The Journal of Systems and Software 85 (2012) 2119–2132

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Fast and accurate link prediction in social networking systems☆

Alexis Papadimitriou*, Panagiotis Symeonidis, Yannis Manolopoulos

Department of Informatics, Aristotle University of Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 26 December 2011
Received in revised form 9 April 2012
Accepted 10 April 2012
Available online 20 April 2012

Keywords:
Link prediction
Friend recommendation
Social networks

ABSTRACT

Online social networks (OSNs) recommend new friends to registered users based on local-based features of the graph (i.e. based on the number of common friends that two users share). However, OSNs do not exploit all different length paths of the network. Instead, they consider only pathways of maximum length 2 between a user and his candidate friends. On the other hand, there are global-based approaches, which detect the overall path structure in a network, being computationally prohibitive for huge-sized social networks. In this paper we provide friend recommendations, also known as the *link prediction problem*, by traversing all paths of a limited length, based on the "algorithmic small world hypothesis". As a result, we are able to provide more accurate and faster friend recommendations. We also derive variants of our method that apply to different types of networks (directed/undirected and signed/unsigned). We perform an extensive experimental comparison of the proposed method against existing link prediction algorithms, using synthetic and three real data sets (Epinions, Facebook and Hi5). We also show that a significant accuracy improvement can be gained by using information about both positive and negative edges. Finally, we discuss extensively various experimental considerations, such as a possible MapReduce implementation of FriendLink algorithm to achieve scalability.

1. Introduction

Online social networks (OSNs) such as Facebook.com, Myspace.com, Hi5-.com, etc. contain gigabytes of data that can be mined to make predictions about who is a friend of whom. OSNs gather information on users' social contacts, construct a large interconnected social network, and recommend other people to users based on their common friends. The premise of these recommendations is that individuals might only be a few steps from a desirable social friend, but not realize it.

In this paper, which is an extension of our previously published work in Papadimitriou et al. (2011), we focus on recommendations based on links that connect the nodes of an OSN, known as the *Link Prediction* problem, where there are two main approaches that handle it (Liben-Nowell and Kleinberg, 2003). The first one is based on local features of a network, focusing mainly on the nodes structure; the second one is based on global features, detecting the overall path structure in a network. For instance, an example of a local-based approach is shown in Fig. 1. Facebook.com or Hi5.com use the following style of recommendation for recommending new friends to a target user U_1: "People you may know: (i) user U_7 because you have two common friends (user U_5 and user U_6) (ii) user U_9 because you have one common friend (user U_8) ...". The list of recommended friends is ranked based on the number of common friends each candidate friend has with the target user.

1.1. Motivation

Compared to approaches which are based on local-based features of a network, we expand user's neighborhood horizon by exploiting paths of greater length. In contrast, they consider only pathways of maximum length 2 between a target user and his candidate friends. In our approach, we assume that a person can be connected to another with many paths of different length (through human chains). For example, in Fig. 1, according to existing OSNs, U_1 would get as friend recommendation with equal probability U_4 or U_7. However, if we take into account also paths of length 3, then U_4 should have a higher probability to be recommended as a friend to U_1. Compared to global-based approaches, which detect the overall path structure in a network, our method is more efficient. This means, that our method, which is based on a limited path traversal, requires less time and space complexity than the global based algorithms. The reason is that we traverse only paths of length ℓ in a network based on the "algorithmic small world hypothesis", whereas global-based approaches detect the overall path structure.

☆ A preliminary version of this paper entitled "Predicting Links in Social Networks of Trust via Bounded Local Path Traversal" has been presented at the 3rd Conference on Computational Aspects of Social Networks (CASON'2011).

* Corresponding author.
E-mail address: apapadi@csd.auth.gr (A. Papadimitriou).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.04.019

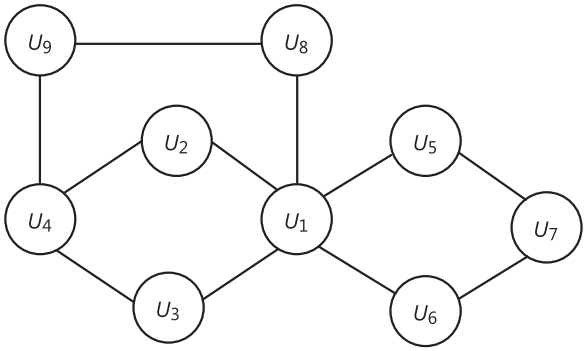

Fig. 1. Social network example.

1.2. Contribution

The contributions of our approach are summarized as follows: (i) We define a new node similarity measure that exploits local and global characteristics of a network. (ii) We provide more accurate friend recommendations, by traversing paths of different length that connect a person to all other persons in an OSN. (iii) We provide higher efficiency than the global-based approaches, by limiting our traversing in ℓ-length paths in a network. (iv) We also derive variants of our method that apply to different types of networks (directed/undirected and signed/unsigned). We show that a significant accuracy improvement can be gained by using information about both positive and negative edges. (v) To run our algorithm with huge sized networks, we discuss its possible MapReduce (Dean and Ghemawat, 2008) implementation. Note that this paper is an extension of our previously published work in Papadimitriou et al. (2011).

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 briefly reviews preliminaries in graphs employed in our approach. Section 4 defines a new node similarity measure in OSNs. A motivating example, the proposed approach, its complexity analysis, and the extension of FriendLink for different types of networks, i.e. signed networks, are described in Section 5. Experimental results are given in Section 6. Also, in Section 7 we discuss the scalability of our method by proposing a possible MapReduce implementation. Finally, Section 8 discusses basic research questions, whereas Section 9 concludes this paper.

2. Related work

Based on his provocative "small world" experiments, Stanley Milgram claimed that everyone in the world could be connected to everyone else via an average small path length (Milgram, 1967). This experiment is also known as the "six degrees of separation", although Milgram did not use this term himself. Recently, Goel et al. (2009) reported experiments for the "algorithmic small-world hypothesis", where half of all chains can be completed in 6–7 steps, supporting the "six degrees of separation" assertion. However, they report that the number of steps in a search chain depends not only on the actual distance between the source and the target, but also on the search strategies of the intermediaries.

The research for link prediction in social networks, tries to infer new interactions among members of a social network that are likely to occur in the near future. There are two main approaches (Liben-Nowell and Kleinberg, 2003) to handle the *link prediction* problem. The first one is based on local features of a network, focusing mainly on the nodes structure; the second one is based on global features, detecting the overall path structure in a network.

Tylenda et al. (2009) proposed methods to incorporate temporal information available on evolving social networks for link prediction. Schifanella et al. (2010) discover the connection between the usage of shared tags and the social links existing between users. When they considered the annotations of the most active users, almost all of the semantic similarity measures considered outperform the neighbor suggestions from the Last.fm system at predicting actual friendship relations. Zheleva et al. (2008) study the predictive power of overlaying friendship and family ties on three real-world social networks. Zhou et al. (2009) and Lü et al. (2009) propose a similar idea to our own but they do not include any attenuation factor experimentation, path normalization or a possible MapReduce implementation scenario to support huge sized networks.

There is a variety of local-based similarity measures (Liben-Nowell and Kleinberg, 2003), which are node-dependent (i.e. Common Neighbors index or else known as Friend of a Friend (FOAF) algorithm, Adamic/Adar index, Jaccard Coefficient, etc.) for analyzing the "proximity" of nodes in a network. FOAF (Chen et al., 2009) is based on the common sense that two nodes v_x, v_y are more likely to form a link in the future, if they have many common neighbors. More complicated local-based measures such as Jaccard Coefficient (Liben-Nowell and Kleinberg, 2003) and Adamic/Adar index (Adamic and Adar, 2005), refine the simple counting of common features by weighting rarer features more heavily. Another well-known local-based similarity measure is Preferential Attachment (Liben-Nowell and Kleinberg, 2003). The basic premise of Preferential Attachment (PA) is that the probability a new edge involves a node is proportional to the current number of its neighbors.

There is a variety of global-based approaches (Liben-Nowell and Kleinberg, 2003) which are path-dependent (i.e. Katz status index, RWR algorithm, SimRank algorithm, etc.). Katz (1953) introduced a status index that computes the important and influential nodes in a social network. Random Walk with Restart (RWR) algorithm (Pan et al., 2004) is based on a Markov chain model of random walks through a graph. In the same direction with RWR, the Markov Diffusion (MD) kernel (Fouss et al., 2006) is based on a discrete-time diffusion Markov model. Moreover, Fouss et al. (2007) and Fouss et al. (2012) proposed a random walk model that computes matrix kernels (i.e. the average commute time, the regularized commute time (RCT), etc.) to capture similarities between any pair of nodes in a network. These matrix kernels have the property of increasing, when the number of paths connecting two nodes increases and when the length of connecting paths decreases. Furthermore, Sarkar and Moore (2007) proposed a truncated commute time random walk model to compute all "interesting" pairs of approximate nearest neighbors in truncated commute times, without computing it between all pairs. Notice that truncated commute time (Sarkar and Moore, 2007) algorithm shares a similar idea with our method, but it questionable that it has not yet been compared with other state-of-art link prediction algorithms. SimRank (Jeh and Widom, 2002) also computes a global-based similarity measure based on the structural context of a network that says "two objects are similar if they are related to similar objects". Finally, proposed an algorithm based on the hierarchical network structure.

The novelty of our approach compared to existing approaches is as follows: (i) Our method can be categorized as a local-based similarity measure, because it relies on a truncated strategy of counting paths in a graph. We compare our method against FOAF, Adamic/Adar and PA algorithms, as representative of the local-based measures, and as will be experimentally shown later, our method outperforms the other methods in terms of accuracy. The reason is that we take into account more information by expanding the user's neighborhood horizon. (ii) In contrast to global-based algorithms, our method is more efficient, because it is based on a

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9
U_1	0	1	1	0	1	1	0	1	0
U_2	1	0	0	1	0	0	0	0	0
U_3	1	0	0	1	0	0	0	0	0
U_4	0	1	1	0	0	0	0	0	1
U_5	1	0	0	0	0	0	1	0	0
U_6	1	0	0	0	0	0	1	0	0
U_7	0	0	0	0	1	1	0	0	0
U_8	1	0	0	0	0	0	0	0	1
U_9	0	0	0	1	0	0	0	1	0

Fig. 2. Adjacency matrix $\mathcal{A}$ of graph $\mathcal{G}$.

local-based similarity measure. Thus, it requires less time and space complexity than global based algorithms. We have compared our method against RWR, Katz, MD and RCT, among others, as representatives of the global based algorithms, and our method outperforms all these methods. The reason is that global methods have to compute the inverse of a $n \times n$ matrix (n is number of vertices in a network) resulting to $O(n^3)$ time complexity, whereas our method requires a linear CPU time to the network size n. Moreover, our method is more effective in terms of accuracy. The reason is that global methods traverse globally the social network, missing to capture adequately the local characteristics of the graph.

3. Preliminaries in graphs

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a set $\mathcal{V}$ of vertices and a set $\mathcal{E}$ of edges such that an edge joins a pair of vertices. In this paper, $\mathcal{G}$ will always be a general undirected and unvalued graph as shown in Fig. 1. $\mathcal{G}$ expresses friendships among users of an OSN and will be used as our running example, throughout the paper.

The adjacency matrix $\mathcal{A}$ of graph $\mathcal{G}$ is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are friends or not. For an undirected graph, the adjacency matrix is symmetric. In Fig. 2, we present the resulting adjacency matrix $\mathcal{A}$ of graph $\mathcal{G}$.

The adjacency matrix of graph $\mathcal{G}$ when raised to the power of 2 results in the matrix shown in Fig. 3, which presents the number of length-2 paths that exist between each pair of graph nodes. In our running example, as shown in Fig. 3, node U_1 has 2 length-2 paths connecting him to U_4 and U_7, and 1 length-2 paths connecting him to U_9. Notice that by raising the adjacency matrix $\mathcal{A}$ to the power of 3, we get the number of length-3 paths between each pair of nodes in $\mathcal{G}$. This process can be repeated for higher powers.

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9
U_1	5	0	0	2	0	0	2	0	1
U_2	0	2	2	0	1	1	0	1	1
U_3	0	2	2	0	1	1	0	1	1
U_4	2	0	0	3	0	0	0	1	0
U_5	0	1	1	0	2	2	0	1	0
U_6	0	1	1	0	2	2	0	1	0
U_7	2	0	0	0	0	0	2	0	0
U_8	0	1	1	1	1	1	0	2	0
U_9	1	1	1	0	0	0	0	0	2

Fig. 3. Adjacency matrix $\mathcal{A}$ of graph $\mathcal{G}$ raised to the power of 2.

4. Defining a node similarity measure

In this section, we define a new similarity measure to determine a way of expressing the proximity among graph nodes. Let v_i and v_j be two graph nodes and $sim(v_i, v_j)$ a function that expresses their similarity. The higher the similarity score between two nodes, the higher the possibility of them being friends.

Suppose that two persons in an OSN want to have a relationship, but the shortest path between them is blocked by a reluctant broker. If there exists another pathway, the two persons are likely to use it, even if it is longer and "less efficient". In general, two persons can use all the pathways connecting them, rather than just the shortest path between them. Thus, our method expands the idea of shortest paths connection between two persons in an OSN.

By traversing all possible paths between a person and all other persons in an online social graph, a person can be connected to another by many possible paths (through human chains). Our method assumes that persons in an OSN can use all the pathways connecting them, proportionally to the pathway lengths. Thus, two persons who are connected with many unique pathways have a high possibility to know each other, proportionally to the length of the pathways they are connected with.

For example, referring back to Fig. 1, if we consider only length-2 paths, then U_1 would get as friend recommendation with equal probability U_4 or U_7. However, if we take into account also length-3 paths, then U_4 should have a higher probability to be recommended as a friend to U_1.

Definition 1. The similarity $sim(v_x, v_y)$ between two graph nodes v_x and v_y is defined as the counts of paths of varying length ℓ from to v_x and v_y:

$$sim(v_x, v_y) = \sum_{i=2}^{\ell} \frac{1}{i-1} \cdot \frac{\left|paths^{i}_{v_x, v_y}\right|}{\prod_{j=2}^{i}(n-j)} \tag{1}$$

where

- n is the number of vertices in a graph G,
- ℓ is the maximum length of a path taken into consideration between the graph nodes v_x and v_y (excluding paths with cycles). By the term "paths with cycles" we mean that a path cannot be closed (cyclic). Thus, a node can exist only one time in a path (e.g. path $v_1 \to v_2 \to v_3 \to v_1 \to v_5$ is not acceptable because v_1 is traversed twice),
- $1/(i-1)$ is an "attenuation" factor that weights paths according to their length ℓ. Thus, a 2-step path measures the non-attenuation of a link with value equals to 1 ($1/(2-1) = 1$). A 3-step path measures the attenuation of a link with value equals to 1/2 ($1/(3-1) = 1/2$), etc. In this sense, we use appropriate weights to allow the lower effectiveness of longer path chains. Notice that we have also tested experimentally other possible attenuation factors such as Katz's original exponential β^{ℓ}, the logarithmic $1/log(i)$, etc. and as will be shown later the attenuation factor $1/(i-1)$ attains the best accuracy results.
- $\left|paths^{\ell}_{v_x, v_y}\right|$ is the number of all length-ℓ paths from v_x to v_y,
- $\prod_{j=2}^{i}(n-j)$ is the number of all possible length-ℓ paths from v_x to v_y, if each vertex in graph G was linked with all other vertices. Notice that, we do not count all paths of length-ℓ that lead from all users to every other user in the social graph.

	U_1	U_2	U_3	U_4	U_5-U_6	U_7	U_8	U_9
U_1	-	1→3→4→2	1→2→4→3	**1→2→4** **1→3→4** **1→8→9→4**	…	**1→5→7** **1→6→7**		1→8→9 1→2→4→9 1→3→4→9
U_2	2→4→3→1	-	2→1→3 2→4→3	2→1→3→4	…	2→1→5→7 2→1→6→7	2→1→8 2→4→9→8	2→4→9 2→1→8→9
U_3	3→4→2→1	3→1→2 3→4→2	-	3→1→2→4	…	3→1→5→7 3→1→6→7	3→1→8 3→4→9→8	3→4→9 3→1→8→9
U_4	4→2→1 4→3→1 4→9→8→1	4→3→1→2	4→2→1→3	-	…		4→9→8 4→2→1→8 4→3→1→8	
U_5	5→7→6→1	5→1→2	5→1→3	5→1→2→4 5→1→3→4	…	5→1→6→7	5→1→8	5→1→8→9
U_6	6→7→5→1	6→1→2	6→1→3	6→1→2→4 6→1→3→4	…	6→1→5→7	6→1→8	6→1→8→9
U_7	7→5→1 7→6→1	7→5→1→2 7→6→1→2	7→5→1→3 7→6→1→3		…	-	7→5→1→8 7→6→1→8	
U_8		8→1→2 8→9→4→2	8→1→3 8→9→4→3	8→9→4 8→1→2→4 8→1→3→4	…	8→1→5→7 8→1→6→7	-	
U_9	9→8→1 9→4→2→1 9→4→3→1	9→4→2 9→8→1→2	9→4→3 9→8→1→3		…			-

Fig. 4. User matrix that contains all paths of length 2 and 3 in graph $\mathcal{G}$ of our running example.

Finally, the similarity is computed for nodes that are connected with paths of length $\ell \geq 2$. This is because when there is a path between two nodes of length 1 they are already friends.

5. The proposed approach

In this section, through a motivating example we first provide the outline of our approach, named FriendLink. Next, we analyze the steps of the proposed algorithm.

5.1. Outline

Our FriendLink approach finds similarities between nodes in an undirected graph constructed from the connection data. The FriendLink algorithm uses as input the connections of a graph $\mathcal{G}$ and outputs a similarity matrix between any two nodes in $\mathcal{G}$. Therefore, friends can be recommended to a target user u according to their weights in the similarity matrix. In the following, to illustrate how our approach works, we apply the FriendLink algorithm to our running example. As illustrated in Fig. 1, 9 users are connected in a graph.

If we have to recommend a new friend to U_1, then there is no direct indication for this task in the original adjacency matrix $\mathcal{A}$, as shown in Fig. 2. However, after performing the FriendLink algorithm, we can get a similarity matrix between any two nodes of graph $\mathcal{G}$ and recommend friends according to their weights.

Firstly, we modify the adjacency matrix $\mathcal{A}$ so that instead of holding 0/1 values, the (i, j) entry of the matrix $\mathcal{A}$ is a list of paths from i to j. The idea is that, if you have the 0/1 adjacency matrix of a graph, and you raise that matrix to the Nth power, then the (v_i, v_j) entry of the result shows how many length-N paths exist from node v_i to node v_j (here the length is measured in the number of traversed edges). Then, instead of just counting the paths, we keep track of all the actual paths themselves. Then, we perform matrix multiplication of the modified adjacency matrix with itself but, instead of multiplying and adding entries, we produce all paths from node v_i to node v_j. As shown in Fig. 4, we have created all paths of length 2 and 3, which connect each node of graph $\mathcal{G}$ to every other graph node. Notice that paths containing loops are excluded.

Next, we update the similarity between nodes v_i and v_j, for each produced length-ℓ path, where v_i is the start node and v_j is the destination node (i.e. all paths of length $[2 \ldots \ell]$). For the calculation of the similarity value between nodes v_i and v_j we use Eq. (1). In our running example, suppose we calculate the similarity between U_1 with U_4 and U_7, respectively. Firstly, as shown in Fig. 4, the similarity between U_1 and U_4 is computed based on the three paths that connect them (1→2→4, 1→3→4, and 1→8→9→4). According to Eq. (1), each of the paths 1→2→4 and 1→3→4 corresponds to a weight of 0.1428 (1 path of length-2 that connects the two nodes divided to the 7 possible paths of length-2 that could exist between them and this ratio is multiplied with an attenuation factor equal to 1), while path 1→8→9→4 corresponds to a weight of 0.0119 (1 path of length-3 that connects the two nodes divided to the 42 possible paths of length-3 that could exist between them and this ratio is multiplied with an attenuation factor equal to 0.5). Thus, the total similarity between U_1 and U_4 equals to 0.2975 (0.1428 + 0.1428 + 0.0119). Secondly, as shown also in Fig. 4, there are two paths (1→5→7 and 1→6→7) that connect U_1 with U_7. The weight of each path is again 0.1428. The total similarity between U_1 and U_7 equals to 0.2856 (0.1428 + 0.1428). Notice that the weight that corresponds to each path of length ℓ is computed as the ratio between the existed paths of length ℓ to the total possible paths of length ℓ, which are calculated by the denominator of Eq. (1).

In Fig. 5, we present the node similarity matrix of graph $\mathcal{G}$. Therefore, new friends can be recommended according to their total weight, which is computed by aggregating all paths connecting them with the target user, proportionally to the length of each path.

In our running example, as shown in Fig. 5, user U_1 would receive user U_4 as friend recommendation. The resulting recommendation is reasonable, because U_1 is connected with more paths to user U_4 than those that connect U_1 and U_7. That is, the FriendLink approach is able to capture the associations among the graph data objects. The associations can then be used to improve the friend recommendation procedure, as will be verified by our experimental results.

5.2. The FriendLink algorithm

In this section, we describe our FriendLink algorithm in detail. Our Friendlink algorithm computes node similarity between any

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9
U_1	0	0	0	**0.2975**	0	0	**0.2856**	0	0.167
U_2	0	0	0.286	0	0146	0.146	0.024	0.156	0.156
U_3	0	0.286	0	0	0.146	0.146	0.024	0.156	0.156
U_4	0.298	0	0	0	0.025	0.025	0	0.167	0
U_5	0	0.146	0.146	0.025	0	0.286	0	0.144	0.015
U_6	0	0.146	0.146	0.025	0.286	0	0	0.144	0.015
U_7	0.286	0.024	0.024	0	0	0	0	0.024	0
U_8	0	0.156	0.156	0.167	0.144	0.144	0.024	0	0
U_9	0.167	0.156	0.156	0	0.015	0.015	0	0	0

Fig. 5. Node similarity matrix. It presents the possibility of two users being friends.

two nodes in a graph $\mathcal{G}$. The initial input of Friendlink is the number n of nodes of $\mathcal{G}$, the adjacency matrix A, and the length ℓ of paths that will be explored in $\mathcal{G}$. To enumerate all simple paths in $\mathcal{G}$, Rubin's algorithm (Rubin, 1978) can be employed. However, Rubin's algorithm uses $O(n^3)$ matrix operations to find all paths of different length between any pair of nodes, where n is the number of nodes in $\mathcal{G}$. In the following, we customize Rubin's algorithm to create only paths of length up to ℓ for our purpose.

As shown in Fig. 6, our FriendLink algorithm consists of a main program and two functions. In the main program, we modify the adjacency matrix so instead of holding 0/1 values, the (i, j) entry of the matrix A is a list of paths from i to j. Then, in the function Combine Paths(), we perform the matrix multiplication algorithm. However, instead of multiplying and adding entries, we concatenate pairs of paths together. Notice that, for simplicity reasons, we do not include the code for loop removals in Fig. 6. Finally, in the function Compute Similarity(), we update the similarity between nodes i and j, for each length-ℓ path we find, where i is the start node and j is the destination node (i.e. all paths of length $[2 \ldots \ell]$). For the update of the similarity value between nodes i and j we use Eq. (1). Notice that, we do not take into account cyclic paths in our similarity measure.

5.3. Complexity analysis

Social networks are large and contain a significant amount of information. Global based algorithms that can be used for link prediction and friend recommendation, such as Random Walk with Restart(RWR) (Tong et al., 2006; Pan et al., 2004), Katz index (Katz, 1953), the Markov Diffusion (MD) kernel (Fouss et al., 2006) and the Regularized Commute Time (RCT) (Fouss et al., 2007, 2012) are computationally prohibitive for large graphs, because they require the inversion of a matrix. For instance, the time complexity of Katz index is mainly determined by the matrix inversion operator, which is $O(n^3)$. There is also a faster version (Foster et al., 2001) of Katz status index that reduces computational complexity from time $O(n^3)$ to $O(n+m)$, where m is the number of edges. RWR algorithm also requires a matrix inversion, which can be pre-computed and stored for faster on-line operations. This choice is fast on query time, but requires additional storage cost (i.e. quadratic space on the number of nodes on the graph). A solution to this is that, the matrix inversion can be computed on the fly, through power iteration. However, its on-line response time is linear to the iteration number and the number of edges. Notice that Tong et al. (2006) proposed a faster version of RWR. However, it is less accurate than the original RWR, which is not an adequate solution to the friend recommendation problem, where accuracy is one of the most important parameters.

Friend of a Friend algorithm (FOAF), as a representative of the local-based methods, considers very small paths (only paths of length 2) between any pair of nodes in $\mathcal{G}$. In particular, for each v_x node, FOAF traverses all its neighbors and then traverses the neighbors of each of v_x's neighbor. Since the time complexity to traverse the neighborhood of a node is simply h (h is the average nodes degree in a network) and our graph $\mathcal{G}$ is sparse, it holds that $h << n$. Thus, the time complexity of FOAF is $O(n \times h^2)$. The space complexity for FOAF is $O(n \times h)$.

Our FriendLink algorithm considers also paths with higher length (l-length paths). Based on Milgram's, 1967 "small-world hypothesis", l can take integer values in the interval [2,6], where for $l = 2$ our FriendLink equals to the FOAF algorithm. Thus, FriendLink's time complexity is $O(n \times h^l)$. The space complexity for FriendLink is also $O(n \times h)$. Notice that in our code we store adjacent nodes using adjacency lists and not a matrix structure. However, for simplicity reasons, in Fig. 6 we present our algorithm using a matrix structure.

5.4. Extending FriendLink for different types of networks

Until this point in our paper analysis, we dealt with un-weighted and undirected networks. However, our algorithm can be easily extended to different types of networks. In this section, we derive variants of FriendLink that apply to directed networks and networks with weighted edges, including the case of edges with negative weights (signed networks). Applying FriendLink to directed graphs can be achieved (i) by simply disregarding the edge directions (Wasserman and Faust, 1994), (ii) or by replacing the original adjacency matrix A with an asymmetric one.

For weighted networks, if edges weights are all positive, FriendLink applies trivially. In some networks, however, edges have positive as well as negative weights. Such signed graphs arise for instance in social networks (i.e. Epinions.com, Shashdot Zoo, etc.) where negative edges denote enmity instead of friendship. In such signed graphs, FriendLink's equation (1), which is based on the adjacency matrix, can be interpreted as weighted sums of powers of the adjacency matrix which denote path count in the network. Thus, if some edges have negative weight, the total weight of a path is counted as the product of the edges's weights, based on the assumption of multiplicative transitivity of the structural balance theory (Hage and Harary, 1983; Leskovec et al., 2010), as formulated in the graph-theoretic language by Hage and Harary (1983).

Structural balance theory considers the possible ways in which triangles on three individuals can be signed. Triangles with three positive signs exemplify the principle that "the friend of my friend is my friend", whereas those with one positive and two negative edges capture the notions "the enemy of my friend is my enemy", "the friend of my enemy is my enemy", and the "enemy of my enemy is my friend".

6. Experimental evaluation

In this section, we compare experimentally our FriendLink algorithm with 8 other link prediction algorithms. In particular, we use in the comparison the Markov diffusion kernel (Fouss et al., 2006), the Regularized commute-time kernel (Fouss et al., 2012), the Random Walk with Restart (Pan et al., 2004) algorithm, the Katz (1953) status index, the Adamic and Adar (2005), the Preferential Attachment (Newman, 2001), the Friend of a Friend (Chen et al., 2009)

Algorithm FriendLink ($\mathcal{G}$, A, n, ℓ)
Input
$\mathcal{G}$: an undirected and unweighted graph
A: adjacency matrix of graph $\mathcal{G}$,
n: number of nodes of graph $\mathcal{G}$,
ℓ: maximum length of paths explored in $\mathcal{G}$,
i: the length of a path
Output
$sim(v_i, v_j)$: similarity between node v_i and node v_j in $\mathcal{G}$

1. **Main Program**
2. **for** $v_i = 1$ to n
3. **for** $v_j = 1$ to n
4. **if** $A(v_i, v_j) = 1$ then
5. $A(v_i, v_j) = v_j$
6. **else**
7. $A(v_i, v_j) = 0$
8. **end if**
9. **end for** v_j
10. **end for** v_i
11. **for** $i = 2$ to ℓ
12. Combine Paths()
13. Compute Similarity(i)
14. **end for** i
15. **End Main Program**

16. **Function** Combine Paths()
17. **for** $v_i = 1$ to n
18. **for** $v_j = 1$ to n
19. **for** $k = 1$ to n
20. **if** $A(v_i, k) <> 0$ and $A(k, v_j) <> 0$ then
21. $A(v_i, v_j) = concatenate(A(v_i, k), A(k, v_j))$
22. **end if**
23. **end for** k
24. **end for** v_j
25. **end for** v_i
26. **return** $A(v_i, v_j)$
27. **End Function**

28. **Function** Compute Similarity()
29. **for** $v_i = 1$ to n
30. **for** $v_j = 1$ to n
31. $denominator = 1$
32. **for** $k = 2$ to i
33. $denominator = denominator$ * (n - k)
34. **end for** k
35. $sim(v_i, v_j) = sim(v_i, v_j) + \frac{1}{i-1} \cdot \frac{\left|paths^{i}_{v_i, v_j}\right|}{denominator}$
36. **end for** v_j
37. **end for** v_i
38. **return** $sim(v_i, v_j)$
39. **End Function**

Fig. 6. The FriendLink algorithm.

and the Shortest Path (Fredman and Tarjan, 1987) algorithm. Our experiments were performed on a 3 GHz Pentium IV, with 2 GB of memory, running Windows XP. All algorithms were implemented in Matlab.

6.1. Algorithms settings

In this following, we present basic information of the algorithms that will be compared experimentally with our proposed method:

The Markov Diffusion kernel: The Markov Diffusion kernel (Fouss et al., 2006) is a distance measure between nodes of a graph. It is based on a discrete-time diffusion Markov model, where an initial state starts from a node v_x and reaches a node v_y after t time steps. The similarity matrix (i.e. Kernel) between nodes of a graph, can be computed by Eq. (2):

$$Kernel_{MD}(t) = (e_{v_x} - e_{v_y})^T \cdot Z_t \cdot Z_t^T \cdot (e_{v_x} - e_{v_y}) \quad (2)$$

with $Z_t = (1/t) \cdot (I - P)^{-1} \cdot (I - P^t) \cdot P$, where I is the identity matrix and P is the transition-probability matrix. Notice that $P = D^{-1}A$, where D is a diagonal matrix containing the outdegrees of the graph nodes. Moreover, e_{v_x} and e_{v_y} are the column vectors of nodes v_x and v_y, respectively.

The Regularized Commute-Time kernel: The Regularized Commute-Time kernel (Fouss et al., 2012) performs a regularization on the commute-time kernel (Fouss et al., 2007). Thus, instead of taking the pseudoinverse of the Laplacian matrix (i.e. L^+), which is not invertible, a simple regularization framework is applied that replaces L^+ with $D - \alpha A$. The similarity matrix (i.e. Kernel) between nodes of a graph, can be computed by Eq. (3):

$$Kernel_{RCT} = (D - \alpha A)^{-1} \quad (3)$$

where D is a diagonal matrix containing the outdegrees of the graph nodes, A is the adjacency matrix with $\alpha \in [0,1]$.

Random Walk with Restart Algorithm: Random Walk with Restart algorithm (Tong et al., 2006; Pan et al., 2004) considers a random walker that starts from node v_x, and chooses randomly among the available edges every time, except that, before he makes a choice, with probability α, he goes back to node v_x (restart). The similarity matrix (i.e. Kernel) between nodes of a graph, can be computed by Eq. (4):

$$Kernel_{RWR} = (I - \alpha P)^{-1} \quad (4)$$

where I is the identity matrix and P is the transition-probability matrix.

Table 1
The algorithms used in the comparison with their parameters and test values.

Algorithm	Abbreviation	Equation	Parameter	Test values
FriendLink	FriendLink	(1)	i	2,3,4,5
Markov Diffusion (Fouss et al., 2006)	MD	(2)	t	1,2,. . .,10,50,100
Regularized Commute Time (Fouss et al., 2012)	RCT	(3)	α	10^{-6}, 10^{-5}, . . ., 0.99
Random Walk with Restart (Tong et al., 2006; Pan et al., 2004)	RWR	(4)	α	10^{-6}, 10^{-5}, . . ., 0.99
Katz Status Index (Katz, 1953)	Katz	(5)	β	0.05,0.005,0.0005
Adamic/Adar (Adamic and Adar, 2005)	AA	(6)	–	–
Preferential Attachment (Barabasi et al., 2002; Newman, 2001)	PA	(7)	–	–
Friend of a Friend (Chen et al., 2009)	FOAF	(8)	–	–
Shortest Path (Fredman and Tarjan, 1987)	SP	–	–	–

Katz status index algorithm: Katz (1953) defines a measure that directly sums over all paths between any pair of nodes in graph $\mathcal{G}$, exponentially damped by length to count short paths more heavily. The similarity between nodes v_x and v_y, can be computed by Eq. (5):

$$score(v_x, v_y) = \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot \left| paths^{\ell}_{v_x, v_y} \right|, \tag{5}$$

where $\left| paths^{\ell}_{v_x, v_y} \right|$ is the number of all length-ℓ paths from v_x to v_y.

Adamic/Adar algorithm: Adamic and Adar (2005) proposed a distance measure to decide when two personal home pages are strongly "related". In particular, they computed features of the pages and defined the similarity between two pages x, y as follows: $\sum_z 1/\log(frequency(z))$, where z is a feature shared by pages x, y. This refines the simple counting of common features by weighting rarer features more heavily. The similarity between nodes v_x and v_y, can be computed by Eq. (6):

$$score(v_x, v_y) = \sum_{z \in \Gamma(v_x) \cap \Gamma(v_y)} \frac{1}{\log \left| \Gamma(z) \right|} \tag{6}$$

where $\Gamma(v_x)$, $\Gamma(v_y)$ are the sets of neighbors of v_x and v_y.

Preferential Attachment: The basic premise of Preferential Attachment is that the probability a new edge involves node v_x is proportional to its degree. Barabasi et al. (2002) and Newman (2001) have further proposed on the basis of empirical evidence, that the probability of that a new edge involves v_x and v_y is correlated with the product of the number of connections of v_x and v_y, corresponding to the measure shown by Eq. (7),

$$score(v_x, v_y) := |\Gamma(v_x) \cdot \Gamma(v_y)| \tag{7}$$

where $\Gamma(v_x)$, $\Gamma(v_y)$ are the sets of neighbors of v_x and v_y.

Friend of a Friend algorithm: The Friend of a Friend (FOAF) algorithm (Chen et al., 2009) relies on the number of friends that two nodes v_x and v_y have in common, as shown by Eq. (8)

$$score(v_x, v_y) := |\Gamma(v_x) \cap \Gamma(v_y)| \tag{8}$$

where $score(v_x, v_y)$ is the number of common friends of v_x and v_y, and $\Gamma(v_x)$, $\Gamma(v_y)$ are the sets of their neighbors. The candidates are recommended to v_x in decreasing order of their score.

Shortest Path algorithm: Shortest Path calculates the shortest distance between any pair of users in the social network. Therefore, users can be recommended to a target user v_x according to their shortest distance in the social network. We use the Frendman–Tarjan algorithm (Fredman and Tarjan, 1987) to calculate the shortest paths between any pair of nodes.

Table 1 summarizes the algorithms used in the experimental evaluation. The second column of Table 1 provides an abbreviation of each algorithm name. Most algorithms require the tuning of a parameter, which is shown in the last two columns of Table 1.

6.2. Real and synthetic evaluation data sets

To evaluate the examined algorithms, we have used a synthetic and three real data sets from Facebook, Hi5 and the Epinions web sites.

We crawled the graph data from the Facebook and Hi5 web sites at two different time periods. In particular, we crawled the Facebook web site on the 30th of October 2009 and on the 15th of December 2010. Our data crawling method was the following: For each user u, we traverse all his friends and then traverse the friends of each of u's friends, etc. From the first crawl of Facebook web site we created a training data set with 3694 users (network size $N = 3.694$, number of edges $E = 13,692$), denoted as Facebook 3.7K, where the initial starting node of our crawling was a random user in Germany. From the second crawl of Facebook web site we created the probe data set with the same users by only preserving 3912 new emerged edges among them and dismissing the 1150 new users that appeared in the second crawl data set. Notice that, we had 120 deletions of previous existed edges and 135 deletions of users in the second crawl data set. We followed the same crawling procedure from the Hi5 web site. From the first crawl of Hi5 web site we created a training data set with 63,329 users and 88,261 edges among them, denoted as Hi5 63K[1], where the initial starting node of our crawling was a random user in the US. From the second crawl of Hi5 web site we created the probe data set with the same users by only preserving 16,512 new emerged edges connecting them and dismissing the 9150 new users that appeared in the data set. Moreover, 1480 edges and 1250 vertices were deleted in the second crawl data set. Based on the above graph statistics the general provision is that edges and vertices are mostly added to the graph and that the graph is expanded steadily. The graph data from the first crawl are used to predict the new links emerging in the second crawl.

We also use in our experiments the Epinions[2] data set, which is a who-trusts-whom social network. In particular, users of Epinions.com express their Web of Trust, i.e. reviewers whose reviews and ratings they have found to be valuable. The Epinions data set is a directed network and, thus, we treat it by simply disregarding the directions of links (Wasserman and Faust, 1994). It contains 49K users and 487K edges among pairs of users. Moreover, we include in our experiments the extended Epinions data set[3] which is a directed and signed network. In particular, the extended Epinions data set contains 131,828 nodes and 841,372 edges, each labeled either trust (positive) or distrust (negative). Of these labeled edges, 85% are positive and 15% are negative. We interpret the weight of a positive edge to be the real value +1 and the negative to be −1.

We calculated several topological properties of the real data sets which are presented in Fig. 7.

[1] http://delab.csd.auth.gr/ symeon.
[2] http://www.trustlet.org/wiki/.
[3] http://snap.stanford.edu/data/soc-sign-epinions.html.

TOPOLOGICAL PROPERTIES:
N = total number of nodes
E = total number of edges
ASD = average shortest path distance between node pairs
ADEG = average node degree
LCC = average local clustering coefficient
GD = graph diameter (maximum shortest path distance)

Data-Set	Type	N	E	ASD	ADEG	LCC	GD
Epinions 49K	Directed unsigned	49288	487183	4.01	19.76	0.26	14
Extended Epinions 132K	Directed signed	131828	841372	4.10	12.76	0.24	14
Facebook 3.7K	Undirected unsigned	3694	13692	4.23	7.2	0.111	10
Hi5 63K	Undirected unsigned	63329	88261	7.18	2.78	0.02	19

Fig. 7. Topological properties of the real data sets.

As shown in Fig. 7, Epinions 49K, Extended Epinions 132K and Facebook 3.7K present (i) a large clustering coefficient (LCC) equal to 0.26, 0.24 and 0.111 respectively, and (ii) a small average shortest path length (ASD) equal to 4.01, 4.1 and 4.233 respectively. These topological features can be mainly discovered in small-worlds networks. Small-world networks have sub-networks that are characterized by the presence of connections between almost any two nodes within them (i.e. high LLC). Moreover, most pairs of nodes are connected by at least one short path (i.e. small ASD).

In contrast, as also shown in Fig. 7, Hi5 63K has a very small LLC (0.02) and a quite big ASD (7.18). In other words, Hi5 data set cannot be considered as a small-world network, since (i) most of its nodes cannot be reached from every other by a small number of hops or steps and (ii) does not have sub-networks that are a few edges shy of being cliques.

The size of real online social networks is huge. For instance, Facebook has over 500 million users with an average of roughly 100 friends each. This means that our data sample collected is extremely small relative to the overall graph. To study the algorithms' efficiency (i.e. time complexity) and effectiveness (i.e. accuracy with controllable sparsity), we also used synthetic network models of different sizes. Although real networks have many complex structural properties (Costa et al., 2007), such as degree heterogeneity, the rich-club phenomenon, etc., as a start point for generating synthetic data sets, we consider a very simple model. In contrast to purely random (i.e., Erdos-Renyi) graphs, where the connections among nodes are completely independent random events, our synthetic model ensures dependency among the connections of nodes, by characterizing each node with a ten-dimensional vector with each element a randomly selected real number in the interval $[-1, 1]$. This vector represents the node's intrinsic features such as the profile of a person. Two nodes are considered to be similar and thus of high probability to connect to each other if they share many close attributes. The synthetic data set was created by the same generator used in Papadimitriou et al. (2011) and Symeonidis et al. (2010). Given a network size N and a mean degree k of all nodes, we start with an empty network with N nodes. At each time step, a node with the smallest degree is randomly selected (there is more than one node having the smallest degree). Among all other nodes whose degrees are smaller than k, this selected node will connect to the most similar node with probability $1-p$, while a randomly chosen one with probability p. The parameter $p \in [0, 1]$ represents the strength of randomness in generating links, which can be understood as noise or irrationality that exists in almost every real system. Based on the above procedure, we have created 3 synthetic data sets based on different network sizes N (1000, 10,000, 100,000), where the degree distribution of the network decreases slowly, closely following a power-law. The average node degree has been calculated to be around 20. We also calculated several topological properties of the derived synthetic data sets which are presented in Fig. 8.

6.3. Experimental protocol and evaluation metrics

As already described in Section 6.2, in our evaluation we consider the division of Facebook 3.7K and Hi5 63K data sets into two sets, according to the exact time stamp of the links downloaded: (i) the training set $\mathcal{E}^T$ is treated as known information and, (ii) the probe set $\mathcal{E}^P$ is used for testing. No information in the probe set is allowed to be used for prediction. It is obvious that $\mathcal{E}^T \cap \mathcal{E}^P = \emptyset$. For each user that has at least one new friend in $\mathcal{E}^P$ we generate friend recommendations based on his friends in $\mathcal{E}^T$. Then, we average the results for each user and compute the final performance of each algorithm.

Epinions and Synthetic data sets do not have time stamps of the edges. The performance of the algorithms is evaluated by applying double cross-validation (internal and external). Each data set was divided into 10 subsets. Each subset ($\mathcal{E}^P$) was in turn used for performance estimation in the external cross-validation. The 9 remaining subsets ($\mathcal{E}^T$) were used for the internal cross-validation. In particular, we performed an internal 9-fold cross-validation to determine the best values of the algorithms' needed parameters. We chose as values for the parameters those providing the best performance on the internal 9-fold cross-validation. Then, their performance is averaged on the external 10-fold cross-validation. The presented results, based on two-tailed t-test, are statistically significant at the 0.05 level.

In our evaluation we consider the following evaluation metrics:

We use the classic precision/recall metric as performance measure for friend recommendations. For a test user receiving a list of k

Data-Set	N	E	ASD	ADEG	LCC	GD
Synthetic-(N=1000, k=20)	1000	10000	4.818	≈ 20	0.014	10
Synthetic-(N=10000, k=20)	10000	100000	8.585	≈ 20	0.007	14
Synthetic-(N=100000,k=20)	100000	1000000	12.899	≈ 20	0.001	18

Fig. 8. Topological properties of the synthetic data sets.

Table 2
MAP for 5 attenuation factors on both synthetic and real data sets.

Attenuation factor	Synthetic 1K	Synthetic 10K	Synthetic 100K	Epinions 49K	Facebook 3.7K	Hi5 63K
$1/(m-1)$	**0.305**	**0.131**	**0.089**	**0.445**	**0.385**	**0.154**
$1/(2m)$	0.244	0.108	0.062	0.390	0.341	0.139
$1/(m^2)$	0.183	0.094	0.041	0.322	0.302	0.099
$1/log(m)$	0.149	0.081	0.027	0.287	0.257	0.045
b^m	0.122	0.043	0.020	0.235	0.211	0.012

recommended friends (top-k list), precision and recall are defined as follows:

Precision is the ratio of the number of relevant users in the top-k list (i.e., those in the top-k list that belong in the probe set $\mathcal{E}^P$ of friends of the target user) to k.

Recall is the ratio of the number of relevant users in the top-k list to the total number of relevant users (all friends in the probe set $\mathcal{E}^P$ of the target user).

Moreover, since we provide to a test user u a top-k list of friends, it is important to consider the order of the presented friends in this list. That is, it is better to have a correct guess in the first places of the recommendation list. Thus, we use the **Mean Average Precision (MAP)** to emphasize ranking of relevant users higher. We define MAP by Eq. (9):

$$MAP = \frac{1}{|N|}\sum_{u=1}^{|N|}\frac{1}{r_u}\sum_{k=1}^{r_u} Precision_u@k \tag{9}$$

where N is the number of users in the probe data set, r_u is the number of relevant users to a user u and $Precision_u@k$ is the precision value at the kth position in the recommendation list for u. Notice that MAP takes into account both precision and recall and is geometrically referred as the area under the Precision-Recall curve.

Furthermore, we use the **AUC statistic** to quantify the accuracy of prediction algorithms and test how much better they are than pure chance, similarly to the experimental protocol followed by Clauset hierarchical structure. AUC is equivalent to the area under the receiver-operating characteristic (ROC) curve. It is the probability that a randomly chosen missing link (a link in $\mathcal{E}^P$) is given a higher similarity value than a randomly chosen non-existent link (a link in $U - \mathcal{E}^T$, where U denotes the universal set). In the implementation, among n times of independent comparisons, if there are n' times the missing link having higher similarity value and n'' times the missing link and nonexistent link having the same similarity value, we define AUC by Eq. (10):

$$AUC = \frac{n' + 0.5 \times n''}{n} \tag{10}$$

If all similarity values are generated from an independent and identical distribution, the accuracy should be about 0.5. Therefore, the degree to which the accuracy exceeds 0.5 indicates how much better the algorithm performs than chance. This is also explained thoroughly at the end of Section 6.5.

6.4. Sensitivity analysis for the FriendLink algorithm

In this section, we study the sensitivity of FriendLink accuracy performance in synthetic and real networks (i) with different attenuation factors, (ii) with different controllable sparsity, (iii) with different controllable randomness/noise in generating links, (iv) with different ℓ values for path traversal and (v) the relations between the basic parameters (i.e. attenuation factors, path lengths, and graph densities) if one parameter is fixed and the other two parameters change.

In Section 4, we presented the definition of our similarity measure (see Eq. (1)). The attenuation factor that was mentioned, weights paths according to their length ℓ. In this section, we test other possible attenuation factors in order to discover the best MAP value that we can attain. In particular, we have tested the following possible attenuation factors: (i) $1/(m-1)$, (ii) $1/2 \cdot m$, (iii) $1/m^2$, (iv) $1/log(m)$ and (v) the Katz's index attenuation factor β^m, where m is the path length. The attenuation factors performance can be seen in Table 2 for all data sets. As shown, the best performance in all data sets is attained by $1/(m-1)$. In the following, we keep the $1/(m-1)$ as the default attenuation factor of the FriendLink algorithm.

Next, we measure the MAP performance that FriendLink attains, with different controllable sparsity. To examine the MAP performance of FriendLink in terms of different network sparsity, we have created for each of the 3 synthetic data sets (1K, 10K and 100K) 5 different sparsity cases, by changing the fraction of observed edges, as shown in Fig. 9a. As expected, as the fraction of edges observed increases, MAP increases too. This is reasonable, since every prediction algorithm is expected to give higher accuracy for a denser network.

In our synthetic model, the parameter $p \in [0, 1]$ represents the strength of randomness/noise in generating links. Next, we test FriendLink's sensitivity with different graph model randomness. As shown in Fig. 9b, when the strength of randomness is weak, FriendLink performs quite well for all three data sets. However, as the strength of randomness becomes high in all data sets FriendLink cannot perform better than pure chance.

The experimental results shown in Fig. 9 basically prove the following two points. Firstly, that the increase in the number of edges observed, will result in an increase in precision attained by Friendlink. We are able to decrease/increase the number of edges observed, and therefore the information available to Friendlink, by taking into account a smaller/bigger part of our training set. Secondly, the noise level plays an important part in Friendlink's sensitivity. This is to be expected, since random edges in our training set will result in Friendlink having a greater difficulty making accurate recommendations.

In Section 5.2, one of the required input values for the FriendLink algorithm is the length ℓ of paths considered in a graph. To improve our recommendations, it is important to fine-tune the ℓ variable. Based on Milgram's, 1967 "small-world hypothesis", ℓ should take integer values in the interval [2,6]. Fig. 10a–c illustrates precision for varying ℓ values for the Epinions 49K, Facebook 3.7K and Hi5 63K data sets, respectively. As expected, precision decreases as the number of recommended friends is increased. The best precision is attained for $\ell = 3$.

Next, we examine the performance of recall metric vs. different values of ℓ. Fig. 11a–c illustrates recall for varying ℓ values for the Epinions 49K, Facebook 3.7K and Hi5 63K data sets, respectively. As expected, recall increases as the number of recommended friends is increased. Once again, the best recall performance is attained for $\ell = 3$. The main reason is that ASD for all data sets is relative small and paths of length 3 can exploit simultaneously local and global characteristics of a graph. In the following, we keep the path equal to $\ell = 3$, as the default value of the FriendLink algorithm.

Finally, we conduct experiments on our Facebook 3.7K data set to investigate possible relations between the basic parameters (i.e. attenuation factors, path lengths, and graph densities)

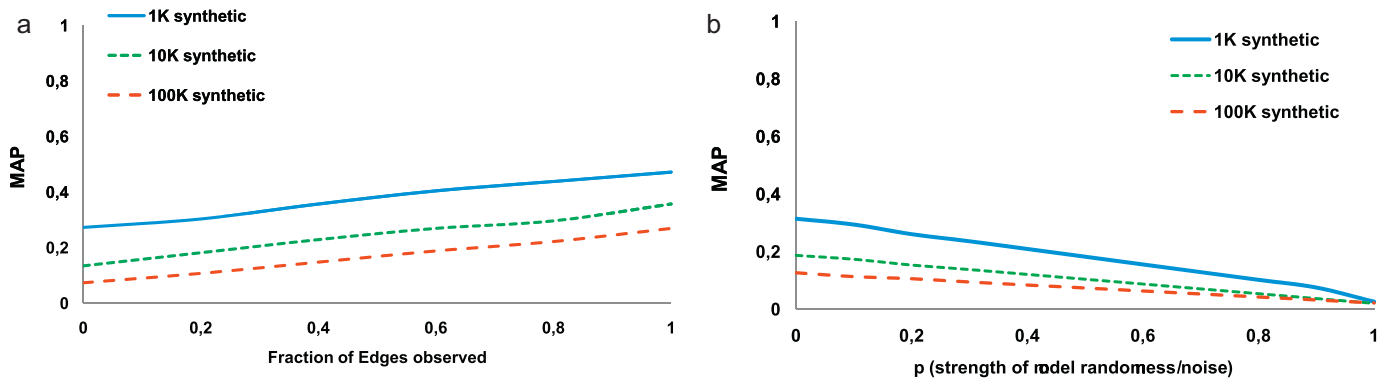

Fig. 9. For 3 synthetic data sets (a) MAP vs. Fraction of Edges observed (b) MAP vs. p randomness/noise graph.

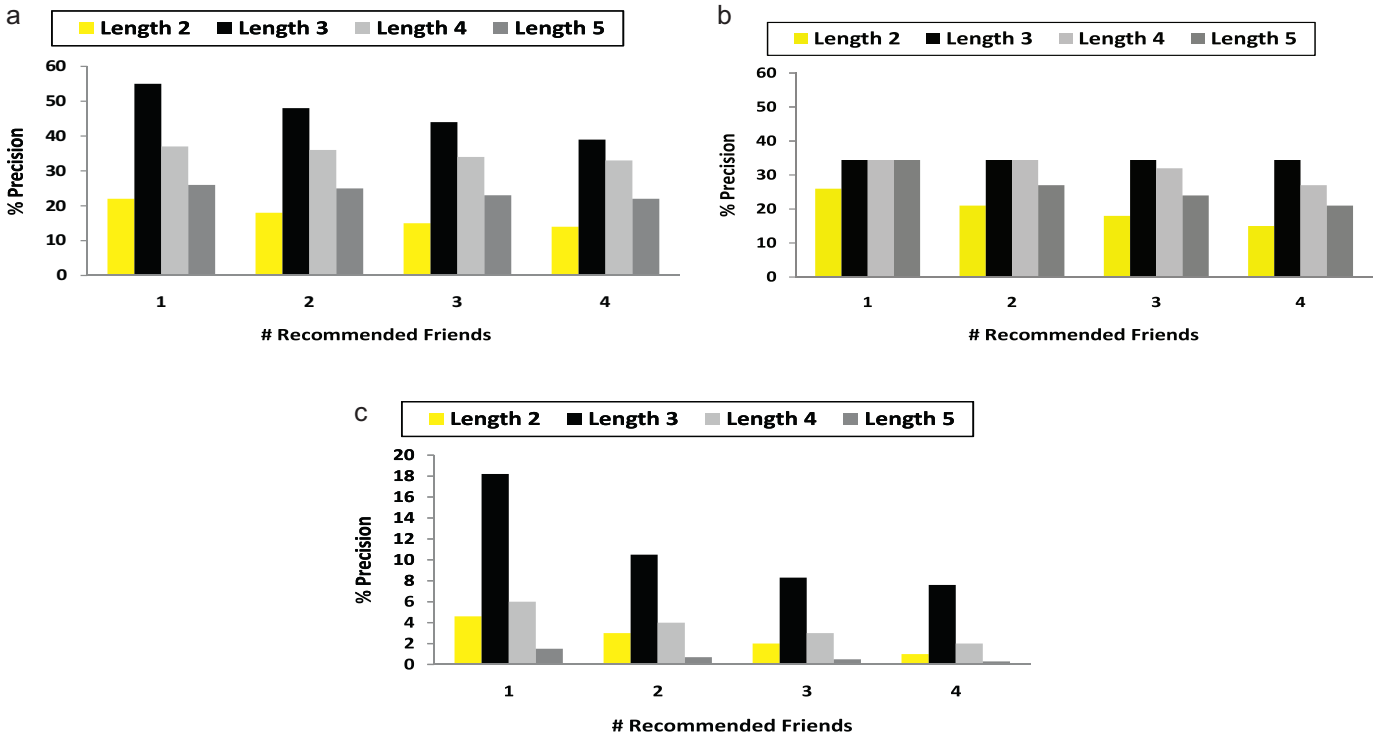

Fig. 10. Precision diagrams for data sets (a) Epinions 49K, (b) Facebook 3.7K and (c) Hi5 63K.

that influence FriendLink's performance. As shown in Fig. 12, if one parameter is fixed, the MAP value of FriendLink varies as the other two parameters change. In particular, Fig. 12(a) shows that the highest MAP value is obtained for path length $\ell = 3$ and graph density of edges observed fixed at 80%. Fig. 12(b), shows that attenuation factor $1/(m-1)$ attains the highest MAP value when $\ell = 3$. Lastly, Fig. 12(c) depicts the relationship between attenuation factors and graph densities.

6.5. Comparison of FriendLink with other methods

In this section, we compare FriendLink against MD, RCT, RWR, Katz, AA, PA, FOAF and SP algorithms. Table 3 presents the MAP values of the tested algorithms for the Epinions 49K, Facebook 3.7K and Hi5 63K data sets, respectively. As shown, FriendLink outperforms the other algorithms in all three real data sets. The reason is that FriendLink exploits local and global characteristics of the graph. In contrast, MD, RCT, RWR, Katz and SP traverse globally the social network, missing to capture adequately the local characteristics of the graph. Moreover, AA, PA, and FOAF fail to provide accurate recommendations because they exploit only local characteristics of the graph. Notice that MAP values are impressive for the Epinions 49K and Facebook 3.7 data sets. The main reason is the topological characteristics of both graphs (i.e. high LCC and small ASD). Both data sets can be considered as small-world networks. That is both networks are strongly localized with most of paths being of short geographical lengths.

Table 3
MAP values of all algorithms for the real data sets.

Algorithm	Epinions 49K	Facebook 3.7K	Hi5 63K
FriendLink	0.445	0.385	0.154
MD	0.392	0.336	0.132
RCT	0.362	0.315	0.121
RWR	0.285	0.225	0.085
Katz	0.265	0.205	0.075
AA	0.140	0.125	0.054
PA	0.132	0.115	0.035
FOAF	0.125	0.105	0.021
SP	0.111	0.096	0.014

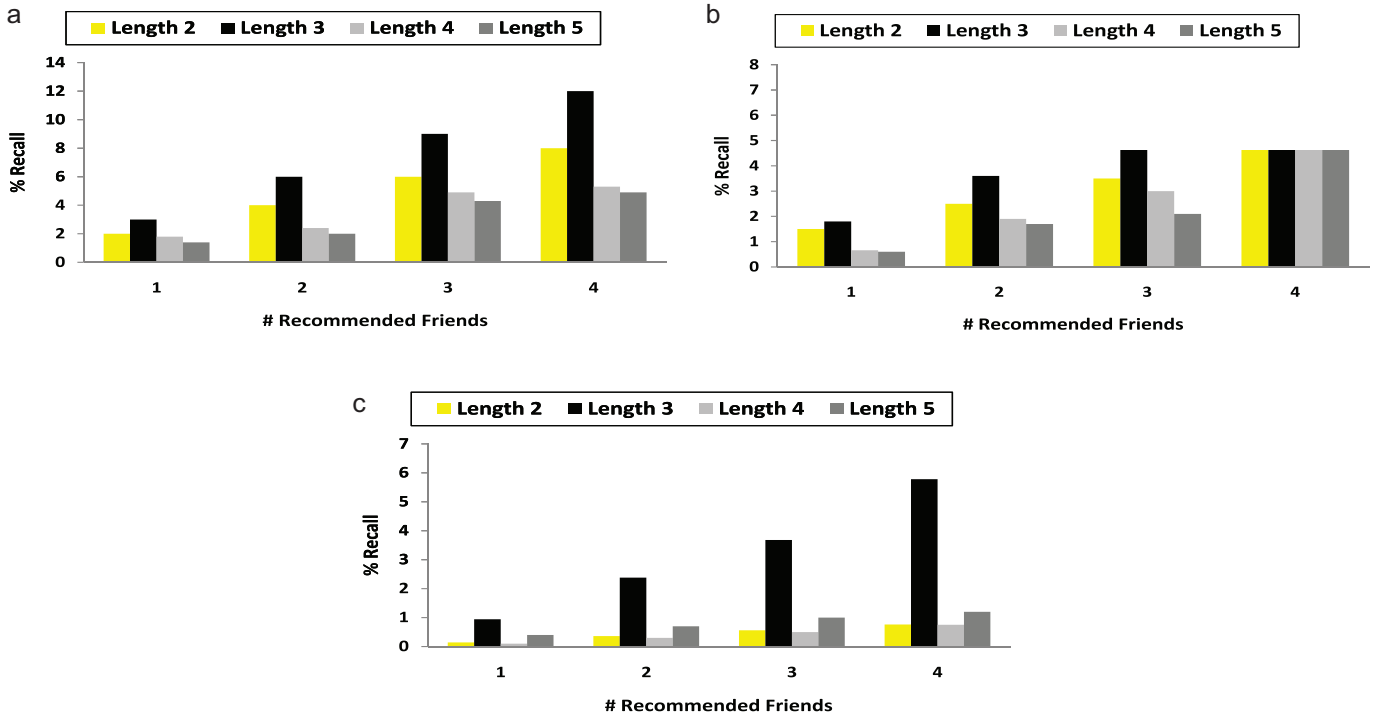

Fig. 11. Recall diagrams for data sets (a) Epinions 49K, (b) Facebook 3.7K and (c) Hi5 63K.

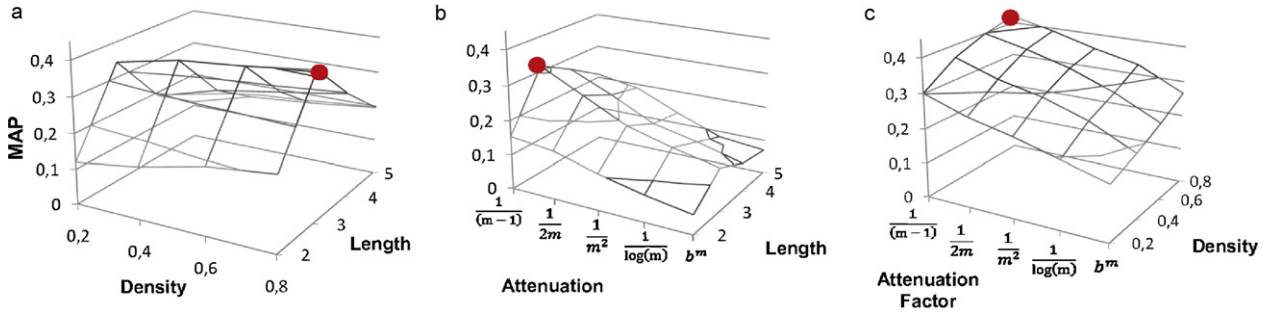

Fig. 12. MAP performance of attenuation factors, length ℓ and density on the Facebook 3.7K data set.

Thus, all algorithms can more easily find a short path that connects a pair of nodes, and recommend friends that are near the target's user neighborhood. In contrast, the overall performance of tested algorithms is significantly decreased with the Hi5 63K data set. The main reason is that the Hi5 63K data set has a small LCC and a high ASD. Therefore, in contrast to both Epinions and Facebook data sets, it cannot be considered as a small world network.

6.6. Comparison to randomness

To more meaningfully represent the algorithms' quality, we use as a baseline algorithm a random predictor which simply randomly selects pairs of users as friends. This random friendship guess is denoted as random predictor. Notice that in terms precision the performance of all tested algorithms should be at least better than the case, where the friend recommendations would be performed randomly. For the synthetic 10K data set, if each user was connected with all others, we would have 49,995,000 $[(10,000 \times 10,000 - 10,000)/2]$ graph edges. However, in the synthetic 10K test data set, we counted the number of the actual graph edges that exist, which amount to 199,980 edges. Thus, if we randomly proposed a new friend to a target user u, we would get a precision of 0.004 (199,980/49,995,000). Following the same procedure for the Epinions 49K and Facebook 3.7K data sets, we get their precision of 0.0004 and 0.002 respectively. The corresponding precision for the Hi5 data set is 0.00004. This value is obtained by dividing the actual graph edges that appear in the 63K data set, which are counted to be 88,261 by the total number of edges that appear in the 63K data set, which is 2,005,249,456 $[(63,329 \times 63,329 - 63,329)/2]$.

Table 4 shows the performance of the algorithms on each data set, in terms of factor improvement over random predictor in terms of precision, when we recommend a top-1 friend to a target user

Table 4
Algorithm performance measured by the factor improvement of precision over random prediction.

Algorithm	Epinions 49K	Facebook 3.7K	Hi5 63K
FriendLink	650	280	5502
MD	593	261	4902
RCT	580	250	4639
RWR	525	235	4013
Katz	451	210	3250
AA	360	151	2601
PA	331	140	2420
FOAF	302	130	2501
SP	250	110	2502

Table 5
Comparison of tested algorithms for the AUC statistic at 25%, 50% and 75% of edges observed.

Algorithm	Epinions 49K			Facebook 3.7K			Hi5 63K		
	25%	50%	75%	25%	50%	75%	25%	50%	75%
FriendLink	0.633	0.684	0.791	0.572	0.682	0.875	0.572	0.641	0.735
MD	0.581	0.664	0.769	0.562	0.662	0.857	0.562	0.619	0.723
RCT	0.572	0.656	0.752	0.553	0.656	0.853	0.553	0.605	0.719
RWR	0.562	0.626	0.741	0.545	0.644	0.843	0.532	0.601	0.710
Katz	0.546	0.603	0.731	0.539	0.625	0.821	0.524	0.585	0.686
AA	0.534	0.578	0.678	0.532	0.619	0.783	0.521	0.579	0.651
PA	0.533	0.574	0.665	0.528	0.610	0.772	0.519	0.562	0.621
FOAF	0.531	0.566	0.653	0.524	0.601	0.762	0.514	0.541	0.620
SP	0.527	0.537	0.594	0.521	0.533	0.610	0.510	0.535	0.591

u. Bold entries represent the best factor improvement attained for each data set. We can see that all 9 methods outperform the random predictor, suggesting that there is indeed useful information contained in the network topology.

Notice that the factor of improvement – in terms of precision for all methods – over randomness is increased as the data sparsity of a data set is increased. For instance, the factor of improvement is enormous for the Hi5 data set, because it presents the larger data sparsity among all real data sets. We note, however, that using this ratio to judge prediction algorithms has an important disadvantage. Some missing connections are much easier to predict than others: for instance, if a network has a heavy-tailed degree distribution and we remove a randomly chosen subset of the edges, the chances are excellent that two high-degree vertices will have a missing connection. Thus, such a connection can be easily predicted by even simple heuristics such as PA or FOAF algorithm.

To overcome the aforementioned limitation and more meaningfully represent the friend recommendation algorithms' accuracy performance, we also use the AUC statistic, which looks at an algorithms overall ability to rank all the missing connections over nonexistent ones, not just those that are easiest to predict. As shown in Table 5, we measure the AUC values vs. the fraction of observed links used in the training set for all real data sets. As shown, as a greater fraction of the network is known, the accuracy becomes even greater, for all methods. FriendLink does far better than pure chance, indicating that it is a strong predictor of missing structure. The main reason is that FriendLink captures effectively the local and global graph features.

6.7. FriendLink accuracy performance in signed networks

In this section, we present the accuracy performance of FriendLink when we take into account positive and negative links of a signed network, i.e. extended Epinions 132K data set. We have two different variants of FriendLink: The first variation considers only positive links and is denoted as *FriendLink*$^{+}$. The second variation considers both positive and negative links and is denoted as *FriendLink*$^{\pm}$. Fig. 13a presents the precision and recall diagram for both versions of FriendLink, whereas Fig. 13b presents the AUC accuracy statistic. Both Figures show that *FriendLink*$^{\pm}$ outperforms *FriendLink*$^{+}$. The reason is that *FriendLink*$^{\pm}$ exploits positive and negative links. This means that if we use information about negative edges for predicting the presence of positive edges we get an accuracy improvement of FriendLink predictions. These results clearly demonstrate that there is, in some settings, a significant improvement to be gained by using information about negative edges, even to predict the presence or absence of positive edges.

Table 6
Time comparison of all tested algorithms for the synthetic and real data sets.

Algorithm	Synthetic 10K	Synthetic 100K	Epinions 49K	Facebook 3.7K	Hi5 63K
FriendLink	50 s	450 s	245 s	26 s	340 s
RCT	81 s	752 s	420 s	45 s	692 s
MD	74 s	698 s	351 s	36 s	480 s
RWR	78 s	702 s	380 s	40 s	520 s
Katz	90 s	811 s	460 s	50 s	617 s
AA	40 s	145 s	69 s	22 s	265 s
PA	39 s	136 s	65 s	24 s	242 s
FOAF	37 s	126 s	55 s	15 s	221 s
SP	62 s	250 s	125 s	29 s	360 s

6.8. Time comparison of FriendLink with other methods

In this section, we compare FriendLink against MD, RCT, RWR, Katz, AA, PA, FOAF and SP algorithms in terms of efficiency using 10K and 100K synthetic and 3 real data sets. We measured the clock time for the off-line parts of all algorithms. The off-line part refers to the building of the similarity matrix between any pair of nodes in a graph. The results are presented in Table 6. As shown, FriendLink outperforms MD, RCT, RWR and Katz, since they calculate the inverse of an $n \times n$ matrix. As expected, AA, PA, and FOAF algorithms, outperform the other algorithms due to their simpler complexity.

Notice that the results depict the time needed to compute the whole similarity matrix. On the other hand, if we were to calculate the similarity matrix of only one user, then the computation would require only part of a second to produce a recommendation.

7. Scalability

There are many difficulties in the study of the link prediction problem. One of them is the huge size of real systems. For instance, Facebook has over 500 million users with an average of roughly 100 friends each. To run our algorithm for huge sized networks, it should be adjusted to support a MapReduce (Dean and Ghemawat, 2008) implementation. MapReduce is a distributed computing model for

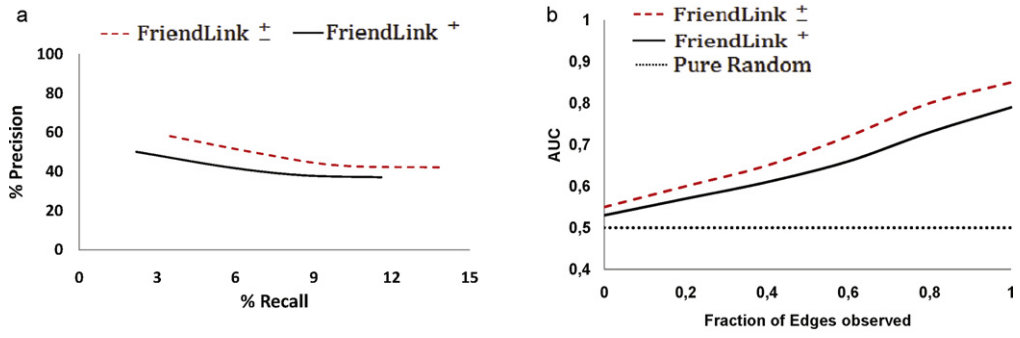

Fig. 13. Accuracy performance of Friendlink in terms of (a) precision/recall and (b) AUC statistic.

Schema of map and reduce functions

map: input	→list(p, ℓ, s)
reduce: list(p, ℓ, s)	→output

Instantiation of the schema for similarity calculation

map: user pair	→list(user pair, length ℓ, similarity)
reduce: (user pair, length ℓ, similarity)	→(user pair, total similarity)

Example for similarity calculation

map: (user1, user2)	→(⟨user1, user2, 2, 0.03⟩, ⟨user1, user2, 3, 0.2⟩ , ⟨user1, user2, 4, 0.14⟩ , ⟨user1, user2, 5, 0.07⟩)
reduce: (⟨user1, user2, 2, 0.1⟩, ⟨user1, user2, 3, 0.4⟩ , ⟨user1, user2, 4, 0.3⟩ , ⟨user1, user2, 5, 0.2⟩)	→(⟨user1, user2 ⟩, 0.44)

Fig. 14. Map and reduce functions in MapReduce.

processing large volumes of data. MapReduce is implemented in three steps: (i) splitting up the computing job into chunks that standard machines can process in a short time, (ii) parallel processing on each sub-part by an independent machine and, (iii) the collection of intermediate values, produced by each machine, in order to calculate the final result. In our case, the calculation of the similarity matrix could be assigned to many machines in the following way. Each machine calculates one of the $2 \ldots \ell$-length paths for a specific pair of users and then sum up the paths to calculate the final similarity value. An example is shown in Fig. 14. As shown in Fig. 14, each Map function on every machine receives as input a pair of users and produces the similarity value for a designated path length ℓ. All values for each pair of users are collected into one final value in the reduce phase. In our example, the similarity values produced by the Map function, which are 0.03, 0.2, 0.14 and 0.07 for path length $\ell = 2, 3, 4, 5$ respectively, will be "reduced" to one final similarity value, which is 0.44, for the respective pair of users.

8. Discussion

Real networks have many complex structural properties (Costa et al., 2007), such as degree heterogeneity, the rich-club phenomenon, the mixing pattern, etc. These network properties are not considered by our synthetic network model, since they are out of the scope of this paper. However, our synthetic network model can be easily extended to better resemble real networks. For example, by applying the degree heterogeneity index (Costa et al., 2007) with a probability p, a synthetic network with different level of degree heterogeneity can be composed.

Also, as it was shown in Section 6.4, the attenuation factor weight for each path of given length plays an important role in the performance of our FriendLink algorithm. One could suggest learning these optimal weights instead of guessing them. One way would be through linear regression. Linear regression analyzes the linear relationship between two variables, Y and X, where in our case Y is a vector that contains the similarities between a given user and the other users in a graph, whereas X is a matrix that contains the paths of different length between the given user and the others of the graph (i.e. the training data of a user). Based on linear regression, it stands that $Y = AX$, where A is a vector which contains the optimal coefficient values of the attenuation factor. In order to find the best coefficient values of the attenuation factor, A can be calculated by equation $A = (X'X)^{-1}X'Y$. Since the similarities Y between a given user and the other users of a graph are not available from the beginning, we can instead consider Y to contain values from the testing data of the user. The computed values of A can then be used as attenuation optimal weights.

9. Conclusions

Online social networking systems have become popular because they allow users to share content, such as videos and photos, and expand their social circle, by making new friendships. In this paper, we introduced a framework to provide friend recommendations in OSNs. Our framework's advantages are summarized as follows:

- We define a new node similarity measure that exploits local and global characteristics of a network. Our FriendLink algorithm, takes into account all ℓ-length paths that connect a person to other persons in an OSN, based on the "algorithmic small world hypothesis".
- We derive variants of our method that apply to different types of networks (directed/undirected and signed/unsigned). We show that a significant accuracy improvement can be gained by using information about both positive and negative edges.
- We performed extensive experimental comparison of the proposed method against 8 existing link prediction algorithms, using synthetic and real data sets (Epinions, Facebook and Hi5). We have shown that our FriendLink algorithm provides more accurate and faster friend recommendations compared to existing approaches.
- Our proposed algorithm also outperforms the existing global-based friend recommendation algorithms in terms of time complexity, as shown experimentally in Section 6.8.
- Finally, in Section 7 we discuss extensively a possible MAP implementation to address the scalability issue.

In the future, we indent to examine ways of improving friend recommendations based on other features that OSNs offer. Except the friendship network, users in OSNs can also form several *implicit* social networks through their daily interactions like co-commenting on people's post, co-rating similarly products, and co-tagging people's photos. The combination of similarity matrices derived from heterogenous explicit or implicit social networks can exploit information from multi-modal social networks and therefore yield to more accurate friend recommendations.

References

Adamic, L., Adar, E., 2005. How to search a social network. Social Networks 27 (3), 187–203.

Barabasi, A.L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Vicsek, T., 2002. Evolution of the social network of scientific collaborations. Physica A 311, 590–614.

Chen, J., Geyer, W., Dugan, C., Muller, M., Guy, I., 2009. Make new friends, but keep the old: recommending people on social networking sites. In: Proceedings 27th International Conference on Human Factors in Computing Systems (CHI'2009), Boston, MA, pp. 201–210.

Costa, L. da F., Rodrigues, F.A., Travieso, G., Villas Boas, P.R., 2007. Characterization of complex networks: a survey of measurements. Advances in Physics 56 (1), 167–242.

Dean, J., Ghemawat, S., 2008. Mapreduce: simplified data processing on large clusters. Communications of the ACM 51, 107–113.
Foster, K.C., Muth, S.Q., Potterat, J.J., Rothenberg, R.B., 2001. A faster Katz status score algorithm. Computational & Mathematical Organization Theory 7, 275–285.
Fouss, F., Yen, L., Pirotte, A., Saerens, M., 2006. An experimental investigation of graph kernels on a collaborative recommendation task. In: Proceedings 6th International Conference on Data Mining (ICDM'2006), Hong Kong, pp. 863–868.
Fouss, F., Pirotte, A., Renders, J.M., Saerens, M., 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge Data Engineering 19 (3), 355–369.
Fouss, F., Francoisse, K., Yen, L., Pirotte, A., Saerens, M., 2012. An experimental investigation of graph kernels on collaborative recommendation and semisupervised classification, Technical Report.
Fredman, M., Tarjan, R., 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34, 596–615.
Goel, S., Muhamad, R., Watts, D., 2009. Social search in 'small-world' experiments. In: Proceedings 18th International World Wide Web Conference (WWW'2009), Madrid, Spain, p. 701.
Hage, P., Harary, F., 1983. Structural models in anthropology.
Jeh, G., Widom, J., 2002. Simrank: a measure of structural-context similarity. In: Proceedings 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'2002), Edmonton, Canada, pp. 538–543.
Katz, L., 1953. A new status index derived from sociometric analysis. Psychometrika 18 (1), 39–43.
Lü, L., Jin, C.-H., Zhou, T., 2009. Similarity index based on local paths for link prediction of complex networks. Phys. Rev. E 80.
Leskovec, J., Huttenlocher, D., Kleinberg, J., 2010. Predicting positive and negative links in online social networks. In: Proceedings 19th International Conference on World Wide Web (WWW'2010), Raleigh, NC, pp. 641–650.
Liben-Nowell, D., Kleinberg, J., 2003. The link prediction problem for social networks. In: Proceedings 12th International Conference on Information and Knowledge Management (CIKM'2003).
Milgram, S., 1967. The small world problem. Psychology Today 22, 61–67.
Newman, M.E.J., 2001. Clustering and preferential attachment in growing networks. Physical Review E 64 (2).
Pan, J., Yang, H., Faloutsos, C., Duygulu, P., 2004. Automatic multimedia cross-modal correlation discovery. In: Proceedings 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'2004), Seattle, WA, pp. 653–658.
Papadimitriou, A., Symeonidis, P., Manolopoulos, Y., 2011. Predicting links in social networks of trust via bounded local path traversal. In: Proceedings 3rd Conference on Computational Aspects of Social Networks (CASON'2011), Salamanca, Spain.
Rubin, F., 1978. Enumerating all simple paths in a graph. IEEE Transactions on Circuits and Systems 25 (8), 641–642.
Sarkar, P., Moore, A., 2007. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In: Proceedings 23rd Conference on Uncertainty in Artificial Intelligence (UAI'2007).
Schifanella, R., Barrat, A., Cattuto, C., Markines, B., Menczer, F., 2010. Folks in folksonomies: social link prediction from shared metadata. In: Proceedings 3rd ACM International Conference on Web Search and Data Mining (WSDM'2010), New York, NY, pp. 271–280.
Symeonidis, P., Tiakas, E., Manolopoulos, Y., 2010. Transitive node similarity for link prediction in social networks with positive and negative links. In: Proceedings 4th International Conference on Recommender Systems (RecSys'2010), Barcelona, Spain, pp. 183–190.
Tong, H., Faloutsos, C., Pan, J., 2006. Fast random walk with restart and its applications. In: Proceedings 6th International Conference on Data Mining (ICDM'2006), Hong Kong, pp. 613–622.
Tylenda, T., Angelova, R., Bedathur, S., 2009. Towards time-aware link prediction in evolving social networks. In: Proceedings 3rd Workshop on Social Network Mining and Analysis (SNA-KDD'2009), pp. 9:1–9:10.
Wasserman, S., Faust, K., 1994. Social network analysis: methods and applications.
Zheleva, E., Getoor, L., Golbeck, J., Kuter, U., Using friendship ties and family circles for link prediction. In: Proceedings 2nd Workshop on Social Network Mining and Analysis (SNA-KDD'2008), Las Vegas, NV, 2008, pp. 97–113.
Zhou, T., Lu, L., Zhang, Y.-C., 2009. Predicting missing links via local information. The European Physical Journal B 71, 623.

Alexis Papadimitriou received a Bachelor (BSc) in Computer Science from Sussex University of the UK in 2004. He also received a Master diploma (MSc) in Distributed Systems from Brighton University in 2005. He has just received his PhD at Aristotle University of Thessaloniki, Greece. His research interests include sensor networks, data mining, and social networks.

Panagiotis Symeonidis received a Bachelor (BA) in Applied Informatics from Macedonia University of Greece in 1996. He also received a Master diploma (MSc) in Information Systems from the same University in 2004. He received his PhD in Web Mining and Information Retrieval for Personalization from the Department of Informatics in Aristotle University of Thessaloniki, Greece in 2008. Currently, he is working as a post-doc researcher at the Department of Informatics, Aristotle University of Thessaloniki, Greece. He has published more than 25 papers in refereed scientific journals and conference proceedings. His work has received over 120 citations. His research interests include web mining (usage mining, content mining and graph mining), information retrieval and filtering, recommender systems, social media in Web 2.0 and online social networks.

Yannis Manolopoulos received his B.Eng (1981) in Electrical Eng. and his Ph.D. (1986) in Computer Eng., both from the Aristotle Univ. of Thessaloniki. Currently, he is Professor at the Department of Informatics of the latter university. He has been with the Department of Computer Science of the Univ. of Toronto, the Department of Computer Science of the Univ. of Maryland at College Park and the Department of Computer Science of the Univ. of Cyprus. He has published more than 200 papers in refereed scientific journals and conference proceedings. His work has received over 2000 citations from over 450 institutional groups. His research interests include databases, data mining, web and geographical information systems, bibliometrics/webometrics.