Scalable Link Prediction in Social Networks based on Local Graph Characteristics

Alexis Papadimitriou
Department of Informatics
Aristotle University
Thessaloniki, Greece
Email: apapadi@csd.auth.gr

Panagiotis Symeonidis
Department of Informatics
Aristotle University
Thessaloniki, Greece
Email: symeon@csd.auth.gr

Yannis Manolopoulos
Department of Informatics
Aristotle University
Thessaloniki, Greece
Email: manolopo@csd.auth.gr

***Abstract*—Online social networks (OSNs) like Facebook, Myspace, and Hi5 have become popular, because they allow users to easily share content or expand their social circle. OSNs recommend new friends to registered users based on local graph features (i.e. based on the number of common friends that two users share). However, OSNs do not exploit all different length paths of the network. Instead, they consider only pathways of maximum length 2 between a user and his candidate friends. On the other hand, there are global approaches, which detect the overall path structure in a network, being computationally prohibitive for huge-size social networks. In this paper, we provide friend recommendations, also known as the *link prediction* problem, by traversing all paths of a bounded length, based on the "algorithmic small world hypothesis". As a result, we are able to provide more accurate and faster friend recommendations. We perform an extensive experimental comparison of the proposed method against existing link prediction algorithms, using two real data sets (Hi5 and Epinions). Our experimental results show that our FriendLink algorithm outperforms other approaches in terms of effectiveness and efficiency in all data sets. Finally, we discuss extensively various experimental considerations, such as a possible MapReduce implementation of FriendLink algorithm to achieve scalability.**

***Keywords*-Social Networks; Link Prediction; friend recommendation; graph theory; similarity measure;**

I. 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 recommend other people to users based on their common friends. The reason is that there is a significant possibility that two users are friends, if they share a large number of common friends.

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 [7] to handle. The first one is based on local features of a network, focusing mainly on the nodes structure; the latter is based on global features, detecting the overall path structure in a network.

This work has been partially funded by the Greek GSRT (project number 10TUR/4-3-3) and the Turkish TUBITAK (project number 109E282) national agencies as part of Greek-Turkey 2011-2012 bilateral scientific cooperation.

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

A. Motivation

Compared to approaches which are based on local features of a network (i.e. Friend of a Friend (FOAF) algorithm or Common Neighbors, Adamic/Adar index, Jaccard Coefficient, etc. - for more details see Section II), we provide friend recommendations, exploiting paths of greater length. In contrast, they consider only pathways of maximum length 2 between a target user and his candidate friends. In our approach, we assume that a person can be connected to another with many paths of different length (through human chains). Thus, two persons connected with many unique pathways of different length have a high probability to know each other, proportionally to the length of the pathways they are connected with.

Our method is more efficient compared to global approaches (i.e Katz status index, RWR algorithm, SimRank algorithm etc.), which detect the overall path structure in a network. In other words, our method, which is based on a bounded path traversal, requires less time and space complexity than the global based algorithms. The reason is that we traverse only paths of length l in a network based on the "algorithmic small world hypothesis", whereas global approaches detect the overall path structure. (for more details see Section on Related Work).

The rest of this paper is organized as follows. Section II summarizes the related work. The proposed approach, its complexity analysis, and its possible extensions to other networks, are described in Section III. Experimental results are given in Section IV. Section V discusses basic research questions, such as a possible MapReduce implementation of FriendLink algorithm, the extension of our synthetic network model to better resemble real networks, and the use of linear regression to learn the attenuation factor used in FriendLink. Finally, Section VI concludes this paper.

II. Related work

There is a variety of local similarity measures [7] (i.e. FOAF algorithm, Adamic/Adar index, Jaccard Coefficient,

etc.) for analyzing the "proximity" of nodes in a network. FOAF [2] is adopted by many popular OSNs, such as facebook.com and hi5.com for the friend recommendation task. FOAF is based on the common sense that two nodes v_x, v_y are more likely to form a link in the future, if they have many common neighbors. In addition to FOAF algorithm, there are also other local-based measures such as Jaccard Coefficient [7] and Adamic/Adar index [1]. Adamic and Adar proposed a distance measure to decide when two personal home pages are strongly "related". In particular, they computed features of the pages and defined the similarity between two pages x, y as follows: $\sum_z \frac{1}{\log(frequency(z))}$, where z is a feature shared by pages x, y. This refines the simple counting of common features by weighting rarer features more heavily.

There is a variety of global approaches [7]. In this paper, as comparison partners of global approaches, we consider Katz status index [6], and Random Walk with Restart algorithm [11], [9] (RWR) algorithm. Katz defines a measure that directly sums over all paths between any pair of nodes in graph $\mathcal{G}$, exponentially damped by length to count short paths more heavily. RWR considers a random walker that starts from node v_x, and chooses randomly among the available edges every time, except that, before he makes a choice, with probability α, he goes back to node v_x (restart). The similarity matrix (i.e. Kernel) between nodes of a graph, can be computed by Equation 1:

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

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

III. The Proposed Approach

In this section we first provide a detailed explanation of our approach, named FriendLink. Then, we perform a complexity analysis comparison of FriendLink with other approaches and finally we extend Friendlink for other types of networks.

A. The FriendLink Algorithm

Friendlink computes node similarity between any two nodes in a graph $\mathcal{G}$. The initial input of Friendlink is the number n of nodes of $\mathcal{G}$, the adjacency matrix A, and the length ℓ of paths that will be explored in $\mathcal{G}$. To enumerate all simple paths in $\mathcal{G}$, Rubin's algorithm [10] can be employed. However, Rubin's algorithm uses O(n^3) matrix operations to find all paths of different length between any pair of nodes. In the following, we customize Rubin's algorithm to create only paths of length up to ℓ for our purpose.

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

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

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

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

```
28. Function Compute Similarity()
29. for i = 1 to n
30.    for j = 1 to n
31.        denominator = 1
32.            for k = 2 to m
33.                denominator = denominator * (n - k)
34.            end for k
35.    sim(i,j) = sim(i,j) + 1/(m-1) · |paths^m_{i,j}| / denominator
36.    end for j
37. end for i
38. return sim(i,j)
39. End Function
```

Figure 1. The FriendLink algorithm.

B. Complexity Analysis

RWR [9] and Katz index [6] as representatives of the global approaches, are computationally prohibitive for large graphs, because they require matrix inversion. For instance, the time complexity of Katz index is mainly determined

by this operation which costs $O(n^3)$. RWR algorithm also requires a matrix inversion, which can be, however, pre-computed [11] and stored for faster on-line operations. In the same direction, there is also a faster version [5] of Katz status index that reduces the computational complexity from time $O(n^3)$ to $O(n+m)$, where m is the number of edges.

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

In contrast to FOAF algorithm which traverses only length-2 paths, our FriendLink algorithm considers also paths with higher length (l-length paths). Based on Milgram's [8] "small-world hypothesis", l can take integer values in the interval [2,6], where for l=2 our FriendLink equals to the FOAF algorithm. Thus, FriendLink's time complexity is O($n \times h^l$). The space complexity for FriendLink is also O($n \times h$). Notice that in our code we store adjacent nodes using adjacency lists and not a matrix structure. However, for simplicity reasons, in Figure 1 we present our algorithm using a matrix structure.

C. Extending FriendLink for different types of Networks

Applying FriendLink to directed graphs can be achieved by: (i) simply disregarding the edge directions [12], or (ii) replacing the original adjacency matrix A with an asymmetric one [12]. For weighted networks, if edges weights are positive, FriendLink applies trivially.

IV. Experimental Evaluation

In this section, we compare experimentally FriendLink with RWR, Katz and FOAF algorithms, respectively. Our experiments were performed on a 3 GHz Pentium IV, with 2 GB of memory, running Windows XP. All algorithms were implemented in Matlab.

A. Real Data Sets

To evaluate the examined algorithms, we have used two real data sets from the Hi5 and Epinions web sites. We crawled the graph data from the Hi5 web site at two different time periods. In particular, we crawled the Hi5 web site on the 15th of April, 2010 and on the 20th of June, 2010. Our data crawling method was the following: for each user u, we traverse all his friends and then traverse the friends of each of u's friends etc.

From the first crawl of Hi5 web site we created a training data set with 63329 users and 88261 edges among them, denoted as Hi5 63K[4], where the initial starting node of our crawling was a random user in the US. From the second crawl of Hi5 web site we created the probe data set with the same users by only preserving 16512 new emerged edges connecting them. The graph data from the first crawl are used to predict the new links emerging in the second crawl.

We also use in our experiments the Epinions[5] data set, which is a who-trusts-whom social network. In particular, users of Epinions.com express their Web of Trust, i.e. reviewers whose reviews and ratings they have found to be valuable. The Epinions data set is a directed network and, thus, we treat it by simply disregarding the directions of links [12]. It contains 49K users and 487K edges among pairs of users.

We calculated several topological properties of the real data sets, which are presented in Figure 2. As shown, Epinions 49K presents (i) a large clustering coefficient (LCC) equal to 0.26, and (ii) a small average shortest path length (ASD) equal to 4.01. 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 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 2, Hi5 63K has a very small LLC (0.02) and a quite big ASD (7.18). In other words, Hi5 data set cannot be considered as a small-world network, since: (i) most of its nodes cannot be reached from every other by a small number of hops or steps, and (ii) does not have sub-networks that can be considered as cliques.

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

Data-Set	N	E	ASD	ADEG	LCC	GD
Hi5 63K	63329	88261	7.18	2.78	0.02	19
Epinions 49K	49288	487183	4.01	19.76	0.26	14

Figure 2. Topological properties of the real data sets.

B. Experimental Protocol and Evaluation Metrics

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

[4] http://delab.csd.auth.gr/∼symeon

[5] http://www.trustlet.org/wiki/

we average the results for each user and compute the final performance of each algorithm. Epinions data sets do not have time stamps of the edges. The performance of the algorithms is evaluated by applying double cross-validation (internal and external). Each data set was divided into 10 subsets. Each subset ($\mathcal{E}^P$) was in turn used for performance estimation in the external cross-validation. The 9 remaining subsets ($\mathcal{E}^T$) were used for the internal cross-validation. In particular, we performed an internal 9-fold cross-validation to determine the best values of the algorithms' needed parameters. In particular, for RWR we set parameter α= 0.001, whereas for Katz we set parameter β=0.005. We chose as values for the parameters those providing the best performance on the internal 9-fold cross-validation. Then, their performance is averaged on the external 10-fold cross-validation. The presented results, based on two-tailed t-test, are statistically significant at the 0.05 level.

We use the classic precision/recall metric as performance measure for friend recommendations. For a test user receiving a list of 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, since we provide to a test user u a top-k list of friends, it is important to consider the order of the presented friends in this list. That is, it is better to have a correct guess in the first places of the recommendation list. Thus, we use the **Mean Average Precision (MAP)** to emphasize ranking of relevant users higher. We define MAP by Equation 2:

$$MAP = \frac{1}{|N|}\sum_{u=1}^{|N|}\frac{1}{r_u}\sum_{k=1}^{r_u} Precision_u@k \qquad (2)$$

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

C. Sensitivity Analysis for the FriendLink Algorithm

In this Section, we study the sensitivity of FriendLink accuracy performance in real networks with: (i) different possible attenuation factors or (ii) different ℓ values for path traversal.

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

Table I
MAP FOR 5 ATTENUATION FACTORS ON OUR REAL DATA SETS.

Attenuation Factor	Epinions 49K	Hi5 63K
$1/(m-1)$	**0.445**	**0.154**
$1/(2m)$	0.390	0.139
$1/(m^2)$	0.322	0.099
$1/log(m)$	0.287	0.045
b^m	0.235	0.012

In Section III-A, one of the required input values for the FriendLink algorithm is the length ℓ of paths considered in a graph. To improve our recommendations, it is important to fine-tune the ℓ variable. Based on Milgram's [8] "small-world hypothesis", ℓ should take integer values in the interval [2,6]. Figures 3c and 3d illustrate precision for varying ℓ values for the Epinions 49K, and Hi5 63K data sets, respectively. As expected, precision decreases as the number of recommended friends is increased. The best precision is attained by $\ell = 3$ in both data sets. Notice that we omit to show results for $\ell = 6$ because precision follows a degraded performance for $\ell = 4$ and $\ell = 5$, respectively. In the following, we keep the $\ell = 3$ as the default maximum length of paths of the FriendLink algorithm.

D. Accuracy Comparison of FriendLink with other methods

We proceed with the comparison of FriendLink with RWR, Katz, and FOAF algorithms, in terms of precision and recall. This reveals the robustness of each algorithm in attaining high recall with minimal losses in terms of precision. We examine the ranked list, which is recommended to a target user, starting from the top friend. In this situation, the recall and precision vary with increasing number of recommended friends. For the Epinions 49K data set, as shown in Figure 3e, our FriendLink algorithm again attains the best precision value of 55% when we recommend a top-1 list of friends. The precision of FriendLink is impressive in this specific data set. The main reason is the topological characteristics of Epinions 49K data set (i.e. high LCC and small ASD). Thus, Epinions 49K can be considered as a small-world network. This experiment shows that FriendLink is more robust in finding relevant users for the test user. The reason is that FriendLink exploits proportionally the ℓ-length pathways between two graph nodes. In contrast, RWR and Katz traverse globally the social network, missing to capture adequately the local characteristics of the graph. Moreover, FOAF fails to provide

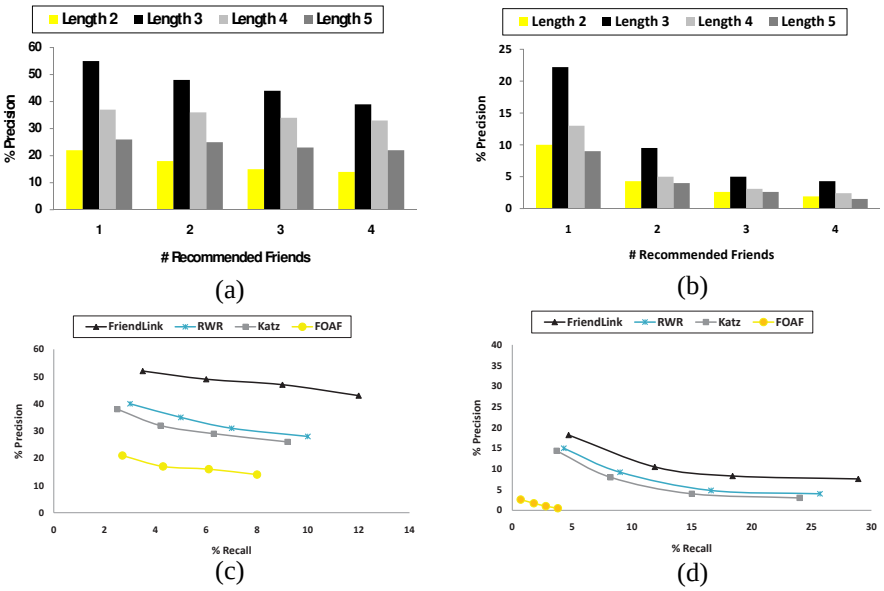

Figure 3. (a) Precision vs. number of recommended friends for Epinions 49K data set, (b) Precision vs. number of recommended friends for Hi5 63K data sets. (c) Precision vs. Recall diagram for Epinions 49K data set, (d) Precision vs. Recall diagram for Hi5 63K data set.

accurate recommendations because it exploits only length-2 paths.

For the Hi5 63K data set, as shown in Figure 3f, our FriendLink algorithm attains the best results. However, the overall performance of FriendLink, RWR and Katz algorithms is significantly decreased compared with the results in the Epinions data set. The first reason is the high sparsity (i.e. ADEG equal to 2.78) of the Hi5 63K data set. The second reason is the fact that Hi5 cannot be considered as a small-world network.

E. Time Comparison of FriendLink with other Methods

In this Section, we compare FriendLink against RWR, Katz and FOAF algorithms in terms of efficiency using both real and synthetic data sets. We measured the clock time for the off-line parts of all algorithms. The off-line part refers to the building of the similarity matrix between any pair of nodes in a graph. The results are presented in Table II. As shown, FriendLink outperforms RWR and Katz, since they calculate the inverse of an $n \times n$ matrix. As expected, FOAF algorithm, outperforms the other algorithms due to its simpler complexity. Notice that the results depict the time needed to compute the whole similarity matrix. On the other hand, if we were to calculate the similarity matrix of only one user, then the computation would require only part of a second to produce a recommendation.

Algorithm	**Epinions 49K**	**Hi5 63K**
FriendLink	245 sec	340 sec
RWR	380 sec	520 sec
Katz	460 sec	617 sec
FOAF	**55 sec**	**221 sec**

Table II
TIME COMPARISON OF ALL TESTED ALGORITHMS FOR THE SYNTHETIC AND REAL DATA SETS.

V. MAP REDUCE IMPLEMENTATION FOR FRIENDLINK ALGORITHM

There are many difficulties in the study of the link prediction problem. A first problem is the huge size of real systems. For instance, Facebook has over 500 million users with an average of roughly 100 friends each. For our algorithm to run for huge sized networks, it should be adjusted to support a MapReduce [4] implementation. MapReduce is a distributed computing model for processing large volumes of data. MapReduce is implemented in three steps: (i) Splitting up the computing job into chunks that standard machines can process in a short time, (ii) parallel processing on each sub-part by an independent machine and, (iii) the collection of intermediate values, produced by each machine, to calculate the final result. In our case, the calculation of the similarity matrix could be assigned to many machines in the following way. Each machine calculates one of the $2 \ldots \ell$-length paths for a specific pair of users and then sum up the paths to calculate the final similarity value. An example is shown in Figure 4. As shown in Figure 4, each Map function on every machine receives as input a pair of users and produces the similarity value for a designated path length ℓ. All values for each pair of users are collected into one final value in the reduce phase. In our example, the similarity values produced by the Map function, which are0.03, 0.2, 0.14 and 0.07 for path length

$\ell = 2, 3, 4, 5$ respectively, will be "reduced" to one final similarity value, which is 0.44, for the respective pair of users.

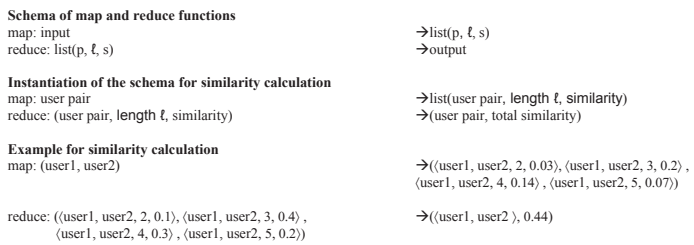

Figure 4. Map and Reduce functions in MapReduce.

Real networks have many complex structural properties [3], such as degree heterogeneity, the rich-club phenomenon, the mixing pattern, etc. These network properties are not considered by our synthetic network model, since they are out of the scope of this paper. However, our synthetic network model can be easily extended to better resemble real networks. For example, by applying the degree heterogeneity index [3] with a probability p, a synthetic network with different level of degree heterogeneity can be composed.

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

VI. Conclusions

In this paper, we introduced a framework to provide friend recommendations in OSNs. We defined a new node similarity measure that exploits local and global characteristics of a network. We performed extensive experimental comparison of the proposed method against existing link prediction algorithms showing that our FriendLink algorithm provides more efficient and more accurate friend recommendations. We also discussed a possible MapReduce implementation of FriendLink, as well as ways to improve our synthetic model to better resemble real networks and also learn the optimal attenuation factor used in FriendLink instead of guessing it. 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] 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 International Conference on Human factors in Computing Systems (CHI)*, pages 201–210, 2009.

[3] L. Costa, F. Rodrigues, G. Travieso, and P. Boas. Characterization of complex networks: A survey of measurements. 56(1):167–242, 2007.

[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51:107–113, January 2008.

[5] K. C. Foster, S. Q. Muth, J. J. Potterat, and R. B. Rothenberg. A faster katz status score algorithm. *Computational and Mathematical Organization Theory*, 7:275–285, 2001.

[6] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[7] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. *Proceedings International Conference on Information and Knowledge Management (CIKM)*, 2003.

[8] S. Milgram. The small world problem. *Psychology Today*, 22:61–67, 1967.

[9] J. Pan, H. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 653–658, 2004.

[10] F. Rubin. Enumerating all simple paths in a graph. *IEEE Transactions on Circuits and Systems*, 25(8):641–642, 1978.

[11] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *Proceedings International Conference on Data Mining (ICDM)*, pages 613–622, 2006.

[12] S. Wasserman and K. Faust. Social network analysis: Methods and applications. 1994.