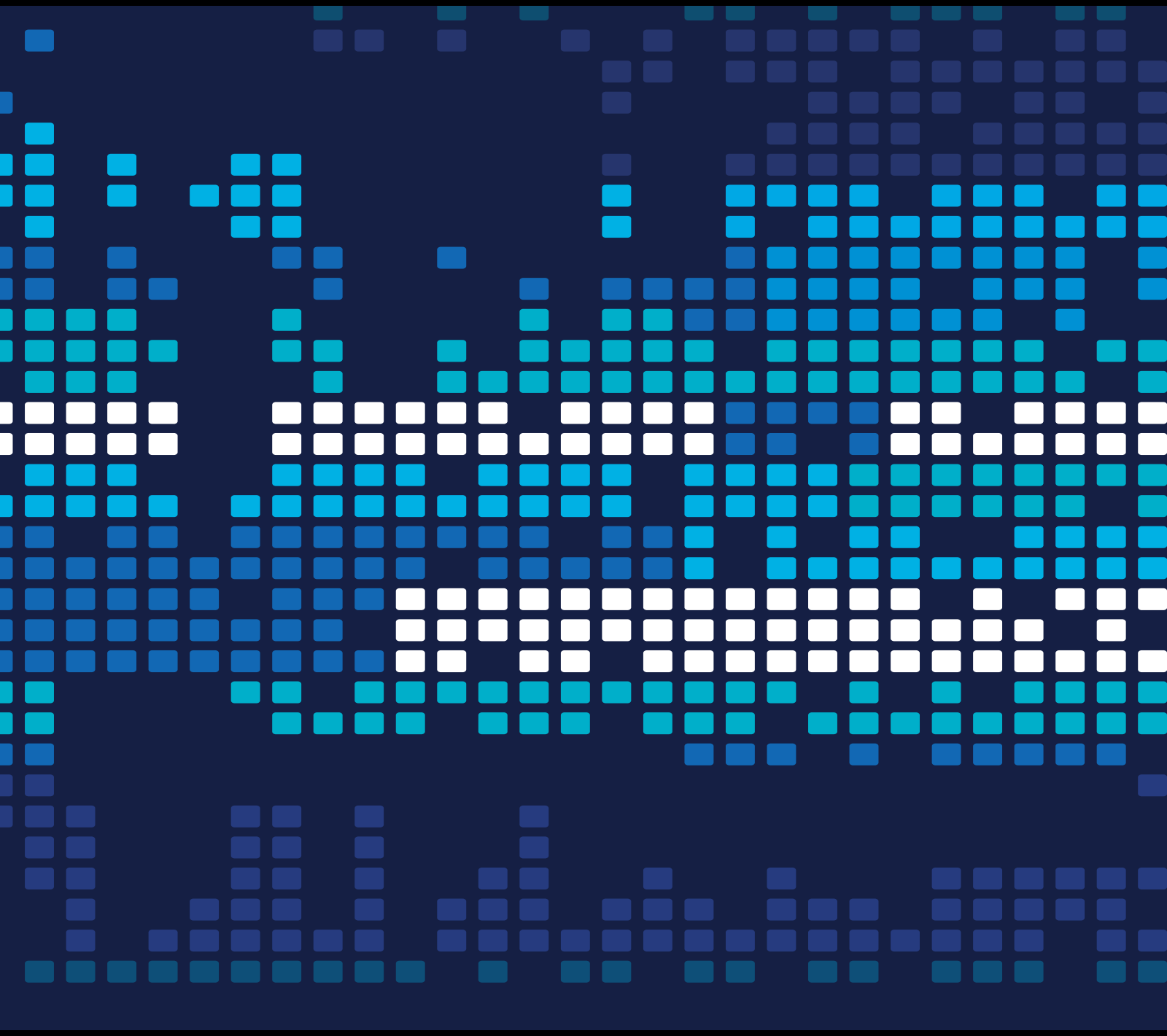
Scientific Programming

Computational Aspects of Social Network Analysis

Guest Editors: Przemyslaw Kazienko, Reda Alhajj, and Jaideep Srivastava



Computational Aspects of Social Network Analysis

Scientific Programming

Computational Aspects of Social Network Analysis

Guest Editors: Przemyslaw Kazienko, Reda Alhajj, and Jaideep Srivastava

Editorial Board

Contents

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 961610, 2 pages
http://dx.doi.org/10.1155/2015/961610

Editorial

Computational Aspects of Social Network Analysis

Przemyslaw Kazienko,[1] Reda Alhajj,[2] and Jaideep Srivastava[3]

[1] *Wroclaw University of Technology, 50-370 Wroclaw, Poland*
[2] *University of Calgary, Calgary, AB, Canada T2N 1N4*
[3] *University of Minnesota, Minneapolis, MN 5545, USA*

Correspondence should be addressed to Przemyslaw Kazienko; przemyslaw.kazienko@pwr.edu.pl

Received 12 March 2015; Accepted 12 March 2015

The interdisciplinary research field commonly called Social Network Analysis (SNA) has attracted many scientists from various disciplines from physics, sociology, anthropology, psychology, management, and also computer science. Various methodologies used in different disciplines are complementary to computational methods used to process data about human activities, profiles, or direct social relationships.

Users of IT systems leave their traces in most of these systems and the Internet is the largest source of data about human behavior, mutual interactions, and collaboration. It refers especially to social media like Twitter or Facebook as well as many other Web 2.0 services. This data is widely utilized to study social phenomenon and if they also respect social relationships, then we can say that SNA method is applied there.

It includes seven papers that come closer to various problems related to efficient processing of large social networks, link prediction, community detection, and analysis of human behavior and skills.

The paper entitled "A Community Based Approach for Link Prediction in Signed Social Networks" provides methods to predict the sign of human relationships: either positive or negative. These approaches are based on analysis of stable social communities and creation of appropriate node reputation rankings. The experiments on real data sets proved the high accuracy of the approach.

The paper entitled "Parallelizing SLPA for Scalable Overlapping Community Detection" is also related to social communities, in particular detection of overlapping groups in the large-scale environment. The extraction of overlapping communities is typically more computationally intensive than disjoint ones, so it requires more sophisticated solutions. The authors proposed multithreaded SLPA algorithm that is almost linearly scalable and provides high quality overlapping social communities, which was confirmed on large real social networks.

The paper entitled "Fast Parallel All Subgraph Enumeration Using Multicore Machines" describes computational methods for efficient enumeration of all subgraphs for a given social network graph. Similar to the previous paper, it operates in the parallel, multicore environment in order to enable processing of large social networks. The method proposed makes use of polynomial heuristic for subgraph isomorphism detection to prune candidate subgraphs and reduce necessary operations.

The paper presents *Skillrank*, a hybrid method to assess quality and confidence of professional skills in social networks. Experts and their expertise are detected, verified, and ranked using specialized trust metrics. It makes use of various pieces of information available within the professional social network: human relationships and endorsements as well as user profiles. The authors have shown that network-based methods can be very effective in accurate prediction and valuation of human skills.

The paper entitled "Link Prediction Methods and Their Accuracy for Different Social Networks and Network Metrics" investigates the correlation between network metrics and accuracy of various link prediction methods. The authors analysed ten different methods for prediction of existence of a new link in the social network and tried to observe which

of them are more suitable for different network types, that is, real social networks with a certain structural profile.

The paper "On Efficient Link Recommendation in Social Networks Using Actor-Fact Matrices" deals with the same problem as the previous paper, prediction of new links in the social network. The authors claim that the computation quality of link recommendation algorithms significantly depends on the social network representation. They found out, in particular, that the actor-fact matrix appears to be the best model for the link recommendation problem.

Finally, computational comparison of human activities online was performed in the last manuscript entitled "The Comparison of Users Activity on the Example of Polish and American Blogosphere." The authors compared two separate online communities: users of Polish and American blogospheres and found significant quantitative differences, also in the dynamics of human habits.

The readers will enjoy reading this special issue to get exposed to various aspects and applications of SNA as presented by some leading researchers in the field.

Acknowledgments

The work was partially supported by The National Science Centre, decision no. DEC-2013/09/B/ST6/02317, and the European Commission under the 7th Framework Programme, Coordination and Support Action, Grant Agreement no. 316097, the ENGINE project.

Przemyslaw Kazienko
Reda Alhajj
Jaideep Srivastava

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 907547, 11 pages
http://dx.doi.org/10.1155/2015/907547

Research Article

The Comparison of Users Activity on the Example of Polish and American Blogosphere

Anna Zygmunt and Bogdan Gliwa

AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Kraków, Poland

Correspondence should be addressed to Anna Zygmunt; azygmunt@agh.edu.pl

Received 21 March 2014; Accepted 26 November 2014

Academic Editor: Reda Alhajj

Blogs are popular way to express opinions on the Internet. Due to their popularity and their public character blogs attract attention of many researchers. In this paper we compare two national blogospheres (Polish and American) from different angles such as characteristics of messages and interactions, structure of social groups, topics discussed in them, and the influence of real-world events on the behavior of such groups. In our approach we try to combine in advanced manner users activity on both the individual and community level. The comparison reveals some differences and various characters of both portals. Methods for analysis of groups dynamics, users roles, and topics in groups are presented.

1. Introduction

Nowadays a large part of our life has moved to the Internet, particularly to the social media. It is hard to imagine that we stop using them. Willingly or not, we are present in them, even passively searching for sources of information. A large part of the official and unofficial life has moved there. There are various reasons for this situation, but one thing must be said with certainty that this is a process that cannot be stopped. The majority of us are only passively involved in it, treating different types of forms of social media as sources of information, that is, places where one can learn something. But there are also people who participate in social media actively and creatively: expressing their opinions, commenting on others, promoting opinions of others, and so forth. They leave so many "traces" of their activities, which can then be analyzed to find interesting patterns of human life, which can be used in marketing, business, politics, or public security domains.

The social media may take many forms, for example, blogs, forums, media sharing systems, microblogging, social networking, and wikis. Among them, blogs play a special role. The term "blogosphere," first introduced by Brad L. Graham in 1999, should be understood as a term describing all blogs. Observing the development of blogosphere, one can say that they have passed a long way from frivolous diaries to very serious sources of information. Undoubtedly, the reason for this situation has become a development tool for creating blogs, as well as the fact that many important people have discovered that blogs are a very good place to express their opinions and to observe an immediate response to them. It is believed that blogs have become a flywheel for the development of online social networking [1]. Now blogs are used as a communication platform and more and more as of source knowledge. Blogs can be treated as web pages with entries arranged in the reverse order (due to chronology). Such pages can contain text, links, pictures, videos, and so forth.

Blogosphere is an interesting source of data for analysis. It is characterized by (in most cases) high dynamics: posts are often added as well as comments on them; one can analyze the reactions of readers to the posts, both in terms of response speed as well as emotion (sentiment analysis). One can analyze themes of posts and find those that receive the greatest interest (getting the most comments) as well as users who generally write such influential posts. Until recently, the analysis of the processes taking place in blogosphere was the domain of research conducted mainly by psychologists and sociologists. These studies were characterized by carrying out analyses to a limited extent due to problems with data

collection. With the development of technological capabilities allowing for automatic and incremental collection of any amount of data from blogosphere and storing them in huge databases have significantly increased the possible directions of research.

The paper presents a comparison in various aspects of users activity in Polish and American blogosphere.

Generally, to our knowledge, there is no such comprehensive comparative analysis of two blogospheres in such a wide range as we have done. Particular areas of research appear in single studies. In some articles the authors analyze groups in blogosphere (but without taking into account the dynamics of change); others examine influential bloggers or analyze topics of discussion. Our approach assumes broad comparison of two national blogospheres by analyzing the structure of the groups that are formed and continued for a period of time, comparing the roles of users played in both the group and the globe in the whole network, as well as the identification of topics of conversation and the study of reaction time for posts in different blogospheres. Such a global approach to the analysis of the users allows creating much more advanced user profiles, at both the individual and global level, as well as finding user's characteristics that are common to different nationalities, as well as those that differentiate them.

The structure of the paper is as follows. In Section 2, current research directions over blogosphere, as well as a review of research on groups and their dynamics, finding roles, and text analysis are presented. Section 3 contains an overview of our algorithms used for finding stable groups, identifying events, finding roles, and identifying topics of posts and comments. In Section 4 both datasets are described in detail and results are presented and discussed. Section 5 concludes and shows possible directions of future works.

2. Related Work

2.1. Blogosphere: Direction of Research. Blogosphere soon became an interesting research area for psychologists and sociologists. The research methodology was largely based on designing questionnaires and asking questions to a properly selected group of respondents (according to, e.g., demography). The results of the analysis were strictly dependent on the truthfulness of responses and the sample size of blogs, which, due to the need for manual processing, was not big. The most interesting subject of research was to determine why people started a blog and reasons they had for continuing writing. They tried to find differences based on gender and demographics of bloggers.

Initially, these analyses concerned a single nationality. Then blogs belonging to representatives of different nations were analyzed to compare and find out if there were any differences related to cultures diversity. Analyses of individual nationalities concerned tracking changes in the demographics of bloggers or certain groups of bloggers were studied.

The vast majority of authors [2–4] concluded that in general motivations for blogging were the same in all analyzed nationalities (self-expression, social interaction, entertainment, passing time, information, and professional advancement), but they had different priority. In [1] motivations for blogging were linked to identity. In [5], types of characters (extroverts, introverts, etc.), language, and gender were analyzed and their impact on the content and topics discussed on blogs was described.

In [6], the group membership was analyzed, but bloggers indicated which group they belonged to and why. In [1] the authors compared bloggers from different countries and analyzed their habits (e.g., differences in activity depending on time of day). The need for research groups of bloggers and analysis of their dynamics was identified, but no studies were carried out.

Since computer scientists started to be interested in the analysis of blogosphere, research has sped up, because there is a real possibility of automatic data collection from the blogosphere using webcrawlers, saving them to big, effective databases and performing virtually any analysis on such data. So there is no need to develop an experiment, invent questions, and collect responses and analyze only data. Usually all data from the page are collected, such as demographic information, text of posts, comments (as well as information about their authors), links, tags, dates, and all other kinds of available information. Directions of research now are much less related to demography, because such data are usually not available. Because all data are available in database, one can freely invent and change the directions of analysis. Generally, this research can be divided into two directions: structure and content analysis.

One of the directions of the analysis was to use methods of social network analysis [7] to analyze the popularity of bloggers (or posts). In [8, 9] Kleinberg algorithm HITS finding hubs and authorities was used to find top bloggers. A-list blogs of the most read, most quoted, and most number of inbound links from others were used.

In other studies [5], the authors attempted to determine what impact, for example, psychological profiles and gender have on the way of writing on blogs. Methods of text processing were used (large blog corpus was collected) in order to extract topics from the text. On the basis of those topics they attempted to create psychological profiles.

The first approach to find clusters in blogospheres and recognize the structure was in [9]. They observed that blogosphere was "selectively interconnected with dense clusters in parts and blogs minimally connected in local neighberhood *[sic]* or flee-floating individually, constituing *[sic]* the majority." In [10] structure A-list was used to find core structures in six national blogospheres. That model was compared with [11]. Differences in cores structures were explained by cultural differences.

In [12, 13] Chinese and German blogospheres were compared to find differences in the structure of pages, length of comments, and time of reactions. It was observed that, in spite of cultural diversity, blogging services worked in a similar way (Chinese bloggers could do more with design of pages).

In [14] data from Polish blogosphere, discussion on MySpace, YouTube comments, and forums BBC were compared according to the length of comments in words and bytes, and it was concluded that overall lengths were similar.

2.2. Groups in Social Networks. Social network is not a homogeneous structure; it rather consists of areas in which vertices communicate to each other more frequently than with vertices outside given area. Such areas are called groups (communities, module, cluster, and subgroups). There are many methods of finding such groups, which can be overlapped (or not) [15, 16]. Finding groups allow simplifying the complex network or analyzing certain processes in micro- and macroscale. Quality of group can be measured by several parameters indicating its size, durability, or importance, for example, density (ratio of the number of links within the group to the maximum possible number of links), cohesion (ratio of the average strength of links between the members to the average strength of their links with people outside the group), or stability between groups (the ratio of the number of people, present in both groups, to the number of all group members). One of the most popular representatives of algorithms finding overlapping groups is CPM (Clique Percolation Method) [17].

2.3. Group Dynamics. Even though most methods have been developed for static environment, many researchers have recognized the need for better reflecting the dynamic nature of the most social networks (especially coming from social media sites) [18, 19]. For dynamic network analysis the common way is to divide given period of time into smaller units called time slots. Then, in each time slot the static network is analyzed and the groups are extracted. Next step is to determine the transitions between groups from neighboring time slots. For this purpose, Greene et al. [16] used the Jaccard index as a measure describing the similarity of groups (the measure is calculated for each pair of groups from neighboring time slots). The value of this measure above arbitrarily defined threshold level means that one group is continuation of another. Some other measures for obtaining transitions between groups have been proposed in literature [20, 21].

Palla et al. in [22] identified basic events (transitions) that may occur in the life cycle of the group: growth, merging, birth, construction, splitting, and death. They did not give any additional conditions. Asur et al. in [18] introduced formal definitions of five critical events. Gliwa et al. proposed in [20] two additional events and gave formal definitions. In [23] new tool GEVi for context-based graphical analysis of social group dynamics was proposed.

2.4. Roles of Users. In social network analysis there are many definitions of role [24–26]. In social media, *role* can be treated as a set of characteristics that describe behavior of individuals and the interactions among them within a social context [27].

Roles in the literature are often discussed in the context of influences [28]. Agarwal et al. in [29] defined influential bloggers and gave their characteristics and described four types of bloggers: active and influential, inactive but influential, active but noninfluential, and inactive and noninfluential.

A lot of studies relate to certain social media and attempt to define their specific roles [30, 31]. For example, an analysis of the basic SNA measures has been used in several studies to define social roles of *starters* and *followers* in blogosphere [32, 33]. *Starters* receive messages mostly from people who are well connected to each other, and therefore they can be identified by low in-degree, high out-degree, and high clustering coefficient in the graph. The distinction between the roles is obtained by combining the difference between the number of in-links and out-links of their blogs.

2.5. Text Mining in Domain of Social Networks. Aggarwal and Wang in [34] provided overview of text mining methods useful for social networks analysis, but in literature text mining combined with SNA is used mostly in some specific cases. Bodendorf and Kaiser in [35] used text mining to extract opinions from texts and then integrated such information with social network analysis approach to find opinion leaders and detect trends in communities. Bartal et al. [36] proposed a method for predicting links in a network based on social network analysis and text data mining approach.

Topic modeling [37] is a statistical technique that uncovers abstract "topics" that can be found in a collection of documents. "Topic" can be defined as a set of words that tend to cooccur in multiple documents, and, therefore, they are expected to have similar semantics. One of the main benefits of this method is that similar texts can be discovered even if they use different vocabulary. One of the most popular methods in topic modeling is Latent Dirichlet Allocation (LDA) [38]. In [39] the authors showed usefulness of topic modeling to analysis of groups dynamics in social networks in blogosphere. Another approach using topic modeling along with social network analysis is presented in [40] where authors track topics in time and automatically assign labels for topics.

3. Methods Used during the Comparison of Different Blogospheres

In this section we describe measures and methods applied to comparison of two blogospheres: American and Polish one. Firstly, we provide definitions of measures utilized to assess different characteristics. Next, we depict methods for analysis of groups dynamics, users roles, and topics in groups.

3.1. Lifetime of a Post. The lifetime lt of a post p can be defined as

$$\mathrm{lt}_p = \max_i \left(t_{c_i}\right) - t_p, \tag{1}$$

where t_p is the date when post p was published and t_{c_i} are dates of comments in the thread of post p.

In other words, lifetime of a post is the range of time between writing the post and the last comment for that post.

3.2. Reaction Time for a Post. The reaction time rt for a post p can be formalized in the following way (symbols used in the definition were explained above):

$$\mathrm{rt}_p = \min_i \left(t_{c_i}\right) - t_p. \tag{2}$$

Reaction time for a post is the range of time between writing the post and the first comment for that post.

3.3. Groups Dynamics. To analyse groups dynamics, whole range of time was divided into smaller periods of time (called later *time slots*). Next, in each time slot, the static network was analysed and the groups were extracted. To identify events between groups from the neighbouring time slots SGCI method [20, 41] was employed, which consists of the following stages: identification of short-lived groups in each time slot, identification of group continuation, separation of the stable groups (lasting for a certain time interval), and the identification of types of group changes (transition between the states of the stable group).

Identification of continuation between groups A and B (from neighbouring time slots) is performed using MJ measure

$$MJ(A,B) = \begin{cases} 0, & \text{if } A = \emptyset \vee B = \emptyset, \\ \max\left(\frac{|A \cap B|}{|A|}, \frac{|A \cap B|}{|B|}\right), & \text{otherwise.} \end{cases} \quad (3)$$

And if the calculated value is above predefined threshold th (in experiments we set th = 0.5) and the ratio of groups size

$$ds(A,B) = \max\left(\frac{|A|}{|B|}, \frac{|B|}{|A|}\right) \quad (4)$$

is below predefined threshold mh (in tests $mh = 50$), then we assumed that group B is a continuation of group A.

Using above measures we can define transition $t_{g_{i,k},g_{i+1,l}}$ between group g_k in ith slot and group g_l in $(i+1)$th time slot as

$$t_{g_{i,k},g_{i+1,l}} : \exists g_{i,k} \wedge \exists g_{i+1,l} \wedge MJ(g_{i,k}, g_{i+1,l}) \geq \text{th} \wedge ds(g_{i,k}, g_{i+1,l}) < mh. \quad (5)$$

Now we can label transitions:

(i) addition: when a small group attaches to big one

$$t_{g_{i,k},g_{i+1,l}} : \frac{|g_{i+1,l}|}{|g_{i,k}|} \geq sh, \quad (6)$$

(ii) deletion: when a small group detached from big one

$$t_{g_{i,k},g_{i+1,l}} : \frac{|g_{i,k}|}{|g_{i+1,l}|} \geq sh, \quad (7)$$

(iii) merge: when many groups join together into bigger one

$$t_{g_{i,k},g_{i+1,l}} : ds(g_{i,k}, g_{i+1,l}) < sh \wedge \left[\exists t_{g_{i,m},g_{i+1,l}} : m \neq k \wedge ds(g_{i,m}, g_{i+1,l}) < sh\right] \wedge \left[\nexists t_{g_{i,k},g_{i+1,n}} : n \neq l \wedge ds(g_{i,k}, g_{i+1,n}) < sh\right], \quad (8)$$

(iv) split: when group divides into 2 or more groups in the next time slot

$$t_{g_{i,k},g_{i+1,l}} : ds(g_{i,k}, g_{i+1,l}) < sh \wedge \left[\exists t_{g_{i,k},g_{i+1,n}} : n \neq l \wedge ds(g_{i,k}, g_{i+1,n}) < sh\right] \wedge \left[\nexists t_{g_{i,m},g_{i+1,l}} : m \neq k \wedge ds(g_{i,m}, g_{i+1,l}) < sh\right], \quad (9)$$

(v) split_merge: combination of event *merge* and *split* for the same transition

$$t_{g_{i,k},g_{i+1,l}} : ds(g_{i,k}, g_{i+1,l}) < sh \wedge \left[\exists t_{g_{i,m},g_{i+1,l}} : m \neq k \wedge ds(g_{i,m}, g_{i+1,l}) < sh\right] \wedge \left[\exists t_{g_{i,k},g_{i+1,n}} : n \neq l \wedge ds(g_{i,k}, g_{i+1,n}) < sh\right], \quad (10)$$

(vi) constancy: simple continuation of a group without significant change of size

$$t_{g_{i,k},g_{i+1,l}} : \frac{\text{abs}(|g_{i,k}| - |g_{i+1,l}|)}{|g_{i,k}|} \leq dh \wedge \left[\nexists t_{g_{i,m},g_{i+1,l}} : m \neq k \wedge ds(g_{i,m}, g_{i+1,l}) < sh\right] \wedge \left[\nexists t_{g_{i,k},g_{i+1,n}} : n \neq l \wedge ds(g_{i,k}, g_{i+1,n}) < sh\right], \quad (11)$$

(vii) change_size: simple continuation of a group with significant change of size

$$t_{g_{i,k},g_{i+1,l}} : \frac{\text{abs}(|g_{i,k}| - |g_{i+1,l}|)}{|g_{i,k}|} > dh \wedge \left[\nexists t_{g_{i,m},g_{i+1,l}} : m \neq k \wedge ds(g_{i,m}, g_{i+1,l}) < sh\right] \wedge \left[\nexists t_{g_{i,k},g_{i+1,n}} : n \neq l \wedge ds(g_{i,k}, g_{i+1,n}) < sh\right], \quad (12)$$

(viii) decay: when a group disappear in the next time slot

$$\nexists t_{g_{i,k},g_{i+1,l}}. \quad (13)$$

In above definitions we used function abs which means absolute value function and some parameters: sh, threshold for ratio of groups size and dh, threshold for groups size differences. In experiments we set value of sh to 10 and value of dh to 0.05.

3.4. Roles of Users. Users can play different roles on a global level and different ones in each of the groups they belong to (local level of roles). The set of roles we use for analysis in this paper was proposed by us in [42].

The presented roles take into consideration responses from other users on the content the user writes (in both the form of posts and comments). To meet such assumptions, we defined *Post* and *Comment Influence*.

Post Influence for author a has the following form (in this definition we use the notation $c(X, \text{cond})$ that means

the number of elements in X that every element of X fulfills condition cond):

$$\begin{aligned}\text{PostInf}_a = {} & 4 \cdot c\left(p_a, pr \geq A_1\right) + 2 \cdot c\left(p_a, pr \geq A_2\right) \\ & + c\left(p_a, pr \geq A_3\right) - c\left(p_a, pr < A_4\right) \\ & - 2 \cdot c\left(p_a, pr < A_5\right) - 4 \cdot c\left(p_a, pr < A_6\right),\end{aligned} \tag{14}$$

where p_a is the posts of author a; pr is the number of comments for a given post excluding the author's comments in his own thread; for global roles we set the following values: $A_1 = 50$, $A_2 = 25$, $A_3 = 10$, $A_4 = 2$, $A_5 = 1$, and $A_6 = 0$; for local roles we set the following values: $A_1 = 10 \cdot B$, $A_2 = 0.25 \cdot A_1$, $A_3 = 0.25 \cdot A_2$, $A_4 = A_5 = 0$, $A_6 = 1$, and B = group Density · group Size.

Comment Influence for author a is calculated in the following way (in this definition we use the notation w(cond) that returns 1 when the condition cond is satisfied, otherwise—0):

$$\begin{aligned}\text{ComInf}_a = {} & 4 \cdot w\left(r_a \geq 1.25\right) + 2 \cdot w\left(r_a \geq 1\right) \\ & + w\left(r_a \geq 0.75\right) - w\left(cr_a < C_1\right) - 2w\left(cr_a < C_2\right) \\ & - 4 \cdot w\left(cr_a < C_3\right),\end{aligned} \tag{15}$$

where r is the number of received comments from other users divided by the number of written comments by given authors; cr is the number of received comments from other users; for global roles we set the following values: $C_1 = 50$, $C_2 = 20$, and $C_3 = 10$; for local roles we set the following values: $C_1 = 0.5 \cdot B$, $C_2 = 0.25 \cdot C_1$, $C_3 = 0.25 \cdot C_2$, and B = group Size · group Density.

Using the above definitions we can describe the set of roles:

(1) Influential User (infUser): PostInf > 2 and ComInf > 0,

(2) Influential Blogger (infBlog): PostInf > 2 and ComInf ≤ 0,

(3) Influential Commentator (infComm): ComInf > 0 and PostInf ≤ 2,

(4) Standard Commentator (comm): c(comments) ≥ 20 and c(posts) ≤ 2,

(5) Not Active (notActive): c(posts) < 1 and c(comments) < 2,

(6) Standard Blogger (stdBlog): user that does not match any from above roles.

3.5. Topics in Groups. Topics for groups were assigned based on clusters uncovered by LDA method. The method for analysis topics in groups was used by us in [23, 39].

Whole method can be described as a set of the following steps. Firstly, we used LDA method provided by mallet tool (http://mallet.cs.umass.edu/) for all posts and the method discovered 350 clusters of words. Next, we manually annotated each cluster by set of topics and joined similar clusters into bigger ones. After that operation, we infer in every comment a set of topics that are referenced by this comment (the network is being built based on writing comments in response to other messages—precise way of building network for each dataset is described in Section 4.1). We consider 2 variants of the method (in results referred to as *method 1* and *method 2*) which differ only in a way of assigning a topic for a comment when LDA could not find any matching topics. The first variant (*method 1*) does not assign any topic if it could not be inferred for given comment, but the second variant (*method 2*) in such case uses topics assigned for the parent comment (if the analysed comment has the parent one and the parent comment has any assigned topics) or the post in the thread where the comment was written. Next step is to assign for the group a set of topics discussed by members of this group (we required that topic should be present in at least 5% of all interactions inside a group to assign such topic for the group).

We can formalize it in the following way. Let us define T as a set of topics (after operation of annotating and joining similar clusters from LDA):

$$T = \{t_1 \cdots t_k\}, \tag{16}$$

members of a group G

$$\text{members}\,(G) = \{a_1 \cdots a_n\}, \tag{17}$$

edges in a group G

$$\text{edges}\,(G) = \left\{e_{xy} : x \in \text{members}\,(G) \wedge y \in \text{members}\,(G)\right\}, \tag{18}$$

topics for edge e_{xy}

$$\text{topics}\left(e_{xy}\right) = \{t_k\} \wedge \text{topics}\left(e_{xy}\right) \subset T. \tag{19}$$

Using above notation we can define topics for a group G

$$\begin{aligned}&\text{topics}\,(G) \\ &= \left\{t_k : \forall_k \exists_x \exists_y \left[e_{xy} \in \text{edges}\,(G) \wedge t_k \in \text{topics}\left(e_{xy}\right)\right]\right. \\ &\quad \left. \wedge \forall_k \frac{t_k}{\sum t_k} \geq h\right\},\end{aligned} \tag{20}$$

where h is a threshold and we used $h = 0.05$.

4. Results

In this section we compare Polish and American blogosphere from different points of view, especially in terms of users activity, groups formation, and topics discussed by users in groups. For this purpose, we chose one dataset as a representative for Polish blogosphere and one for American one.

Table 1: Comparison of data quantity in both datasets.

Measure	*Salon24*	*Huffington Post*
Number of posts	380 700	414 225
Number of posts without comments	74 979 (19.7%)	45 604 (11%)
Average number of comments in one post	18.65	48.28
Number of comments	5 703 140	17 796 819
Number of comments to posts	2 781 303 (48.77%)	6 961 369 (39.12%)
Number of comments to other comments	2 921 837 (51.23%)	10 753 162 (60.88%)
Number of authors	31 750	680 341
Number of authors of posts	10 131 (31.91%)	1 027 (0.15%)
Number of authors of comments	29 536 (93.03%)	661 676 (97.26%)

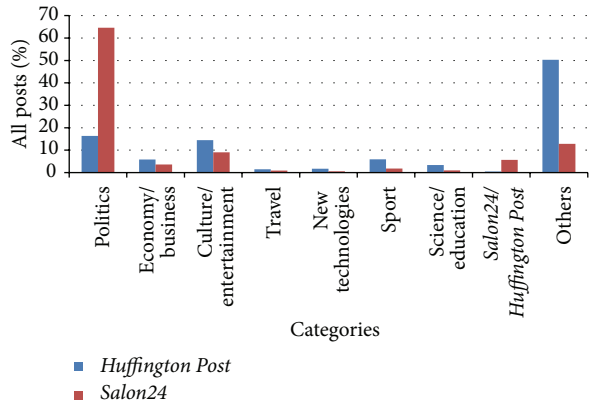


Figure 1: Categories of posts.

4.1. Datasets Description. The first dataset contains data from the portal *Salon24* (http://www.salon24.pl/) (Polish blogosphere). This portal comprises blogs from different subjects, but political ones constitute the largest part of them (as you can see in Figure 1). The data from this dataset is from time range 1.01.2008–6.07.2013. Whole period of time was divided into overlapping time slots, each lasting 7 days and the neighbouring slots overlap each other by 4 days. After this operation the dataset contains 504 slots. In every time slot a static network is built according to *comments model* introduced in [43]; that is, the users are nodes and relations between them are built in the following way: from user who wrote the comment to the user who was commented on or, if the user whose comment was commented on is not explicitly referenced in the comment (by using @ and name of author of comment), the target of the relation is the author of post.

The second dataset is the *Huffington Post* dataset (http://www.huffingtonpost.com/) (American blogosphere) which contains news and blogs from various subjects (we can see in Figure 1 that political topics constitute significant part of all posts, but this topic does not outnumber other ones as it was in the case of *Salon24*). This dataset contains data from period 1.01.2010–14.11.2013. Similarly as for Salon24 dataset, the whole period of time was divided into overlapping time slots, each lasting 7 days with overlap equal to 4 days, which produced 442 slots (but for the analysis we used slots in this dataset starting from 97 because in previous one there were some slots where groups were not found). In *Huffington* dataset networks in time slots are built in similar way as for *Salon24* dataset (edges between an author of given comment and an author of a comment the response is addressed for, or, if a comment is not an answer for another comment, between an author of given comment and an author of a post), but in this case the explicit references between comments exist (hierarchical structure of comments).

Moreover, due to the performance issues of group extraction method in order to detect communities, we eliminated the edges with weight equal to one in each time slot. But for other types of analyses (such as role finding) we conducted them on full graphs without any edge removal.

4.2. Basic Statistics. As we can observe in Table 1 the *Huffington Post* dataset is bigger than *Salon24* one. Threads in *Huffington Post* are also longer; that is, on average posts have more comments in *Huffington Post* than in *Salon24*. We can see that in both datasets the responses to other comments represent a substantial part of all comments. Another interesting fact is that authors of posts in *Huffington Post* constitute much smaller fraction of all authors (less than 1%) as compared with *Salon24* (almost 32%). This means that character of both portals is quite different. In *Salon24* a significant number of users have a contribution to creating posts and informing about new events from the world, but in *Huffington Post* the users are oriented towards commenting on posts and this portal plays a role more similar to an Internet newspaper.

4.3. Lifetime of Posts. Figure 2 presents lifetime of posts (it is a cumulative chart so it depicts percentage of all posts that have lifetime equal or less than specified value). We can see that almost 90% of posts in *Salon24* have their lifetime up to 1 week, but similar lifetime in *Huffington Post* is achieved after 2 months (8 weeks). This means that in *Salon24* posts older than 1-2 weeks are rarely commented on and the attention of users is brought mostly by new posts, which is a bit different than in *Huffington Post* where significant part of users comments also on older posts than 1 week. Such a difference in lifetime of posts between these 2 datasets also emphasizes higher dynamics in *Salon24*.

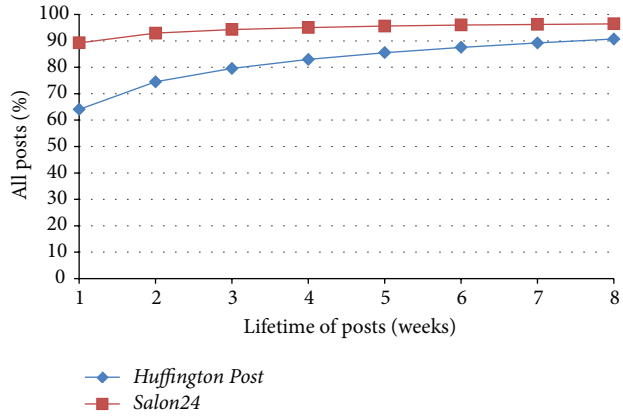


Figure 2: Lifetime for posts.

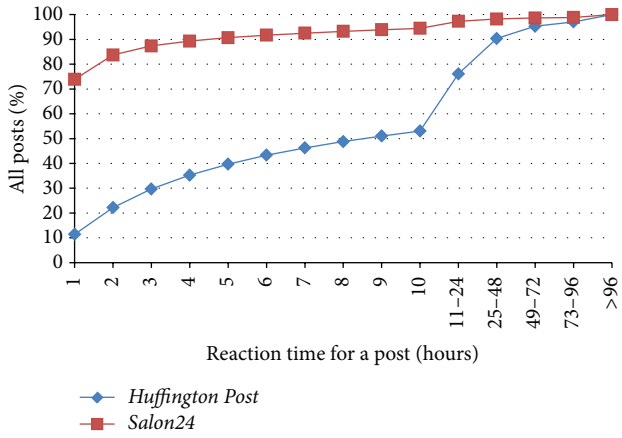


Figure 3: Reaction time for a post.

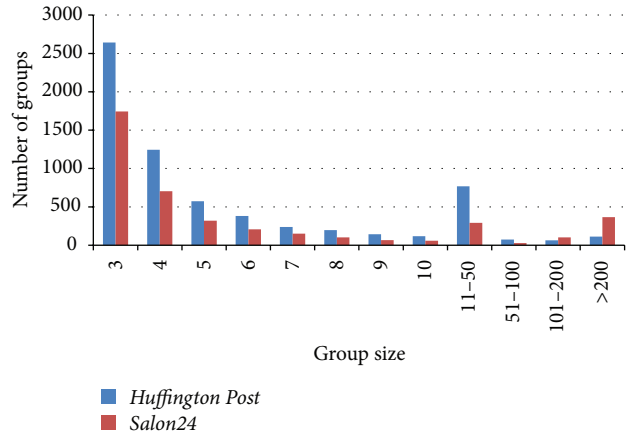


Figure 4: Number of stable groups at given size.

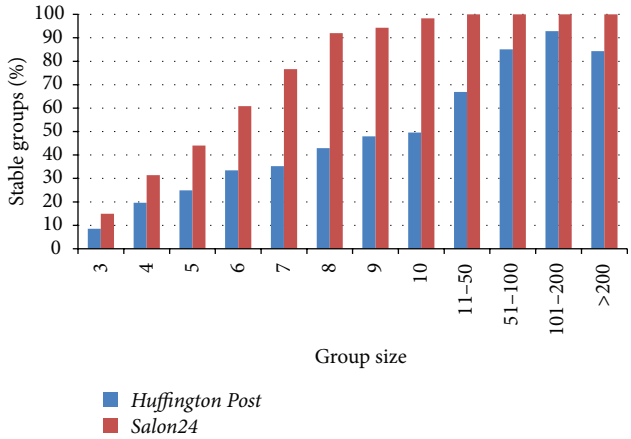


Figure 5: Percentage of stable groups in relation to all groups.

4.4. Reaction Time for Posts. Figure 3 depicts reaction times for a post in both datasets. One can notice a big difference in dynamics between *Huffington Post* and *Salon24*—in the first hour after publishing a post in *Salon24* 73.9% of all posts received at least one comment, but in *Huffington Post* only 11.4% of all posts. After 2 days after writing a post, in both blog portals more than 90% of posts were commented on. We also investigated the amount of time needed for a half of all posts to get the first comment. For *Huffington Post* we need about 8 hours, but in *Salon24* it is sufficient to wait only 32 minutes after writing a post to receive a comment.

4.5. Groups and Their Dynamics. For group extraction we used CPM method (CPMd version which is designed to discover groups in directed networks) from CFinder (http://www.cfinder.org/) tool for k equals 3.

Figure 4 presents number of stable groups with their size in both datasets. One can notice that *Huffington Post* contains more groups overall. Moreover, the mentioned dataset comprises more small and medium size groups, but *Salon24* has more big groups.

In Figure 5 we can see the fraction of stable groups in relation to all groups. Stable groups have additional restriction that they have to be present in at least given number (in experiments we used value 3 due to the fact that the presence in 2 time slots is not hard to achieve because slots are overlapping) of time slots. One can observe that the lowest fraction of groups is stable for groups with small size and increases with group size. Furthermore, we can notice that *Salon24* has higher fraction of stable groups than *Huffington Post*.

Figure 6 depicts number of evolution events in both datasets. *Huffington Post* includes a large amount of medium size groups, so there are more events related to joining and dividing groups with similar size (i.e., *merge*, *split* events). Conversely, *Salon24* contains a relatively large number of huge groups, so in this dataset the events related to joining and dividing groups with substantial difference of size, that is, *addition*, *deletion* events, dominate over ones with similar size.

4.6. Reaction for Real-World Events. Figures 7 and 8 present number of groups and evolution events for *Huffington Post* and *Salon24* with marking key events from real world. One can notice some correlation between peaks on these charts

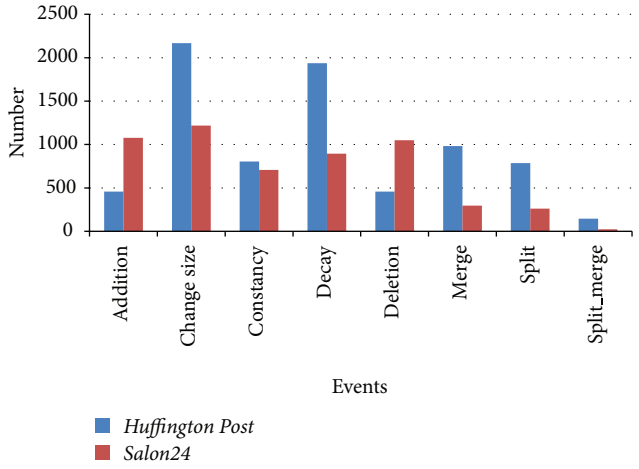


Figure 6: Number of events.

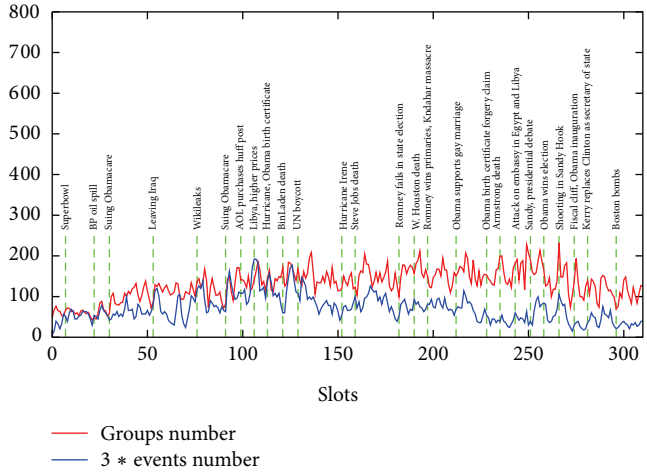


Figure 7: Number of groups and evolution events in time and correlation with real-world events for *Huffington Post*.

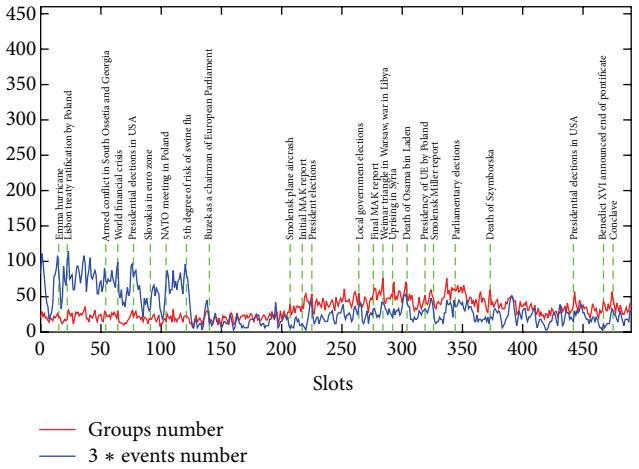


Figure 8: Number of groups and evolution events in time and correlation with real-world events for *Salon24*.

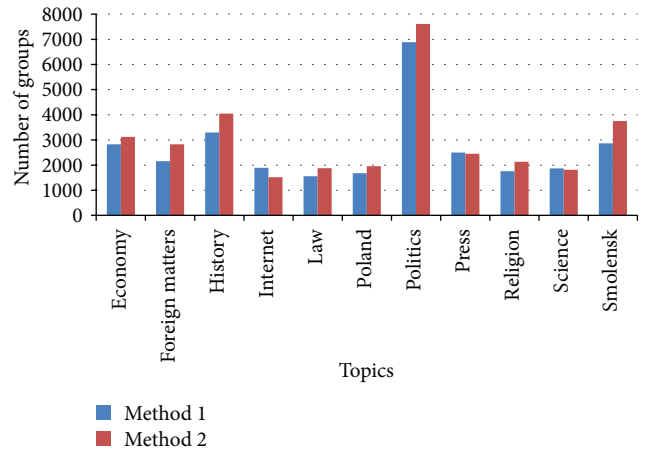


Figure 9: Topics discussed in at least 10% of all groups in *Salon24*.

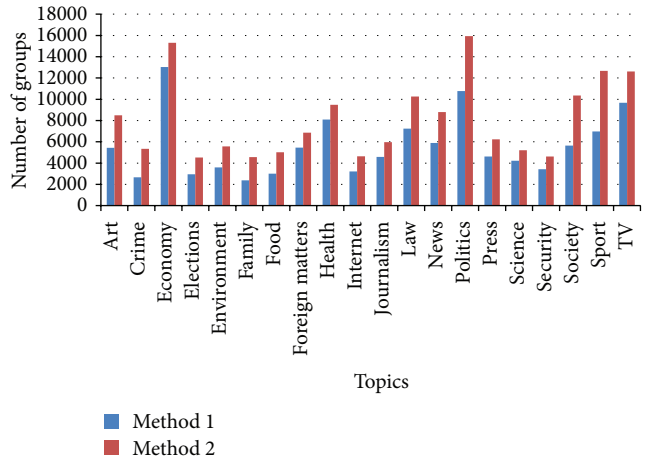


Figure 10: Topics discussed in at least 10% of all groups in *Huffington Post*.

and mentioned events. It means that blog portals are a kind of mirror that reflects actual events from real world and such events influence on groups in blogosphere to a large degree.

4.7. Topics in Groups. Figures 9 and 10 describe most popular topics discussed in groups in both blogospheres. Each chart presents topics being present in at least 10% of all groups. We used 2 methods to assess topics in groups, both described in Section 3.5. The motivation for introducing the second method was to determine topics for larger number of groups (e.g., in *Huffington Post* using the first method we assigned topics for about 75% of all groups and using the second method we assigned topics for about 93% of all groups).

One can notice that *Huffington Post* contains more different topics, but in *Salon24* one can observe that topics related to *politics* are dominating. Another interesting thing is the topic of *Smolensk* which appears frequent in groups in *Salon24* and it concerns Polish President airplane crash in Smolensk (10.04.2010) and other events related to investigation of this catastrophe.

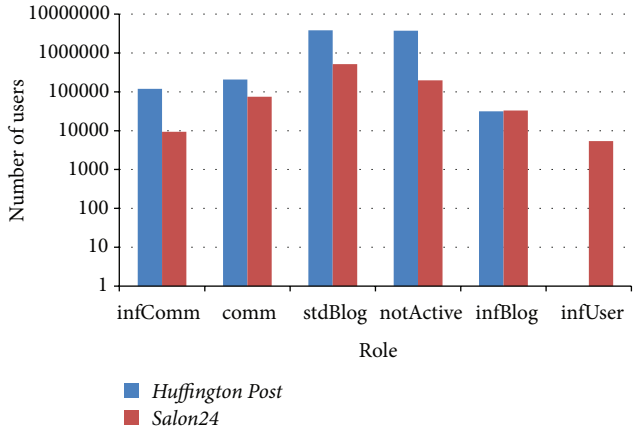


Figure 11: Global roles of users.

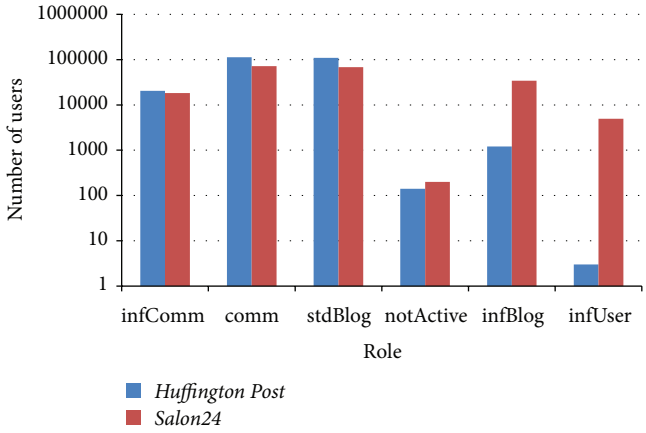


Figure 12: Local roles of users.

When we look into results of both methods to associate topics for groups, we can spot that they are quite similar (in terms of proportions for different topics).

4.8. Global and Local Roles of Users. Figures 11 and 12 show number of users with global and local roles (roles on the level of a group), respectively. For global roles, we can notice that in *Huffington Post* users with a role of *Influential User* (users with this role write influential posts and influential comments) almost do not exist (there is only one person with such a role), which is very different from *Salon24*. This difference can be explained by various types of nature of these portals—in American portal there is very small fraction of authors of posts and they rarely write any comments.

As far as local roles are concerned, one can notice a few interesting observations. Firstly, the number of inactive users is much lower than in previous case—this means that most inactive users (actually, the conditions in experiments let them write no more than one comment) are outside groups which is understandable. Moreover, the number of *Influential Bloggers* and *Influential Users* is smaller in American portal than in Polish one. The difference has its roots in different nature of portals (as we explained above) and the fact that in *Huffington Post* the responses to a post constitute a smaller fraction of all responses in comparison with *Salon24* (which can be seen in Table 1).

5. Conclusion

In the paper, a comparative analysis of two different blogospheres, Polish and American, is presented. This approach is based on a comprehensive analysis of the structure and content of blogosphere.

The preliminary analysis of the structure of both blogospheres shows that discussions conducted in *Salon24* are much more intense: generally the first comment appears much more quickly, but the lifetime of the post is much shorter than in *Huffington Post*. Discussions in *Huffington Post* are much more stable. The structure of groups is different: in *Huffington Post* there are smaller groups of comparable size, which is the reason why there are more events *split* and *merge* (characteristic of groups of similar size). In *Salon24* there is a greater variation in the group size and thus different events dominate (*deletion*, *addition*).

Differences in the number of these groups are significant: in *Huffington Post* there are three times more groups than in *Salon24*. A probable reason for this is a considerable difference in the ratio of the number of posts to the number of comments: in *Huffington Post* most people write comments, but very few write posts (for *Salon24* the situation is different).

In turn, events have a big impact on the dynamics of both blogospheres. Due to the different nature of *Huffington Post*, where few people write posts and most comments, some roles, which are in *Salon24*, in *Huffington Post* are not present. As far as topics discussed in groups are considered, *Salon24* is more oriented on topics related to politics, but *Huffington Post* is more diverse.

So, the comparison of two blogospheres gave interesting results: in some aspects nationality does not matter but sometimes has a big impact on user behavior. One can see differences in the characteristics of people from different countries in the context of their activity in the social media (taking into account their dynamic nature), for example, categories of interesting topics, speed of reaction to novelty, and way of reaction according to the categories of the world events. Presenting approach may have many practical applications. It can, for example, support sociologists and psychologists in their research on behavioral analysis in different national communities (e.g., among emigrants). The results of our experiments show that, for example, in marketing, making user profiles, one should take into account nationality, and therefore product marketing campaigns should be differentiated depending on countries (e.g., global advertising campaign). Similarly, to predict customer behavior, one should take into account the context of nationalities. These observations can be used in the development of election campaigns.

Research can be continued in several ways. One of them is analyzing and comparing differences in sentiment, for example, which nation is more optimistic? Another direction of research could be comparing the ability to predict the future

of groups in both blogospheres. Furthermore, extension of comparison to other national blogospheres possibly could reveal some characteristics related to their nationality.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The research reported in the paper was partially supported by the Grants nos. INNOTECH-K2/IN2/89/182461/NCBR/13 and 008/R/ID1/2011/01 from the Polish National Centre for Research and Development.

References

[1] M. Kobayashi, "Blogging around the globe: mitivations, privacy concerns, and social networking," in *Computational Social Networks: Security and Privacy*, A. Abraham, Ed., chapter 3, pp. 55–86, Springer, London, UK, 2012.

[2] S. Penderson, *Why Blog?: Motivations for Blogging*, Woodhead, Cambridge, UK, 2010.

[3] B. A. Nardi, D. J. Schiano, M. Gumbrecht, and L. Swartz, "Why we blog," *Communications of the ACM*, vol. 47, no. 12, pp. 41–46, 2004.

[4] K. D. Trammell, A. Tarkowski, J. Hofmokl, and A. M. Sapp, "Rzeczpospolita blogów [Republic of Blog]: examining polish bloggers through content analysis," *Journal of Computer-Mediated Communication*, vol. 11, no. 3, pp. 702–722, 2006.

[5] A. J. Gill, S. Nowson, and J. Oberlander, "What are they blogging about? Personality, topic and motivation in blogs," in *Proceedings of the 3rd AAAI International ICWSM Conference*, E. Adar, M. Hurst, T. Finin, N. S. Glance, N. Nicolov, and B. L. Tseng, Eds., The AAAI Press, San Jose, Calif, USA, May 2009.

[6] M. Taki, *Bloggers and the blogosphere in lebanon & syria meanings and activities [Ph.D. thesis]*, University of Westminster, London, UK, 2010.

[7] P. J. Carrington, J. Scott, and S. Wasserman, Eds., *Models and Methods in Social Network Analysis*, Cambridge University Press, Cambridge, UK, 2005.

[8] D. Obradovic and S. Baumann, "Identifying and analysing Germany's top blogs," in *KI 2008: Advances in Artificial Intelligence*, A. Dengel, K. Berns, T. M. Breuel, F. Bomarius, and T. Roth-Berghofer, Eds., vol. 5243 of *Lecture Notes in Computer Science*, pp. 111–118, Springer, Berlin, Germany, 2008.

[9] S. C. Herring, I. Kouper, J. C. Paolillo et al., "Conversations in the blogosphere: an analysis 'from the bottom up,'" in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS '05)*, vol. 4, p. 107.2, IEEE Computer Society, Washington, DC, USA, January 2005.

[10] D. Obradovic and S. Baumann, "A journey to the core of the blogosphere," in *International Conference on Advances in Social Network Analysis and Mining (ASONAM '09)*, N. Memon and R. Alhajj, Eds., pp. 1–6, IEEE Computer Society, 2009.

[11] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social Networks*, vol. 21, no. 4, pp. 375–395, 2000.

[12] H. Yilin, F. Caroli, and T. Mandl, "The Chinese and the German blogosphere: an empirical and comparative analysis," in *Mensch & Computer*, T. Gross, Ed., pp. 149–158, Oldenbourg, 2007.

[13] T. Mandl, "Comparing chinese and german blogs," in *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia (HT '09)*, pp. 299–308, ACM, New York, NY, USA, July 2009.

[14] P. Sobkowicz, M. Thelwall, K. Buckley, G. Paltoglou, and A. Sobkowicz, "Lognormal distributions of user post lengths in Internet discussions—a consequence of the Weber-Fechner law?" *EPJ Data Science*, vol. 2, no. 1, article 2, 2013.

[15] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[16] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM '10)*, pp. 176–183, IEEE Computer Society, Washington, DC, USA, 2010.

[17] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[18] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 4, article 16, 2009.

[19] M. Spiliopoulou, "Evolution in social networks: a survey," in *Social Network Data Analytics*, C. C. Aggarwal, Ed., pp. 149–175, Springer, New York, NY, USA, 2011.

[20] B. Gliwa, S. Saganowski, A. Zygmunt, P. Bródka, P. Kazienko, and J. Koźlak, "Identification of group changes in blogosphere," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '12)*, pp. 1201–1206, Istanbul, Turkey, August 2012.

[21] P. Bródka, S. Saganowski, and P. Kazienko, "GED: the method for group evolution discovery in social networks," *Social Network Analysis and Mining*, vol. 3, no. 1, pp. 1–14, 2013.

[22] G. Palla, A.-L. Barabási, and T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, no. 7136, pp. 664–667, 2007.

[23] B. Gliwa, A. Zygmunt, and A. Byrski, "Graphical analysis of social group dynamics," in *Proceedings of the 4th International Conference on Computational Aspects of Social Networks (CASoN '12)*, pp. 41–46, IEEE, November 2012.

[24] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, Cambridge,UK, 1994.

[25] E. Gleave, H. T. Welser, T. M. Lento, and M. A. Smith, "A conceptual and operational definition of 'social role' in online community," in *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS '09)*, pp. 1–11, IEEE Computer Society, January 2009.

[26] H. T. Welser, D. Cosley, G. Kossinets et al., "Finding social roles in wikipedia," in *Proceedings of the iConference (iConference '11)*, pp. 122–129, ACM, New York, NY, USA, 2011.

[27] V. Junquero-Trabado and D. Dominguez-Sal, "Building a role search engine for social media," in *Proceedings of the 21st International Conference Companion on World Wide Web (WWW '12)*, A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, Eds., pp. 1051–1060, ACM, 2012.

[28] E. Keller and J. Berry, *One American in Ten Tells the Other Nine How to Vote, Where to Eat and, What to Buy*, The Free Press, New York, NY, USA, 2003.

[29] N. Agarwal, H. Liu, L. Tang, and P. S. Yu, "Modeling blogger influence in a community," *Social Network Analysis and Mining*, vol. 2, no. 2, pp. 139–162, 2012.

[30] R. D. Nolker and L. Zhou, "Social computing and weighting to identify member roles in online communities," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 87–93, September 2005.

[31] A. Zygmunt, "Role identification of social networkers," in *Encyclopediaof Social Network Analysis and Mining*, R. Alhajj and J. Rokne, Eds., pp. 1598–1606, Springer, New York, NY, USA, 2014.

[32] D. L. Hansen, B. Shneiderman, and M. A. Smith, "Visualizing threaded conversation networks: mining message boards and email lists for actionable insights," in *Active Media Technology*, A. An, P. Lingras, S. Petty, and R. Huang, Eds., vol. 6335 of *Lecture Notes in Computer Science*, pp. 47–62, Springer, Berlin, Germany, 2010.

[33] M. Mathioudakis and N. Koudas, "Efficient identification of starters and followers in social media," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '09)*, pp. 708–719, ACM, Saint-Petersburg, Russia, March 2009.

[34] C. C. Aggarwal and H. Wang, "Text mining in social networks," in *Social Network Data Analytics*, C. C. Aggarwal, Ed., pp. 353–378, Springer, New York, NY, USA, 2011.

[35] F. Bodendorf and C. Kaiser, "Detecting opinion leaders and trends in online communities," in *Proceedings of the 4th International Conference on Digital Society (ICDS '10)*, L. Berntzen, F. Bodendorf, E. Lawrence, M. Perry, and S. Smedberg, Eds., pp. 124–129, February 2010.

[36] A. Bartal, E. Sasson, and G. Ravid, "Predicting links in social networks using text mining and SNA," in *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM '09)*, pp. 131–136, IEEE Computer Society, Washington, DC, USA, July 2009.

[37] Y. Huang, "Support vector machines for text categorization based on latent semantic indexing," Tech. Rep., Electrical and Computer Engineering Department, The Johns Hopkins University, 2003.

[38] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[39] B. Gliwa, A. Zygmunt, and S. Podgorski, "Incorporating text analysis into evolution of social groups in blogosphere," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '13)*, pp. 931–938, Krakow, Poland, September 2013.

[40] M. Nguyen, T. Ho, and P. Do, "Social networks analysis based on topic modeling," in *Proceedings of the IEEE RIVF International Conference on Computing and Communication Technologies: Research, Innovation, and Vision for Future (RIVF '13)*, pp. 119–122, November 2013.

[41] A. Zygmunt, P. Bródka, P. Kazienko, and J. Koźlak, "Key person analysis in social communities within the blogosphere," *Journal of Universal Computer Science*, vol. 18, no. 4, pp. 577–597, 2012.

[42] B. Gliwa, A. Zygmunt, and J. Koźlak, "Analysis of roles and groups in blogosphere," in *Proceedings of the 8th International Conference on Computer Recognition Systems (CORES '13)*, vol. 226 of *Advances in Intelligent Systems and Computing*, pp. 299–308, Springer, Cham, Switzerland, 2013.

[43] B. Gliwa, J. Koźlak, A. Zygmunt, and K. Cetnarowicz, "Models of social groupsin blogosphere based on information about comment addressees and sentiments," in *Proceedings of the 4th International Conference on Social Informatics*, vol. 7710 of *Lecture Notes in Computer Science*, pp. 475–488, Springer, Lausanne, Switzerland, 2012.

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 901321, 11 pages
http://dx.doi.org/10.1155/2015/901321

Research Article

Fast Parallel All-Subgraph Enumeration Using Multicore Machines

Saeed Shahrivari and Saeed Jalili

Computer Engineering Department, Tarbiat Modares University (TMU), Tehran 14115-111, Iran

Correspondence should be addressed to Saeed Jalili; sjalili@modares.ac.ir

Received 28 January 2014; Revised 21 November 2014; Accepted 21 November 2014

Academic Editor: Przemyslaw Kazienko

Enumerating all subgraphs of an input graph is an important task for analyzing complex networks. Valuable information can be extracted about the characteristics of the input graph using all-subgraph enumeration. Notwithstanding, the number of subgraphs grows exponentially with growth of the input graph or by increasing the size of the subgraphs to be enumerated. Hence, all-subgraph enumeration is very time consuming when the size of the subgraphs or the input graph is big. We propose a parallel solution named *Subenum* which in contrast to available solutions can perform much faster. Subenum enumerates subgraphs using edges instead of vertices, and this approach leads to a parallel and load-balanced enumeration algorithm that can have efficient execution on current multicore and multiprocessor machines. Also, Subenum uses a fast heuristic which can effectively accelerate non-isomorphism subgraph enumeration. Subenum can efficiently use external memory, and unlike other subgraph enumeration methods, it is not associated with the main memory limits of the used machine. Hence, Subenum can handle large input graphs and subgraph sizes that other solutions cannot handle. Several experiments are done using real-world input graphs. Compared to the available solutions, Subenum can enumerate subgraphs several orders of magnitude faster and the experimental results show that the performance of Subenum scales almost linearly by using additional processor cores.

1. Introduction

Enumerating subgraphs of a given size has been shown to be a very useful task in the area of complex network analysis. Subgraphs can be used to identify building blocks and functional and nonfunctional characteristics in social, biological, chemical, and technological graphs [1]. An interesting application is subgraph mining which can be used to extract functional properties. A good example is finding *network motifs*, which are defined as connected subgraphs that occur significantly more frequently than expected [2]. One of the best known approaches for finding network motifs is to enumerate all subgraphs and then extract significant motifs after omitting frequent subgraphs that occur in random networks [3]. There are also many other applications in areas like data mining, statistics, systems biology, chemoinformatics, social networks, telecommunications, and web mining.

Although subgraph enumeration is a useful task, it is a computational challenging problem [4]. Enumeration can be classified into two distinct problems: enumerating all labeled subgraphs and enumerating nonisomorphic subgraphs, that is, subgraphs that have identical structure but different vertex labels. In the first problem, all of the subgraphs of a given size should be enumerated. On the other hand, in the second problem which is much more important, all of the nonisomorphic subgraphs of a given size must be enumerated. Both problems are very time consuming because the number of both labeled and nonisomorphic subgraphs increases exponentially by giving a bigger subgraph size or a larger input graph for subgraph enumeration.

As the size of the input graph increases, the number of subgraphs of size k increases exponentially (in the worst case $C(n, k)$ for a complete graph) [5]. The number of nonisomorphic subgraphs, which can be calculated using the Polya enumeration theorem [6], also increases exponentially as k increases. Therefore, by increasing the subgraphs size or the input graph's size, subgraph enumeration will take more time. When nonisomorphic subgraphs are enumerated, the problem becomes more complicated because an additional mechanism must be used to identify isomorphic subgraphs.

There is no known polynomial algorithm for subgraph isomorphism problem yet, and this overcomplicates the subgraph enumeration problem [7].

Due to the complex nature of subgraph enumeration problem, it is a very challenging and time-consuming problem. Available sequential algorithms tend to take a lot of time to do the job [3]. Hence, a good solution is to use parallel and distributed systems to accelerate subgraph enumeration [8]. Several other recent works targeting parallel subgraph enumeration have been proposed recently [8]. However, most of the related works are based on message passing interface (MPI) and hence are designed to work on cluster computing systems [8, 9]. In contrast, our goal is to provide a fast and easy to use tool for subgraph enumeration on commodity multicore and multiprocessor machines and to the best of our knowledge it has not yet been done. For this reason, we present a parallel solution, named *Subenum,* which is designed for faster and more scalable subgraph enumeration on multicore and multiprocessor machines. Subenum provides fast and efficient methods for counting and dumping both all and just nonisomorphic subgraphs.

Subenum's strength compared to other similar works can be classified into three categories. First, we have presented a new edge-based parallel subgraph enumeration algorithm named *PSE*, which is an improved version of the well-known sequential ESU algorithm. PSE provides a parallel and load-balanced approach for subgraph enumeration. The second strength is using a custom polynomial-time heuristic for detecting isomorphic subgraphs. The last strength is using a combination of external sorting and the nauty canonical labeling algorithm which enables Subenum to enumerate nonisomorphic subgraphs even when the number of subgraphs is so big that they cannot be stored in the main memory.

For evaluating the performance of Subenum we have performed several experiments on real-world graphs from different areas like social network, biological networks, software engineering, and electrical circuits. During the experiments, we compared Subenum's performance to state-of-the-art algorithms and implementations. Experimental results show that Subenum provides a parallel, load-balanced, and effective solution for all-subgraph enumeration problem. Compared to the fastest available tools for nonisomorphic subgraph enumeration, Subenum enumerates subgraphs several times faster and is able to reduce execution time from days to hours. In addition, Subenum is able to handle large graphs and also large subgraph sizes while other solutions fail to handle them.

2. Related Work

Related works for subgraph enumeration can be categorized into three main classes [8]: all-subgraph enumeration, single-subgraph enumeration, and subgraph-set enumeration. In all-subgraph enumeration (our problem), all of the subgraphs of size k of the original graph must be enumerated [1, 3, 4, 10]. Nevertheless, other conditions can also be defined for subgraphs for example, subgraphs of size k that have an Eulerian path. In single-subgraph enumeration, all of the isomorphic subgraphs of a predefined individual subgraph of size k must be enumerated [5]. Finally, in the subgraph-set enumeration, isomorphic subgraphs of a given set of subgraphs of size k must be enumerated [2]. As stated before, our solution is for the first kind of enumeration, that is, all-subgraph enumeration. Hence, we concentrate on related works that enumerate all subgraphs of size k of a given input graph. Interested reader can find deeper discussions in [8, 11–15].

The most notable efforts for all-subgraph enumeration problem are done in the network motif finding problem. As stated before, one of the best known exact approaches for finding network motifs is via all-subgraph enumeration and then counting nonisomorphic subgraphs [3]. The most notable works in this sector are mfinder [1], Kavosh [3], ESU aka FANMOD [4, 14], FPF [10], gtriesScanner [16], FaSe [17], NetMODE [18], and QuateXelero [19]. Note that gtriesScanner and FaSe use the ESU algorithm for subgraph enumeration, but in conjunction with ESU, they use the G-Tries data structure to accelerate subgraph isomorphism detection. Also note that FANMOD is limited to subgraphs smaller than 9 and NetMODE is limited to subgraphs smaller than 7. Compared to this group of related works, our solution has three strengths: parallel execution, using a heuristic (ordered labeling) for subgraph isomorphism, and external memory based isomorphic subgraphs counting.

Since subgraph enumeration is a time-consuming task, some recent works have used cluster computing to tackle the problem. Most of the available works for parallel all-subgraph enumeration are based on MPI. The most notable MPI-based solutions are discussed in [8, 20]. More works are done for parallel single-subgraph enumeration [2, 9, 21, 22]. Some recent works have used the MapReduce programming model [23] and Hadoop [24] for efficient single-subgraph enumeration on cloud and cluster computing systems. The most mentionable works are [25–29]. However, these works are also based on cluster and cloud computing systems. In contrast to available related work, Subenum presents a parallel solution that can boost the speed of all-subgraph enumeration problem using parallel processing capabilities of current commodity multicore and multiprocessor systems which are more accessible than expensive and complex solutions like cluster and parallel computing. There are some other similar but more complex problems like colored subgraph enumeration and motif finding [30, 31], but in order to keep this section short, we skip them. The interested reader can refer to [32] for more information.

3. Preliminaries

In mathematics, a graph is a collection of points that are connected by some links. The points of a graph are called vertices and the links are called edges. In this paper, if we use G to denote a graph, then $V(G)$ is used to present the vertices of G and $E(G)$ is used to present the edges of G. Vertices and edges of a graph can be assigned labels, weights, or colors. However, we assume graphs to be directed, simple, and unweighted. In other words, we assume that just the

vertices take labels and the edges are directed and do not have weights and also there is at most one edge between two vertices.

For a vertex set $V' \subseteq V$ its open neighborhood $N(V')$ is the set of all vertices, $V - V'$, which are adjacent to at least one vertex of V'. For a vertex $v \in V - V'$ its exclusive neighborhood with respect to V' denoted by $N_{excl}(v, V')$ is the set of all vertices neighboring v that do not belong to $V' \cup N(V')$.

The graph H is a subgraph of G, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An induced subgraph of G on the vertices set N denoted by $G[N]$ is a subgraph of G with N as the vertex set containing all edges between vertices of N that are in $E(G)$. When we say that we are enumerating subgraphs of size k of a graph like G we mean that we are enumerating induced subgraphs of G. Two subgraphs G_1 and G_2 are isomorphic if and only if there is a one to one correspondence between their vertices, and there is an edge between two vertices of G_1 if and only if there is an edge between the corresponding vertices in G_2. Actually, there is no polynomial time algorithm for graph isomorphism problem yet [7].

ESU Enumeration Algorithm. The most well-known algorithm for subgraph enumeration is the ESU algorithm [14]. ESU assumes that vertices are labeled by unique integer values. The basic idea of ESU algorithm is to start from each vertex v and enumerate all subgraphs of size k that contain v and vertices that have a bigger label than v, that is, the subgraphs that are v-rooted. ESU enumerates each subgraph just once. Details of the ESU algorithm are given in Algorithm 1.

4. Subenum: A Solution for All-Subgraph Enumeration

The easiest approach for parallel subgraph enumeration is to enumerate subgraphs rooted from each vertex in parallel using the ESU algorithm. However, this approach results unbalanced parallel tasks. Usually, there is a great variance in the number of subgraphs rooted from each vertex because vertices with higher degrees tend to participate in more subgraphs. For example in one of our experiments, more than 20% of subgraphs were enumerated from an identical vertex. Hence, this naïve approach causes unbalanced parallel loads and sometime this unbalanced load can cause inefficient parallelism.

Our idea for parallel enumeration is to enumerate subgraphs containing each edge in parallel. Enumerating subgraphs using edges causes more fine-grained parallel tasks because each vertex will be decomposed into several edges. Hence, the whole process is more load-balanced than vertex-based enumeration. For this purpose we need an algorithm for enumerating all subgraphs of size k that contain a specific edge $e(v, w)$. For this purpose, we have designed *Edge-based Subgraph Enumeration* (*ESE*) algorithm. ESE itself is an extended version of the ESU algorithm. However, in contrast to ESU, ESE is nonrecursive. Details of ESE algorithm are given in Algorithm 2.

Having defined the edge-based enumeration algorithm, we can explain *Parallel Subgraph Enumeration* (*PSE*) algorithm which uses the ESE algorithm as a building block. The procedure of PSE is simple. First, we put all of the edges of the input graph into a shared queue (for the case of bidirectional edges we put just one of them). Then, we use p concurrent threads to pick edges from the shared queue and enumerate subgraphs of each edge using p instances of ESE algorithm in parallel. The shared queue of edges between threads causes a more load-balanced parallelism. A more formal description of PSE is given in Algorithm 3.

Algorithm 3 enumerates all subgraphs of size k. However, if we want to enumerate nonisomorphic subgraphs, we need a mechanism to detect isomorphic subgraphs. One of the most efficient methods for graph isomorphism detection is *graph canonization*. Graph canonization produces a *canonical label* for every graph. Canonical labeling is completely graph invariant. Graph G is isomorphic to H if and only if canonical label of G equals canonical label of H [33]. There are some practical algorithms for canonical labeling like *nauty* [34], *bliss* [35], and *traces* [36]. However, there is no known polynomial-time algorithm for canonical labeling [36].

Most of the competing solutions use nauty for canonical labeling. When they find a new subgraph, first they find its canonical labeling using the nauty algorithm. Then, the new canonical label is looked up against a set of visited canonical labels. If the new canonical label is present in the set, then this subgraph is omitted else and the new canonical label is added to the label set. Some solutions, like FaSe, use a more sophisticated solution like G-tries instead of a lookup table for storing subgraphs, but the whole process is the same. This approach has two shortcomings. First, for each found subgraph, we need to generate its canonical labeling which can take exponential time in the worst case [34]. The second problem is the obligation to keep all of the unique canonical labels that are seen before in the main memory. These two shortcomings lead to unnecessary usage of processor and memory resources. Specially, when the size of subgraphs is big, for example, when k is more than 8, it would be impractical to keep all canonical labels in the main memory of a commodity workstation because there are millions of canonical labels.

To overcome these shortcomings, we propose a two-phase subgraph isomorphism solution that works with external storage. In the first phase, we use a fast $O(v^2)$ heuristic called *ordered labeling* to eliminate a considerable portion of isomorphic subgraphs. Then in the second phase, we use the nauty algorithm to eliminate all remaining isomorphic subgraphs. Advantages of our two-phase solution are as follows: (i) faster execution time and (ii) the ability to handle situations where the number of nonisomorphism subgraphs exceeds the main memory limits. A schematic of our proposed two-phase subgraph isomorphism detection solution is given in Figure 1. A flowchart for the ordered labeling step (first step of the first phase) is also given in Figure 2.

As shown in Figure 2, we use an intermediate set residing in the main memory for early duplicate ordered label

Input: A graph G and an integer k: $1 < k \leq |V(G)|$
Output: All subgraphs of size k
(1) **for each** vertex $v \in V(G)$ **do:**
(a) $V_{Extension} \leftarrow \{u \in N(\{v\}) : u > v\}$
(b) ExtendSubGraph($\{v\}, V_{Extension}, v$)
ExtendSubGraph($V_{Subgraph}, V_{Extension}, v$)
(1) **if** $|V_{Subgraph}| = k$ ***then output*** $G[V_{Subgraph}]$ and **return**
(2) **while** $V_{Extension} \neq \varnothing$ **do**
(a) remove a vertex w from $V_{Extension}$
(b) $V'_{Extension} \leftarrow V_{Extension} \cup \{u \in N_{excl}(w, V_{Subgraph}) : u > v\}$
(c) ExtendSubGraph($V_{Subgraph} \cup \{w\}, V'_{Extension}, v$)

Algorithm 1: ESU subgraph enumeration algorithm.

Input: A graph G, and an integer k: $1 < k \leq |V(G)|$, and an edge $e(v, w)$
Output: All subgraphs of size k that contain $e(v, w)$
(1) **let** *Stack* be a stack of tuples
(2) **if** $v > w$ **then swap** v and w //to guarantee that v is smaller than w
(3) $V_{Extension} \leftarrow \{u \in N(\{v\}) : u > w\} \cup \{u \in N_{excl}(w, \{v\}) : u > v\} - \{v, w\}$
(4) **push** new tuple($\{v, w\}, V_{Extension}, v$) into *Stack*
(5) **while** *Stack* is not empty **do**:
(a) $top \leftarrow$ pop the tuple on top of the stack
(b) **if** $|top[0]| = k$ ***then output*** $G[top[0]]$ and **return** //top[0] is the first item of the tuple
(c) **while** $top[1] \neq \varnothing$ **do:** //top[1] is the extension set
(i) remove a vertex x from $top[1]$
(ii) $V'_{Extension} \leftarrow top[1] \cup \{u \in N_{excl}(x, V_{Subgraph}) : u > top[2]\}$ //top[2] is the root
(iii) **push** new tuple($top[0] \cup \{x\}, V'_{Extension}, top[2]$) into *Stack*

Algorithm 2: ESE enumeration algorithm.

Input: A graph G, an integer k: $1 < k \leq |V(G)|$ as the size of subgraphs, and an integer p as the number of concurrent threads
Output: All subgraphs of size k
(1) **let** Q be an empty list.
(2) **for each** edge $e(v, w) \in E(G)$ **do:**
(a) **if** $e(w, v)$ is not in Q **then** insert $e(v, w)$into Q
(3) spawn p threads
(4) **for each** thread **do** in parallel:
(a) **while** Q is not empty **do**:
(i) pick an edge $e(v, w)$ from Q
(ii) enumerate all subgraphs of size k containing e using ESE algorithm
(5) wait **until** all threads are done

Algorithm 3: PSE subgraph enumeration algorithm.

detection and when the size of the set exceeds the memory limit, we spill ordered labels set to external memory, that is, a file on disk. To generate an ordered labeling for a subgraph, we reorder the adjacency matrix considering degree of each vertex. Then, we concatenate rows of the reordered adjacency matrix to generate the ordered label for that subgraph. Algorithm 4 gives a more formal explanation of ordered labeling algorithm. Actually, ordered labeling algorithm just changes the labels of vertices and the graph structure is not changed. Hence, ordered labeling algorithm preserves graph isomorphism class and canonical labeling. Figure 3 shows an example of ordered labeling.

According to Figure 1, after generating ordered labeling for subgraphs and dumping them to a file on external storage, we have a file containing pairs of ordered labels and their frequencies. Afterwards, we use the nauty algorithm to generate a canonical label of each ordered label. Having a file containing canonical labels and frequencies for each subgraph, first we sort the file by canonical labels using parallel external merge sort algorithm and then, we traverse

Input: A subgraph G represented with its adjacency matrix M
Output: A binary string of length $|V(G)|^2$ as the ordered labeling for G
(1) **let** L be a list of vertices, and initially $L = \varnothing$
(2) **for each** vertex $v \in V(G)$ **do:**
  (a) insert v to L
(3) sort L by degree of each vertex
(4) let *Lookup* be a lookup table and $Lookup[x]$ as the value associated to x.
(5) **for each** vertex $v \in L$ **do:**
  (a) **set** $Lookup[v]$ equal to rank of v in L
(6) let N be a binary matrix of size M filled with zeros
(7) **for each** $M_{i,j}$ in M **do**: $//A_{i,j}$ denotes the element of matrix A in row i and column j
  (a) **if** $M_{i,j} = 1$ **then set** $N_{Lookup[i],Lookup[j]}$ to 1
(8) **return** concatenation of rows of N

Algorithm 4: Ordered labeling algorithm.

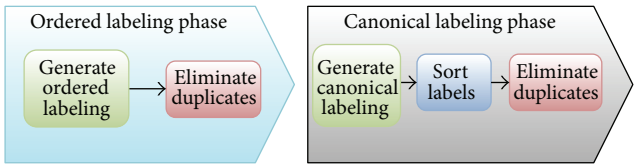


Figure 1: Two-phase isomorphism detection.

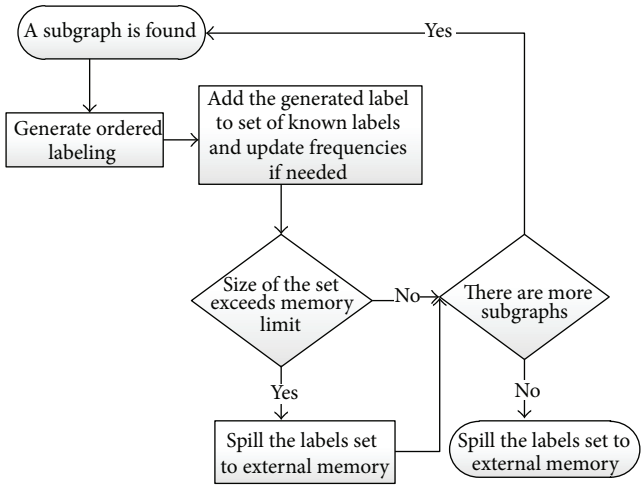


Figure 2: Generating ordered labeling for subgraphs.

the sorted file and detect duplicate canonical labels and merge frequencies of duplicate labels. At last, we have unique canonical labels and their frequencies, that is, nonisomorphic subgraphs and their frequencies.

4.1. Complexity Analysis. During the ordered labeling heuristic we perform $O(k^2)$ lookups, assuming k as the size of the subgraph. Hence, if we use a data structure like *hash table* that provides $O(1)$ expected lookups, then the time complexity of the ordered labeling algorithm would be $O(k^2)$ which is far better than traditional canonical labeling algorithms that have time complexity of $O(k!)$.

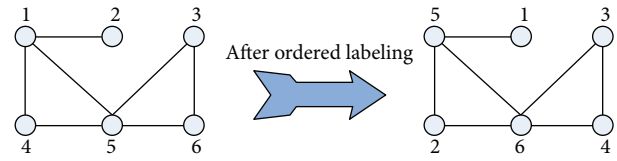


Figure 3: An example of ordered labeling.

For enumeration of subgraphs of size k, whether isomorphism detection is done or not, we need to enumerate all subgraphs of size k. In the worst case, for a complete input graph of size n, the number of induced subgraphs of size k is $C(n, k)$. On average, for a general input graph of size n, the number of subgraphs of size k should be exponential [4]. Hence, if we assume α as the number of subgraphs of size k, β as the number of isomorphic classes for subgraphs, and p as the number of processors, generating ordered labeling of all subgraphs needs $O(\alpha \cdot k^2/p)$ operations, eliminating duplicate ordered labels needs execution of standard parallel merge sort algorithm which has complexity of $O(\beta \cdot \log \beta/p)$, and applying nauty on unique ordered labels needs $O(\beta \cdot k!/p)$. Hence, the overall time complexity of Subenum is $O(\alpha \cdot k^2/p + \beta \cdot \log \beta/p + \beta.k!/p)$ which is far more better than other tools that use nauty directly which have complexity of $O(\alpha \cdot k!)$, because β is smaller than α [3, 4].

5. Experimental Results

In order to evaluate the performance and effectiveness of Subenum, we performed various experiments on different real-world graphs. For this purpose, we selected some well-known graphs from various fields like social networks, biology, communication, web graphs, and peer-to-peer networks. We have used eight different graphs: Elegans (neuronal network of Caenorhabditis elegans [37]), Jazz (network of jazz musicians [38]), School (face to face contact patterns in a primary school [39]), Vidal (proteome-scale map of human binary protein-protein interactions [40]), Gnutella (structure of Gnutella p2p network from August 31, 2002 [41]), Slash (slashdot social network from February 2009

Table 1: The properties of graphs used in experiments.

	Elegans	Jazz	School	Vidal	Gnutella	Slash	Tweet	Notre
Number of vertices	297	198	238	3,133	62,586	82,168	81,306	325,729
Number of edges	2,345	2,742	5,539	6,726	147,892	948,464	1,768,149	1,497,134
avg (deg.)	14.46	27.69	46.54	4.10	4.72	48.77	38.21	6.77
σ (deg.)	12.94	17.41	19.85	6.79	5.70	19.81	67.93	42.87

Table 2: The effectiveness of ordered labeling heuristic for subgraph isomorphism detection.

		Subgraph Size				
		3	4	5	6	7
Elegans	Number of subgraphs	47,322	1,394,259	43,256,069	1,309,307,357	37,818,052,163
	Number of ordered labels	20	552	24,745	961,476	31,104,089
	Number of nonisomorphic subgraphs	13	197	7,072	286,376	9,584,962
Jazz	Number of subgraphs	67,414	1,833,618	49,500,654	1,266,953,062	30,166,157,456
	Number of ordered labels	5	45	862	32,493	2,291,205
	Number of nonisomorphic subgraphs	4	24	267	5,647	237,008
School	Number of subgraphs	205,796	8,581,352	348,596,925	13,140,615,595	451,141,199,919
	Number of ordered labels	5	45	862	32,515	2,409,520
	Number of nonisomorphic subgraphs	4	24	267	5,647	237,319
Vidal	Number of subgraphs	86,715	2,161,170	62,607,036	1,901,854,904	58,919,388,890
	Number of ordered labels	42	766	18,201	411,148	8,637,628
	Number of nonisomorphic subgraphs	3	24	267	4,909	97,094
Gnutella	Number of subgraphs	1,564,126	23,646,400	449,446,489	9,806,726,769	234,415,296,091
	Number of ordered labels	7	70	933	12,787	170,594
	Number of nonisomorphic subgraphs	5	32	291	2,714	25,230

[42]), Tweet (social circles from Twitter [43]), and Notre (web graph of Notre Dame [44]). Main properties of these graphs are tabulated in Table 1. The first four graphs are small; for example, they have less than 10,000 vertices and edges. On the other hand, the latter four graphs are larger, for example, more than tens of thousands of vertices and up to one million edges.

The main goal of our experiments is to evaluate the overall speed of Subenum compared to available state-of-the-art algorithms. We divide the experiments into three sections. First, we evaluate effectiveness of ordered labeling heuristic for subgraph isomorphism detection. Then, we evaluate scalability and parallelism performance of Subenum on multicore and multiprocessor machines. Finally, we compare ultimate speed of Subenum to some of the available tools for subgraph enumeration (FANMOD, Kavosh, G-Tries, and FaSe) considering different input graphs and subgraph sizes.

We used two machines during our experiments. The first machine was a four-core Intel i7-2600 CPU having 8 GB of RAM and running Windows 7 64-bit edition. The second machine had two 6-core Intel Xeon-E5620 CPUs and 32 GB of RAM running Ubuntu 12.04. The 4-core i7 machine is mainly used for comparison with other tools, while the 12-core Xeon machine is mainly used for parallelism and scalability experiments. For better scalability, we used Azul Zing JVM on the Xeon machine. Subenum is coded in the Java programming language and its source code is available via GitHub at https://github.com/shahrivari/subenum.

5.1. Effectiveness of Ordered Labeling Heuristic. For testing the effectiveness of ordered labeling heuristic, we applied ordered labeling on some of the input graphs considering subgraphs of various sizes. The details about the effectiveness of ordered labeling heuristic are given in Table 2.

Three numbers are reported per subgraph size and input graph in Table 2: the number of subgraphs, the number of ordered labels, and the number of nonisomorphic subgraphs. As the numbers show, the numbers of ordered labels are much smaller than the numbers of subgraphs and close to the number of nonisomorphic subgraphs. This shows that Subenum calls the expensive nauty algorithm significantly fewer times (in orders of the number of ordered labels) while other solutions call nauty per each found subgraph. For example, considering the subgraphs of size 6 for Gnutella graph, Subenum calls nauty 12,787 times, while other tools call nauty more than 9 billion times.

5.2. Parallelism and Scalability. Subenum is inherently designed for running on multicore and multiprocessor machines. Hence, an important performance factor is the scalability of Subenum. That is to say, we want to know how much speedup is gained when additional processors are available to Subenum. For this purpose, we calculated the speed-up of Subenum running with different counts of threads. We performed both all-subgraph enumeration and nonisomorphic subgraph enumeration. For calculating the speed-up value, we divided the execution time of multithreaded

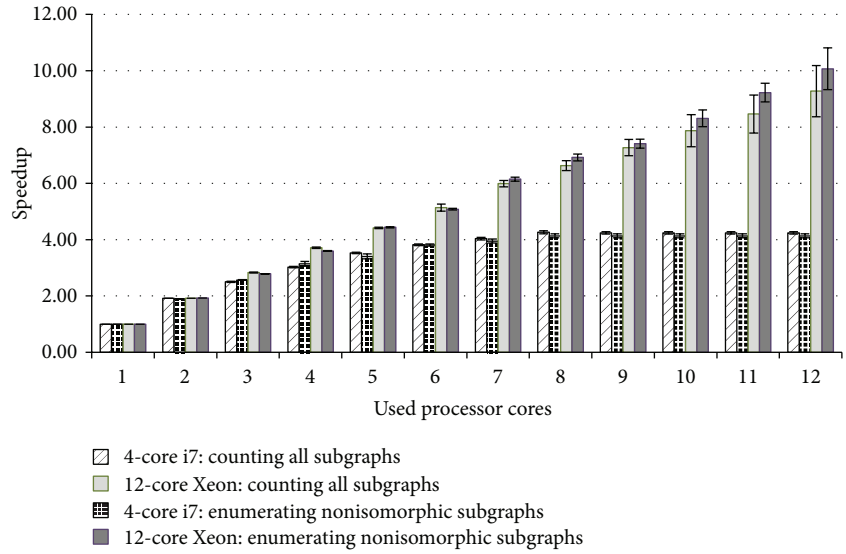


Figure 4: Overall speedup values considering different graphs.

version to the execution time of the single threaded version. Figure 4 shows an overall view of Subenum's scalability using multithreads. For this experiment, we executed Subenum using different number of threads on all of the input graphs and enumerated subgraphs of sizes 5 and 6 for the first four graphs and subgraphs of size 3 and size 4 for the latter four graphs. As Figure 4 shows, Subenum can reach a near-linear speedup when additional threads of execution are used until threads count reaches the number of available processor cores. Note that the small improvements after increasing the number of threads to values greater than number of cores are due to *HyperThreading* feature of Intel CPUs which allows each core to run two logical threads simultaneously. More details of speedup values for each input graph are given in Figure 5 which shows the increase of speedup values for each input graph by using more threads.

5.3. Comparison to Other Solutions. The main goal of Subenum is to provide a faster solution for all-subgraph enumeration and nonisomorphic subgraph enumeration problems compared to available solutions. In this part of the paper, we compare the performance of Subenum to the best known available software for all-subgraph enumeration problem. The comparison is made to Kavosh, FANMOD, gtrieScanner, and FaSe. During the experiments of this section, we used the 4-core i7 machine.

All of the other solutions are sequential and comparing Subenum which is a parallel solution to sequential solutions is not very fair because Subenum can use all of the available cores, while others just use a single core. For this reason, in this experiment we used the 4-core i7 machine that has fewer cores compared to the 12-core Xeon machine. For better comparison, for every graph and subgraph size, we reported the performance of Subenum when using a single core, too. Note that Subenum is programmed in Java, while all of the other solutions are programmed in C/C++ which has proven to produce faster executable programs because of producing

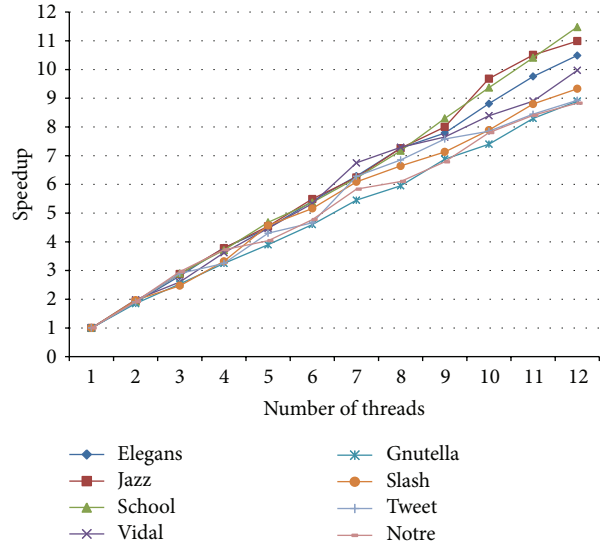


Figure 5: The speedup for each graph using different number of threads on 12-core Xeon machine.

native machine code in contrast to Java that compiles to byte code which executes in the *Java Virtual Machine* (*JVM*).

For more clarity, we divided the input graphs into two groups. The first group consists of smaller graphs: Elegans, Jazz, School, and Vidal. The second group consists of larger graphs: Gnutella, Slash, Tweet, and Notre. Since the graphs of first group are smaller, larger subgraph sizes can be enumerated, while for the second group, enumerating large subgraphs like 8 can take months and even years.

Figure 6 gives an overall performance comparison of different solutions for the first group of graphs. For this experiment, we enumerated nonisomorphic subgraphs of sizes 5 and 6 for each input graph and reported the average normalized times. As Figure 6 shows, for all of the input graphs Subenum is the fastest solution. When Subenum is

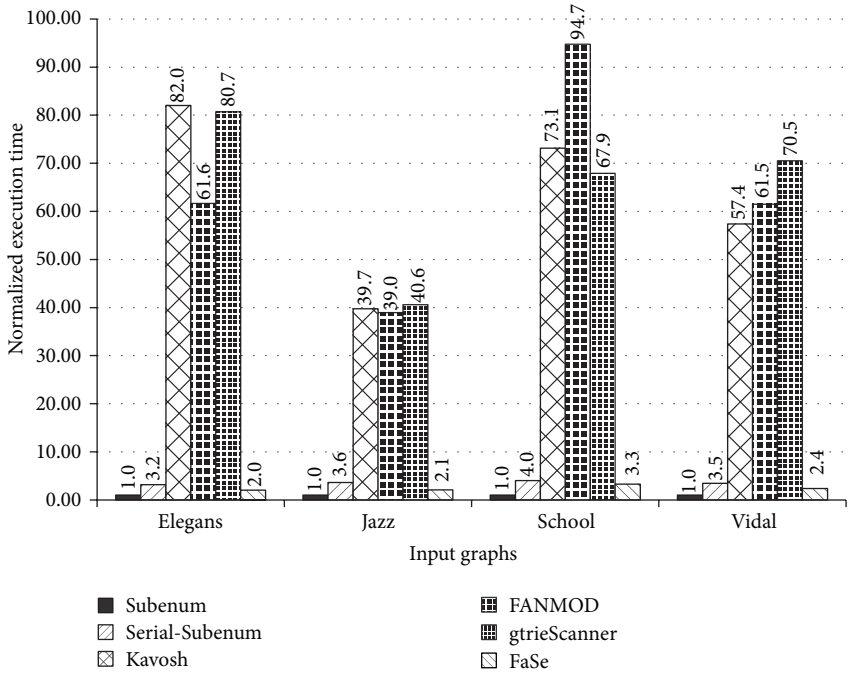


Figure 6: Average normalized execution times for smaller input graph (subgraphs of sizes 5 and 6).

executed in sequential mode, FaSe is faster, but it does not reach performance of Subenum when all of the 4 cores of the i7 CPU are used. We believe that the better performance of FaSe in the sequential mode is due to better performance of C++ compared to Java. Using Figure 6, we can also conclude that there is a great performance gap between Subenum and FaSe compared to other solutions. The main reason behind this issue is the better methods that Subenum and FaSe use to deal with subgraph isomorphism detection.

More details are given in Table 3. The execution times for each input graph and different subgraph sizes are given for each solution. For all input graphs and subgraph sizes, Subenum delivers the fastest execution time. An interesting point is the failure of other tools when larger subgraphs are enumerated. When large subgraphs (e.g., 8) are enumerated, other tools crash due to memory issues. These cases are denoted by "*Out of Mem.*" in Table 3. The main reason behind this is the large count of nonisomorphic subgraphs. Since other tools keep all nonisomorphic subgraphs in the main memory, the main memory fills up and the tools crash. For the cases in which the execution of a solution took more than a week, we did not proceed and reported these cases by an estimation like ">1 *week*".

We performed the same experiments for the larger input graphs, too. Kavosh and gtrieScanner failed to load all of the graphs. Inspecting their code shows that they use a Boolean matrix for storing edges. Hence, they need $O(v^2)$ space, considering v as the number of vertices. FaSe can just handle the Gnutella and Slash graphs and fails due to insufficient memory for the rest of the input graphs. FANMOD performs better in handling large graphs. However, it is much slower than Subenum and FaSe. The details of execution times are given in Table 4.

6. Conclusion and Further Work

The number of both isomorphic and nonisomorphic subgraphs of a given graph grows exponentially when the size of input graph or the subgraphs to be enumerated is increased. Hence, the only available solution for accelerating all-subgraph enumeration problem is to use parallel and distributed systems. We presented a new parallel solution, named Subenum, for the all-subgraph enumeration problem on multicore and multiprocessor systems. In contrast to available parallel solutions that are designed for execution on cluster computing systems, Subenum is designed for faster execution on commodity multicore and multiprocessor desktop and workstation systems.

The novelties of Subenum can be summarized in three points. First, we designed a new parallel subgraph enumeration algorithm named PSE that provides a load-balanced and parallel procedure suited for subgraph enumeration on multicore and multiprocessor systems. Second, we offered a new, simple, and polynomial heuristic for subgraph isomorphism detection problem and we showed that it is very effective for pruning candidate subgraphs. And lastly, using a parallel external sorting solution we enabled Subenum to enumerate nonisomorphic subgraphs even when they are so large that they do not fit in the main memory. Our practical experiments on different real-world input graphs showed that Subenum is a scalable parallel solution and can easily outperform the fastest available tools like FANMOD and Kavosh on commodity multicore machines.

For further work, we plan to develop a distributed subgraph enumeration solution using the MapReduce programming model and the Hadoop framework. We have done most of the work and the preliminary results are encouraging.

Table 3: The execution times of different tools for small graphs in seconds.

	Tool	Subgraph size			
		5	6	7	8
Elegans	Subenum	**1.7**	**39**	**1,175**	**37,993**
	Serial Subenum	4.2	130	4,553	147,806
	FANMOD	55.2	2,453	85,465	Out of Mem.
	Kavosh	53.1	3,285	119,286	Out of Mem.
	gtrieScanner	52.7	3,233	Out of Mem.	Out of Mem.
	FaSe	2.2	81	Out of Mem.	Out of Mem.
Jazz	Subenum	**1.7**	**40**	**1,051**	**29,216**
	Serial Subenum	5.3	145	4,059	111,876
	FANMOD	46.3	1,578	49,660	Out of Mem.
	Kavosh	45.6	1,611	50,310	Out of Mem.
	gtrieScanner	43.7	1,649	Out of Mem.	Out of Mem.
	FaSe	2.7	84	2,663	Out of Mem.
School	Subenum	**8.8**	**285**	**17,062**	>1 month
	Serial Subenum	32.5	1,157	68,326	>1 month
	FANMOD	604	27,230	>1 week	Out of Mem.
	Kavosh	405	21,085	>1 week	Out of Mem.
	gtrieScanner	399	19,554	>1 week	Out of Mem.
	FaSe	20	954	42,156	Out of Mem.
Vidal	Subenum	**1.9**	**52**	**1,756**	**61,374**
	Serial Subenum	5.7	181	**6,851**	238,449
	FANMOD	76	3,239	147,369	>1 month
	Kavosh	76	3,016	143,159	>1 month
	gtrieScanner	82	3,720	197,432	>1 month
	FaSe	3.2	124	3,780	Out of Mem.

Table 4: The execution times of different tools for large graphs in seconds.

	Tool	Subgraph size			
		4	5	6	7
Gnutella	Subenum	**2.3**	**30**	**687**	**19,288**
	Serial Subenum	11.9	153	3,237	87,796
	FANMOD	15	442	11,715	324,152
	Kavosh	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	gtrieScanner	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	FaSe	3.0	49	1,108	29,423
Slash	Subenum	**1,040**	**460,928**	>1 year	>1 year
	Serial Subenum	3,963	>1 week	>1 year	>1 year
	FANMOD	67,047	>1 month	>1 year	>1 year
	Kavosh	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	gtrieScanner	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	FaSe	2,373	>1 week	>1 year	>1 year
Tweet	Subenum	**3,791**	>1 month	>1 year	>1 year
	Serial Subenum	**14,671**	>1 month	>1 year	>1 year
	FANMOD	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	Kavosh	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	gtrieScanner	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	FaSe	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
Notre	Subenum	**23,273**	>1 month	>1 year	>1 year
	Serial Subenum	**82,833**	>1 month	>1 year	>1 year
	FANMOD	451,156	>1 year	>1 year	>1 year
	Kavosh	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	gtrieScanner	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.
	FaSe	Out of Mem.	Out of Mem.	Out of Mem.	Out of Mem.

Another opportunity is using available powerful and low cost Graphical Processing Units (GPU). Due to the complexity of the problem, using parallel GPU based solutions like CUDA may also bring a huge performance boost.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[2] P. Ribeiro and F. Silva, "g-tries: an efficient data structure for discovering network motifs," in *Proceedings of the ACM Symposium on Applied Computing (SAC '10)*, pp. 1559–1566, 2010.

[3] Z. R. M. Kashani, H. Ahrabian, E. Elahi et al., "Kavosh: a new algorithm for finding network motifs," *BMC Bioinformatics*, vol. 10, no. 1, article 318, 2009.

[4] S. Wernicke and F. Rasche, "FANMOD: a tool for fast network motif detection," *Bioinformatics*, vol. 22, no. 9, pp. 1152–1153, 2006.

[5] J. Grochow and M. Kellis, "Network motif discovery using subgraph enumeration and symmetry-breaking," in *Research in Computational Molecular Biology*, T. Speed and H. Huang, Eds., vol. 4453 of *Lecture Notes in Computer Science*, pp. 92–106, Springer, Berlin, Germany, 2007.

[6] F. Harary and E. Palmer, "The enumeration methods of Redfield," *American Journal of Mathematics*, vol. 89, no. 2, pp. 373–384, 1967.

[7] D. S. Johnson, "The NP-completeness column," *ACM Transactions on Algorithms*, vol. 1, no. 1, pp. 160–176, 2005.

[8] P. Ribeiro, F. Silva, and L. Lopes, "Parallel discovery of network motifs," *Journal of Parallel and Distributed Computing*, vol. 72, no. 2, pp. 144–154, 2012.

[9] Z. Zhao, M. Khan, V. S. A. Kumar, and M. V. Marathe, "Subgraph enumeration in large social contact networks using parallel color coding and streaming," in *Proceedings of the 39th International Conference on Parallel Processing (ICPP '10)*, vol. 10, pp. 594–603, September 2010.

[10] F. Schreiber and H. Schwbbermeyer, "Towards motif detection in networks: frequency concepts and fiexible search," in *Proceedings of the International Workshop on Network Tools and Applications in Biology*, pp. 91–102, 2004.

[11] M. Kiyomi, *Studies on subgraph and supergraph enumeration algorithms [Ph.D. thesis]*, The Graduate University for Advanced Studies, 2006.

[12] P. Ribeiro, *Efficient and scalable algorithms for network motifs discovery [Ph.D. thesis]*, Porto University, 2011.

[13] A. Masoudi-Nejad, F. Schreiber, and Z. R. M. Kashani, "Building blocks of biological networks: a review on major network motif discovery algorithms," *IET Systems Biology*, vol. 6, no. 5, pp. 164–174, 2012.

[14] S. Wernicke, "Efficient detection of network motifs," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 3, no. 4, pp. 347–359, 2006.

[15] E. Wong, B. Baur, S. Quader, and C.-H. Huang, "Biological network motif detection: principles and practice," *Briefings in Bioinformatics*, vol. 13, no. 2, pp. 202–215, 2012.

[16] P. Ribeiro and F. Silva, "G-tries: a data structure for storing and finding subgraphs," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 337–377, 2014.

[17] P. Paredes and P. Ribeiro, "Towards a faster network-centric subgraph census," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*, pp. 264–271, IEEE, August 2013.

[18] X. Li, D. S. Stones, H. Wang, H. Deng, X. Liu, and G. Wang, "NetMODE: network motif detection without Nauty," *PLoS ONE*, vol. 7, no. 12, Article ID e50093, 2012.

[19] S. Khakabimamaghani, I. Sharafuddin, N. Dichter, I. Koch, and A. Masoudi-Nejad, "QuateXelero: an accelerated exact network motif detection algorithm," *PLoS ONE*, vol. 8, no. 7, Article ID e68073, 2013.

[20] W. Tie, J. W. Touchman, Z. Weiyi, E. B. Suh, and X. Guoliang, "A parallel algorithm for extracting transcriptional regulatory network motifs," in *Proceedings of the 5th IEEE Symposium on Bioinformatics and Bioengineering (BIBE '05)*, pp. 193–200, October 2005.

[21] M. Schatz, E. Cooper-Balis, and A. Bazinet, *Parallel Network Motif Finding*, 2008.

[22] P. Ribeiro, F. Silva, and L. Lopes, "Efficient parallel subgraph counting using G-tries," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 217–226, 2010.

[23] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[24] T. White, *Hadoop: The Definitive Guide*, Yahoo Press, 2012.

[25] Z. Zhao, G. Wang, A. R. Butt, M. Khan, V. S. A. Kumar, and M. V. Marathe, "SAHAD: subgraph analysis in massive networks using Hadoop," in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS '12)*, pp. 390–401, Shanghai, China, May 2012.

[26] Z. Zhao, "Subgraph querying in relational networks: a mapreduce approach," in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops (IPDPSW '12)*, pp. 2502–2505, May 2012.

[27] J. Cohen, "Graph twiddling in a MapReduce world," *Computing in Science and Engineering*, vol. 11, no. 4, Article ID 5076317, pp. 29–41, 2009.

[28] B. Wu and Y. Bai, "An efficient distributed subgraph mining algorithm in extreme large graphs," in *Artificial Intelligence and Computational Intelligence*, vol. 6319 of *Lecture Notes in Computer Science*, pp. 107–115, Springer, Berlin, Germany, 2010.

[29] F. N. Afrati, D. Fotakis, and J. D. Ullman, "Enumerating subgraph instances using map-reduce," in *Proceedings of the 29th International Conference on Data Engineering (ICDE '13)*, pp. 62–73, April 2013.

[30] A. G. Rudi, S. Shahrivari, S. Jalili, and Z. R. M. Kashani, "RANGI: a fast list-colored graph motif finding algorithm," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 2, pp. 504–513, 2013.

[31] G. Blin, F. Sikora, and S. Vialette, "GraMoFoNe: a cytoscape plugin for querying motifs without topology in protein-protein interactions networks," in *Proceedings of the 2nd International Conference on Bioinformatics and Computational Biology (BICoB '10)*, pp. 38–43, March 2010.

[32] G. Blin, *Combinatorial objects in bio-algorithmics: related problems and complexities [Ph.D. thesis]*, Université Paris-Est, 2012.

[33] L. Babai and E. M. Luks, "Canonical labeling of graphs," in *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pp. 171–183, 1983.

[34] B. D. McKay, "Practical graph isomorphism," *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.

[35] T. Junttila and P. Kaski, "Engineering an efficient canonical labeling tool for large and sparse graphs," in *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics*, pp. 135–149, January 2007.

[36] H. Katebi, K. Sakallah, and I. Markov, "Conflict anticipation in the search for graph automorphisms," in *Logic for Programming, Artificial Intelligence, and Reasoning*, N. Bjørner and A. Voronkov, Eds., vol. 7180 of *Lecture Notes in Computer Science*, pp. 243–257, Springer, Berlin, Germany, 2012.

[37] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs," *Bioinformatics*, vol. 20, no. 11, pp. 1746–1758, 2004.

[38] M. G. Pablo and L. Danon, "Community structure in jazz," *Advances in Complex Systems*, vol. 6, no. 4, pp. 565–573, 2003.

[39] J. Stehlé, N. Voirin, A. Barrat et al., "High-resolution measurements of face-to-face contact patterns in a primary school," *PLoS ONE*, vol. 6, no. 8, Article ID e23176, 2011.

[40] A.-C. Gavin, P. Aloy, P. Grandi et al., "Proteome survey reveals modularity of the yeast cell machinery," *Nature*, vol. 440, no. 7084, pp. 631–636, 2006.

[41] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, article 2, 2007.

[42] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th International World Wide Web Conference (WWW '10)*, pp. 641–650, April 2010.

[43] J. Leskovec and J. McAuley, "Learning to discover social circles in ego networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 539–547, December 2012.

[44] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: diameter of the World-Wide Web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 451476, 13 pages
http://dx.doi.org/10.1155/2015/451476

Research Article

Skillrank: Towards a Hybrid Method to Assess Quality and Confidence of Professional Skills in Social Networks

Jose María Álvarez-Rodríguez,[1,2] Ricardo Colomo-Palacios,[3] and Vladimir Stantchev[4]

[1] *Universidad Carlos III de Madrid, Avenida Universidad 30, Leganés, 28911 Madrid, Spain*
[2] *Wrocław University of Technology, Wyspianskiego 27, 50-370 Wrocław, Poland*
[3] *Østfold University College, B R A Veien 4, 1783 Halden, Norway*
[4] *SRH University Berlin, Ernst-Reuter-Platz 10, 10587 Berlin, Germany*

Correspondence should be addressed to Vladimir Stantchev; vladimir.stantchev@srh-hochschule-berlin.de

Received 3 February 2014; Revised 1 November 2014; Accepted 21 November 2014

Academic Editor: Przemyslaw Kazienko

The present paper introduces a hybrid technique to measure the expertise of users by analyzing their profiles and activities in social networks. Currently, both job seekers and talent hunters are looking for new and innovative techniques to filter jobs and candidates where candidates are trying to improve and make their profiles more attractive. In this sense, the Skillrank approach is based on the conjunction of existing and well-known information and expertise retrieval techniques that perfectly fit the existing web and social media environment to deliver an intelligent component to integrate the user context in the analysis of skills confidence. A major outcome of this approach is that it actually takes advantage of existing data and information available on the web to perform both a ranked list of experts in a field and a confidence value for every professional skill. Thus, expertise and experts can be detected, verified, and ranked using a suited trust metric. An experiment to validate the Skillrank technique based on precision and recall metrics is also presented using two different datasets: (1) ad hoc created using real data from a professional social network and (2) real data extracted from the LinkedIn API.

1. Introduction

In recent years, social network research has been carried out using data collected from online interactions and from explicit relationship links in online social network platforms like, for instance, Facebook and Linkedin [1]. Among these tasks, expert and people search is one of the most challenging tasks that one can try in social networks [2, 3].

Expertise represents the skill of answering some questions or conducting some activities [4]. Thus, the focus of expertise location is finding an answer, a solution, or a person with whom details of a problem can be discussed [5] or a task can be performed [6]. In other words, expert finding addresses the task of identifying the right person with the appropriate skills and knowledge. Effective management of expertise can benefit both organizations and individuals by easing the access to knowledge, as well as sharing and applying knowledge [7].

In this light, expert finding involves two main aspects including expertise identification ("*Who are the experts on Topic X?*") and expertise selection ("*What does Expert Y know?*") [8]. In the later topic, expert profiling turns the expert-finding task around and asks the following: *What topic(s) does a person know about?* [9]. Topics such as expertise relevance and authority within a community have been pointed out as some of the factors to assess expert's competence [10, 11]. Given that complete and accurate expert profiles enable people and search engines to effectively and efficiently locate the most appropriate experts for an information need [9], this paper presents an expert profiling approach to analyze expert's skills confidence by means of hybrid soft computing techniques. One of the main advantages of Skillrank is the use of LinkedIn as a source of expertise. LinkedIn is likely the most notable example of business-oriented social networking site. The company was founded in December 2002 and launched six months late. LinkedIn

reports by December 2014 more than 330 million users in 200 countries and territories. In this professional social networking site, users are allowed to track and publish their career paths, skills and past experiences, the size and tenure of the teams with whom they've worked, and the roles they played on each team [12]. LinkedIn users self-report their expertise and ask members of their social network to provide positive references or recommendations for them [13]. Although LinkedIn has been used in the literature for expertise search [14, 15], to the best of authors knowledge, there is not a study devoted to the application of self-disclosure and social network integrators to assess the quality and confidence of professional skills in this network.

On the other hand, a good number of techniques have been designed to exploit the information available in social networks and, in general, to address problems that contain an implicit graph. The well-known algorithm PageRank [16] by Google Inc. was developed to assign a measure of importance to each web page. This algorithm works by counting the number and quality of links to a page to determine a rough estimate of how important a website is. The underlying assumption is that more important websites are likely to receive more links from other websites. In the same way, the HITS (Hyperlink-Induced Topic Search) algorithm [17] also known as "hubs and authorities" is a kind of analysis technique that also rates web pages. It was designed by Kleinberg from the Department of Computer Science at Cornel and the idea behind hubs and authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming; that is, some web pages are known as hubs and serve as hubs that compile large directories of web pages. These directories are not actually authoritative in some topic but a good hub represents a page that points to many other pages and a good authority page is expected to be linked to many hubs. The main restriction of the HITS algorithm lies in its applicability since it only operates in a small subgraph. This subgraph is considered to be query dependent, whenever the search contains a different query phrase, the seed changes as well as the HITS algorithm ranks the seed nodes according to their authority and hub weights. The SPEAR (Spamming-resistant Expertise Analysis and Ranking) algorithm [18] is another tool for ranking users in social networks by their expertise and influence within a community. It is also a graph-based technique to measure the expertise of users by analyzing their activities and interaction. The main idea behind this technique lies on the ability of users to find new and high-quality information on the Internet. This algorithm is an extension of the aforementioned HITS algorithm including two main elements: (1) Mutual reinforcement of user expertise and document quality and (2) Discoverers versus followers. The combination of both elements has been demonstrated to reward quality over quantity of user activities and that is why it has been also applied to detect spam attacks [19]. Although graph analysis techniques [20] have been widely used to study social networks (e.g., trend detection, opinion mining, sentiment analysis, information retrieval, etc.) and, in most of cases, the PageRank algorithm can be seen as a precursor of this kind of approach, there is still lack of techniques to deal with quality over quantity. In this sense, the SPEAR algorithm offers us a technique that can be applied to a rather wide area of domains such as assessment of skills quality. In the context of graph-based algorithms for expertise ranking, the ExpertRank [10] algorithm proposes a novel technique to evaluate the expertise of users based on both document-based relevance and one's authority in this or her knowledge community. Authors modified the PageRank algorithm to evaluate one's authority so that it reduces the effect of certain biasing communication behavior in online communities. As an important cornerstone and relevant to this work, they explored three different expert ranking strategies that combine document-based relevance and authority: linear combination, cascade ranking, and multiplication scaling. This evaluation has been done using a popular online knowledge community showing that the proposed algorithm achieves the best performance when both document-based relevance and authority are considered.

In this paper a reinterpretation and extension of the SPEAR algorithm, called Skillrank, is presented. Furthermore, the evaluation of the presented approach is carried out by comparing existing approaches for expertise ranking such as the HITS and SPEAR algorithms to the proposed technique when tests are executed on top of two datasets extracted from the LinkedIn API. To do so, a panel of experts has established a set of expected results and values that are compared to the real results provided by each algorithm with the aim of obtaining measures of precision and recall.

The remainder of this paper is organized as follows. Section 2 presents the related literature. The proposed approach for skill ranking is illustrated in Section 3. In Section 4, experiment evaluations are conducted to compare our approach with other methods. Section 5 presents main conclusions and future research directions.

2. State of the Art

Expert and credibility finding is not a new issue in literature. As a result of this, literature has vastly reported works on the topic and even produced relevant surveys on the topic for example [21, 22]. Methodologies of expert finding can be divided into three categories [4, 7]: Content-Based Approach, Network-Based Approach, and Hybrid Approach. On the other hand, other works [10], propose a different taxonomy of existing expert finding systems. These authors indicate that these systems are based on four kinds of expertise indicators: self-disclosed information, authored documents, social network analysis and hybrid techniques [23]. An analysis performed by these authors reveal that hybrid techniques are not combining self-disclosure indicators with social network analysis or document-based indicators. In other words, authors underline that self-disclosure indicators can be seen as isolated indicators that need deeper analysis.

In [9], authors make in-depth review of benchmarking techniques and components that constitute a test collection with special emphasis on error analysis. They also give an overview of different test collections for expert profiling and

expert finding. In [24], author reviews more recent examinations of the validity of a test collection approach and evaluation measures as well as he outlines trends in current research exploiting query logs and live labs to finally show that, despite its age, this long-standing evaluation method is still a highly valued tool for retrieval research. Furthermore, the Text REtrieval Conference (TREC) or the Yandex Personalized Web Search Challenge also falls in this area of methods for assessment ad-hoc datasets through train and test processes in different topics. For instance, in 2008, the main topic of the TREC conference was focused on expert finding where a dataset from the Tilburg University (http://ilk.uvt.nl/uvt-expert-collection/documentation/documentation.html) was used as input of a competition to find and rank experts. Usually these evaluation methodologies are in charge of asserting the results of an information retrieval process by comparing expected results to actual results and taking into account measures [25] of precision, recall, sensitivity, or stability. On the other hand, relevance assessment methods are usually created by a panel of experts and a good number of collections can be found in different domains such as web search, movie/tourism recommendation, medical diagnosis, and so forth. Finally statistical significance tests are used to estimate the average of system performance according to a set of queries that can be generalized. In this sense, the Wilcoxon test, Student's t-test, and the Fisher pairwise comparison are common techniques to assess a P value under certain degrees of freedom. All these techniques have been reviewed in [9] and they are relevant to this work due to the fact that a validation of the expected results must be done to assess if the Skillrank algorithm is properly working. To do so, the validation section introduces the evaluation method that is a combination of an ad hoc collection built by a panel expert with measures of precision and recall.

Regarding online skills evaluation, in [26] a technique is introduced to establish a credibility rank (also known as Skillrank) to online profiles based on user's confirmations and six requirements for online skills evaluation. According to these six requirements, a credibility model is defined and populated from on-line profiles. Afterwards, a pilot is implemented to show the functional architecture that supports the online evaluation of skills. Nevertheless, the real evaluation of skills and online profiles is still an open issue and, only the architecture is presented. Authors also comment some of the limitations of their approach: (1) skills are evaluated as they are and a scale will be necessary to establish an order of expertise and (2) spread of experience and skills are also under evaluation. On the other hand, they also raise some relevant questions in the evaluation of online profiles: *What kind of information can be incorporated to enrich the skill evaluation model?* and *How half-time jobs, activities or tasks can also be included in the evaluation model?* This work is very closely related to the approach presented in this paper. However, they have been focused in the definition of a skill model and a pilot architecture instead of comparing different algorithms working on the same datasets. Other recent works [27] can also be found applying gamification techniques to build online personal skills and boost the learning process. Thus, it is possible to ensure the acquisition of skills from the early stages of learning by analyzing the online behavior and interactions [28] between peers. Finally, other works are also paying attention to the evaluation of online profiles with different purposes such as digital inclusion. As an example, authors in [29] theorize how people's online social networking skills may condition their uses of various digital media for communication.

On the other hand, reputation management systems have emerged to understand the influence of individuals or groups in a certain group. Different metrics with a particular level of effectiveness [30] are applied to assess the reputation in a social network, mailing list or any other collaborative site. For instance the *Stackoverflow* system, a question and answer system [31], uses a simple formula to establish a level of "*karma*" for each individual depending on their participation in the system. The Research Gate site, a social network for science and research, also establishes a score depending on publications and contributions that you have added to the site. In this case, reputation is passed from researcher to researcher, allowing us to build and leverage our reputation based on anything we choose to contribute. Interactions or activities in this social network will determine our score by looking in our activities (how our peers receive them) but also at who these peers are. Higher scores will be reached as much as higher scores peers interact with our activities; it can be seen as an application of the Spreading Activation technique [32] that has been widely used in information/document retrieval and recommending systems [33].

The idea behind of all these reputation management systems lies in a set of internal metrics (they are usually private to avoid fraudulent profiles) that are collected in just one value to create a rank of users by tracking their activity: asking and responding questions, using online feedback of other users (How much a response is better than another?), and so forth. These systems are also relevant to professional social network sites such as LinkedIn, ResearchGate, or Xing in which users try to complete, at the most, their profiles adding own education, professional experience, rewards, publications, and so forth as well as feedback from their connections to improve and enrich their profiles. In this sense, talent hunters have got a new way of detecting specialists in a topic by performing advanced search through tools in these websites. Nevertheless the access to this valuable information is commonly restricted and only quantitative information can be found. In this context some works have emerged for topic extraction systems and online reputation management [34] which they use a set of evaluation metrics based on handmade metadata annotation to assess the quality of different factors. Thus, trust and provenance analysis is becoming a major challenge to avoid vandalism, fraud, and so forth in public profiles, more specifically in user and company profiles. Existing works for example [35] are then focused on applying techniques to characterize profiles in reputation management systems to demonstrate through algorithms such as Eigen Trust, TNA-SL, or distributed approaches [8] are secure enough to effectively manage trust in communities.

In the field of information retrieval, expertise retrieval [36], and expert finding [37] systems have been widely studied to provide a new approach to tackle the discovery

of experts in some area [15]. Currently, practices as such competitions [38] or challenges are common techniques to retrieve experts based on their performance in a certain topic. The main objective of expertise retrieval [36] lies in the application of existing information retrieval techniques as building blocks to design advanced algorithms that can serve to create content-based links between topics and people. Nevertheless, authors have also outlined both applications to other domains such as entity retrieval as well as some conjectures on what the future may hold for expertise retrieval research. For instance, last times new novel approaches [39] are emerging to include time and evolution of skills over time as variables in expertise retrieval processes.

As a closely related field to expertise retrieval, community detection [40] in social networks is a widely active research area to segment large communities by a certain criteria. In the specific case of expertise in some topic, these algorithms can be applied to narrow down the search of experts. These techniques can be roughly divided into two groups: (1) global and (2) local approaches. The first ones assume knowledge of the entire network while local ones only assume knowledge of subcommunities featured by some attributes such as location. Global detection algorithms were firstly proposed by Girvan and Newman by iteratively removing edges until the social graph is partitioned (each partition can be consider as a community). The key point of these techniques lies in the selection of the edge to be removed and, in general, some metrics such as betweeness centrality are calculated for each edge. Thus, a large social network is divided into high-dense connected communities that share some features and, therefore, can be considered sub-communities. The main drawback of global approaches lies in the necessity of knowing the full graph (it is usually expensive in terms of time and size). In order to decrease the complexity of handling a large graph, local approaches aim for detecting communities in a more scalable and applicable way starting with a set of seed nodes to detect implicit communities. For instance, Clauset's algorithm uses intracommunity and intercommunity measures to iteratively establish or remove the connection between two nodes. Thus from a starting node communities dynamically emerge. Although these approaches are completely correct to detect underlying communities the use or inferring of dynamic attributes of nodes (users in most of cases) is still an open issue that has been studied in some works such as [41] and it allows a better and more accurate community partition. As a possible application of these aforementioned techniques, the detection of violent communities, hostility, or rivalry is currently under study [42] since the internet is understood to be a social space conducive to increased hostility, greater disinhibition, and increased social freedom. Moreover, these authors see a link between virtual hostility and actual violence. In social networks research, [43] predicts user personality by mining social interactions including Aggression-Hostility traits and [44] modeled online social interactions incorporating the effects of hostile interactions.

Finally the relevance of expertise ranking in social networks and Internet has been presented in some works to understand and exploit enterprise know-how [45], find competence gaps and learning needs inside corporations [46], improve Scrum processes [47], improve human to human interactions [48], or tackling information asymmetries in electronic marketplaces [49] to name a few.

In conclusion, a list of methods and techniques for benchmarking has been introduced with the aim of comparing existing approaches to assess personal skills quality. Furthermore, existing works related to reputation management systems and, more specifically, trust and provenance analysis can be also applied to the Skillrank technique since methods to evaluate public profiles are an emerging topic due to the current use of the web. That is why the Skillrank technique seeks for providing an innovative method to assess user profiles from a qualitative point of view through an agnostic technique that can help both talent hunters and managers to exactly know where a capability or skill can be found in their connections or employees with a certain degree of trust and provenance.

3. Skillrank: Reinterpreting the SPEAR Algorithm to Assess Skills Quality in Professional Social Networks

3.1. Summary of the HITS and SPEAR Algorithms. As the previous section has introduced, the HITS algorithm [50] identifies good authorities and hubs for a certain topic by assigning two numbers to a page: an authority and a hub weight where weights are recursively defined. A higher authority weight occurs if the page is pointed to by pages with high hub weights. A higher hub weight occurs if the page points to many pages with high authority weights. More specifically in the context of web search, the HITS algorithm first collects a base document set for each query. After that it recursively calculates the hub and authority values for each document. In order to gather the base document set I, first, a root set R matching the query is fetched from the search engine. Once this root set is configured for each document $r \in R$, a set of documents that point to r another set of documents L' that are pointed to by r are added to the set I as R's neighborhood. Then, for each document $i \in I$, let a_i and h_i be the authority and hub values, respectively, that are initialized to 1. While the values have not converged, the algorithm iteratively proceeds as follows.

(1) For all $i' \in I$ which points to i,

$$a_i = \sum_{i'} h_{i'}. \tag{1}$$

(2) For all $i' \in I$ which is pointed to i,

$$h_i = \sum_{i'} a_{i'}. \tag{2}$$

(3) Normalize a_i and h_i values so that $\sum_i a_i = \sum_i h_i = 1$.

A good hub increases the authority weight of the pages it points to. A good authority increases the hub weight of the pages that point to it. The idea is then to apply the two operations above alternatively until equilibrium values for

the hub and authority weights are reached. The author also demonstrated that the algorithm will likely converge but the bound on the number of iterations is unknown (in practice the algorithm converges quickly). New improved versions of this algorithm have emerged such as BHITS by giving a document a default authority weight of $1/k$ if the document is in a group of k documents on a first host which link to a single document on a second host, and a default hub weight of $1/l$ if there are l links from the document on a first host to a set of documents on a second host. Nevertheless and according to its authors, this new version of the algorithm generated bad results when a root link has few in-links but a large number of out-links that are not relevant to the query.

On the other hand, the SPEAR algorithm [18, 19] makes use of the HITS definition to introduce the concept of expert, someone with a high level of knowledge, technique, or skills in a particular domain. This implies that experts are reliable sources of relevant resources and information but with two main assumptions.

(1) Mutual reinforcement of user expertise and document quality. The expertise of a user in a particular domain will depend on the quality of the documents he/she has found. In the same way, quality of documents will depend on the expertise of the user who has found them. This is an issue that has been studied in Psychology and it states that expertise involves the ability of selecting best and relevant information in a certain context. The SPEAR algorithm is based on this assumption and an expert should be someone who selects by quality instead of quantity.

(2) Discoverers versus followers. The second assumption of the SPEAR algorithm lies in the definition of a *discoverer* (expert user that finds high-quality and relevant information) versus a *follower* (an user that annotates a document after a *discoverer does*).

Under the aforementioned assumptions the SPEAR algorithm produces a ranking of users with regard to a set of one or more tags. It assumes that a topic of interest is represented by a tag t. The algorithm works as follows [18, 19].

(i) Firstly the set of tags R_t is extracted from an underlying folksonomy in a certain social network. Each tag is represented by the tuple $r = (u, t, d, c)$ where u is the user, c is the time when the tag t was assigned to the document d, and $c_1 < c_2$ if c_1 refers to an earlier time than c_2.

(ii) Then, the next vectors are defined:

  (a) a vector $\vec{E} = (e_1, e_2, \ldots, e_M)$ containing the expertise scores of users where $M = |U_t|$ is the number of unique users in R_t,

  (b) a vector $\vec{Q} = (q_1, q_2, \ldots, q_N)$ containing the quality scores of documents where $N = |D_t|$ is the number of unique documents in R_t.

(iii) According to the first assumption, mutual reinforcement refers to the idea that the expertise score of a user depends on the quality scores of the documents to which he tags with t, and the quality score of a document depends on the expertise score of the users who assign tag t to it. Authors define an adjacency matrix A of size $M \times N$ where $A_{i,j} = 1$ if user i has annotated with the tag t the document j, and $A_{i,j} = 1$ otherwise. Based on this matrix, the calculation of expertise and quality scores is an iterative process similar to that of the HITS algorithm: $\vec{E} = \vec{Q} \times A^T$ and $\vec{Q} = \vec{E} \times A$.

(iv) On the other hand, the second assumption is implemented by changing the definition of the aforementioned adjacency matrix. Instead of assigning either 0 or 1 (like the HITS algorithm) the following equation is used to populate the initial values of the matrix A.

  (a) $A_{i,j} = |\{u \mid (u, t, d_j, c), (u_i, t, d_j, c_i) \in R_t \wedge c_i < c\}| + 1$.

  (b) Thus, the cell $A_{i,j}$ is equal to 1 plus the number of users who have assigned tag t to document d_j after user u_i. Hence, if u_i is the first to assign t to d_j, $A_{i,j}$ will be equal to the total number of users who have assigned t to d_j. If u_i is the most recent user to assign t to d_j, $A_{i,j}$ will be equal to 1. The effect of such initialization is that matrix A represents a sorted timeline of any users who tagged a given document d_j.

(v) The last step is to assign a proper credit score to users by applying a credit scoring function C to each element $A_{i,j}$. According to the authors three different functions could be applied to the matrix A.

  (a) A linear credit score $C(x) = x$. This function was initially discarded by the authors because discoverers of a popular document would receive a comparatively higher expertise score although they might have not contributed in any other document thereafter.

  (b) An increasing function but with a decreasing first derivative to retain the ordering of the scores in A. Authors demonstrated that this kind of function enables the possibility of keeping discoverers score higher than followers but differences between higher scores will be reduced to avoid the undesirable effect of assigning high expertise scores to users who were the first in tagging a few set of popular documents but without further contribution in high-quality documents thereafter. Finally, the authors selected the function $C(x) = \sqrt{x}$ as credit score for their experiments.

3.2. Skillrank in Online Communities. A simplistic definition of an online community or social network is a set of $C = \{U, F, DF, R\}$, where U is the set of users that interact with each other, F is a set of static features or attributes, DF is a set of dynamic or inferred attributes that define the

community, and R is the set of all resources generated by users. More specifically, a user $u_i \in U$ is also described by a set of attributes $u_i = \{S, D\}$, where S is the set of static attributes that describe the user profile and they are usually defined by the own user. On the other hand, D represents a dynamic set of attributes that can be inferred or predicted by tracking the user's activity and interaction in the context of social network C. Furthermore, any social network can be divided into different subcommunities (subgraphs) C_k that are also communities, and by extension, a social network can be also be defined as the union of several subcommunities $C = \bigcup_1^k \{C_k\}$. Formally, let K be an index set, and for each $k \in K$ then the family of sets $\{C_k : k \in K\}$ is the union set that represents an online-community:

$$C = \bigcup_1^k \{\{U_1, F_1, DF_1, R_1\}, \{U_2, F_2, DF_2, R_2\}, \ldots, \{U_k, F_k, DF_k, R_k\}\}. \quad (3)$$

Commonly, the set of users U_k are not disjoint sets, so a user u_i can be a member of several subcommunities. Nevertheless, the set of features F_k, dynamic features DF_k and resources R_k could be shared among subcommunities but they could be also disjoint sets depending of the characteristics of the social network.

Following these definitions, we can describe a social network such as *LinkedIn* containing a subcommunity "*MyLinkedIn*" that can be also partitioned in several subcommunities such as "*MyUniversity*" or "*MyWork*". According to the theoretical model,

(i) $C_{Linkedin} = \{U_{Linkedin}, F_{Linkedin}, DF_{Linkedin}, R_{Linkedin}\}$ where

(ii) $U_{Linkedin}$ is the set of all registered users.

(iii) $F_{Linkedin} = \{id = 1, type =$ "professional social network", name = "*Linkedin*", $descriptors = \{d_1, d_2, \ldots, d_k\} \ldots\}$ is a set of key-value pairs.

(iv) $DF_{Linkedin} = \{trends = \{t_1, t_2, \ldots, t_3\}, posts = \{(p_1, u_1), \ldots, (p_k, u_k)\}, time\}$ is also a set of dynamic key-value pairs in a certain moment *time*.

(v) On the other hand we can also define this social network by the union of several disjoint subcommunities. Thus $C_{Linkedin} = \{C_{MyLinkedin} \cup C_k\}$ where $C_{MyLinkedin} = \{C_{\text{MyUniversity}} \cup C_{\text{MyWork}}\}$.

(vi) Finally, users will be members of some community so u_{me} represents a *LinkedIn* user that creates the subcommunity "*MyLinkedIn*". This user and its subcommunity generate resources such as "posts," "connections," "endorsements," and so forth as an example $r^1_{U_k} = \{user = u_{me}, type =$ "endorsement", $time = timestamp$, $tag =$ "*skill*"$\}$ describes the resource in the community "*MyLinkedIn*" that was created by the user u_k using a "endorsement" in a certain moment *time* on the user u_{me}.

Although communities, users, and resources can be described through different static attributes there is still a set of dynamic or behavioral features that must be inferred to make a better description of foreknown communities and to be able to create new intercommunity relationships. Since communities, user endorsements, and so forth are evolving characteristics, it is necessary to analyze emerging or implicit user's behaviors [41, 51]. In this sense, the aforementioned community detection algorithms follow a similar approach but studying the structure of the social graph instead of analyzing contents.

Here, we propose the adaptation of the SPEAR algorithm to support the quality assessment of endorsements generated by a subcommunity; more specifically the following contexts can be identified.

(i) Community $C_{MyLinkedin}$ (Local context). Figure 1 shows that a user u_1 endorses another user u_{me} with the skill "java" in the time t_1. After that another user u_2 belonging to the same sub-community generated by user u_{me} also uses the same endorsement but at time t_2 where $t_1 < t_2$. The assumption behind this behavior is that after seeing the new endorsement (made by u_1) u_2 also realizes that this endorsement is correct and adds again the same endorsement to the user u_{me}. This situation implies that the first post (see *discoverer* in the SPEAR algorithm) has activated new annotations (see *follower* in the SPEAR algorithm) reinforcing both: (1) the skill "java" in user u_{me} and (2) the initial annotation of the user u_1. Similarly, if a user u_2 notices that a user u_1 has endorsed another user u_{me} at time t_1 this can lead to an endorsement of user u_1 for the same skill by user u_2 at time t_2 where $t_1 < t_2$ (Figure 2). Finally Figure 3 depicts the situation in which a user is activated by some activity but instead of applying the same tag she uses another tag to annotate knowledge of user u_1 (the one that started the interaction).

(ii) Community $C_{Linkedin}$ (Global context). Figure 4 shows that an user u_1 endorses another user u_{me} with the skill "java" in the time t_1. After that another user u_2, outside of the subcommunity (represented by a dashed circle) also uses the same endorsement but in time t_2, where $t_1 < t_2$, to endorse user u_1. The idea behind this behavior is that after seeing the new endorsement (made by u_1) u_2 also realizes that this endorsement can be applied to u_1. This situation implies that the first post (see *discoverer* in the SPEAR algorithm) has activated new annotations (see *follower* in the SPEAR algorithm) reinforcing the skill of the user u_1.

On the other hand, Figures 3 and 5 depict a situation in which an user u_1 assigns a skill to user u_{me} in time t_1 but although other user u_2 is activated and assigns another skill, different from the one assigned by u_1, there is not actually a correlation between them and both assignments can be interpreted as independent endorsements. The Skill-rank technique covers the aforementioned scenarios to take advantage of the data delivered by tracking user activities.

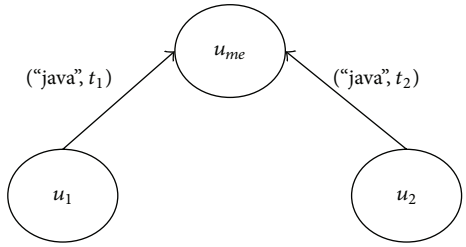


Figure 1: Correlated-endorsements to the same user u_{me} in a subcommunity.

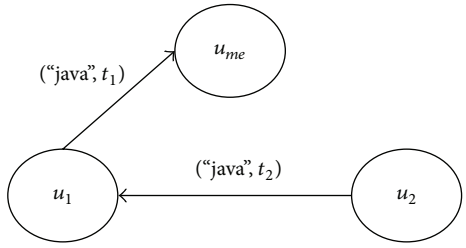


Figure 2: Correlated-endorsements to different users in a subcommunity.

Taking into account the inputs required by the SPEAR algorithm, the following vectors are redefined and the pseudocode of the algorithm is also presented in Listing 1.

(i) The set of skills S_t is extracted from the activities generated by a subcommunity C_k in a certain social network. Each endorsement is also represented by the tuple $r = (u_s, s_k, u_t, c)$ where u_s is the source user that endorses with the skill s_k to a target user u_t at time c.

(ii) Then, the next vectors are also redefined:

(a) a vector $\vec{E} = (e_1, e_2, \ldots, e_M)$ containing the expertise scores of users where $M = |U_t|$ is the number of unique users in C_k,

(b) a vector $\vec{Q} = (q_1, q_2, \ldots, q_N)$ containing the quality scores of skills where $N = |S_t|$ is the number of unique skills in C_k.

According to these definitions the unique difference between the original version of the SPEAR algorithm and this new version seems to be the naming of elements ("document" by "skill"). Nevertheless, the Skillrank facilitates a two-step process to run the SPEAR algorithm in C before C_k with the aim of populating both vectors $\vec{E}$ and $\vec{Q}$ with real values. Hence, a new interpretation of the adjacency matrix can be done. Instead of considering the adjacency matrix for the whole social network or folksonomy, each user will generate an adjacency matrix in which rows represent connections and columns skills respectively (see Table 1). The interpretation of this table is as follows: 0 represents that the user u_k have not yet endorsed using the skill s_k while another value such 2 in cell (u_1, s_2) and 1 in cell (u_2, s_1) represents that user u_1 used the skill s_2 before user u_2.

On the other hand, an analysis of the temporal and spatial complexity of the algorithm can be carried out to show the computational complexity of the technique depicted in *Error! Reference source not found.* Firstly and regarding the spatial complexity, the algorithm makes use of a set of skills S_t which contains all skill endorsements registered for a community C_k, the aforementioned vectors $\vec{E}$ and $\vec{Q}$ and a matrix A of dimensions: $M = |U_t|$, number of users in a community C_k and $N = |S_t|$, number of unique skills in a community C_k. Secondly and regarding the temporal complexity, the standard SPEAR algorithm performs k iterations containing a main operation (matrix multiplication) two times, an operation to transpose a matrix and two vector normalizations. Assuming that the adjacency matrix, A, has dimension $M \times N$, the computation complexity of the algorithm can be expressed through the next expression using the *Big-O* notation: $O(k[2 * (M * N * N) + (M * N) + M + N]) = O(k[M * N^2]) = O(M * N^2)$.

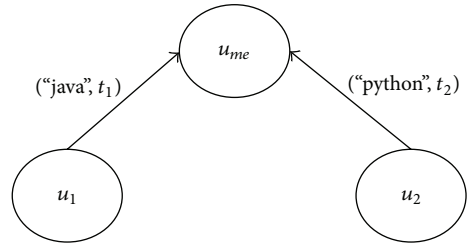


Figure 3: Independent endorsements to the same user u_{me} in a subcommunity.

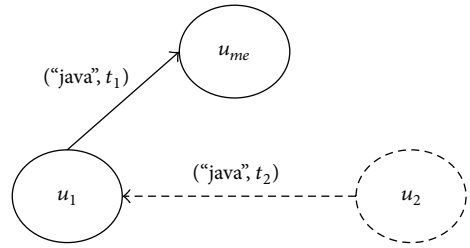


Figure 4: Correlated-endorsements to different users in a community.

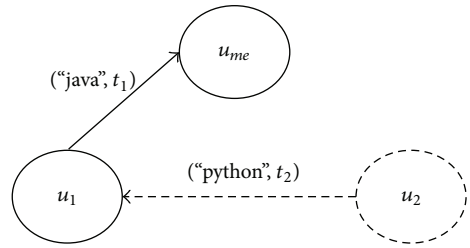


Figure 5: Independent endorsements to the same user u_{me} in a community.

4. The Case Study

To illustrate the performance in terms of precision and recall of the presented algorithms, HITS and SPEAR, with regards to the adaptation designed in Skillrank a case study

Input: Number of users M
Input: Number of skills N
Input: A set of skills $S_t \in C_k = \{(u_s, s_k, u_t, c)\}$
Input: Credit scoring function C (the same as in the standard SPEAR)
Input: Number of iterations k
Output: A list L of users.
(1) Set $\vec{E}$ to be the vector $(1, 1, \ldots, 1) \in Q^M$
(2) Set $\vec{Q}$ to be the vector $(1, 1, \ldots, 1) \in Q^N$
(3) $A \leftarrow$ Generate Adjacency Matrix (S_t, C)
(4) for $i = 1$ to k do
(5) $\quad \vec{E} \leftarrow \vec{Q} \times A^t$
(6) $\quad \vec{Q} \leftarrow \vec{E} \times A$
(7) $\quad$ Normalize $\vec{E}$
(8) $\quad$ Normalize $\vec{Q}$
(9) end for
(10) $L \leftarrow$ Sort users by their expertise scores in $\vec{E}$ and quality skills scores in $\vec{Q}$
(11) return L

Listing 1: Skillrank pseudocode. Reinterpreting the SPEAR algorithm [18, 19].

Table 1: Example of generated adjacency matrix for a user u.

	s_1	s_2	$\cdots$	s_N
u_1	0	2	0	2
u_2	2	1	3	1
$\vdots$	3	5	0	4
u_M	4	2	1	0

using different datasets is provided. Here, the evaluation of performance is not a mere question since there is a lack of real datasets containing the required information of users, connections, skills and time. To mitigate this problem, a synthetic dataset, as the basis of the experiment, has been designed after collecting real data from the LinkedIn API (currently, this API provides access to valuable but incomplete information that must be fixed by the own users). Thus, simulated communities, users, skills, and endorsements are generated to study the behavior of the different approaches. To carry out both experiments the following steps have been carried out.

(1) Select and prepare dataset. For every dataset to be evaluated a set of tuples in the form $r = (u_s, s_k, u_t, c)$ must be provided.

(2) Create a dataset for unit testing purposes. To do so a panel of experts has established a category, using an official competence scale [52], for every user and skill.

(3) Definition of precision and recall. In order to calculate both measures, next definitions are also required (given an user):

(i) true positives (tp): "number of skills that were expected to reach a certain level of quality."
(ii) false positives (fp): "number of skills that have reached a different level of quality."
(iii) true negative (tn): "number of skills that were not expected to reach a certain level of quality."
(iv) false negative (fn) "number of skills that have not reached a different level of quality."

Once we have the aforementioned definitions, precision and recall can be defined and calculated as follows.

(i) Precision is defined as "*the number of user skills that have reached the proper level of quality established by the panel of experts*":

$$\text{Precision} = P = \frac{\text{tp}}{\text{tp} + \text{fp}}. \tag{4}$$

(ii) Recall is defined as "*the number of user skills that have not reached the proper level of quality established by the panel of experts*":

$$\text{Recall} = R = \frac{\text{tp}}{\text{tp} + \text{fn}}. \tag{5}$$

(iii) F_1 score is then defined as

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6}$$

(4) Inclusion of the frequency as a basic technique for each user and skill. Given a user u_k and a skill s_k the quality of the skill is calculated as follows:

$$F_{u_k}^{s_k} = \frac{\text{number of tuples } (u_s, s_k, u_k, c)}{\text{number of connections of } u_k}. \tag{7}$$

(5) Run the experiment for every dataset and technique.

(6) Configure all techniques with default parameters (credit score function, etc.) as previous section has presented.

(7) Extract measures of precision, recall, and F_1 by comparing expected results to real results.

4.1. Design of the Experiment. The first step to run the experiments lies in the proper creation of a synthetic dataset inspired by real data extracted from the LinkedIn API. To do so a community $C_{\text{synthetic}}$ must be modeled including the required input parameters for the target algorithms. Thus, a set of users, $U_{\text{synthetic}}$ containing 10 different profiles has been designed including an average between 30 and 50 connections per user (these values have been inferred from the real data). On the other hand, a set of skills S_t, see Table 2, must be also designed according to next features: (1) technical, professional, and management skills must be available for each user and (2) all skills must be, at least, in one profile but no all profiles contain all skills. Finally, a set of endorsements in the form $r = (u_s, s_k, u_t, c)$ are generated from each user being u_s the user/connection that assigns the skill s_k to the user u_t in a certain time c.

Once the input dataset is designed, it is necessary to create a dataset containing the expected results with the aim of performing automatic unit testing. To do so, a panel experts that has already participated easing the access to their profiles in LinkedIn, has also established for their real connections a level of expertise for each skill in S_t. Thus, this dataset contains a set of tuples in the form $r_k = (u_k, s_k, l_k)$, where u_k is an user with a level of competence l_k on the skill s_k. The different levels of competence have been taken from [52] in which authors present "The Individual Competency Index (ICI)," see Table 3.

After the creation of the input and test datasets, the algorithms and unit tests can be executed to finally extract the measures of precision, recall, and F_1 and compare the different techniques. Last step involves the creation of a function to convert numerical values into a level of expertise. To do so, a percentile rank for every level of expertise is defined. The aforementioned steps have been also followed to perform the same experiment on the LinkedIn dataset. As a final remark it is relevant to discuss some research limitations that have emerged during the creation of both datasets.

(i) The use of the LinkedIn API is restricted and it is not possible to access all information that is available through the public website. Thus, some relevant information with regards to the skills is missing such as who has endorsed someone. To overcome this issue our panel of experts and collaborators were asked to complete this information.

(ii) Another issue in the use of the LinkedIn API lies in the lack of time for each endorsement. This is a critical point since algorithms are based in this assumption. To overcome this issue we have follow two strategies: (1) ask the panel of experts and collaborators to estimate a date in which the endorsements were created and (2) estimate the time of the endorsement by using the join data in the social network.

(iii) The LinkedIn API also provides a level of proficiency for each skill. Nevertheless these features cannot be used since it is not available in all skills and it is based on a particular taxonomy.

(iv) Finally, in order to access all required information, an URL to query the official LinkedIn REST API (https://developer.linkedin.com/apis) was designed as shown in Listing 2. Nevertheless and due to privacy setting of the API, every participant in the experiments was asked to execute this request through the APIgee service (https://apigee.com/console/linkedin) using their own OAuth credentials and to send us the request's output as XML. Through this request the user will be asked to grant access to their full profile. Then, all public personal information, a profile containing: first name, last name, headline, industry, location, number of connections, summary, specialties, positions, associations, honors, interests, publications, patents, languages, skills (id, proficiency, and years), certifications, education, courses, volunteer, three-current-positions, number of recommenders, and connections (and their full profile) will be gathered using the LinkedIn REST API.

Table 2: Set of selected skills S_t.

Id	Skill
s_1	Java
s_2	Python
s_3	Data mining
s_4	UML
s_5	MySQL
s_6	CMMI
s_7	Sales management
s_8	Negotiation
s_9	Technical management
s_{10}	Business management

4.2. Results and Discussion. After the execution of the different techniques, the averaged measures (for all skills) and for every user in dataset $C_{\text{synthetic}}$ are presented in Table 4. Obviously, the first technique based on the number of times a user has been endorsed is not actually relevant in terms of quality as results show. On the other hand, the HITS algorithm provides better results in terms of precision but the drawbacks of this algorithm (not considering time as a relevant variable—see Section 3.1) implies a low-precision in some users with a behavior close to the frequency-based technique. As an improvement or more accurate version of the HITS algorithm, the SPEAR technique seems to get better results that are closer to the expert's opinion. Here, it is clear that the assumption of time as a key-variable to assess quality is a determinant to detect the level of expertise. Finally, the Skillrank technique that previously configures the level of

Table 3: The Individual Competency Index (ICI).

Id	Conceptual knowledge	Description
l_0	None	Level 0 denotes a lack of competence in a specific area or topic.
l_1	Basic	Level 1 denotes an understanding of fundamentals and some initial practical application.
l_2	Intermediate	Level 2 denotes a solid conceptual understanding and some practical application.
l_3	Advanced	Level 3 denotes significant conceptual knowledge and practical experience in performing a competency to a consistently high standard.
l_4	Expert	Level 4 denotes extensive knowledge, refined skill, and prolonged experience in performing a defined competency at the highest standard.

Table 4: Aggregated measures S_t in dataset $C_{\text{synthetic}}$.

User	$F^{s_k}_{u_k}$			HITS			SPEAR			Skillrank		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
u_1	0.43	0.80	0.56	0.67	0.79	0.73	0.73	0.87	0.79	0.80	0.81	0.80
u_2	0.25	0.89	0.39	0.65	0.78	0.71	0.90	0.71	0.79	0.65	0.72	0.68
u_3	0.44	0.83	0.58	0.63	0.82	0.71	0.82	0.80	0.81	0.85	0.86	0.85
u_4	0.52	0.74	0.61	0.63	0.86	0.73	0.83	0.86	0.84	0.86	0.77	0.81
u_5	0.48	0.84	0.61	0.56	0.79	0.66	0.67	0.84	0.75	0.77	0.77	0.77
u_6	0.43	0.79	0.56	0.53	0.78	0.63	0.89	0.72	0.80	0.84	0.89	0.86
u_7	0.35	0.71	0.47	0.53	0.89	0.66	0.82	0.82	0.82	0.86	0.85	0.85
u_8	0.46	0.84	0.59	0.74	0.88	0.80	0.80	0.86	0.83	0.80	0.84	0.82
u_9	0.45	0.73	0.56	0.74	0.87	0.80	0.82	0.84	0.83	0.66	0.90	0.76
u_{10}	0.29	0.77	0.42	0.68	0.89	0.77	0.69	0.90	0.78	0.78	0.78	0.78
Average	0.41	0.79	0.54	0.64	0.84	0.72	0.80	0.82	0.80	0.79	0.82	0.80

expertise of every user before making endorsement seems to have a similar behavior to the SPEAR algorithm. Although in some cases there is a relevant gain, the truth is that values in both techniques are very similar and to actually assert that Skillrank is better than the simple version of the SPEAR technique more data should be used.

Following this discussion, Table 5 shows the results of the different techniques using real data. In general, a decrease of precision can be found in this table with regards to previous results. This can be explained due to the fact that this dataset is not customized and real behavior of users and skills is found implying, in general, worse results.

As final remark, a change in the parameters such as the set of users and skills could lead us to get better results. Nevertheless, this initial effort will be used as a baseline to compare further improvements. Regarding similar approaches that have been implemented, as the related work section has outlined, the presented approach is closely related to [26] since the same problem is being addressed. The main difference is that they have also outlined a functional architecture while we focus here in addressing some of the existing open issues: alignment of the skills quality to an existing competency index. On the other hand, we have tried to reuse the most existing techniques. That is why we have made use of the well-known techniques such as the HITS and SPEAR algorithms that have been demonstrated to detect experts under certain characteristics of a graph. The Skillrank technique gets inspiration of these techniques to adapt the underlying concepts and execution steps to the problem of quality assessment of skills available in online profiles.

5. Conclusions and Future Work

The present paper has introduced different techniques to assess quality in graph-based structures. The well-known algorithms HITS and SPEAR have been also presented as inspiration for the Skillrank technique. This approach reinterprets the notions and underlying concepts of the SPEAR algorithm to apply them to the context of skills quality assessment in professional social networks. On the other hand two main experiments have been conducted using synthetic and real data to evaluate the behavior of the aforementioned techniques in terms of precision and recall. Both approaches—the SPEAR and Skillrank algorithms—have shown similar results in the test datasets implying that these techniques can be meaningfully applied to assess quality of skills. From another perspective the quality assessment of user profiles and more specifically user skills is an active research area that ranges from applying expertise retrieval techniques to expertise profiling, topic extraction, and so forth. In this sense there are still some open issues that must be tackled in order to provide automatic methods for user profiling, talent hunter or expert finding processes. Currently, a lot of professional social networks are emerging but the problem of creating groups of users by a certain topic is becoming a major challenge since it is necessary to improve trust and provenance of information or user's activities. The relevance of this work is that it can serve to manage enterprise know-how and to detect experts inside organizations. Their competitiveness can be increased by better human resources management processes that could

```
https://api.LinkedIn.com/v1/people/~/connections:(id,first-name,last-name,formatted-name,email-address,headline,industry,
location,num-connections,summary,specialties,positions,site-standard-profile-request,public-profile-url,api-standard-profile-
request,proposal-comments,associations,honors,interests,publications,patents,languages,skills:(id,skill,proficiency,years),
certifications,educations,courses,volunteer,three-current-positions,num-recommenders,following,job-bookmarks,date-of-
birth,member-url-resources,connections)
```

Listing 2: URL to extract user data from the LinkedIn API.

Table 5: Aggregated measures S_t in dataset C_{Linkedin}.

User	$F^{s_k}_{u_k}$			HITS			SPEAR			Skillrank		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
u_1	0.39	0.68	0.50	0.53	0.68	0.60	0.66	0.71	0.68	0.79	0.79	0.79
u_2	0.22	0.80	0.35	0.59	0.69	0.64	0.56	0.83	0.67	0.77	0.75	0.76
u_3	0.40	0.60	0.48	0.40	0.70	0.51	0.58	0.87	0.70	0.78	0.89	0.83
u_4	0.23	0.80	0.36	0.55	0.60	0.57	0.75	0.70	0.72	0.67	0.71	0.69
u_5	0.40	0.60	0.48	0.43	0.76	0.55	0.74	0.82	0.78	0.84	0.89	0.86
u_6	0.33	0.69	0.45	0.63	0.68	0.65	0.53	0.77	0.63	0.67	0.89	0.76
u_7	0.50	0.78	0.61	0.67	0.62	0.64	0.64	0.79	0.71	0.71	0.76	0.73
u_8	0.27	0.60	0.37	0.60	0.73	0.66	0.52	0.88	0.65	0.75	0.87	0.81
u_9	0.30	0.67	0.41	0.66	0.69	0.67	0.81	0.80	0.80	0.81	0.84	0.82
u_{10}	0.45	0.80	0.58	0.63	0.76	0.69	0.72	0.89	0.80	0.73	0.81	0.77
Average	0.35	0.70	0.46	0.57	0.69	0.62	0.65	0.81	0.71	0.75	0.82	0.78

take advantage of exploiting existing data generated by tracking user's activities. From a technical point of view Skillrank is a first substantial effort (including a good number of "artisan" tasks) and new capabilities such as including new data sources to assess skills quality, use of more advanced ordered weighted averaging (OWA) operators and adaptation of other datasets for experimenting purposes should be added in the future as well as new variables in the core of the algorithm. Finally, we also plan to release the information of our experiments using some existing standard such as nanopublications to ease the reuse and comparison with new techniques.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This wok has been partially supported by the European Commission (programme LifeLong Learning—action Leonardo da Vinci—Transfer of Innovation) through project "ECQA Certified Social Media Networker Skills" (2011-1-ES1_LEO05-35930), by the Spanish Ministry of Economy and Competitiveness through INNPACTO project Post-Via 2.0 (IPT-2011-0973-410000), by the German Federal Ministry of Education and Research trough project "OpSIT—Optimaler Einsatz von Smart-Items-Technologien in der Stationären Pflege" (16SV6048), and by the German Federal Ministry of Economics and Technology through project "PrevenTAB" (KF3144902DB3).

References

[1] F. Bonchi, C. Castillo, A. Gionis, and A. Jaimes, "Social network analysis and mining for business applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 22, 2011.

[2] M. Neshati, D. Hiemstra, E. Asgari, and H. Beigy, "Integration of scientific and social networks," *World Wide Web*, vol. 17, no. 5, pp. 1051–1079, 2014.

[3] K. Musiał and P. Kazienko, "Social networks on the Internet," *World Wide Web*, vol. 16, no. 1, pp. 31–72, 2013.

[4] X. Tang and C. C. Yang, "Ranking user influence in healthcare social media," *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 4, article 73, pp. 1–21, 2012.

[5] T. Schleyer, B. S. Butler, M. Song, and H. Spallek, "Conceptualizing and advancing research networking systems," *ACM Transactions on Computer-Human Interaction*, vol. 19, no. 1, article 2, 2012.

[6] R. Colomo-Palacios, E. Tovar-Caro, Á. García-Crespo, and J. M. Gómez-Berbís, "Identifying technical competences of IT professionals: the case of software engineers," *International Journal of Human Capital and Information Technology Professionals*, vol. 1, no. 1, pp. 31–43, 2010.

[7] Y. Xu, X. Guo, J. Hao, J. Ma, R. Y. K. Lau, and W. Xu, "Combining social network and semantic concept analysis for personalized academic researcher recommendation," *Decision Support Systems*, vol. 54, no. 1, pp. 564–573, 2012.

[8] D. W. McDonald and M. S. Ackerman, "Expertise recommender: a flexible recommendation system and architecture," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 231–240, New York, NY, USA, December 2000.

[9] R. Berendsen, M. de Rijke, K. Balog, T. Bogers, and A. van den Bosch, "On the assessment of expertise profiles," *Journal of the American Society for Information Science and Technology*, vol. 64, no. 10, pp. 2024–2044, 2013.

[10] G. A. Wang, J. Jiao, A. S. Abrahams, W. Fan, and Z. Zhang, "ExpertRank: a topic-aware expert finding algorithm for online knowledge communities," *Decision Support Systems*, vol. 54, no. 3, pp. 1442–1451, 2013.

[11] R. Colomo-Palacios, I. González-Carrasco, J. L. López-Cuadrado, A. Trigo, and J. E. Varajao, "I-Competere: using applied intelligence in search of competency gaps in software project managers," *Information Systems Frontiers*, vol. 16, no. 4, pp. 607–625, 2014.

[12] A. Capiluppi, A. Serebrenik, and L. Singer, "Assessing technical candidates on the social web," *IEEE Software*, vol. 30, no. 1, pp. 45–51, 2013.

[13] M. Taylor and D. Richards, "Finding and validating expertise," in *Proceedings of the 19th European Conference on Information Systems (ECIS '11)*, Helsinki, Finland, 2011.

[14] E. Ben Ahmed, A. Nabli, and F. Gargouri, "Group extraction from professional social network using a new semi-supervised hierarchical clustering," *Knowledge and Information Systems*, vol. 40, no. 1, pp. 29–47, 2014.

[15] V. Boeva, M. Krusheva, and E. Tsiporkova, "Measuring expertise similarity in expert networks," in *Proceedings of the 6th IEEE International Conference Intelligent Systems (IS '12)*, pp. 53–57, Sofia, Bulgaria, September 2012.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: bringing order to the web," Stanford Digital Library Technologies Project, 1998.

[17] J. M. Kleinberg, "Hubs, authorities, and communities," *ACM Computing Surveys*, vol. 31, no. 4, article 5es, 1999.

[18] C. A. Yeung, M. G. Noll, N. Gibbins, C. Meinel, and N. Shadbolt, "Spear: spamming-resistant expertise analysis and ranking in collaborative tagging systems," *Computational Intelligence*, vol. 27, no. 3, pp. 458–488, 2011.

[19] M. G. Noll, C.-M. Au Yeung, N. Gibbins, C. Meinel, and N. Shadbolt, "Telling experts from spammers: expertise ranking in folksonomies," in *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*, pp. 612–619, July 2009.

[20] B. Hoppe and C. Reinelt, "Social network analysis and the evaluation of leadership networks," *The Leadership Quarterly*, vol. 21, no. 4, pp. 600–619, 2010.

[21] T. Lappas, K. Liu, and E. Terzi, "A survey of algorithms and systems for expert location in social networks," in *Social Network Data Analytics*, C. C. Aggarwal, Ed., pp. 215–241, Springer, New York, NY, USA, 2011.

[22] X. Liu, G. A. Wang, A. Johri, M. Zhou, and W. Fan, "Harnessing global expertise: a comparative study of expertise profiling methods for online communities," *Information Systems Frontiers*, vol. 16, no. 4, pp. 715–727, 2014.

[23] J. J. Jung and P. Kazienko, "Advances on social network applications," *Journal of Universal Computer Science*, vol. 18, no. 4, pp. 454–456, 2012.

[24] M. Sanderson, "Test collection based evaluation of information retrieval systems," *Foundations and Trends in Information Retrieval*, vol. 4, no. 4, pp. 247–375, 2010.

[25] K. Hofmann, S. Whiteson, and M. D. Rijke, "Fidelity, soundness, and efficiency of interleaved comparison methods," *ACM Transactions on Information Systems*, vol. 31, no. 4, article 17, pp. 1–43, 2013.

[26] T. Haselmann, A. Winkelmann, and G. Vossen, "Towards a conceptual model for trustworthy skills profiles in online social networks," in *Information Systems Development*, J. Pokorny, V. Repa, K. Richta et al., Eds., pp. 285–296, Springer, New York, NY, USA, 2011.

[27] M.-E. Del-Moral Pérez and A.-P. Guzmán-Duque, "CityVille: collaborative game play, communication and skill development in social networks," *Journal of New Approaches in Educational Research*, vol. 3, no. 1, pp. 11–19, 2014.

[28] A. Cabrales, A. Calvó-Armengol, and Y. Zenou, "Social interactions and spillovers," *Games and Economic Behavior*, vol. 72, no. 2, pp. 339–360, 2011.

[29] Y. P. Hsieh, "Online social networking skills: the social affordances approach to digital inequality," *First Monday*, vol. 17, no. 4, 2012.

[30] G. E. Bolton, E. Katok, and A. Ockenfels, "How effective are electronic reputation mechanisms? An experimental investigation," *Management Science*, vol. 50, no. 11, pp. 1587–1602, 2004.

[31] D.-R. Liu, Y.-H. Chen, W.-C. Kao, and H.-W. Wang, "Integrating expert profile, reputation and link analysis for expert finding in question-answering websites," *Information Processing and Management*, vol. 49, no. 1, pp. 312–329, 2013.

[32] J. M. Alvarez-Rodríguez, J. E. L. Gayo, and P. O. de Pablos, "An extensible framework to sort out nodes in graph-based structures powered by the spreading activation technique: the ONTOSPREAD approach," *International Journal of Knowledge Society Research*, vol. 3, no. 4, pp. 57–71, 2012.

[33] J. M. Alvarez, L. Polo, W. Jimenez, P. Abella, and J. E. Labra, "Application of the spreading activation technique for recommending concepts of well-known ontologies in medical systems," in *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM-BCB '11)*, pp. 626–635, August 2011.

[34] E. Amigó, D. Spina, B. Beotas, and J. Gonzalo, "Towards an evaluation framework for topic extraction systems for online reputation management," in *Proceedings of the 1st Workshop on Dynamic Networks and Knowledge Discovery (DyNaK '10)*, pp. 101–111, September 2010.

[35] A. G. West, A. J. Aviv, J. Chang et al., "QuanTM: a quantitative trust management system," in *Proceedings of the 2nd European Workshop on System Security (EUROSEC '09)*, pp. 28–35, Nuremberg, Germany, March 2009.

[36] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, and L. Si, "Expertise retrieval," *Foundations and Trends in Information Retrieval*, vol. 6, no. 2-3, pp. 127–256, 2012.

[37] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci, "Choosing the right crowd: expert finding in social networks," in *Proceedings of the 16th International Conference on Extending Database Technology (EDBT '13)*, pp. 637–648, Genoa, Italy, March 2013.

[38] Ç. Aslay, N. O'Hare, L. M. Aiello, and A. Jaimes, "Competition-based networks for expert finding," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*, pp. 1033–1036, New York, NY, USA, August 2013.

[39] J. Rybak, K. Balog, and K. Nørvåg, "Temporal expertise profiling," in *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings*, vol. 8416 of *Lecture Notes in Computer Science*, pp. 540–546, 2014.

[40] S. Fortunato and A. Lancichinetti, "Community detection algorithms: a comparative analysis: invited presentation, extended abstract," in *Proceedings of the 4th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'09)*, p. 27, 2009.

[41] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: inferring user profiles in online social networks," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM '10)*, pp. 251–260, February 2010.

[42] D. U. Patton, R. D. Eschmann, and D. A. Butler, "Internet banging: new trends in social media, gang violence, masculinity and hip hop," *Computers in Human Behavior*, vol. 29, no. 5, pp. A54–A59, 2013.

[43] A. Ortigosa, R. M. Carro, and J. . Quiroga, "Predicting user personality by mining social interactions in Facebook," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 57–71, 2014.

[44] C.-C. Musat, B. Faltings, and P. Roussille, "A model of online social interactions based on sentiment analysis and content similarity," *HUMAN* , vol. 2, no. 1, pp. 55–66, 2013.

[45] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F. J. García-Peñalvo, and E. Tovar-Caro, "Competence gaps in software personnel: a multi-organizational study," *Computers in Human Behavior*, vol. 29, no. 2, pp. 456–461, 2013.

[46] F. J. García-Peñalvo, R. Colomo-Palacios, and M. D. Lytras, "Informal learning in work environments: training with the Social Web in the workplace," *Behaviour and Information Technology*, vol. 31, no. 8, pp. 753–755, 2012.

[47] R. Colomo-Palacios, I. González-Carrasco, J. L. López-Cuadrado, and Á. García-Crespo, "Resyster: a hybrid recommender system for scrum team roles based on fuzzy and rough sets," *International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 4, pp. 801–816, 2012.

[48] P. Kazienko, N. Szozda, T. Filipowski, and W. Blysz, "New business client acquisition using social networking sites," *Electronic Markets*, vol. 23, no. 2, pp. 93–103, 2013.

[49] V. Stantchev and G. Tamm, "Reducing information asymmetry in cloud marketplaces," *International Journal of Human Capital and Information Technology Professionals*, vol. 3, no. 4, pp. 1–10, 2012.

[50] L. Li, Y. Shang, and W. Zhang, "Improvement of HITS-based algorithms on web documents," in *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, pp. 527–535, New York, NY, USA, May 2002.

[51] A. Hannak, P. Sapiezynski, A. M. Kakhki et al., "Measuring personalization of web search," in *Proceedings of the 22nd international conference on World Wide Web (WWW '13)*, pp. 527–538, 2013.

[52] B. Succar, W. Sher, and A. Williams, "An integrated approach to BIM competency assessment, acquisition and application," *Automation in Construction*, vol. 35, pp. 174–189, 2013.

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 450215, 9 pages
http://dx.doi.org/10.1155/2015/450215

Research Article

On Efficient Link Recommendation in Social Networks Using Actor-Fact Matrices

Michał Ciesielczyk, Andrzej Szwabe, and Mikołaj Morzy

Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland

Correspondence should be addressed to Mikołaj Morzy; mikolaj.morzy@put.poznan.pl

Received 28 February 2014; Revised 21 November 2014; Accepted 21 November 2014

Academic Editor: Reda Alhajj

Link recommendation is a popular research subject in the field of social network analysis and mining. Often, the main emphasis is put on the development of new recommendation algorithms, semantic enhancements to existing solutions, design of new similarity measures, and so forth. However, relatively little scientific attention has been paid to the impact that various data representation models have on the performance of recommendation algorithms. And by performance we do not mean the time or memory efficiency of algorithms, but the precision and recall of recommender systems. Our recent findings unanimously show that the choice of network representation model has an important and measurable impact on the quality of recommendations. In this paper we argue that the computation quality of link recommendation algorithms depends significantly on the social network representation and we advocate the use of actor-fact matrix as the best alternative. We verify our findings using several state-of-the-art link recommendation algorithms, such as SVD, RSVD, and RRI using both single-relation and multirelation dataset.

1. Introduction

Link recommendation, along with link prediction, is a popular research topic in the domain of social network analysis and mining [1]. Numerous algorithms have been proposed over the years [2]. The main objective of link recommendation and prediction is to predict, based on the historical data, unobserved relationships and interactions between actors of a social network [3]. It should be stressed here that the term "link" is used here freely, as the task can refer to predicting possible (existing or future) relationships between people, recommending interesting resources to actors of the network, or discovering latent similarities between objects. Usually, a distinction is drawn between link prediction (where the task is to evaluate the probability of a given relationship's existence between actors) and link recommendation (where the task is to select top k resources relevant to a given actor). One can see however that it is relatively easy to combine the two tasks under a single framework. For the sake of brevity we will refer to both problems as "link recommendation" throughout this paper. Link recommendation is predicated on the existence of data, either panel data or event data [4]. Panel data refer to snapshots of the social network taken at certain intervals and representing possibly a coarse-grained view of existing relationships. In contrast, event data refer to detailed records of activities between actors in the network. Event data is time-stamped and fine-grained and often results from automated measurements or transactions. These two types of data are merged and processed and split into a training set and a test set for the purpose of training of link recommendation models.

Although link recommendation tasks have attracted significant attention of the scientific community over the last years, in our opinion relatively little work has been done on the impact of data representation models on the quality of recommendations. By far the most popular data representation model is an actor-object matrix, where actors of the social network are represented as rows and objects that are the subject of recommendations are represented as columns. The cells of such matrix may contain either binary flags to denote the existence of a relation (e.g., Adam likes "The Police"), or a value of the relation, both discrete and numerical (e.g., Beth ate at "Pizza Paradise" and rated it with 4.5 stars). One may note that the social network need not be

a bipartite graph. When the relation is defined between actors (e.g., Carol likes Douglas), the actor-object matrix becomes simply a square matrix. The situation becomes slightly more complex in case of multirelational social networks, where multiple different relations, of possibly varying semantics, may exist between actors in the network. A typical example is a network where actors may express both fondness of and rejection of certain objects (e.g. Eve likes to watch comedy movies but she hates horror movies). If the storage of relation values is permitted by a given data model, multirelational networks may be modeled by assigning distinct values (or sets of values) to particular relations, but for a binary actor-object matrix it is necessary to represent each relation by a separate matrix and to include processing of multiple matrices by the recommendation algorithm.

In this paper we argue that actor-object matrix is not the optimal data model for recommendation algorithms. Our experiments conclusively show that transformation from the actor-object to the actor-fact matrix improves recommendation quality significantly, as measured by the popular "area under receiver-operator characteristic curve" (AUROC) measure. We perform extensive experiments on a large real-world dataset to support our claims. Given the fact that the vast majority of link recommendation algorithms for social networks compute actor-object, actor-actor, or object-object similarities by applying linear algebra on data representation matrices, the superiority of actor-fact matrix representation becomes quite obvious (in particular for methods which are generally based on singular value decomposition paradigm). The original contribution of this paper consists in the introduction of two elements:

(i) a data representation method based on a binary actor-fact matrix,

(ii) a similarity quasimeasure based on the 1-norm length of the Hadamard product of the given tuple of vectors.

Our key finding is that the proposed data representation and the new similarity measure, when combined with reflexive matrix processing, significantly outperform state-of-the-art collaborative filtering methods based on the use of a standard actor-object matrix.

Our paper is organized as follows. In Section 2 we report on the related work on the subject and we present the referenced recommendation algorithms. Section 3 introduces the concept of the actor-fact matrix. In Section 4 we present the evaluation methodology of the actor-fact matrix representation and we report the results of conducted experiments in Section 5. The paper concludes in Section 6 with a brief summary.

2. Related Work

By far the most popular approach to link recommendation in social networks is collaborative filtering using an input matrix which represents each actor as a vector in the space of objects and each object as a vector in the space of actors. Many previous works consider building a model of collaborative similarity from a model of content-based interobject relations to be the most promising hybrid link recommendation technique [5, 6]. As far as algebraic representations of graph data is concerned, the actor-fact matrix model is similar to the model described in [7]. Indeed, our model was inspired by the semantic data model of RDF triples. Also, as far as the algebraic transformation of the graph data is concerned, the model presented in this paper may be regarded as similar to RDF data search methods which are based on spreading activation realized by means of iterative matrix data processing [8] or single multiplication by a random projection matrix [7]. However, the latter method is limited to the RDF graph node search using a traditional bilateral similarity measure, whereas we extend the model by using a vector-space quasisimilarity measure which allows to efficiently compute the likelihood of an unknown relationship.

In our evaluation we use three main types of collaborative filtering recommender algorithms. The baseline is established by a simple popularity-based algorithm favoring objects having the highest number of positive relationships in the train set [9]. Next, we have employed several different approaches to the input matrix decomposition. Firstly, we have used the algorithm based on reflexive random indexing [10]. Secondly, we have used two types of algorithms that are based on the singular value decomposition: a traditional implementation of the method (PureSVD), in which actor vectors are represented as combinations of object vectors without any specific parameterization, and an implementation of the randomized singular value decomposition (RSVD) [11], which is a combination of the reflexive random indexing and SVD. We have chosen so since SVD-based methods have been long considered to be the most efficient recommender engines in real world settings [12–15].

Section 5 presents the results of conducted experiments. Since our data have the form of binary prepositions (i.e. our social network is a signed network), the evaluation of the proposed method is oriented on the task of finding relevant links [16] rather than on the minimization of recommendation rating error. Classification metrics, such as area under ROC (AUROC), measure the probability of making correct or incorrect decisions by the recommender algorithm about whether an object is relevant. Moreover, classification metrics tolerate the differences between actual and predicted values, as long as they do not lead to wrong decisions. Thus, these metrics are appropriate to examine binary relevance relationships. In particular, while using AUROC it is assumed that the ordering among relevant items does not matter. According to [17], AUROC is equivalent to the probability of the system being able to choose properly between two objects, one randomly selected from the set of relevant objects and one randomly selected from the set of nonrelevant objects. For this reason, the results of the theoretical research are evaluated by means of experiments based on quality measures that are probabilistically interpretable such as AUROC.

3. Actor-Fact Matrix

Let us recall that our model is influenced by the semantic model of RDF triples. Each RDF triple combines information about the predicate that relates a subject to an object. We

consider a generic social network (for simplicity we constrain ourselves to nonvalued relations, but the proposed method may be easily extended to valued relations) which conceptually consists of a set of actors $A = \{a_1, \ldots, a_n\}$, a set of objects $O = \{o_1, \ldots, o_m\}$, and a set of relations $R = \{R_1, \ldots, R_l\}$, where each relation R_i represents a function $R_i : A \times O \to \{0, 1\}$. Let us now combine all actors, objects, and possible predicates into a single set $E = A \cup O \cup R$. Furthermore, let $|A| = n$, $|O| = m$, and $|R| = l$. Of course, there is no requirement to have the set of actors be separate from the set of objects; that is, in general it is possible that $A \subseteq O$. It should be noted though that if sets A and O would overlap, that is, if they would be represented by the same vectors, it would not be possible to take advantage of the semantics of actors constituting relationships. In other words, putting actors and objects together into a single set would make it impossible to distinguish between semantically correct relationships, such as "Alice likes apples," and semantically incorrect relationships, such as "apples like Alice." Being able to encode such semantics directly in social network matrix representation is obviously a very desirable property, but this issue is out of the scope of this paper.

We refer to the set of actual instances of relations as the set of *facts* denoted by F, and let $|F| = f$. The binary actor-fact matrix is defined as $X = [x_{i,j}]_{(n+m+l)\times f}$, where each column of the matrix X represents a single fact (i.e., an existing dyad connected in the social network by a relation), each row of the matrix X represents an entity (actor, object, or relation), and each column contains exactly three nonzero entries, that is, for each j there exist exactly three nonzero entries $x_{i_1,j}$, $x_{i_2,j}$, and $x_{i_3,j}$, such that $1 \le i_1 \le n$, $n+1 \le i_2 \le n+m$, and $n+m+1 \le i_3 \le n+m+l$ (the rows containing these three nonzero entries correspond to the actor, object, and relation of a given dyad, or, in the RDF parlance, to the subject, predicate, and the object of a triple). At the same time the number of nonzero entries in each row represents the number of dyads in which a given actor/object participates, or the number of dyads of a given relation.

Let us consider a simple social network depicted in Figure 1. It represents two different relationships between actors *Alice*, *Bob*, *Titanic*, and *Star Wars*. The relationships between these actors include *liking* and *being a friend of*. Implicitly, we understand that *liking* is a relationship between an actor representing a person and an actor representing a movie, whereas *being a friend of* is a relationship between two actors representing persons. This network can be easily transformed into the actor-fact model. There are three facts that exist in this network:

(i) $fact_1$: Alice is a friend of Bob,

(ii) $fact_2$: Alice likes Titanic,

(iii) $fact_3$: Bob likes Star Wars.

In our actor-fact matrix representation $fact_1$ will constitute a column in the matrix X and this column will have nonzero entries for cells $X[Alice, fact_1]$, $X[is\ a\ friend\ of, fact_1]$, and $X[Bob, fact_1]$. The entire social network from Figure 1 is presented in the actor-fact matrix representation in Table 1.

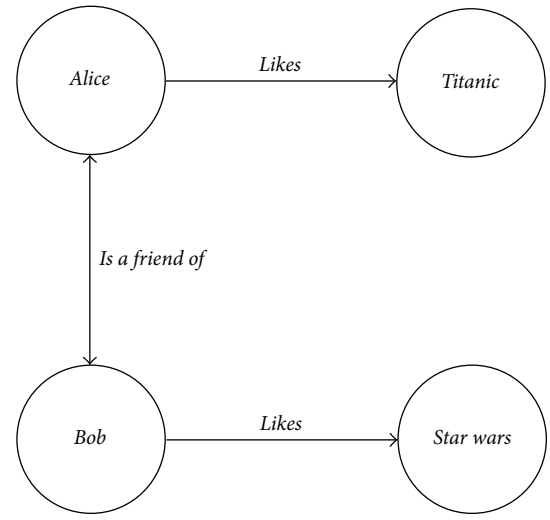


Figure 1: Example of a social network.

Table 1: Actor-fact matrix representing the network from Figure 1.

	$fact_1$	$fact_2$	$fact_3$
Alice	1	1	0
Bob	1	0	1
Is a friend of	1	0	0
Likes	0	1	1
Titanic	0	1	0
Star Wars	0	0	1

When using the actor-fact matrix as the data representation, one has to perform the prediction generation step in a special way. Initially, as in many of the most accurate collaborative filtering methods, the missing values of the input matrix are estimated. In order to achieve this, the input matrix X is processed into its reconstructed form $\widehat{X}$ using one of the evaluated recommendation algorithms. Afterwards, each of the predictions is calculated as the 1-norm length of the Hadamard product of row vectors. The Hadamard product (also known as the Schur product or the entrywise product) of two vectors $X = [x_1, \ldots, x_n]$ and $Y = [y_1, \ldots, y_n]$ of the same length n is defined as

$$X \circ Y = [x_1 \cdot y_1, \ldots, x_n \cdot y_n]. \quad (1)$$

Each dyad forms the proposition which is the subject of the likelihood estimation. More formally, the prediction value $p_{i,j,k}$ is calculated according to the formula

$$p_{i,j,k} = \left\| \widehat{x_i} \circ \widehat{x_j} \circ \widehat{x_k} \right\|_1, \quad (2)$$

where $\widehat{x_i}$, $\widehat{x_j}$, and $\widehat{x_k}$ are the row vectors of the reconstructed matrix $\widehat{X} = [\widehat{x}_{i,j}]_{(n+m+l)\times f}$ corresponding to the elements of the given dyad, and the symbol $\circ$ represents the Hadamard product.

For instance, using the proposed measure on the example shown in Table 1 one may predict the likelihood of Alice liking the movie Titanic (i.e., the likelihood of the joint

incidence of actors *Alice*, *likes*, and *Titanic* represented by row vectors $X[Alice,] = [1,1,0]$, $X[likes,] = [0,1,1]$, and $X[Titanic,] = [0,1,0]$, resp.). This likelihood equals $\|[1,1,0] \circ [0,1,1] \circ [0,1,1]\| = 1$. Conversely, the likelihood of any nonexistent fact, such as *Bob likes Titanic*, equals 0. Naturally, the practical value of such a measure is to estimate the likelihood of missing links after the application of appropriate collaborative filtering algorithms.

The proposed formula may be seen as a generalization of the dot product formula, as in the hypothetical case of measuring quasisimilarity of two (rather than three) vectors, the formula is equivalent to the dot product of the two vectors. It should be also noted that the measure may be easily extended to larger number of vectors. The interpretation of the proposed formula as the likelihood of the joint incidence of two or more facts represented as vectors is based on the quantum information retrieval model [18]. It has to be admitted that, for the methods presented in this paper, the coordinates of modeled entities' representations do not formally denote probabilities. Therefore, formally speaking, the proposed method may be regarded as a technique for providing the likelihood of the joint incidence of two or more events represented as vectors, which is inspired by the quantum information retrieval model of probability calculation.

4. Evaluation Methodology

Let us now present the evaluation methodology for the experiments. Our goal is to quantitatively compare two matrix-based methods of social network representation: the classical actor-object matrix and the new actor-fact matrix from the point of view of the link recommendation task. Taking into consideration that link recommendation tasks may vary, we have additionally considered two subproblems: a one-class link recommendation, where the aim is to discover only the missing links of a single relation (e.g., for a given actor recommend to her a set of possible new friends), and the biclass link recommendation, where the aim is to discover missing links of one particular relation while not recommending any of the links of another relation (e.g., for a given actor, show him possible friends who share theatrical preferences, but do not recommend any new movies). The combination of the two results in the following four scenarios:

(i) S1: using single relation and an actor-object matrix $B = [b_{i,j}]_{n \times m}$, where n is the number of actors and m is the number of objects, a scenario which corresponds to friend recommendations using only information on friendship between actors,

(ii) S2: using two antagonistic relations and an actor-object matrix $B = [b_{i,j}]_{n \times m}$, where n is the number of actors and m is the number of objects, a scenario which corresponds to friend recommendations using information on friendship and dislike between actors,

(iii) S3: using single relation and an actor-fact matrix $X = [x_{i,j}]_{(n+m+l) \times f}$, where n is the number of actors, m is the number of objects, and $l = 1$ is the number of predicates (in this case a single predicate),

(iv) S4: using two antagonistic relations and an actor-fact matrix $X = [x_{i,j}]_{(n+m+l) \times f}$, where n is the number of actors, m is the number of objects, and $l = 2$ is the number of predicates.

In order to evaluate the effect of data representation model on collaborative filtering methods, we have decided to use one of the most widely referenced datasets in the recommender systems area. We have deliberately chosen to turn a typical recommender system dataset into an artificial social network, instead of using a genuine network (e.g., Facebook friend graph or Twitter followers graph), because we also wanted to compare our results with previous results; thus we needed a well-established benchmark dataset. MovieLens ML100k set was collected over various periods of time from Internet users who expressed their opinions on different movies in order to receive personalized recommendations. It contains 100 000 ratings of 1682 movies given by 943 unique users. Each rating which is above the average for a given movie has been treated as an indication that a user likes the movie. Analogically, each rating below the average has been used as an indication that a user dislikes the movie. Finally, train and test data sets were generated by randomly dividing the set of all known facts into two subsets. The data were divided according to the specified training ratio, denoted by tr. To compensate for the impact that the randomness in the dataset partitioning has on the results of the presented methods, each plot in this section shows a series of values that represent averaged results of individual experiments.

As we have previously stated, four recommendation algorithms are used: a simple popularity-based method, a traditional SVD (PureSVD), a randomized version of SVD (RSVD), and a reflexive random indexing (RRI). To clarify, the actual algorithm being used is the collaborative filtering (CF), but it works on a matrix decomposed using the above algorithms. The decomposition of the original matrix is of course necessary to make collaborative filtering computation feasible in practice. In real social networks the size of the matrix (actor-object, and actor-fact in particular) is so huge that vector similarity computation in original dimensions is impossible. Each of the methods has been tested using the following parameters (where applicable):

(i) vector dimension: 256, 512, 768, 1024, 1536, 2048;

(ii) seed length: 2, 4, 8;

(iii) SVD k-cut: 2, 4, 6, 8, 10, 12, 14, 16, 20, 24.

The number of dimensions (i.e., the SVD k-cut value), which we have used in the experiments, may appear as quite small when compared to a typical LSI application scenario. This choice has been made in order to avoid overfitting, in accordance with the assumptions concerning the dimensionality reduction sensitivity presented in [14]. Moreover, it has been observed that for each investigated scenario the optimal algorithm performance was achieved for the SVD k-cut value that was less or equal to 16, so experiments for k-cuts higher than 24 were not necessary. We have also varied the number of reflections used in RRI and RSVD between 3 and 15.

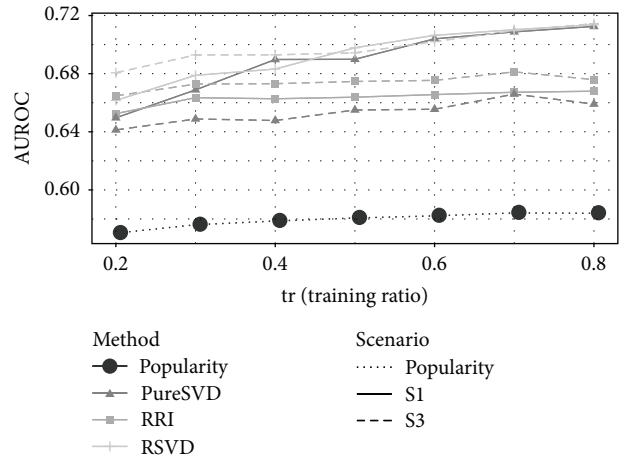


Figure 2: AUROC results for S1 and S3 scenarios.

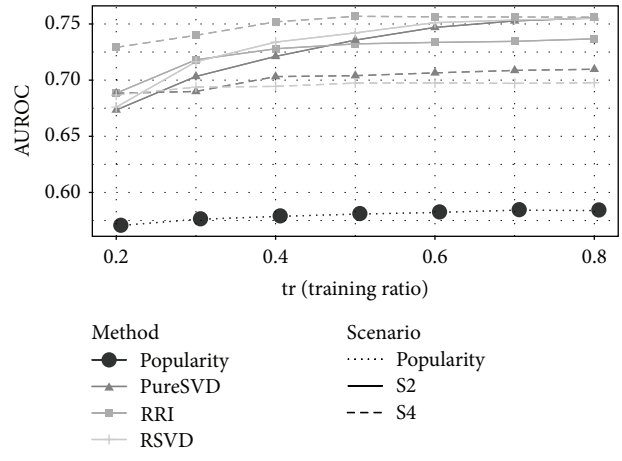


Figure 3: AUROC results for S2 and S4 scenarios.

5. Experiments

Figures 2 and 3 show a comparison of the investigated recommendation algorithms, each using either the classical actor-object or the actor-fact matrix data representation. The comparison has been performed using the AUROC measure and datasets of various sparsity. The presented results have been obtained using optimized parameters for each method and each data model. Figure 2 presents AUROC evaluation results obtained for the case of using the network consisting of a single relation (i.e., only positive ratings), whereas Figure 3 presents analogical results obtained for the case of using the full network, that is, the one containing both positive and negative relations.

As it has been confirmed experimentally, the actor-fact data representation matrix obtains recommendation quality which is higher than the analogical results obtained with the use of the classical actor-object matrix representation. It can be observed that the advantage of the proposed model is especially visible in the case of employing the full network containing both positive and negative relations, and the RRI method. Such behavior is the result of the more native ability to represent multiple relations provided by the actor-fact model.

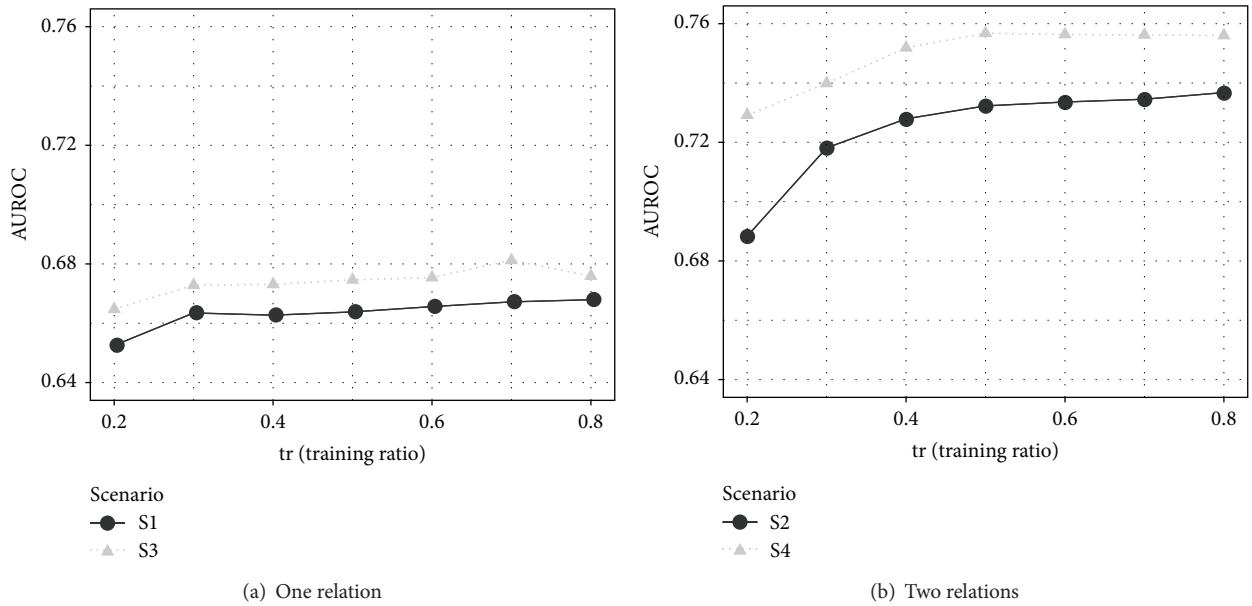
One may realize that the popularity-based algorithm, instead of modeling actors' preference profiles, simply reflects the ratio between the number of positive relation instances (hits) and negative relation instances (misses) for the most popular objects in a given network. Since a random procedure is used to divide the dataset into a train set and a test set, the values of AUROC observed for the popularity-based algorithm are almost identical for the case of both tr = 0.2 and tr = 0.8, which additionally confirms the reliability of the AUROC measurement.

In Figures 4, 5, and 6 the impact of the data representation method on the performance evaluation results is presented. As can be seen, the application of the new fact-based data representation method, accompanied with the Hadamard-based reconstruction technique, improves the results of using RRI for both single and multiple relations networks (see Figures 4(a) and 4(b)). Moreover, for the case of using the network with multiple relations, RRI outperforms any other presented method. It may be concluded that, in the context of the proposed data representation scheme, the calculation of the l-norm length of the Hadamard product is an operation that is synergic to the reflective data processing.

On the other hand, the application of the new representation method, accompanied by the reconstruction technique based on the Hadamard product, decreases the quality of results of using PureSVD for both single and multiple relation networks (see Figures 5(a) and 5(b)). The reason of such behavior is the fact that the prediction method based on the Hadamard product is not compatible with the data processing techniques based on the SVD decomposition. In the case of using the SVD dimensionality reduction, an input matrix reconstruction result should rather be used directly as the set of prediction values. The comparatively low quality of the method based on PureSVD and Hadamard product may be explained by the nonprobabilistic nature of SVD results: it is especially evident in cases when the vectors multiplied together (by means of the Hadamard product) have negative coordinates, which indicates that they obviously have no probabilistic interpretation.

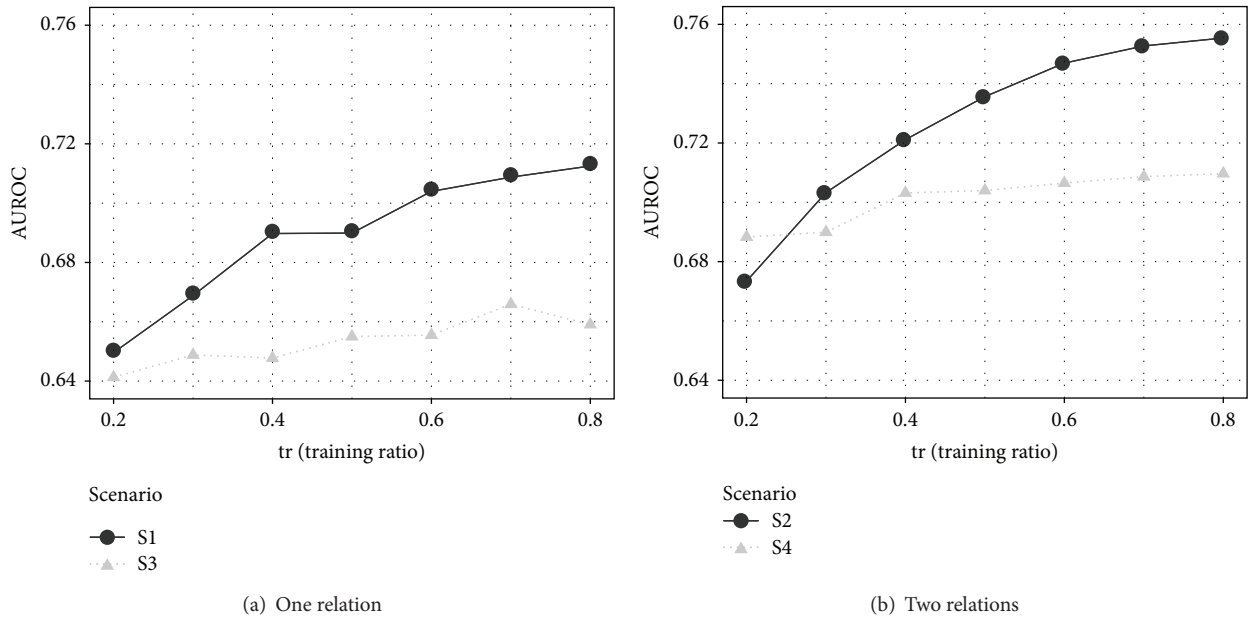
Furthermore, in the case of using RSVD (see Figures 6(a) and 6(b)), which is a combination of RI-based preprocessing and SVD-based vector space optimization, the application of the new data representation method improves the performance when single relation network is concerned (especially for small numbers of the ratio tr). On the other hand, the application of the new data representation method decreases the system performance for the multiple relations network scenario for the same reasons as in the case of using PureSVD. It may be additionally concluded that when the methods based on dimensionality reduction are used, the new representation method performs relatively (i.e., with respect to results obtained for standard representation methods) better for smaller values of the ratio tr, that is, for sparser datasets for which the recommendation task is harder.

Figure 7 presents the performance of the recommender algorithms as compared in the investigated scenarios (i.e., in scenarios S1–4). It may be concluded that, as it was already



(a) One relation (b) Two relations

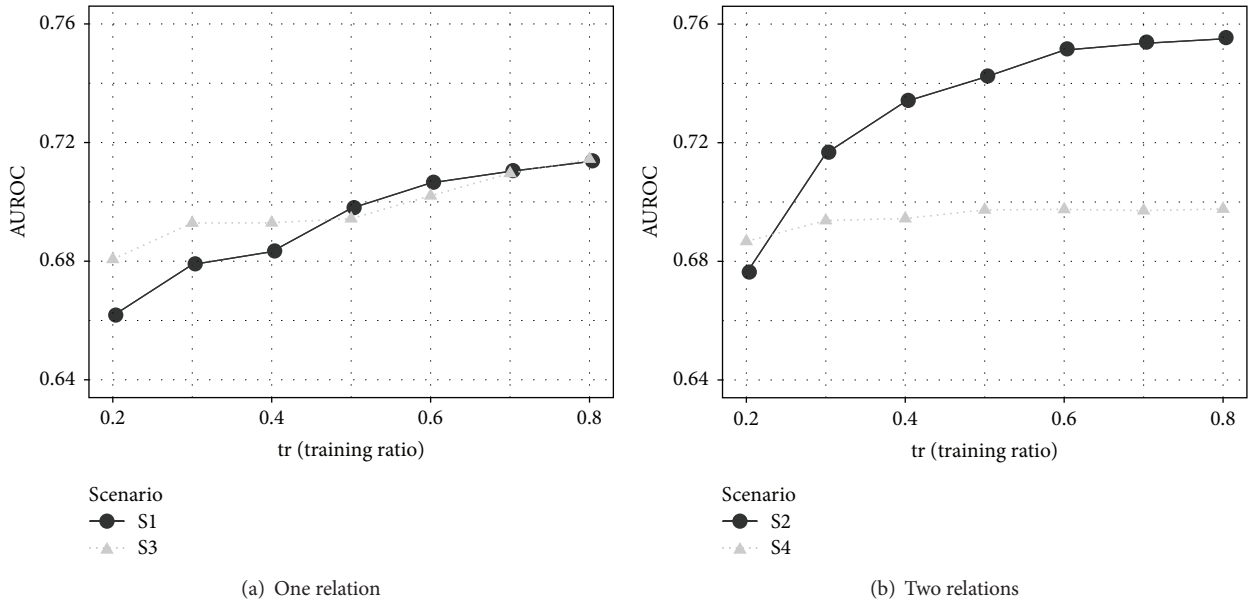
Figure 4: Impact of the input data representation method on AUROC results achieved using the RRI method.



(a) One relation (b) Two relations

Figure 5: Impact of the input data representation method on AUROC results achieved using the PureSVD method.

shown in [11], the RSVD method outperforms other methods (i.e., PureSVD and RRI) when the standard input data representation is used. As far as the S1 scenario is concerned, that is, the one with the standard data representation based on the actor-object coincidence matrix single relation, it may be seen that, in general, the decomposition-based methods (i.e., PureSVD and RSVD) achieve comparable recommendation quality and that, in general, these methods perform better than RRI (for various values of the training ratio). It may also be seen that the decomposition-based methods behave quite differently in the S3 scenario, in which the novel, fact-based data representation is used: in such case, RSVD is the method which not only outperforms all the other methods compared in the scenario (including PureSVD), but also provides a high recommendation quality for various values of the training ratio. When analyzed together, S1 and S3 scenarios show the superiority of RSVD in cases when single relation network is used. Moreover, as long as RSVD is combined with the fact-based data representation, it provides recommendation quality that is the most reliable, which is



(a) One relation

(b) Two relations

Figure 6: Impact of the input data representation method on AUROC results achieved using the RSVD method.

higher than the quality observed when any other method is used for the majority of investigated values of the training ratio. In the case of scenario S4 (fact-based data representation with multiple relations) RRI method outperforms both decomposition-based methods, which shows the compatibility of the Hadamard-based reconstruction technique with the reflective processing of multirelational data. Such combination, that is, the application of RRI together with the fact-based multirelational data representation, provides the highest recommendation quality among all the combinations presented in this paper. As the RRI method does not involve any computationally expensive spectral decomposition, this result may be very valuable from the perspective of the practical applicability of the RRI-based link recommendation systems in real-world scenarios.

The results of the experiments presented herein clearly indicate that the presence of the additional information about the negative relation improves the recommendation quality. The results for S2 and S4 scenarios (see Figure 3) are significantly better than the results obtained in S1 and S3 scenarios (see Figure 2). However, the main conclusion from the experiments is that the best quality is observed in scenario S4 (in which the proposed data representation and prediction method has been applied) for the case of the RRI-based data processing application.

The results of the comparison show that, in general, as long as the proposed multirelational actor-fact matrix data representation is used, the reflective processing methods (in particular RRI) outperform the well-known SVD-based dimensionality reduction methods. While trying to explain this observation, one may note that the typical actor-object matrix (representing only positive relations between actors and objects) is equivalent to a part of another much bigger matrix. This bigger matrix may be obtained as the result of multiplying the actor-fact matrix (with both actors and objects represented as the rows) by its transposition. The "submatrix" of the bigger matrix (together with its transposed "clone") is just a typical collaborative filtering matrix—it represents the "magnitudes" of the actor-object positive preference relation. Demonstrating this correspondence between the object-fact matrix format and widely used actor-object matrix format (typically used together with the SVD-based dimensionality reduction) requires an additional matrix multiplication (i.e., an additional reflection). Therefore, it may be expected that, as long as the proposed data representation is used, only reflective data processing methods can take full advantage of using it by applying appropriately many reflections. To put this observation (confirmed by the results of the experiments presented herein) in other words, while using the fact-based data representation, SVD-based collaborative filtering methods need at least one more matrix multiplication to provide the recommendation quality comparable to the quality achieved by means of the optimized reflective matrix processing.

6. Conclusions

The new framework proposed in this paper consists of two core elements: the new data representation method based on the actor-fact matrix and the new prediction calculation technique based on the Hadamard product of vectors. The 1-norm length of the Hadamard product vector may be seen as a natural extension of the vector dot product (in this case as a kind of group inner product of the three vectors representing the actor, the object, and the relation) whereas the dot product may be seen as an elemental step of the matrix multiplication, that is, the basic operation used in reflective matrix processing. Therefore, the calculation of the 1-norm length of the Hadamard product vector may be regarded as an operation compatible with the reflective matrix processing,

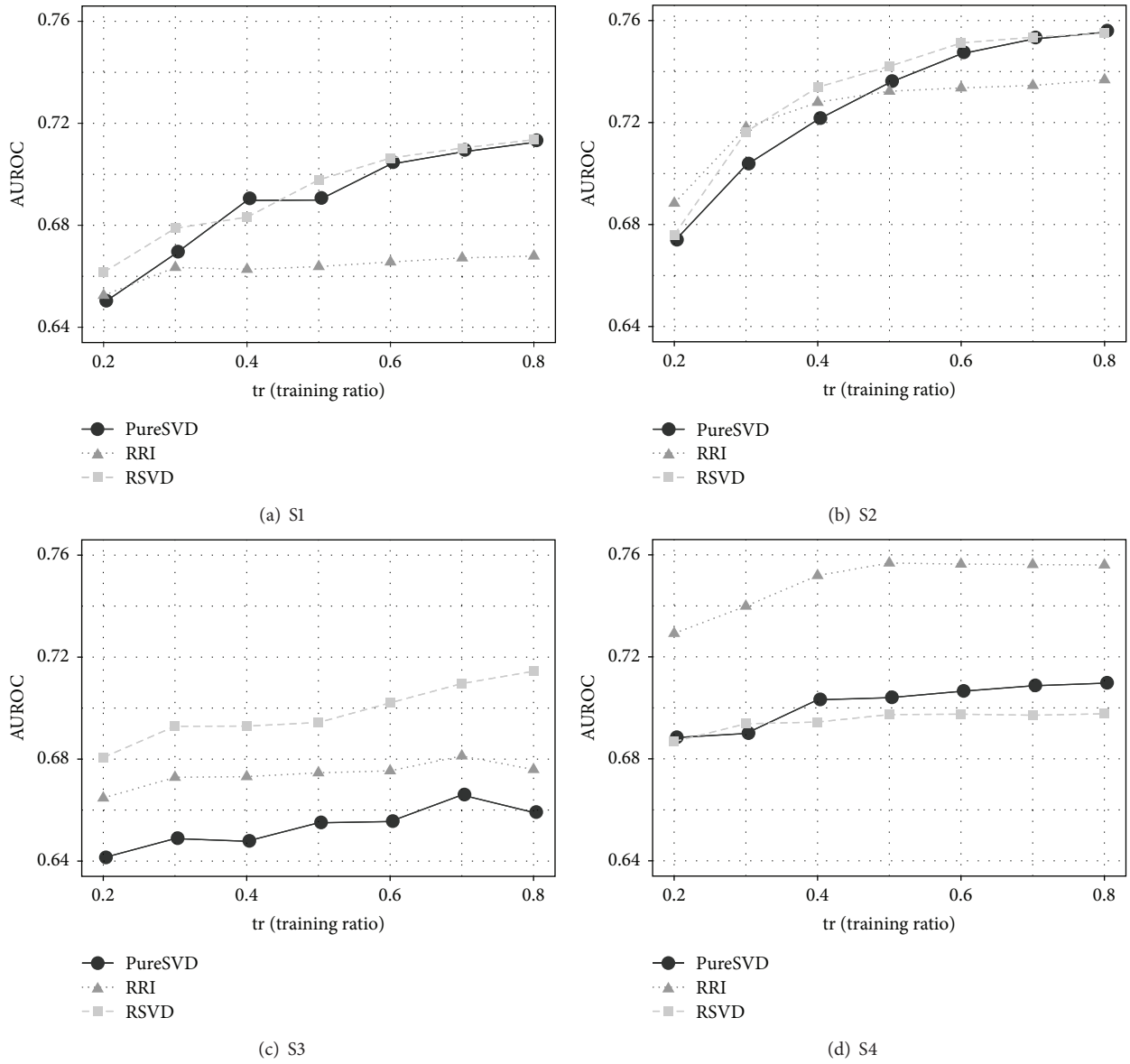


Figure 7: Comparison of all recommender algorithms for various scenarios.

seen as an "additional reflection" (i.e., the next step of the reflective data exploration process). This observation may additionally explain why the optimal number of reflections for the RRI method in the S4 scenario is relatively small (equal to 3 for each training ratio). On the other hand, the prediction based on the Hadamard product does not suit well the data processing techniques based on SVD decomposition. This explains relatively weak results of the dimensionality reduction methods in the scenarios in which the proposed data modeling method is used. In the case of using the techniques based on the dimensionality reduction, the input matrix reconstruction result is used directly as the set of the prediction values and an additional step of the Hadamard product calculation procedure is not required.

We have shown that the proposed fact-based approach to social network representation allows to improve the quality of collaborative filtering. The application of the proposed actor-fact matrix in systems featuring the most widely known methods for input data processing, such as the SVD-based dimensionality reduction and the reflective matrix processing, has been investigated. We have also shown that using the actor-fact matrix together with reflective data processing enables us to design a collaborative system outperforming systems based on the application of the dimensionality reduction techniques.

We have demonstrated the superiority of multiple matrix data reflections by realizing a new kind of spreading activation. However, the purpose of the spreading activation mechanism introduced herein is to realize the probabilistic reasoning about any fact that may be composed of actors, relations, and objects appearing in the network. To state it more precisely, in order to estimate the probability that

a given fact represents a true statement, the three constituents of the fact are independently primed. On the basis of the three independently generated vectors, each one representing levels of node activation obtained as the result of priming the node represented by the vector, the 1-norm length of a Hadamard product is applied to measure holistically (i.e., by taking into account the state of all nodes) the amount of joint similarity of the three fact constituents or, more precisely, their representations that have been obtained as the result of the spreading activation procedure.

While taking the perspective of related areas of research (such as Web scale reasoning), one may find it particularly interesting to investigate our proposal of using the 1-norm length of the Hadamard product as the measure of an unknown dyad likelihood. The authors believe that, due to probabilistic reasoning as a vector-space technique, the introduced solution provides basic means for extending the capacity for reasoning on social networks beyond the boundaries provided by currently used nonstatistical methods. In our opinion, the application of introduced methods (in particular, the new actor-fact data representation and the new Hadamard product likelihood calculation) leads to a significant link recommendation quality improvement, at least for the case of using the reflective matrix processing. Although this paper provided an evaluation using only the two relations scenario, one may also find the proposed approach to matrix-based propositional data representation to be promising from the perspective of its extendability to truly multirelational applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

Mikołaj Morzy has been supported by the National Science Centre Grant 2011/03/B/ST6/01563. Michał Ciesielczyk and Andrzej Szwabe have been supported by the National Science Centre Grant DEC-2011/01/D/ST6/06788.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[2] L. L. Linyuan and T. Zhou, "Link prediction in complex networks: a survey," *Physica A: Statistical Mechanics and Its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[3] R. Lichtenwalter and N. V. Chawla, "Link prediction: fair and effective evaluation," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '12)*, pp. 376–383, August 2012.

[4] C. Lee, B. Nick, U. Brandes, and P. Cunningham, "Link prediction with social vector clocks," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2013.

[5] B. Mobasher, X. Jin, and Y. Zhou, "Semantically enhanced collaborative filtering on the Web," in *Proceedings of the 1st European Web Mining Forum (EWMF '03)*, pp. 57–76, Springer, Cavtat-Dubrovnik, Croatia, September 2003.

[6] J. Salter and N. Antonopoulos, "CinemaScreen recommander agent: combining collaborative and content-based filtering," *IEEE Intelligent Systems*, vol. 21, no. 1, pp. 35–41, 2006.

[7] D. Damljanovic, J. Petrak, M. Lupu et al., "Random indexing for finding similar nodes within large RDF graphs," in *The Semantic Web: ESWC 2011 Workshops*, vol. 7117 of *Lecture Notes in Computer Science*, pp. 156–171, Springer, Berlin, Germany, 2012.

[8] P. Todorova, A. Kiryakov, D. Ognyano, I. Peikov, R. Velkov, and Z. Tashev, "Conclusions from experimental data and combinatorics analysis," Tech. Rep., The Large Knowledge Collider (LarKC), 2009.

[9] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proceedings of the 4th ACM Recommender Systems Conference (RecSys '10)*, pp. 39–46, ACM, September 2010.

[10] T. Cohen, R. Schvaneveldt, and D. Widdows, "Reflective Random Indexing and indirect inference: a scalable method for discovery of implicit connections," *Journal of Biomedical Informatics*, vol. 43, no. 2, pp. 240–256, 2010.

[11] M. Ciesielczyk and A. Szwabe, "RSVD-based dimensionality reduction for recommender systems," *International Journal of Machine Learning and Computing*, vol. 1, no. 2, pp. 170–175, 2011.

[12] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC '00)*, pp. 158–167, 2000.

[15] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," in *Applications of Data Mining to Electronic Commerce*, pp. 115–153, Springer, New York, NY, USA, 2001.

[16] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[17] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[18] I. Pitowsky, "Quantum mechanics as a theory of probability," in *Physical Theory and Its Interpretation*, vol. 72 of *The Western Ontario Series in Philosophy of Science*, pp. 213–240, Springer, Dordrecht, The Netherlands, 2006.

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 172879, 13 pages
http://dx.doi.org/10.1155/2015/172879

Research Article

Link Prediction Methods and Their Accuracy for Different Social Networks and Network Metrics

Fei Gao, Katarzyna Musial, Colin Cooper, and Sophia Tsoka

Department of Informatics, School of Natural and Mathematical Sciences, King's College London, Strand Campus, London WC2R 2LS, UK

Correspondence should be addressed to Katarzyna Musial; katarzyna.musial@kcl.ac.uk

Received 27 February 2014; Revised 21 November 2014; Accepted 21 November 2014

Academic Editor: Jeffrey C. Carver

Currently, we are experiencing a rapid growth of the number of social-based online systems. The availability of the vast amounts of data gathered in those systems brings new challenges that we face when trying to analyse it. One of the intensively researched topics is the *prediction of social connections between users*. Although a lot of effort has been made to develop new prediction approaches, the existing methods are not comprehensively analysed. In this paper we investigate the correlation between network metrics and accuracy of different prediction methods. We selected six time-stamped real-world social networks and ten most widely used link prediction methods. The results of the experiments show that the performance of some methods has a strong correlation with certain network metrics. We managed to distinguish "prediction friendly" networks, for which most of the prediction methods give good performance, as well as "prediction unfriendly" networks, for which most of the methods result in high prediction error. Correlation analysis between network metrics and prediction accuracy of prediction methods may form the basis of a metalearning system where based on network characteristics it will be able to recommend the right prediction method for a given network.

1. Introduction

Network structures have been studied for many years. First research in this area can be traced back to 1736 when Euler defined and solved the Seven Bridges problem of Königsberg [1]. Since then, for a long time, networks have been mainly studied by mathematicians and this resulted in a very prominent research field known today as the graph theory. There was not much ground breaking development in the complex network research area until 1960s, when the Erdos-Renyi random graph model (ER-model) was introduced [2, 3]. This is the simplest model of complex network. Due to the fact that there was a lack of large real-world data, most of the work had been done on theoretical analysis of phenomena existing in networked structures (e.g., phase transition).

Over the years data collection techniques have significantly improved our ability to store massive and heterogenous network data. During the time when ER-model was introduced, progress has also been made by sociologists in researching real-world human relationships [4, 5]. A new wave of research was set off by Watts and Strogatz who published a paper about the small-world effect in 1998 [6] and introduction of the scale-free network model by Barabási and Albert one year later [7].

As the accessibility of database systems and Internet is growing, more and more real-world network datasets are available. The available information about people and their activities is much richer and more complex than ever before. The complex network concept is an abstract form of various real-world networks, for example, biological networks such as protein-protein interaction networks, metabolic networks [8, 9], human networks and disease spread [10–13], scientific collaboration networks [14, 15], and online social networks [16–19].

Link prediction in complex network is one of the popular research topics. Most of the researchers focus on the link prediction problem [20] which is very valuable for solving real-world problems. Generally, the prediction problem is mainly studied from two angles: (i) network structure and (ii) attributes of nodes and connections. Structure refers to the way in which nodes that compose the network are interconnected. It reflects the information about network topology. Majority of the progress in the area of structure based prediction has been made by mathematicians and

physicists. Some of the well-known structure based prediction methods are Common Neighbour, Jaccard's Index, Adamic/Adar Index, Katz, and so forth (for a review of the methods please see [21]).

The link prediction problem also has been studied from the angle of the network attribute information. The attribute information refers to description of the features of nodes. Such information is difficult to show directly in the network graph. It can be for example, done by labelling nodes; for example, 1 depicts node that represents woman and 2 means that node represents man. The majority of attribute-based prediction methods follow a machine learning approach; that is, they use classification-based methods to make predictions. Widely used methods include Decision Tree, Support Vector Machine (SVM), and Naïve Bayes [22, 23]. In [24–26], authors report that the performance of link prediction improves when machine learning approaches are used. However, this is done using additional network information that is not always available. We would like to emphasize that, in our work, we are interested in the methods that only require the basic network structure information and thus we do not include machine learning methods in our study.

However, although much effort has been made, there is still no prominent prediction method that could provide a satisfactory performance. Thus, there is still a huge research gap that needs to be addressed.

1.1. Research Motivation. In the realm of network prediction, many efforts have been done on exploring new prediction methods that could provide better performance. However, the methods presented in most of the studies only improve the prediction result significantly for the network used in the study. There is a lack of systematic research that would enable to reveal the reason why the methods are good predictors when it comes to some of the networks but very bad when other networks are considered.

This paper addresses this problem, by exploring the correlation between network metrics and prediction accuracy of different methods. We expect that such approach will enable to find the reasons why methods performance varies on different networks. Apart from having a further understanding of the prediction methods, the study is also important as a theoretical base for developing new prediction methods. This could be relevant to many subjects. The prediction methods could help to find the relationships between proteins which might not be easily observed directly due to the interaction complexity. For example, new interactions can be inferred from the existing known interaction networks [27, 28] which shows a much better performance than prediction purely by chance. Online market targeting might also benefit from the network prediction which has already been applied in real-world industries. For example, Google and Amazon recommend customers the potential goods and services that they might be interested in which is a kind of link prediction that predicts the link between customers and products.

Beyond that, analysis of the link prediction problem in a time series approach could help researchers gain a better understanding of the evolution of the networks. Many works have been done to study the dynamics of complex network [29–31]. The achievement of network prediction analysis could help explain the mechanism of the network evolution.

1.2. Contributions. The main contribution of our study is that we look at the link prediction as a time series problem and systematically analysed the correlation between network metrics and methods accuracy. In addition, in our experiments, we also find that for some networks, most of the prediction methods could provide a good performance while for some other networks, most methods are relatively powerless. We name them "prediction friendly" networks and "prediction unfriendly" networks, respectively.

The paper is structured as follows. Section 2 presents the prediction methods and performance metrics used in our experiments. Section 3 presents how the dataset were selected and processed. In Sections 4 and 5 we introduce the experimental design and present obtained results. We conclude the paper in Section 6.

2. Link Prediction Problem

Link prediction problem has been extensively studied by members of the complex network community. Liben-Nowell and Kleinberg have formalised the link prediction problem in [20] in the following way.

Let $G(V, L)$ be a network within the time period of $G[t, t_1]$ where V represents the set of nodes and L represents the set of links. For the next time period $G(t_1, t_2]$, the network might change. The link prediction focuses on how to predict the evolution of links, that is, how $L_{[t,t_1]}$ will differ from $L_{(t_1,t_2]}$.

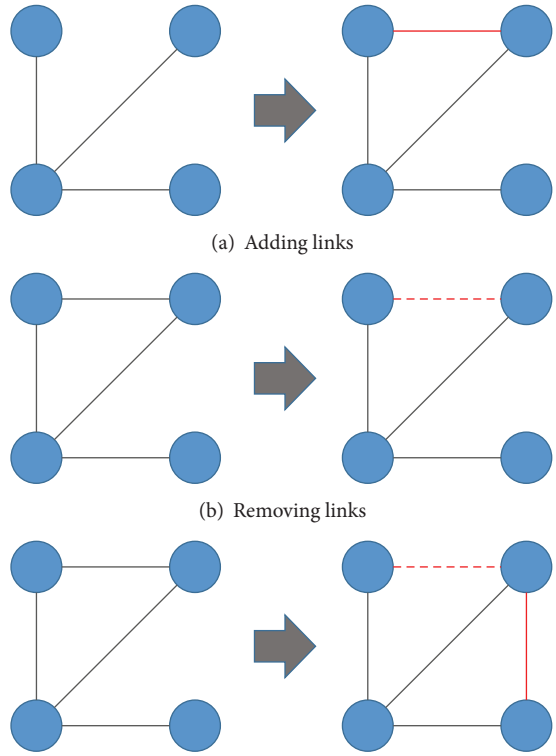
Researchers with background in physics and mathematics usually deal with the problem by focusing on the topology information of the networks. Researchers with machine learning and data mining background favour to solve the problem with considering the nodes' attribute information. There are three types of link prediction problems as shown in Figure 1: we can consider (i) only adding links to the existing network, (ii) only removing links from the existing structure, and (iii) both, adding and removing links at the same time.

Adding Links. Adding links (Figure 1(a)) means that in the next time window a new link will be created between existing nodes. There can be one or more newly created links.

Removing Links. Removing links (Figure 1(b)) means that the link will disappear in the next time window. Similar to the situation when new links are added, one or more links can be removed in one time step.

Adding and Removing Links. This problem is the combination of two previously described problems. It means that from one time window to another both appearance and disappearance of links can be predicted (Figure 1(c)).

In this research, we will only focus on the first type of link prediction problem which only aims at predicting the appearance of links. The main reason for this is that the vast majority of existing methods for real-world data focus on this problem, so it means that we have big enough base to perform correlation analysis.



(a) Adding links

(b) Removing links

(c) Adding and removing links

Figure 1: Link prediction problems.

2.1. Prediction Methods. We select and present a brief description of ten commonly used prediction methods that use topology information about networks in the prediction process. Throughout this section the symbols x, y denote nodes, N denotes number of nodes in the network, and k is the average degree. $\Gamma(x)$ and $\Gamma(y)$ denote the neighbour sets of these nodes and k_x and k_y denote the degree number of node x and y, respectively.

Common Neighbours. This method is based on the assumption that two nodes with many common neighbours will be connected in the future. The more common neighbours the two users have, the higher the probability that a relationship between them will emerge. As a basic and intuitive method, Common Neighbours approach is usually used as a baseline to judge the performance of other methods [17, 20, 21, 40]. The complexity of this method, as introduced in [41], is $O(Nk^2)$.

$$\left|\Gamma(x) \cap \Gamma(y)\right|. \tag{1}$$

Jaccard's Coefficient. The Jaccard Coefficient, also known as Jaccard index or Jaccard similarity coefficient, is a statistic measure used for comparing similarity of sample sets. It is usually denoted as $J(x, y)$ where x and y represent two different nodes in a network. In link prediction, all the neighbours of a node are treated as a set and the prediction is done by computing and ranking the similarity of the neighbour set of each node pair. This method is based on Common Neighbours method and its complexity is also $O(Nk^2)$. The mathematical expression of this method is as follows [20]:

$$\left|\frac{\Gamma(x) \cap \Gamma(y)}{\Gamma(x) \cup \Gamma(y)}\right|. \tag{2}$$

Preferential Attachment. Due to the assumption that the node with high degree is more likely to get new links [42], preferential attachment was introduced as a prediction method. The degree of both nodes in a pair needs to be considered for the prediction. Same as common neighbours, this is also a basic prediction method which is usually used as a baseline to measure the performance of other prediction methods. This method will calculate similarity score for each pair of nodes within the network rather than only the neighbour of nodes; thus the complexity of preferential attachment is $O(N^2k^2)$. This method can be expressed as

$$|\Gamma(x)| * \left|\Gamma(y)\right|. \tag{3}$$

Adamic/Adar Index. It was initially designed to measure the relation between personal home pages. As shown in (4), the more friends z has, the lower score it will be assigned to. Thus, the common neighbour of a pair of nodes with few neighbours contributes more to the Adamic/Adar score (AA) value than this with large number of relationships.

In real-world social network, it can be interpreted as follows: if a common acquaintance of two people has more friends, then it is less likely that he will introduce the two people to each other than in the case when he has only few friends. It shows good results in predicting the friendship according to personal homepage and Wikipedia Collaboration Graph, but in the experiment of predicting author collaboration, it shows a poor accuracy prediction [16]. It is another method that is based on common neighbour; the complexity is also the $O(Nk^2)$. It is calculated as

$$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}, \tag{4}$$

where z is a common neighbour of node x and node y.

$Katz_\beta$. This method takes lengths of all paths between each pair of nodes into consideration [43]. According to (5), the number of paths between node x and node y with length l (written as $|\text{paths}_{xy}^{\langle l \rangle}|$) is calculated and then multiplied by a factor β^l. By summing up all the results for the given two nodes with path length from 1 to ∞, a prediction score for the pair of nodes (x, y) is obtained. Katz is a prediction method based on the topology of whole network and thus its calculation is more complex than other methods in this section. The complexity is mainly determined by the matrix inversion operator, which is $O(N^3)$ [41, 44]:

$$\sum_{l=1}^{\infty} \beta^l \cdot \left|\text{paths}_{xy}^{\langle l \rangle}\right|. \tag{5}$$

The parameter β, as shown in (5), is used to adjust the weight of path with different length. When an extremely small β is chosen, the longer paths will contribute less to the score in comparison to shorter ones so that the result will be close to the common neighbours.

It is one of the prediction methods that, as it will be shown in further sections, achieves high prediction accuracy in many experiments.

Cosine Similarity. The idea of this method is based on the dot product of two vectors. It is often used to compare documents in text mining [21]. In network prediction problem, this method is expressed as

$$\frac{|\Gamma(x)| \left|\Gamma(y)\right|}{\|\Gamma(x)\| * \left\|\Gamma(y)\right\|}. \tag{6}$$

For each pair of nodes with common neighbours, this method will perform a vector multiplication and thus the complexity is $O(Nk^3)$.

Sørensen Index. This index [45] is designed for comparing the similarity of two samples and originally used in analysis plant sociology. The complexity of this method is $O(Nk^2)$. It is defined as

$$\frac{2\left|\Gamma(x) \cap \Gamma(y)\right|}{k_x + k_y}. \tag{7}$$

Hub Promoted Index. HPI is proposed for analysing metabolic networks as shown in [46]. The property of this index is that the links adjacent to hubs are likely to obtain a higher similarity score. The complexity of the method is $O(Nk^2)$. It is expressed as

$$\frac{\left|\Gamma(x) \cap \Gamma(y)\right|}{\min\left\{k_x, k_y\right\}}. \tag{8}$$

Hub Depressed Index. Approach that uses the idea of hub in totally different manner than HPI is Hub Depressed Index (HDI). It gives links adjacent to hub a lower score. Its complexity is the same as Hub Promoted Index, $O(Nk^2)$. It is defined as

$$\frac{\left|\Gamma(x) \cap \Gamma(y)\right|}{\max\left\{k_x, k_y\right\}}. \tag{9}$$

Leicht-Holme-Newman Index. LHNI [47] was proposed to quantify the similarity of nodes in networks. It is based on the concept that two nodes are similar if their immediate neighbours in the network are themselves similar. As another common neighbour based method, its complexity is $O(Nk^2)$. It is defined as

$$\frac{\left|\Gamma(x) \cap \Gamma(y)\right|}{k_x * k_y}. \tag{10}$$

All of the methods presented in this section are following similar approach. The required input for each method is the adjacency matrix that represents a network in which there are only 0 and 1 (0 when there is no link between two given nodes and 1 when the links between two given nodes exist). The output of each method is a similarity matrix in which each element represents the similarity score of a pair of nodes within the network and it is calculated according to the equation used in a given method.

2.2. Prediction Performance Metrics. In order to measure the performance of a prediction method, we need to use historical network data. Link prediction is a time related activity; therefore, we should use time-stamped dataset, and according to the time stamp, separate the data into two sets, $G_{t,t_1}(V, L_1)$ as training set for prediction methods and $G_{t_1,t_2}(V, L_2)$ as unknown future network for testing where $t < t_1 < t_2$. Those two networks must consist of the same set of nodes V. The number of possible links that is denoted by U is $|V| * (|V| - 1)/2$. The link prediction method, in principle, provides a similarity score for each nonexisting link $(U - L_1)$ and for most methods, a higher score means higher likelihood that the link will appear in the future. Final prediction is done by ordering this score list and selecting top N links with the highest score.

In our work, AUC is used for quantifying the accuracy of prediction method. It is the area under the receiver operating characteristic curve [48]. In the context of network link prediction, AUC can be interpreted as the probability that a

Table 1: Original dataset information.

Dataset name	Time range	Vertices	Edges
Enron E-mail Communication[a]	1998/11–2002/07	87,273	1,148,072
Facebook Wall Posts[b]	2008/01–2009/01	63,731	1,269,502
Flickr Friendship[c]	2006/11–2007/05	2,302,925	33,140,018
PWr E-mail Communication[d]	2008/11–2009/05	14,316	49,950
UC Irvine Messages[e]	2004/03–2004/10	1,899	59,835
YouTube Friendship[f]	2006/12–2007/07	3,223,589	12,223,774

This table shows the original information about the datasets used in the experiments.
[a]The Email network among employees of Enron. Nodes in the network are individual employees and edges are individual emails [32].
[b]The wall posts from the Facebook New Orleans networks [33].
[c]The social network of Flickr users and their friendship connections. It is collected by taking a snapshot of the network on November 2, 2006, and recording it daily until December 3, 2006, and then again daily between February 3, 2007, and May 18, 2007 [34, 35].
[d]The Email Communication of Wrocław University of Technology [36].
[e]The network contains messages sent between the users of an online community of students from the University of California, Irvine. A node represents a user. An edge represents sent message. Multiple edges denote multiple messages [37].
[f]The social network of YouTube users and their friendship connections between December 10, 2006, and January 15, 2007, and again daily between February 8, 2007, and July 23, 2007 [38, 39].

randomly chosen missing link $(L_1 \cup L_2 - L_1)$ is given a higher similarity score than a randomly chosen pair of unconnected links $(U - (L_1 \cup L_2))$ [49]. The algorithmic implementation of AUC follows the approach in [21]. It is calculated as

$$\frac{n' + 0.5n''}{n}, \tag{11}$$

where n is the number of times that we randomly pick a pair of links from missing links set and unconnected links set; n' is the number of times that the missing link got a higher score than unconnected link, while n'' is the number of times when they are equal. The AUC value will be 0.5 if the score is generated from an independent and identical distribution. Thus, the degree to which the AUC exceeds 0.5 indicates how much better the predictions are when compared to prediction by chance.

3. Data Preparation

All six datasets used in experiments are real-world social networks, five of them come from Koblenz Network Collection (KONECT [50]) and another one from the Wrocław University of Technology (see Table 1).

3.1. Dataset Selection. Datasets for the experiments have to meet certain requirements: (i) they have to represent data about users' interactions or any other type of activity that enables to define connections between users and (ii) those activities have to be time stamped. As described in Section 2, the link prediction problem is a time series problem that looks into the evolution of networks in time. Time-stamp is thus necessary. Table 1 shows the original dataset information that was selected based on these two criteria.

3.2. Data Processing. To make the data suitable for the experiments, first the preprocessing of datasets has been performed. It consists of the following three steps.

(1) *Select Data Samples*. For each dataset, we first randomly select 6000–8000 user records (8000 samples are selected due to the calculation capacity. As for some dense networks, 8000 nodes are also too big, so we choose 6000) from the original dataset as the sample user data. As UC Irvine Messages only contain 1899 users, so we leave them as they are. The specific sample numbers are shown in Table 2.

(2) *Split the Data into Training and Testing Sets*. Prediction in a time series problem means the dataset should be divided into train and test sets based on time stamps available. As the dataset of Flickr and YouTube are collected by taking snapshot of the network which is different from other four datasets, we take the first day snapshot as the training set and the remaining data as the test set. The other four networks are split according to the time scale with a ratio of approximate training time: test time = 80% : 20% as shown in Table 2.

(3) *Extract Connected Network*. Dividing data into training and testing sets can cause the isolation of some nodes or cliques. This, in turn, generates noise for measuring the accuracy of prediction methods as the methods we selected can not predict unconnected nodes. To eliminate the impact of this noise, we extract the giant component from training dataset as our final training set $G_{t,t_1}(V, L_1)$. The final test set $G_{t_1,t_2}(V, L_2)$ is obtained by extracting the network with all the nodes that exist in $G_{t,t_1}(V, L_1)$ from the original test set obtained from step (2). For nodes existing in the final training set but not present in the original test set, we just keep and leave them isolated in the final test set as it is formed by link disappearing.

After all, we get the train set $G_{t,t_1}(V, L_1)$ and test set $G_{t_1,t_2}(V, L_2)$ as described in Section 2.2 where both sets have the same nodes V.

Table 2: Dataset details.

Dataset name	Train time range	Test time range	Sample nodes	Final nodes
Enron E-mail Communication	1998/11–2001/12	2002/01–2002/07	8000	5208
Facebook Wall Posts	2008/01–2008/11	2008/12–2009/01	8000	5784
Flickr Friendship	Snapshot on 2006/11/02	2006/11/03–2006/12/03 & 2007/02/03–2007/05/18	6000	5949
PWr E-mail Communication	2008/11–2009/04	2009/04-2009/05	8000	5208
UC Irvine Messages	2004/03–2004/08	2004/08–2004/10	1899	1666
YouTube Friendship	Snapshot on 2006/12/10	2006/12/11–2007/01/15 & 2007/02/08–2007/07/23	6000	6000

The time range of train and test set, the number of sample nodes selected from the original dataset and number of nodes in the giant component which are used as the final nodes set for the experiment are presented in the table.

4. Experimental Design

In order to be able to apply all selected methods and taking into account the types of datasets available, the network is represented as a binary unweighted network. This enables us to reach a consistent and comprehensive review of the existing methods.

First, the prediction methods described in Section 2.1 will be applied to each of the processed training sets to get the similarity matrix as the prediction result. The prediction results will be then evaluated using the testing set and the AUC for each method will be calculated.

For the implementation of those methods, we applied the toolbox that is presented in [21] and all the experiments were implemented in Matlab.

As stated before, the main goal of the research is to explore the correlations between the accuracy of different prediction methods and network metrics. For the training set of each network, the network metrics are calculated with toolboxes provided by KONECT [50] and MIT Strategic Engineering research group. The metrics we calculate include the following.

Global Clustering Coefficient. It is defined in [51] as

$$\text{GCC} = \frac{3 * \text{number of triangles in the network}}{\text{Number of connected triples of vertices}}. \quad (12)$$

It shows the transitivity of the network as a whole. The coefficient range is between 0 and 1.

Average Clustering Coefficient [6]. It is based on local clustering C_l. For each vertex l, its local clustering coefficient can be calculated by

$$C_l = \frac{\text{Number of triples connected to vertex } l}{\text{Number of triples centered on vertex } l} \quad (13)$$

and then the ACC can be calculated as

$$\text{ACC} = \frac{1}{v}\sum_{l} C_l, \quad (14)$$

where v is the number of nodes in a network.

Network Density. The ratio between number of existing links and number all possible links within a given network.

$$\text{Network Density} = \frac{\text{Number of Existing Links}}{\text{Number of all possible links}}, \quad (15)$$

where

$$\text{Number of all possible links} = \frac{v * (v-1)}{2}, \quad (16)$$

where v is the number of nodes in the network.

Gini Coefficient [52]. I the network theory Gini coefficient is defined as

$$G = \frac{2\sum_{i-1}^{n} id_i}{n\sum_{i-1}^{n} d_i} - \frac{n+1}{n}, \quad (17)$$

where $d_1 \le d_2 \le d_3 \le \cdots \le d_n$ is the sorted list of degrees in the network and n is the number of nodes in a network. Its value is between 0 and 1, where 0 denotes total equality between degrees and 1 denotes dominance of single node.

Diameter. It is the longest path out of the set of all shortest paths in the network.

$$\text{Diameter} = \max_{i,j} d(i,j), \quad (18)$$

where $d(i, j)$ is the shortest path between node i and j.

Average Shortest Path. The average number of the shortest paths between each pair of vertices is

$$\text{ASP} = \frac{1}{v \cdot (v-1)} \cdot \sum_{i!=j} d(i,j). \quad (19)$$

Once the accuracy of prediction for each method and the metrics for each network are calculated, the correlation between them will be analysed. The Pearson's Coefficient [53] is used to measure the correlation between accuracy of network prediction method and selected network metrics. It is a widely used statistic method to measure linear correlation between two variables, say W and Z. It is calculated as

$$\frac{\sum_{i=1}^{n}\left(W-\overline{W}\right)\left(Z-\overline{Z}\right)}{\sqrt{\sum_{i=1}^{n}\left(W_i-\overline{W}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Z-\overline{Z}\right)^2}}. \quad (20)$$

Table 3: Theoretical GCC and ASP of random, real, and regular network.

	Random network	YouTube	Regular network
Nodes	6,000	6,000	6,000
Links	54,596	54,596	54,596
GCC	0.0030	0.0286	0.7064
ASP	2.9983	3.0709	164.8500
UC Irvine			
Nodes	1,666	1,666	1,666
Links	11,582	11,582	11,582
GCC	0.00835	0.0197	0.6919
ASP	2.8186	3.0463	59.9108
PWr			
Nodes	6,335	6,335	6,335
Links	15,334	15,334	15,334
GCC	0.0008	0.0048	0.5547
ASP	5.5499	4.0162	654.3060
Flickr			
Nodes	5,949	5,949	5,949
Links	387,719	387,719	387,719
GCC	0.0219	0.0658	0.7442
ASP	1.7845	2.3447	22.8198
Facebook			
Nodes	5,784	5,784	5,784
Links	14,507	14,507	14,507
GCC	0.0009	0.0341	0.5633
ASP	5.3717	5.7235	576.5205
Enron			
Nodes	5208	5208	5208
Links	23977	23977	23977
GCC	0.0018	0.0290	0.6586
ASP	3.8548	3.6818	282.8037

The coefficient value is between −1 and 1 where −1 means that two variables are negatively linearly correlated and 1 means that they are positively linearly correlated.

5. Experiment Result

5.1. Network Profiles. The values of network metrics for each of the extracted social networks are presented in Table 5. As it is much easier to set up relationship between people in online social network than in real-world network, the average shortest paths in our experiments are all smaller than six, the number suggested by the six degrees of separation theory [54]. The average shortest path of the six selected networks is 3.65. This reflects the small-world property of the networks. People are closer to each other in online social networks than in face-to-face networks. This phenomenon was also pointed out in [55] where authors established that the average shortest path of Twitter is 3.43.

The degree distributions of the six networks, shown in Figure 3, indicate that they are scale-free networks as the distributions follow the power law.

Table 4: Analytical formulas for GCC and ASP in random and regular networks.

	Random network	Regular network
GCC	$\frac{k}{v}$	$\frac{3(k-2)}{4(k-1)}$
ASP	$\frac{\log v}{\log k}$	$\frac{v}{2k}$

k is the average degree and v is the number of nodes in the network.

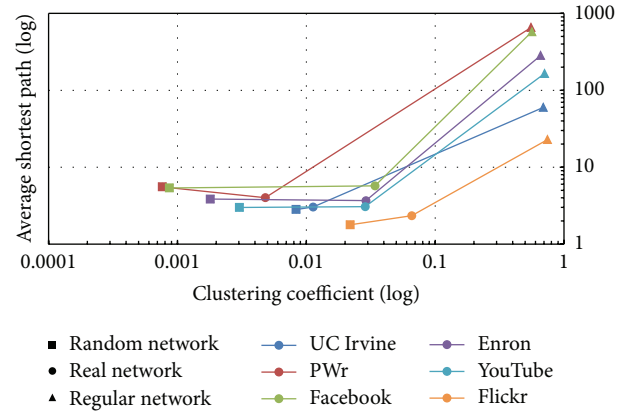
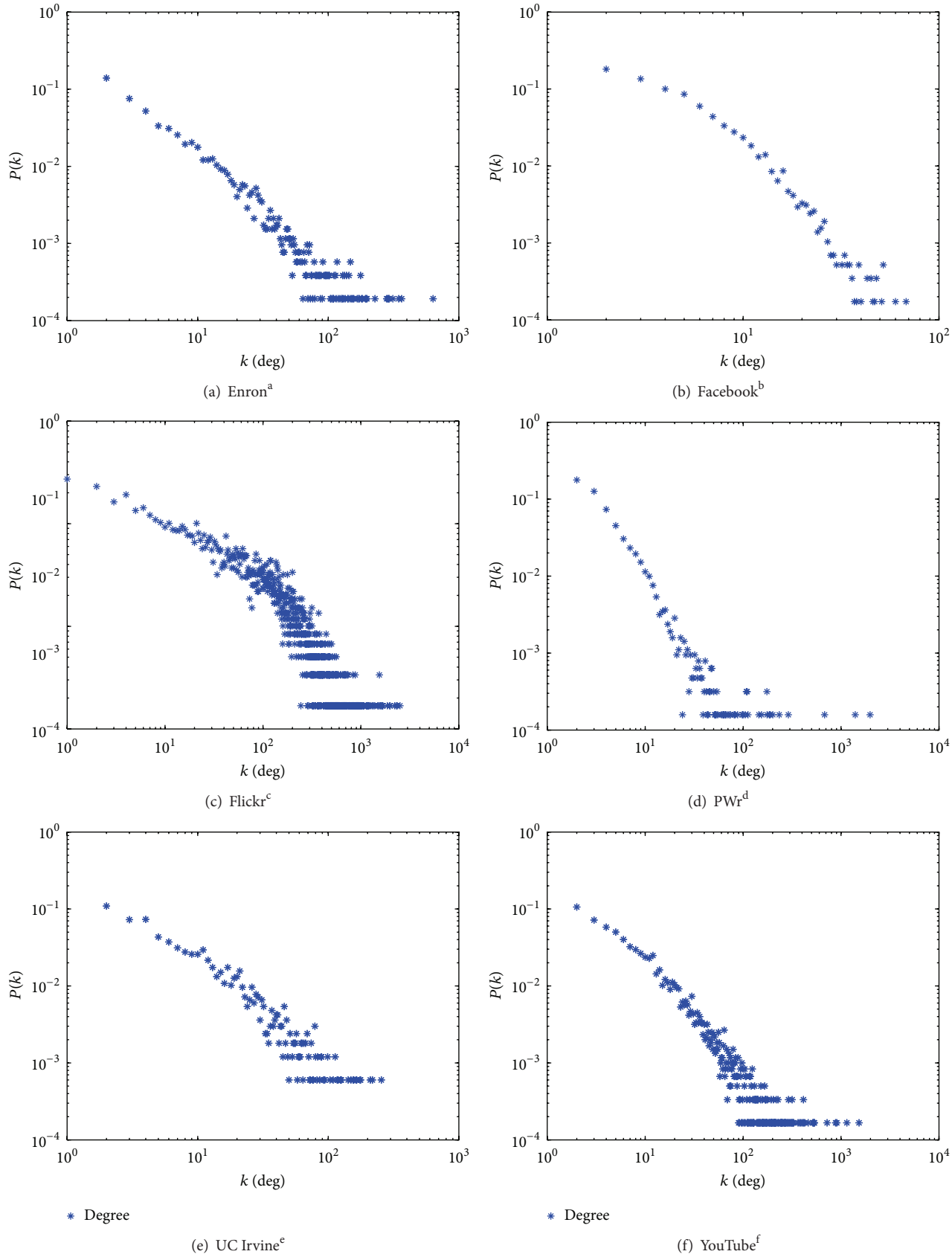


Figure 2: GCC and ASP calculated for six analysed networks (circle) and corresponding random (square) and regular (triangle) networks generated using the same number of nodes and connections as in real-world networks. Different colours depict different networks and corresponding random and regular networks—see legend. For example green circle is a datapoint for Facebook network, green square is a datapoint for random network corresponding to Facebook network and green triangle is a datapoint for regular network corresponding to Facebook network.

We also compared the GCC and ASP metrics of the real network with the theoretical metrics of random network and regular network that have the same number of nodes and links. The analytical formulas for GCC and ASP in random and regular networks with a given number of nodes and links are given in Table 4. The results of calculations for each analysed network are presented in Table 3.

Figure 2 plots the metrics of six analysed networks and related theoretical networks, respectively. It shows that the clustering coefficients of the analysed networks are all between random and regular networks. Meanwhile, the average shortest paths of real-world networks are all very close to the random networks. This two phenomena indicate the small-world property of analysed structures. Taking into account both metrics and node degree distribution, it can be concluded that those networks are a combination of small-world and scale-free networks.

5.2. Prediction Results. The prediction results are summarised in Table 6. Katz method achieved the best average performance and the overall performance is ranked as Katz > Preferential Attachment > Adamic-Adar > Common Neighbours > Cosine Similarity > Jaccard Index > Hub Depressed



(a) Enron[a]

(b) Facebook[b]

(c) Flickr[c]

(d) PWr[d]

(e) UC Irvine[e]

(f) YouTube[f]

FIGURE 3: The Degree distributions. The degree distributions are all following the power law with exponent of [a]Enron, $r = 1.85$; [b]Facebook, $r = 1.82$; [c]Flickr, $r = 1.25$; [d]PWr, $r = 2.19$; [e]UC Irvine, $r = 1.56$; [f]YouTube, $r = 1.56$.

Table 5: Network metrics results.

Datasets	GCC	ACC	Network density	Gini Coefficient	Diameter	Ave. shortest path
Facebook	0.0341	0.1176	0.0008674	0.473	16	5.7235
Flickr	0.0658	0.3294	0.0219	0.5931	6	2.3447
UC Irvine	0.0197	0.1075	0.0084	0.6394	7	3.0463
PWr	0.0048	0.2666	0.00076	0.6407	16	4.0162
Enron	0.029	0.1946	0.0018	0.7172	10	3.6818
YouTube	0.0286	0.2838	0.003	0.7222	5	3.0709

Table 6: Prediction methods accuracy result (AUC).

Datasets	AUC									
	CN	JI	PA	AA	Katz$_\beta$[a]	Cosin	Sor	HPI	HDI	LHN
Facebook	0.6688	0.6758	0.6803	0.6753	0.8369	0.6738	0.6715	0.6708	0.6694	0.6694
Flickr	0.89	0.8702	0.841	0.8922	0.8839	0.8812	0.865	0.844	0.8511	0.6944
UC Irvine	0.6625	0.6421	0.8412	0.6738	0.8048	0.6414	0.6359	0.6303	0.6427	0.6322
PWr	0.6815	0.6466	0.7924	0.6913	0.7979	0.651	0.6514	0.6422	0.6491	0.6382
Enron	0.8157	0.7937	0.9015	0.8196	0.9312	0.7921	0.7995	0.794	0.7977	0.7881
YouTube	0.8525	0.7957	0.9109	0.8571	0.9157	0.7938	0.7503	0.8017	0.7984	0.7587
Average	0.7618	0.7374	0.8279	0.7682	0.8617	0.7389	0.7289	0.7305	0.7374	0.6968
Variance	0.0105	0.0091	0.0071	0.0099	0.0032	0.0095	0.0084	0.0087	0.0083	0.0041

The accuracy of selected prediction methods measured by AUC. The average performance and the variance for each method are also listed.
[a]In our experiment, we choose $\beta = 0.0005$.

Index > Hub Promoted Index > Sørensen > Leicht-Holme-Newman Index. By comparing the variance of each method, we find that the Katz also provides the most stable prediction performance among those methods while Common Neighbours is the worst performing approach. Overall, we find that Katz and preferential attachment provide good prediction accuracy together with a relative stability.

To study the prediction results from the perspective of each network please see Figure 4. The prediction results of different methods align on the vertical lines for each network, respectively. From this figure, we find that, for some networks, most of the prediction methods could provide a good prediction result. Such networks include Flickr, Enron, and YouTube. We call this type of networks the "prediction friendly" network. Apart from this type of network, there are also some networks for which most of the prediction approaches provide fairly low accuracy, such as Facebook, UC Irvine, and PWr. Similarly, we call those networks "prediction unfriendly" networks. Please note that, in the experiments, for both prediction friendly and unfriendly networks, Katz$_\beta$ always provides a good performance level.

5.3. Correlation between Prediction Accuracy and Network Metrics. Table 7 shows Pearson's linear correlation coefficient of prediction accuracy and network metrics. The closer the absolute value to 1, the higher the correlation between analysed factors. Figure 5 presents a heat-map plot to show the degree of linear relation between the two factors where we use the absolute value of Pearson's Coefficient. The brighter the colour in the heat-map is, the stronger a given network metric and the accuracy of prediction method are correlated.

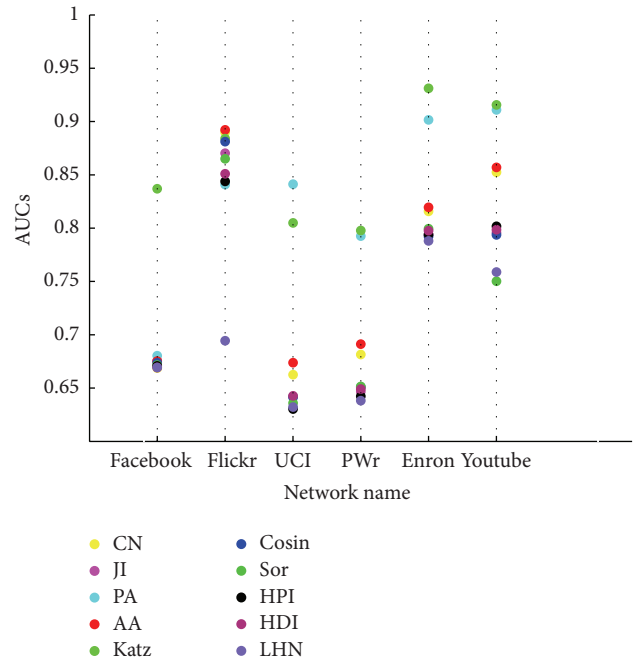


Figure 4: The AUC prediction results for each network.

In Figure 5, we can see that the preferential attachment and Gini Coefficient provide the highest correlation coefficient (0.94) which indicates that they generally follow a linear relationship. This is not a surprise. For a network with a high Gini Coefficient, there exist some nodes with dominant high

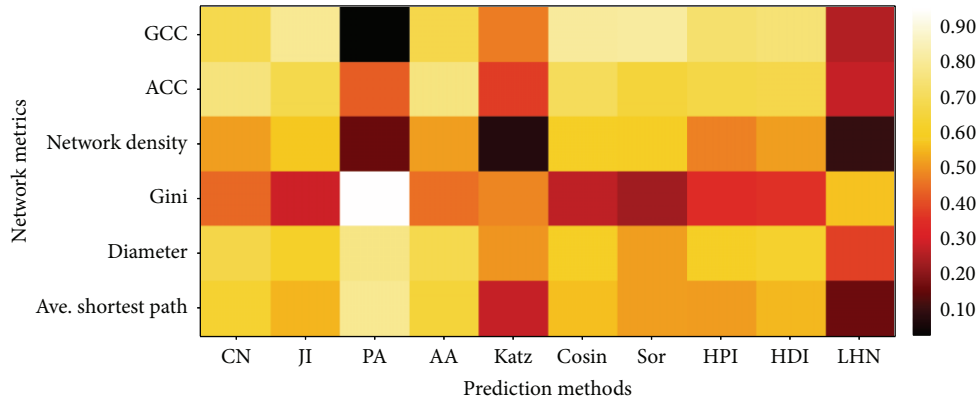


Figure 5: Heat-map of network metrics and prediction methods correlation. As for the Pearson Coefficient, both 1 and −1 stand for linear relationship (positive and negative); we use the absolute value of correlation coefficient in this figure to indicate whether the two factors are linearly correlated.

Table 7: Pearson Correlation of prediction methods accuracy and network metrics.

	CN	JI	PA	AA	Katz$_\beta$	Cosine	Sor	HPI	HDI	LHN	Average
GCC	0.68	0.79	0.05	0.68	0.47	0.80	0.81	0.73	0.74	0.27	0.60
ACC	0.75	0.68	0.43	0.76	0.39	0.70	0.65	0.67	0.68	0.30	0.60
Network Density	0.52	0.58	0.18	0.52	0.09	0.61	0.61	0.48	0.52	−0.12	0.40
Gini	0.45	0.30	0.94	0.46	0.49	0.29	0.25	0.36	0.37	0.57	0.45
Diameter	−0.67	−0.61	−0.77	−0.68	−0.51	−0.61	−0.52	−0.61	−0.63	−0.39	−0.60
ASP	−0.63	−0.55	−0.79	−0.65	−0.29	−0.57	−0.52	−0.52	−0.56	−0.18	−0.53

This table shows the correlation between prediction methods accuracy and network metrics calculated with Pearson's linear correlation coefficient. The number is within the range of [−1, 1] where 1 is completely positive correlation, 0 is no correlation, and −1 is completely negative correlation.

Table 8: Metrics rank of networks.

Dataset	GCC	ACC	Diameter	ASP	Ave. rank
PWr	6	3	5	5	4.75
Facebook	2	5	5	6	4.5
UC Irvine	5	6	3	2	4
Enron	3	4	4	4	3.75
YouTube	4	2	1	3	2.5
Flickr	1	1	2	1	1.25

degrees. It just reflects the phenomenon of "rich get richer" which is also the assumption of preferential attachment method. So we can say that preferential attachment could lead to a high Gini Coefficient and thus Preferential Attachment, on the other hand, could also describe how a network with high Gini Coefficient evolves by giving a better prediction result.

Cosine-GCC and Sor-GCC also provide a correlation coefficient above 0.8. We can draw the conclusion that Cosine Similarity and Sorensen Index method perform better in a network with higher GCC than they do in networks with smaller GCC.

The diameter and average shortest path shows a negative linear relation to almost all of the prediction methods (excluding Katz and LHN where the negative correlation is weak). Both the average shortest path and the network diameter reflect how easy it is to get from one node in a network to another one. Shorter path as well as smaller diameter means a higher probability that a pair of randomly picked nodes will be connected. Negative correlation between those two metrics and prediction accuracies of different methods means that most of the methods work well in the situations where networks feature short ASP and in consequence small Diameter. This is additionally supported by the fact that global clustering coefficient is positively correlated with those of the prediction methods meaning that these methods work well with networks with high clustering coefficient. Based on the above we can say that prediction methods positively correlated with GCC and negatively with ASP and diameter will work well in the situation where analysed network is of small-world type. In the same time they will work neither in random networks where GCC is very low nor in regular networks where ASP is very long.

It should be clear that the Pearson's Coefficient does not indicate the accuracy of the method. For example, although the prediction method Katz does not show strong correlation to any of the network metrics, it still provides the best result in our experiments. The reason can be found in Table 6, where it is shown that Katz always provides a high prediction accuracy regardless of the tested network metrics.

The most important value of our correlation study lies in the variety of prediction methods used in the experiments. The prediction with methods combination could be a way to

improve accuracy and this will be investigated in the future. The correlation between methods and network metrics could be used to determine the weight of different prediction methods in the combination process.

5.4. Prediction Friendly and Unfriendly Networks. Table 7 also shows the average correlation of network metrics and prediction accuracy. As we know the closer the absolute value of correlation to 1, the stronger the linear relation. Here we take 0.5 as a threshold for strong correlation. According to this, we find that there are four metrics strongly correlated with the prediction accuracy which includes GCC, ACC, Diameter, and ASP. So it is reasonable to assume that these metrics could be used to classify the prediction friendly and unfriendly networks. We ranked each of the analysed networks according to the metrics that have strong correlation with prediction accuracy and based on this for each network we calculate the average ranking (Table 8). Top three ranked networks (with the small average ranks) are the prediction friendly networks and the other three are prediction unfriendly networks. It can be seen that the prediction friendly networks usually have large global and local clustering coefficient, a short average shortest path, and small diameter. It suggests that networks with the structural profile similar to small-world network are easier to predict than networks similar to random structures.

6. Conclusions

In this research, we look into the correlation between ten prediction methods and different network metrics in six time-stamped social networks. The study of network metrics confirmed that the node degree distribution of real-world social networks follows a power law distribution. We also found that the average shortest path of online social network is much smaller than six. This might be due to the fact that online relationships are much easier to setup. The results of the prediction accuracy show that the best method among the tested ones is Katz_β. It is also the most stable technique from all tested ones. preferential attachment is the second best method that also provides a good prediction accuracy. In addition, for some "prediction friendly" networks, most of prediction methods could provide a good performance while for some others, called in here as "prediction unfriendly" networks, most prediction methods are lack of power.

The Pearson correlation coefficient enabled us to investigate the relationship between network metrics and prediction accuracy. Our research showed that some methods are highly correlated with certain network metrics (e.g., PA-Gini, Sor-GCC, and Cosine-Gcc).

There are several further directions of the presented study. As discovered, for some networks, most prediction methods could provide a good performance which we name them as "prediction friendly networks." Similarly, we also find the existence of "prediction unfriendly" networks. Section 5.4 explores the prediction friendly and unfriendly network classification according to the metrics ranking. The problem is that it does not provide an exact threshold that could be used to classify networks. It is out of scope of this research but is a very interesting topic for another study that we plan to conduct.

Based on the results of correlation between network metrics and the prediction accuracy, another possible work is to develop a new prediction approach which combines several existing methods. We can also extend this research to many other networks, not only social ones, which might be good for finding some more general relations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] N. L. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory*, New York, NY, USA, The Clarendon Press, 2nd edition, 1986.

[2] P. Erdős and A. Rényi, "On random graphs. I," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[3] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, pp. 17–61, 1960.

[4] J. Travers and S. Milgram, "An experimental study of the small world problem," *Sociometry*, vol. 32, no. 4, pp. 425–443, 1969.

[5] S. Milgram, "The small world problem," *Psychology Today*, vol. 2, pp. 60–67, 1967.

[6] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[7] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[8] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.

[9] A. D. King, N. Pržulj, and I. Jurisica, "Protein complex prediction via cost-based clustering," *Bioinformatics*, vol. 20, no. 17, pp. 3013–3020, 2004.

[10] R. Pastor-Satorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Physical Review Letters*, vol. 86, no. 14, pp. 3200–3203, 2001.

[11] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: an eigenvalue viewpoint," in *Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS '03)*, pp. 25–34, October 2003.

[12] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *ACM Transactions on Information and System Security*, vol. 10, no. 4, article 1, 26 pages, 2008.

[13] N. A. Christakis and J. H. Fowler, "The spread of obesity in a large social network over 32 years," *The New England Journal of Medicine*, vol. 357, no. 4, pp. 370–379, 2007.

[14] Z. Huang, X. Li, and H. Chen, "Link prediction approach to collaborative filtering," in *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries (JCDL '05)*, pp. 141–142, ACM, New York, NY, USA, June 2005.

[15] F. Molnar, "Link prediction analysis in the Wikipedia Collaboration graph," 2011, http://www.cs.rpi.edu/ magdon/courses/casp/projects/Molnar.pdf.

[16] L. A. Adamic and E. Adar, "Friends and neighbors on the Web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[17] W. Cukierski, B. Hamner, and B. Yang, "Graph-based features for supervised link prediction," in *Proceedings of the International Joint Conference on Neural Network (IJCNN '11)*, pp. 1237–1244, August 2011.

[18] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici, "Link prediction in social networks using computationally efficient topological features," in *Proceedings of the IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT '11) and IEEE International Conference on Social Computing (SocialCom '11)*, pp. 73–80, Boston, Mass, USA, October 2011.

[19] K. Juszczyszyn, K. Musiał, and M. Budka, "Link prediction based on Subgraph evolution in dynamic social networks," in *Proceedings of the IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT '11) and IEEE International Conference on Social Computing (SocialCom '11)*, pp. 27–34, October 2011.

[20] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM '03)*, pp. 556–559, ACM, New York, NY, USA, November 2003.

[21] L. Lü and T. Zhou, "Link prediction in complex networks: a survey," *Physica A: Statistical Mechanics and Its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[22] O. J. Mengshoel, R. Desai, A. Chen, and B. Tran, "Will we connect again? Machine learning for link prediction in mobile social networks," 2013.

[23] Z. Liu, Q.-M. Zhang, L. Lü, and T. Zhou, "Link prediction in complex networks: a local naïve Bayes model," *Europhysics Letters*, vol. 96, no. 4, Article ID 48007, 2011.

[24] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD '10)*, pp. 243–252, New York, NY, USA, July 2010.

[25] K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu, "Stochastic relational models for discriminative link prediction," in *Advances in Neural Information Processing Systems*, pp. 333–340, MIT Press, Boston, Mass, USA, 2007.

[26] M. Al Hasan, V. Chaoji, S. Salem, and Z. Mohammed, "Link prediction using supervised learning," in *Proceedings of the SDM 6th workshop on Link Analysis, Counterterrorism and Security*, 2006.

[27] X.-W. Chen and M. Liu, "Prediction of protein-protein interactions using random decision forest framework," *Bioinformatics*, vol. 21, no. 24, pp. 4394–4400, 2005.

[28] P. Aloy and R. B. Russell, "InterPreTS: protein interaction predictionthrough tertiary structure," *Bioinformatics*, vol. 19, no. 1, pp. 161–162, 2003.

[29] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: structure and dynamics," *Physics Reports: A Review Section of Physics Letters*, vol. 424, no. 4-5, pp. 175–308, 2006.

[30] M. Budka, K. Juszczyszyn, K. Musial, and A. Musial, "Molecular model of dynamic social network based on e-mail communication," *Social Network Analysis and Mining*, vol. 3, no. 3, pp. 543–563, 2013.

[31] C. Guido, A. Chessa, I. Crimaldi, and F. Pammolli, *The Evolution of Complex Networks: A New Framework*, 2012.

[32] B. Klimt and Y. Yang, "The Enron corpus: a new dataset for email classification research," in *Proceedings of the 15th European Conference on Machine Learning (ECML '04)*, pp. 217–226, September 2004.

[33] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in Facebook," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Online Social Networks*, pp. 37–42, Barcelona, Spain, August 2009.

[34] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Growth of the Flickr social network," in *Proceedings of the Workshop on Online Social Networks*, pp. 25–30, 2008.

[35] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Growth of the flickr social network," in *Proceedings of the 1st ACM SIGCOMM Workshop on Social Networks (WOSN '08)*, pp. 25–30, ACM, August 2008.

[36] P. Kazienko, K. Musiał, and A. Zgrzywa, "Evaluation of node position based on email communication," *Control and Cybernetics*, vol. 38, no. 1, pp. 67–86, 2009.

[37] T. Opsahl, "Triadic closure in two-mode networks: redefining the global and local clustering coefficients," *Social Networks*, vol. 35, no. 2, pp. 159–167, 2013.

[38] A. Mislove, *Online social networks: measurement, analysis, and applications to distributed information systems [Ph.D. thesis]*, Rice University, 2009.

[39] A. Mislove, *Online social networks: measurement, analysis, and applications to distributed information systems [Ph.D. thesis]*, Department of Computer Science, Rice University, 2009.

[40] H. R. de Sa and R. B. C. Prudencio, "Supervised link prediction in weighted networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '11)*, pp. 2281–2288, August 2011.

[41] L. Lü, C.-H. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks," *Physical Review E*, vol. 80, Article ID 046122, 2009.

[42] M. E. J. Newman, "Clustering and preferential attachment in growing networks," *Physical Review E*, vol. 64, no. 2, Article ID 025102, 2001.

[43] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[44] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.

[45] T. Sørensen, *A Method of Establishing Groups of Equal Amplitude in Plants Ociology Based on Similarity of Species and Its Application to Analyses of The Vegetation on Danish Commons*, vol. 5 of *Biologiske Skrifter*, E. Munksgaard, 1948.

[46] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[47] E. A. Leicht, P. Holme, and M. E. J. Newman, "Vertex similarity in networks," *Physical Review E*, vol. 73, no. 2, Article ID 026120, 2006.

[48] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[49] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, pp. 98–101, 2008.

[50] J. Kunegis, "KONECT—the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*, pp. 1343–1350, May 2013.

[51] M. E. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[52] J. Kunegis and J. Preusse, "Fairness on the Web: Alternatives to the power law," in *Proceedings of the 3rd Annual ACM Web Science Conference (WebSci '2)*, pp. 175–184, June 2012.

[53] J. L. Rodgers and A. W. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[54] M. E. J. Newman, A. L. Barabási, and D. J. Watts, Eds., *The Structure and Dynamics of Networks*, Princeton Studies in Complexity, Princeton University Press, Princeton, NJ, USA, 2006.

[55] R. Bakhshandeh, M. Samadi, Z. Azimifar, and J. Schaeffer, "Degrees of separation in social networks," in *Proceedings of the 4th International Symposium on Combinatorial Search (SoCS '11)*, pp. 18–23, July 2011.

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 602690, 10 pages
http://dx.doi.org/10.1155/2015/602690

Research Article

A Community-Based Approach for Link Prediction in Signed Social Networks

Saeed Reza Shahriary,[1] Mohsen Shahriari,[2] and Rafidah MD Noor[1]

[1]*Department of Computer System & Technology, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia*
[2]*Advanced Community and Information System, RWTH Aachen University, Ahornstraße 55, 52056 Aachen, Germany*

Correspondence should be addressed to Saeed Reza Shahriary; shahriarysaeedreza@gmail.com,
Mohsen Shahriari; shahriari@dbis.rwth-aachen.de and Rafidah MD Noor; fidah@siswa.um.edu.my

Received 28 February 2014; Accepted 8 October 2014

Academic Editor: Przemyslaw Kazienko

In signed social networks, relationships among nodes are of the types positive (friendship) and negative (hostility). One absorbing issue in signed social networks is predicting sign of edges among people who are members of these networks. Other than edge sign prediction, one can define importance of people or nodes in networks via ranking algorithms. There exist few ranking algorithms for signed graphs; also few studies have shown role of ranking in link prediction problem. Hence, we were motivated to investigate ranking algorithms availed for signed graphs and their effect on sign prediction problem. This paper makes the contribution of using community detection approach for ranking algorithms in signed graphs. Therefore, community detection which is another active area of research in social networks is also investigated in this paper. Community detection algorithms try to find groups of nodes in which they share common properties like similarity. We were able to devise three community-based ranking algorithms which are suitable for signed graphs, and also we evaluated these ranking algorithms via sign prediction problem. These ranking algorithms were tested on three large-scale datasets: Epinions, Slashdot, and Wikipedia. We indicated that, in some cases, these ranking algorithms outperform previous works because their prediction accuracies are better.

1. Introduction

Recently, social network analysis has attracted great deal of attentions. In social networks, nodes and edges, respectively, indicate people and relationships among them [1]. Social networks are dynamic and evolve over time via registering new members, deleting profiles, and adding/removing some edges or connections among entities [2]. Hence, plenty of studies have investigated this field in order to model these structures. One of the most important problems in social networks is link prediction which can be stated as follows: with how much determinism one can predict forming (lacking) of edge between two people based on available structure of the graph? The importance of this subject is originated from the natural sparsity of social networks [2]. In other words, social networks encompass highly dynamic structure; therefore, available links are just a subset of possible relations among people and some new links will form in future. Link predication is also widely used in retrieving lost data and it probably helps to construct the graph [3].

One could model social systems by using signed relationships. Inherently in signed graphs, most of relations are positive or negative such as likes and dislikes or trusts and distrusts [4]. Negative edges play an important role in signed networks and these negative links impress greatly on importance of nodes in the system. Studying negative relationships in signed graphs can help in analyzing and better understanding social ecosystems. Link prediction in signed graph appears in the form of predicting sign of edge between two people. Therefore, one important question which comes to mind in signed networks is that how accurately sign of an edge can be predicted according to local and global behavioral patterns in the network. Not only sign prediction enables us to have better understanding of social relations but also it can be utilized in several applications such as recommender systems and online social networks, in which

they offer new friends for users. In these networks, users have capability of expressing their views toward others via binary −1 and +1 values [5, 6].

In this paper, three community-based ranking algorithms for ranking of nodes have been proposed, and we have studied their impacts on the edge sign prediction problem. In order to study the impact of proposed ranking algorithms on signed prediction problem, we extracted the features of the predictor based on reputation and optimism introduced in [7]. Reputation of a node shows how much reputable a node is in the system and optimism denotes voting pattern of the node toward others. We assess our method by utilizing logistic regression classifier and running algorithms on real social network datasets. The structure of this paper is organized as follows.

In Section 2, related works are brought. In Section 3, we mainly introduce proposed algorithms; moreover, the problem of sign prediction is defined and rank-based features are introduced as features for the prediction task. We also separately go through community detection problem, community-based ranking algorithms, and the logistic regression classifier. In Section 4, datasets for experimental purposes are introduced and implementation results are also demonstrated. In Section 5, the discussion is made and finally in Section 6, conclusion and future directions are mentioned.

2. Related Works

There are two major categories of methods used in link prediction: firstly, those approaches that utilize local information of the graph which focus on the local structure of nodes. Among local approaches, [8] has the best performance in link prediction between two specific nodes. Common neighbor index is also known as friend of friend algorithm (FOAF) is used by many online social networks for recommending friends such as Facebook. FOAF determines the similarity of two nodes that tend to communicate with each other on the basis of counting number of joint neighbors [9]. Other metrics for computing similarity are based on preferential attachments, where these measures are calculated based on multiplying or summing of nodes degree. Second category concentrates on global structure of the network and detecting overall features in order to find how strongly two nodes are similar. There are also diverse global approaches which use the whole adjacency matrix in order to predict hidden links, for instance, shortest path algorithm, PWR algorithm, and SimiRank algorithm [1, 10].

In sign prediction, the most notable and remarkable methods are divided into two categories: Belief Matrix Model [5] and machine learning approaches [11]. Belief Matrix Model was introduced by [5] and was proposed for predicting trust or distrust between two particular users in signed networks. It was the fundamental model in sign prediction of edges. Reference [11] employed the idea of signed triads and used logistic regression model and some local feature in order to predict sign of edges in social networks. The features that were introduced by [11] are categorized in two classes: first one is on the base of the positive/negative ingoing/outgoing degree of nodes which basically collect the local information of nodes. And the second group is based on the extracted principles from social psychology, in which we are able to determine the type of u and v relation by utilizing the information of third party like w.

Ranking of nodes has tight relationship with sign prediction problem so we also investigate ranking in signed networks. Ranking of nodes is the problem of computing how much important or trustable a node is in networks [12]. The centrality measures like betweenness [13], closeness [14], and eigenvector centrality [15] were introduced to compute nodes' importance degree in the network. Other algorithms like HITS [16] and PageRank [17] were added in 1990. All of these ranking algorithms are designed for positive graphs and there are merely several literatures for ranking of nodes in signed networks. The simplest ranking algorithm for signed graphs is prestige, where number of positive and negative incoming links determine ranking of each node [18]. Another ranking algorithm is PageTrust that was introduced by [19]. This method is extension of PageRank, and the main difference is that nodes with negative incoming links will be visited less in random walk process. Exponential ranking is another chief method of ranking for signed graphs [12]. In exponential ranking, the value of ranking vector globally is obtained from local trust values. Another ranking algorithm for signed networks that is greatly similar to HITS was proposed by [20]. This method utilizes the concept of Bias and Deserve which underestimates the vote of optimistic and pessimistic nodes. Reference [3] also proposed new ranking algorithms for signed networks, namely, Modified HITS and Modified PageRank.

Because we propose community-based ranking algorithms, we should go through community detection problem. Community detection algorithms help to prepare more dominant recommendation systems and web page clustering which have great effect on better searches [21]. Community detection algorithms attempt to cluster edges/nodes in order to have minimum number of edges between densely communities [22]. One of the most widely used methods for community detection in unsigned graphs was proposed by [23]. As for signed networks, [24] proposed a two-step spectral approach which was an extension to modularity. The main problem related to modularity is resolution limit in which very small communities might not be detected. In order to address this problem, [22] proposed new method for detecting communities on signed graphs by extending potts model. Reference [25] also introduced useful approach that works on the base of blocking method.

3. Method

In this paper, authors intend to investigate the community-based problem of predicting sign of links in signed social networks. Hence, in this section as well as proposed algorithms and methods, the problems of sign prediction and community detection will be discussed in detail.

3.1. Edge Sign Prediction. In order to define the problem formally, it can be assumed that we have a signed directed graph $G(V, E)$ that V represents set of vertices and E shows

set of edges where customers and users can vote positively and negatively toward each other. So the aforementioned notation V represents users of site and E indicates +1 and −1 relations among them. In all over the paper, the person who gives positive vote and receives it, is named trustor and trustee, respectively [2, 26]. The sign prediction problem can be defined as follows. Suppose that signs of some links in the network are hidden, and the goal is to reliably predict values of these edges by current information in the graph. The sign prediction problem tries to find signs of hidden edges with negligible error [1, 27]. In this work we propose state-of-the-art community-based ranking algorithms and we evaluate their effectiveness via sign prediction problem on three datasets: Epinoins, Slashdot, and Wikipedia.

To this end, [7] already introduced rank-based features named Optimism and Reputation to connect ranking problem with sign prediction. Rank-based reputation of node i indicates patterns of voting toward this node. Meaningfully, rank-based reputation of node i not only considers number of positive/negative incoming links toward node i but also it takes into account ranks of nodes who vote toward node i. In other words, when a person receives several positive incoming links, s/he might not be very reputable because one should consider rank of voters toward node i. If the users who vote toward node i are high rank, then node i can be considered reputable, but if they are not high rank we cannot say that node i is reputable although the number of positive incoming links toward node i is relatively high. The following equation can better describe rank-based reputation [7]:

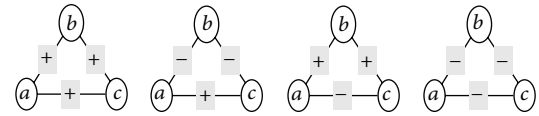
$$\mathrm{RBR}_i = \frac{\left|R_{\mathrm{in}}^{(+)}(i)\right| - \left|R_{\mathrm{in}}^{(-)}(i)\right|}{\left|R_{\mathrm{in}}^{(+)}(i)\right| + \left|R_{\mathrm{in}}^{(-)}(i)\right|}, \tag{1}$$

where RBR_i is the value of rank-based reputation of node i, $R_{\mathrm{in}}^{(+)}(i)$ indicates sum of rank values of nodes who positively voted toward node i, and, similarly, $R_{\mathrm{in}}^{(-)}(i)$ is sum of rank values of nodes who negatively voted toward node i. In the same vein, one can define rank-based optimism of node i as follows:

$$\mathrm{RBO}_i = \frac{\left|R_{\mathrm{out}}^{(+)}(i)\right| - \left|R_{\mathrm{out}}^{(-)}(i)\right|}{\left|R_{\mathrm{out}}^{(+)}(i)\right| + \left|R_{\mathrm{out}}^{(-)}(i)\right|}, \tag{2}$$

where RBO_i is the value of rank-based optimism of node i, $R_{\mathrm{out}}^{(+)}(i)$ refers to sum of rank values of nodes whom node i positively voted toward them, and similarly, $R_{\mathrm{out}}^{(-)}(i)$ is sum of rank values of nodes in which node i negatively voted toward them. As formula (2) shows, node i which generates several positive outgoing links might not be optimistic because this set might contain nodes in which they are low rank [3]. In order to compute these features, we need algorithms to rank nodes. As for ranking algorithms, we propose three community-based ranking algorithms in the next sections.

3.2. Community-Based Ranking Algorithms to Compute RBR and RBO. In this section we propose three ranking algorithms in which all of them work based on community detection problem in signed graphs. In other words, firstly, we run a community detection algorithm on signed networks. The results will be disjoint communities of nodes. As all community detection algorithms work based on a density based approach in which they try to maximize density of intracluster edges and minimize between cluster edges, so intracluster nodes are more dense and close. From social perspective, intracluster nodes might know each other better (this is the notion behind our community-based ranking algorithm). Meaningfully, nodes in the same community are much more familiar than nodes that are in different communities. Via using this philosophy about intracluster nodes, we change previous ranking algorithm like Prestige, HITS, and PageRank [3] to have influence of intra- and extracluster nodes with parameters α and $(1-\alpha)$, respectively. Then we can use ranking-based features of [7] for the case of sign prediction. Because first phase of the algorithms is community detection, so we investigate community detection problem and a sample community detection algorithm in signed graphs in Section 3.2.1. In this paper a community detection algorithm based on social balance theory is utilized. In Section 3.3, ranking algorithms based on community detection phase are introduced.



(a) Stable configurations (b) Unstable configurations

Figure 1: All possible states of balance theory.

3.2.1. Community Detection. The algorithm used in this paper is based on structural balance theory [28]. In balance theory, there are four possible states when nodes are in signed relations in social networks [29]. One can differentiate these states by number of positive and negative edges in each triad [30]. On the base of strong social balance theory, when all of nodes have positive relation or two nodes share the same enemy, these states are called stable. Similarly, cases with all nodes have negative edges or with two positive edges are unstable states [31]. Regarding this definition, a network with more than three nodes is structurally balanced if all the possible triads are stable [32] (Figure 1).

The basic structural theorem states that these triples can be partitioned into two distinct sets in which all the positive relations are inside sets and negative ones are among them [33, 34]. In other words, negative edges connect positive sets. On the basis of this definition, a network is called k-balanced if all positive edges are located in k number of different categories, and these sets are joined with negative relations [35]. In reality, rarely there are structurally balanced networks. There are always some edges that destabilize the graph and transform it into unstable configuration. Therefore, the number of positive edges between clusters and the number of negative edges inside clusters should be minimized [24]. In fact, the problem is like finding the best sets with minimum number of positive relations between partitions and also minimum number of negative edges inside sets [36].

Reference [25] introduced one criterion function which makes decision based on counting number of elements having conflict with k-balanced theory. It can be defined as if one considers N as number of negative edges inside clusters, and P as number of positive edges between clusters, then number of inconsistencies which is denoted by $I(c)$ can be mentioned as

$$I(c) = \beta \times N + (1 - \beta) \times P, \tag{3}$$

where β is the importance factor that is assigned to positive and negative inconsistencies.

If $\beta = 0.5$ then positive and negative relations contribute equally on amount of inconsistency. And for the case that $0.5 < \beta \leq 1$ the negative relations have more impact on result and, when $0 < \beta \leq 0.5$, positive edges have higher influence. The ideal condition is created when P and N have the smallest amount, so better result is achieved. Because $I(c)$ show error, the algorithm tries to find minimum value for N and P via using a hill-climbing optimization technique [25]. Other community detection approaches suitable for signed graphs can also be used in this phase.

3.3. Community-Based Ranking Methods. In this paper, we propose three new ranking algorithms for signed complex networks that are dependent on community detection. We introduce a method that ties the concept of ranking algorithm and community detection. Suppose that we intend to compute the rank of node i in the network. First of all, we cluster nodes in the network in such a way that each node belongs to one community in the graph (first phase, algorithm referenced in Section 3.2.1), so for calculating rank of node i by taking influence of other nodes in the network, we give priority and high importance to the nodes that belong to the same community, which node i belongs to. These algorithms are described in detail, in the following subsections (second phase of ranking). We will verify rationality of these ranking algorithms in Results section.

3.3.1. PBCD (Prestige Ranking Algorithm Based on Community Detection). Prestige is the simplest algorithm in signed complex network [18]. In this method, the most important factor for determining ranking of each node in the system is the number of positive and negative incoming nodes that each node receives from others. In other words, if a node has many positive incoming links, therefore, its prestige is high in the network. And it is also true for negative links, if the number of positive incoming links is less than negative ones, the node has low prestige in comparison with the other nodes in the system [3]. The idea of community-based ranking inspires us to incorporate our method with some well-known ranking algorithms like prestige. The proposed prestige can be stated as follows:

$$\begin{aligned}\Pr(i) = {} & \left[\alpha \times \frac{\left|id_{\text{in}}^{(+)}(i)\right| - \left|id_{\text{in}}^{(-)}(i)\right|}{\left|id_{\text{in}}^{(+)}(i)\right| + \left|id_{\text{in}}^{(-)}(i)\right|}\right] \\ & + \left[(1-\alpha) \times \frac{\left|od_{\text{in}}^{(+)}(i)\right| - \left|od_{\text{in}}^{(-)}(i)\right|}{\left|od_{\text{in}}^{(+)}(i)\right| + \left|od_{\text{in}}^{(-)}(i)\right|}\right],\end{aligned} \tag{4}$$

where α is the impact factor to determine degree of importance of nodes that are in the same community that node i belongs to, and $id_{\text{in}}^{+}(i)$ and $id_{\text{in}}^{-}(i)$, respectively, indicate set of nodes who positively and negatively voted toward i and these nodes are members of the same community that node i belongs to. Similarly, $od_{\text{in}}^{+}(i)$ and $od_{\text{in}}^{-}(i)$ show set of nodes who positively and negatively voted toward node i and these nodes are members of other communities which are different from the community that node i belongs to. Moreover, $||$ represents magnitude of the set of nodes and the in subscript indicates that we only consider incoming links from $id(i)$ and $od(i)$ sets toward node i. Finally, $i \in c, c \in C$: c is the community which node i belongs to and C is the disjoint subgraph of all clusters detected via the algorithm. In this notation, all members of $id(i)$ are in c and none of $od(i)$ members are in c. As intraclusters nodes of communities are more close to each other, ranking algorithm can utilize the influence of intra clusters nodes closeness.

3.3.2. HBCD (HITS Ranking Algorithm Based on Community Detection). HITS algorithm was introduced by [16], and it was mainly proposed for exploiting helpful information in order to analyze structure of links and has been applied in various applications. This algorithm works based on hub and authority vectors. These vectors are initialized with some predefined (random) values and converge after some recursive iterations [16].

Reference [3] introduced modified version of HITS in which the graph is divided into two positive and negative parts and then run the algorithm on each graph separately. In HITS algorithm, there are two vectors: authority and hub, which finally converge after enough iteration. We propose a new version of HITS in which there is distinction between importance of local neighbor nodes and those members of different communities that node i belongs to. The HITS algorithm based on community detection can be stated as follows:

$$\begin{aligned} a_i^{(+)}(t+1) = {} & \alpha \times \left(\sum_{j \in id_{\text{out}}^{(+)}(i)} h_j^{(+)}(t)\right) \\ & + (1-\alpha) \times \left(\sum_{j \in od_{\text{out}}^{(+)}(i)} h_j^{(+)}(t)\right), \\ a_i^{(-)}(t+1) = {} & \alpha \times \left(\sum_{j \in id_{\text{out}}^{(-)}(i)} h_j^{(-)}(t)\right) \\ & + (1-\alpha) \times \left(\sum_{j \in od_{\text{out}}^{(-)}(i)} h_j^{(-)}(t)\right), \\ h_i^{(+)}(t+1) = {} & \alpha \times \left(\sum_{j \in id_{\text{in}}^{(+)}(i)} a_j^{(+)}(t)\right) \\ & + (1-\alpha) \times \left(\sum_{j \in od_{\text{in}}^{(+)}(i)} a_j^{(+)}(t)\right), \end{aligned}$$

$$h_i^{(-)}(t+1) = \alpha \times \left(\sum_{j \in id_{\text{in}}^{(-)}(i)} a_j^{(-)}(t) \right) + (1-\alpha) \times \left(\sum_{j \in od_{\text{in}}^{(-)}(i)} a_j^{(-)}(t) \right), \tag{5}$$

where α indicates importance factor that is given to the local neighbors and $h(t)$ and $a(t)$ show hub and authority vectors at time t. $id_{\text{in/out}}^{(+)}(i)$ and $id_{\text{in/out}}^{(-)}(i)$, respectively, represent set of nodes who have relations (positive/negative or incoming/outgoing) with node i and they are members of the same community that node i belongs to. In a similar manner, $od_{\text{in/out}}^{(+)}(i)$ and $od_{\text{in/out}}^{(-)}(i)$ indicate set of nodes in which they have relations (positive/negative, incoming/outgoing) with node i and they are members of different communities that node i belongs to. Finally, in and out subscripts, respectively, show that $id(i)$ or $od(i)$ represents set of nodes that voted toward node i (incoming links toward node i) or being voted by node i (outgoing links from node i). Hub and authorities are initialized with some random values and they converge after enough iteration.

3.3.3. RBCD (PageRank Algorithm Based on Community Detection). PageRank is one of the most widely used methods for ranking of nodes [37]. It was extracted from Google Larry page. PageRank uses the concept of random walk that leads to probability distribution which computes the possibility of randomly going from one node to another one, and finally gets to one specific node. This algorithm initially was introduced for graphs with positive and unsigned edges, especially for web pages on the internet. Reference [3] introduced modified version of PageRank in which the graph is divided into two parts and the algorithm is run on each graph separately, and finally for calculating ranking vector, negative ranking vector is subtracted from positive one. Our proposed ranking algorithm states that each node in the network belongs to specific community or cluster, and the general idea of specifying ranking is on the basis of utilizing other nodes' information. In other words, to compute rank of node i, we give priority to the nodes that belong to the same community that node i belongs to. Based on this opinion, PageRank based on community detection can be defined as

$$\text{PR}_i^{(+)}(t+1) = \alpha \times \left(\beta \times \sum_{j \in id_{\text{in}}(i)} \frac{\text{PR}_j^{(+)}(t)}{\left| d_{\text{out}}^{(+)}(j) \right|} + (1-\beta) \times \frac{1}{N} \right) + (1-\alpha) \times \left(\beta \times \sum_{j \in od_{\text{in}}(i)} \frac{\text{PR}_j^{(+)}(t)}{\left| d_{\text{out}}^{(+)}(j) \right|} + (1-\beta) \times \frac{1}{N} \right),$$

$$\text{PR}_i^{(-)}(t+1) = \alpha \times \left(\beta \times \sum_{j \in id_{\text{in}}(i)} \frac{\text{PR}_j^{(-)}(t)}{\left| d_{\text{out}}^{(-)}(j) \right|} + (1-\beta) \times \frac{1}{N} \right) + (1-\alpha) \times \left(\beta \times \sum_{j \in od_{\text{in}}(i)} \frac{\text{PR}_j^{(-)}(t)}{\left| d_{\text{out}}^{(-)}(j) \right|} + (1-\beta) \times \frac{1}{N} \right), \tag{6}$$

where in the above equations α denotes importance factor is assigned to the local neighbor nodes, and t shows the rank at the time of t. β indicates the forgetting factor, and N represents number of nodes in the graph. $id_{\text{in}}(i)$ shows set of incoming links to node i that belong to the same community that node i belongs to. Similarly $od_{\text{in}}(i)$ indicates set of incoming links to node i that belong to different communities that node i belongs to. $d_{\text{out}}^{(+)}(j)$ and $d_{\text{out}}^{(-)}(j)$ involves number of positive and negative outgoing links from node j, respectively.

3.4. Classifier. Classification is the process of assigning data to one of predetermined classes. Here our classes are the mapped values of positive and negative signs to +1 and −1 values. This process is done via using training set which contains some features to constitute a classifier and then evaluation via using a test set. A brief explanation of the classifier used in our work is brought here.

Logistic regression: logistic regression or sigmoid function is a monotonic, continuous function which lies between −1 and +1 values and is a method of learning function of the form $f : X \rightarrow Y$ or $P(Y \mid X)$. They can be mathematically defined as follows:

$$P(Y = 1 \mid x) = \frac{1}{1 + \exp\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)},$$
$$P(Y = -1 \mid x) = \frac{1}{1 + \exp\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)}. \tag{7}$$

Y is the discrete values of classes which here are +1 and −1, X is the input vector of discrete or continuous values, and w_i are some learning coefficient learnt by the model. Classifiers are evaluated based on accuracy. Accuracy metric is calculated based on TP, FP, TN, and FN as follows [38, 39]:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{8}$$

We used 10-fold validation in order to evaluate generalization of the models. Cross validation is a classical method in which the dataset are partitioned into k folds. The first one is used as test set and the remaining folds are considered as training sets. In the next phase, the second fold is selected as the test set and the remaining are chosen as training sets [40]. We utilized *WEKA* software for computing accuracy that is available through http://www.cs.waikato.ac.nz/ml/weka.

4. Experiments

In order to verify proposed algorithms introduced in Section 3.3 we executed them on three large datasets.

In the following sections these datasets are introduced and achieved results are depicted.

4.1. Datasets. Datasets which are used for experimental purposes are Wikipedia, Epinions, and Slashdot. These online social networks are available through http://snap.stanford edu/data/. In the following, we explain briefly these datasets.

Wikipedia. Volunteers from all around the world collaborate to write this free cyclopedia. A user can take the role of administrator with additional access to some technical features by getting vote of other users. Administrators are responsible for maintenance purposes. A user is nominated as administrator, and then Wikipedia members elect users as administrators via public dialogues and talks. Totally, 7000 users took part in elections, and 100,000 votes were received from 2800 delegated elections. 1200 elections lead to promotion and 1500 elections were not successful and did not produce a winner. Half of voters are current administrators and the remaining are ordinary users [6].

Epinions. Epinions is a review online social network that users can express their views about diverse items like music, TV shows, and hardware. The site members are able to vote positively and negatively toward each other. As a matter of fact, this online social network contains information about who-trusts-whom. The data set is made of 131828 nodes and contains 841372 edges in which 85% of them are positive [41].

Slashdot. Slashdot is a website related to technology and science. It is well known due to its specific users and has introduced itself as user-submitted and science evaluated news. In other words, links of news and summary of different issues are submitted by users, and each story becomes the topic of series of talks. Selected readers that take the role of moderators send ratings to Slashdot. The responsibility of these readers is appointing tags for each comment. Slashdot does not show scores to the users but lets them arrange comments on the base of assigned points. Slashdot also has a service that users have the ability to tag each other as friend or opponent. Therefore, links between users are friend/opponent relationship. The data set is made of 82144 nodes and contains 549202 edges in which 77% of them are positive [42].

4.2. Results. Via using introduced community detection of Section 3.2.1, the communities in Epinions, Slashdot, and Wikipedia are detected. We examine our proposed method on different size of communities, where it can be specified through input parameter of community detection algorithm. Reference [43] introduced a function for determining number of clusters in Epinions, Slashdot, and Wikipedia, and it can be observed that this method has high accuracies when the number of clusters is between four and ten, so we also checked our approach for seven cases starting from four to ten. For ranking nodes, Prestige Based on Community Detection (PBCD), PageRank Based on Community Detection (RBCD), and HITS Based on Community Detection (HBCD) are run on these datasets. In order to differentiate between nodes which belong to various communities, we defined

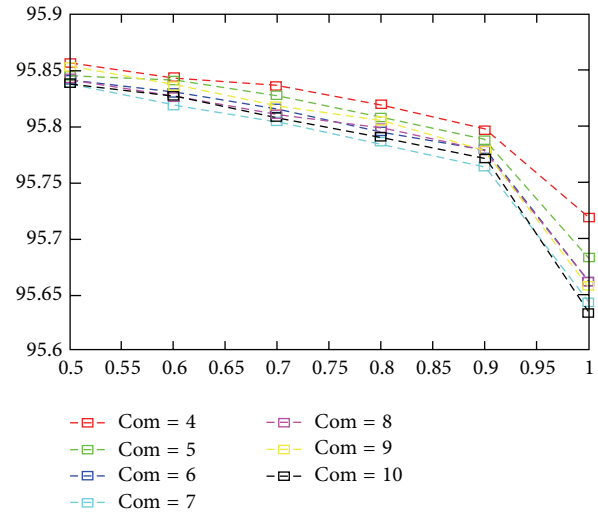


Figure 2: y axis shows prediction accuracy of PBCD ranking algorithm on Epinions dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

impact factor α for local nodes and $(1 - \alpha)$ for foreign ones where α can contains values of $(0.5, 0.6, \ldots, 0.9, 1)$. When $0.5 < \alpha \leq 1$, local nodes have more privilege than the others, and $\alpha = 0.5$ shows normal ranking algorithms of [3]. In other words, in the case ($\alpha = 0.5$), no community structure is considered. In order to define accuracy of edge sign prediction, rank based features are extracted and in order to evaluate generalization of the models we used 10-fold cross validation. Accuracy of introduced methods is evaluated with various size of communities and different values of α. The results are shown in forms of figures. In the following, significant findings related to each dataset are stated.

4.2.1. PBCD on Datasets. In Epinions and Slashdot, for all size of communities, $\alpha = 0.5$ contains the maximum values. In fact outputs of PBCD on Epinions and Slashdot datasets indicate that using community-based ranking algorithms might slightly degrade prediction accuracy. However, the prediction accuracy is still high for different values of α greater than 0.5 and for all community numbers. This lower accuracy might be because of special property of dataset or community detection algorithm. Results for Epinions and Slashdot are illustrated in Figures 2 and 3, respectively. Results related to Wikipedia are shown in Figure 4. In the case Com (number of communities) equals 4, 6, and 7, prediction accuracy is higher than the case $\alpha = 0.5$. Generally high precisions extract properties of dataset and offers better model for prediction. In Figure 4, where Com = 6, $\alpha = 0.6$ contains maximum value which is equal to 88.7467.

4.2.2. RBCD on Datasets. Analogously to PBCD, we implemented RBCD ranking algorithm on datasets. Outputs for Epinions are illustrated in Figure 5. In Epinions, all the number of communities has acceptable accuracies. Com = 10 with $\alpha = 1$ contains the maximum value of 95.515 among

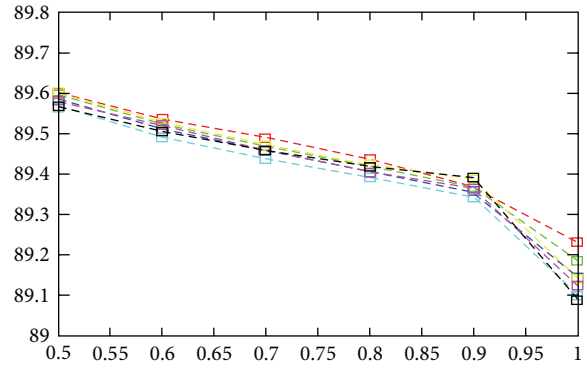


Figure 3: y axis shows prediction accuracy of PBCD ranking algorithm on Slashdot dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

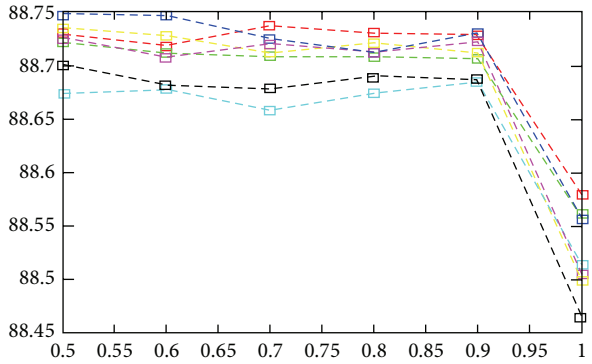


Figure 4: y axis shows prediction accuracy of PBCD ranking algorithm on Wikipedia dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

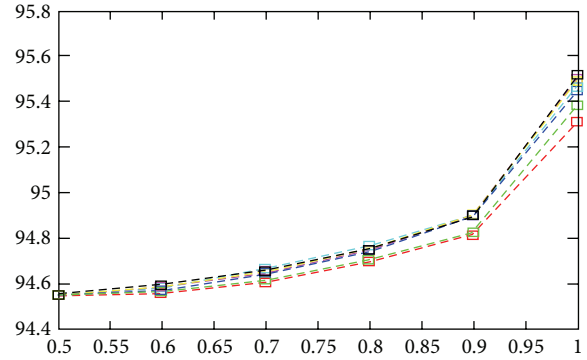


Figure 5: y axis shows prediction accuracy of RBCD ranking algorithm on Epinions dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

all communities. RBCD have similar outputs in Slashdot; when $\alpha = 1$, it has highest precision with maximum value of 89.335 for 10 communities. Results related to Slashdot are represented in Figure 6. RBCD also produced satisfactory results in Wikipedia. When $\alpha = 0.5$ it contains minimum value of 88.350 among all communities. In case of Com = 8,

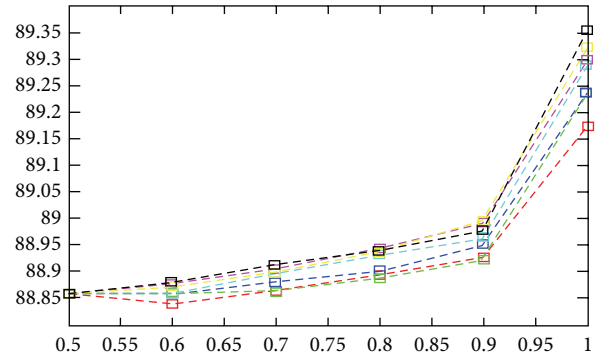


Figure 6: y axis shows prediction accuracy of RBCD ranking algorithm on Slashdot dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

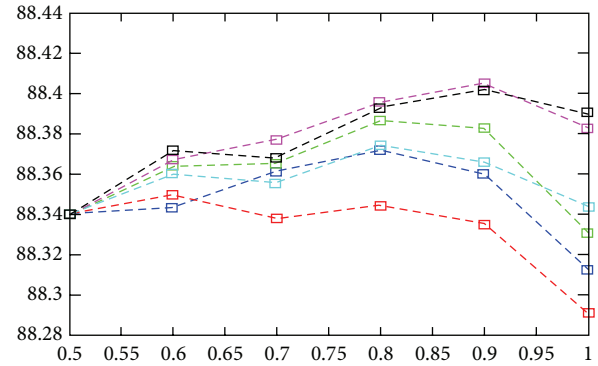


Figure 7: y axis shows prediction accuracy of RBCD ranking algorithm on Wikipedia dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

maximum accuracy of 88.405 is for $\alpha = 0.9$. When Com = 5, 6, 7, RBCD reach maximum accuracies at $\alpha = 0.8$ in which their values are 88.386, 88.372, and 88.374, respectively.

Similarly, for communities nine and ten maximum values are 88.4021 and 88.402, respectively, with $\alpha = 0.9$. Outputs for Wikipedia are shown in Figure 7.

4.2.3. HBCD on Datasets. We also implemented HBCD on datasets. Achieved results for Epinions are shown in Figure 8. In Epinions, $\alpha = 1$ has best accuracy for all communities. Moreover, community number four with value of 95.620 contains maximum among them. As for Slashdot, $\alpha = 1$ has the best accuracy and Com = 9 generates accuracy of 89.37 which has the highest value among all communities. Outputs for Slashdot are illustrated in Figure 9. There is the same story in Wikipedia. $\alpha = 1$ has the best accuracy and proves our new method. Among all of them community number seven has the best accuracy with value of 88.410. Related results for Wikipedia are presented in Figure 10.

5. Discussion

Random guessing of edge sign prediction on original datasets results in accuracy prediction of approximately 80 percent.

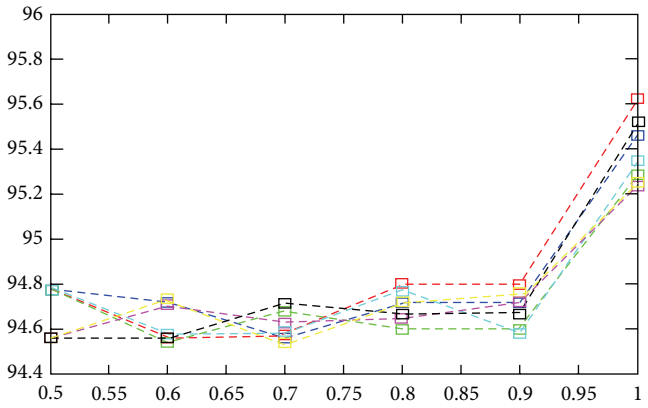


Figure 8: y axis shows prediction accuracy of HBCD ranking algorithm on Epinions dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

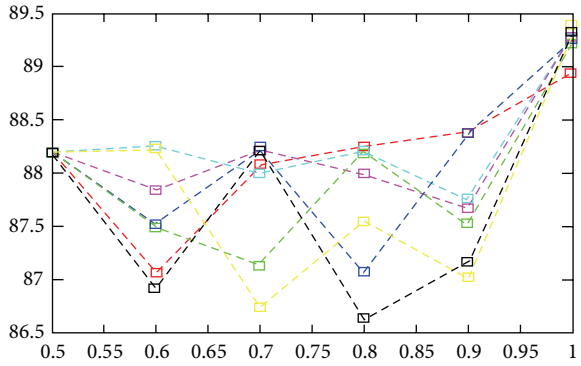


Figure 9: y axis shows prediction accuracy of HBCD ranking algorithm on Slashdot dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

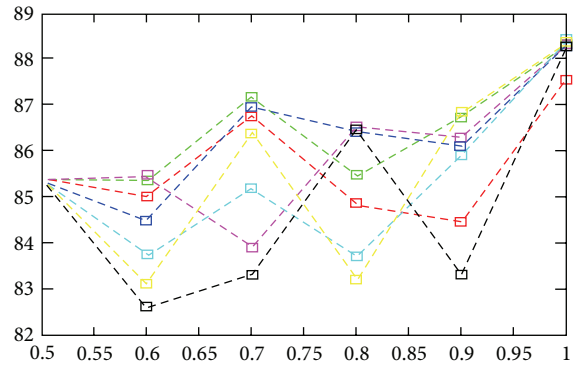


Figure 10: y axis shows prediction accuracy of HBCD ranking algorithm on Wikipedia dataset and x axis indicates different values of α. Different colors show number of communities detected by the community detection algorithm.

Table 1: Prediction accuracy using various methods on original datasets.

	Epinion	SlashDot	Wikipedia
Prediction accuracy of Leskovec [5]			
Leskovec	90.751	87.117	83.835
Prediction accuracy of Shahriari [3]			
Normal prestige	95.872	89.604	88.747
MPR	94.550	88.859	88.34
MHITS	94.770	88.185	88.359
Prediction accuracy achieved from our proposed algorithms			
PBCD	95.853	89.536	88.747
RBCD	95.515	89.355	88.405
HBCD	95.629	89.380	88.405

As Table 1 indicates accuracy of our work improves the rate of prediction about 10 to 15 percent.

Reference [5] applied degree-based features on Epinions, Slashdot, and Wikipedia and performed sign prediction with precisions of 90.751, 87.117, and 83.835, respectively. It can be perceived in Table 1 that the prediction rate of our work has significant improvement in comparison to [5].

Reference [3] also introduced new ranking algorithms, namely, MPR, MHITS, and improved precision achieved by [5]. In order to compare result of this paper with [3], we can consider $\alpha = 0.5$, as normal Prestige, MPR, and MHITS. In other words, in our proposed formulas, $\alpha = 0.5$ indicates that nodes inside and outside community have the same privilege. All in all, except precision of PBCD on Epinions and Slashdot, in other cases, our method produces better results in comparison with [3]. To put in a nutshell, we find out that our proposed approach outperforms previous works related to sign prediction problem, and it is indicated that local nodes in communities have higher impact on reputation and importance of other nodes in the network. Moreover, number of communities can be estimated in these real-world datasets by analyzing output presented in the previous section. For example, in Wikipedia, PBCD algorithm produces best accuracies for community number six, and community number eight yields the best result for RBCD. Similarly HBCD detect seven communities in Wikipedia. It is very obvious that result of each algorithm is different with another one but it can be easily perceived that all of these numbers are close to each other. Therefore we can deduce that these community-based ranking algorithms are able to approximate the number of communities in these datasets.

6. Conclusion and Future Works

Complex networks have multidisciplinary roles in science comprising artificial intelligence, economics, and chemistry in which their usages have been increasing. Social networks as one branch of complex networks have got a lot of attention recently. One principal topic in social networks is to investigate evolution of graphs; thus researchers are trying to take prediction algorithms in order to find hidden relationships in these networks. A significant factor in predicting relations is ranking of people in societies. The aforementioned concepts

are expressed in social networks as sign prediction and ranking of nodes, respectively.

Nodes ranking algorithms which intend to determine how much a node is reputable in a network are studied in our document. Three community-based ranking algorithms are proposed in this paper. These ranking algorithms have two phases. In the first phase, nodes are assigned to different communities by applying community detection algorithm (in this phase, different community detection algorithms can be applied). In the second phase, rank of each node is computed based on its membership to its neighbors' communities and its incoming/outgoing positive/negative links. So we investigated the effect of community detection on the accuracy of sign prediction problem and compared our work with [3, 5]. Eventually, we deduced that our ranking algorithms outperform both methods and community-based ranking algorithms produce better accuracies in some cases.

Our experiment was performed to check which community number has the best accuracy. In this case, results may be affected by properties of this community detection algorithm. In future research, we intend to compare impact of different community detection methods on accuracy of edge sign prediction problem. The problem of overlapping community detection, especially, has gained much attention. Hence this problem and its effect on signed prediction can be investigated. We are also interested in working on parameter-free community detection methods suitable for signed graphs. Finally, to check the reliability of the method, it is good to test these approaches in person to person recommenders.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. Liben-Nowell and J. Kleinberg, " The link prediction problem for social networks," in *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM '03)*, pp. 556–559, November 2003.

[2] Y. Dong, J. Tang, S. Wu et al., "Link prediction and recommendation across heterogeneous social networks," in *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM '12)*, pp. 181–190, December 2012.

[3] M. Shahriari and M. Jalili, "Ranking nodes in signed social networks," *Social Network Analysis and Mining*, vol. 4, article 172, 2014.

[4] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The Slashdot Zoo: mining a social network with negative edges," in *Proceedings of the 18th International World Wide Web Conference (WWW '09)*, pp. 741–750, April 2009.

[5] 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*, pp. 641–650, New York, NY, USA, April 2010.

[6] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon, "Exploiting longer cycles for link prediction in signed networks," in *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM '11)*, pp. 1157–1162, October 2011.

[7] M. Shahriari, O. A. Sichani, J. Gharibshah, and M. Jalili, "Predicting sign of edges in social networks based on users reputation and optimism," *Transactions on Knowledge Discovery from Data*. In press.

[8] L. Adamic and E. Adar, "How to search a social network," *Social Networks*, vol. 27, no. 3, pp. 187–203, 2005.

[9] J. Chen, W. Geyer, C. Dugan, M. Muller, I. Guy, and M. Carmel, "Make new friends, but keep the old: recommending people on social networking sites," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*, pp. 201–210, 2009.

[10] P. Symeonidis, E. Tiakas, and Y. Manolopoulos, "Transitive node similarity for link prediction in social networks with positive and negative links," in *Proceedings of the 4th ACM Recommender Systems Conference (RecSys '10)*, pp. 183–190, September 2010.

[11] 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*, pp. 641–650, 2010.

[12] V. A. Traag, Y. Nesterov, and P. Van Dooren, "Exponential ranking: taking into account negative links," in *Social Informatics*, vol. 6430 of *Lecture Notes in Computer Science*, pp. 192–202, Springer, Berlin, Germany, 2010.

[13] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[14] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978.

[15] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *The Journal of Mathematical Sociology*, vol. 2, no. 1, pp. 113–120, 1972.

[16] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[17] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 56, no. 18, pp. 3825–3833, 2012.

[18] K. Zolfaghar and A. Aghaie, "Mining trust and distrust relationships in social Web applications," in *Proceedings of the 6th IEEE International Conference on Intelligent Computer Communication and Processing*, pp. 73–80, 2010.

[19] C. de Kerchove and P. van Dooren, "The PageTrust algorithm: how to rank web pages when negative links are allowed?" in *Proceedings of the 8th SIAM International Conference on Data Mining*, pp. 346–352, April 2008.

[20] A. Mishra and A. Bhattacharya, "Finding the bias and prestige of nodes in networks based on trust scores," in *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*, pp. 567–576, April 2011.

[21] P. Anchuri and M. M. Ismail, "CommunityDetection in Signed Networks".

[22] V. A. Traag and J. Bruggeman, "Community detection in networks with positive and negative links," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 80, no. 1, Article ID 036115, 7 pages, 2009.

[23] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[24] P. Anchuri and M. Magdon-Ismail, "Communities and balance in signed networks: a spectral approach," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 235–242, August 2012.
[25] P. Doreian and A. Mrvar, "Partitioning signed social networks," *Social Networks*, vol. 31, no. 1, pp. 1–11, 2009.
[26] T. Zhang, H. Jiang, Z. Bao, and Y. Zhang, "Characterization and edge sign prediction in signed networks," *Journal of Industrial and Intelligent Information*, vol. 1, no. 1, pp. 19–24, 2013.
[27] S. H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang, "Friend or frenemy? Predicting signed ties in social networks," in *Proceedings of the 35th Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*, pp. 555–564, August 2012.
[28] D. Cartwright and F. Harary, "Structural balance: a generalization of Heider's theory," *Psychological Review*, vol. 63, no. 5, pp. 277–293, 1956.
[29] P. Doreian, "Evolution of human signed networks," *metodološki Zvezki*, vol. 1, no. 2, pp. 277–293, 2004.
[30] M. Szell, R. Lambiotte, and S. Thurner, "Multirelational organization of large-scale social networks in an online world," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 31, pp. 13636–13641, 2010.
[31] S. Maniu, B. Cautis, and T. Abdessalem, "Building a signed network from interactions in Wikipedia," *Proceedings of the 1st ACM SIGMOD Workshop on Databases and Social Networks (DBSocial '11)*, pp. 19–24, 2011.
[32] S. R. T. Antal, P. L. Krapivsky, and S. Redner, "Social balance on networks: the dynamics of friendship and enmity," *Physica D. Nonlinear Phenomena*, vol. 224, no. 1-2, pp. 130–136, 2006.
[33] P. Doreian, "A multiple indicator approach to blockmodeling signed networks," *Social Networks*, vol. 30, no. 3, pp. 247–258, 2008.
[34] S. A. Marvel, J. Kleinberg, R. D. Kleinberg, and S. H. Strogatz, "Continuous-time model of structural balance," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, no. 5, pp. 1771–1776, 2011.
[35] N. P. Hummon and P. Doreian, "Some dynamics of social balance processes: bringing Heider back into balance theory," *Social Networks*, vol. 25, no. 1, pp. 17–49, 2003.
[36] G. Adejumo, P. R. Duimering, and Z. Zhong, "A balance theory approach to group problem solving," *Social Networks*, vol. 30, no. 1, pp. 83–99, 2008.
[37] L. Page, S. Brin, R. Motawani, and T. Winograd, "The page rank citation ranking:bringing order to the web," 1999.
[38] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, USA, 2006.
[39] T. M. Mitchell, *Machine Learning*, McGraw Hill, 1st edition, 1997.
[40] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*, pp. 1477–1488, 2013.
[41] J. Kunegis, "What is the added value of negative links in online social networks?" in *Proceedings of the 22nd International Conference on World Wide Web*, pp. 727–736, 2013.
[42] K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. S. Dhillon, and A. Tewari, "Prediction and clustering in signed networks: a local to global perspective," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1177–1213, 2014.
[43] A. Javari and M. Jalili, "Cluster-based collaborative filtering for sign prediction in social networks with positive and negative links," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 2, article 24, 2014.

Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 461362, 18 pages
http://dx.doi.org/10.1155/2015/461362

Research Article

Parallelizing SLPA for Scalable Overlapping Community Detection

Konstantin Kuzmin,[1] Mingming Chen,[1] and Boleslaw K. Szymanski[1,2]

[1] *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*
[2] *The Faculty of Computer Science and Management, Wrocław University of Technology, 50-370 Wrocław, Poland*

Correspondence should be addressed to Konstantin Kuzmin; kuzmik@rpi.edu

Received 3 March 2014; Accepted 17 November 2014

Academic Editor: Przemyslaw Kazienko

Communities in networks are groups of nodes whose connections to the nodes in a community are stronger than with the nodes in the rest of the network. Quite often nodes participate in multiple communities; that is, communities can overlap. In this paper, we first analyze what other researchers have done to utilize high performance computing to perform efficient community detection in social, biological, and other networks. We note that detection of overlapping communities is more computationally intensive than disjoint community detection, and the former presents new challenges that algorithm designers have to face. Moreover, the efficiency of many existing algorithms grows superlinearly with the network size making them unsuitable to process large datasets. We use the Speaker-Listener Label Propagation Algorithm (SLPA) as the basis for our parallel overlapping community detection implementation. SLPA provides near linear time overlapping community detection and is well suited for parallelization. We explore the benefits of a multithreaded programming paradigm and show that it yields a significant performance gain over sequential execution while preserving the high quality of community detection. The algorithm was tested on four real-world datasets with up to 5.5 million nodes and 170 million edges. In order to assess the quality of community detection, at least 4 different metrics were used for each of the datasets.

1. Introduction

Analysis of social, biological, and other networks is a field which attracts significant attention as more and more algorithms and real-world datasets become available. In social science, a community is loosely defined as a group of individuals who share certain common characteristics [1]. Based on similarity of certain properties, social agents can be assigned to different social groups or communities. Knowledge of communities allows researchers to analyze social behaviors and relations between people from different perspectives. As social agents can exhibit traits specific to different groups and play an important role in multiple groups, communities can overlap. Usually, there is no a priori knowledge of the number of communities and their sizes. Quite often, there is no ground truth either. Knowing the community structure of a network empowers many important applications. Communities can be used to model, predict, and control information dissemination. Marketing companies, advertisers, sociologists, and political activists are able to target specific interest groups. The ability to identify key members of a community provides a potential opportunity to influence the opinion of the majority of individuals in the community. Ultimately, the community opinion can be changed when only a small fraction of the most influential nodes accepts a new opinion [2].

Biological networks such as neural, metabolic, protein, genetic, or pollination networks and food webs model interactions between components of a system that represent some biological processes [3]. Nodes in such networks often correspond to genes, proteins, individuals, or species. Common examples of such interactions are infectious contacts, regulatory interaction, and gene flow.

The majority of community detection algorithms operate on networks which might have strong data dependencies between the nodes. While there are clearly challenges in

designing an efficient parallel algorithm, the major factor which limits the performance is scalability. Most frequently, a researcher needs to have community detection performed for a dataset of interest as fast as possible subject to the limitations of available hardware platforms. In other words, for any given instance of a community detection problem, the total size of the problem is fixed while the number of processors varies to minimize the solution time. This setting is an example of a strong scaling computing. Since the problem size per processor varies with the number of processors, the amount of work per processor goes down as the number of processors is increased. At the same time, the communication and synchronization overhead does not necessarily decrease and can actually increase with the number of processors, thus limiting the scalability of the entire solution.

There is yet another facet of scaling community detection solutions. As more and more hardware computing power becomes available, it seems quite natural to try to uncover the community structure of increasingly larger datasets. Since more compute power currently tends to come rather in a form of increased processor count than in a single high performance processor (or a small number of such processors), it is crucial to provide enough data for each single processor to perform efficiently. In other words, the amount of work per processor should be large enough so that communication and synchronization overhead is small relative to the amount of computation. Moreover, a well-designed parallel solution should demonstrate performance which at least does not degrade and perhaps even improves when run on larger and larger datasets.

Accessing data that is shared between several processes in a parallel community detection algorithm can easily become a bottleneck. Several techniques have been studied, including shared-nothing, master-slave, and data replication approaches, each having its merits and drawbacks. Shared-memory architectures make it possible to build solutions that require no data replication at all since any data can be accessed by any processor. One of the key design features of our multithreaded approach is to minimize the amount of synchronization and achieve a high degree of concurrency of code running on different processors and cores. Provided that the data is properly partitioned, the parallel algorithm that we propose does not suffer performance penalties when presented with increasing amounts of data. Quite the contrary, results show that, with larger datasets, the values of speedup avoid saturation and continue to improve up to maximal processor counts.

Validating the results of community detection algorithms presents yet another challenging task. After running a community detection algorithm how do we know if the resulting community structure makes any sense? If a network is known to have some ground truth communities then the problem is conceptually clear—we need to compare the output of the algorithm with the ground truth. It might sound like an easy problem to solve but in reality there are many possible ways to compare different community structures of the same network. Unfortunately, there is no one single method that can be used in any situation. Rather a combination of metrics can tell us how far our solution is from that represented by the ground truth. As mentioned earlier, for many real-life datasets it is not feasible to come up with any kind of ground truth communities at all. Without them, comparative study of values obtained from different metrics for community structures output by different algorithms seems to be the only way of judging the quality of community detection.

The rest of the paper is organized as follows. An overview of relevant research on parallel community detection is presented in Section 2. Section 3 provides an overview of the sequential SLPA algorithm upon which we base our parallel implementation. It also discusses details of the multithreaded community detection on a shared-memory multiprocessor machine along with busy-waiting techniques and implicit synchronization used to ensure correct execution. We describe the way we partition the data and rearrange nodes within a partition to maximize performance. We also discuss the speedup and efficiency accomplished by our approach. Detailed analysis of the quality of community structures detected by our algorithm for four real-life datasets relative to ground truth communities (when available) and based on the sequential SLPA implementation is given in Section 4. Finally, in Section 5, closing remarks and conclusions are provided. We also discuss some of the limitations of the presented solution and briefly describe future work.

2. Related Work

During the last decade substantial effort has been put into studying network clustering and analysis of social and other networks. Different approaches have been considered and a number of algorithms for community detection have been proposed. As online social communities and the networks associated with them continue to grow, the parallel approaches to community detection are regarded as a way to increase efficiency of community detection and therefore receive a lot of attention.

The clique percolation technique [4] considers cliques in a graph and performs community detection by finding adjacent cliques. The k-means clustering algorithm partitions m n-dimensional real vectors into k n-dimensional clusters where every point is assigned to a cluster in such a way that the objective function is minimized [5]. The objective function is the within-cluster sum of squares of distances between each point and the cluster center. There are several ways to calculate initial cluster centers. A quick and simple way to initialize cluster centers is to take the first k points as the initial centers. Subsequently at every pass of the algorithm the cluster centers are updated to be the means of points assigned to them. The algorithm does not aim to minimize the objective function for all possible partitions but produces a local optimal solution instead, that is, a solution in which for any cluster the within-cluster sum of squares of distances between each point and the cluster center cannot be improved by moving a single point from one cluster to another. Another approach described in [6] utilizes an iterative scan technique in which density function value is gradually improved by adding or removing edges. The algorithm implements a shared-nothing architectural

approach. The approach distributes data on all the computers in a setup and uses master-slave architecture for clustering. In such an approach, the master may easily become a bottleneck when the application requires a large number of processors because of the network size. A parallel clustering algorithm is suggested in [7], which is a parallelized version of DBSCAN [8].

A community detection approach based on propinquity dynamics is described in [9]. It does not use any explicit objective function but rather performs community detection based on heuristics. It relies on calculating the values of topology-based propinquity which is defined as a measure of probability that two nodes belong to the same community. The algorithm works by consecutively increasing the network contrast in each iteration by adding and removing edges in such a way as to make the community structure more apparent. Specifically, an edge is added to the network if it is not already present and the propinquity value of the endpoints of this proposed edge is above a certain threshold, called the *emerging threshold*. Similarly, if the propinquity value of the endpoints of an existing edge is below a certain value, called the *cutting threshold*, then this edge is removed from the network. Since inserting and removing edges alters the network topology, it affects not only propinquity between individual nodes but also the overall propinquity of the entire topology. The propinquity of the new topology can then be calculated and used to guide the subsequent changes to the topology in the next iteration. Thus, the whole process called *propinquity dynamics* continues until the difference between topologies obtained in successive iterations becomes small relative to the whole network.

Since both topology and propinquity experience only relatively small changes from iteration to iteration, it is possible to perform the propinquity dynamics incrementally rather than recalculating all propinquity values in each iteration. Optimizations of performing incremental propinquity updates achieve a running time complexity of $O((|V| + |E|) \cdot |E|/|V|)$ for general networks and $O(|V|)$ for sparse networks.

It is also shown in [9] that community detection with propinquity dynamics can efficiently take advantage of parallel computation using message passing. Nodes are distributed among the processors which process them in parallel. Since it is essential that all nodes are in sync with each other, the bulk synchronous parallel (BSP) model is used to implement the parallel framework. In this model, the computation is organized as a series of *supersteps*. Each superstep consists of three major actions: receiving messages sent by other processors during the previous superstep, performing computation, and sending messages to other processors. Synchronization in BSP is explicit and takes the form of a barrier which gathers all the processors at the end of the superstep before continuing with the next superstep. Two types of messages are defined for the processors to communicate with each other. The first type is used to update propinquity maps that each processor stores locally for its nodes. Messages of the second type contain parts of the neighbor sets that a processor needs in its local computation.

A number of researchers explored a popular MapReduce parallel programming model to perform network mining operations. For example, a PeGaSus library (Peta-Scale Graph Mining System) described in [10] is built upon using Hadoop platform to perform several graph mining tasks such as PageRank calculations, spectral clustering, diameter estimation, and determining connected components. The core of PeGaSus is a GIM-V function (generalized iterated matrix-vector multiplication). GIM-V is capable of performing three operations: combining two values, combining all the values in the set, and replacing the old value with a new one. Since GIM-V is general, it is also quite universal. All other functions in the library are implemented as function calls to GIM-V with proper custom definitions of the three GIM-V operations. Fast algorithms for GIM-V utilize a number of optimizations like using data compression, dividing elements into blocks of fixed size, and clustering the edges. Finding connected components with GIM-V is essentially equivalent to community detection. The number of iterations required to find connected components is at most the diameter of the network. One iteration of GIM-V has the time complexity of $O(((|V| + |E|)/P) \log((|V| + |E|)/P))$ where P is the number of processors in the cluster. Running PeGaSus on an M45 Hadoop supercomputer cluster shows that GIM-V scales linearly with the number of machines increasing from 3 to 90. Accordingly, PeGaSus is able to reduce time execution on real-world networks containing up to hundreds of billions of edges from many hours to a few minutes.

A HEigen algorithm introduced in [11] is an eigensolver for large scale networks containing billions of edges. It is built upon the same MapReduce parallel programming model as PeGaSus and is capable of computing k eigenvalues for sparse symmetric matrices. Similar to PeGaSus, HEigen scales almost linearly with the number of edges and processors and performs well in up to a billion edge scale networks. Its asymptotic running time is the same as that of PeGaSus' GIM-V.

In [12] the authors consider a disjoint partitioning of a network into connected communities. They propose a massively parallel implementation of an agglomerative community detection algorithm that supports both maximizing modularity and minimizing conductance. The performance is evaluated on two different threaded hardware architectures: a multiprocessor multicore Intel-based server and massively multithreaded Cray XMT2. This approach is shown to scale well on two real-world networks with up to tens of millions of nodes and several hundred million edges.

Another method for partitioning a network into disjoint communities is scalable community detection (SCD) [13]. This two-phase algorithm works by optimizing the value of an objective function and is capable of processing undirected and unweighted graphs.

SCD uses the weighted community clustering (WCC) metric proposed in [14] as the objective function. Instead of performing simple edge counting, WCC works with more sophisticated graph structures, such as triangles. The quality of a partition is measured based on the number of triangles in a graph. Intuitively, more connections between the nodes within a community correspond to a larger number of triangles. Communities tend to have many highly connected nodes which are much more likely to close triangles with

each other rather than with nodes from other communities. Thus, for a particular node and community the value of WCC quantifies the level of cohesion of the node to the community. The metric is defined not only for individual nodes and communities but also for a community as a whole and the entire partition. One of the advantages of WCC over modularity is that it does not have a resolution limit problem. In addition, optimization of WCC is mathematically guaranteed to produce cohesive and structured communities. The measure of cohesion is defined for a partition as some real-valued function f called *degree of cohesion*. For each subset of nodes f assigns a value in the range $[0, 1]$ such that high values of f correspond to good communities and low values of f correspond to bad communities. For a given context (social, biological, etc.) an adequate metric f captures the features specific to this context. For example, for social networks the cohesion of a community depends on the number of triangles closed by the nodes inside this community. Furthermore, triangles are also used as a good indicator of community structure.

The operation of SCD consists of two phases which are executed sequentially. The first stage comprises graph cleanup and initial partitioning. Cleanup is performed by removing the edges which do not close any triangles. Then the graph is partitioned based on the values of the clustering coefficient of every node. Nodes are taken in the order of decreasing clustering coefficient and placed in communities together with their neighbors. Such partitioning yields communities with high values of WCC which is beneficial for the subsequent optimization process.

The second phase is responsible for the refinement of the initial partition. WCC optimization process consists of iterations which are repeated as long as the value of WCC for the entire partition keeps improving. In order to improve the value of WCC, the best of the following three movements is chosen for each node. These are the only movements which can potentially improve the WCC score:

(i) keep the network unchanged;

(ii) remove a node from its current community and place it in its own singleton community;

(iii) move a node from one community to another.

After movements for all the nodes have been selected, the WCC metric is calculated for the entire partition and compared to the previous value to determine if there is an overall improvement in the score. The refinement process stops when there has been no improvement (or improvement was less than a specified threshold) during the most recent iteration.

Computing the value of WCC directly at each iteration and for each node is computationally expensive and therefore should be avoided, especially for high degree nodes. In order to speed up calculations, it is possible to exploit the fact that the refinement process operates using the improvement of the score and therefore computing the absolute value of WCC is not necessary. Instead of calculating WCC directly, SCD uses certain graph statistics to build a WCC estimator. The estimator evaluates the improvement of WCC only once per iteration spending just $O(1)$ time per node.

Assuming that graphs have a quasilinear relation between the number of nodes and the number of edges, and the number of iterations of the refinement process is a constant, the overall running time complexity of SCD is $O(m \cdot \log n)$, where n is the number of nodes and m is the number of edges in the graph.

The advantage of the SCD algorithm is its amenability to parallelization. This is due to the fact that during the optimization process improvements of WCC are considered for every node individually and independently of other nodes. Therefore, the best movement can be calculated for all nodes simultaneously using whatever parallel features the underlying computing platform has to offer. Moreover, applying the moves to all nodes is also done in parallel.

SCD is implemented in C++ as a multithreaded application which uses OpenMP API for parallelization. Concurrency during the refinement process is achieved by considering improvements of WCC and then applying movements independently for each node. Benchmark datasets used in experiments include a range of networks of different sizes: Amazon, DBLP, Youtube, LiveJournal, Orkut, and Friendster. All of these graphs contain ground truth communities which are required to evaluate the quality of communities produced by SCD.

Normalized mutual information (NMI) and average F_1 score are used to evaluate the quality of community detection. SCD is compared against the following algorithms: Infomap, Louvain, Walktrap, BigClam, and Oslom. No distinction is made between methods which perform only disjoint community detection and those that are capable of detecting overlapping communities. The output of each algorithm is compared against ground truth communities without regard to possible overlaps. Although the values of NMI and average F_1 scores obtained for SCD are close to the results of other algorithms, it outperforms its competition on almost all datasets.

In terms of runtime performance, SCD is much faster than the majority of other algorithms used in the experiment. In a single threaded mode, the largest of the datasets used (Friendster) was processed in about 12 hours. SCD scales almost linearly with the number of edges in the graph. Using multiple threads can reduce the processing time even further. With 4 threads it takes a little bit over 4 hours to perform community detection on the Friendster network. Although the values of speedup are not explicitly presented, it can be inferred that the advantage of using multiple threads varies considerably depending on the dataset. The best case seems to be the Orkut graph for which speedup grows linearly as the number of threads is increased from 1 to 4. However, since the scope of parallelization in the experiment is modestly limited to just 4 threads, it is unclear how the scalability of the multithreaded SCD behaves when more than 4 cores are utilized.

A family of label propagation community detection algorithms includes label propagation algorithm (LPA) [15], community overlap propagation algorithm (COPRA) [16], and speaker-listener label propagation algorithm (SLPA) [17].

The main idea is to assign identifiers to nodes and then make them transmit their identifiers to their neighbors. With node identifiers treated as labels, a label propagation algorithm simulates the exchange of labels between connected nodes in the network. At each step of the algorithm each and every node that has at least one neighbor receives a label from one of its neighbors. Nodes keep a history of labels that they have ever received organized as a histogram which captures the frequency (and therefore the rank) of each label. The number of steps, or iterations, of the algorithm determines the number of labels each node accumulates during the label propagation phase. Being one of the parameters of the algorithm, the number of iterations eventually affects the accuracy of community detection. Clearly, the running time of the label propagation phase is linear with respect to the number of iterations. The algorithm is guaranteed to terminate after a prescribed number of iterations. When it does, communities data is extracted from nodes' histories.

Staudt and Meyerhenke [18] proposed PLP, PLM, and EPP algorithms for nonoverlapping community detection, that is, determining a partitioning of the node set.

Parallel label propagation (PLP) algorithm is a variation of the sequential LPA capable of performing detection of nonoverlapping communities in undirected weighted graphs. PLP differs from the original formulation of the label propagation algorithm [15] in that it avoids explicitly randomizing the node order and relies instead on asynchronism of concurrently executed PLP code threads. This way it saves the cost of explicit randomization. In order to optimize code execution even further, nodes are divided into active nodes and inactive nodes. Since labels of inactive nodes cannot be updated in the current iteration, the label propagation process is only performed on active nodes, thus reducing the amount of computation.

The termination criterion used by PLP is also different from the original description [15]. PLP uses a threshold value to stop processing. The value of the threshold is determined empirically and set to $n \cdot 10^{-5}$, where n is the number of nodes in the graph. Therefore, for the majority of graphs which were included in the experiment, the number of iterations is relatively small (from 2 to about a hundred). Moreover, no justification is provided for this formula which establishes a relation between the number of iterations and the number of nodes in the graph. Although it is claimed that "clustering quality is not significantly degraded by simply omitting these iterations," it is also admitted that "while the PLP algorithm is extremely fast, its quality might not always be satisfactory for some applications." No results are presented to show how the number of iterations affects the quality of community detection or how the modularity scores of PLP compare to those of the competition.

A locally greedy, agglomerative (bottom-up) multilevel community detection method called parallel Louvain method (PLM) is based on modularity optimization. Starting from some initial partition, nodes are moved from one community to another as long as it increases the objective function, that is, modularity. When modularity reaches a local optimum, a graph is coarsened and modularity optimization process is repeated.

Ensemble preprocessing (EPP) algorithm is a combination of several community detection methods. Its main goal is to form a classifier which decides if a pair of nodes should belong to the same community. EPP requires a preprocessing step which is performed by several parallel PLP instances running concurrently. The consensus of several base classifiers is used to form core communities which are coarsened to reduce the problem size.

Ensemble multilevel (EML) method is a recursive extension of the ensemble preprocessing algorithm. First, the core clustering is produced. Then the graph is contracted to a smaller graph, and the same algorithm is called recursively until a predefined termination condition is met.

All algorithms in [18] are created in C++. Parallel code is implemented using OpenMP API. Nodes are distributed between the threads and processed concurrently. Performance of the algorithms is compared to several other community detection methods: CLU_TBB, RG, CGGC, CGGCi, and the original sequential Louvain implementation. In order to compare the quality of results produced by different algorithms, Staudt and Meyerhenke use modularity [19]. Although modularity is very popular it was shown to suffer from the resolution limit and is also known to have other issues and limitations. There are other community quality metrics as well as modified versions of the original definition of modularity which overcome some of these problems [20]. However, modularity is the only measure used to compare the quality of communities produced by different algorithms in this experiment.

A shared-memory multiprocessor machine was used to test the performance and community quality of different algorithms. EML performed poorly while PLP and PLM were found to pay off with respect to either the execution time or community detection quality.

PLP was the fastest algorithm tested. It demonstrated linear strong scaling in the range 2–16 threads for uk-2002, the largest network which participated in all experiments. No data on scaling results for other datasets were provided. Since only one graph describes speedup for PLP, it is difficult to measure the values exactly, but they are approximately 0.92 for 2 threads (i.e., slower than with a single thread), 1.45 for 4 threads, 2.6 for 8 threads, and 4.6 for 16 threads. The running time drops in a slightly sublinear manner with the number of threads, although the absolute values of speedup are quite modest, and efficiency slowly goes down from 35% for 4 threads to 29% for 16 threads.

In almost all the cases, EPP was able to improve the values of modularity achieved by PLM. However, this advantage comes at the cost of running on average 10 times slower. At the same time, scalability of EPP remains unclear since no data is provided on the running time performance for different ensemble sizes.

For uk-2007-05 which was the largest graph used in the experiments, only the processing time of 120 seconds using the PLP algorithm and a parallel configuration with 32 threads is reported. No information is provided about scalability tests with this graph for other numbers of threads. In addition, due to memory constraints a different hardware platform with larger memory and a different CPU had to be

used to process this network. Therefore, the results are not directly comparable to those of other datasets. Although it is also mentioned that "a modularity of 0.99598 is reached for the uk-2007-05 graph in 660 seconds," it is not clear under which conditions this result was achieved. There is no mention of any other results concerning uk-2007-05, including any comparisons with other algorithms. Despite mentioning that uk-2007-05 requires "more than 250 GB of memory in the course of an EPP run," no EPP results for this graph are reported either.

In [21] we designed a multithreaded parallel community detection algorithm based on the sequential version of SLPA. Although only unweighted and undirected networks have been used to study the performance of our parallel SLPA implementation, an extension for the case of weighted and directed edges is straightforward and does not affect the computational complexity of the method. To facilitate such generalization, each undirected edge is represented with two directed edges connecting two nodes in opposite directions. In effect, the number of edges that are represented internally in code, is doubled, but the code is capable of running on directed graphs. A distinctive feature of our parallel solution is that, unlike other approaches described above, it is capable of performing overlapping community detection and has a parameter enabling balancing the running time and community detection quality.

In this paper, we further explore the multithreaded parallel programming paradigm that was used in [21] and test its performance on several real-world networks that range in size from several hundred thousand nodes and a few million edges to almost 5.5 million nodes and close to 170 million edges. We also provide a detailed analysis of the quality of communities detected with the parallel algorithm.

3. Parallel Linear Time Community Detection

The SLPA [17] is a sequential linear time algorithm for detecting overlapping communities. SLPA iterates over the list of nodes in the network. Each node i randomly picks one of its neighbors n_i and the neighbor then selects randomly a label l from its list of labels and sends it to the requesting node. Node i then updates its local list of labels with l. This process is repeated for all the nodes in the network. Once it is completed, the list of nodes is shuffled and the same processing repeats again for all nodes. After t iterations of shuffling and processing label propagation, every node in the network has a label list of length t, as every node receives one label in each iteration. After all iterations are completed, postprocessing is carried out on the list of labels and communities are extracted. We refer interested readers to the full paper [17] for more details on SLPA.

It is obvious that the sequence of iterations executed in SLPA algorithm makes the algorithm sequential and it is important for the list of labels updated in one iteration to be reflected in the subsequent iterations. Therefore, the nodes cannot be processed completely independently of each other. Each node is a neighbor of some other nodes; therefore, if the lists of labels of its neighbors are updated, it will receive a label randomly picked from the updated list of labels.

3.1. Multithreaded SLPA with Busy-Waiting and Implicit Synchronization. Our multithreaded implementation closely follows the algorithm described in [21] with minor improvements and bug fixes. In the multithreaded SLPA, we adopt a busy-waiting synchronization approach. Each thread performs label propagation on a subset of nodes assigned to this particular thread. This requires that the original network to be partitioned into subnetworks with one subnetwork to be assigned to each thread. Although partitioning can be done in several different ways depending on our objective, in this case the best partitioning will be that which makes every thread spend the same amount of time processing each node. Label propagation for any node consists of forming a list of labels by selecting a label from every neighbor of this node and then selecting a single label from this list to become a new label for this node. In other words, the ideal partitioning would guarantee that at every step of the label propagation phase each thread deals with a node that has exactly the same number of neighbors as nodes that are being processed by other threads. Thus, the ideal partitioning would divide the network in such a way that a sequence of nodes for every thread consists of nodes with the same number of neighbors across all the threads. Such partitioning is illustrated in Figure 1. $T_1, T_2, \ldots, T_p$ are p threads that execute SLPA concurrently. As indicated by the arrows, time flows from top to bottom. Each thread has its subset of nodes $n_{i1}, n_{i2}, \ldots, n_{ik}$ of size k where i is the thread number, and node neighbors are $m_1, m_2, \ldots, m_k$. A box corresponds to one iteration. There are t iterations in total. Dashed lines denote points of synchronization between the threads.

In practice, this ideal partitioning will lose its perfection due to variations in thread start-up times as well as due to uncertainty associated with thread scheduling. In other words, in order for this ideal scheme to work perfectly, hard synchronization of threads after processing every node is necessary. Such synchronization would be both detrimental to the performance and unnecessary in real-life applications.

Instead of trying to achieve an ideal partitioning we can employ a much simpler approach by giving all the threads the same number of neighbors that are examined in one iteration of the label propagation phase. It requires providing each thread with such a subset of nodes that the sum of all indegrees is equal to the sum of all indegrees of nodes assigned to every other thread. In this case, for every iteration of the label propagation phase every thread will examine the same overall number of neighbors for all nodes that are assigned to this particular thread. Therefore, every thread will be performing, roughly, the same amount of work per iteration. Moreover, synchronization then is only necessary after each iteration to make sure that no thread is ahead of any other thread by more than one iteration. Figure 2 illustrates such partitioning. As before, $T_1, T_2, \ldots, T_p$ are p threads that execute SLPA concurrently. As shown by the arrows, time

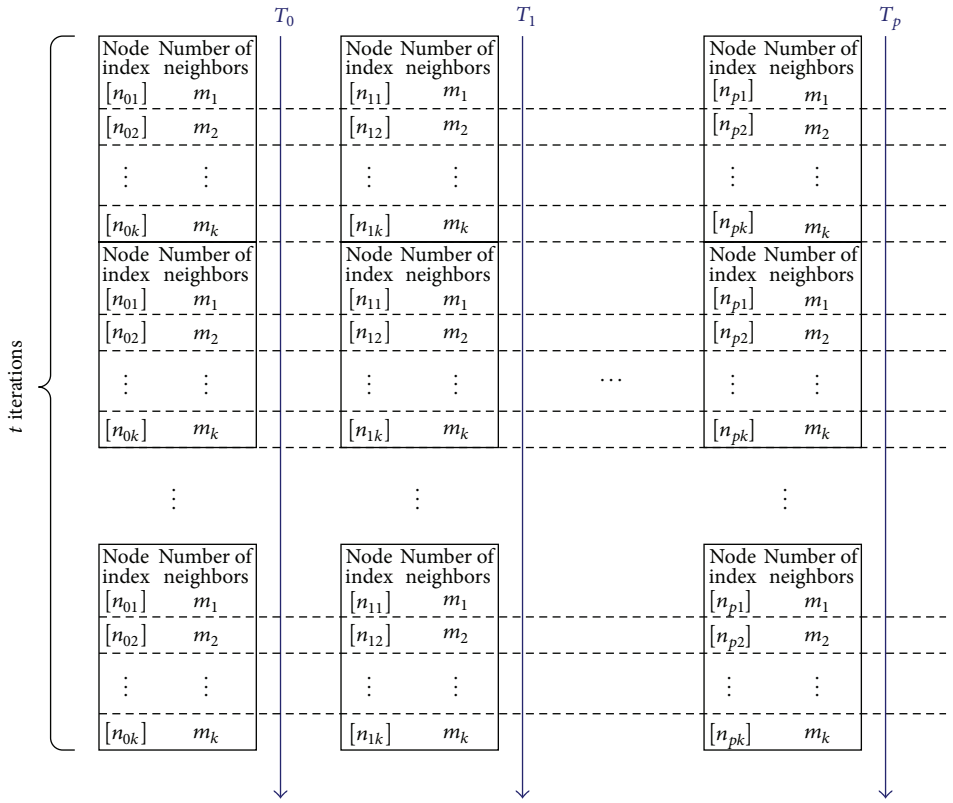


Figure 1: Ideal partitioning of the network for multithreaded SLPA.

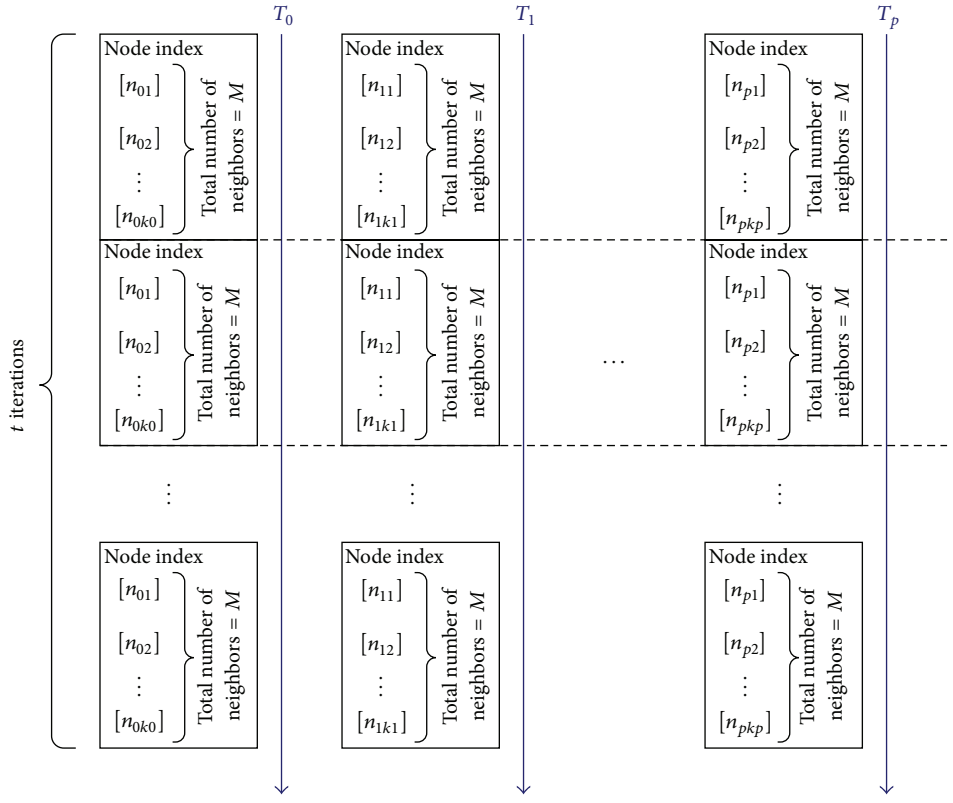


Figure 2: A better practical partitioning of the network for multithreaded SLPA.

flows from top to bottom. However, each thread now has its subset of nodes $n_{i1}, n_{i2}, \ldots, n_{ik_i}$ of size k_i where i is the thread number. In other words, threads are allowed to have different number of nodes that each of them processes, as long as the total number of node neighbors $M = \sum_{i=1}^{k_i} m_i$ is the same across all the threads. A box still corresponds to one iteration. There are t iterations in total. Dashed lines denote points of synchronization between the threads.

We can employ yet an even simpler approach of just splitting nodes equally between the threads in such a way that every thread gets the same (or nearly the same) number of nodes. It is important to understand that this approach is based on the premise that the network has small variation of local average of node degrees across all possible subsets of nodes of equal size. If this condition is met, then, as in the previous case, every thread performs approximately the same amount of work per iteration. Our experiments show that for many real-world networks this condition holds, and we accepted this simple partitioning scheme for our multithreaded SLPA implementation.

Given the choice of the partitioning methods described above, each of the threads running concurrently is processing all the nodes in its subset of nodes at every iteration of the algorithm. Before each iteration, the whole subset of nodes processed by a particular thread needs to be shuffled in order to make sure that the label propagation process is not biased by any particular order of processing nodes. Additionally, to guarantee the correctness of the algorithm, it is necessary to ensure that no thread is more than one iteration ahead of any other thread. The latter condition places certain restrictions on the way threads are synchronized. More specifically, if a particular thread is running faster than the others (for whatever reasons), it has to eventually pause to allow other threads to catch up (i.e., to arrive at a synchronization point no later than one iteration behind this thread). This synchronization constraint limits the degree of concurrency of this multithreaded solution.

It is important to understand the importance of partitioning the network nodes into subsets to be processed by the threads in respect to the distribution of edges across different network segments. In our implementation we use a very simple method of forming subsets of nodes for individual threads. First, a subset for the first thread is formed. Nodes are read sequentially from an input file. As soon as a new node is encountered, it is added to the subset of nodes processed by the first thread. After the subset of nodes for the first thread has been filled, a subset of nodes for the second thread is formed, and so on. Although simple and natural, this approach works well on networks with high locality of edges. For such networks, if the input file is sorted in the order of node numbers, nodes are more likely to have edges to other nodes that are assigned to the same thread. This leads to partitioning where only a small fraction (few percent) of nodes processed by each thread have neighbors processed by other threads.

Algorithm 1 shows the label propagation phase of our multithreaded SLPA algorithm which is executed by each thread. First, each thread receives a subset of nodes that it processes called *ThreadNodesPartition*. An array of dependencies *Used* is first initialized and then filled in such a way that it contains 1 for all threads that process at least one neighbor of the node from *ThreadNodesPartition* and 0 otherwise. This array of dependencies *Used* is then transformed to a more compact representation in the form of a dependency array D. Elements of array D contain thread numbers of all the threads which process any neighbor of a node that this thread processes. *Dsize* is the size of array D. If no node that belongs to the subset processed by this thread has neighbors processed by other threads, then array D is empty and $Dsize = 0$. If, for example, nodes that belong to the subset processed by this thread have neighbors processed by threads 1, 4, and 7, then array D has three elements with values of 1, 4, and 7, and $Dsize = 3$. After the dependency array has been filled, the execution flow enters the main label propagation loop which is controlled by counter t and has max T iterations. At the beginning of every iteration, we ensure that this thread is not ahead of the threads on which it depends by more than one iteration. If it turns out that it is ahead, this thread has to wait for the other threads to catch up. Then the thread performs a label propagation step for each of the nodes it processes which results in a new label being added to the list of labels for each of the nodes. Finally, the

```
ThreadPartition ← CreatePartition(InputFile)
p ← number of threads
for j = 1 to j < p do
  Used[j] ← 0
end for
for all v such that v is in ThreadNodesPartition do
  for all w such that w has an edge to v
    k ← getProcessorForNode(w)
    Used[k] ← 1
  end for
end for
Dsize ← 0
for j = 1 to j < p do
  if Used[j] > 0 then
    D[Dsize] ← j
    Dsize ← Dsize + 1
  end if
end for
while t < maxT do
  for j = 0 to j < Dsize − 1 do
    while t − t of thread D[j] > 1 do
      Do nothing
    end while
  end for
  for all v such that v is in myPartition do
    l ← selectLabel(v)
    Add label l to labels of v
  end for
  t ← t + 1
end while
```

Algorithm 1: Multithreaded SLPA.

iteration counter is incremented, and the next iteration of the loop is considered.

In order to even further alleviate the synchronization burden between the threads and minimize the sequentiality of the threads as much as possible, another optimization technique can be used. We note that some nodes which belong to a set processed by a particular thread have connection only to nodes that are processed by the same thread (we call them internal nodes), while other nodes have external dependencies. We say that a node has an external dependency when at least one of its neighbors belongs to a subset of nodes processed by some other thread. Since there are nodes with external dependencies, synchronization rules described above must be strictly followed in order to ensure correctness of the algorithm and meaningfulness of the communities it outputs. However, nodes with no external dependencies can be processed within a certain iteration independently from the nodes with external dependencies. It should be noted that a node with no external dependencies is not completely independent from the rest of the network since it may well have neighbors of neighbors that are processed by other threads.

It follows that processing of nodes with no external dependencies has to be done within the same iteration framework as for nodes with external dependencies but with less restrictive relations in respect to the nodes processed by other threads. In order to utilize the full potential of the technique described above, it is necessary to split the subset of nodes processed by a thread into two subsets, one of which contains only nodes with no external dependencies and the other one contains all the remaining nodes. Then, during the label propagation phase of the SLPA, nodes that have external dependencies are processed first in each iteration. Since we know that by the time such nodes are processed the remaining nodes (those with no external dependencies) cannot influence the labels propagated to nodes processed by other threads (due to the symmetry of the network), it is safe to increment the iteration counter for this thread, thus allowing other threads to continue their iterations if they have been waiting for this thread in order to be able to continue. Meanwhile, this thread can finish processing nodes with no external dependencies and complete the current iteration.

This approach effectively allows a thread to report completion of the iteration to the other threads sooner than it has been completed by relying on the fact that the work which remains to be completed cannot influence nodes processed by other threads. This approach, though seemingly simple and intuitive, leads to noticeable improvement of the efficiency of parallel execution (as described in Section 3.2) mainly due to decreasing the sequentiality of execution of multiple threads by signaling other threads earlier than in the absence of such splitting.

An important peculiarity arises when the number of nodes with external dependencies is only a small fraction (few percent) of all the nodes processed by the thread. In this case it would be beneficial to add some nodes without external dependencies to the nodes with external dependencies and process them together before incrementing the iteration counter. The motivation here is that nodes must be shuffled in each partition separately from each other to preserve the order of execution between partitions. Increasing partition size above the number of external nodes improves shuffling in the smaller of the two partitions.

```
Internal ← CreateInternalPartition(InputFile)
External ← CreateExternalPartition(InputFile)
p ← number of threads
/* Unchanged code from Algorithm 1 omitted */
while t < maxT do
  for j = 0 to j < Dsize − 1 do
    while t − t of thread D[j] > 1 do
      Do nothing
    end while
  end for
  for all v such that v is in External do
    l ← selectLabel(v)
    Add label l to labels of v
  end for
  t ← t + 1
  for all v such that v is in Internal do
    l ← selectLabel(v)
    Add label l to labels of v
  end for
end while
```

Algorithm 2: Multithreaded SLPA with splitting of nodes.

The remaining nodes without external dependencies can be processed after incrementing the iteration counter, as before. In order to reflect this optimization factor we introduce an additional parameter called the splitting ratio. A value of this parameter indicates the percentage of nodes processed by the thread before incrementing the iteration counter. For instance, if we say that splitting of 0.2 is used it means that at least 20% of nodes are processed before incrementing the iteration counter. If after initial splitting of nodes into two subsets of nodes with external dependencies and without external dependencies it turns out that there are too few nodes with external dependencies to satisfy the splitting ratio, some nodes that have no external dependencies are added to the group of nodes with external dependencies just to bring the splitting ratio to the desired value.

Algorithm 2 shows our multithreaded SLPA algorithm that implements splitting of nodes processed by a thread into a subset of nodes with external dependencies and a subset with no external dependencies. The major difference from Algorithm 1 is that, instead of processing all the nodes before incrementing the iteration counter, we first process a subset of nodes that includes nodes that have neighbors processed by other threads, then we increment the iteration counter, and then we process the rest of the nodes.

Since in [21] we studied the impact of selecting different values of the splitting ratio, it was not our main focus here. We simply accepted a splitting ratio of 0.2 and kept it fixed for all the test runs. Our major objective was to ensure that all parallel and sequential runs are performed with exactly the same code base and provide identical runtime

conditions and parameters, so that results of our performance evaluation and community detection quality metrics are directly comparable.

3.2. Performance Evaluation of the Multithreaded Solution. We performed runs on a hyper-threaded Linux system operating on top of a Silicon Mechanics Rackform nServ A422.v3 machine. Processing power was provided by 64 cores organized as four AMD Opteron 6272 central processing units (2.1 GHz, 16-core, G34, 16 MB L3 Cache) operating over a shared 512 GB bank of random access memory (RAM) (32 × 16 GB DDR3-1600 ECC Registered 2R DIMMs) running at 1600 MT/s Max. The source code was written in C++03 and compiled using g++ 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5).

Four datasets have been used to test the performance of the multithreaded solution and the quality of community detection. Three of these datasets (com-Amazon, com-DBLP, and com-LiveJournal) have been acquired from Stanford Large Network Dataset Collection (http://snap.stanford.edu/data/) which contains a selection of publicly available real-world networks (SNAP networks).

Undirected Amazon product copurchasing network (referred to as com-Amazon) was gathered, described, and analyzed in [22]. From the dataset information [23], it follows that it was collected by crawling Amazon website. A *Customers Who Bought This Item Also Bought* feature of the Amazon website was used to build the network. If it is known that some product i is frequently bought together with product j, then the network contains an undirected edge from i to j. For each product category defined by Amazon, there is a corresponding ground truth community. Each connected component in a product category is treated as a separate ground truth community.

Since small ground truth communities having less than 3 nodes had been removed, it was necessary to modify the original com-Amazon network to ensure that only nodes that belong to ground truth communities can appear in communities detected by the multithreaded parallel algorithm. Otherwise, comparison of communities produced by the community detection algorithm and the ground truth communities would not be feasible. The modified com-Amazon network was obtained from the original one by removing nodes which are not found in any ground truth community and all the edges connected to those nodes. While the original Amazon network consists of 334,863 nodes and 925,872 undirected edges, the modified dataset has 319,948 nodes and 1,760,430 directed edges. As outlined in Section 2, each undirected edge is internally converted to a pair of edges. Therefore, 925,872 undirected edges from the original network correspond to 1,851,744 directed edges in the internal representation of the code, and since some of the edges were incident to removed nodes, the resulting number of directed edges left in the network was 1,760,430.

The DBLP computer science bibliography network (referred to as com-DBLP) was also introduced and studied in [22]. According to the dataset information [24], it provides a comprehensive list of research papers in computer science. If two authors publish at least one paper together, then the nodes corresponding to these authors will be connected with an edge in a coauthorship network. Ground truth communities are based on authors who published in journals or conferences. All authors who have at least one publication in a particular journal or conference form a community. Similarly to the com-Amazon network, each connected component in a group is treated as a separate ground truth community. Small ground truth communities (less than 3 nodes) have also been removed.

The DBLP dataset was also modified to facilitate comparison with ground truth communities as described above for the com-Amazon network. Since DBLP is also undirected, the same considerations about the number of edges that were provided above for the com-Amazon network also apply to com-DBLP. The original DBLP network contains 317,080 nodes and 1,049,866 undirected edges, while the modified version has 260,998 nodes and 1,900,118 directed edges.

Another network from [22] that we are using to evaluate the performance of the multithreaded parallel implementation of SLPA and the quality of communities it produces is a LiveJournal dataset (referred to as com-LiveJournal). The dataset information page [25] describes LiveJournal as a free online blogging community where users declare friendship with each other. LiveJournal users can form groups and allow other members to join them. For the purposes of evaluating the quality of communities we are treating the com-LiveJournal network as having no ground truth communities. The LiveJournal network is undirected and contains 3,997,962 nodes and 34,681,189 pairs of directed edges. Since we are not comparing the communities found by the community detection algorithm with the ground truth communities, no modification of the original network is necessary.

The fourth dataset is a snapshot of the Foursquare network as of October 11, 2013. This dataset contains 5,499,157 nodes and 169,687,676 edges. No information about ground truth communities is available.

We calculated speedup using the formula shown in (1) and efficiency according to (2):

$$\text{Speedup} = \frac{T_1}{T_p}, \tag{1}$$

where Speedup is the actual speedup calculated according to (1) and p is the number of processors or computing cores:

$$\text{Efficiency} = \frac{\text{Speedup}}{p}. \tag{2}$$

All the experiments were run with 1,000 iterations (the value of $\max T$ was set to 1000) for all networks. On one hand, a value of 1,000 for the number of iterations provides a sufficient amount of work for the parallel portion of the algorithm, so that the overhead associated with creating and launching multiple threads does not dominate the label propagation running time. On the other hand, 1,000 iterations are empirically enough to produce meaningful communities since the number of labels in the history of every label is statistically significant. At the same time,

although running the algorithm for 1,000 iterations on certain datasets (especially larger ones) was in some cases (mainly for smaller core counts) taking a few days, it was still feasible to complete all runs on all four networks in under two weeks.

We conducted one set of measurements by considering only time for the label propagation phase since it is at this stage that our multithreaded implementation differs from the original sequential version. Time necessary to read an input file and construct in-memory representation of the nodes and edges as well as any auxiliary data structures was not included in this timing. All postprocessing steps and writing output files have also been excluded.

However, for an end user it is not the label propagation time (or any other single phase of the algorithm) that is important but rather the total running time. Users care about the time it took for the code to run: from the moment a command was issued until the resulting communities files have been written to a disk. Therefore, we conducted a second set of measurements to gather data on total execution time of our multithreaded parallel SLPA implementation. Since the total execution time includes not only a highly parallel label propagation stage but also file I/O, threads creation and cleanup, and other operations which are inherently sequential, it is to be expected that the values of both speedup and efficiency are going to be worse than in the case when only label propagation phase is considered.

Since the hardware platform we used provides 64 cores, every thread in our tests executes on its dedicated core. Therefore, threads do not compete for central processing unit (CPU) cores (unless there is interference from the operating system or other user processes running concurrently). They are executed in parallel, and we can completely ignore thread scheduling issues in our considerations. Because of this, we use terms "thread" and "core" interchangeably when we describe results of running the multithreaded SLPA. The number of cores in our runs varies from 1 to 64. However, we observed a performance degradation when the number of threads is greater than 32. This performance penalty is most likely caused by the memory banks organization of our machine. Speedup and efficiency are calculated using (1) and (2) defined earlier. No third-party libraries or frameworks have been used to set up and manage threads. Our implementation relies on Pthreads application programming interface (POSIX threads) which has implementations across a wide range of platforms and operating systems.

We noticed that the compiler version and compilation flags can each play a crucial role not only in terms of how efficiently the code runs but also in terms of the sole ability of code to execute in the multithreaded mode. Unfortunately, little, if anything is clearly and unambiguously stated in compiler documentation regarding implications of using various compiler flags to generate code for execution on multithreaded architectures. For the most part, developers have to rely on their own experience or common sense and experiment with different flags to determine the proper set of options which would make the compiler generate effective code capable of flawlessly executing multiple threads.

For instance, when the compiler runs with either -O2 or -O3 optimization flag to compile the multithreaded SLPA

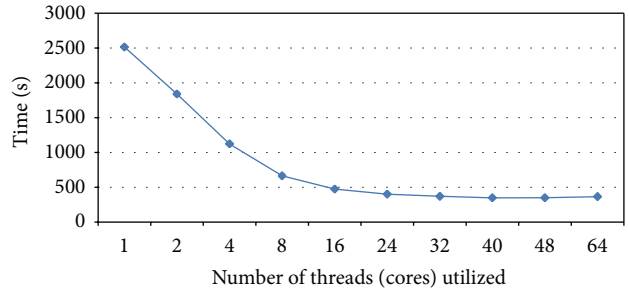


Figure 3: Label propagation time for com-Amazon network at different number of cores.

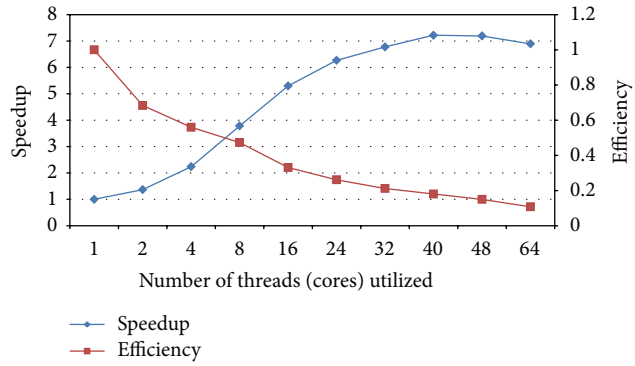


Figure 4: Speedup and efficiency for com-Amazon network (considering only label propagation time) at different number of cores.

the resulting binary code simply deadlocks at execution. The reason for deadlock is exactly the optimization that compiler performs ignoring the fact that the code is multithreaded. This optimization leads to threads being unable to see updates to the shared data structures performed by other threads. In our case such shared data structure is an array of iteration counters for all the threads. Evidently, not being able to see the updated values of other threads' counters quickly leads threads to a deadlock.

Another word of caution should be offered regarding some of the debugging and profiling compiler flags. More specifically, compiling code with -pg flag which generates extra code for a profiling tool *gprof* leads to substantial overhead when the code is executed in a multithreaded manner. The code seems to be executing fine but with a speedup of less than 1. In other words, the more threads are being used the longer it takes for the code to run regardless of the fact that each thread is executed on its own core and therefore does not compete with other threads for CPU. It is also counterintuitive since using more threads should result in a smaller subset of nodes that each thread processes.

The results of performance runs of our multithreaded parallel implementation are presented in Figures 3–19. (Data export was performed using Daniel's XL Toolbox add-in for Excel, version 6.51, developed by Daniel Kraus, Würzburg, Germany.)

Figures 3, 5, 7, and 9 show the time it took to complete the label propagation phase of the multithreaded parallel

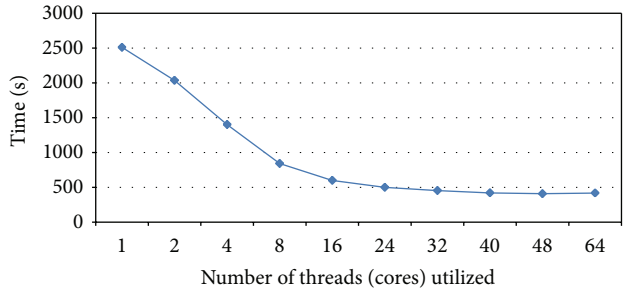


Figure 5: Label propagation time for com-DBLP network at different number of cores.

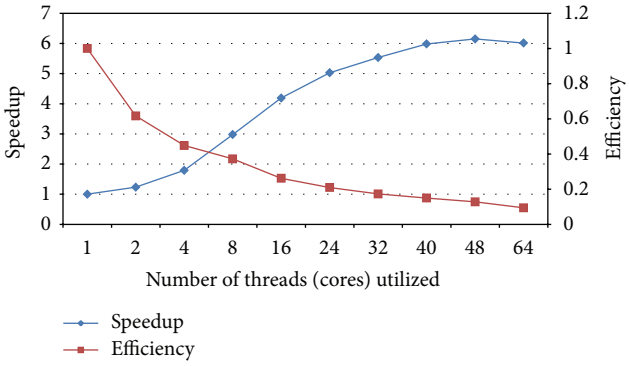


Figure 6: Speedup and efficiency for com-DBLP network (considering only label propagation time) at different number of cores.

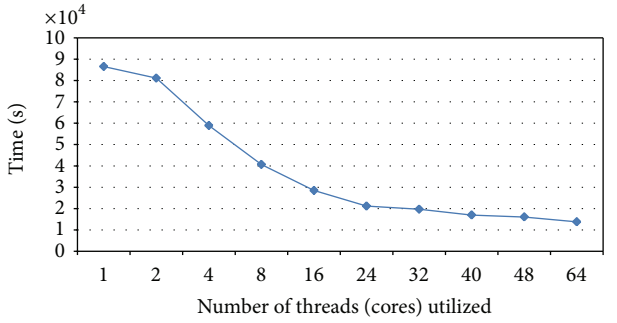


Figure 7: Label propagation time for com-LiveJournal network at different number of cores.

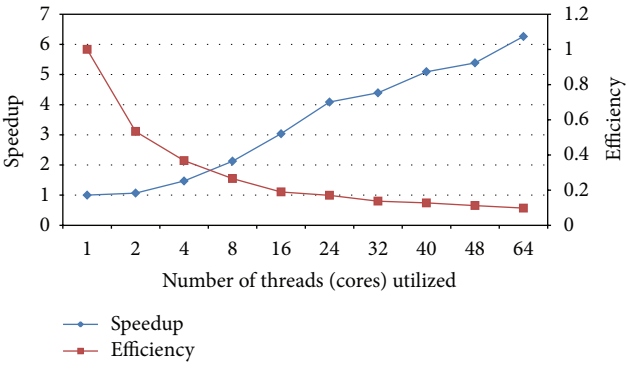


Figure 8: Speedup and efficiency for com-LiveJournal network (considering only label propagation time) at different number of cores.

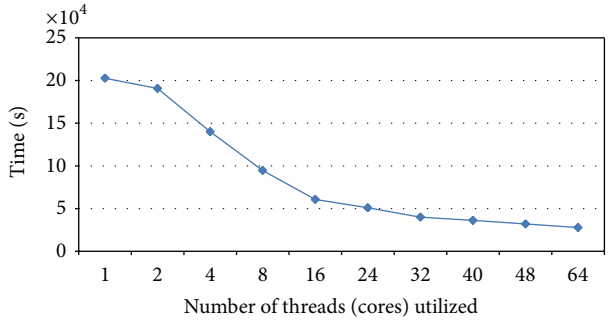


Figure 9: Label propagation time for Foursquare network at different number of cores.

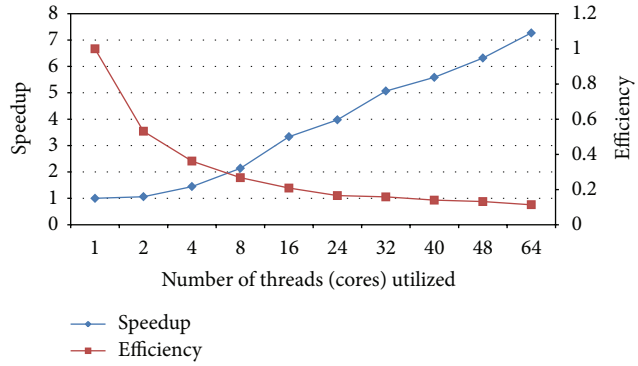


Figure 10: Speedup and efficiency for Foursquare network (considering only label propagation time) at different number of cores.

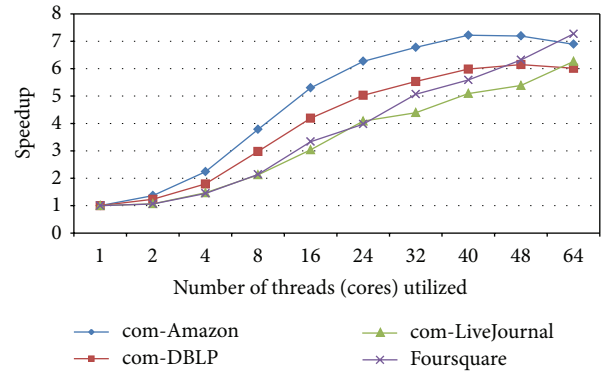


Figure 11: Speedup for all datasets (considering only label propagation time) at different number of cores.

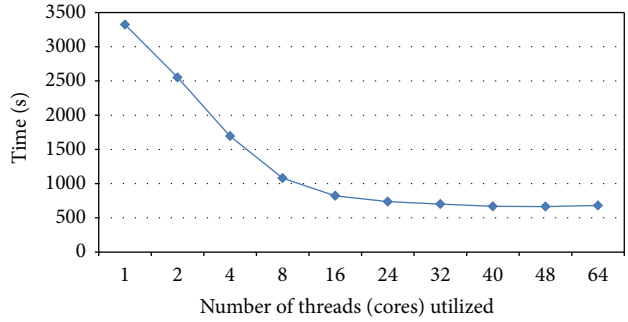


Figure 12: Total execution time for com-Amazon network at different number of cores.

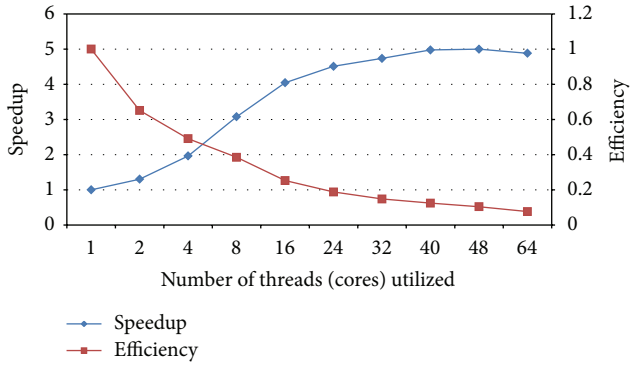


Figure 13: Speedup and efficiency for com-Amazon network (considering total execution time) at different number of cores.

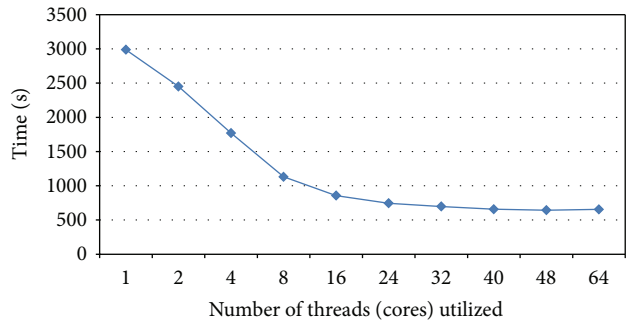


Figure 14: Total execution time for com-DBLP network at different number of cores.

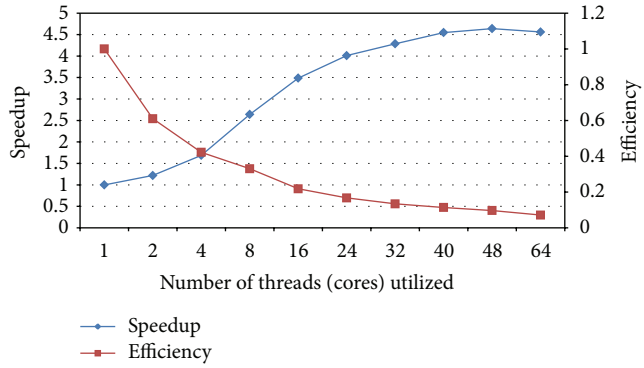


Figure 15: Speedup and efficiency for com-DBLP network (considering total execution time) at different number of cores.

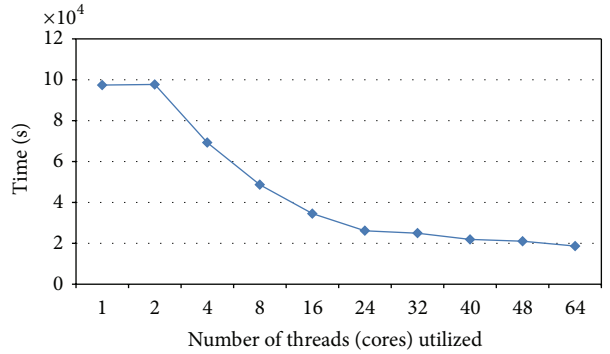


Figure 16: Total execution time for com-LiveJournal network at different number of cores.

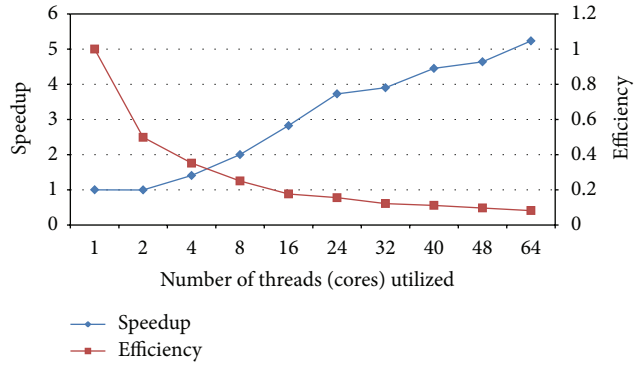


Figure 17: Speedup and efficiency for com-LiveJournal network (considering total execution time) at different number of cores.

SLPA on four datasets (com-Amazon, com-DBLP, com-LiveJournal, and Foursquare, resp.) for the number of cores varying from 1 to 64. It can be seen that for smaller core counts the time decreases nearly linearly with the number of threads. For larger number of cores the label propagation time continues to improve but at a much slower rate. In fact, for 32 or more cores, there is almost no improvement of the label propagation time on smaller datasets (com-Amazon and com-DBLP). At the same time, larger datasets (com-LiveJournal and Foursquare) improve label propagation times all the way through 64 cores. As outlined in Section 1, this is clearly something to be expected since in a strong scaling setting enough workload should be supplied to parallel processes to compensate for the overhead of creating multiple threads and maintaining communication between them.

This trend is even more evident in Figures 4, 6, 8, and 10 which plot the values of speedup and efficiency for the four datasets (com-Amazon, com-DBLP, com-LiveJournal, and Foursquare, resp.) and the number of cores from 1 to 64. As the number of cores increases, the speedup also grows but not as fast as the number of utilized cores, so efficiency drops. The saturation of speedup is quite evident for smaller networks (com-Amazon and com-DBLP) and corresponds to regions with no improvement of the label propagation time that we noticed earlier. Similarly, the values of speedup continue to improve (although at decreasing rates) for larger datasets (com-LiveJournal and Foursquare) even at 64 cores. Nonetheless, the efficiency degrades since speedup gains are small relative to an increase in core count. Such behavior can be attributed to several factors. First of all, as the number of cores grows while the network (and hence the number of nodes and edges) stays the same, each thread gets fewer nodes and edges to process. Approaching the limit of the thread size can cause the overhead of creating and running threads to outweigh the benefits of parallel execution for a sufficiently small thread size. Furthermore, as the number of cores grows, the number of neighbors of nodes with external dependencies

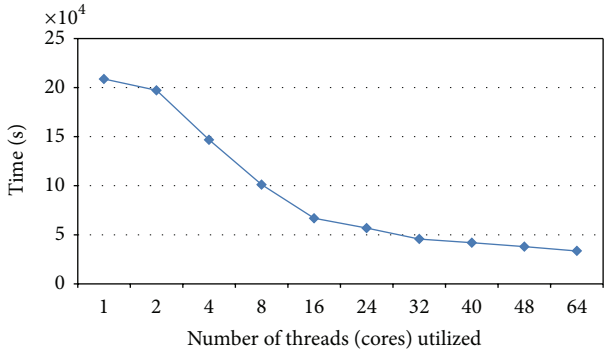


Figure 18: Total execution time for Foursquare network at different number of cores.

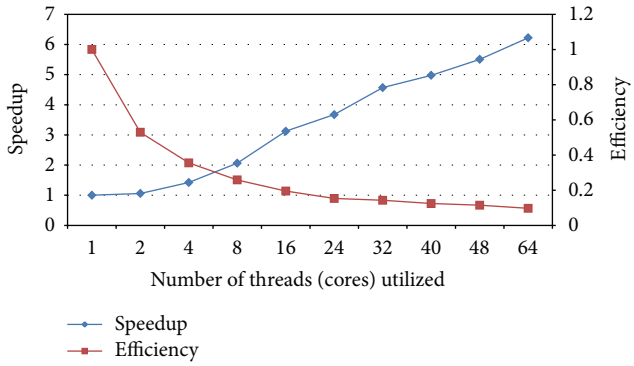


Figure 19: Speedup and efficiency for Foursquare network (considering total execution time) at different number of cores.

increases (both because each thread gets fewer nodes and there are more threads to execute them). More nodes with external dependencies, in turn, means that threads are more dependent on other threads.

However, for the sake of fair data interpretation, we need to remember that the definition of efficiency which we are using here is based on (2). It only takes into account the parallel execution speedup observed on a certain number of cores. The cost of cores is completely ignored in this formula. More realistically, the cost should be considered as well. The price paid for a modern computer system is not linear with the number of processors and cores. Within a certain range of the number of cores per system as the architecture moves towards higher processor and core counts, each additional core costs less. That is why the pure parallel efficiency defined by (2) should be effectively multiplied by the cost factor for making decisions regarding the choice of hardware to run community detection algorithms on real-life networks. After such multiplication, the efficiency including cost is going to be much more favorable to higher core counts than the efficiency given by (2).

Figure 11 combines plots of speedup values based on the label propagation time for all four datasets. Overall, the values of speedup do not vary considerably between the networks used in the experiments. However, it is quite evident that the shape of the curves is slightly different. On one hand, there is com-Amazon and com-DBLP for which the values of speedup reach local maximum at fewer than maximal number of cores. On the other hand, speedup values for com-LiveJournal and Foursquare are strictly increasing as the number of cores ranges between 1 and 64.

This observation is just additional evidence of the behavior discussed earlier. Smaller networks are too small to effectively use large core counts which leads to the saturation of speedup. The performance of multithreaded parallel SLPA on larger datasets continues to improve at almost a constant rate in a wide range of core counts between 4 and 64. It is also worth noting that, as long as a network is large enough to justify the overhead of multithreaded execution, different datasets yield almost identical speedup values. Although more testing would be required to firmly assert that speedup is independent of the size of the dataset, such behavior would be easy to explain. Indeed, speedup performance of the algorithm depends primarily on the properties of the graph (e.g., the number of edges crossing the boundary between the node sets processed by different cores) rather than on the size of the network. Such a feature is quite desirable in community detection since it enables the application to provide a user with an estimate of the overall execution time once the network is loaded and partitioned between the cores.

Figures 12, 14, 16, and 18 present the total community detection time of the multithreaded parallel SLPA on four datasets (com-Amazon, com-DBLP, com-LiveJournal, and Foursquare, resp.) for the number of cores varying from 1 to 64. Although clearly the total running time exceeds the label propagation phase, the difference in many cases is not that significant. This is especially true for larger datasets (com-LiveJournal and Foursquare) which, as we discussed above, is something to be expected. The fact that the label propagation phase is a dominating component of the total running time justifies our efforts to increase performance by replacing sequential label propagation with a parallel implementation.

The values of speedup and efficiency calculated based on the total execution time rather than label propagation time are plotted in Figures 13, 15, 17, and 19 for the four datasets (com-Amazon, com-DBLP, com-LiveJournal, and Foursquare, resp.) and the number of cores between 1 and 64. Although these values are worse than those calculated based only on the label propagation time, they provide a more realistic view of the end-to-end performance of our multithreaded SLPA implementation. In real life the speedup values of around 5 to 6 still constitute a substantial improvement over the sequential implementation, meaning, for example, that you would only have to spend 8 hours waiting for your community detection results instead of 2 days.

Figure 20 shows combined plots of speedup values for all four datasets considering the total execution time. Just like in Figure 11, the values of speedup for different networks are quite similar. The same two types of curves are observed which correspond to a group of relatively small (com-Amazon and com-DBLP) and large (com-LiveJournal and Foursquare) networks.

However, there are some differences. First, the absolute values of speedup are lower when we consider the total

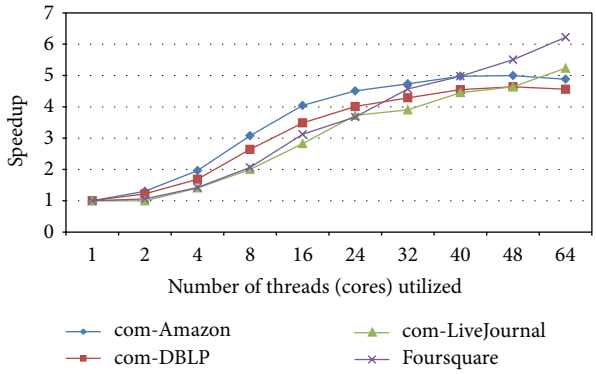


Figure 20: Speedup for all datasets (considering total execution time) at different number of cores.

execution time instead of just the label propagation phase. This is clearly something to be expected since the total execution time includes many operations (e.g., reading the input graph and writing output communities, partitioning the network between the cores, etc.) which cannot be made efficiently parallel. Second, the difference in speedup for different datasets even within the same group (e.g., large datasets) is greater than it was in Figure 11. The reason for that is also the effect of the limiting factor of sequential operations. Since we are considering the total execution time here, the size of the dataset affects speedup more significantly than in the case when only label propagation time was taken into account.

4. The Quality of Community Detection

In this section, we will evaluate the quality of the community structure detected with multithreaded version of SLPA [17] on the four datasets, Amazon, DBLP, Foursquare, and LiveJournal, introduced in Section 3.2. Amazon and DBLP have ground truth communities, while Foursquare and LiveJournal do not. Our only concern here is whether the community structure discovered by multithreaded SLPA has the quality similar to that detected by sequential SLPA [17] since we have already shown the effectiveness of sequential SLPA, compared with other community detection algorithms, in [17, 26]. Each metric value in Tables 1 and 2 is the average of results from ten runs of the community detection algorithm. The tested values of threshold r of SLPA are $r = 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4$, and 0.45.

We calculate *variation of information* (*VI*), *normalized mutual information* (*NMI*), *F-measure*, and *normalized Van Dongen metric* (*NVD*) [20] of the community structures detected by sequential SLPA and multithreaded SLPA on Amazon and DBLP, presented in Tables 1 and 2. Notice that *VI*, *NMI*, *F-measure*, and *NVD* are intended to measure the quality of disjoint communities. However, we could still use them here to evaluate the quality of overlapping communities, although the values of *NMI*, *F-measure*, and *NVD* may not be in the range of [0, 1]. There are mainly two reasons why we adopt their disjoint versions. On one hand,

Table 1: Metric values of the community structures detected by sequential SLPA and multithreaded SLPA on Amazon (bold font denotes the best value for each metric).

Algorithm	*VI*	*NMI*	*F*-measure	*NVD*
Sequential SLPA	**65.2664**	1.6113	**1.5318**	**−0.5647**
Multithreaded SLPA	65.4445	**1.6132**	1.5034	−0.5552

Table 2: Metric values of the community structures detected by sequential SLPA and multithreaded SLPA on DBLP (bold font denotes the best value for each metric).

Algorithm	*VI*	*NMI*	*F*-measure	*NVD*
Sequential SLPA	**30.4591**	0.963	**0.4112**	**0.4306**
Multithreaded SLPA	30.8962	**0.9675**	0.4029	0.4521

we are only concerned whether multithreaded SLPA has almost the same performance with sequential SLPA, in other words, whether communities detected by sequential and parallel runs have values of *VI*, *NMI*, *F-measure*, and *NVD* close to each other. On the other hand, *VI*, *F-measure*, and *NVD* do not have definitions for overlapping communities yet. *NMI* has its overlapping version [27], but it takes a very long time to calculate its value on large networks, like Amazon and DBLP. It can be seen from Tables 1 and 2 that the metric values of the community structures detected by sequential SLPA and multithreaded SLPA on Amazon and DBLP are very close to each other, which indicates that multithreaded SLPA has almost the same performance with sequential SLPA on Amazon and DBLP.

We then compute modularity (Q) [19], *intradensity*, *contraction*, *expansion*, and *conductance* [22, 28] of the community structures found by sequential SLPA and multithreaded SLPA on Foursquare and LiveJournal, presented in Tables 3 and 4. Notice that the modularity we adopt here is also applicable to disjoint communities, so its value may not be in the range of $[0, 1]$. The reasons for using the disjoint version of modularity echo those given in the case of *VI*, *NMI*, *F-measure*, and *NVD* metrics. In addition, there are several overlapping versions for modularity [29–34], and it is not clear which one is the best. Tables 3 and 4 show that the metric values of the community structures detected by sequential SLPA and multithreaded SLPA on Foursquare and LiveJournal are also very close to each other, which implies that multithreaded SLPA has almost the same performance with sequential SLPA on Foursquare and LiveJournal.

Comparisons between different community detection algorithms are not always easy to make due to substantially different implementations which might even require mutually exclusive architectural features or software components (shared-memory versus distributed memory machines, programming languages compiled to native code versus development systems based on virtual machines or interpretation, and so on).

Table 3: Metric values of the community structures detected by sequential SLPA and multithreaded SLPA on Foursquare (bold font denotes the best value for each metric).

Algorithm	Q	Intradensity	Contraction	Expansion	Conductance
Sequential SLPA	0.7608	**0.3651**	**3.6683**	**2.5137**	**0.3849**
Multithreaded SLPA	**0.7682**	0.3535	3.5766	2.6358	0.4055

Table 4: Metric values of the community structures detected by sequential SLPA and multithreaded SLPA on LiveJournal (bold font denotes the best value for each metric).

Algorithm	Q	Intradensity	Contraction	Expansion	Conductance
Sequential SLPA	0.6834	**0.3174**	**4.4735**	**2.4332**	**0.3777**
Multithreaded SLPA	**0.6929**	0.2969	4.0367	2.8901	0.4333

It is also important to consider the type of output communities that an algorithm can produce. As mentioned earlier, overlapping community detection is more computationally intensive than disjoint. While the majority of other parallel solutions perform only disjoint community detection, our multithreaded SLPA can produce either disjoint or overlapping communities, depending on the value of threshold r.

Even though execution time is certainly one of the most important performance measures for an end user, it is often not suitable for direct comparisons between different implementations of community detection methods. Unlike execution time which depends on specific hardware, operating systems, code execution environments, compiler optimizations, and other factors, speedup evens out architectural and algorithmic differences. It is therefore a much better way to compare runtime performance of community detection algorithms.

Another important factor that makes it hard to compare the results produced by competing methods is the use of different datasets. Although several datasets seem to appear more often than the others (e.g., Amazon, DBLP, and LFR) there is no established set of datasets which are publicly available and widely accepted as a benchmark for high performance community detection. If such a benchmark existed, it should have contained a balanced blend of both real-world and synthetic datasets of varying size (from hundreds of thousands of nodes and edges to billion scale networks) carefully selected so that it does not give a priori advantage to any of the possible approaches to community detection.

There are datasets which are supplied with so-called ground truth communities, although in some cases it is very questionable whether these communities in fact represent the ground truth. For other networks, it is not feasible to establish the ground truth at all. Again, there is no established consensus on whether datasets with or without ground truth communities (or a combination of both types) should be evaluated. Different researchers approach this problem differently, mainly depending on the datasets to which they have access. There is also a problem of using proprietary datasets which might not be available to other researchers to test their community detection implementations.

Besides using different datasets, researchers also use different metrics to evaluate the quality of community detection. A decade or so ago, modularity was the dominating player on the community quality field. However, after it was discovered that the original formulation of modularity suffers from several drawbacks, a number of new or extended metrics have been proposed and a number of old, almost forgotten methods have been rediscovered. A detailed review of different existing and emerging metrics can be found in [20]. Still, there is no agreement on which metric (or combination of metrics) should be chosen as an authoritative measure of the quality of community detection performed by a certain algorithm.

From all of the above, it follows that performing fair comparisons of different community detection implementations is difficult. To take just one example, let us consider PLP/EPP, SCD, and multithreaded SLPA.

(i) Both PLP/EPP and SCD methods (see Section 2) are only able to detect disjoint communities while multithreaded SLPA performs overlapping community detection.

(ii) Experiments with SCD were only conducted with the number of threads ranging from 1 to 4. In contrast, in our approach described in Section 3, we evaluate the method and show its scalability for all datasets being tested, including large graphs, and the number of cores ranging from 1 to 64. PLP was tested for a slightly wider range of parallel configurations (1 to 16 threads) but only for one dataset, uk-2002. For the Foursquare network which is similar in size to uk-2002, the values of speedup demonstrated by multithreaded SLPA (see Figure 19) are comparable to the results of SCD described in Section 2.

(iii) Modularity is the only measure of community quality considered by PLP/EPP. SCD uses *NMI* and average F_1 score. Multithreaded SLPA uses several different metrics, including *NMI* and *F-measure*. However, for the reasons explained above the values of *NMI* and *F-measure* may not be in the conventional range of $[0, 1]$. Therefore, it is not feasible to directly compare the values of community quality metrics obtained in our experiments with the SCD results.

In conclusion, the community structure found by multithreaded SLPA has almost the same quality as that discovered by sequential SLPA. Moreover, we have demonstrated in [17, 26] that sequential SLPA is very competitive compared to other community detection algorithms, which implies the effectiveness of multithreaded SLPA on community detection.

5. Conclusion and Future Work

In this paper, we evaluated the performance of a multithreaded parallel implementation of SLPA and showed that using modern multiprocessor and multicore architectures can significantly reduce the time required to analyze the structure of different networks and output communities. We found that despite the fact that the rate of speedup slows down as the number of processors is increased, it still pays off to utilize as many cores as the underlying hardware has available. Our multithreaded SLPA implementation was proven to be scalable in terms of both increasing the number of cores and analyzing increasingly larger networks. Furthermore, the properties of the detected communities closely match those produced by the base sequential algorithm, as verified using several metrics. Given a sufficient number of processors, the parallel SLPA can reliably process networks with millions of nodes and accurately detect meaningful communities in minutes and hours.

In our future work, we plan to explore other parallel programming paradigms and compare their performance with our multithreaded approach.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the US Government.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research was sponsored in part by the Army Research Laboratory under Cooperative Agreement no. W911NF-09-2-0053, by the EU's 7FP Grant Agreement no. 316097, and by the Polish National Science Centre, the Decision no. DEC-2013/09/B/ST6/02317.

References

[1] R. E. Park, *Human Communities: The City and Human Ecology*, vol. 2, Free Press, 1952.

[2] J. Xie, S. Sreenivasan, G. Korniss, W. Zhang, C. Lim, and B. K. Szymanski, "Social consensus through the influence of committed minorities," *Physical Review E*, vol. 84, no. 1, Article ID 011130, 8 pages, 2011.

[3] P. Sah, L. O. Singh, A. Clauset, and S. Bansal, "Exploring community structure in biological networks with random graphs," BioRxiv, 2013.

[4] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[5] J. A. Hartigan and M. A. Wong, "Algorithm as 136: a k-means clustering algorithm," *Journal of the Royal Statistical Society Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[6] J. Baumes, M. Goldberg, and M. Magdon-Ismail, "Efficient identification of overlapping communities," in *Intelligence and Security Informatics*, pp. 27–36, Springer, Berlin, Germany, 2005.

[7] X. Xu, J. Jäger, and H.-P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," *Data Mining and Knowledge Discovery*, vol. 3, no. 3, pp. 263–290, 1999.

[8] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, vol. 1996, pp. 226–231, AAAI Press, 1996.

[9] Y. Zhang, J. Wang, Y. Wang, and L. Zhou, "Parallel community detection on large networks with propinquity dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 997–1006, ACM, July 2009.

[10] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "Pegasus: a petascale graph mining system implementation and observations," in *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM '09)*, pp. 229–238, IEEE, 2009.

[11] U. Kang, B. Meeder, and C. Faloutsos, "Spectral analysis for billion-scale graphs: discoveries and implementation," in *Advances in Knowledge Discovery and Data Mining: 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part II*, vol. 6635 of *Lecture Notes in Computer Science*, pp. 13–25, Springer, Berlin, Germany, 2011.

[12] E. J. Riedy, H. Meyerhenke, D. Ediger, and D. A. Bader, "Parallel community detection for massive graphs," in *Parallel Processing and Applied Mathematics*, vol. 7203, pp. 286–296, Springer, Berlin, Germany, 2012.

[13] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*, pp. 225–236, World Wide Web Conferences Steering Committee, Seoul, Republic of Korea, April 2014.

[14] A. Prat-Pérez, D. Dominguez-Sal, J. M. Brunat, and J.-L. Larriba-Pey, "Shaping communities out of triangles," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*, pp. 1677–1681, ACM, November 2012.

[15] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 76, no. 3, Article ID 036106, 2007.

[16] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, Article ID 103018, 2010.

[17] J. Xie and B. K. Szymanski, "Towards linear time overlapping community detection in social networks," in *Advances in Knowledge Discovery and Data Mining*, pp. 25–36, Springer, Berlin, Germany, 2012.

[18] C. L. Staudt and H. Meyerhenke, "Engineering high-Performance community detection heuristics for massive graphs," in *Proceedings of the 42nd Annual International Conference on Parallel Processing*, pp. 180–189, Lyon, France, October 2013.

[19] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, Article ID 026113, 2004.

[20] M. Chen, K. Kuzmin, and B. K. Szymanski, "Community detection via maximization of modularity and its variants," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 46–65, 2014.

[21] K. Kuzmin, S. Y. Shah, and B. K. Szymanski, "Parallel overlapping community detection with SLPA," in *Proceedings of the International Conference on Social Computing (SocialCom '13)*, pp. 204–212, IEEE, September 2013.

[22] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, Beijing, China, 2012.

[23] J. Leskovec, *Amazon Product Co-Purchasing Network and Ground-Truth Communities*, 2014, http://snap.stanford.edu/data/com-Amazon.html.

[24] "DBLP collaboration network and groundtruth communities," 2014, http://snap.stanford.edu/data/com-DBLP.html.

[25] LiveJournal social network and groundtruth communities, 2014, http://snap.stanford.edu/data/com-LiveJournal.html.

[26] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: the state-of-the-art and comparative study," *ACM Computing Surveys*, vol. 45, no. 4, Article ID 2501657, 2013.

[27] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, Article ID 033015, 2009.

[28] M. Chen, T. Nguyen, and B. K. Szymanski, "A new metric for quality of network community structure," *ASE Human Journal*, vol. 2, no. 4, pp. 226–240, 2013.

[29] S. Zhang, R.-S. Wang, and X.-S. Zhang, "Identification of overlapping community structure in complex networks using fuzzy c-means clustering," *Physica A: Statistical Mechanics and Its Applications*, vol. 374, no. 1, pp. 483–490, 2007.

[30] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó, "Fuzzy communities and the concept of bridgeness in complex networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 77, no. 1, Article ID 016107, 2008.

[31] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 8, pp. 1706–1712, 2009.

[32] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri, "Extending the definition of modularity to directed graphs with overlapping communities," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2009, no. 3, Article ID P03024, 2009.

[33] H.-W. Shen, X.-Q. Cheng, and J.-F. Guo, "Quantifying and identifying the overlapping community structure in networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2009, no. 7, Article ID P07042, 2009.

[34] D. Chen, M. Shang, Z. Lv, and Y. Fu, "Detecting overlapping communities of weighted networks via a local algorithm," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 19, pp. 4177–4187, 2010.