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

Transitive Node Similarity: Predicting and Recommending Links in Signed Social Networks*

Panagiotis Symeonidis · Eleftherios Tiakas

the date of receipt and acceptance should be inserted later

Abstract Online social networks (OSNs) like Facebook, Myspace, and Hi5 have become popular, because they allow users to easily share content. 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). However, OSNs do not exploit the whole structure 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 approaches, which detect the overall path structure in a network, being computationally prohibitive for huge-size social networks. In this paper, we define a basic node similarity measure that captures effectively local graph features (i.e., by measuring proximity between nodes). We exploit global graph features (i.e., by weighting paths that connect two nodes) introducing transitive node similarity. We also derive variants of our method that apply to different types of networks (directed/undirected and signed/unsigned). We perform extensive experimental comparison of the proposed method against existing recommendation algorithms using synthetic and real data sets (Facebook, Hi5 and Epinions). Our experimental results show that our FriendTNS algorithm outperforms other approaches in terms of accuracy and it is also time efficient. Finally, we show that a significant accuracy improvement can be gained by using information about both positive and negative edges.

Keywords Social Networks, Link Prediction

* A preliminary version of this paper entitled "Transitive Node Similarity for Link Prediction in Social Networks with Positive and Negative Links" has been presented at the 4th ACM Conference on Recommender Systems (RECSYS 2010).

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

E. Tiakas
Department of Informatics, Aristotle University, Thessaloniki, 54124, Greece.
E-mail: tiakas@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 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. Thus, friend recommendation services allow users to get to know one's friends of friends and, hence, expand their own social circle.

In OSNs, two people can mutually agree to be listed as friends, to share information items such as photos, news, etc. Friendship links on OSNs are initiated by any of two people. For example, person A might find person B on the OSN and request to add her as a friend. Person B then receives an email, and can either accept the undirected friendship, or choose to reject it.

Individuals can also link themselves to others, using various other ways. For example, anyone can create a group and invite others to join. Two persons may belong in the same group or appear in pictures, which are tagged with their names.

In this paper, 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. The first approach is based on local features of a network, focusing mainly on the network structure; the second approach is based on global features, detecting the overall path structures in a network.

1.1 Motivation

Facebook.com and Hi5.com, as shown in Figure 1, have adopted a local method for recommending new friends to a target user v_8: "People you may know : (i) users v_2, v_3, v_5, v_6 because you have one common friend (user v_1) (ii) user v_4 because you have one common friend (user v_9) ...". Therefore, they provide friend recommendations consider only pathways of maximum length 2. The list of recommended friends is ranked based on the number of common friends each candidate friend has with the target user. But in the aforementioned example, the list of recommended friends cannot be ranked, because the number of common friends is the same for all recommended friends. Thus, user v_8 gets as friend recommendation user v_2 or v_3 or v_4 or v_5 or v_6 with equal probability.

However, if we take into account the "strong" connection between v_8 and v_9 (due to the fact that v_9 does not share many edges with others) then v_4 should have a higher probability to be recommended as a friend to v_8. In contrast, other candidate friends (e.g. v_2, v_3, v_5, v_6) should have a lower probability to be recommended as friends to v_8 because of the "loose" connection between v_8 and v_1 (due to the fact that v_1 shares many edges with other nodes).

Compared to existing approaches, our method takes into account the local and the global features of a network. In particular, we define a basic local similarity

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

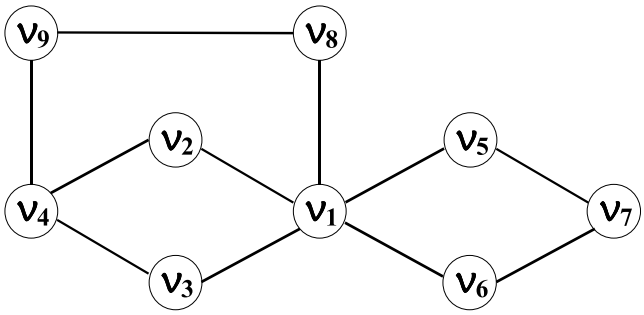


Fig. 1 Example of a Social Network.

measure that captures effectively the proximity between graph nodes. We also exploit global graph features by introducing transitive node similarity. Thus, two persons that are connected with a path have a high probability to know each other, depending on: (i) the length of the path they are connected with, and (ii) the degree of similarity between the neighbor nodes that form that pathway.

Moreover, many recent existing approaches propose complex link prediction models using a combination of measures to express the similarity/distance between nodes. They can give much better accuracy in link prediction than the simple measure approaches. However, their increased complexity deteriorates the performance of the recommendation engine, as they require more time and complex calculations. Therefore, they are not practical in large real networks where recommendations must be made on-the-fly and very quick. The proposed approach can provide similar accuracy in link prediction with simpler calculations and lower complexities. Therefore, our method can be used in real-time recommendation applications.

Finally, in contrast to the bulk of research on social networks that has focused almost exclusively on positive interpretations of links between people, we also study how the interplay between positive and negative relationships affects the structure of on-line social networks. We connect our analysis to theories of signed networks, such as the Structural Balanced theory [15], and the Status theory [18, 19]. More details about these theories can be found later in Section 3.6.

1.2 Contribution

The contributions of our approach are summarized as follows:

- A generalized framework for providing friend recommendations in OSNs is provided.
- We provide accurate friend recommendations, exploiting node similarity between the pairs of connected nodes in an OSN.
- We define a transitive node similarity measure in OSNs by taking into account the shortest paths between persons in an OSN.
- We provide an algorithm that exploits the similarity between a selected starting node to other nodes of the network by progressively discovering shortest

paths and calculating the corresponding transitive similarities of the discovered nodes on-the-fly. When a specific defined accuracy has been reached to the similarity calculations during a shortest path traversal, the discovery procedure stops in that path. In this way the algorithm discovers and expands only the required parts of the network in order to compute the top similarities for the selected user with the specified accuracy. Therefore, it keeps the time and space complexity low even in large networks. These advantages can be useful for cases where long term digital preservation of the social network connections is needed, because as data in large networks (with billions of nodes) are aggregated over time, our method is scalable to efficiently process them.
- We perform extensive experimental comparison of the proposed method against existing recommendation algorithms using Facebook and Hi5 data sets.
- Our method substantially improves accuracy of friend recommendations with respect to previous methods, as will be shown experimentally.
- 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.

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 briefly reviews the proposed method and preliminaries in graph theory employed in our approach. Section 3.2 defines a node similarity measure in OSNs. A motivating example, the proposed algorithm and a derivation of variants of our method that apply to different types of networks (directed/undirected and signed/unsigned) are described in Section 3.4. Experimental results are given in Section 4. A discussion is presented in Section 5. Finally, Section 6 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 "six degrees of separation" [25]. That is, for a randomly chosen pair of individuals, there exists with high probability a short chain of intermediaries that connect them, where "short" is usually interpreted to the logarithm of the population size. The "algorithmic small-world hypothesis" states also that ordinary individuals can effectively "navigate" these short chains themselves [1,17,20,29,32]. That is, individuals who attempt to locate new friends in an OSN can effectively traverse chains of referrals. Moreover, according to homophily theory [3] (i.e., "love of the same") individuals tend to prefer the same things that similar other users do like. Goel et al. [12] 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 area of link prediction in social networks, tries to infer which new interactions among members of a social network are likely to occur in the near future. There are two main approaches [20] that handle the *link prediction* problem. 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.

– There is a variety of local similarity measures [20] (i.e., Adamic/Adar index, Jaccard Coefficient, Common Neighbors index, etc.) for analyzing the "proximity" of nodes in a network. Among these indices, Adamic/Adar [1] index is reported [20] to attain the best performance in predicting new links in a social network. Adamic/Adar index, which is similar to Jaccard Coefficient (a commonly used similarity metric in information retrieval), measures how strongly "related" two web pages are. 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 and v_y are more likely to form a link in the future, if they have many common neighbors. Furthermore, other local similarity measures are based on preferential attachment [20]. The basic premise of preferential attachment is that the probability that a new edge involves a node is proportional to the current number of its neighbors (i.e., the big gets bigger).
– There is a variety of global approaches [20] (i.e., Shortest Path algorithm, RWR algorithm, SimRank algorithm etc.). Liben and Kleinberg [20] claimed that the identification of the shortest path between any pair of nodes in a graph can be used for link prediction (friend recommendation). The computation of the shortest path between two nodes, can be made using any well-known shortest path algorithm [8, 11]. RWR algorithm [27] (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 who chooses randomly among the available edges every time, except that, before he makes 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. In the same direction with RWR, Fouss et al. [10] proposed a Random walk model that computes quantities (the average commute time, the pseudoinverse of the Laplacian Matrix of a graph, etc.) to capture similarities between any pair of nodes in a network. These quantities have the property of increasing, when the number of paths connecting two nodes increases and when the length of paths that connects them decreases. SimRank [16, 33] also computes a global similarity measure based on the structural context of a network that says "two objects are similar if they are related to similar objects". Recently, Clauset et al. [7] proposed an algorithm based on the hierarchical network structure. First, they use a hierarchical random graph to statistically fit the real network data. Then, the dependence of the connection probability on the depth of the nodes in the hierarchy can be inferred. One can predict the missing links of the network according to the connection probability by ranking them in a descending order. Finally, Blondel et al. [5] considered path-based similarity measures between nodes of different directed networks [26], i.e., based on asymmetric adjacency matrices, which is a more complex situation than the one we consider.

The novelty of our approach compared to existing approaches is as follows:

– In contrast to global algorithms, such as the Random Walk with Restart (RWR) algorithm [27], the Shortest Path [8, 11] algorithm etc., our method also

takes into account local graph features (i.e., the weighted similarity between nodes that may share many edges with others). We selected RWR and Shortest Path algorithms as representatives of the global algorithms and compared them with our method. As will be shown experimentally later, our method outperforms RWR and Shortest Path. The reason is, they traverse globally the social network, missing to capture adequately the local graph characteristics.
- In contrast to local similarity measures, such as FOAF [6] algorithm (also known as the Common Neighbors index [21]), the Adamic/Adar [1] index etc., we take into account also global graph features (i.e., paths connecting any pair of nodes in an OSN). We have compared our method against FOAF algorithm and Adamic/Adar index, as representatives of the local-based measures. As will be shown experimentally later, our method outperforms FOAF and Adamic/Adar index. The reason is, we do not take into account only pathways of length 2 to compute similarity between a pair of nodes in an OSN. Instead, we use an extensive similarity measure that takes into account transitive node similarity.

Finally, besides the aforementioned link prediction algorithms that are based solely on graph structure, there are also other methods that exploit other data sources such as messages among users, user ratings, co-authored papers, 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 [22] 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. Cha et al. collected and analyzed large-scale traces of information dissemination in the Flickr social network. They experimentally derived that over 50% of users find their favorite pictures (i.e., pictures they bookmark) from their friends in an OSN. TidalTrust [13] and MoleTrust [24] are also hybrid approaches that combine the rating data of collaborative filtering systems with the link data of trust-based social networks (i.e., Epinions.com) in order to improve the friend recommendation accuracy. In contrast to the above methods, we focus only on recommendations based on the link structure of an OSN and thus, we will exclude them from our experimental comparison.

3 The proposed method

In this section we present the main strategy of the proposed methodology: Firstly, a specific starting node (user) is selected for which we will make recommendations. Then, we compute the similarities from the selected node to other nodes (users) of the network based on specific predefined similarity measures, which we present in the following subsections. The similarities of neighboring nodes are calculated instantly, and all other required similarities are calculated through a progressive process, which discovers shortest paths and updates the similarities transitively on-the-fly. When a specific predefined threshold value is reached, through a path,

then the discovery procedure stops in that path. Finally, the calculated similarities are sorted descending and the nodes (users) having the top-k similarity values are recommended to the selected starting node (user).

3.1 Preliminaries in Graphs

In this subsection, we present the most important notations and the corresponding definitions used throughout the rest of the paper. Notice that, we use: (i) calligraphic notation for sets, (ii) lowercase notation for integer numbers, (iii) uppercase notation for matrices, and (iv) bold lowercase notation for vectors, as shown in Table 1.

Symbol	**Description**
$\mathcal{G}$	undirected and unweighted graph
v_i	node of a graph $\mathcal{G}$
e_i	edge of a graph $\mathcal{G}$
$\mathcal{V}$	set of graph nodes
$\mathcal{E}$	set of graph edges
$n = \|\mathcal{V}\|$	number of nodes in graph $\mathcal{G}$
$m = \|\mathcal{E}\|$	number of edges in graph $\mathcal{G}$
A	adjacency matrix of graph $\mathcal{G}$
R	incidence matrix of graph $\mathcal{G}$
S	basic similarity matrix of graph $\mathcal{G}$
ES	extended similarity matrix of graph $\mathcal{G}$
$\mathbf{r_i}$	node vector of node v_i in R
$sim(v_i, v_j)$	basic similarity between v_i and v_j
$esim(v_i, v_j)$	extended similarity between v_i and v_j
$deg(v_i)$	degree of node v_i

Table 1 Frequently used notations.

Let $\mathcal{G}$ be a graph with a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Every edge is defined by a specific pair of graph nodes (v_i, v_j), where $v_i, v_j \in \mathcal{V}$. First, we assume that the graph $\mathcal{G}$ is undirected and un-weighted, thus the graph edges do not have any weights, plus the order of nodes in an edge is not important. Therefore, (v_i, v_j) and (v_j, v_i) denote the same edge on $\mathcal{G}$. Later we will extend our study without this assumption in directed and in weighted/signed graphs. We also assume that the graph $\mathcal{G}$ does not have multiple edges, thus if two nodes v_i, v_j are connected with an edge of $\mathcal{E}$, then there is no other edge in $\mathcal{E}$ also connecting them. Finally, we assume that there are no loop edges on $\mathcal{G}$ (i.e., a node can not be connected to itself). The graph expressing friendships among users of an OSN, which can be seen in Figure 1, will be used as our running example throughout the rest of the paper. For illustrating the calculations in the running example we will use well-known representations, such as the adjacency matrix $A_{n\times n}$, and the incidence matrix $R_{m\times n}$. The adjacency matrix of our running example is depicted in Figure 2. The incidence matrix of our running example is depicted in Figure 3.

To later define our proposed similarity measure between any pair of nodes in a graph $\mathcal{G}$, we use a node vector space model based on the incidence matrix R of $\mathcal{G}$. Using this model, each node can be encoded as a binary vector where each

$$A = \begin{array}{c} \begin{matrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} & \mathbf{v_5} & \mathbf{v_6} & \mathbf{v_7} & \mathbf{v_8} & \mathbf{v_9} \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \mathbf{v_3} \\ \mathbf{v_4} \\ \mathbf{v_5} \\ \mathbf{v_6} \\ \mathbf{v_7} \\ \mathbf{v_8} \\ \mathbf{v_9} \end{matrix}$$

Fig. 2 Adjacency Matrix of the Social Network.

$$R = \begin{array}{c} \begin{matrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} & \mathbf{v_5} & \mathbf{v_6} & \mathbf{v_7} & \mathbf{v_8} & \mathbf{v_9} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} \mathbf{e_1{=}(v_1{,}v_2)} \\ \mathbf{e_2{=}(v_1{,}v_3)} \\ \mathbf{e_3{=}(v_1{,}v_5)} \\ \mathbf{e_4{=}(v_1{,}v_6)} \\ \mathbf{e_5{=}(v_1{,}v_8)} \\ \mathbf{e_6{=}(v_2{,}v_4)} \\ \mathbf{e_7{=}(v_3{,}v_4)} \\ \mathbf{e_8{=}(v_4{,}v_9)} \\ \mathbf{e_9{=}(v_5{,}v_7)} \\ \mathbf{e_{10}{=}(v_6{,}v_7)} \\ \mathbf{e_{11}{=}(v_8{,}v_9)} \end{matrix}$$

Fig. 3 Incidence Matrix of the Social Network.

component reflects the appearance of a particular edge in the node. Thus, any node v_i of $\mathcal{G}$ is represented by a binary vector derived from the corresponding column of v_i in R. We will call this vector representation of a node v_i in graph $\mathcal{G}$ as the *node vector* $\mathbf{r_i}$.

3.2 The Basic Node Similarity Measure

In this Section, we define a basic node similarity measure to determine the proximity between any pair of neighbor nodes in a graph $\mathcal{G}$. Based on our basic node similarity measure, the probability that a new edge involves a node is inversely proportional to the number of its neighbors. This contradicts with preferential at-

tachment [20], which claims that the probability that a new edge involves a node is proportional to its current number of neighbors. The intuition behind it is that, if two adjacent nodes have a lot of other adjacent nodes, the two nodes are less likely to become friends. In the same direction, if the shortest path between two nodes goes through a lot of high degree nodes, the two nodes are less likely to become friends. As we will later experimentally show (see Section 4.7), our similarity measure outperforms the preferential attachment measure.

Therefore, if v_i and v_j are two neighbor connected nodes of $\mathcal{G}$, we define a specific function $sim(v_i, v_j)$ that expresses their corresponding similarity in the range [0,1] and has all the required properties (i.e., positivity, reflexivity, symmetry etc.) of a well-defined measure. The more similar the nodes are, the more the value of $sim(v_i, v_j)$ will be close to 1. On the contrary, the more dissimilar the nodes are, the more the value of $sim(v_i, v_j)$ will be close to 0.

To capture proximity between node vectors, we apply the Jaccard Coefficient, which is able to measure the degree of overlap between node vectors, in contrast to other measures (i.e., dot product, Euclidean distance etc.), which cannot measure it. In particular, we use an extension of the Jaccard Coefficient that contains the cosine similarity metric as we have binary vectors. This extension is also called the Tanimoto coefficient [30], and for two binary vectors $\mathbf{a}, \mathbf{b}$ is defined as:

$$sim(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||^{\mathbf{2}} + ||\mathbf{b}||^{\mathbf{2}} - \mathbf{a} \cdot \mathbf{b}}$$

By applying the above measure in our node vector space model, we must replace the vectors $\mathbf{a}, \mathbf{b}$ with the node vectors $\mathbf{r_i}, \mathbf{r_j}$ of matrix R, and then the following equation is derived:

$$sim(v_i, v_j) = \frac{\mathbf{r_i} \cdot \mathbf{r_j}}{||\mathbf{r_i}||^2 + ||\mathbf{r_j}||^2 - \mathbf{r_i} \cdot \mathbf{r_j}}$$

Finally, by substitution of vector operations between $\mathbf{r_i}, \mathbf{r_j}$ in the previous equation with the corresponding values of the incidence matrix R, we derive the following equivalent equation:

$$sim(v_i, v_j) = \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}{\sum_{h=1}^{m} R[e_h, v_i]^2 + \sum_{h=1}^{m} R[e_h, v_j]^2 - \sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]} \quad (1)$$

The squares of R can be dropped as R takes only Boolean values, thus we have:

$$sim(v_i, v_j) = \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}{\sum_{h=1}^{m} R[e_h, v_i] + \sum_{h=1}^{m} R[e_h, v_j] - \sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}$$

Note also that the term $\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]$ in the final derived equation, expresses the number of edges that the nodes v_i, v_j share, whereas the terms $\sum_{h=1}^{m} R[e_h, v_i]$, $\sum_{h=1}^{m} R[e_h, v_j]$ are equal to the degrees of nodes v_i, v_j respectively.

The basic node similarity measure satisfies the positivity property, returning values into the interval [0,1]. Note that the maximum value of similarity (equal to 1) can be reached, when the two nodes are connected with only one edge and have no connections with other nodes, and the minimum value (equal to 0) can be reached when the two nodes do not share any edge. Moreover, Equation 1 can be simplified by using Theorem 1.

Theorem 1 *If the basic node similarity measure of Equation 1 is applied in a graph $\mathcal{G}$ satisfying all mentioned assumptions of Section 3.1 (i.e., $\mathcal{G}$ is an undirected and unweighted graph, which does not have multiple or loop edges), then it can be simplified to the following:*

$$sim(v_i, v_j) = \begin{cases} 1, & \text{if } v_i = v_j \\ 0, & \text{if } v_i \neq v_j \wedge (v_i, v_j) \notin \mathcal{E} \wedge (v_j, v_i) \notin \mathcal{E} \\ \frac{1}{deg(v_i)+deg(v_j)-1}, & \text{otherwise} \end{cases}$$

where $deg(v_i)$ and $deg(v_j)$ are the degrees of nodes v_i and v_j, respectively.

Proof If $v_i = v_j$, then due to the reflexivity property (proved in Section 7 (Appendix)) we have: $sim(v_i, v_j) = 1$.

The fact that $(v_i, v_j) \notin \mathcal{E}$ and $(v_j, v_i) \notin \mathcal{E}$ means that nodes v_i, v_j do not share any edges. Thus the term $\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]$ in Equation 1 is equal to 0 and $sim(v_i, v_j) = 0$.

If nodes v_i, v_j share one edge, then they can not share any other edge as explained in Section 3.1. Thus, the term $\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]$ in Equation 1 is equal to 1. Moreover, the terms $\sum_{h=1}^{m} R[e_h, v_i], \sum_{h=1}^{m} R[e_h, v_j]$ are equal to the degrees of nodes v_i, v_j, respectively. In that case we have:

$$sim(v_i, v_j) = \frac{1}{deg(v_i) + deg(v_j) - 1},$$

and the theorem has been proven. □

Henceforth, Theorem 1 will be used in defining our basic similarity measure, which is based on the inverse sum of node degrees. However, someone could suggest the usage of any other local-based similarity measure [20] as described in Section 2. For this reason, our basic measure will be later experimentally compared with other measures, which are also based on the nodes degree and the preferential attachment process [20]: the sum of nodes degree and the product of nodes degree.

Now, let us calculate some similarity values on the graph of Figure 1 using Equation 1. The similarity between nodes v_1 and v_2 is: $sim(v_1, v_2)=\frac{1}{5+2-1}=\frac{1}{6}=0.16$. The similarity between nodes v_2 and v_4 is: $sim(v_2, v_4)=\frac{1}{2+3-1}=\frac{1}{4}=0.25$. Thus, the similarity score between nodes v_1, v_2 is less than that of v_2, v_4 because the degree

of node v_1 is greater than that of v_4, whereas v_1 shares only one of its total 5 edges.

Collecting all similarity values between the nodes of a graph $\mathcal{G}$, we construct the *basic node similarity matrix* S of $\mathcal{G}$, which is an $n \times n$ matrix having n rows and n columns labeled by the graph nodes. The basic node similarity matrix values are defined as follows:

$$S[v_i, v_j] \;=\; sim(v_i, v_j)$$

In our running example, the basic node similarity matrix is depicted in Figure 4, where all values are rounded to the third decimal digit. As shown, user v_9 is more similar with user v_8 than user v_4. This is reasonable, because user v_4 is connected with 2 other nodes (v_2 and v_3), while user v_8 is connected with only 1 other node (v_1).

$$S = \begin{array}{c} \begin{matrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} & \mathbf{v_5} & \mathbf{v_6} & \mathbf{v_7} & \mathbf{v_8} & \mathbf{v_9} \end{matrix} \\ \begin{bmatrix} 1 & 0.167 & 0.167 & 0 & 0.167 & 0.167 & 0 & 0.167 & 0 \\ 0.167 & 1 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 \\ 0.167 & 0 & 1 & 0.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0.25 & 1 & 0 & 0 & 0 & 0 & 0.25 \\ 0.167 & 0 & 0 & 0 & 1 & 0 & 0.333 & 0 & 0 \\ 0.167 & 0 & 0 & 0 & 0 & 1 & 0.333 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.333 & 0.333 & 1 & 0 & 0 \\ 0.167 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.333 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.333 & 1 \end{bmatrix} \end{array} \begin{matrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \mathbf{v_3} \\ \mathbf{v_4} \\ \mathbf{v_5} \\ \mathbf{v_6} \\ \mathbf{v_7} \\ \mathbf{v_8} \\ \mathbf{v_9} \end{matrix}$$

Fig. 4 Basic Node Similarity Matrix.

The basic node similarity measure, defined in Equation 1, satisfies fundamental properties, which are proved in Section 7 (Appendix) such as: positivity, reflexivity, and symmetry. Therefore, the basic similarity measure is well-defined for further use in applications such as node clustering, graph visualization etc. As expected, the basic node similarity matrix S has the following properties:

- Due to the positivity property, all values of S are positive numbers in the interval [0,1]. The maximum value of similarity (equal to 1) arises, when two nodes are connected with only one edge and have no connections with other nodes, whereas the minimum value (equal to 0) arises when two nodes do not share any edge.
- Due to the reflexivity property, all values of the main diagonal in S are equal to 1.
- Due to the symmetry property, S is a symmetric square matrix.

3.3 Extending the Node Similarity Measure

Based on Theorem 1, the similarity values between all non-neighbor nodes in a graph $\mathcal{G}$ are zero. For instance, in our running example, the similarity value between nodes v_1 and v_4 is zero, because they do not share any edge. However, users v_1

and v_4 have both user v_2 as a common friend, and thus they could be related in some way.

By using a transitive similarity we can efficiently solve this problem. In our method, we define a transitive node similarity, between two nodes v_i and v_j, denoted as extended similarity. Extended similarity is calculated by the product of the basic similarities between the nodes of the shortest path from v_i to v_j. This ensures that extended similarities will become smaller than all the basic similarities, which define it, as the following theorem holds.

Theorem 2 *Let $p = \{v_{p_1}, v_{p_2}, ..., v_{p_k}\}$ be any path from node $v = v_{p_1}$ to $u = v_{p_k}$. Then,*

$$esim(v,u) \leq sim(v_{p_i}, v_{p_{i+1}}), \forall i \in \{1,2,\ldots,k-1\}$$

Proof All basic similarities are numbers into $[0,1]$. Thus, let $sim(v_{p_i}, v_{p_{i+1}}) = \frac{1}{a_i}, \forall i \in \{1,2,\ldots,k-1\}$ where $a_i \geq 1$. Then,

$$esim(v,u) = \prod_{i=1}^{k-1} sim(v_{p_i}, v_{p_{i+1}}) = \prod_{i=1}^{k-1} \frac{1}{a_i}$$

Therefore, it is sufficient to prove $\forall i \in \{1,2,\ldots,k-1\}$ that:

$$\prod_{j=1}^{k-1} \frac{1}{a_j} \leq \frac{1}{a_i}$$

or equivalently:

$$\frac{1}{a_1 \cdot a_2 \cdot ... \cdot a_{k-1}} \leq \frac{1}{a_i}$$
$$\Leftrightarrow a_1 \cdot a_2 \cdot ... \cdot a_{k-1} \geq a_i$$
$$\Leftrightarrow a_1 \cdot a_2 \cdot ... \cdot a_{i-1} \cdot a_{i+1} \cdot ... \cdot a_{k-1} \geq 1$$

which always holds, and the theorem has been proven. □

However, for the calculation of the extended similarity someone could suggest to calculate the products of all possible paths between any pair of nodes and then to select the maximum extended similarity value. This choice has a prohibitive computational cost. Thus, in our method, we calculate the product of the shortest path, by imposing a penalty to the long distance nodes.

For the above reasons, we choose the *shortest path* among all possible paths between the two nodes. This shortest path expresses the minimum number of edges required to connect the two nodes, as all edges of graph $\mathcal{G}$ are not weighted. Therefore, we define the following extended node similarity measure for any two nodes of $\mathcal{G}$:

$$esim(v_i, v_j) = \begin{cases} 0, & \text{if there is no path between } v_i, v_j \\ sim(v_i, v_j), & \text{if } v_i, v_j \text{ are neighbors} \\ \prod_{h=1}^{k} sim(v_{p_h}, v_{p_{h+1}}), & \text{otherwise} \end{cases} \quad (2)$$

where v_{p_1}=v_i, $v_{p_{k+1}}$=v_j and the nodes v_{p_h} (for h=2,...,k) are all the intermediate nodes that the shortest path from v_i to v_j passes through. Note that, in case

that v_i, v_j are neighbor nodes, the shortest path between them is the single edge connecting them, and this explains why $esim(v_i, v_j) = sim(v_i, v_j)$.

It is also important to note that if there are several paths from v_i to v_j having the same length with the shortest (i.e., all these paths can be alternative shortest paths), we select the path that maximizes the product of the basic similarity values in order to define the corresponding extended similarity score.

In our running example, according to the previous definition, the extended similarity between nodes v_1 and v_4 using Equation 2 equals:

$$sim(v_1, v_4) = sim(v_1, v_2) \cdot sim(v_2, v_4) = \frac{1}{6} \cdot \frac{1}{4} = 0.042$$

as the shortest path between v_1, v_4 is: $v_1 \rightarrow v_2 \rightarrow v_4$ (the alternative path $v_1 \rightarrow v_3 \rightarrow v_4$ has the same length and the same similarity score since nodes v_3 and v_2 have equal degrees). Note that the extended similarity score between nodes v_1, v_4 is less than the basic similarity score of v_1, v_2 (0.167) and v_2, v_4 (0.25).

Collecting all the extended similarity values between the nodes of a graph $\mathcal{G}$, we construct the *extended node similarity matrix* ES of $\mathcal{G}$. It is a matrix which has the same dimensionality and structure with the basic node similarity matrix S. Its values are defined as follows:

$$ES[v_i, v_j] \; = \; esim(v_i, v_j)$$

In our running example, the extended node similarity matrix is depicted in Figure 5, where all values are rounded to the third decimal digit.

$$ES = \begin{array}{c} \begin{matrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} & \mathbf{v_5} & \mathbf{v_6} & \mathbf{v_7} & \mathbf{v_8} & \mathbf{v_9} \end{matrix} \\ \begin{bmatrix} 1 & 0.167 & 0.167 & 0.042 & 0.167 & 0.167 & 0.056 & 0.167 & 0.056 \\ 0.167 & 1 & 0.028 & 0.25 & 0.028 & 0.028 & 0.009 & 0.028 & 0.062 \\ 0.167 & 0.028 & 1 & 0.25 & 0.028 & 0.028 & 0.009 & 0.028 & 0.062 \\ 0.042 & 0.25 & 0.25 & 1 & 0.007 & 0.007 & 0.002 & 0.083 & 0.25 \\ 0.167 & 0.028 & 0.028 & 0.007 & 1 & 0.028 & 0.333 & 0.028 & 0.009 \\ 0.167 & 0.028 & 0.028 & 0.007 & 0.028 & 1 & 0.333 & 0.028 & 0.009 \\ 0.056 & 0.009 & 0.009 & 0.002 & 0.333 & 0.333 & 1 & 0.009 & 0.003 \\ 0.167 & 0.028 & 0.028 & 0.083 & 0.028 & 0.028 & 0.009 & 1 & 0.333 \\ 0.056 & 0.062 & 0.062 & 0.25 & 0.009 & 0.009 & 0.003 & 0.333 & 1 \end{bmatrix} \begin{matrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \mathbf{v_3} \\ \mathbf{v_4} \\ \mathbf{v_5} \\ \mathbf{v_6} \\ \mathbf{v_7} \\ \mathbf{v_8} \\ \mathbf{v_9} \end{matrix} \end{array}$$

Fig. 5 Extended Node Similarity Matrix.

It is important to note that using the extended node similarity in a connected graph, such as the graph $\mathcal{G}$ of our running example, all values of ES will be positive numbers (non-zero values). This is due to the fact that there is always a shortest path between any pair of node of a connected graph. However, if a graph is not connected, then for nodes belonging to the same connected component, the extended node similarity scores will be still non-zero, whereas zero values will exist for pairs of nodes belonging to different components.

The extended node similarity measure satisfies all fundamental properties, which are proved in Section 7 (Appendix), and advantages that the basic node similarity measure has: positivity, reflexivity, symmetry. All these properties, affect also the similarity values of the extended node similarity matrix ES:

- Due to the positivity property, all values of ES are positive numbers in the interval [0,1]. Moreover, the zero values appear only if we do not have connected nodes (there is no path between the two nodes).
- Due to the reflexivity property, all values of the main diagonal in ES are equal to 1.
- Due to the symmetry property, ES is a symmetric square matrix.

3.4 The FriendTNS Algorithm

In this section, we present the proposed algorithm, denoted as *FriendTNS* (Friend Transitive Node Similarity), we analyze its steps, provide implementation details and discuss its time and space complexity.

The main strategy of the *FriendTNS* algorithm is simple: to compute the similarities from a specific starting node (the selected user) v_0 to other nodes (users) of the network by progressively discovering shortest paths and calculating the corresponding transitive similarities $s[]$ of the discovered nodes on-the-fly, following Theorem 1 and Equation 2. When a specific defined accuracy (according to a precision parameter p) has been reached to the remaining similarity calculations, during a shortest path traversal, the discovery procedure stops in that path. In that way the algorithm discovers and expands only the required parts of the network in order to compute the top similarities for the selected user with the specified accuracy. Therefore, it keeps the time and the space complexity low even in large networks. Moreover, the precision p works as a natural bound and keeps the calculations in a specific small part of the network around the user v_0, which is sufficient to calculate the top-k similarities. All these properties makes the algorithm able to work efficiently also in very large networks, in dynamic environments, and in environments that the network structure is not known in advance.

The algorithm input is the graph $\mathcal{G}$ of the network, the node v_0 which represents the target user that will take friend recommendations, the number k of friends that will be recommended to him, and p the desired precision. The output is the recommendations array $recom[1...k]$ which holds the corresponding IDs of the recommended users as friends of v_0. The outline of the *FriendTNS* algorithm is depicted in Figure 6.

In the first part of the algorithm (lines 1-3), FriendTNS initializes the variables and the arrays. The array $s[1...n]$ holds the similarity calculations for the target user v_0, while $dist[1...n]$ holds the distances for the shortest path calculations. H is a min-heap with the node distance as a prioritization key. The array $deg[1...n]$ holds the computed degrees of the discovered nodes, while $ind[1...n]$ holds an index of the corresponding node IDs during the sorting process of the node similarity values.

The second part of the algorithm (lines 4-14), is based on a well-known Dijkstra-like shortest path procedure, but modified for the required similarity calculations. Important modifications/additions are:

- The computation of the node degrees in line 6 is made only for the discovered nodes, which are required for the calculation of the basic similarity value $sim(v, u)$ that is used in line 9.
- When an update on the distance of a node u from v_0 occurs, then the transitive similarity of that node is also updated according to the newly discovered path

Algorithm **Friend-TNS-Algorithm(G,v_0,k,p)**
Input: graph G, user node v_0, recommendations k, precision p

```
01. s[1...n] = 0, deg[1...n] = 0, ind[1...n] = 0, recom[1...k] = 0
02. H = ∅, dist[1...n] = ∞
03. v = v_0, dist[v_0] = 0, s[v_0] = 1, insert v_0 into H
04. while H is not empty do
05.    for all adjacent nodes u of v (u neighbor of v)
06.       compute the degrees of nodes u, v: deg[u], deg[v] (if not known)
07.       if dist[u] > dist[v] + 1 then
08.          dist[u] = dist[v] + 1
09.          s[u] = s[v] * sim(v, u)
10.       end-if
11.       if u ∉ H ∧ s[u] ≥ p then insert u into H
12.    end-for
13.    v = get top item of H and remove it from H
14. end-while
15. sort the derived similarity list s[] with a descending order
16. and keep an index of the corresponding node IDs in ind[]
17. h = 0
18. for i = 1 to n
19.    v = ind[i]
20.    if node v is not a neighbor of v_0 then
21.       recom[h] = v, h++
22.       if h > k then exit for
23.    end-if
24. end-for
25. return recom[]
```

Fig. 6 Outline of FriendTNS Algorithm.

from the starting node v_0 to u on-the-fly (line 9). Note that in line 9 the similarity value $sim(v, u)$ is the basic similarity between the nodes v, u as they are adjacent, thus the formula of Theorem 1 is used for this calculation.

– The newly discovered node u is inserted into the heap H only when its corresponding discovered shortest path returns a transitive similarity value greater than the precision p. This means that if the transitive similarity value goes under p the expansion of the path stops. As already mentioned, this works as a natural bound and keeps the expansion in a specific small part of the network. It is important to note that due to the transitivity calculations and Theorem 2, every newly discovered edge in a path produces progressively smaller (or equal) similarity values, thus there will be always a stop-point in any path. In our experiments, after testing, we used $p = 0.000001$ which is enough for the similarity calculations in the selected networks. A further decrease of the value of p (increase of the precision) does not provide different recommendations. Therefore, any transitive similarities smaller than 0.000001 are not calculated (they remain 0), and there is no need of further expansion in the corresponding paths. However, the user can further calibrate (increase or decrease) the precision as desired. Please notice that in very dense networks, there are a lot of nodes with very high degrees. In such cases, the basic similarity values can become very small, and consequently the extended similarity values may exceed the standard double precision limits, even between nodes that have small

network distances. Therefore, even if we set the precision p near the limits of double numbers, some extended similarity values may become zero due to underflows. Thus, the final recommendations will be biased. In order to overcome this limitation, special structures to handle the calculations of such decimal numbers should be developed.

In the last part of the algorithm (lines 15-25), friends are recommended to v_0 according to their similarity values in $s[]$. Therefore, we sort the similarity list $s[]$ (line 15), we keep an index $ind[]$ for the corresponding node IDs (line 16), and we recommend the top-k nodes (users), which are not already friends of v_0 (lines 17-25).

In our running example, user v_8 would receive user v_4 as friend recommendation, because his similarity score (0.083) is greater than the similarity score of users v_2, v_3, v_5, v_6 (0.028). Note that the similarity values of the neighbor nodes of v_8 (and v_8 itself) are ignored as these are already friends of the target user v_8. The resulting recommendation is reasonable, due to the fact that user v_9 (which is responsible for recommending user v_4 to target user v_8) does not share many edges with others. In contrast, user v_1 (which is responsible for recommending users v_2, v_3, v_5, v_6 to target user v_8) shares many edges with others. Thus, our *FriendTNS* algorithm is able to capture the associations among the graph nodes.

3.5 Implementation Details and Complexity

FriendTNS keeps the graph nodes and edges in memory using an adjacency list representation (not an adjacency matrix), which requires an $O(n+m)$ space, where n is the total number of nodes and m is the total number of edges. All other arrays $(s, dist, deg, ind, recom)$ require $O(n)$ space. Therefore, our FriendTNS total space complexity is $O(n + m)$.

For the time complexity analysis, lines 1-3 and 17-25 have a computational complexity of $O(n)$. Moreover, for lines 15-16 we used the quick-sort algorithm which has a complexity of $O(n \log n)$. For the similarity calculations part (lines 4-14), we used a Fibonacci heap as in the shortest path algorithm of Fredman-Tarjan [11], which returns a complexity of $O(m+n \log n)$. Therefore, the total time complexity of FriendTNS is: $O(n) + O(n \log n) + O(m + n \log n) = O(m + n \log n)$.

3.6 Extending FriendLink for different types of Networks

Until this point, 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 FriendTNS that apply to directed networks and networks with weighted edges, including the case of edges with negative weights (signed networks).

Applying FriendTNS to directed graphs can be achieved (i) by simply disregarding the edge directions [31], or (ii) by inserting into the adjacency list representation only the directed edges. We followed the second case in our experimental evaluation for the directed networks.

Applying FriendTNS to weighted graphs can be achieved by holding the weights in the adjacency list representation, and by changing the distance calculations of

lines 7,8 of the FriendTNS algorithm (Figure 6), to: $dist[u] > dist[v] + w(u,v)$ and $dist[u] = dist[v] + w(u,v)$, respectively, where $w(u,v)$ denotes the weight of the edge (u,v) in the graph.

In signed and directed networks edges have positive (+1) as well as negative (-1) weights. Such signed directed graphs arise for instance in social networks (i.e., Epinions.com, Shashdot Zoo, etc.) where negative edges denote enmity or status instead of friendship. In such signed and directed graphs, FriendTNS's basic similarity measure of Theorem 1, which is the inverse of the sum of nodes' degree, can be adjusted accordingly based on the Status theory [18,19].

In this theory, a positive edge (v_i, v_j) means that v_i regards v_j as having a higher status than herself, while a negative edge (v_i,v_j) means that v_i regards v_j as having a lower status than herself. Assuming that all participants in the system agree on this status ordering, status theory predicts that when the direction of an edge is flipped, its sign should flip as well. As shown in Figure 7, to determine the sign of edge (v_i, v_j), we first flip the directions of the edges between v_k and v_i and between v_j and v_k, so that they point from v_i to v_k and from v_k to v_j. We also flip the signs accordingly as we do this. Then, we can define the sign of (v_i,v_j) to be the sign of the sum of the final signs of (v_i,v_k) and (v_k,v_j).

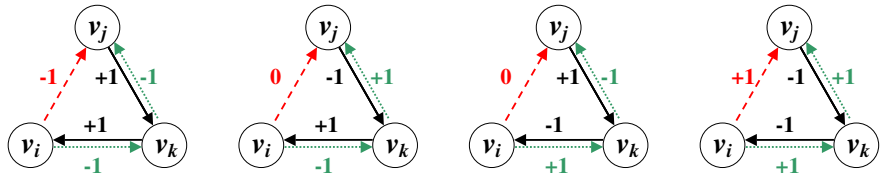


Fig. 7 The prediction of the sign of edge v_i,v_j (dashed line) based on the Status Theory.

Notice that status theory makes no prediction (zero value) when the two signs cancel out, which will be later handled by our extensive similarity measure.

Based on Status theory [18,19], 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. In the following, our basic similarity measure is transformed, so that it can take into account the aforementioned properties of Status Theory.

$$sim(v_i, v_j) = \frac{1}{\sigma(v_i) + \sigma(v_j) - 1},$$

where

$$\sigma(x) = deg^+_{in}(x) + deg^-_{out}(x) - deg^+_{out}(x) - deg^-_{in}(x).$$

As already stated, in networks with negative edge weights the concept of transitivity has to take into account negative values. Thus, for our extended similarity measure of Theorem 2, if some edges have negative weight, the total weight of a shortest path can be calculated as the product of the edges's weights, based on the assumption of multiplicative transitivity of the structural balance theory [15,

19], as formulated in the graph-theoretic language by Hage and Harary (1983). It is also important to note that the positivity property of the similarity values does not hold in the case of signed networks, as both basic and extended similarities can be positive or negative, based on the derived sigma values.

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". 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 an 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 8. Notice, that multiplicative transitivity can be also applied for all different-length shortest paths found in a signed graph.

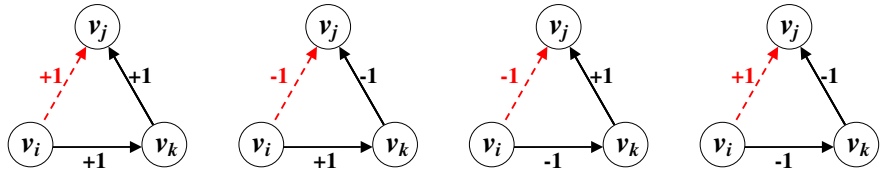


Fig. 8 The prediction of the sign of edge v_i,v_j (dashed line) based on the Structural Balanced Theory.

4 Experimental Evaluation

In this section, we compare experimentally our approach with existing friend recommendation algorithms. Henceforth, our proposed approach is denoted as FriendTNS. We use in the comparison the Random Walk with Restart [27] algorithm, the Shortest Path[8] algorithm, the Adamic and Adar [1] algorithm and the Friend of a Friend [4] algorithm, denoted as RWR, Shortest Path, Adamic/Adar and FOAF, respectively. Our experiments were performed on a 3 GHz Pentium IV, with 2 GB of memory. All algorithms were implemented in C. To evaluate the examined algorithms, we have generated synthetic data sets and chosen three real data sets from the Facebook, Hi5, and Epinions web sites.

4.1 Algorithms Settings

For each algorithm of our evaluation, next we will briefly describe the specific used settings:

FriendTNS algorithm: Our FriendTNS algorithm recommends friends to a target user v based on a user similarity matrix it constructs. Users can be recommended to v according to their weights associated with each $\{v, v_i\}$ pair in the user similarity matrix.

Random Walk with Restart: The "random walk with restart" (RWR) is a variation of the PageRank algorithm and will be used as a representative of the family of graph-based methods such as PageRank and HITS. It operates as follows: consider a random walker that starts from node v_x. The random walker chooses randomly among the available edges every time, except that before making a choice, he goes back to node v_x with probability c (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, as shown by Equation 3:

$$\mathbf{r}_{v_x} = c \cdot A \cdot \mathbf{r}_{v_x} + (1 - c) \cdot \mathbf{e}_{v_x} \tag{3}$$

where $\mathbf{e}_{v_x}$ is the $n \cdot 1$ starting vector with the v_x-th element equal to 1 and 0 for the other elements of the vector, whereas A is the adjacency matrix of graph $\mathcal{G}$.

Equation 3 defines a linear system problem where $\mathbf{r}_{v_x}$ is a $n \cdot 1$ ranking vector and element r_{v_x,v_y} is the relevance score of node v_y wrt node v_x, as shown by Equation 4:

$$\mathbf{r}_{v_x} = (1 - c) \cdot (I - c \cdot A)^{-1} \cdot \mathbf{e}_{v_x} \tag{4}$$

In our experiments, we tuned the c parameter with test values $(5 * 10^{-6}, 5 * 10^{-5}, ..., 5 * 10^{-1})$. The best recommendation accuracy attained with $c = 5 * 10^{-3}$. However, we have to notice that as c grows from 0.005 to 0.5 the recommendation accuracy slightly drops, which shows that RWR is insensitive to the choose of c.

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 according to their shortest distance in the social network. We use the Frendman-Tarjan algorithm [11] to calculate the shortest paths between any pair of nodes.

Adamic/Adar algorithm: For a node v_x, let $\Gamma(v_x)$ denote the set of neighbors of v_x in graph $\mathcal{G}$. A number of approaches are based on the idea that two nodes v_x, v_y are more likely to form a link in the future if their sets of neighbors $\Gamma(v_x), \Gamma(v_y)$ have large overlap; this follows the natural intuition that such nodes v_x, v_y represent people with many friends in common, and hence are more likely to come into contact themselves.

The Jaccard coefficient measures the probability that both v_x, v_y have a feature f, for a randomly selected feature f that either v_x or v_y has. Adamic and Adar [1] considered a related 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. Thus, for computing the similarity between two nodes v_x, v_y in a graph $\mathcal{G}$, we can use Equation 5:

$$score(v_x, v_y) = \sum_{z \in \Gamma(v_x) \cap \Gamma(v_y)} \frac{1}{\log |\Gamma(z)|} \tag{5}$$

Friend of a Friend algorithm: Friend of a Friend algorithm (FOAF) algorithm leverages only social network information of friending based on the intuition that

"if many of my friends consider Alice a friend, perhaps Alice could be my friend too". The clear intuition behind it [6], is the primary algorithmic foundation of the "People You May Know" feature on Facebook, which is one of the few known people recommenders deployed on a social networking site. Formally speaking, if we define predicate $F(v_i, v_j)$ to be true if and only if v_i is a friend of v_j, the algorithm can be described as follows: for a user v_x being the recipient of a recommendation, its recommendation candidate set is defined as follows [6]: $RC(v_x) = \{\text{user } v_c | \exists \text{ user } v_i \text{ s.t. } F(v_x, v_i) \text{ and } F(v_i, v_c)\}$. For each candidate $v_c \in RC(v_x)$, its common friends set is: $CF(v_x, v_c) = \{\text{user } v_i | F(v_x, v_i) \text{ and } F(v_i, v_c)\}$, which represents the friends of v_x that connect to v_c and thus serve as a bridge between v_x and v_c. We then define the score of each candidate v_c for recipient v_x as the size of $CF(v_x, v_c)$.

The candidates are recommended to v_x in decreasing score order. For a single recommended candidate v_c, we supply the common friends in $CF(v_x, v_c)$ as the explanation for recommending v_c. Thus, Facebook.com provides friend recommendations, considering only pathways of maximum length 2 between an individual and his possible friends in a social network. Therefore, users can be recommended to v_x according to the number on length-2 paths connecting them with him in the social network.

4.2 Real Data Sets

We used the Epinions[4] 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. It contains 49K users and 487K directed edges among pairs of users. Moreover, we crawled the Facebook website on the 30th of October, 2009. 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. We created a data set with 3694 users, denoted as Facebook 3.7K. Moreover, from the Hi5 web site, we crawled 63329 users and all of their friends, denoted as Hi5 63K[5], available from the Hi5 site on the 15th of April, 2009. Finally, we also use in our comparison the extended signed Epinions 132K data set from the Stanford Large Network Dataset Collection (SNAP)[6], which consist of positive and negative edges. A positive edge implies friendship/trust whereas a negative edge implies enmity/distrust.

4.3 Generation of Synthetic Network Model

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. 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 by means of a ten-dimensional vector with each element a randomly selected

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

[5] Our Facebook and Hi5 data sets are available in our web site: `http://delab.csd.auth.gr/~symeon`

[6] http://snap.stanford.edu/data/index.html

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. In particular, we define the similarity between two nodes as the scalar product of their corresponding vectors.

Given a network size N and the degree k of each node, 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 $1 - pr$, while a randomly chosen one with probability pr. This process will be terminated when all nodes are of degree k. The parameter $pr \in [0, 1]$ 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 2 synthetic data sets based on different network sizes N (1000, 100000), with k nodes degree equal to 10 for the first synthetic data set and with k equal to 100 for the second synthetic data set, whereas pr is equal to 0.2 for both data sets. For further reading about the procedure of links generation for the synthetic data set see [23].

4.4 Basic Topological Properties of the Real and Synthetic Data Sets

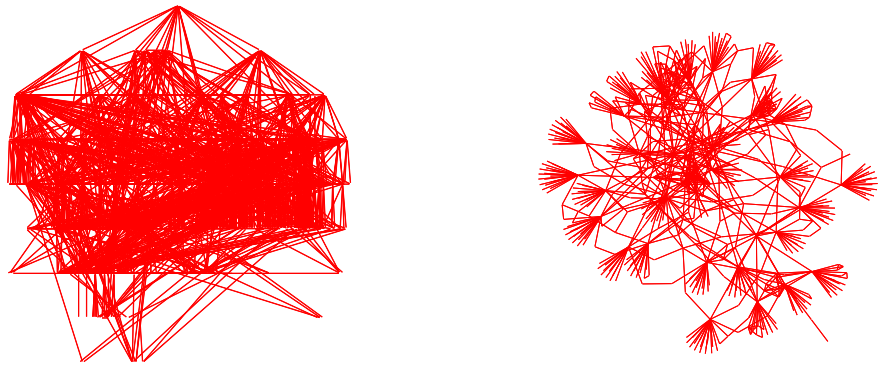
We calculated several topological properties of the synthetic and real data sets which are presented in Figure 9.

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
Facebook 3.7K	Undirected	3694	13692	3.73	7.41	0.32	10
Hi5 63K	Undirected	63329	88261	7.18	2.78	0.02	19
Epinions 49K	Directed	49288	487183	4.01	19.76	0.26	14
Synthetic (N=1000, k=10)	Undirected	1000	5000	2.81	10	0.01	4
Synthetic (N=100000, k=100)	Undirected	100000	5000000	3.56	100	0.001	16
Epinions 132K	Signed	131828	841372	1.78	6.38	0.24	14

Fig. 9 Topological properties of the real and synthetic data sets.

As shown in Figure 9, Epinions 49K and Facebook 3.7K data sets present (i) a large clustering coefficient (LCC), and (ii) a small average shortest path length (ASD). 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



(a) (b)

Fig. 10 Virtual planar graph representation of (a) Epinions 49K and (b) Hi5 63K data sets.

LCC). Moreover, most pairs of nodes are connected by at least one short path (i.e., small ASD).

In contrast, as also shown in Figure 9, Hi5 63K has a very small LCC (0.02) and a quite big ASD (7.18). In other words, Hi5 data set can not be considered as a small-world network, since (i) most of its nodes can not 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.

4.5 Sparsity Analysis of the Real Data Sets

The *sparsity* of a social network is determined by its node degrees. The smaller the degrees of nodes in a social network, the sparser it is. Most real OSNs are very sparse, as the degree of their nodes is significantly smaller than their total number of nodes: $(deg(v_i) << n, \forall v_i \in \mathcal{V})$.

To examine the sparsity of Epinions 49K and Hi5 63K data sets, we enumerated their edges and node degrees and designed their virtual graph representations. Figure 10 depicts the graph visualizations of the tested data sets using Frick's GEM and Cone algorithms of the *Tulip* project [2]. We only present a fraction of the graph visualization. In particular, we present the node with the highest degree (in the center of the graph visualizations), and all connected nodes to it in a distance of 7 hops. As shown, Epinions 49K is more dense than Hi5 63K, having many more edges and higher degree nodes. As will be experimentally shown later, a higher network sparsity harms the algorithms' accuracy performance. Notice that for Facebook 3.7K data set, we have verified that the graph representation is similar to Epinions 49K on a smaller scale. The reason is that they both can be considered as small world networks, having similar main characteristics (high LCC, high ADEG, and small ASD).

4.6 Experimental Protocol and Evaluation Metrics

Our evaluation considers the random division of friends of each target user into two sets: (i) the training set $\mathcal{E}^T$ is treated as known information and, (ii) the probe

set $\mathcal{E}^P$ is used for testing and no information in the probe set is allowed to be used for prediction. It is obvious that, $\mathcal{E} = \mathcal{E}^T \cup \mathcal{E}^P$ and $\mathcal{E}^T \cap \mathcal{E}^P = \oslash$. Therefore, for a target user we generate the recommendations based only on the friends in $\mathcal{E}^T$.

Each experiment has been repeated 30 times (each time a different training set is selected at random) and the presented measurements, based on two-tailed t-test, are statistically significant at the 0.05 level. All algorithms have the task to predict the friends of the target users in the probe set.

We use the classic precision/recall metric as performance measure for friend recommendations. For a test user receiving a list of k 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, 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 et al. and other papers [7,23]. AUC is equivalent to the area under the receiver-operating characteristic (ROC) curve. In our problem context, the AUC statistic can be interpreted as the probability that a randomly chosen missing link (a true positive) is given a higher score than a randomly chosen non-existent link (a true negative). In particular, 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 than a randomly chosen non-existent link and n'' times the missing link and nonexistent link having the same similarity value, we define AUC by Equation 6:

$$AUC = \frac{n' + 0.5 \times n''}{n} \tag{6}$$

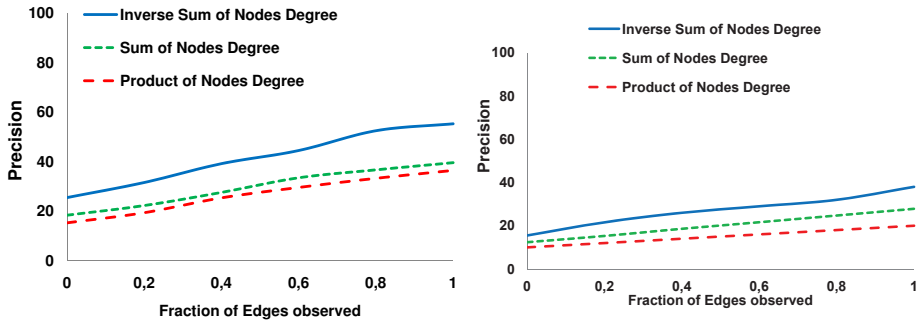
If all similarity values are generated from an independent and identical distribution, the accuracy should be about 0.5. Therefore, the degree to which the accuracy exceeds 0.5 indicates how much better the algorithm performs than pure chance.

4.7 Sensitivity Analysis of FriendTNS on synthetic networks

In this Section, we study the sensitivity of FriendTNS in terms of accuracy performance. In particular, we test how the performance of FriendTNS is affected, when (i) it is combined with different basic similarity metrics, (ii) it runs on synthetic data sets with different controllable sparsity, and (iii) it runs with different graph model randomness.

As the basic similarities of our proposed algorithms are calculated using the inverse sum of node degrees (Theorem 1), it is very interesting to compare the precision of our basic similarity measure with the corresponding precision of two other similarity measures, which are based on preferential attachment process [20]:

the sum of node degrees and the product of node degrees. The basic premise of preferential attachment is that the probability that a new edge involves node v_i is proportional to current number of neighbors of v_i. Thus, we used the three aforementioned measures to calculate the extended similarities of FriendTNS algorithm for the synthetic 1K data set and then, we computed the precision attained by each measure vs. the fraction of observed links used in the training set (pr parameter is fixed to 0.2). Figure 11a depicts the results.



(a) (b)

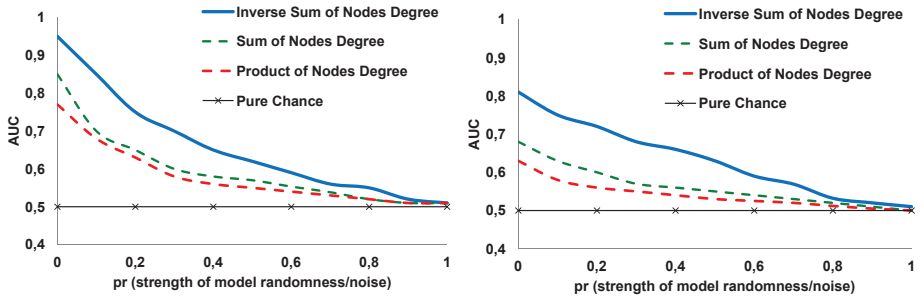
Fig. 11 Precision vs. different basic similarity measures for (a) the synthetic 1K synthetic data set and (b) the 100K synthetic data set.

We observe that the inverse sum of nodes degree measure outperforms both other measures. The same result holds for the 100K synthetic data set, as shown in Figure 11b. In the following, we adopt the inverse sum of nodes degree measure as the default basic similarity measure of FriendTNS. Notice also that, as we increase the fraction of observed edges, the precision of all algorithms is increased too. This is reasonable, since every prediction algorithm is expected to give higher accuracy for a denser network.

In our synthetic model, the parameter $pr \in [0, 1]$ represents the strength of randomness in generating links. Next, we test FriendTNS's sensitivity with different graph model randomness. For the 1K synthetic data set, as shown in Figure 12a, when the strength of randomness is weak, the inverse sum of nodes degree performs better than the other metrics. However, as the strength of randomness becomes high all metrics cannot perform better than pure chance. The same result holds for the 100K synthetic data set, as shown in Figure 12b.

4.8 Accuracy Comparison of FriendTNS with other methods

Next, we proceed with the comparison of FriendTNS with RWR, Shortest Path, Adamic-Adar, and FOAF algorithms, in terms of precision and recall. For this comparison, the percentage of links used in the training E^T set is fixed to 50% of the $\mathcal{E}$ set of graph edges. The precision vs. recall curve will reveal the robustness of each algorithm in attaining high recall with minimal losses in terms of precision. We examine the top-k ranked list, which is recommended to a target user, starting from the top friend. In this situation, the recall and precision vary as we proceed



(a) (b)

Fig. 12 AUC vs. *pr* strength of graph model randomness for (a) the synthetic 1K data set and (2) the synthetic 100K data set.

with the examination of the top-k list. Notice that the default value of k parameter is set to 10 unless it is different declared in the text.

For the Facebook 3.7K data set, in Figure 13 we plot a precision vs. recall curve for all algorithms. As expected, all algorithms' precision falls as k increases. In contrast, as k increases, recall for all algorithms increases as well. FriendTNS attains the best results with impressive high precision. The reason is that FriendTNS exploits global and local features of the social graph by combining the basic with the extended similarity measure. In contrast, RWR traverses only globally the social network, missing to capture adequately the local characteristics of the graph. Moreover, Shortest Path does not take into account the increased similarity between connected nodes that do not share many edges with others. Furthermore, Adamic/Adar and FOAF algorithms exploit only local features of the social network.

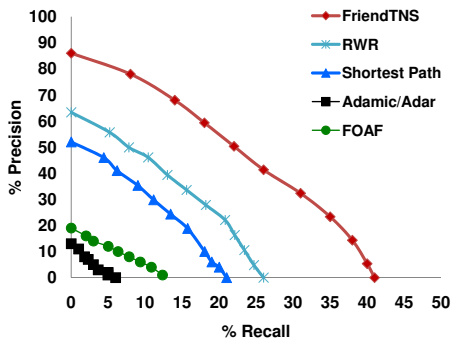


Fig. 13 Comparison of FriendTNS, RWR, Shortest Path, Adamic/Adar and the FOAF algorithm for the Facebook 3.7K data set.

The precision of FriendTNS is impressive in this specific data set. The main reason is the topological characteristics of Facebook 3.7K data set. It presents (i) a large clustering coefficient (LCC) equal to 0.32, and (ii) a small average shortest path length (ASD) equal to 3.73. Thus, Facebook 3.7K data set can be considered

as a small-world network. That is, small-world networks have links that connect almost any two nodes (i.e.high LLC). Moreover, most pairs of nodes are connected by at least one short path (i.e. small ASD). Thus, individuals who attempt to locate new friends in an small-world network can more easily find them and recommender systems are more effective.

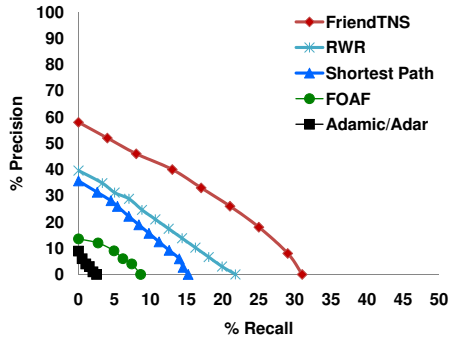


Fig. 14 Comparison of FriendTNS, RWR, Shortest Path, Adamic/Adar, and the FOAF algorithm for the Epinions 49K data set.

For the Epinions 49K data set, as shown in Figure 14, we also plot precision vs. recall curve for all algorithms. FriendTNS again attains the best results. The precision of FriendTNS is again quite high. Based on its topological features, Epinions 49K can be also considered as a small-world network, since it presents high LCC and small ASD.

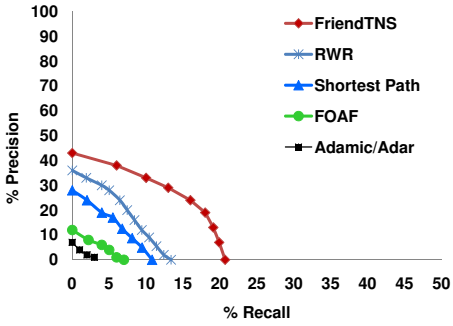


Fig. 15 Comparison of FriendTNS, RWR, Shortest Path, Adamic/Adar, and the FOAF algorithm for the Hi5 63K data set.

For the Hi5 63K data set, as shown in Figure 15, we plot precision vs. recall curve for all algorithms. However, the overall performance of FriendTNS, RWR and Shortest Path algorithms is significantly decreased compared with the results in both previous data sets. The main reason is the high sparsity (i.e., very small ADEG equal to 2.78) of the Hi5 data set. Moreover, it has very small LCC and quite big ASD (7.18). In other words, Hi5 data set can not be considered as a

small-world network. That is, most of its nodes can not be reached easily from other nodes and there are no cliques. Thus, a recommender system can not be as effective as it was in the previous two data sets.

4.9 Comparison of FriendTNS with Randomness

In this Section, we compare FriendTNS and the other methods with a pure chance baseline algorithm that predicts missing links, by simply randomly selecting pairs of nodes in graph $\mathcal{G}$, to be connected (i.e., to be friends).

One way to quantify the effectiveness of a link prediction method [20], is the ratio between the probability that the top-ranked pair of nodes is connected and the probability that a randomly chosen pair of nodes is connected. For instance, for the Facebook 3.7K data set, if each user was connected with all others, we would have 6820971 $[\frac{3694*3694-3694}{2}]$ graph edges. However, in the Facebook 3.7K test data set, we have 13692 edges. If the percentage of links used in the training E^T set were fixed to 50% of the $\mathcal{E}$ set of graph edges, then the remaining edges in the probe E^P set would be 6846. Thus, if we randomly proposed a new top-1 friend to a target user u, the probability that he is a friend is 0.001 ($\frac{6846}{6820971}$). This means that for the Facebook 3.7K data set, if we divide FriendTNS's precision (0.83) with the random's predictor precision (0.001), we get the factor improvement (830 times) of FriendTNS over this random predictor. Following the same procedure for the Epinions 49K and the Hi5 63K data set, the random's predictor precision is computed to 0.000022 and 0.0002, respectively.

This random friendship guess is denoted as Random predictor. Figure 16 shows each algorithm's performance on each data set (Facebook 3.7K, Epinions 49K, and Hi5 63K), in terms of precision as a factor of improvement over Random predictor, when we recommend a top-1 friend to a target user u and E^T=50%. Each algorithm's performance is obtained by averaging over 30 independent realizations. Bold entries represent the best algorithm performance. We can see that all five methods significantly outperform the random predictor. 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 three 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 nodes will have a missing connection. Thus, such a connection can be easily predicted by even simple heuristics such as the product of nodes degree or the 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. For the Epinions 49K data set, as shown in Figure 17a, we plot a curve for AUC vs. the fraction of observed links used in the training set. As shown, as a greater fraction of the network is known, the accuracy becomes even greater, for all methods. FriendTNS does far better than pure chance, indicating that it is a strong predictor

	Facebook 3.7K	Epinions 49K	Hi5 63K
FriendTNS	**830**	**2950**	**19789**
RWR	631	2002	17543
Shortest Path	519	1932	15151
Adamic-Adar	204	650	5502
FOAF	151	531	4544

Fig. 16 Factor of improvement of FriendTNS, RWR, Shortest Path, Adamic/Adar, and the FOAF algorithm over Random Predictor

of missing structure. The main reason is that FriendTNS captures effectively the local and global graph features. Notice that we also verify the same algorithms' rank for the Facebook and Hi5 data sets, as shown in Figures 17b and 17c, respectively.

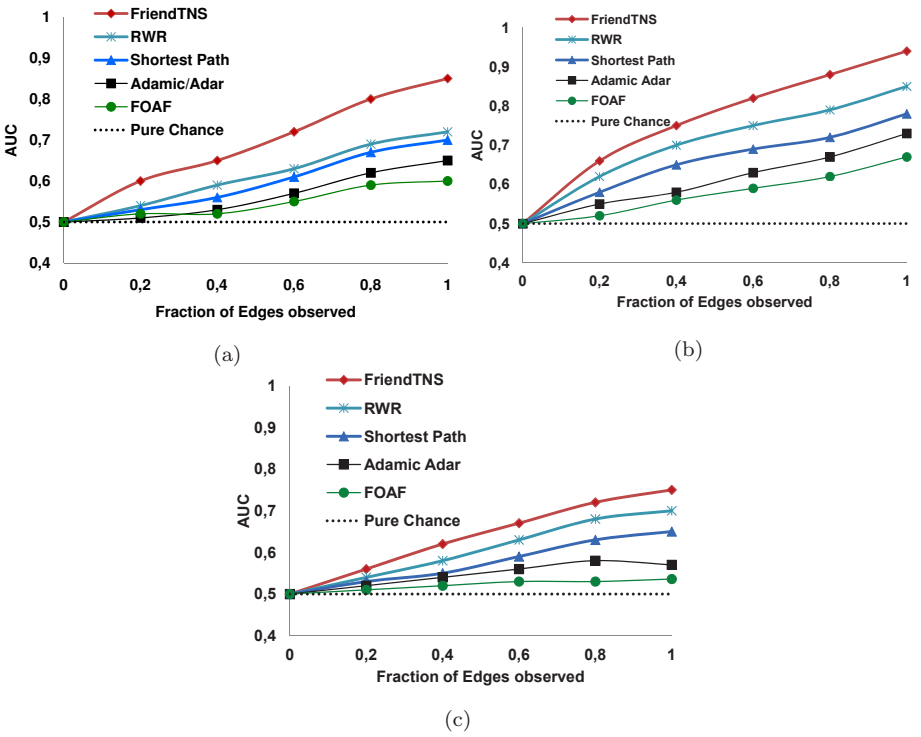


Fig. 17 Comparison of FriendTNS, RWR, Shortest Path, Adamic/Adar, FOAF and the pure chance algorithm for (a) the Epinions 49K, (b) the Facebook 3.7K and (c) the Hi5 data sets.

4.10 Time Comparison with other methods

In this section, we compare FriendTNS against the RWR, Shortest Path, Adamic/Adar, and FOAF algorithms in terms of time complexity. We have created 2 synthetic data sets based on different network sizes N (1000, 100000), where N is the total

number of nodes in the network (and is the main time complexity factor). For the first synthetic data set the k nodes degree is equal to 10, whereas k is equal to 100 for the second one. All recorded times are refer to the total required time for calculating similarities for a target node with all other nodes in a graph. Each algorithm's performance is obtained by averaging over 30 independent realizations. Figure 18 depicts the results. As shown, RWR present the worst results because it calculates the inverse of an $n \times n$ matrix. FOAF and Adamic/Adar algorithms, outperform Shortest Path and RWR due to their simpler complexity since they are local-based similarity measures. However, as already shown in Section 4.8, both methods give the worst results in terms of accuracy. Moreover, FriendTNS outperforms all other methods since its threshold parameter p causes the calculations to be performed only in small parts of the network around the selected user.

Data-Set	FriendTNS	Shortest Path	Adamic/Adar	RWR	FOAF
Synthetic-(N=1000, k=10)	0.001sec	0.011sec	0.003sec	0.017sec	0.002sec
Synthetic-(N=100000, k=100)	0.117sec	1.05sec	0.394sec	1.621sec	0.280sec

Fig. 18 Time complexity of the synthetic data sets. A smaller value is better.

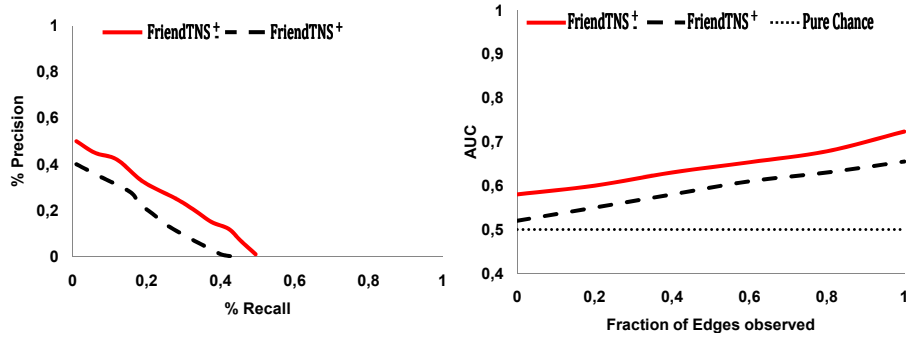
4.11 FriendTNS Accuracy Performance in Signed Networks

In this section, we present the accuracy performance of FriendTNS 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 $FriendTNS^{+}$. The second variation considers both positive and negative links and is denoted as $FriendTNS^{+}_{-}$. Figure 19a presents the precision and recall diagram for both versions of FriendTNS, whereas Figure 19b presents the AUC accuracy statistic. Both Figures show that $FriendTNS^{+}_{-}$ outperforms $FriendTNS^{+}$. The reason is that $FriendTNS^{+}_{-}$ 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 FriendTNS 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.

5 Discussion

There are many difficulties in the study of the link prediction problem. A first problem is the data sparsity [28] of the real online social networks. That is, 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 density and also study sparsity in real data sets.

A second problem is the huge size of real systems. For instance, Facebook in December 2012 announced that it counts more than 1 billion registered users



(a)

(b)

Fig. 19 Accuracy performance of FriendTNS in terms of (a) precision/recall and (b) AUC statistic.

and more than 500 millions of active users in any given day [7] with an average of roughly 150 friends each. This means that our data sample collected is extremely small relative to the overall graph. To study computational complexity issues, we tested our algorithm's performance with synthetic network models of different sizes (different total number of nodes).

Real networks have many complex structural properties [9], such as nodes' degree heterogeneity, the rich-club phenomenon, the mixing pattern, etc. These network properties are not considered by our simple synthetic network model, because they are out of the scope of this paper. However, our synthetic network model can be easily extended to be closer to the real networks. For example, by applying the nodes' degree heterogeneity index [9] with a probability pr, a synthetic network with different level of nodes' degree heterogeneity can be composed. Notice also that we use in our experiments the Epinions 49K and the extended signed Epinions 132K data sets, which are benchmarks for many other scientific works and can be downloaded from the Stanford Large Network Dataset Collection (SNAP).[8]

Finally, many real networks are evolving all the time in a day-to-day basis, and new emerging nodes/edges are added in the network structure. FriendTNS can incrementally update the similarity values between nodes, without re-calculating them from scratch. That is, FriendTNS discovers and expands only the required parts of the network and computes similarities only for the target user, keeping time and space complexity low. In other words, FriendTNS does not make calculations for the whole graph (i.e., for all users). In contrast, it performs calculations only for the target user and for the current network structure. Therefore, the similarities of the target user are calculated on-the-fly, even if there are changes in the network structure. Notice that in cases of deletions or changes in the connections of existing nodes, the affected similarities must be recalculated. The aforementioned advantages of our algorithm can be useful for cases, where long term digital preservation of the social network connections is needed. Thus, as data in large networks (with billions of nodes) are aggregated over time, our method is scalable to efficiently process them.

[7] Facebook's statistics included in this section were obtained from http://newsroom.fb.com
[8] http://snap.stanford.edu/data/index.html

6 Conclusions

In this paper, we proposed the FriendTNS algorithm to provide more accurate friend recommendations by computing effectively the proximity between any pair of connected nodes in an OSN. We defined a transitive node similarity measure in OSNs by taking into account local and global features of a social graph. We also derived variants of our method that apply to different types of networks (directed/undirected, weighted/unweighted, and signed/unsigned). We performed an experimental comparison of the proposed method against existing recommendation algorithms using the three real social networks (Facebook, Epinions, and Hi5 data sets). We have shown that our FriendTNS algorithm provides more accurate friend recommendations compared to existing approaches. In the future, we want to examine other ways of improving friend recommendations based on other features that OSNs offer, such as photo and video tagging, groups and common applications. The combination of such features can provide information on different ways that users are connected and therefore yield to more accurate friend recommendations.

References

1. L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.
2. D. Auber. Tulip : a huge graph visualization framework. *Graph Drawing Softwares, Mathematics and Visualization*, pages 105–126, 2003.
3. H. Bisgin, N. Agarwal, and X. Xu. A study of homophily on social media. *World Wide Web*, 15(2):213–232, 2012.
4. F. Blog. *http://blog.facebook.com/blog.php?post=15610312130*.
5. V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review*, 46(4):647–666, 2004.
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 *Proceedings 27th International Conference on Human Factors in Computing Systems (CHI)*, pages 201–210, 2009.
7. A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453, 2008.
8. T. Cormen, C. Leiserson, R. Rivest, and S. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2001.
9. L. Costa, F. Rodrigues, G. Travieso, and P. Boas. Characterization of complex networks: A survey of measurements. 56(1):167–242, 2007.
10. 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.
11. M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.
12. S. Goel, R. Muhamad, and D. Watts. Social search in 'small-world' experiments. In *Proceedings 18th International World Wide Web Conference (WWW)*, pages 701–710, Madrid, Spain, 2009.
13. J. Golbeck. Personalizing applications through integration of inferred trust values in semantic web-based social networks. In *Semantic Network Analysis Workshop at the 4th International Semantic Web Conference*, 2005.
14. I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. In *Proceedings of 13th International Conference on Intelligent User Interfaces (IUI)*, pages 77–86, 2009.
15. P. Hage and F. Harary. *Structural models in anthropology*, volume 1983. Cambridge University Press Cambridge, 1983.

16. G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 538–543, 2002.
17. J. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proceedings 32nd ACM Symposium on Theory of Computing (STOC)*, pages 163–170, Portland, OR, 2000.
18. J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World Wide Web (WWW)*, pages 641–650, 2010.
19. 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 (CHI)*, pages 1361–1370, 2010.
20. 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.
21. L. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *National Academy of Sciences (PNAS)*, 102(33):11623–11628, 2005.
22. S. Lo and C. Lin. Wmr: a graph-based algorithm for friend recommendation. In *Proceedings of IEEE/ACM International Conference on Web Intelligence (WIC)*, pages 121–128, Hong Kong, China, 2006.
23. L. Lü, C.-H. Jin, and T. Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.
24. P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of International/Federated Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, pages 492–508, 2004.
25. S. Milgram. The small world problem. *Psychology Today*, 22:61–67, 1967.
26. K. Musiał and P. Kazienko. Social networks on the internet. *World Wide Web*, 16(1):31–72, 2013.
27. J. Pan, H. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 653–658, Seattle, WA, 2004.
28. M. Rattigan and D. Jensen. The case for anomalous link discovery. *SIGKDD Explorations*, 7(2):41–47, 2005.
29. O. Simsek and D. Jensen. Decentralized search in networks using homophily and degree disparity. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 304–310, Edinburgh, Scotland, 2005.
30. T. Tanimoto. *IBM Internal Technical Report*, 1957.
31. S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.
32. D. Watts, P. Dodds, and M. Newman. Identity and search in social networks. *Science Magazine*, 296(5571):1302–1305, 2002.
33. W. Yu, W. Zhang, X. Lin, Q. Zhang, and J. Le. A space and time efficient algorithm for simrank computation. *World Wide Web*, 15(3):327–353, 2012.

7 Appendix

The basic node similarity measure, defined in Equation 1, satisfies fundamental properties such as: positivity, reflexivity, symmetry.

Theorem 3 *The basic node similarity measure satisfies the positivity property.*

Proof For any two nodes v_i, v_j of graph $\mathcal{G}$ we have:

$$sim(v_i, v_j) = \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}{\sum_{h=1}^{m} \left(R[e_h, v_i]^2 + R[e_h, v_j]^2 - R[e_h, v_i] \cdot R[e_h, v_j] \right)}$$

$$= \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}{\sum_{h=1}^{m} \Big[(R[e_h, v_i] - R[e_h, v_j])^2 + R[e_h, v_i] \cdot R[e_h, v_j] \Big]}$$

$$= \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}{\sum_{h=1}^{m} (R[e_h, v_i] - R[e_h, v_j])^2 + \sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}$$

All sums in the later equation are positive numbers, thus the positivity property holds: $sim(v_i, v_j) \geq 0$. Moreover, as the second sum on the denominator is equal with the numerator, we have that $sim(v_i, v_j) \leq 1$, and the theorem has been proven. □

Theorem 4 *The basic node similarity measure satisfies the reflexivity property.*

Proof For any node v_i of graph $\mathcal{G}$ we have:

$$sim(v_i, v_i) = \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_i]}{\sum_{h=1}^{m} \big(R[e_h, v_i]^2 + R[e_h, v_i]^2 - R[e_h, v_i] \cdot R[e_h, v_i] \big)}$$

$$= \frac{\sum_{h=1}^{m} R[e_h, v_i]^2}{\sum_{h=1}^{m} \big(R[e_h, v_i]^2 + R[e_h, v_i]^2 - R[e_h, v_i]^2 \big)} = 1$$

and the theorem has been proven. □

Theorem 5 *The basic node similarity measure satisfies the symmetry property.*

Proof For any two nodes v_i, v_j of graph $\mathcal{G}$ we have:

$$sim(v_i, v_j) = \frac{\sum_{h=1}^{m} R[e_h, v_i] \cdot R[e_h, v_j]}{\sum_{h=1}^{m} \big(R[e_h, v_i]^2 + R[e_h, v_j]^2 - R[e_h, v_i] \cdot R[e_h, v_j] \big)}$$

$$= \frac{\sum_{h=1}^{m} R[e_h, v_j] \cdot R[e_h, v_i]}{\sum_{h=1}^{m} \big(R[e_h, v_j]^2 + R[e_h, v_i]^2 - R[e_h, v_j] \cdot R[e_h, v_i] \big)} = sim(v_j, v_i)$$

and the theorem has been proven. □

The extended node similarity measure satisfies all fundamental properties and advantages that the basic node similarity measure has: positivity, reflexivity, symmetry.

Theorem 6 *The extended node similarity measure satisfies the positivity property, and, furthermore, it returns values in the interval [0,1].*

Proof If there is no path between v_i, v_j, then we have: $esim(v_i, v_j) = 0$. In case that v_i, v_j are neighbors, we have: $esim(v_i, v_j) = sim(v_i, v_j) \in [0, 1]$, otherwise we have: $esim(v_i, v_j) = \prod_{h=1}^{k} sim(v_{p_h}, v_{p_{h+1}})$. As all $sim(v_{p_h}, v_{p_{h+1}})$ values are into [0,1], the same will also hold for their product. Thus the theorem has been proven. □

Theorem 7 *The extended node similarity measure satisfies the reflexivity property.*

Proof We have that: $esim(v_i, v_i)$=$sim(v_i, v_i)$=1, and the theorem has been proven. □

Theorem 8 *The extended node similarity measure satisfies the symmetry property.*

Proof In case there is no path between v_i, v_j, we have $esim(v_i, v_j) = 0$. Also we have $esim(v_j, v_i) = 0$, thus: $esim(v_i, v_j) = esim(v_j, v_i)$. In case v_i, v_j are neighbors, we have: $esim(v_i, v_j) = sim(v_i, v_j) = sim(v_j, v_i) = esim(v_j, v_i)$. Otherwise, the shortest path from v_j to v_i will be the same path from v_i back to v_j, as the graph $\mathcal{G}$ is undirected and unweighted. Therefore, we will pass through the same nodes in both directions and the similarity values of the intermediate edges will be also the same. Thus, we have:

$$esim(v_i, v_j) = \prod_{h=1}^{k} sim(v_{p_h}, v_{p_{h+1}}) = \prod_{h=k}^{1} sim(v_{p_{h+1}}, v_{p_h}) = esim(v_j, v_i)$$

Thus the theorem has been proven. □