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

Spectral Clustering for Link Prediction in Social Networks with Positive and Negative Links

Panagiotis Symeonidis · Nikolaos Mantas

the date of receipt and acceptance should be inserted later

Abstract Online social networks (OSNs) recommend new friends to registered users based on local features of the graph (i.e. based on the number of common friends that two users share). Real OSNs (e.g. Facebook) do not exploit all network structure. Instead, they consider only pathways of maximum length 2 between a user and his candidate friends. This can limit the accuracy of prediction. On the other hand, there are global approaches, which detect the overall path structure in a network, being computationally prohibitive for huge-size social networks. In this paper, we provide friend recommendations, by performing *multi-way spectral clustering*, which uses information obtained from the top few eigenvectors and eigenvalues of the normalized Laplacian matrix and computes a multi-way partition of the data. As a result, it produces a less noisy matrix, which is smaller and more compact than the original one, focusing on main linking trends of the social network. Thus, we are able to provide fast and more accurate friend recommendations. Moreover, spectral clustering compared to traditional clustering algorithms, such as k-means and DBSCAN, which assume globular (convex) regions in Euclidean space, is more flexible, in capturing the non-connected components of a social graph and a wider range of cluster geometries. We perform an extensive experimental comparison of the proposed method against existing link prediction algorithms, the k-means and two-way nCut clustering algorithms, using synthetic and three real data sets (Hi5, Facebook and Epinions). Our experimental results show that our SpectralLink algorithm outperforms the local approaches, the k-means and two-way nCut clustering algorithms in terms of effectiveness, whereas it is more efficient than the global approaches. We show that a significant accuracy improvement can be gained by using information about both positive and negative edges.

P. Symeonidis
Department of Informatics, Aristotle University, Thessaloniki, 54124, Greece.
E-mail: symeon@csd.auth.gr

N. Mantas
Department of Informatics, Aristotle University, Thessaloniki, 54124, Greece.
E-mail: mantas@csd.auth.gr

1 Introduction

Online social networks (OSNs) such as Facebook.com[1], Myspace[2], Hi5.com[3], 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 the network structure.

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. Notice that there is a difference between the *Link Prediction* and the *Edge Sign Prediction* problems [22]. The latter knows that there is a link between two nodes and tries to predict its sign. In contrast, *Link Prediction* predicts if a link will occur between two nodes.

This paper focuses on the *Link Prediction* problem. There are two main approaches [24] that handle it. The first approach is based on local features of a network, focusing mainly on the nodes structure; the second approach is based on global features, detecting the overall path structure in a network. For instance, as an example of a local approach, as shown in Figure 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 three common friends (users U_5, U_6, and U_{10}) (ii) user U_4 because you have two common friends (users U_2 and U_3) (iii) 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.

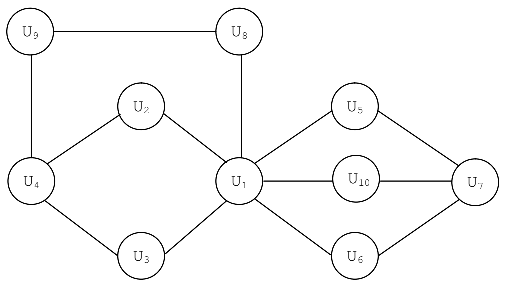


Fig. 1 Social Network Example.

In this paper, we provide friend recommendations, by performing *multi-way spectral clustering*, which uses information obtained from the top few eigenvectors and eigenvalues of the normalized Laplacian matrix and computes a multi-way partition of the data.

Compared to approaches which are based on local features of a network (i.e. Common Neighbors index or else known as FOAF algorithm, Adamic/Adar index,

[1] `http://www.facebook.com`
[2] `http://www.myspace.com`
[3] `http://www.hi5.com`

Jaccard Coefficient, etc. - for more details see Section Related Work), we provide friend recommendations, by exploiting the normalized Laplacian matrix of the social graph, which captures the overall network structure. Instead, they consider only pathways of maximum length 2 between a target user and his candidate friends, which results to lower accuracy prediction as will be shown experimentally later.

Compared to global approaches (i.e Katz status index, RWR algorithm, SimRank algorithm etc.), which also operate on the overall structure of a network (i.e. initial adjacency matrix), our method is more efficient. The reason is that, our method is based on the top few eigenvectors and eigenvalues of the normalized Laplacian matrix, requiring less time and space complexity than the global algorithms, as will be shown in Section 4.4. Solving a standard eigenvalue problem for all eigenvectors takes O(n^3) operations, where n is the number of nodes in a graph. This becomes impractical for applications with n on the order of millions. However, real social networks have often the following properties [33]: 1) The graphs are often only locally connected and the resulting eigensystem is very sparse, and 2) only the top few eigenvectors are needed for graph partitioning. These special properties of our problem can be fully exploited by an eigensolver called the Lanczos method [13], resulting to faster time complexity than global algorithms.

Compared to traditional clustering algorithms, such as k-means and DBSCAN, which make explicit or implicit assumptions that clusters form globular (convex) regions in Euclidean space, the normalized laplacian matrix has some desirable properties that make it more suitable for real social networks, which often have non-connected components with non-globular shapes. Firstly, it is positive semi-definite with k non-negative real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_k$. The number of 0 eigenvalues equals the number of the connected components in a graph. Thus, spectral clustering is more flexible than k-means, in capturing (i) the non-connected components of a social graph, and (ii) a wider range of cluster geometries and shapes [37].

The contributions of our approach are summarized as follows: (i) For the first time, to the best of our knowledge, spectral clustering has been used for providing friend recommendations in OSNs. (ii) We provide more accurate friend recommendations than local approaches and k-means, by detecting a wider range of network structure and cluster geometries. This reveals the latent associations between users of a social network, as will be shown experimentally later. (iii) We provide higher efficiency than the global approaches. Our approach, by performing dimensionality reduction of the normalized Laplacian matrix, results to a smaller and more compact graph matrix than the original one, as will be also shown experimentally. (iv) We define two new node similarity measures that exploit local and global characteristics of a network. In particular, we calculate the similarity between nodes that belong in the same cluster and similarity between nodes that belong in different clusters. (v) Compared to the bulk of research on social networks that has focused almost exclusively on positive interpretations of links between people, we also study the interplay between positive and negative relationships. (vi) We perform extensive experimental comparison of the proposed method against existing link prediction algorithms, the k-means and two-way nCut clustering algorithms, using synthetic and three real data sets.

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. A motivating example, the proposed approach, an extension for signed networks and its complexity analysis, are described in Section 4. Experimental results are given in Section 5. Section 6 discusses important issues of our proposed method. Finally, Section 7 concludes this paper.

2 Related work

The mainstreaming class of algorithms for link prediction, known as similarity-based algorithms, are classified into local, quasi-local, and global approaches [27]. The local approaches focus mainly on the local nodes structure, whereas the global approaches, detect the overall path structure in a network. Quasi-local approaches traverse paths of a limited length between a target user and his candidate friends.

There is a variety of local-based similarity measures [24, 25], which are node-dependent (i.e. Common Neighbors index or else known as FOAF [6] algorithm, Adamic/Adar [2] index, Jaccard Coefficient, etc.) for analyzing the "proximity" of nodes in a network. Common Neighbors index, also known as Friend of a Friend algorithm (FOAF) [6], is adopted by many popular OSNs, such as facebook.com and hi5.com for the friend recommendation task. FOAF 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. In addition to FOAF algorithm, there are also more complicated local-based measures such as Jaccard Coefficient and Adamic/Adar index. Jaccard Coefficient [24] is a commonly used similarity metric in Information Retrieval. To measure proximity between two nodes v_x and v_y, Jaccard Coefficient measures the ratio of the number of common neighbors between v_x and v_y to the number of non-common neighbors. Adamic/Adar index [2], which is based on Jaccard Coefficient, measures how strongly "related" two web pages are. To do this, it exploits features of the web pages and defines a similarity measure between them, by refining the simple counting of common features (Jaccard Coefficient) by weighting rarer features more heavily. In the same direction, Davis et al. [9] introduced a novel probabilistically weighted extension of the Adamic/Adar measure for heterogenous information networks. Recently, Tsourakakis et al. [36] proposed a simple algorithm that for making link recommendations in online social networks by recommending those links that create as many triangles as possible.

There is a variety of global approaches [24] which are path-dependent (i.e Katz [20] status index, RWR [30] algorithm, SimRank [19] algorithm, the commute time [12] algorithm etc.). Leo Katz [20] introduced a status index derived from sociometric analysis. His method computes the important and influential nodes in a social network. He also used the concept of attenuation in influence transmitted through intermediaries nodes. RWR algorithm [30] (Random Walk with Restart algorithm) is based on a Markov-chain model of random walk through a graph. RWR considers a random walker that starts from node v_x and chooses randomly among the available edges every time, except that before making a choice, with probability c he goes back to node v_x (restart). Thus, the relevance score of node v_x with respect to node v_y is defined as the steady-state probability r_{v_x,v_y} that the random walker will finally stay at node v_y.

To provide a good tradeoff of recommendation accuracy (global approaches) and computational complexity (local approaches), there are also introduced variations of global approaches, which traverse only paths of a limited length. These approaches are known as quasi-local approaches [27]. For example, truncated Katz [27] (tKatz) is a truncated version of Katz [20] algorithm that chooses to stop after reaching paths of a maximum length ℓ_{max}. In the same direction, Symeonidis et al. [31] provides friend recommendations by exploiting paths of greater length than local approaches do (e.g. FOAF considers only pathways of maximum length 2 between a user and his candidate friends).

Spectral clustering, is a popular modern clustering algorithms. Its efficacy is mainly based on the fact that it does not make any assumptions on the form of the clusters [33,37]. This property comes from the mapping of the original space to an eigensystem. Due to this virtue, Spectral clustering is applied in many different research areas, such as bioinformatics [16] for clustering biological sequence data and computer imaging [33] for image segmentation.

There are two categories of spectral clustering algorithms based on the number of eigenvectors they use. The first category [33] uses a matrix of affinities between nodes and clusters the nodes based on the second smallest eigenvector of the Laplacian matrix. Then, recursively uses the second smallest eigenvector to further partition these clusters. The second category, which is similar to our method, directly computes a multi-way partition of the data [28,29]. In particular, it selects the top k eigenvectors and their corresponding eigenvalues. Then, it extracts the clusters by finding the approximate equal elements in the selected eigenvectors using any clustering algorithm e.g. k-means. Recently, Yan et al. [37] proposed a general framework for fast approximate spectral clustering in which a distortion-minimizing local transformation is first applied to the data. This framework is based on a theoretical analysis that provides a statistical characterization of the effect of distortion on the mis-clustering rate. Moreover, Abbassi and Mirronki [1] proposed a spectral method for designing a recommender system for blogs. However, the fact that they do not weight differently the similarities between nodes that belong to the same cluster and nodes that belong to different clusters is questionable. Finally, Jerome Kunegis and Andreas Lommatzsch [21] proposed a unified framework for learning link prediction and edge weight prediction functions in large networks, based on the transformation of a graph's algebraic spectrum.

The novelty of our approach compared to existing approaches is as follows: (i) Recently, extensive empirical analysis has demonstrated that FOAF [6] algorithm, performs better than other complicated variants [25] such as Adamic-Adar index and Jaccard Coefficient. Thus, we compare our method against FOAF algorithm as representative of the local-based measures, and as will be experimentally shown later, our method outperforms FOAF algorithm in terms of accuracy. The reason is that we do not take into account only pathways of length 2 to compute similarity between any pair of nodes in an OSN. Instead, we perform a dimensionality reduction of the entire social graph, which reveals the latent associations between the nodes of the graph. (ii) In comparison to global algorithms, such as the Katz index [20] and the Random Walk with Restart (RWR) algorithm [30], our method is more efficient. This means, that our method, which is based on the top few eigenvectors and eigenvalues of the normalized Laplacian matrix, requires less time and space complexity than global algorithms. We have compared our method against RWR, as representatives of the global algorithms, and as will be shown experimentally later, our method outperforms RWR in terms of accuracy and time complexity. (iii) In comparison to traditional clustering algorithms, such as k-means and DBSCAN, our method is more flexible, because it captures (i) the non-connected components of a social graph, and (ii) a wider range of cluster geometries and shapes [37]. Thus, it results in better friend recommendations. We have compared our method against k-means, as representatives of the clustering algorithms, and as will be shown experimentally later, our method is more effective than k-means.

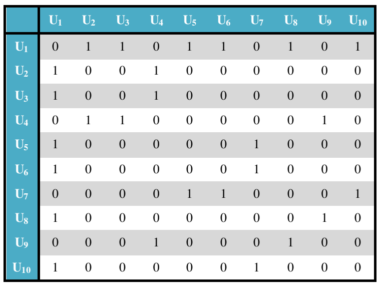
Besides the aforementioned link prediction algorithms that are based solely on the graph structure, there are alternative methods that exploit other data sources such as messages among users, co-authored paper, common tagging etc. For instance, Ido Guy et al. [14], proposed a novel user interface widget for providing users with recommendations of people. Their people recommendations were based on aggregated information collected from various sources across IBM organization (i.e. common tagging, common link structure, common co-authored papers etc.). Chen et al. [6] evaluated four recommender algorithms (Content Matching, Content-plus-Link, FOAF algorithm and, SONAR) to help users discover new friends on IBM's OSN. Lo and Lin [26] proposed two algorithms, denoted as *weighted minimum message ratio* (WMR) and *weighted information ratio* (WIR), respectively, which generate a friend list based on real-time message interaction among members of an OSN. Furthermore, Agarwal and Bharadwaj [3] proposed a collaborative filtering framework for friends recommendation in social networks based on their interaction intensity and adaptive user similarity.

There are also other methods [4, 22] that try to solve the link prediction problem as a binary classification problem, where the two class labels are the existence or not of a link between two nodes. These methods usually exploit several non-topological features (e.g. overlap of interest between two authors, keyword match count between author's publications etc.) For example, Hasan et al. [4] compared well-known classification algorithms (decision tree, k-NN, multilayer perceptron, SVM, RBF network) and found that they improve the accuracy of link prediction substantially. In comparison to the above methods, we focus only on recommendations based on the link structure of an OSN (i.e. we use features based on network topology only) and thus, we will exclude them from our experimental comparison.

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 unweighted graph as shown in Figure 1. $\mathcal{G}$ expresses friendships among users of an OSN and will be used as our running example, throughout the paper.

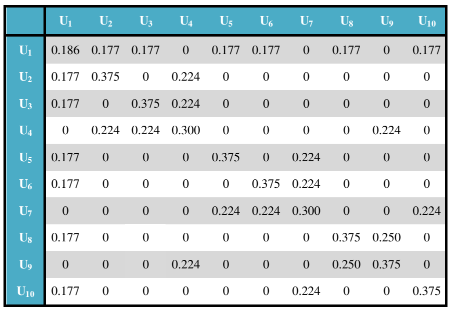
The adjacency matrix $\mathcal{A}$ of graph $\mathcal{G}$ is a matrix with rows and columns labelled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are friends or not. For an undirected graph, the adjacency matrix is symmetric. In Figure 2, we present the resulting adjacency matrix $\mathcal{A}$ of graph $\mathcal{G}$. Notice that in $\mathcal{A}$ we use zeros along the diagonals, to depict that a node is not connected to itself. In case of a large graph $\mathcal{G}$, it is important to note that its adjacency matrix $\mathcal{A}$ can be characterized by high dimensionality and sparsity.



	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}
U_1	0	1	1	0	1	1	0	1	0	1
U_2	1	0	0	1	0	0	0	0	0	0
U_3	1	0	0	1	0	0	0	0	0	0
U_4	0	1	1	0	0	0	0	0	1	0
U_5	1	0	0	0	0	0	1	0	0	0
U_6	1	0	0	0	0	0	1	0	0	0
U_7	0	0	0	0	1	1	0	0	0	1
U_8	1	0	0	0	0	0	0	0	1	0
U_9	0	0	0	1	0	0	0	1	0	0
U_{10}	1	0	0	0	0	0	1	0	0	0

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

The spectral algorithms are based on eigenvectors of Laplacians, which are a combination of the adjacency and the degree matrix. The normalized laplacian matrix of graph $\mathcal{G}$ is computed by Equation $\mathcal{L} = \mathcal{D}^{-\frac{1}{2}} \times (\mathcal{D}$ - $\mathcal{A}) \times \mathcal{D}^{-\frac{1}{2}}$, where $\mathcal{D}$ is the degree matrix of graph $\mathcal{G}$. The normalized laplacian matrix $\mathcal{L}$ is positive semi-definite and has n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$. As shown in Figure 3, we have computed the corresponding normalized laplacian matrix of our running example.



	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}
U_1	0.186	0.177	0.177	0	0.177	0.177	0	0.177	0	0.177
U_2	0.177	0.375	0	0.224	0	0	0	0	0	0
U_3	0.177	0	0.375	0.224	0	0	0	0	0	0
U_4	0	0.224	0.224	0.300	0	0	0	0	0.224	0
U_5	0.177	0	0	0	0.375	0	0.224	0	0	0
U_6	0.177	0	0	0	0	0.375	0.224	0	0	0
U_7	0	0	0	0	0.224	0.224	0.300	0	0	0.224
U_8	0.177	0	0	0	0	0	0	0.375	0.250	0
U_9	0	0	0	0.224	0	0	0	0.250	0.375	0
U_{10}	0.177	0	0	0	0	0	0.224	0	0	0.375

Fig. 3 Normalized Laplacian matrix $\mathcal{L}$ of graph $\mathcal{G}$.

Table 1 presents the most important symbols and their corresponding definitions, which are used frequently in the sequel.

4 The Proposed Approach

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

Symbol	**Description**
$\mathcal{G}$	undirected and unvalued graph
$\mathcal{V}$	set of graph nodes (vertices)
$\mathcal{E}$	set of graph edges
$\mathcal{A}$	adjacency matrix of graph $\mathcal{G}$
$\mathcal{D}$	degree matrix of graph $\mathcal{G}$
$\mathcal{L}$	normalized laplacian matrix of graph $\mathcal{G}$
u_i	eigenvector of $\mathcal{L}$
λ_i	eigenvalue of $\mathcal{L}$
v_i	graph node
e_i	graph edge
$sim(v_i, v_j)$	similarity between nodes v_i and v_j
n	number of vertices in graph $\mathcal{G}$

Table 1 Symbols used throughout the study.

4.1 Outline

Here, we describe how SpectralLink is applied on OSNs and how the recommendation of friends is performed according to the detected associations.

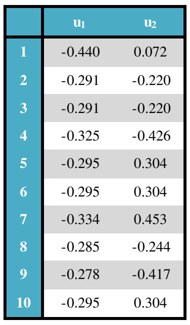
When using an OSN, users explicitly declare their friends so that they are able to share information items with them (i.e. photos, news etc.) After some time, the social network accumulates a set of connection data (graph of friendships), which can be represented by an undirected graph similar to that of Figure 1.

Our SpectralLink approach finds similarities between nodes in an undirected graph constructed from these connection data. The SpectralLink 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 SpectralLink algorithm to our running example. As illustrated in Figure 1, 10 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 Figure 2. However, after performing the SpectralLink algorithm, we can get a similarity matrix between any two nodes of graph $\mathcal{G}$ and recommend friends according to their weights.

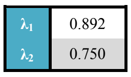
Firstly, SpectralLink computes the first k eigenvectors $u_1, \ldots, u_k$ with the corresponding $\lambda_1, \ldots, \lambda_k$ eigenvalues of $\mathcal{L}$ based on Equation $\mathcal{L} \times \mathcal{U} = \lambda \times \mathcal{U}$, where $\mathcal{U}$ matrix has columns the eigenvectors $u_1, \ldots, u_k$ and nodes $v_i \in R$, with $i = 1, \ldots, n$, correspond to the i-row of $\mathcal{U}$. In our running example, we compute the first k=2 eigenvectors and $\lambda = 2$ eigenvalues of $\mathcal{L}$, respectively, as shown in Figure 4a and Figure 4b.

Secondly, we cluster nodes v_i of $\mathcal{U}$ with the k-means algorithm into clusters $C_1, \ldots, C_k$. In our running example, k is equal to 2. Thus, we will partition data in 2 clusters. In Figure 5a, we present vector IDX with $i = 1, \ldots, n$ rows, which correspond to the assignment of a node v_i in one of the two clusters. Thus, node U_1 is assigned in cluster C_1, node U_2 is assigned in cluster C_2, etc. Moreover, based on the k-means algorithm, we can compute the centroids of each cluster. This information is shown, in Figure 5b. Based on the distances of each node from each cluster centroid we can define matrix D, which is shown in Figure 5c. Vector IDX



	u_1	u_2
1	-0.440	0.072
2	-0.291	-0.220
3	-0.291	-0.220
4	-0.325	-0.426
5	-0.295	0.304
6	-0.295	0.304
7	-0.334	0.453
8	-0.285	-0.244
9	-0.278	-0.417
10	-0.295	0.304

(a)

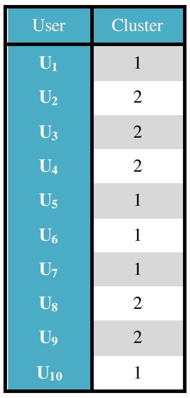


λ_1	0.892
λ_2	0.750

(b)

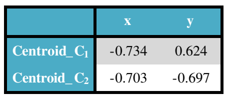
Fig. 4 In our running example, we computed the (a) first k=2 eigenvectors and (b) the first $\lambda = 2$ eigenvalues of $\mathcal{L}$.

and matrix D will be used in the next step of SpectralLink in order to calculate the similarity between nodes that belong to the same cluster and similarity between nodes that belong to different clusters.



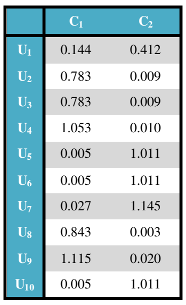
User	Cluster
U_1	1
U_2	2
U_3	2
U_4	2
U_5	1
U_6	1
U_7	1
U_8	2
U_9	2
U_{10}	1

(a)



	x	y
Centroid_C_1	-0.734	0.624
Centroid_C_2	-0.703	-0.697

(b)



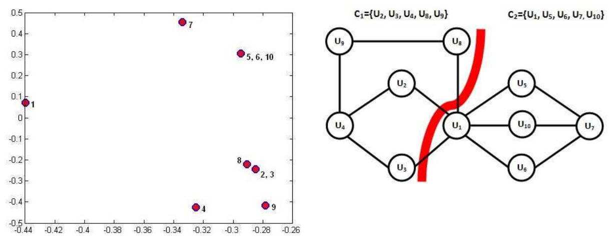
	C_1	C_2
U_1	0.144	0.412
U_2	0.783	0.009
U_3	0.783	0.009
U_4	1.053	0.010
U_5	0.005	1.011
U_6	0.005	1.011
U_7	0.027	1.145
U_8	0.843	0.003
U_9	1.115	0.020
U_{10}	0.005	1.011

(c)

Fig. 5 For our running example, we computed the (a) vector IDX which assigns each u_i node in a specific cluster, (b) the coordinates in the 2-D space of each cluster centroid and (c) matrix D with the distances of each node from the centroid of each cluster.

Moreover, in Figure 6a, we present the 10 nodes of our running example, in the 2-dimensional space based on the first 2 eigenvectors of $\mathcal{L}$ matrix. Additionally, in Figure 6b, we present the resulting partition of the 10 nodes of graph $\mathcal{G}$ in 2 clusters. Thus, the nodes that are assigned in cluster $C_1 = \{U_2,U_3,U_4,U_8,U_9\}$, whereas the nodes that are assigned in cluster $C_2 = \{U_1,U_5,U_6,U_7,U_{10}\}$. As shown, the partition of k-means is in accordance with the visual representation in the 2-dimensional space of the nodes in Figure 6a.

Thirdly, to quantify the similarity between nodes, we are based on triangle inequality which states that for any triangle the sum of the lengths of any two sides



(a) (b)

Fig. 6 For our running example, we present the (a) 2-D space plot of nodes of graph $\mathcal{G}$ based on the second eigenvector of $\mathcal{L}$ and (b) the resulting partition of the 10 nodes of graph $\mathcal{G}$ in 2 clusters.

must be greater than the length of the remaining side. Since we have calculated matrix D with the distances of each node from the centroid of each cluster, based on triangle inequality the distance (i.e. dissimilarity) between any pair of nodes i and j is bounded in this space: $|D(i, IDX(i)) - D(j, IDX(j))| \leq dist(i,j) \leq D(i, IDX(i)) + D(j, IDX(j))$.

For similarity bounded by 0 and 1, when similarity is one (i.e. exactly similar), the distance (dissimilarity) is zero and when the similarity is zero (i.e. very different), the dissimilarity is one. Thus, to quantify the similarity between nodes that belong to the same cluster, SpectralLink uses Equation 1:

$$SimSC(i,j) = 1 - |D(i, IDX(i)) - D(j, IDX(j))| \tag{1}$$

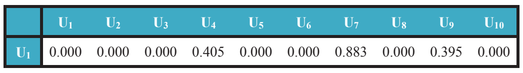
The intuition behind Eq. 1 is that if the lower bound of the reverse triangular inequality (i.e. $|D(i, IDX(i)) - D(j, IDX(j))|$) is very small (i.e. close to 0) then the similarity between nodes i and j will be very high (close to 1). In our running example the similarity between nodes U_1 and U_7 that belong to same cluster C_1 based on Equation 1 is : $1 - |0.144 - 0.027| = 0.883$.

Moreover, to quantify the similarity between nodes that belong to different clusters we use Equation 2:

$$SimDC(i,j) = \frac{1}{1 + D(i, IDX(j)) + D(j, IDX(i))}, \tag{2}$$

where $SimDC(i,j)$ is bounded in [0,1]. Thus, in our running example the similarity between nodes U_1 and U_4 that belong to different clusters based on Equation 2 is : $\frac{1}{1+D(1,2)+D(4,1)} = \frac{1}{1+0.412+1.053} = 0.405$. It is obvious that Equation 1 promotes similarity between nodes that belong to the same cluster. In contrast, Equation 2 penalizes similarities between nodes that belong to different clusters. Notice that for the friend recommendation task, we have to rank similarities and recommend those friends that have the highest similarity with the target user. It is obvious that most of recommended friends should be within the same cluster with the one that the target user belongs to. There will be recommended friends from different clusters when the number of recommendations exceeds the number of users that are in the same cluster with the target user.

In Figure 7, we present the similarities of user U_1 with other nodes. For readability reasons, we put zero values to already adjacent nodes. In our running example, as shown in Figure 7, user U_1 would receive user U_7 as friend recommendation. The resulting recommendation is reasonable, because U_1 has 3 common friends with user U_7. In contrast, U_1 has only 2 common friends with user U_4. That is, the SpectralLink 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.



	U₁	U₂	U₃	U₄	U₅	U₆	U₇	U₈	U₉	U₁₀
U₁	0.000	0.000	0.000	0.405	0.000	0.000	0.883	0.000	0.395	0.000

Fig. 7 In our running example, similarities between U_1 and other users. It presents the possibility of two users being friends.

4.2 The SpectralLink Algorithm

In this section, we describe our SpectralLink algorithm in detail. Our Spectrallink algorithm computes node similarity for a node v_i with each node v_j in a graph $\mathcal{G}$.

As shown in Figure 8, our SpectralLink algorithm is based on matrix $\mathcal{L}$ of a graph $\mathcal{G}$. It takes the first k eigenvectors $u_1, \ldots, u_k$ of $\mathcal{L}$. Then, based on these eigenvectors it clusters nodes $v_1 \ldots v_n$ of graph $\mathcal{G}$ with k-means algorithm. Next, based on the distances of each node v_i from the nearest cluster centroid it calculates similarities between a test user node and the other nodes in graph $\mathcal{G}$. Finally, for a test user (node) we rank the calculated similarities with other users (nodes) and recommend to him the top ranked nodes as his possible friends.

4.3 Extending SpectralLink for Signed Networks

In this Section, we derive variants of SpectralLink that apply to networks with weighted edges, including the case of edges with negative weights (signed networks).

In signed networks edges have positive (+1) as well as negative (-1) weights. Such signed graphs arise for instance in social networks (i.e. Epinions.com, Shashdot Zoo, etc.) where negative edges denote distrust instead of trustiness. In such signed graphs, SpectralLink algorithm, can be adjusted accordingly. Firstly, we can use an alternative definition of diagonal degree matrix [17,21] by using the absolute diagonal degree matrix $\mathcal{D}_{ii}=\sum_{j=1}^{n}|\mathcal{A}_{ij}|$. Thus, $\mathcal{D}_{ii}$ is defined as the sum of the absolute weights of incident edges. This means that if all edge weights are in $\{+1, -1\}$, $\mathcal{D}_{ii}$ is simply the number of nodes adjacent to the node i, regardless of edge signs.

Then, we can define the signed normalized Laplacian matrix, by giving $\mathcal{L} = \mathcal{D}^{-\frac{1}{2}} \times (\mathcal{D} - \mathcal{A}) \times \mathcal{D}^{-\frac{1}{2}}$. As the unsigned normalized Laplacian matrix, the signed normalized Laplacian matrix is positive semi-definite (i.e., it has non-negative eigenvalues, where its smallest eigenvalue is zero) and it can therefore be used as a kernel.

```
Algorithm SpectralLink ($v$, $\mathcal{G}$, $\mathcal{A}$, $n$, $k$)
Input
    $v$: the selected node
    $\mathcal{G}$: an undirected and unweighted graph
    $\mathcal{A}$: adjacency matrix of graph $\mathcal{G}$,
    $n$: number of node of graph $\mathcal{G}$,
    $k$: number of clusters
Output
    $\mathbf{v}_{v_j}$: similarity between node $v$
    with each node $v_j$ in $\mathcal{G}$
------------------------------------------------------------
1. Compute the diagonal degree matrix $\mathcal{D}$ with elements:
        $\mathcal{D}_{ii}=\sum_{j=1}^{n}\mathcal{A}_{ij}$
2. Compute the normalized Laplacian matrix:
        $\mathcal{L} = \mathcal{D}^{-\frac{1}{2}} \times (\mathcal{D}$ - $\mathcal{A}) \times \mathcal{D}^{-\frac{1}{2}}$
3. Find the first $k$ eigenvectors $u_1, \dots, u_k$ of $L$
4. Let matrix $\mathcal{U} \in \mathcal{R}$ contain $u_1, \dots, u_k$ eigenvectors
    as columns and nodes $v_i \in R$, with $i = 1, \dots, n$,
    correspond to the $i$-row of $\mathcal{U}$
5. Cluster the nodes $v_i$ with k-means algorithm into
    clusters $C_1, \dots, C_k$
6. For node $v$ compute its similarity
    with each node $v_j$ that belongs in the same cluster
    based on Equation 1
7. For node $v$ compute its similarity
    with each node $v_j$ that belongs to a different cluster
    based on Equation 2
```

Fig. 8 The SpectralLink algorithm.

However, when a signed graph is unbalanced (i.e., each connected component contains a cycle with an odd number of negatively weighted edges), then the signed normalized Laplacian matrix can be positive-definite (i.e., it has only positive eigenvalues). Notice that based on the assumption of multiplicative transitivity of the structural balance theory [15,22], an unbalanced graph cannot be partitioned into two sets with all negative edges across the sets and positive edges within the sets. Moreover, if the smallest eigenvalue of a graph is greater than zero it characterizes the amount of conflict in a graph, which is an indication of how well can be clustered the nodes of the graph.

Structural balance theory [15], 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". Concretely, this means that if v_k forms a triad with the edge (v_i,v_j), then structural balance theory posits that v_i,v_j should have that sign that causes the triangle on v_i,v_j,v_k to have an odd number of positive signs, just as each of the principles above have a odd number of occurrences of the word "friend". In other words, sim(v_i,v_j) = sim(v_i,v_k) * sim(v_i,v_k), as shown in Figure 9.

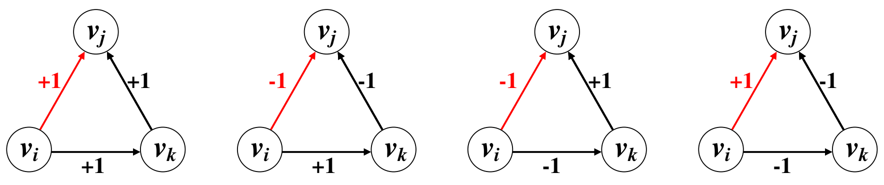


Fig. 9 The prediction of the sign of edge v_i,v_j (red edge) based on the Structural Balanced Theory.

Figure 9 shows only the cases where a triangle is balanced. In particular, a triangle of three positive edges and a triangle of one positive and two negative edges are both balanced. On the other hand, a triangle of two positive and one negative edge is not balanced, as is a triangle of three negative edges. Notice that, in an unbalanced triangle, the multiplicative transitivity of the structural balance theory [15, 22] does not hold.

4.4 Complexity Analysis

Solving a standard eigenvalue problem for all eigenvectors takes O(n^3) operations, where n is the number of graph nodes. This becomes impractical for applications with n on the order of millions. However, real social networks have often the following properties: 1) The graphs are often only locally connected and the resulting eigensystems are very sparse, and 2) only the top few eigenvectors are needed for graph partitioning. These special properties of our problem can be fully exploited by an eigensolver called the Lanczos method [13]. The time complexity of a Lanczos algorithm is O($m \times n$) + O($m \times M(n)$), where m is a usually small constant number of matrix-vector computations required, n is the number of nodes in a graph, and $M(n)$ is the cost of a matrix-vector computation of $\mathcal{L} \times x$, where $\mathcal{L}$ is the normalized Laplacian matrix and x is an eigenvector.

5 Experimental Evaluation

In this Section, we compare experimentally our SpectralLink algorithm with 7 other link prediction algorithms. In particular, we use in the comparison the K-means, the Two-way Ncut Clustering [33], the Friend of a Friend [6], the Adamic and Adar [2], the Random Walk with Restart [30], the Katz [20] status index, and the Regularized commute-time [11] algorithms. All algorithms were implemented in Matlab.

5.1 Algorithms Settings

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

K-means clustering algorithm: Given a set of nodes (v_1, v_2, $\ldots$,v_n) of a graph $\mathcal{G}$ and its adjacency matrix, k-means aims to partition the n nodes into k sets ($k < n$) C=(C_1, C_2,$\ldots$, C_k) so as to minimize the within-cluster sum of squared error(SSE), as shown by Equation 3:

$$SSE = \sum_{i=1}^{k} \sum_{v_x \in C_i} dist(v_x, c_i)^2, \tag{3}$$

where v_x is a node in cluster C_i, c_i is the centroid point for cluster C_i and *dist* is the standard Euclidean distance between two nodes in Euclidean space. k-means chooses k initial centroids, where k is a user-specified parameter, namely, the number of clusters desired. Each node is then assigned to the closest centroid, and each collection of nodes assigned to a centroid is a cluster. The centroid of each cluster is then updated based on the nodes assigned to the cluster. This procedure is repeated until no node changes cluster, or equivalently, until the centroids remain the same. After the clusters formation, for a node v_i compute its similarity with each node v_j that belongs in the same cluster based on Equation 1. Moreover, for a node v_i compute its similarity with each node v_j that belongs to a different cluster based on Equation 2.

Two-way Ncut clustering algorithm: Given a set of nodes (v_1, v_2, $\ldots$,v_n) of a graph $\mathcal{G}$, and its adjacency matrix, two-way normalized cut (two-way Ncut) algorithm aims to bipartition the n nodes so as to minimize the Normalized Cut [33]. In particular, two-way Ncut solves the generalized eigenvalue problem for the second smallest eigenvalue, as shown by Equation 4:

$$(D - A)y = \lambda D y, \tag{4}$$

where D is a diagonal matrix, A is the adjacency matrix, λ is the second smallest eigenvalue and y is the second smallest eigenvector. Two-way Ncut uses the eigenvector with the second smallest eigenvalue to bipartition the graph, and decides if the current partition should be subdivided again by checking *Ncut* variable (i.e. if it is below a pre-specified value).

Friend of a Friend algorithm: The Friend of a Friend (FOAF) algorithm [6] relies on the number of friends that two nodes v_x and v_y have in common, as shown by Equation 5:

$$score(v_x, v_y) := |\Gamma(v_x) \cap \Gamma(v_y)| \tag{5}$$

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.

Adamic/Adar algorithm: Adamic and Adar [2] 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: $\sum_z \frac{1}{\log(frequency(z))}$, 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 Equation 6:

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

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

Random Walk with Restart Algorithm: Random Walk with Restart algorithm [35, 30] 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 Equation 7:

$$Kernel_{RWR} = (I - \alpha P)^{-1} \tag{7}$$

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

Katz status index algorithm: Katz [20] 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 Equation 8:

$$score(v_x, v_y) = \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot \left| paths^{\ell}_{v_x,v_y} \right|, \tag{8}$$

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

The Regularized Commute-Time kernel: The Regularized Commute-Time kernel [11] performs a regularization on the commute-time kernel [12]. 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 Equation 9:

$$Kernel_{RCT} = (D - \alpha A)^{-1} \tag{9}$$

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

Table 2 summarizes the algorithms used in the experimental evaluation. The second column of Table 2 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 2.

Table 2 The algorithms used in the comparison with their parameters and test values.

Algorithm	**Abbreviation**	**Equation**	**Parameter**	**Test Values**
SpectralLink	SpectralLink	(1,2)	k	$100, 200, 300, \dots, 5000$
K-means	K-means	(3)	k	$100, 200, 300, \dots, 5000$
Two-way Ncut Clustering [33]	Ncut	(4)	$Ncut$	$2^{-24}, 2^{-52}$
Friend of a Friend [6]	FOAF	(5)	-	-
Adamic/Adar [2]	AA	(6)	-	-
Random Walk with Restart [30,35]	RWR	(7)	α	$10^{-6}, 10^{-5}, \dots, 0.99$
Katz Status Index [20]	Katz	(8)	β	0.05,0.005,0.0005
Regularized Commute Time [11]	RCT	(9)	α	$10^{-6}, 10^{-5}, \dots, 0.99$

5.2 Real and Synthetic Evaluation Data Sets

To evaluate the examined algorithms, we have used 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=13692), denoted as Facebook 3.7K[4], 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. 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 63329 users and 88261 edges among them, denoted as Hi5 63K[5], 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 16512 new emerged edges connecting them. The graph data from the first crawl are used to predict the new links emerging in the second crawl. Moreover, we use in our comparison the Epinions[6] 132K data set, which is a who-trusts-whom social network that consist of positive and negative edges. A positive edge implies trust whereas a negative edge implies distrust. Notice that we have also run experiments with a human protein-protein interaction network [18] and the results are similar and analogous with those that will be presented in the following for the friendship social networks.

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. To study the algorithms' computational complexity performance, we used synthetic networks models of different sizes.

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. This is in accordance with characteristics of friendship social networks, such as Balance (Friend of a Friend) and Homophily (Birds of a feather) Theories [10]. We used the generator proposed in [34] and created 2 synthetic data sets based on different network sizes n (50000, 100000), by keeping an identical m nodes degree equal to 50 and for both data sets (p is fixed to 0.2). Finally, we calculated several topological properties of the derived synthetic and the real data set which are presented in Figure 10.

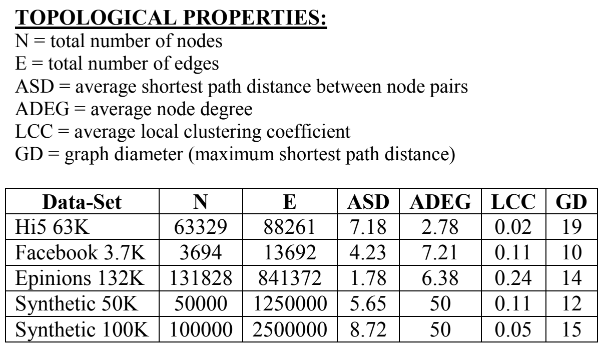
5.3 Experimental Protocol and Evaluation Metrics

As already described in Section 5.2, 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

[4] http://delab.csd.auth.gr/∼symeon/facebook.txt

[5] http://delab.csd.auth.gr/∼symeon/hi5.txt

[6] http://www.trustlet.org/wiki/Downloaded_Epinions_dataset



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	N	E	ASD	ADEG	LCC	GD
Hi5 63K	63329	88261	7.18	2.78	0.02	19
Facebook 3.7K	3694	13692	4.23	7.21	0.11	10
Epinions 132K	131828	841372	1.78	6.38	0.24	14
Synthetic 50K	50000	1250000	5.65	50	0.11	12
Synthetic 100K	100000	2500000	8.72	50	0.05	15

Fig. 10 Topological properties of the synthetic and the real data set.

set is allowed to be used for prediction. It is obvious that $\mathcal{E}^T \cap \mathcal{E}^P = \oslash$. 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.

We use the classic precision/recall metric as performance measure for friend recommendations. For a test user receiving a list of n recommended friends (top-n list), precision and recall are defined as follows:

Precision is the ratio of the number of relevant users in the top-n list (i.e., those in the top-n list that belong to the future set of friends of the target user) to n.

Recall is the ratio of the number of relevant users in the top-n list to the total number of relevant users (all friends in the future set of the target user).

Moreover, since we provide to a target user u a top-n 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 Equation 10:

$$MAP = \frac{1}{|N|} \sum_{u=1}^{|N|} \frac{1}{r_u} \sum_{k=1}^{r_u} Precision_u@k \tag{10}$$

where N is the number of users in the test data set, r_u is the number of relevant users to a user u and $Precision_u@k$ is the precision value at the k-th 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 [7]. AUC is equivalent to the area under the receiver-operating characteristic (ROC) curve [7]. 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 Equation 11:

$$AUC = \frac{n' + 0.5 \times n''}{n} \tag{11}$$

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 5.7.

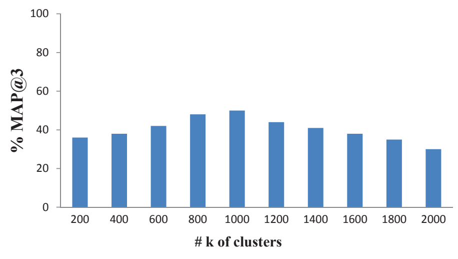
5.4 Sensitivity Analysis for the SpectralLink Algorithm

In this Section, we study the sensitivity of SpectralLink accuracy performance in a synthetic and a real network (i) with different k number of clusters and (ii) with different controllable sparsity.

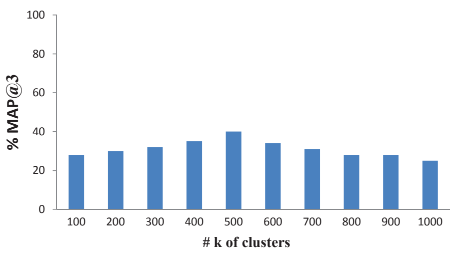
In Section 4.2, one of the required input values for the SpectralLink algorithm is the number k of clusters. To improve our recommendations in terms of effectiveness, it is important to fine-tune the k variable. For the synthetic 50K data set, we examine the performance of MAP metric when we recommend a top-3 friend list (i.e. %MAP@3) vs. different values of k. Figure 11a illustrates MAP for varying k values in the synthetic 50K data set. As expected, the best MAP performance of SpectralLink is attained with $k = 1000$ clusters. The reason is the average nodes degree (ADEG) of the 50K data set, which is equal to 50. Thus, with a number k=1000 of clusters, we get an average cluster size, which corresponds to ADEG of this data set. In the following, we keep $k = 1000$ as the default initial value for the SpectralLink algorithm for this data set.

For the Facebook 3.7K data set, we follow the same tuning procedure. Figure 11b illustrates MAP values for varying number k of clusters. The best result is attained for $k = 500$. Once again, the initial k number is analogous to ADEG (i.e. 7.21) and the network size (i.e. 3694/7.21 = 512). We have to notice that the number of selected clusters could reduce the gains over the predicting accuracy. This means that our method requires a fine-tuning on the number of selected clusters. However, the final number of selected clusters can be easily estimated by dividing the N number of nodes in a graph with ADEG.

Next, we measure the accuracy that SpectralLink attains, with different controllable sparsity. To examine the accuracy performance of SpectralLink in terms of different network sparsity, we have created for the 50K synthetic data set 5 different sparsity cases, by changing the m number of friends that a node has (50,



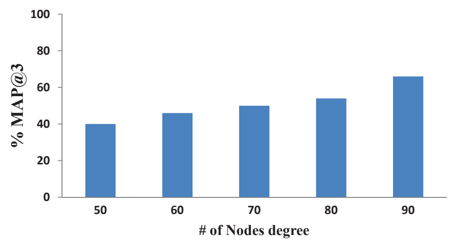
(a)



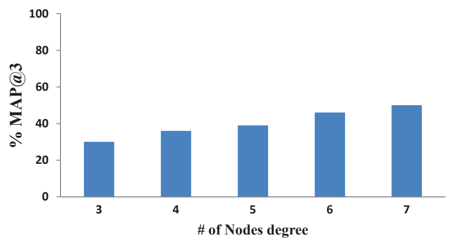
(b)

Fig. 11 MAP diagrams for data sets (a) Synthetic 50K, and (b) Facebook 3.7K.

60, 70, 80, 90), as shown in Figure 12a. As expected, with k increasing, the MAP increases too.



(a)



(b)

Fig. 12 MAP diagram for the synthetic 50K and the Facebook 3.7 data sets presenting the increase in MAP values when a larger amount of neighbors is known for a data set. The data sets represented are the (a) Synthetic 50K, and (b) Facebook 3.7K.

For the Facebook 3.7K data set, we also examine 5 different sparsity cases, by changing the m number of friends that a node has (i.e. 3, 4, 5, 6, 7), as shown in Figure 12b. As expected, the best MAP value is attained when we consider more adjacent nodes (i.e. m equal to 7). This is reasonable since the ADEG of Facebook 3.7K data set is equal to 7.21. SpectralLink can predict more effectively new friends for larger m values, since in such cases the network density is increased.

5.5 Accuracy Comparison of SpectralLink with other methods

In this Section, we compare SpectralLink against K-means, Two-way Ncut, FOAF, AA, Katz, RWR, and RCT algorithms. Table 3 presents the MAP values of the tested algorithms for the Facebook 3.7K and Hi5 63K data sets, respectively, when we recommend three friends. As shown, SpectralLink outperforms the other algorithms in both real data sets. The reason is that SpectralLink outperforms K-means, because it takes into consideration also the degree of connectivity of a graph. Moreover, SpectralLink is more flexible than k-means, because it captures a wider range of cluster geometries and shapes and not only cyclic clusters. As also shown, SpectralLink outperforms Two-way Ncut because the latter relies only on the second eigenvector, cutting the subsequent eigenvectors, which might be perfect partitioning vectors. Furthermore, RWR, Katz and RCT traverse globally the social network, failing to capture adequately the local characteristics of the graph. Finally, FOAF and AA fail to provide accurate recommendations because they exploit only local characteristics of the graph. Notice that MAP values are high for Facebook 3.7 data set. The main reason is the topological characteristics of the graph (i.e. high LCC and small ASD). It can be considered as a small-world network [5]. That is the network is strongly localized with most of paths being of short geographical lengths. 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 the topological characteristics of Hi5 63K data set (i.e. high ASD=7.18 and small ADEG=2.78). Based on this characteristics, Hi5 63K cannot be considered as a small-world network. Thus, it is not well-connected and results in lower recommendation accuracy.

Algorithm	Facebook 3.7K	Hi5 63K
SpectralLink	**0.395**	**0.161**
K-means	0.334	0.115
Two-way Ncut	0.365	0.135
FOAF	0.105	0.021
AA	0.125	0.054
RWR	0.225	0.085
Katz	0.205	0.075
RCT	0.315	0.121

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

5.6 Time Comparison of SpectralLink with other Methods

In this Section, we compare SpectralLink, K-means, Two-way Ncut, FOAF, AA, Katz, RWR, and RCT algorithms in terms of efficiency using two synthetics and two real data sets. We have created 2 synthetic data sets based on different network sizes n (50000, 100000), by keeping an m nodes degree equal to 50 for both data sets. Then, we measured the clock time for the off-line parts of all algorithms. The off-line part refers to the average computation time for calculating the similarities for a target node. The results are presented in Table 4.

As shown, SpectralLink outperforms two-way Ncut. The reason is that two-way Ncut is computationally wasteful, since it is a recursive algorithm and only the second eigenvector is used in each bipartition. Moreover, SpectralLink outperforms RWR, Katz and RCT, which presents the worst time complexity because it calculates the inverse of an $n \times n$ matrix, whereas SpectralLink performs calculations on the decomposed normalized Laplacian matrix. As expected, FOAF outperforms the other algorithms due to its simpler complexity. However, as already shown in Section 5.5, FOAF performs the worst results in terms of accuracy prediction. This means, that it is not suitable for the friend recommnedation task, even if FOAF presents small time complexity.

Algorithm	**Synthetic 50K**	**Synthetic 100K**	**Facebook 3.7K**	**Hi5 63K**
SpectralLink	0.456 sec	0.825 sec	0.085 sec	0.562 sec
K-means	0.532 sec	1.152 sec	0.105 sec	0.745 sec
Two-way Ncut	0.499 sec	0.987 sec	0.098 sec	0.652 sec
FOAF	**0.150 sec**	**0.280 sec**	**0.029 sec**	**0.179 sec**
AA	0.164 sec	0.298 sec	0.036 sec	0.197 sec
RWR	0.833 sec	1.621 sec	0.135 sec	1.106 sec
Katz	0.945 sec	1.711 sec	0.152 sec	1.198 sec
RCT	0.897 sec	1.653 sec	0.145 sec	1.156 sec

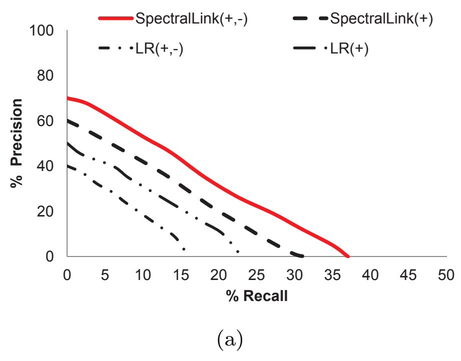
Table 4 Time comparison of all tested algorithms for the synthetic and real data sets.

5.7 SpectralLink Effectiveness in Signed Networks

In this Section, we present the accuracy performance of SpectralLink when we take into account positive and negative links of a signed network, i.e. extended Epinions 132K data set.

We will use as comparison partner in our experiments, a model proposed by Jure Leskovec et al. [22] that applies featured-based classification for predicting links. In particular, their model uses the logistic regression as a classifier, which is trained based on local topological graph features, and predicts whether there exists an edge between a pair of nodes or not (two class labels). These features/variables take into account the existence of negative links. Based on Status theory [23, 22], the positive nodes' in-degree $deg^{+}_{in}(x)$ and the negative nodes' in-degree $deg^{-}_{out}(x)$ of a node x increase its status. In contrast, the positive nodes' out-degree $deg^{+}_{out}(x)$, and the negative nodes out-degree $deg^{-}_{in}(x)$ decrease its status. Thus, by exploiting these features we can run a logistic regression model to predict new friends based on the existence of edges that imply enmity or distrust. We have to notice that Leskovec et al. [23, 22] concentrate in the "Edge Sign Prediction" problem. However, their model can be also applied in the "Link Prediction" problem. Henceforth, their method that takes into account only positive links is denoted as LR(+), whereas their method that takes into account both positive and negative links is denoted LR(+,−). In our model, we have two different variants of SpectralLink: The first variation considers only positive links and is denoted as SpectralLink(+). The second variation considers both positive and negative links and is denoted as SpectralLink(+,−).

Figure 13a presents the precision and recall diagram for both versions of SpectralLink and LR, whereas Figure 13b presents the AUC accuracy statistic. Both Figures show that SpectralLink(+,−) outperforms SpectralLink(+). The reason is that SpectralLink(+,−) 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 SpectralLink 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. Finally, SpectralLink(+,−) outperforms LR(+,−). The reason is that it exploits the normalized Laplacian matrix of the social graph, which captures the overall network structure. In contrast, LR(+,−) focuses only on local topological graph characteristics (e.g. nodes' degree).



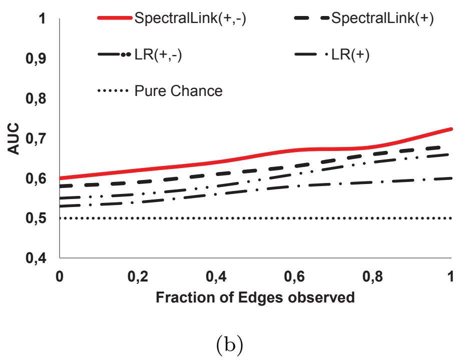


Fig. 13 Accuracy performance of SpectralLink and LR in terms of (a) precision/recall and (b) AUC statistic.

6 Discussion

A basic problem of social networks is data sparsity [32]. Data sparsity can be increased by the so-called "losing links". That is, users of social networks usually remove friends from their friends' list, when they do not anymore keep contacts with them. However, the prior probability of a link is typically quite small for building a statistical model. To overcome this limitation, we studied a synthetic network model with controllable sparsity. As it was experimentally shown, a higher network sparsity harms all algorithms' accuracy performance.

Real networks have many complex structural properties [8], 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 [8] with a probability p, a synthetic network with different level of degree heterogeneity can be composed.

This paper concerns unweighted and undirected networks. However, our algorithm can be easily extended to more general cases. For example, we can handle the directed networks by replacing the original adjacency matrix A by an asymmetric one. Also, this paper concerns the prediction problem in static networks. In reality, many networks are continuously evolving, and the links created in different times should be assigned with different weights. Our algorithm could deal with weighted networks by replacing A by a weighted matrix.

Finally, notice that in our running example, we have assumed the existence of a well-connected component. However, in real social networks a graph is not fully connected. This means that we can calculate a similarity between nodes, which belong to the same connected component, whereas we compute zero values for pairs of nodes belonging to different components. To overcome this limitation, we could connect the different components with an artificial link.

7 Conclusions

In this paper, we introduced a framework that uses multi-way spectral clustering to provide friend recommendations in OSNs. We compared our method with well-known link prediction algorithms, the k-means and the two-way nCut clustering algorithms, using two synthetic and three real data sets (Hi5, Facebook and Epinions). We have shown that our SpectralLink algorithm provides more accurate and fast friend recommendations. In future, we indent to improve friend recommendations by combining explicit with implicit social networks. Implicit social networks provide valuable information by similar users' who co-comment on written posts, co-rate products and co-participate in groups.

References

1. Z. Abbassi and V. S. Mirrokni. A recommender system based on local random walks and spectral methods. In *Workshop on Knowledge Discovery on the Web (WebKDD'2007) in conjuction with the 1st International Workshop on Social Networks Analysis (SNA-KDD 2007)*, pages 139–153, 2007.
2. L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.
3. V. Agarwal and K. Bharadwaj. A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity. *Social Network Analysis and Mining (to appear)*, pages 1–21, 2013.
4. M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.
5. B. Caci, M. Cardaci, and M. Tabacchi. Facebook as a small world: a topological hypothesis. *Social Network Analysis and Mining*, pages 1–5, 2011.
6. J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 201–210, 2009.
7. A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.
8. L. Costa, F. Rodrigues, G. Travieso, and P. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.
9. D. Davis, R. Lichtenwalter, and N. Chawla. Supervised methods for multi-relational link prediction. *Social Network Analysis and Mining (to appear)*, 2013.
10. M. Fazel-Zarandi, H. Devlin, Y. Huang, and N. Contractor. Expert recommendation based on social drivers, social network analysis, and semantic data representation. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 41–48. ACM, 2011.
11. F. Fouss, K. Francoisse, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of graph kernels on collaborative recommendation and semisupervised classification. *Technical Report*, 2009.
12. F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.*, 19(3):355–369, 2007.
13. G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins Studies in Mathematical Sciences, 1983.
14. I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 77–86, 2009.
15. P. Hage and F. Harary. Structural models in anthropology. 56, 1983.
16. H. Higham, G. Kalna, and M. Kibble. Spectral clustering and its use in bioinformatics. *J. Comput. Appl. Math.*, 204(1), 2007.
17. Y. Hou. Bounds for the least laplacian eigenvalue of a signed graph. *Acta Mathematica Sinica*, 21:955–960, 2005.

18. N. Iakovidou, P. Symeonidis, and Y. Manolopoulos. Multiway spectral clustering link prediction in protein-protein interaction networks. In *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on*, pages 1–4. IEEE, 2010.
19. G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002.
20. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.
21. J. Kunegis and A. Lommatzsch. Learning spectral graph transformations for link prediction. In *Proc. Int. Conf. in Machine Learning*, 2009.
22. J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings 19th international conference on World Wide Web (WWW'2010)*, pages 641–650, Raleigh, NC, 2010.
23. J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1361–1370. ACM, 2010.
24. D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM)*, 2003.
25. L. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(7):1019–1031, 2007.
26. S. Lo and C. Lin. Wmr: a graph-based algorithm for friend recommendation. In *Proceedings of the IEEE/ACM International Conference on Web Intelligence (WIC)*, pages 121–128, Hong Kong, China, 2006.
27. L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.
28. M. Maila and J. Shi. A random walks view of spectral segmentation. In *International Conference on AI and Statistics (AISTAT)*, 2001.
29. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.
30. J. Pan, H. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *KDD '04: Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 653–658, 2004.
31. A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos. Friendlink: Link prediction in social networks via bounded local path traversal. In *Computational Aspects of Social Networks (CASoN), 2011 International Conference on*, pages 66–71. IEEE, 2011.
32. M. Rattigan and D. Jensen. The case for anomalous link discovery. *SIGKDD Explorations*, 7(2):41–47, 2005.
33. J. Shi and J. Malik. Normalized cuts and image segmentation. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 731, 1997.
34. P. Symeonidis, E. Tiakas, and Y. Manolopoulos. Transitive node similarity for link prediction in social networks with positive and negative links. In *Proceedings of the 4th International Conference on Recommender Systems (RecSys'2010)*, pages 183–190, Barcelon, Spain, 2010.
35. H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM '06: Proceedings of the 6th International Conference on Data Mining*, pages 613–622. IEEE Computer Society, 2006.
36. C. Tsourakakis, P. Drineas, E. Michelakis, I. Koutis, and C. Faloutsos. Spectral counting of triangles via element-wise sparsification and triangle-based link recommendation. *Social Network Analysis and Mining*, 1(2):75–81, 2011.
37. D. Yan, L. Huang, and M. Jordan. Fast approximate spectral clustering. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 907–916, 2009.