show similar results in Jester, where LocPat outperforms Guha and Fouss.

4.2.3 Neighborhood Networks and Rounding

Figure 18 plots the percentage of edges in the neighborhood network considered by variants of LocPat in FilmTrust. We average the percentage over all users. FoafAll and LocPat on average consider about 11% of the edges. Cocite uses 9%. Coupling and Intro consider only around 2%. This result shows that LocPat provides competitive recommendations while considering only a small local neighborhood network. Note that the percentage of edges is not necessarily correlated to the coverage and prediction error. For example, FoafAll considers 10% of edges and produces best prediction and average coverage, whereas Cocite uses 9% of edges but yields bad prediction and low coverage. However, depending on the network, there exist cases where the local neighborhood network contains almost all edges. For example, in MovieLens, the local neighborhood networks considered by Cocite contain on average 96% of the edges. This is why Cocite reaches near perfect coverage in MovieLens.

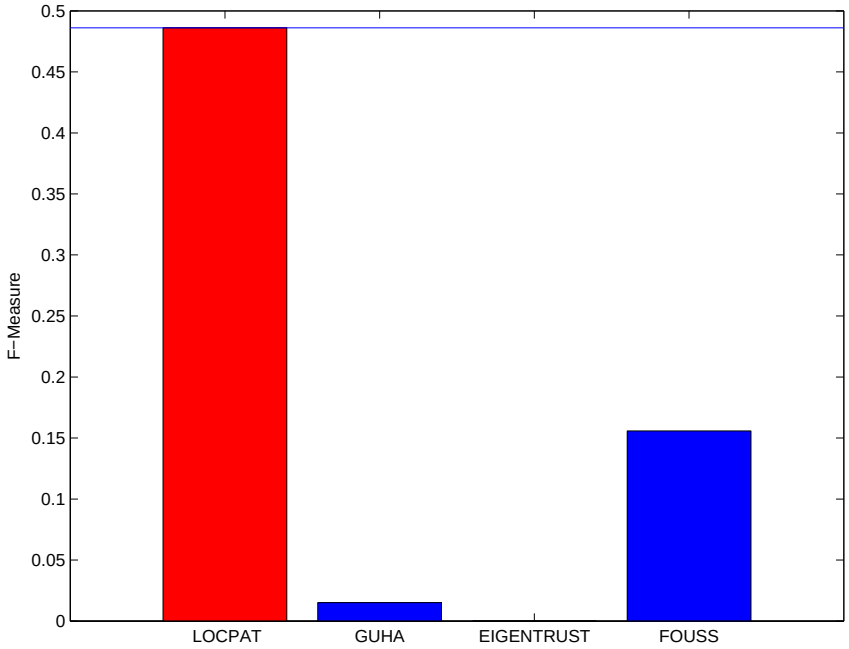


Fig. 15 F-Measure in MovieLens.

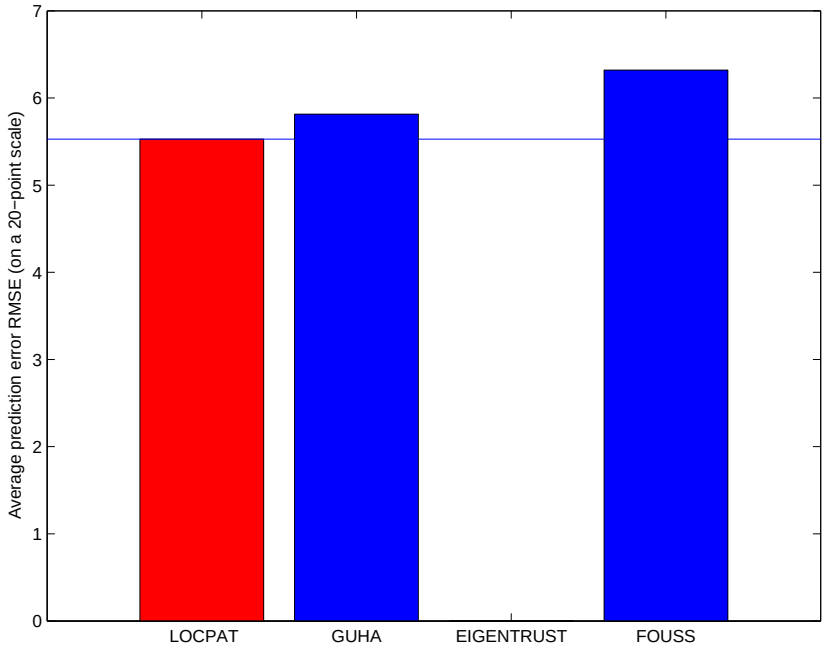


Fig. 16 Average prediction error in Jester.

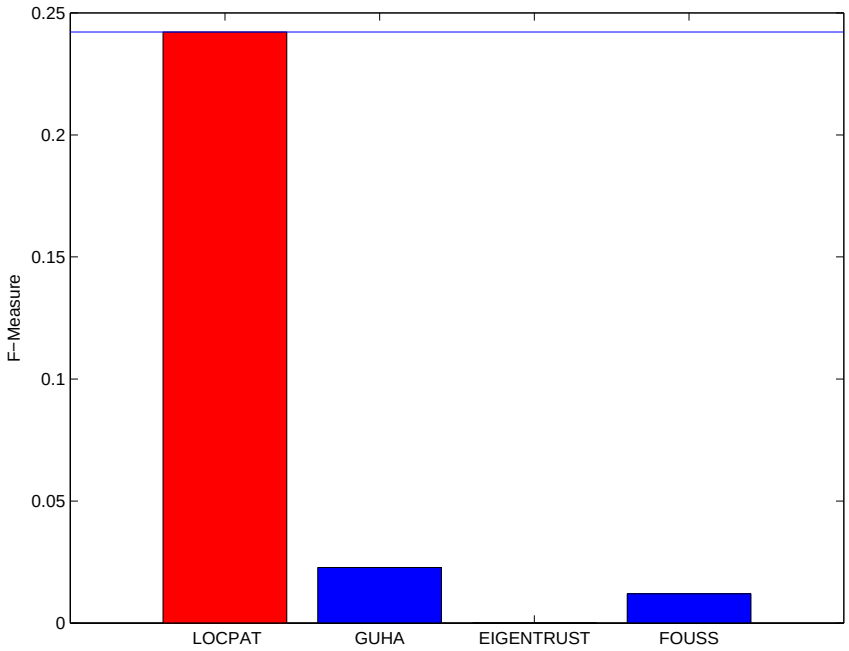


Fig. 17 F-Measure in Jester.

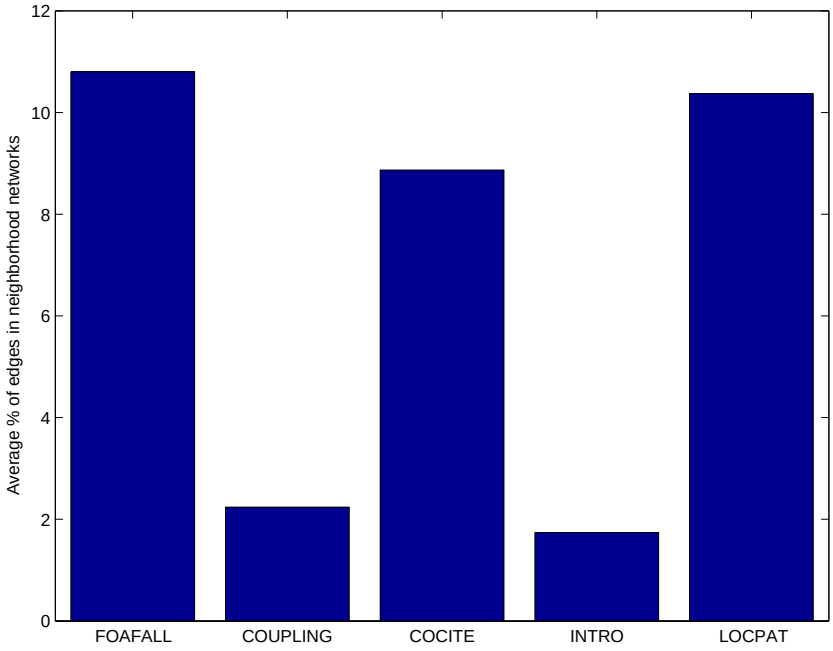


Fig. 18 Average percentage of edges considered by variants of LocPat in FilmTrust.

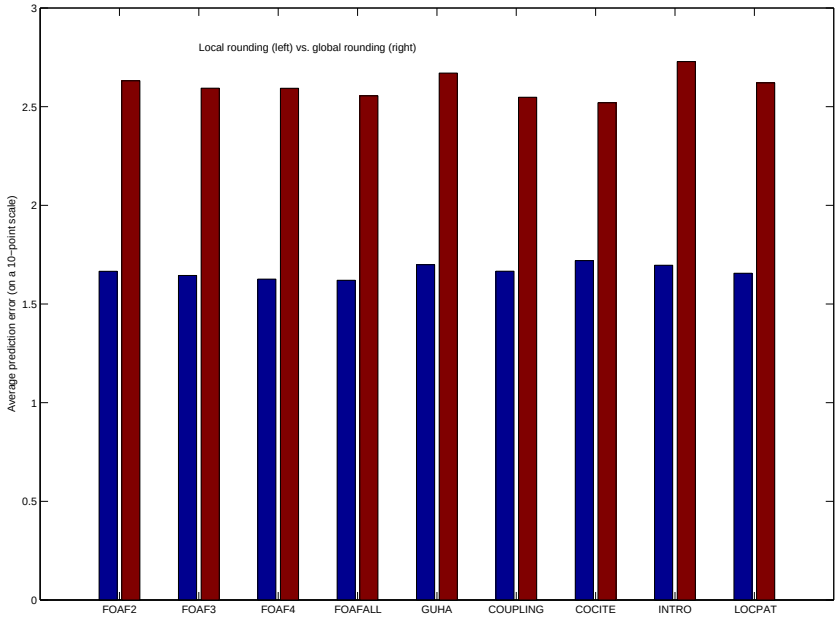


Fig. 19 Average prediction error with local and global rounding in FilmTrust.

Figure 19 compares the local and global rounding heuristics in FilmTrust. Global rounding assigns trust values based on the trust value distribution of the whole dataset, whereas our proposed local rounding considers only distribution of the trust values given by the requester. Our result shows local rounding dominates global rounding in terms of the prediction error, indicating local rounding can better reflect personal tastes. Also, local rounding is better compatible with decentralized setting because it requires knowledge of only local trust values.

4.2.4 Efficiency

Figure 20 shows the running time of all approaches. For LocPat variants, we plot the average running among all users. For Guha, EigenTrust, and Fouss, the running time for each user is identical because for each user, these approaches consider the whole network. EigenTrust is the most efficient approach, followed by LocPat and Fouss. Guha is the worst because of its slow convergence. EigenTrust requires only matrix-vector multiplication because it only calculates one value for each node. The other approaches require matrix multiplication to calculate one value for each pair of nodes.

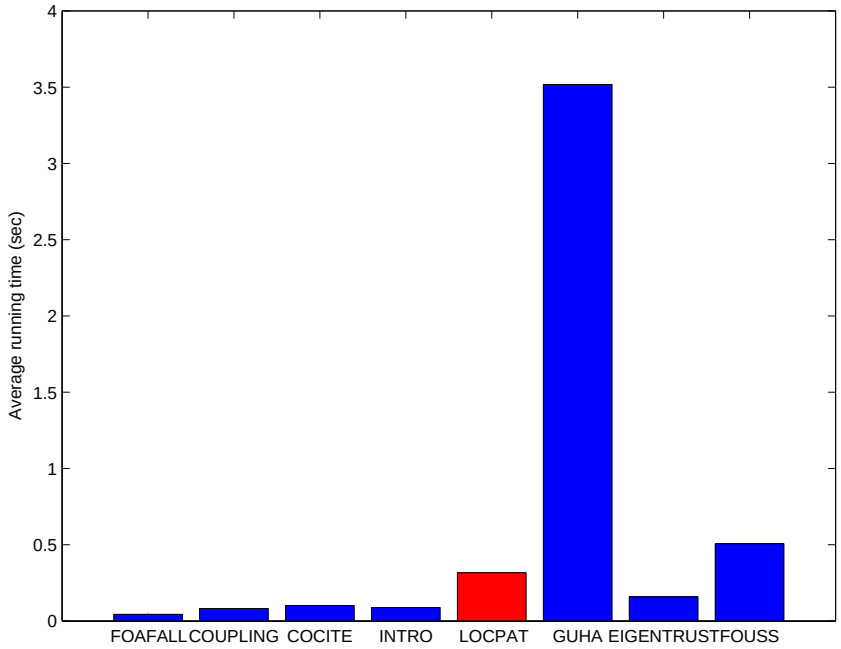


Fig. 20 Running time in FilmTrust.

Summary

In terms of effectiveness, our experiments show that LocPat produces effective recommendations in four independently developed datasets involving real users. LocPat outperforms (better or similar) three other approaches. Especially considering that LocPat only requires a local neighborhood network, whereas the other approaches need the whole graph as input. We also show that the proposed local rounding provides more accurate predictions than the global heuristics. In terms of flexibility and applicability, we show that LocPat can be devised as different variants by using different structure graphs. By using appropriate variants, LocPat can provide comprehensive recommendations and predictions in social networks and ratings networks, for which traditional friend-of-a-friend propagation is not applicable. We also show that LocPat is efficient. Another contribution is that through these variants, LocPat can guide agents to understand the underlying relationships in a certain dataset and to use the appropriate structure graphs for generating desirable recommendations. For example, in social networks, besides the widely used idea of friend-of-a-friend, LocPat suggests using the coupling and introduction patterns. In ratings networks, LocPat with cocitation is the most appropriate pattern for recommending items to people.

5 Conclusions

We present LocPat, a personalized recommendation approach that provides recommendations to a requester in an agent network. Our approach is built on a

vertex similarity measurement between graphs. The similarity measurement is defined by a mutual reinforcing relation. We show that by calculating the similarity between the agent network and a structure graph, we can view the similarity score as an indicator of how much the agent can be trusted by the requester. We also provide a simple rounding method that can translate similarity scores into trust value predictions. The LocPat approach offers important benefits over traditional approaches. First, LocPat provides flexible personalized recommendation from the perspective of an agent. Second, LocPat supports customization through the definition of suitable structure graphs. Third, LocPat only requires a local subgraph of the agent network. Fourth, LocPat is computationally efficient.

A possible future direction is to study how LocPat can be extended to provide different recommendations. For example, Hang and Singh [11] design a personalized service composition model for estimating trustworthiness of the subservices underlying a composition. However, their model fails to provide a mechanism for selecting the subservices—recommending compositions. Based on LocPat, we should be able to construct a structure graph that satisfies their scenario.

Acknowledgements This work is supported by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI), and by the Army Research Laboratory in its Network Sciences Collaborative Technology Alliance (NS-CTA) under Cooperative Agreement Number W911NF-09-2-0053.

References

1. Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. Journal of Web Semantics **5**(2), 58–71 (2007)
2. Ben-Shimon, D., Tsikinovsky, A., Rokach, L., Meisels, A., Shani, G., Naamani, L.: Recommender system from personal social networks. In: Proceedings of the 5th Atlantic Web Intelligence Conference, pp. 47–55. Springer Berlin / Heidelberg (2007)
3. Blondel, V.D., Gajardo, A., Heymans, M., Senellart, P., Dooren, P.V.: A measure of similarity between graph vertices: Applications to synonym extraction and web searching. SIAM Review **46**(4), 647–666 (2004)
4. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**(1–7), 107–117 (1998)
5. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent mining: The synergy of agents and data mining. IEEE Intelligent Systems **24**(3), 64–72 (2009)
6. Fouss, F., Pirotte, A., Renders, J.M., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge and Data Engineering **19**(3), 355–369 (2007)
7. Giannella, C., Bhargava, R., Kargupta, H.: Multi-agent systems and distributed data mining. In: M. Klusch, S. Ossowski, V. Kashyap, R. Unland (eds.) Cooperative Information Agents VIII, *Lecture Notes in Computer Science*, vol. 3191, pp. 1–15. Springer Berlin / Heidelberg (2004)
8. Golub, G.H., Loan, C.F.V.: Matrix Computations, 3 edn. The Johns Hopkins University Press (1996)
9. Gray, E., Seigneur, J.M., Chen, Y., Jensen, C.: Trust propagation in small worlds. In: Proceedings of the 1st International Conference on Trust Management, pp. 239–254. Springer-Verlag, Berlin, Heidelberg (2003)
10. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: WWW: Proceedings of the 13th International Conference on World Wide Web, pp. 403–412. ACM Press (2004)
11. Hang, C.W., Singh, M.P.: Trustworthy service selection and composition. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **6**(1), 5:1–5:17 (2011)

12. Hang, C.W., Wang, Y., Singh, M.P.: Operators for propagating trust and their evaluation in social networks. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), vol. 2, pp. 1025–1032. IFAAMAS, Budapest (2009)
13. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543. ACM Press, New York, NY, USA (2002)
14. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: WWW: Proceedings of the 12th International Conference on World Wide Web, pp. 640–651. ACM Press (2003)
15. Katz, Y., Golbeck, J.: Social network-based trust in prioritized default logic. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), pp. 1345–1350. AAAI Press, Menlo Park (2006)
16. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46**(5), 604–632 (1999)
17. Klusch, M., Lodi, S., Moro, G.: The role of agents in distributed data mining: Issues and benefits. In: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology, pp. 211–217. IEEE Computer Society, Washington, DC, USA (2003)
18. Kunegis, J., Lommatzsch, A.: Learning spectral graph transformations for link prediction. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 561–568. ACM Press, New York, NY, USA (2009)
19. Kuter, U., Golbeck, J.: Using probabilistic confidence models for trust inference in web-based social networks. ACM Transactions on Internet Technology (TOIT) **10**(2), 1–23 (2010)
20. Leicht, E.A., Holme, P., Newman, M.E.J.: Vertex similarity in networks. Physical Review E **73**, 026,120 (2006)
21. Levien, R.: Attack resistant trust metrics. Ph.D. thesis, UC Berkeley (2003)
22. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7), 1019–1031 (2007)
23. Lorrain, F., White, H.C.: Structural equivalence of individuals in social networks. Journal of Mathematical Sociology **1**, 49–80 (1971)
24. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering, pp. 117–128. IEEE Computer Society, Washington, DC, USA (2002)
25. Miller, B.N., Albert, I., Lam, S.K., Konstan, J.A., Riedl, J.: MovieLens unplugged: Experiences with an occasionally connected recommender system. In: Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI), pp. 263–266. ACM Press, New York, NY, USA (2003)
26. Moemeng, C., Gorodetsky, V., Zuo, Z., Yang, Y., Zhang, C.: Agent-based distributed data mining: A survey. In: L. Cao (ed.) Data Mining and Multi-agent Integration, chap. 3, pp. 47–58. Springer (2009)
27. Nathanson, T., Bitton, E., Goldberg, K.: Eigentaste 5.0: Constant-time adaptability in a recommender system using item clustering. In: Proceedings of the ACM Conference on Recommender Systems, pp. 149–152. ACM Press, New York, NY, USA (2007)
28. Quercia, D., Hailes, S., Capra, L.: Lightweight distributed trust propagation. In: Proceedings of the 7th IEEE International Conference on Data Mining (ICDM), pp. 282–291 (2007)
29. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic Web. In: The Semantic Web: Proceedings of the 2nd International Semantic Web Conference (ISWC), *LNCS*, vol. 2870, pp. 351–368. Springer (2003)
30. Shani, G., Chickering, M., Meek, C.: Mining recommendations from the web. In: Proceedings of the ACM Conference on Recommender Systems, pp. 35–42. ACM Press, New York, NY, USA (2008)
31. Tavakolifard, M.: Similarity-based techniques for trust management. In: Z.U.H. Usmani (ed.) Web Intelligence and Intelligent Agents, chap. 11, pp. 233–250. InTech (2010)
32. Wang, Y., Singh, M.P.: Trust representation and aggregation in a distributed agent system. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), pp. 1425–1430. AAAI Press, Boston, MA, USA (2006)
33. Yu, B., Singh, M.P.: Distributed reputation management for electronic commerce. Computational Intelligence **18**(4), 535–549 (2002)

34. Ziegler, C.N., Lausen, G.: Spreading activation models for trust propagation. In: EEE: Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service, pp. 83–97. IEEE Computer Society, Washington, DC, USA (2004)