Alan Frieze
Paul Horn
Paweł Prałat (Eds.)

LNCS 6732

Algorithms and Models for the Web Graph

8th International Workshop, WAW 2011
Atlanta, GA, USA, May 2011
Proceedings

Springer

Lecture Notes in Computer Science 6732

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Alan Frieze Paul Horn Paweł Prałat (Eds.)

Algorithms and Models for the Web Graph

8th International Workshop, WAW 2011
Atlanta, GA, USA, May 27-29, 2011
Proceedings

Springer

Volume Editors

Alan Frieze
Carnegie Mellon University, Department of Mathematical Sciences
Pittsburgh, PA 15213, USA
E-mail: alan@random.math.cmu.edu

Paul Horn
Emory University, Department of Mathematics and Computer Sciences
400 Dowman Drive, W401, Atlanta, GA 30322, USA
E-mail: phorn@mathcs.emory.edu

Paweł Prałat
West Virginia University, Department of Mathematics
Morgantown, WV 26506-6310, USA
E-mail: pralat@math.wvu.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-21285-7 e-ISBN 978-3-642-21286-4
DOI 10.1007/978-3-642-21286-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: Applied for

CR Subject Classification (1998): F.2, G.2, H.4, C.2, H.3, H.2.8, E.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The 8th Workshop on Algorithms and Models for the Web Graph (WAW 2011) took place at Emory University in Atlanta, GA, May 27–29, 2011. This is an annual meeting, which is traditionally co-located with another, related, conference. WAW 2011 was co-located with the 15th International Conference on Random Structures and Algorithms (RSA 2011). Co-location of the workshop and conference provided opportunities for researchers in two different but interrelated areas to interact and to exchange research ideas. It was an effective venue for the dissemination of new results and for fostering research collaboration.

The World Wide Web has become part of our everyday life, and information retrieval and data mining on the Web are now of enormous practical interest. The algorithms supporting these activities combine the view of the Web as a text repository and as a graph, induced in various ways by links among pages, hosts, and users. The aim of the workshop was to further the understanding of graphs that arise from the Web and various user activities on the Web, and stimulate the development of high-performance algorithms and applications that exploit these graphs. The workshop gathered the researchers who are working on graph-theoretic and algorithmic aspects of related complex networks, including citation networks, social networks, biological networks, molecular networks, and other networks arising from the Internet.

This volume contains the papers presented during the workshop. There were 19 submissions. Each submission was reviewed by four Program Committee members. Papers were submitted and reviewed using the EasyChair online system. The committee members decided to accept 10 papers.

May 2011

Alan Frieze
Paul Horn
Paweł Prałat

Organization

General Chairs

Andrei Z. Broder	Yahoo! Research, USA
Fan Chung Graham	University of California San Diego, USA

Organizing Committee

Alan Frieze	Carnegie Mellon University, USA
Paul Horn	Emory University, USA
Paweł Prałat	West Virginia University, USA

Sponsoring Institutions

Google
Internet Mathematics
Microsoft Research New England
National Science Foundation
Telefonica Research
West Virginia University
Yahoo! Research

Program Committee

Dimitris Achlioptas	UC Santa Cruz, USA
Luca de Alfaro	Google and University of California Santa Cruz, USA
Tanya Berger-Wolf	University of Illinois, USA
Anthony Bonato	Ryerson University, Canada
Christian Borgs	Microsoft Research, USA
Andrei Broder	Yahoo! Research, USA
Jennifer Tour Chayes	Microsoft Research, USA
Colin Cooper	King's College, UK
Anirban Dasgupta	Yahoo! Research, USA
Alessandro Flammini	Indiana University, USA
Abraham Flaxman	University of Washington, USA
Alan Frieze	Carnegie Mellon University, USA
Michael Goodrich	UC Irvine, USA
Fan Chung Graham	UC San Diego, USA
Adam Henry	West Virginia University, USA
Paul Horn	Emory University, USA

Jeannette Janssen	Dalhousie University, Canada
Ravi Kumar	Yahoo! Research, USA
Kevin Lang	Yahoo! Research, USA
Silvio Lattanzi	Google, USA
Stefano Leonardi	Sapienza, University of Rome, Italy
Jure Leskovec	Stanford University, USA
Nelly Litvak	University of Twente, The Netherlands
Lincoln Lu	University of South Carolina, USA
Milena Mihail	Georgia Tech, USA
Michael Mitzenmacher	Harvard University, USA
Muthu Muthukrishnan	Rutgers University, USA
Alessandro Panconesi	Sapienza, University of Rome, Italy
Paweł Prałat	West Virginia University, USA
Josep M. Pujol	Telefonica Research, Spain
D. Sivakumar	Yahoo! Research, USA
Joel Spencer	NYU, USA

Table of Contents

A Spectral Algorithm for Computing Social Balance

Evimaria Terzi[1] and Marco Winkler[2,⋆]

[1] Boston University, USA
evimaria@cs.bu.edu
[2] Julius Maximilians Universität Würzburg, Germany
marco.winkler@physik.uni-wuerzburg.de

Abstract. We consider social networks in which links are associated with a sign; a positive (negative) sign indicates friendship (animosity) between the connected nodes. Recent work studies such large online signed networks by applying theories that stem from the notion of *social balance*. Computing the social balance of a signed network requires counting the distinct configurations of the signed edges within all possible triangles that appear in the network. A naive algorithm for such counting would require time that is cubic to the total number of nodes; such an algorithm is infeasible for large signed networks that are generated from online applications.

In this paper, we present an efficient spectral algorithm that computes approximate counts of the signed-triangle configurations. The essence of the algorithm lies in associating the eigenvalues of the adjacency matrix of a signed network with its signed-triangle configurations. Our experiments demonstrate that our algorithm introduces only a small error in the computed quantities while, at the same time, it achieves significant computational speedups.

1 Introduction

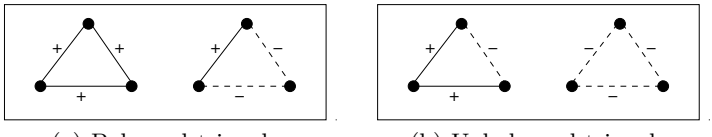
The interplay between positive and negative interactions between people defines the smooth functioning of society. In a similar way, in the online world, positive and negative relationships between users play a significant role in the function and evolution of online social networks. For example, users on Wikipedia can vote for or against the nomination of others as admins [4]. Users on Q&A systems (e.g., Yahoo! answers) get promoted by other users if they answer questions correctly; otherwise they get demoted. Users on Epinions can express trust or distrust of others [9,18]; participants on Slashdot can declare others to be either friends or foes [3,13,14]. Even the links between blogposts of different bloggers can be positive when the one blogger endorses the statements of the other or negative if the users express difference in opinions.

The focus of this paper is on computational problems that arise in *signed networks*. In such networks, every link is annotated with either a positive or

⋆ This was done while the author was visiting Boston University.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 1–13, 2011.

a negative sign. The positive links represent friendship while the negative ones represent antagonism. A basic tool in the study of signed networks is the notion of *social balance* also referred to as *structural balance* [11]. The signed triangles of a network are the very basic entities to be considered in social-balance studies. Up to node permutation, there are four possible configurations of a signed triangle. These configurations are shown in Fig. 1. The two triangles with an odd number of plus edges are *balanced*(Fig. 1(a)): they satisfy the adages that "the enemy of my enemy is my friend" and "the friend of my enemy is my enemy". The two triangles with an even number of plus edges (Fig. 1(b)) do not comply with this logic of friendship, and are considered *unbalanced*. For balanced triangles, the product of the edge signs is positive; for unbalanced triangles the same product has a negative sign. This definition of balanced and unbalanced triangles has its origins in social psychology studies [5,11] and it has been used as a basis for studying the dynamics of network formation [1] as well as the behavior of users in online social networks [15,16].



(a) Balanced triangles (b) Unbalanced triangles

Fig. 1. Balanced and unbalanced configurations of signed triangles

Given an undirected and signed graph $G = (V, E)$, a very natural question, also associated with social-balance studies, is to compute the fraction of triangles in G that are balanced. If we denote this quantity by $\widehat{b}$, then, the fraction of unbalanced triangles is $\widehat{u} = 1 - \widehat{b}$.

A naive algorithm for computing $\widehat{b}$ in a given graph counts all possible triangles in the graph and finds the ones that are balanced. For a graph with n nodes, such an algorithm goes over all the possible $\binom{n}{3}$ triangles of the graph and requires time $\mathcal{O}(n^3)$. This cubic running time makes this algorithm impractical for large datasets.

The contribution of this paper is to associate the quantity $\widehat{b}$ with the eigenvalues of the network's adjacency matrix. This connection allows us to develop a spectral algorithm that can efficiently approximate the fraction of balanced triangles. Our results highlight the relationship between the structural balance of a network and its underlying structure, as expressed in the eigenvalue decomposition of the matrix representation of the network. Although our methodology applies to arbitrary networks, we present our results for complete graphs first. Then, we show how they generalize to incomplete graphs. In incomplete graphs, apart from the existing edges that are all signed by a "+" or a "-" sign, there are also non-existing (i.e., *neutral*) edges. All signed networks coming from online social networks are incomplete; in such large datasets it is unrealistic to assume

that there is information about the positive or negative relationships between all pairs of nodes. Our experiments show that algorithm works extremely well in practice. As a by-product of our experimental evaluation we get a study of the eigenvalue spectrum of the adjacency matrices of signed networks.

The rest of the paper is organized as follows: in Section 2 we review work related to social-balance studies and spans scientific areas ranging from sociology and physics to computer science. We give some basic definitions and describe some computational tools in Section 3. In Section 4 we describe our methodology for the special case of complete graphs; we extend it for incomplete graphs in Section 5. Our experimental evaluation is given in Section 6 and we conclude the paper in Section 7.

2 Related Work

The principles underlying structural balance are based on theories in social psychology, and they date back to the work of Heider in the 1940s [11,12]. Cartwright and Harary [5,6,10] generalized these principles and expressed them in terms of graphs in the 1950s. All these studies focus on building psychological and sociological models for social balance and, naturally, they do not consider the computational aspects of these models.

More recent work focuses on mathematical models, that attempt to capture how the structural balance of a network can evolve from dynamic changes to the links' signs over time [1,17]. This line of work mostly focuses on questions related to the evolution of friendships and antagonisms between the nodes and does not consider the algorithmic problems related to structural balance.

Analysis of *online* signed social networks, through the lenses of structural-balance theories, has only appeared recently in the work of Leskovec et. al. [15,16]. More specifically, Leskovec et. al. have analyzed theories of balance and status in the context of social-media sites, investigating the extent to which each theory helped explain the linking behavior of users to these sites [16]. In a more recent work, the same authors developed methods for predicting the sign of the edge between users in signed online social networks [15]. Our work is complementary to this; we provide algorithmic tools for computing basic quantities associated with social balance. Such computations will allow the techniques developed in the past to become more efficient.

The idea of associating the triangles of a network with the eigenvalues of its adjacency matrix is not new: it has recently appeared in the work of Tsourakakis et. al. [19,20]. However, Tsourakakis et. al. focus on unsigned networks and on the computation of the total number of distinct (unsigned) triangles in them. On the other hand, we consider signed networks and signed-triangle configurations, and our goal is to count the proportion of balanced and unbalanced triangles in such networks. In that respect, our task is more complicated and the techniques developed by Trourakakis et. al. do not directly apply to our setting.

3 Basics

For the rest of the paper we assume a social network $G = (V, E)$ consisting of a set of n nodes, i.e., $|V| = n$. The edges of the network are *undirected* and *signed*. Each existing edge $\{i, j\} \in E$ is labeled with either a *plus* or a *minus* sign. A plus sign corresponds to feelings of friendship, while a minus sign indicates animosity between nodes i and j. Not all edges need to be present; a non-existing edge between two nodes corresponds to "neutral" feelings.

Given a graph $G = (V, E)$, we use $\mathbf{A}$ to represent the $n \times n$ *adjacency* matrix of G. If edge $\{i, j\}$ is signed with a "plus" ("minus"), then $\mathbf{A}_{ij} = 1$ ($\mathbf{A}_{ij} = -1$). If edge $\{i, j\}$ does not exist, then $\mathbf{A}_{ij} = 0$. Since the graph G is undirected, matrix $\mathbf{A}$ is *symmetric*. In addition to the adjacency matrix $\mathbf{A}$, we also define the *connectivity* matrix of G, which we denote by $\mathbf{G}$. Matrix $\mathbf{G}$ is also symmetric and $\mathbf{G}_{ij} = 1$ if nodes i and j are connected by an edge (irrespective of the sign); otherwise $\mathbf{G}_{ij} = 0$.

Following traditional linear-algebra notation, we use $\text{trace}(\mathbf{A})$ ($\text{trace}(\mathbf{G})$) to denote the sum of the diagonal elements of $\mathbf{A}$ ($\mathbf{G}$). That is, $\text{trace}(\mathbf{A}) = \sum_{i=1}^{n} \mathbf{A}_{ii}$ and $\text{trace}(\mathbf{G}) = \sum_{i=1}^{n} \mathbf{G}_{ii}$.

As shown in Fig. 1, there are four possible configurations of signed triangles (up to node permutation). In the theory of social balance, the triangle configurations with even number of minus signs are considered as *balanced* (Fig. 1(a)). On the other hand, the two configurations with odd number of minus signs correspond to *unbalanced* triangles (Fig. 1(b)). Notice that the product of the edge signs is positive for a balanced triangle and negative for an unbalanced one. If b and u denote the number of balanced and unbalanced triangles in G, we are interested in computing the following two quantities: (1) the *fraction of balanced triangles in* G, denoted by $\widehat{b}$ and (2) the *fraction of unbalanced triangles in* G, denoted by $\widehat{u}$. Since $\widehat{b} + \widehat{u} = 1$, computing $\widehat{b}$ suffices to achieve our goal.

Given a signed graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$ and connectivity matrix $\mathbf{G}$ we know the following:

Proposition 1 ([19,20]). *The number of distinct triangles that node i participates in is $\delta_i = \frac{1}{2}\mathbf{G}^3_{ii}$.*

Furthermore, we can associate the diagonal elements of $\mathbf{A}^3$ with the balanced and unbalanced triangles a single node participates in. The following proposition summarizes this relationship.

Proposition 2. *For a node i, let b_i be the number of balanced and u_i the number of unbalanced triangles i participates in. Then, $\frac{1}{2}\mathbf{A}^3_{ii} = (b_i - u_i)$.*

Proof. Consider the expansion of the diagonal elements of $\mathbf{A}^3$

$$\mathbf{A}^3_{ii} = \sum_{j=1}^{n} \sum_{k=1}^{n} \mathbf{A}_{ij}\mathbf{A}_{jk}\mathbf{A}_{ki}. \tag{1}$$

For balanced triangles, the product of the edge signs is positive, whereas for unbalanced ones it is negative. For the case that any of the edges is missing

(and therefore the triangle does not exist), the product yields zero. Therefore, the sum over all possible configurations, i.e., the value of $\mathbf{A}^3_{ii}$, is increased by one if the triangle of a path $i \to j \to k$ is balanced. It is decreased by one, if it is unbalanced. However, since the graph is undirected, each unique triangle is counted twice in $\mathbf{A}^3_{ii}$; again, triangle $\{i, j, k\}$ is counted both as $i \to j \to k$ and as $i \to k \to j$. Consequently we have to divide by two.

Propositions 1 and 2 will become useful in proving the main theorems of the paper in Sections 4 and 5. We will also use the following results related to the eigenvalue decomposition of any real symmetric matrix $\mathbf{X}$[1]. First, recall that every real, symmetric $n \times n$ graph has n real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. We also know the following:

Proposition 3 ([8]). *For any $n \times n$ matrix $\mathbf{X}$, the sum of the diagonal elements of $\mathbf{X}$ is equal to the sum of its eigenvalues. That is, if $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of $\mathbf{X}$, then we have that trace $(\mathbf{X}) = \sum_{i=1}^{n} \lambda_i$.*

Proposition 4 ([8]). *Let $\mathbf{X}$ be a real, $n \times n$ and symmetric matrix with eigenvalue decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ and diag $(\mathbf{\Lambda}) = (\lambda_1, \ldots, \lambda_n)$. Then, for integer k, the eigenvalue decomposition of matrix $\mathbf{X}^k$ is $\mathbf{X}^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^T$. Where $\mathbf{\Lambda}^k$ is a diagonal matrix with diag$(\mathbf{\Lambda}) = (\lambda_1^k, \ldots, \lambda_n^k)$.*

4 Balanced Triangles in Complete Networks

In this section, we show how the fraction of balanced triangles $\widehat{b}$ of a fully connected signed network can be expressed as a function of the eigenvalues of the network's adjacency matrix. We express this relationship formally in Theorem 1.

Theorem 1. *Let $G = (V, E)$ be a fully connected signed network with adjacency matrix $\mathbf{A}$. If $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of $\mathbf{A}$, then the fraction of balanced triangles $\widehat{b}$ of G can be computed as follows:*

$$\widehat{b} = \frac{1}{2} + \frac{1}{4n(n-1)(n-2)} \sum_{i=1}^{n} \lambda_i^3. \tag{2}$$

Before getting into the details of the proof, observe that in a fully-connected signed graph all nodes participate in the same number of signed triangles. That is, we have the following fact.

Fact 1. *Every node $i \in V$ participates in the same number of distinct triangles. That is, if we denote this number by δ_i, we have that $\delta_1 = \delta_2 = \ldots = \delta_n = \delta$ and $\delta = \binom{n-1}{2}$.*

[1] Recall that both the adjacency and the connectivity matrices $\mathbf{A}$ and $\mathbf{G}$ are real and symmetric.

This is simply because every node participates in triangles formed by all combinations of pairs of the remaining $n-1$ nodes (excluding the node itself).

For every node i, the δ triangles it participates in can be partitioned into b_i balanced ones and u_i unbalanced ones. That is,

$$\delta = b_i + u_i \ , \forall i \in \{1, 2, ..., n\}\,. \tag{3}$$

Due to Proposition 2, we have that

$$\frac{\mathbf{A}^3_{ii}}{2} = b_i - u_i \tag{4}$$

By (3) and (4), we obtain that

$$b_i = \frac{1}{2}\delta + \frac{1}{4}\mathbf{A}^3_{ii} \text{ and } u_i = \frac{1}{2}\delta - \frac{1}{4}\mathbf{A}^3_{ii}.$$

Therefore, the total number of balanced and unbalanced triangles, denoted by b and u, will be

$$b = \frac{1}{3}\sum_{i=1}^{n} b_i = \frac{1}{3}\left[\frac{1}{2}n\delta + \frac{1}{4}\text{trace}\left(\mathbf{A}^3\right)\right] \text{ and } u = \frac{1}{3}\sum_{i=1}^{n} u_i = \frac{1}{3}\left[\frac{1}{2}n\delta - \frac{1}{4}\text{trace}\left(\mathbf{A}^3\right)\right].$$

We divide both quantities by 3 because every triangle has three participating nodes, and thus, is triple-counted. Consequently, the fraction of balanced triangles in the network, $\widehat{b}$, is

$$\widehat{b} = \frac{b}{b+u} = \frac{1}{4n\delta}\text{trace}\left(\mathbf{A}^3\right) + \frac{1}{2} = \frac{1}{4n(n-1)(n-2)}\text{trace}\left(\mathbf{A}^3\right) + \frac{1}{2}.$$

The last derivation is due to the fact that $\delta = \binom{n-1}{2}$. Using Propositions 3 and 4, we get the desired result.

Algorithmic implications: Equation (2) implies that we can compute the fraction of balanced triangles by simply computing the eigenvalues of $\mathbf{A}$ and take their third power (this is due to Proposition 4). However, computing all the n eigenvalues of $\mathbf{A}$ still requires time $\mathcal{O}(n^3)$. Fortunately, if one is content with approximations of $\widehat{b}$, Theorem 1 can lead to an algorithm with significantly smaller running time. Assume the permutation of the eigenvalues of $\mathbf{A}$ (i.e., $\lambda_1, \ldots, \lambda_n$) in decreasing order of their magnitude. Then, instead of using all the n eigenvalues in the computation of $\widehat{b}$, we can compute $\widehat{b}_k$ by only using the top-k eigenvalues with the largest magnitude. Therefore, computing these eigenvalues requires time only $\mathcal{O}(n^2k)$ – this is because the standard algorithms for computing eigenvalue decomposition output the eigenvalues sequentially and in decreasing order of their magnitude [8]. For small values of k, the savings in terms of running time can be significant. Our algorithm can further leverage the efficient approximation algorithms for computing eigenvalues of large matrices [7]. Our experiments demonstrate that very small values of k (compared

to n) suffice to give significant computational gains with insignificant accuracy loses. This is because, as our experiments demonstrate, the effective rank of $\mathbf{A}$ is small.

5 Balanced Triangles in Arbitrary Networks

Here we show that even when the signed graph is not a clique, we can express the fraction of balanced triangles as a function of the eigenvalues of the adjacency matrix $\mathbf{A}$ and the connectivity matrix $\mathbf{G}$.

Theorem 2. *Let $G = (V, E)$ be a symmetric signed network (not necessarily fully connected) with adjacency matrix $\mathbf{A}$ and connectivity matrix $\mathbf{G}$. Also let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of $\mathbf{A}$ and $\mu_1, \mu_2, \ldots, \mu_n$ the eigenvalues of $\mathbf{G}$. Then, the fraction of balanced triangles in G can be expressed as follows:*

$$\widehat{b} = \frac{1}{2}\left(1 + \frac{\sum_{i=1}^{n} \lambda_i^3}{\sum_{i=1}^{n} \mu_i^3}\right). \tag{5}$$

The main idea of the proof of Theorem 2 is similar to that of Theorem 1. The only difference is that now, the number of triangles δ_i that a node i participates in is different for every node i and can be computed using Proposition 1. If b_i and u_i denote the number of balanced and unbalanced triangle that node i participates in, then we have that

$$\delta_i = b_i + u_i. \tag{6}$$

Every balanced triangle contributes a (+1) (all three edges positive or two edges negative and one positive). Thus, the diagonal entries of $\mathbf{A}^3$ are numbers of the form

$$\frac{\mathbf{A}^3_{ii}}{2} = b_i - u_i. \tag{7}$$

Using Proposition 1 and Equations (6) and (7), we get

$$b_i = \frac{1}{4}\left(\mathbf{G}^3_{ii} + \mathbf{A}^3_{ii}\right) \quad \text{and} \quad u_i = \frac{1}{4}\left(\mathbf{G}^3_{ii} - \mathbf{A}^3_{ii}\right).$$

The total numbers of balanced and unbalanced triangles of the graph, denoted by b and u respectively, are then given by

$$b = \frac{1}{3}\sum_{i=1}^{n} b_i = \frac{1}{12}\left[\operatorname{trace}\left(\mathbf{G}^3\right) + \operatorname{trace}\left(\mathbf{A}^3\right)\right]$$

$$u = \frac{1}{3}\sum_{i=1}^{n} u_i = \frac{1}{12}\left[\operatorname{trace}\left(\mathbf{G}^3\right) - \operatorname{trace}\left(\mathbf{A}^3\right)\right].$$

As before, the division by 3 is due to the fact, that the summation goes over all nodes in the network, what results in a triple-counting. It follows that the fraction of balanced triangles in the network, is

$$\widehat{b} = \frac{b}{b+u} = \frac{1}{2}\left(1 + \frac{\text{trace}\left(\mathbf{A}^3\right)}{\text{trace}\left(\mathbf{G}^3\right)}\right).$$

Using Propositions 3 and 4, we can replace the quantities $\text{trace}\left(\mathbf{A}^3\right)$ and $\text{trace}\left(\mathbf{G}^3\right)$ by $\sum_{i=1}^{n} \lambda_i^3$ and $\sum_{i=1}^{n} \mu_i^3$ respectively, and get the desired result.

Algorithmic implications: Equation (2) implies that we can compute the fraction of balanced triangles by computing the eigenvalues of $\mathbf{A}$ and take their third power (this is due to Proposition 4). However, computing all the n eigenvalues of $\mathbf{A}$ still requires time $\mathcal{O}(n^3)$. Fortunately, if one is content with approximations of $\widehat{b}$, Theorem 2 can lead to an algorithm with significantly smaller running time. Instead of computing all the n eigenvalues of $\mathbf{A}$ and $\mathbf{G}$, we can evaluate Equation (5) using only the k largest-magnitude eigenvalues of matrices $\mathbf{A}$ and $\mathbf{G}$. We call the algorithm that uses Theorem 2 and the first k eigenvalues of $\mathbf{G}$ and $\mathbf{A}$ for computing $\widehat{b}$, the SPECTRAL(k) algorithm. For small values of k, the running time of SPECTRAL(k) is very small. Further computational improvements can be achieved by leveraging more efficient approximation algorithms for eigenvalue computations [7]. Our experiments demonstrate that the values of k that give satisfactory approximations of $\widehat{b}$ are very small because, as our experiments show, the effective ranks of $\mathbf{A}$ and $\mathbf{G}$ are both small and similar.

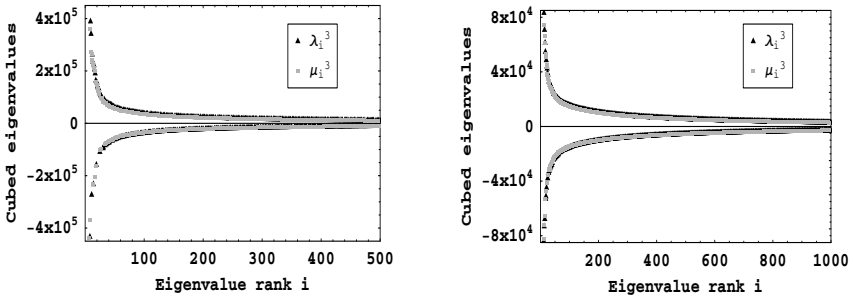
6 Experiments

Our experiments both with real and synthetic datasets demonstrate that a very small number of eigenvalues of the adjacency and the connectivity matrix suffices to provide accurate approximations of $\widehat{b}$.

Datasets: For our experiments we use two real signed-graph datasets: **Epinions** and **Slashdot**. We have downloaded both datasets from `http://snap.stanford.edu/data/`[2]. Since both datasets were directed, we obtained their undirected versions as follows: For every directed edge $e(i \to j)$ such that $e(j \to i)$ does not exist, we create an undirected edge that retains the same sign as the original directed edge. If both $e(i \to j)$ and $e(j \to i)$ exist and have the same sign, we again, created undirected edge between i and j with the same sign. If edges $e(i \to j)$ and $e(j \to i)$ have contradicting signs, we leave nodes i and j to be disconnected. The resulting **Epinions** dataset has $n = 131,828$ nodes and $|E| = 711,210$ undirected edges; 83% of these edges are positive. The resulting **Slashdot** dataset contains $n = 77,357$ nodes and $|E| = 468,554$ edges; 76% of these edges are signed positive.

We also generate synthetic scale-free (**SF**) graphs using the generative model proposed by Barabási and Albert [2]. We convert the generated unsigned graphs into signed graphs by randomly assigning a "+' (or a "-") sign to every edge of the graph with probability equal to p_+. The value of p_+ is a parameter that we vary for our experimental evaluation.

[2] We used the file *soc-sign-Slashdot081106.txt.gz* for Slashdot.



(a) **Epinions** dataset (b) **Slashdot** dataset

Fig. 2. Distribution of the cubed eigenvalues λ_i^3 and μ_i^3 of the adjacency matrix $\mathbf{A}$ and the connectivity matrix $\mathbf{G}$ for the Epinions and the Slashdot graph, ordered by magnitude in decreasing order

6.1 Evaluating Spectral on Real-World Signed Networks

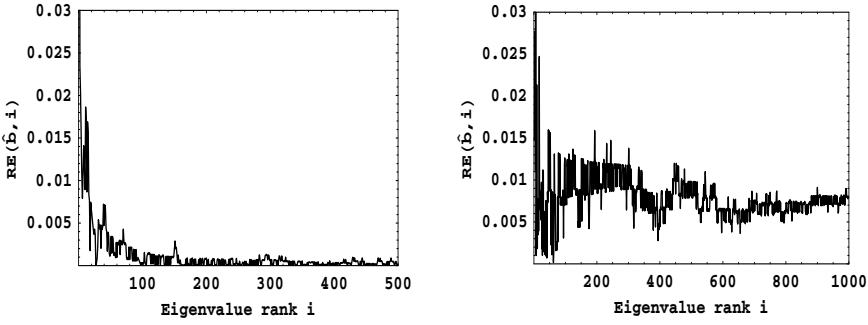
Recall, that Theorems 1 and 2 indicate that, in order to compute $\widehat{b}$, it suffices to know the sum of the cubed eigenvalues of the adjacency ($\mathbf{A}$) and the connectivity ($\mathbf{G}$) matrices. Since our objective is to approximate these sums by considering only some of the eigenvalues, we need to show that the first eigenvalues contribute the major part to the total sum. Therefore, as a first experiment, we examine the distributions of the cubed eigenvalues of $\mathbf{A}$ and $\mathbf{G}$ for the **Epinions** and **Slashdot** datasets. Figures 2(a) and 2(b) show the result for the first cubed eigenvalues of the **Epinions** and **Slashdot** graphs, ordered by magnitude. There are two crucial features, which justify the approximation of the sum over all eigenvalues by the ones with the largest magnitudes. First, the distribution of the magnitude versus the rank is highly skewed and therefore the early summands contribute the most. Notice that the skew is identical for the eigenvalues of $\mathbf{A}^3$ and $\mathbf{G}^3$ matrices, meaning that both matrices have the same effective rank. Second, the signs of the eigenvalues switch between positive and negative. Therefore, many of the eigenvalues in the tail cancel each other out.

To quantify how well the actual result of $\widehat{b}$ can be approximated, we evaluate the relative error

$$\text{Re}\left(\widehat{b}, i\right) = \frac{|\widehat{b}_i - \widehat{b}|}{\widehat{b}}. \tag{8}$$

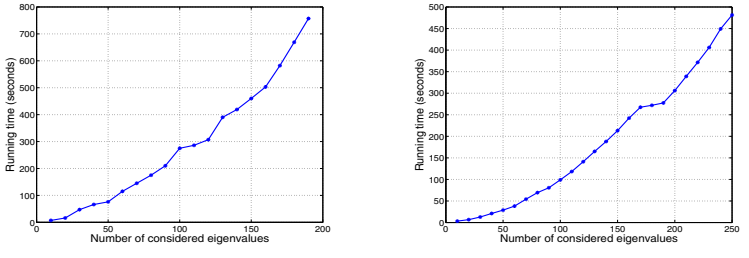
The term $\widehat{b}_i$ is the result of Spectral that considers the first i eigenvalues of $\mathbf{A}$ and $\mathbf{G}$. The exact result $\widehat{b}$ is obtained by counting all the existing triangles using the exact cubic algorithm that we call Naive.

The relative error of the output of Spectral, as a function of the number of considered eigenvalues, i, for the **Epinions** and the **Slashdot** datasets, are shown in Figures 3(a) and 3(b) respectively. For the **Epinions** data, taking into account 10, out of a total of 131,828 eigenvalues, gives an approximation with



(a) **Epinions** dataset (b) **Slashdot** dataset

Fig. 3. Relative error for the approximation of $\widehat{b}$ obtained by SPECTRAL for the **Epinions** and the **Slashdot** datasets, when considering only the i eigenvalues with the largest magnitude



(a) **Epinions** dataset (b) **Slashdot** dataset

Fig. 4. Running time (in seconds) of SPECTRAL as a function of the number of considered eigenvalues

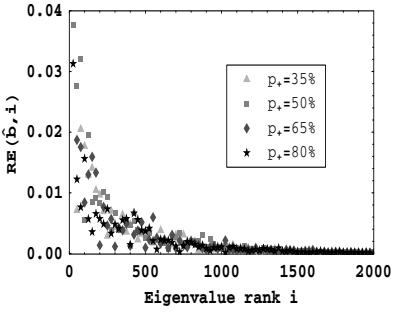
relative error less than 2%. Similarly, for the **Slashdot** data, taking into account ≈ 25 eigenvalues, out of the $77,357$ total eigenvalues, leads to an approximation that has less than 2% relative error.

Figures 4(a) and 4(b) show the running time (in seconds) of the SPECTRAL algorithm on **Epinions** and **Slashdot** data, as a function of the number of considered eigenvalues of $\mathbf{G}$ and $\mathbf{A}$. One can see, that the running time of the algorithm scales almost linearly to the number of eigenvalues. In numbers: for the **Epinions** dataset, NAIVE needs 753 seconds for exact computation, while SPECTRAL gives an estimate with relative error less than 0.9% in 7 seconds. For the **Slashdot** dataset, SPECTRAL with only 10 eigenvalues runs in just 3 seconds and provides a solution with relative error less than 0.7%. Compare this to the running time of NAIVE that is 262 seconds for the same dataset. Note that although the error is not necessarily dropping as the number of

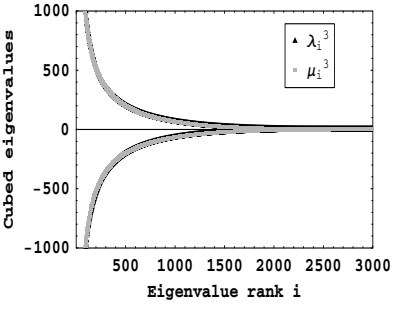
considered eigenvalues increases, it never exceeds 1%. Since more eigenvalues increase the running time of SPECTRAL, using the top-10 magnitude eigenvalues is a reasonable rule of thumb.[3] Neither of our implementations were optimized for run-time experiments, however, the times we report here are indicative of the computational savings achieved by the SPECTRAL algorithm.

6.2 Evaluating SPECTRAL on Synthetic Signed Networks

Here, we further investigate the accuracy of SPECTRAL for signed networks that have different underlying topologies. For that, we generate **SF** graphs with $n = 3000$ nodes and $|E| = 30,000$ edges. We also examine the impact of the parameter p_+ on the overall performance of the algorithm. Figure 5(a) shows the relative error obtained by SPECTRAL, as a function of the number of considered eigenvalues. Every point is an average over four different network realizations with the same set of generation parameters. As we can see, p_+ hardly affects the accuracy of the approximation. Consideration of approximately 100 eigenvalues (out of the total of $3,000$) leads to an error of less than 2%. In Figure 5(b), we observe that the eigenvalues of a **SF** graph with $p_+ = 80\%$ exhibit the same two crucial properties that were pointed out in real graphs. That is, the magnitude of the eigenvalues are highly skewed. This skew is almost identical for both **G** and **A** matrices and similar both for eigenvalues with positive and negative signs. It turns out that, for **SF** graphs, the shape of these distributions is independent of the value of p_+.



(a) Relative error of the approximation of $\widehat{b}$ obtained by SPECTRAL for $n = 3000$ and $p_+ \in \{35\%, 50\%, 65\%, 80\%\}$, as a function of the number of eigenvalues considered.



(b) Distribution of the cubed eigenvalues λ_i^3 and μ_i^3 of matrices **A** and **G**; values ordered by magnitude in decreasing order and $p_+ = 80\%$

Fig. 5. Experiments with Scale-free graphs

[3] We implemented SPECTRAL using Mathematica and NAIVE using Java. The running times of NAIVE reported here are generous, since we rearrange the input so that it allows NAIVE to do efficient in-memory computations.

7 Conclusions

Given a signed undirected network, we have presented a spectral algorithm for computing the fraction of balanced triangles in the network. The basis of our algorithm lies in the dependency between the the fraction of balanced triangles of the network and the eigenvalues of the adjacency and the connectivity matrices that describe the signed network. After establishing the form of this connection theoretically, we have exploited it in order to device Spectral, a fast algorithm that approximates the fraction of the balanced triangles in the network. In an extensive experimental evaluation, both on real and generated signed networks, we have demonstrated that the fraction of balanced and unbalanced triangles can be approximated very efficiently using our algorithm. In addition to that, our experiments also revealed certain interesting properties of the eigenvalues of the adjacency and the connectivity matrices of signed networks. We view our methodology as a useful computational tool for the studies of structural balance in large online signed social networks.

References

1. Antal, T., Krapivsky, P.L., Redner, S.: Social balance on networks: The dynamics of friendship and enmity. Physica D 130 (2006)
2. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286 (1999)
3. Brzozowski, M.J., Hogg, T., Szabá, G.: Friends and foes: ideological social networking. In: Human Factors in Computing Systems, CHI (2008)
4. Burke, M., Kraut, R.: Mopping up: modeling wikipedia promotion decisions. In: ACM Conference on Computer Supported Cooperative Work (CSCW), pp. 27–36. ACM Press, New York (2008)
5. Cartwright, D., Harary, F.: Structure balance: A generalization of heider's theory. Psychological Review 63(5), 277–293 (1956)
6. Davis, J.A.: Structural balance, mechanical solidarity, and interpersonal relations. American Journal of Sociology 68, 444–462 (1963)
7. Drineas, P., Kannan, R., Mahoney, M.W.: Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. SIAM J. Comput. 36(1), 158–183 (2006)
8. Golub, G., Loan, C.V.: Matrix Computations. Johns Hopkins Press, Baltimore (1989)
9. Guha, R.V., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: International Conferece on World Wide Web (WWW), pp. 403–412 (2004)
10. Harary, F.: On the notion of balance of a signed graph. Michigan Math. Journal 2(2), 143–146 (1953)
11. Heider, F.: Social perception and phenomenal causality. Psychological Rev. 51(6), 358–374 (1944)
12. Heider, F.: Attitudes and cognitive organization. Journal of Psychology 21, 107–112 (1946)
13. Kunegis, J., Lommatzsch, A., Bauckhage, C.: The slashdot zoo: Mining a social network with negative edges. In: International Conference on World Wide Web (WWW), pp. 741–750 (2009)

14. Lampe, C., Johnston, E., Resnick, P.: Follow the reader: filtering comments on slashdot. In: Human Factors in Computing Systems, CHI (2007)
15. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: International CIonference on World Wide Web (WWW), pp. 641–650 (2010)
16. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Signed networks in social media. In: Human Factors in Computing Systems (CHI), pp. 1361–1370 (2010)
17. Marvel, S., Kleinberg, J., Strogatz, S.: The energy landscape of social balance. Physical Review Letters 103(19) (2009)
18. Massa, P., Avesani, P.: Controversial users demand local trust metrics: an experimental study on epinions.com community. In: National Conference on Artificial Intelligence (AAAI), pp. 121–126 (2005)
19. Tsourakakis, C.E.: Fast counting of triangles in large real networks without counting: Algorithms and laws. In: International Conference on Data Mining (ICDM), pp. 608–617 (2008)
20. Tsourakakis, C.E., Drineas, P., Michelakis, E., Koutis, I., Faloutsos, C.: Spectral counting of triangles in power-law networks via element-wise sparsification. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 66–71 (2009)

High-Ordered Random Walks and Generalized Laplacians on Hypergraphs

Linyuan Lu* and Xing Peng*

University of South Carolina
lu@math.sc.edu, pengx@mailbox.sc.edu

Abstract. Despite of the extreme success of the spectral graph theory, there are relatively few papers applying spectral analysis to hypergraphs. Chung first introduced Laplacians for regular hypergraphs and showed some useful applications. Other researchers treated hypergraphs as weighted graphs and then studied the Laplacians of the corresponding weighted graphs. In this paper, we aim to unify these very different versions of Laplacians for hypergraphs. We introduce a set of Laplacians for hypergraphs through studying high-ordered random walks on hypergraphs. We prove the eigenvalues of these Laplacians can effectively control the mixing rate of high-ordered random walks, the generalized distances/diameters, and the edge expansions.

1 Introduction

Many complex networks have richer structures than graphs can have. Inherently they have hypergraph structures: interconnections often cross multiple nodes. Treating these networks as graphs causes a loss of some structures. Nonetheless, it is still popular to use graph tools to study these networks; one of them is the Laplacian spectrum. Let G be a graph on n vertices. The *Laplacian* $\mathcal{L}$ of G is the $(n \times n)$-matrix $I - T^{-1/2}AT^{-1/2}$, where A is the adjacency matrix and T is the diagonal matrix of degrees. Let $\lambda_0, \lambda_1, \ldots, \lambda_{n-1}$ be the eigenvalues of $\mathcal{L}$, indexed in non-decreasing order. It is known that $0 \leq \lambda_i \leq 2$ for $0 \leq i \leq n-1$. If G is connected, then $\lambda_1 > 0$. The first nonzero Laplacian eigenvalue λ_1 is related to many graph parameters, such as the mixing rate of random walks, the graph diameter, the neighborhood expansion, the Cheeger constant, the isoperimetric inequalities, expander graphs, quasi-random graphs, etc [1, 2, 6, 9, 10, 20, 21].

In this paper, we define a set of Laplacians for hypergraphs. Laplacians for regular hypergraphs was first introduced by Chung [8] in 1993 using homology approach. The first nonzero Laplacian eigenvalue can be used to derive several useful isoperimetric inequalities. It seems hard to extend Chung's definition to general hypergraphs. Other researchers treated a hypergraph as a multi-edge graph and then defined its Laplacian to be the Laplacian of the corresponding multi-edge graph. For example, Rodríguez [25] showed that the approach above

* This author was supported in part by NSF grant DMS 1000475.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 14–25, 2011.

had some applications to bisections, the average minimal cut, the isoperimetric number, the max-cut, the independence number, the diameter etc.

What are "right" Laplacians for hypergraphs? To answer this question, let us recall how the Laplacian was introduced in the graph theory. One of the approaches is using geometric/homological analogue, where the Laplacian is defined as a self-joint operator on the functions over vertices. Another approach is using random walks, where the Laplacian is the symmetrization of the transition matrix of the random walk on a graph. Chung [6] took the first approach and defined her Laplacians for regular hypergraphs. In this paper, we take the second approach and define the Laplacians through high-ordered random walks on hypergraphs.

A high-ordered walk on a hypergraph H can be roughly viewed as a sequence of overlapped oriented edges $F_1, F_2, \ldots, F_k$. For $1 \leq s \leq r-1$, we say $F_1, F_2, \ldots, F_k$ is an s-walk if $|F_i \cap F_{i+1}| = s$ for each i in $\{1, 2, 3, \ldots, k-1\}$. For example, a *Hamiltonian s-cycle* is a special s-walk which covers each vertex exactly once. There are many papers [4, 5, 13, 14, 15, 16, 18, 19, 23, 24] studying Hamiltonian s-cycles in hypergraphs. The detail definition of high-ordered random walks will be given later. The choice of s enables us to define a set of Laplacian matrices $\mathcal{L}^{(s)}$ for H. For $s = 1$, our definition of Laplacian $\mathcal{L}^{(1)}$ is the same as the definition in [25]. For $s = r-1$, while we restrict to regular hypergraphs, our definition of Laplacian $\mathcal{L}^{(r-1)}$ is similar to Chung's definition [8]. We will discuss their relations in the last section. Our definition of Laplacians are also closely related to the singular values used in an unpublished work of Butler [3].

In this paper, we show several applications of the Laplacians of hypergraphs, such as the mixing rate of high-ordered random walks, the generalized diameters, and the edge expansions. Our approach allows users to select a "right" Laplacian to fit their special need.

Our definition of Laplacians for hypergraphs depends on previous knowledge of the Laplacian for weighted graphs and Eulerian directed graphs, which can be found at the full version of this paper [22]. The definition of Laplacians for hypergraphs will be given in section 2. In section 3, we will prove some properties of the Laplacians of hypergraphs. In section 4, we will consider several applications using the Laplacians of hypergraphs. In last section, we will comment on the future direction.

2 Definition of the s-th Laplacian

For a positive integer s and a vertex set V, let $V^{\underline{s}}$ be the set of all (ordered) s-tuples consisting of s distinct elements in V. Let $\binom{V}{s}$ be the set of all unordered (distinct) s-subset of V. Let $\mathbf{1}$ be the row (or the column) vector with all entries of value 1 and I be the identity matrix. For a row (or column) vector f, the norm $\|f\|$ is always the L_2-norm of f.

Let H be an r-uniform hypergraph (or an r-graph for short) with the vertex set $V(H)$ (or V for short) and the edge set $E(H)$. We assume $|V(H)| = n$ and

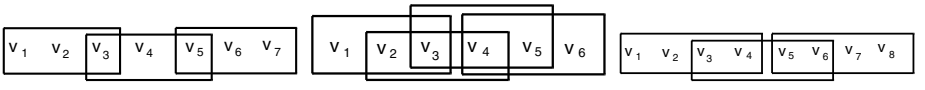
$E(H) \subseteq \binom{V}{r}$. For a vertex subset S such that $|S| < r$, the *neighborhood* $\Gamma(S)$ is $\{T | S \cap T = \emptyset$ and $S \cup T$ is an edge in $H\}$. Let the *degree* d_S of S in H be the number of edges containing S, i.e, $d_S = |\Gamma(S)|$. For $1 \leq s \leq r-1$, an s-walk of length k is a sequence of vertices

$$v_1, v_2, \ldots, v_j, \ldots, v_{(r-s)(k-1)+r}$$

together with a sequence of edges $F_1, F_2, \ldots, F_k$ such that

$$F_i = \{v_{(r-s)(i-1)+1}, v_{(r-s)(i-1)+2}, \ldots, v_{(r-s)(i-1)+r}\}$$

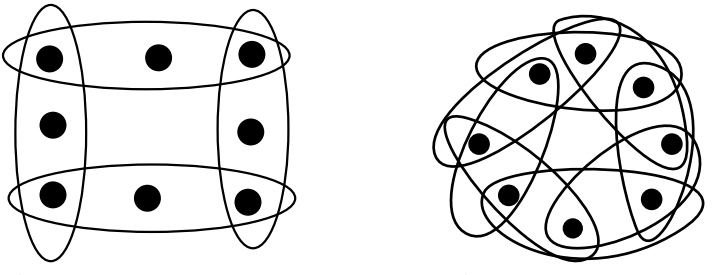
for $1 \leq i \leq k$. Here are some examples of s-walks as shown in Figure 1.



A 1-walk in a 3-graph A 2-walk in a 3-graph A 2-walk in a 4-graph

Fig. 1. Three examples on an s-walk in a hypergraph

For each i in $\{0, 1, \ldots, k\}$, the *i-th stop* x_i of the s-walk is the ordered s-tuple $(v_{(r-s)i+1}, v_{(r-s)i+2}, \ldots, v_{(r-s)i+s})$. The initial stop is x_0, and the terminal stop is x_k. An s-walk is a *s-path* if stops (as ordered s-tuples) are distinct. If $x_0 = x_k$, then an s-walk is *closed*.



A 1-cycle in a 3-graph A 2-cycle in a 3-graph

Fig. 2. An example of a loose cycle and a tight cycle in a 3-graph

An *s-cycle* is a special closed s-walk such that $v_1, v_2, \ldots, v_{(r-s)k}$ are distinct and $v_{(r-s)k+j} = v_j$ for $1 \leq j \leq s$ (see Figure 2). An s-cycle is a *loose* cycle if $s \leq \frac{r}{2}$ (particularly $s = 1$); an s-cycle is a *tight* cycle if $s > \frac{r}{2}$ (particularly $s = r-1$). An s-cycle is *Hamiltonian* if it covers each vertex in H exactly once. In the literature, Katona and Kierstead [15] first studied Hamiltonian tight cycles. Rödl-Ruciński-Szemerédi [23, 24], Kühn et al [16, 18, 19], and Hán-Schacht [14] studied Hamiltonian s-cycles for a full range of s. Frieze and Dudek [4, 5, 13] studied Hamiltonian s-cycles in random r-uniform hypergraphs.

For $1 \le s \le r-1$ and $x, y \in \mathrm{V}^{\underline{s}}$, the s-distance $d^{(s)}(x,y)$ is the minimum integer k such that there exists an s-path of length k starting from x and ending at y. A hypergraph H is *s-connected* if $d^{(s)}(x,y)$ is finite for every pair (x,y). If H is s-connected, then the *s-diameter* of H is the maximum value of $d^{(s)}(x,y)$ for $x, y \in \mathrm{V}^{\underline{s}}$.

A random s-walk with initial stop x_0 is an s-walk generated as follows. Let x_0 be the sequence of visited vertices at initial step. At each step, let S be the set of last s vertices in the sequence of visited vertices. A random $(r-s)$-set T is chosen from $\Gamma(S)$ uniformly; the vertex in T is added into the sequence one by one in an arbitrary order.

For $0 \le \alpha \le 1$, an α-lazy random s-walk is a modified random s-walk such that with probability α, one can stay at the current stop; with probability $1-\alpha$, append $r-s$ vertices to the sequence as selected in a random s-walk.

For $x \in \mathrm{V}^{\underline{s}}$, let $[x]$ be the s-set consisting of the coordinates of x.

2.1 Case $1 \le s \le r/2$

For $1 \le s \le r/2$, we define a weighted undirected graph $G^{(s)}$ over the vertex set $\mathrm{V}^{\underline{s}}$ as follows. Let the weight $w(x,y)$ be $|\{F \in E(H) : [x] \sqcup [y] \subseteq F\}|$. Here $[x] \sqcup [y]$ is the disjoint union of $[x]$ and $[y]$. In particular, if $[x] \cap [y] \neq \emptyset$, then $w(x,y) = 0$.

For $x \in \mathrm{V}^{\underline{s}}$, the degree of x in $G^{(s)}$, denoted by $d_x^{(s)}$, is given by

$$d_x^{(s)} = \sum_y w(x,y) = d_{[x]} \binom{r-s}{s} s!. \tag{1}$$

Here $d_{[x]}$ means the degree of the set $[x]$ in the hypergraph H. When we restrict an s-walk on H to its stops, we get a walk on $G^{(s)}$. This restriction keeps the length of the walk. Therefore, the s-distance $d^{(s)}(x,y)$ in H is simply the graph distance between x and y in $G^{(s)}$; the s-diameter of H is simply the diameter of the graph $G^{(s)}$.

A random s-walk on H is essentially a random walk on $G^{(s)}$. It can be constructed from a random walk on $G^{(s)}$ by inserting additional random $r-2s$ vertices T_i between two consecutive stops x_i and x_{i+1} at time i, where T_i is chosen uniformly from $\Gamma([x_i] \cup [x_{i+1}])$ and the vertex in T_i is inserted between x_i and x_{i+1} in an arbitrary order.

Therefore, we define the *s-th Laplacian* $\mathcal{L}^{(s)}$ of H to be the Laplacian of the weighted undirected graph $G^{(s)}$.

The eigenvalues of $\mathcal{L}^{(s)}$ are listed as $\lambda_0^{(s)}, \lambda_1^{(s)}, \ldots, \lambda_{\binom{n}{s}s!}^{(s)}$ in the non-decreasing order. Let $\lambda_{\max}^{(s)} = \lambda_{\binom{n}{s}s!}^{(s)}$ and $\bar{\lambda}^{(s)} = \max\{|1-\lambda_1^{(s)}|, |1-\lambda_{\max}^{(s)}|\}$. For some hypergraphs, the numerical values of $\lambda_1^{(s)}$ and $\lambda_{\max}^{(s)}$ are shown in Table 1 at the end of this section.

2.2 The Case $r/2 < s \leq r-1$

For $r/2 < s \leq r-1$, we define a directed graph $D^{(s)}$ over the vertex set $V^{\underline{s}}$ as follows. For $x, y \in V^{\underline{s}}$ such that $x = (x_1, \ldots, x_s)$ and $y = (y_1, \ldots, y_s)$, let (x, y) be a directed edge if $x_{r-s+j} = y_j$ for $1 \leq j \leq 2s-r$ and $[x] \cup [y]$ is an edge of H.

For $x \in V^{\underline{s}}$, the out-degree d_x^+ in $D^{(s)}$ and the in-degree d_x^- in $D^{(s)}$ satisfy

$$d_x^+ = d_{[x]}(r-s)! = d_x^-.$$

Thus $D^{(s)}$ is a Eulerian directed graph. We write $d_x^{(s)}$ for both d_x^+ and d_x^-. Now $D^{(s)}$ is strongly connected if and only if it is weakly connected.

Note that an s-walk on H can be naturally viewed as a walk on $D^{(s)}$ and vice versa. Thus the s-distance $d^{(s)}(x, y)$ in H is exactly the directed distance from x to y in $G^{(s)}$; the s-diameter of H is the diameter of $D^{(s)}$. A random s-walk on H is one-to-one corresponding to a random walk on $D^{(s)}$.

For $\frac{r}{2} < s \leq r-1$, we define the *s-th Laplacian* $\mathcal{L}^{(s)}$ as the Laplacian of the Eulerian directed graph $D^{(s)}$.

The eigenvalues of $\mathcal{L}^{(s)}$ are listed as $\lambda_0^{(s)}, \lambda_1^{(s)}, \ldots, \lambda_{\binom{n}{s}s!}^{(s)}$ in the non-decreasing order. Let $\lambda_{\max}^{(s)} = \lambda_{\binom{n}{s}s!}^{(s)}$ and $\bar{\lambda}^{(s)} = \max\{|1-\lambda_1^{(s)}|, |1-\lambda_{\max}^{(s)}|\}$. For some hypergraphs, the numerical values of $\lambda_1^{(s)}$ and $\lambda_{\max}^{(s)}$ are shown in Table 1 at the end of this section.

2.3 Examples

Let K_n^r be the complete r-uniform hypergraph on n vertices. Here we compute the values of $\lambda_1^{(s)}$ and $\lambda_{\max}^{(s)}$ for some K_n^r (see Table 1).

Table 1. The values of $\lambda_1^{(s)}$ and $\lambda_{\max}^{(s)}$ of some complete hypergraphs K_n^r

H	$\lambda_1^{(4)}$	$\lambda_1^{(3)}$	$\lambda_1^{(2)}$	$\lambda_1^{(1)}$	$\lambda_{\max}^{(1)}$	$\lambda_{\max}^{(2)}$	$\lambda_{\max}^{(3)}$	$\lambda_{\max}^{(4)}$
K_6^3			3/4	6/5	6/5	3/2		
K_7^3			7/10	7/6	7/6	3/2		
K_6^4		1/3	5/6	6/5	6/5	3/2	1.76759	
K_7^4		3/8	9/10	7/6	7/6	7/5	7/4	
K_6^5	0.1464	1/2	5/6	6/5	6/5	3/2	3/2	1.809
K_7^5	0.1977	5/8	9/10	7/6	7/6	7/5	3/2	1.809

Remark: From the table above, we observe $\lambda_1^{(s)} = \lambda_{\max}^{(s)}$ for some complete hypergraphs. In fact, this is true for any complete hypergraph K_n^r. We point out the following fact without the proof. For an r-uniform hypergraph H and an integer s such that $1 \leq s \leq \frac{r}{2}$, $\lambda_1^{(s)}(H) = \lambda_{\max}^{(s)}(H)$ holds if and only if $s = 1$ and H is a 2-design.

3 Properties of Laplacians

In this section, we prove some properties of the Laplacians for hypergraphs.

Lemma 1. *For $1 \le s \le r/2$, we have the following properties.*

1. *The s-th Laplacian has $\binom{n}{s}s!$ eigenvalues and all of them are in $[0, 2]$.*
2. *The number of 0 eigenvalues is the number of connected components in $G^{(s)}$.*
3. *The Laplacian $L^{(s)}$ has an eigenvalue 2 if and only if $r = 2s$ and $G^{(s)}$ has a bipartite component.*

Proof: Items 1 and 2 follow from the facts of the Laplacian of $G^{(s)}$. If $\mathcal{L}^{(s)}$ has an eigenvalue 2, then $G^{(s)}$ has a bipartite component T. We want to show $r = 2s$. Suppose $r \ge 2s + 1$. Let $\{v_0, v_2, \ldots, v_{r-1}\}$ be an edge in T. For $0 \le i \le 2s$ and $0 \le j \le s - 1$, let $g(i, j) = is + j \mod (2s + 1)$ and $x_i = (v_{g(i,0)}, \ldots, v_{g(i,s-1)})$. Observe $x_0, x_1, \ldots, x_{2s}$ form an odd cycle in $G^{(s)}$. Contradiction. □

The following lemma compares $\lambda_1^{(s)}$ and $\lambda_{\max}^{(s)}$ for different s.

Lemma 2. *Suppose that H is an r-uniform hypergraph. We have*

$$\lambda_1^{(1)} \ge \lambda_1^{(2)} \ge \ldots \ge \lambda_1^{(\lfloor r/2 \rfloor)}; \tag{2}$$

$$\lambda_{\max}^{(1)} \le \lambda_{\max}^{(2)} \le \ldots \le \lambda_{\max}^{(\lfloor r/2 \rfloor)}. \tag{3}$$

Remark: We do not know whether similar inequalities hold for $s > \frac{r}{2}$.

Proof: Let T_s be the diagonal matrix of degrees in $G^{(s)}$ and $R^{(s)}(f)$ be the Rayleigh quotient of $\mathcal{L}^{(s)}$. It suffices to show $\lambda_1^{(s)} \le \lambda_1^{(s-1)}$ for $2 \le s \le r/2$. The eigenvalue $\lambda_1^{(s)}$ can be defined via the Rayleigh quotient $\lambda_1^{(s)} = \inf_{f \perp T_s \mathbf{1}} R^{(s)}(f)$ where $R^{(s)}(f) = \frac{\sum_{x \sim y}(f(x)-f(y))^2 w(x,y)}{\sum_x f(x)^2 d_x}$.

Pick a function $f : V^{\underline{(s-1)}} \to R$ such that $\langle f, T_{s-1}\mathbf{1}\rangle = 0$ and $\lambda_1^{(s-1)} = R^{(s-1)}(f)$. We define $g : \mathrm{V}^{\underline{s}} \to R$ as follows

$$g(x) = f(x'),$$

where x' is a $(s-1)$-tuple consisting of the first $(s-1)$ coordinates of x with the same order in x. Applying equation 1, we get

$$\langle g, T_s\mathbf{1}\rangle = \sum_{x \in \mathrm{V}^{\underline{s}}} d_x^{(s)} g(x) = \sum_{x \in \mathrm{V}^{\underline{s}}} g(x) d_{[x]} \binom{r-s}{s} s!.$$

We have

$$\begin{aligned}
\sum_x g(x) d_{[x]} &= \sum_x \sum_{F:[x]\subseteq F} g(x) \\
&= \sum_{x'} \sum_{F:[x']\subseteq F} (r-s+1) f(x') \\
&= \sum_{x'} d_{[x']}(r-s+1) f(x') \\
&= \frac{r-s+1}{\binom{r-s+1}{s-1}(s-1)!} \sum_{x'} f(x') d_{x'}^{(s-1)} = 0.
\end{aligned}$$

Here the second last equality follows from equation (1) and the last one follows from the choice of f. Therefore,

$$\sum_x g(x)d_x^{(s)} = (r-s+2)(r-s+1)\sum_{x'} f(x')d_{x'}^{(s-1)}.$$

Thus $\langle g, T_s\mathbf{1}\rangle = 0$. Similarly, we have

$$\sum_x g(x)^2 d_x^{(s)} = (r-s+2)(r-s+1)\sum_{x'} f(x')^2 d_{x'}^{(s-1)}.$$

Putting them together, we obtain

$$\sum_x g(x)^2 d_x^{(s)} = (r-s+2)(r-s+1)\sum_{x'} f(x')^2 d_{x'}^{(s-1)}.$$

By the similar counting method, we have

$$\begin{aligned}\sum_{x\sim y}(g(x)-g(y))^2 w(x,y) &= \sum_{x\sim y}\sum_{F:[x]\sqcup[y]\subseteq F}(g(x)-g(y))^2\\ &= \sum_{x'\sim y'}\sum_{F:[x']\sqcup[y']\subseteq F}(r-s+1)(r-s+2)(f(x')-f(y'))^2\\ &= (r-s+1)(r-s+2)\sum_{x'\sim y'}(f(x')-f(y'))^2 w(x',y').\end{aligned}$$

Thus, $R^{(s)}(g) = R^{(s-1)}(f) = \lambda_1^{(s-1)}$ by the choice of f. As $\lambda_1^{(s)}$ is the infimum over all g, we get $\lambda_1^{(s)} \le \lambda_1^{(s-1)}$.

The inequality (3) can be proved in a similarly way. Since $\lambda_{\max}^{(s)}$ is the supremum of the Rayleigh quotient, the direction of inequalities are reversed. □

Lemma 3. *For $r/2 < s \le r-1$, we have the following facts.*

1. *The s-th Laplacian has $\binom{n}{s}s!$ eigenvalues and all of them are in $[0,2]$.*
2. *The number of 0 eigenvalues is the number of strongly connected components in $D^{(s)}$.*
3. *If 2 is an eigenvalue of $L^{(s)}$, then one of the s-connected components of H is bipartite.*

The proof is trivial and will be omitted.

4 Applications

We show some applications of Laplacians $\mathcal{L}^{(s)}$ of hypergraphs in this section. The missing proofs of theorems in this section can be found at the full version of this paper [22].

4.1 The Random s-Walks on Hypergraphs

For $0 \leq \alpha < 1$ and $1 \leq s \leq r/2$, after restricting an α-lazy random s-walk on a hypergraph H to its stops (see section 2), we get an α-lazy random walk on the corresponding weighted graph $G^{(s)}$. Let $\pi(x) = d_x/\mathrm{vol}(\mathrm{V}^{\underline{s}})$ for any $x \in \mathrm{V}^{\underline{s}}$, where d_x is the degree of x in $G^{(s)}$ and $\mathrm{vol}(\mathrm{V}^{\underline{s}})$ is the volume of $G^{(s)}$. We have the following theorem.

Theorem 1. *For $1 \leq s \leq r/2$, suppose that H is an s-connected r-uniform hypergraph H and $\lambda_1^{(s)}$ (and $\lambda_{\max}^{(s)}$) is the first non-trivial (and the last) eigenvalue of the s-th Laplacian of H. For $0 \leq \alpha < 1$, the joint distribution f_k at the k-th stop of the α-lazy random walk at time k converges to the stationary distribution π in probability. In particular, we have*

$$\|(f_k - \pi)T^{-1/2}\| \leq (\bar{\lambda}_\alpha^{(s)})^k \|(f_0 - \pi)T^{-1/2}\|,$$

where $\bar{\lambda}_\alpha^{(s)} = \max\{|1-(1-\alpha)\lambda_1^{(s)}|, |(1-\alpha)\lambda_{\max}^{(s)} - 1|$, and f_0 is the probability distribution at the initial stop.

For $0 < \alpha < 1$ and $r/2 < s \leq r-1$, when restricting an α-lazy random s-walk on a hypergraph H to its stops (see section 2), we get an α-lazy random walk on the corresponding directed graph $D^{(s)}$. Let $\pi(x) = d_x/\mathrm{vol}(\mathrm{V}^{\underline{s}})$ for any $x \in \mathrm{V}^{\underline{s}}$, where d_x is the degree of x in $D^{(s)}$ and $\mathrm{vol}(\mathrm{V}^{\underline{s}})$ is the volume of $D^{(s)}$. We have the following theorem.

Theorem 2. *For $r/2 < s \leq r-1$, suppose that H is an s-connected r-uniform hypergraph and $\lambda_1^{(s)}$ is the first non-trivial eigenvalue of the s-th Laplacian of H. For $0 < \alpha < 1$, the joint distribution f_k at the k-th stop of the α-lazy random walk at time k converges to the stationary distribution π in probability. In particular, we have*

$$\|(f_k - \pi)T^{-1/2}\| \leq (\sigma_\alpha^{(s)})^k \|(f_0 - \pi)T^{-1/2}\|,$$

where $\sigma_\alpha^{(s)} \leq \sqrt{1 - 2\alpha(1-\alpha)\lambda_1^{(s)}}$, and f_0 is the probability distribution at the initial stop.

Remark: The reason why we require $0 < \alpha < 1$ in the case $r/2 < s \leq r-1$ is $\sigma_0(D^{(s)}) = 1$ for $r/2 < s \leq r-1$.

4.2 The s-Distances and s-Diameters in Hypergraphs

Let H be an r-uniform hypergraph. For $1 \leq s \leq r-1$ and $x, y \in \mathrm{V}^{\underline{s}}$, the s-distance $d^{(s)}(x,y)$ is the minimum integer k such that there is an s-path of length k starting at x and ending at y. For $X, Y \subseteq \mathrm{V}^{\underline{s}}$, let $d^{(s)}(X,Y) = \min\{d^{(s)}(x,y) \mid x \in X, y \in Y\}$. If H is s-connected, then the s-diameter $\mathrm{diam}^{(s)}(H)$ satisfies

$$\mathrm{diam}^{(s)}(H) = \max_{x,y \in \mathrm{V}^{\underline{s}}} \{d^{(s)}(x,y)\}.$$

For $1 \leq s \leq \frac{r}{2}$, the s-distances in H (and the s-diameter of H) are simply the graph distances in $G^{(s)}$ (and the diameter of $G^{(s)}$), respectively. We have the following theorems.

Theorem 3. *Suppose H is an r-uniform hypergraph. For integer s such that $1 \leq s \leq \frac{r}{2}$, let $\lambda_1^{(s)}$ (and $\lambda_{\max}^{(s)}$) be the first non-trivial (and the last) eigenvalue of the s-th Laplacian of H. Suppose $\lambda_{\max}^{(s)} > \lambda_1^{(s)} > 0$. For $X, Y \subseteq \mathrm{V}^{\underline{s}}$, if $d^{(s)}(X,Y) \geq 2$, then we have*

$$d^{(s)}(X,Y) \leq \left\lceil \frac{\log \sqrt{\frac{\mathrm{vol}(\bar{X})\mathrm{vol}(\bar{Y})}{\mathrm{vol}(X)\mathrm{vol}(Y)}}}{\log \frac{\lambda_{\max}^{(s)}+\lambda_1^{(s)}}{\lambda_{\max}^{(s)}-\lambda_1^{(s)}}} \right\rceil.$$

Here $\mathrm{vol}(*)$ *are volumes in* $G^{(s)}$.

Remark: We know $\lambda_1^{(s)} > 0$ if and only if H is s-connected. The condition $\lambda_{\max}^{(s)} > \lambda_1^{(s)}$ holds unless $s = 1$ and every pair of vertices is covered by edges evenly (i.e., H is a 2-design).

Theorem 4. *Suppose H is an r-uniform hypergraph. For integer s such that $1 \leq s \leq \frac{r}{2}$, let $\lambda_1^{(s)}$ (and $\lambda_{\max}^{(s)}$) be the first non-trivial (and the last) eigenvalue of the s-th Laplacian of H. If $\lambda_{\max}^{(s)} > \lambda_1^{(s)} > 0$, then the s-diameter of an r-uniform hypergraph H satisfies*

$$diam^{(s)}(H) \leq \left\lceil \frac{\log \frac{\mathrm{vol}(\mathrm{V}^{\underline{s}})}{\delta^{(s)}}}{\log \frac{\lambda_{\max}^{(s)}+\lambda_1^{(s)}}{\lambda_{\max}^{(s)}-\lambda_1^{(s)}}} \right\rceil.$$

Here $\mathrm{vol}(\mathrm{V}^{\underline{s}}) = \sum_{x \in \mathrm{V}^{\underline{s}}} d_x = |E(H)| \frac{r!}{(r-2s)!}$ *and* $\delta^{(s)}$ *is the minimum degree in* $G^{(s)}$.

When $r/2 < s \leq r-1$, the s-distances in H (and the s-diameter of H) is the directed distance in $D^{(s)}$ (and the diameter of $D^{(s)}$), respectively. We have the following theorems.

Theorem 5. *Let H be an r-uniform hypergraph. For $r/2 < s \leq r-1$ and $X, Y \subseteq \mathrm{V}^{\underline{s}}$, if H is s-connected, then we have*

$$d^{(s)}(X,Y) \leq \left\lfloor \frac{\log \frac{\mathrm{vol}(\bar{X})\mathrm{vol}(\bar{Y})}{\mathrm{vol}(X)\mathrm{vol}(Y)}}{\log \frac{2}{2-\lambda_1^{(s)}}} \right\rfloor + 1.$$

Here $\lambda_1^{(s)}$ *is the first non-trivial eigenvalue of the Laplacian of* $D^{(s)}$, *and* $\mathrm{vol}(*)$ *are volumes in* $D^{(s)}$.

Theorem 6. *For $r/2 < s \leq r-1$, suppose that an r-uniform hypergraph H is s-connected. Let $\lambda_1^{(s)}$ be the smallest nonzero eigenvalue of the Laplacian of $D^{(s)}$. The s-diameter of H satisfies*

$$diam^{(s)}(H) \leq \left\lceil \frac{2\log \frac{\mathrm{vol}(\mathrm{V}^{\underline{s}})}{\delta^{(s)}}}{\log \frac{2}{2-\lambda_1^{(s)}}} \right\rceil.$$

Here $\mathrm{vol}(\mathrm{V}^{\underline{s}}) = \sum_{x \in \mathrm{V}^{\underline{s}}} d_x = |E(H)| r!$ *and* $\delta^{(s)}$ *is the minimum degree in* $D^{(s)}$.

4.3 The Edge Expansions in Hypergraphs

In this subsection, we prove some results on the edge expansions in hypergraphs. Note that there was some attempt to generalize the edge discrepancy theorem from graphs to hypergraphs [3].

Let H be an r-uniform hypergraph. For $S \subseteq \binom{V}{s}$, we recall that the volume of S satisfies $\mathrm{vol}(S) = \sum_{x\in S} d_x$. Here d_x is the degree of the set x in H. In particular, we have

$$\mathrm{vol}\left(\binom{V}{s}\right) = |E(H)|\binom{r}{s}.$$

The *density* $e(S)$ of S is $\frac{\mathrm{vol}(S)}{\mathrm{vol}(\binom{V}{s})}$. Let $\bar{S}$ be the complement of S in $\binom{V}{s}$. We have $e(\bar{S}) = 1 - e(S)$.

For $1 \le t \le s \le r - t$, $S \subseteq \binom{V}{s}$, and $T \subseteq \binom{V}{t}$, let

$$E(S,T) = \{F \in E(H) : \exists x \in S, \exists y \in T, x \cap y = \emptyset, \text{ and } x \cup y \subseteq F\}.$$

Note that $|E(S,T)|$ counts the number of edges contains $x \sqcup y$ for some $x \in S$ and $y \in T$. Particularly, we have $\left|E\left(\binom{V}{s}, \binom{V}{t}\right)\right| = |E(H)|\frac{r!}{s!t!(r-s-t)!}$.

Theorem 7. *For $1 \le t \le s \le \frac{r}{2}$, $S \subseteq \binom{V}{s}$, and $T \subseteq \binom{V}{t}$, let $e(S,T) = \frac{|E(S,T)|}{|E(\binom{V}{s},\binom{V}{t})|}$. We have*

$$|e(S,T) - e(S)e(T)| \le \bar{\lambda}^{(s)}\sqrt{e(S)e(T)e(\bar{S})e(\bar{T})}. \tag{4}$$

Now we consider the case that $s > \frac{r}{2}$. Due to the fact $\sigma_0^{(s)} = 1$, we have to use the weaker expansion theorem 7 in [22]. Note that $\left|E\left(\binom{V}{s}, \binom{V}{t}\right)\right| = |E(H)|\frac{r!}{(r-s-t)!s!t!}$. We get the following theorem.

Theorem 8. *For $1 \le t < \frac{r}{2} < s < s + t \le r$, $S \subseteq \binom{V}{s}$, and $T \subseteq \binom{V}{t}$, let $e(S,T) = \frac{|E(S,T)|}{|E(\binom{V}{s},\binom{V}{t})|}$. If $|x \cap y| \ne \min\{t, 2s - r\}$ for any $x \in S$ and $y \in T$, then we have*

$$|\frac{1}{2}e(S,T) - e(S)e(T)| \le \bar{\lambda}^{(s)}\sqrt{e(S)e(T)e(\bar{S})e(\bar{T})}. \tag{5}$$

Nevertheless, we have the following strong edge expansion theorem for $\frac{r}{2} < s \le r - 1$. For $S, T \subseteq \binom{V}{s}$, let $E'(S,T)$ be the set of edges of the form $x \cup y$ for some $x \in S$ and $y \in T$. Namely,

$$E'(S,T) = \{F \in E(H) \mid \exists x \in S, \exists y \in T, F = x \cup y\}.$$

Observe that

$$\left|E'\left(\binom{V}{s}, \binom{V}{s}\right)\right| = |E(H)|\frac{r!}{(r-s)!(2s-r)!(r-s)!}.$$

Theorem 9. *For $\frac{r}{2} < s \le r-1$ and $S, T \subseteq \binom{V}{s}$, let $e'(S,T) = \frac{|E'(S,T)|}{|E'(\binom{V}{s},\binom{V}{s})|}$. We have*

$$|e'(S,T) - e(S)e(T)| \le \bar{\lambda}^{(s)} \sqrt{e(S)e(T)e(\bar{S})e(\bar{T})}. \tag{6}$$

5 Concluding Remarks

In this paper, we introduced a set of Laplacians for r-uniform hypergraphs. For $1 \le s \le r-1$, the s-Laplacian $\mathcal{L}^{(s)}$ is derived from the random s-walks on hypergraphs. For $1 \le s \le \frac{r}{2}$, the s-th Laplacian $\mathcal{L}^{(s)}$ is defined to be the Laplacian of the corresponding weighted graph $G^{(s)}$. The first Laplacian $\mathcal{L}^{(1)}$ is exactly the Laplacian introduced by Rodrìguez [25].

For $\frac{r}{2} \le s \le r-1$, the s-th Laplacian $\mathcal{L}^{(s)}$ is defined to be the Laplacian of the corresponding Eulerian directed graph $D^{(s)}$. At first glimpse, $\sigma_0(D^{(s)})$ might be a good parameter. However, it is not hard to show that $\sigma_0(D^{(s)}) = 1$ always holds. Our work is based on (with some improvements) Chung's recent work [11, 12] on directed graphs.

Let us recall Chung's definition of Laplacians [8] for regular hypergraphs. An r-uniform hypergraph H is d-regular if $d_x = d$ for every $x \in \mathrm{V}^{\underline{r-1}}$. Let G be a graph on the vertex set $\mathrm{V}^{\underline{r-1}}$. For $x, y \in \mathrm{V}^{\underline{r-1}}$, let xy be an edge if $x = x_1x_2, \ldots, x_{r-1}$ and $y = y_1x_2, \ldots, x_{r-1}$ such that $\{x_1, y_1, x_2, \ldots, x_{r-1}\}$ is an edge of H. Let A be the adjacency matrix of G, T be the diagonal matrix of degrees in G, and K be the adjacency matrix of the complete graph on the edge set $\mathrm{V}^{\underline{r-1}}$. Chung [8] defined the Laplacian $\mathcal{L}$ such that $\mathcal{L} = T - A + \frac{d}{n}(K + (r-1)I)$.

This definition comes from the homology theory of hypergraphs. Firstly, $\mathcal{L}$ is not normalized in Chung's definition, i.e., the eigenvalues are not in the interval $[0, 2]$. Secondly, the add-on term $\frac{d}{n}(K+(r-1)I)$ is not related to the structures of H. If we ignore the add-on term and normalize the matrix, then we essentially get the Laplacian of the graph G. Note G is disconnected, then $\lambda_1(G) = 0$ and it is not interesting. Thus Chung added the additional term. The graph G is actually very closed to our Eulerian directed graph $D^{(r-1)}$. Let B be the adjacency matrix of $D^{(r-1)}$. In fact we have $B = QA$, where Q is a rotation which maps $x = x_1, x_2 \ldots, x_{r-1}$ to $x' = x_2 \ldots, x_{r-1}, x_1$. Since $d_x = d_{x'}$, Q and T commute, we have

$$\begin{aligned}
(T^{-1/2}BT^{-1/2})'(T^{-1/2}BT^{-1/2}) &= T^{-1/2}B'T^{-1}BT^{-1/2} \\
&= T^{-1/2}A'Q'T^{-1}QAT^{-1/2} \\
&= T^{-1/2}A'T^{-1}Q'QAT^{-1/2} \\
&= T^{-1/2}A'T^{-1}AT^{-1/2}.
\end{aligned}$$

Here we use the fact $Q'Q = I$. This identity means that the singular values of $I - \mathcal{L}^{(r-1)}$ is precisely equal to 1 minus the Laplacian eigenvalues of the graph G.

Our definitions of Laplacians $\mathcal{L}^{(s)}$ seem to be related to the quasi-randomness [7, 17] of hypergraphs. We are very interested in this direction. Many concepts such as the s-walk, the s-path, the s-distance, and the s-diameter, have their independent interest.

References

[1] Aldous, D., Fill, J.: Reversible Markov chains and random walks on graphs (in preparation)
[2] Alon, N.: Eigenvalues and expanders. Combinatorica 6, 86–96 (1986)
[3] Butler, S.: A new discrepancy definition for hypergraphs (unpublished)
[4] Dudek, A., Frieze, A.M.: Loose Hamilton cycles in random uniform hypergraphs. Electronic Journal of Combinatorics 18-1, P48 (2011)
[5] Dudek, A., Frieze, A.M.: Tight Hamilton cycles in random uniform hypergraphs (submitted)
[6] Chung, F.: Diameters and eigenvalues. J. of the Amer. Math. Soc. 2, 187–196 (1989)
[7] Chung, F., Graham, R.L.: Quasi-random hypergraphs. Random Structure and Algorithms 1-1, 105–124 (1990)
[8] Chung, F.: The Laplacian of a hypergraph. In: Friedman, J. (ed.) Expanding graphs DIMACS series, pp. 21–36 (1993)
[9] Chung, F., Faber, V., Manteuffel, T.A.: An upper bound on the diameter of a graph from eigenvalues associated with its Laplacian. Siam. J. Disc. Math. 7-3, 443–457 (1994)
[10] Chung, F.: Spectral graph theory. AMS Publications. Providence (1997)
[11] Chung, F.: Laplacians and the Cheeger inequality for directed graphs. Annals of Comb. 9, 1–19 (2005)
[12] Chung, F.: The diameter and Laplacian eigenvalues of directed graphs. Electronic Journal of Combinatorics 13(4) (2006)
[13] Frieze, A.M.: Loose Hamilton cycles in random 3-uniform hypergraphs. Electronic Journal of Combinatorics 17(28) (2010)
[14] Hán, H., Schacht, M.: 3 Dirac-type results for loose Hamilton cycles in uniform hypergraphs. J. Comb. Theory Ser. B 100, 332–346 (2010)
[15] Katona, G.Y., Kierstead, H.A.: Hamiltonian chains in hypergraphs. J. of Graph Theory 30-3, 205–212 (1999)
[16] Keevash, P., Kühn, D., Mycroft, R., Osthus, D.: Loose Hamilton cycles in hypergraphs (submitted)
[17] Kohayakawa, Y., Rödl, V., Skokan, J.: Hypergraphs, quasi-randomness, and conditions for regularity. J. Combin. Theory Ser. A 97-2, 307–352 (2002)
[18] Kühn, D., Mycroft, R., Osthus, D.: Hamilton l-cycles in k-graphs (submitted)
[19] Kühn, D., Osthus, D.: Loose Hamilton cycles in 3-uniform hypergraphs of high minimum degree. J. Combin. Theory Ser. B 96-6, 767–821 (2006)
[20] Lawler, G.F., Sokal, A.D.: Bounds on the L^2 spectrum for Markov chains and Markov processes: a generalization of Cheeger's inequality. Transactions of the American Mathematical Society 309, 557–580 (1988)
[21] Mihail, M.: Conductance and convergence of markov chainsa combinatorial treatment of expanders. In: Proc. of 30th FOCS, pp. 526–531 (1989)
[22] Lu, L., Peng, X.: High-ordered Random Walks and Generalized Laplacians on Hypergraphs (full version), http://arxiv.org/abs/1102.4409
[23] Rödl, V., Ruciński, A., Szemerédi, E.: A Dirac-type theorem for 3-uniform hypergraphs. Combin. Probab. Comput. 15(1-2), 229–251 (2006)
[24] Rödl, V., Ruciński, A., Szemerédi, E.: An approximate Dirac-type theorem for k-uniform hypergraphs. Combinatorica 28-2, 229–260 (2008)
[25] Rodrìguez, J.A.: Laplacian eigenvalues and partition problems in hypergraphs. Applied Mathematics Letters 22, 916–921 (2009)

Detecting the Structure of Social Networks Using (α, β)-Communities*,**

Jing He[2], John Hopcroft[1], Hongyu Liang[2],
Supasorn Suwajanakorn[1], and Liaoruo Wang[1]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853
{jeh17,ss932,lw335}@cornell.edu
[2] Institute for Theoretical Computer Science, Tsinghua University, Beijing, China
{hejing2929,hongyuliang86}@gmail.com

Abstract. An (α, β)-*community* is a subset of vertices C with each vertex in C connected to at least β vertices of C (self-loops counted) and each vertex outside of C connected to at most α vertices of C ($\alpha < \beta$) [9]. In this paper, we present a heuristic (α, β)-Community algorithm, which in practice successfully finds (α, β)-communities of a given size. The structure of (α, β)-communities in several large-scale social graphs is explored, and a surprising core structure is discovered by taking the intersection of a group of massively overlapping (α, β)-communities. For large community size k, the (α, β)-communities are well clustered into a small number of disjoint cores, and there are no isolated (α, β)-communities scattered between these densely-clustered cores. The (α, β)-communities from the same group have significant overlap among them, and those from distinct groups have extremely small pairwise resemblance. The number of cores decreases as k increases, and there are no bridges of intermediate (α, β)-communities connecting one core to another. The cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. Further, similar experiments on random graph models demonstrate that the core structure displayed in various social graphs is due to the underlying social structure of these real-world networks, rather than due to high-degree vertices or a particular degree distribution.

1 Introduction

Much of the early work on finding communities in social networks focused on partitioning the corresponding graph into disjoint communities [3, 5, 8, 10–14]. Algorithms often required dense graphs and conductance was taken as the measure of the goodness of a community [4, 7, 8, 15]. To identify well-defined communities in social networks, one needs to realize that an individual may belong to

* Authors are listed alphabetically.

** This research was partially supported by the U.S. Air Force Office of Scientific Research under Grant FA9550-09-1-0675, the National Natural Science Foundation of China under Grant 60553001, and the National Basic Research Program of China under Grant 2007CB807900 and 2007CB807901.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 26–37, 2011.

multiple communities at the same time and is likely to have more connections to individuals outside of his/her community than inside. For example, a person in the theoretical computer science community is likely to have many connections to individuals outside of the theoretical computer science community, who may be his/her family members, or enroll in his/her institution, or attend his/her religious group. One approach to finding such overlapping communities is that of Mishra et al. [9], where the concept of an (α, β)-community was introduced and several algorithms were given for finding an (α, β)-community in a dense graph provided there is an advocate for the community. An advocate for a community is an individual who is connected to a large fraction of the members of that community.

In this paper, we discuss the concept of (α, β)-community, and develop a heuristic (α, β)-Community algorithm that in practice efficiently finds (α, β)-communities of a given size. Further, we thoroughly explore the structure of (α, β)-communities in several large-scale social networks. Surprisingly, in a Twitter friendship graph with 112,957 vertices and 481,591 edges, there are 6,912 distinct (α, β)-communities of size 200 among the 45,361 (α, β)-communities returned by the algorithm. Moreover, these (α, β)-communities are neatly categorized into a small number of massively overlapping clusters. Specifically, the (α, β)-communities from the same cluster have significant overlap ($> 90\%$) among them, while the (α, β)-communities from distinct clusters have extremely small ($< 5\%$) pairwise resemblance. This leads to the notion of a core which is the intersection of a group of massively overlapping (α, β)-communities, where the core also shares a significant overlap ($> 75\%$) with every member (α, β)-community in that group.

The total number and average size of cores in the Twitter graph as functions of the community size k are given in Table 1. Interestingly, as the size k increases, some cores merge into larger ones while others simply disappear. Moreover, cores may fracture when they merge into larger ones, with a fraction of vertices disappearing from larger cores and reappearing later. Among the interesting questions we explore in this paper are why (α, β)-communities correspond to well-defined clusters and why there is no bridge of (α, β)-communities connecting one cluster to another. A bridge is a sequence of intermediate (α, β)-communities where adjacent subsets of the sequence have substantial overlap but the first and last subsets have little overlap. Other intriguing questions include whether different types of social networks incorporate fundamentally different social structures, and what it is about the structure of social networks that leads to the structure of cores as in the Twitter graph and why some networks do not display this structure as in random graph models.

By taking the intersection of a group of massively overlapping (α, β)-Communities obtained from repeated experiments, we can eliminate the random factors and extract the underlying structure with multiple runs of the (α, β)-Community algorithm. In social graphs, for large community size k, the (α, β)-communities are well clustered into a small number of disjoint cores, and there are no isolated (α, β)-communities scattered between these densely-clustered

cores. The number of cores decreases as k increases and becomes relatively small for large k. The cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. Moreover, the cores correspond to dense regions of the graph, and there are no bridges of intermediate (α, β)-communities connecting one core to another. In contrast, the cores found in several random graph models usually have significant overlap among them, and the number of cores does not necessarily decrease as k increases. Extensive experiments demonstrate that the core structure displayed in various large-scale social graphs is indeed due to the underlying social structure of the networks, rather than due to high-degree vertices or a particular degree distribution.

The rest of this paper is organized as follows. First, we introduce the definition of an (α, β)-community in Section 2 and show their frequent existence. Then, we prove the NP-hardness of finding an (α, β)-community and present the heuristic (α, β)-Community algorithm. In Section 3, we apply the algorithm to various large-scale social graphs and random graphs to explore, analyze, and demonstrate the core structure in social networks. We conclude in Section 4 with comments on the problems considered and future work.

2 Preliminaries

The concept of (α, β)-community was proposed by Mishra et al. [9] as a powerful tool for graph clustering and community discovery. In [9], an (α, β)-community refers to a cluster of vertices with each vertex in the cluster adjacent to at least a β-fraction of the vertices in the cluster and each vertex outside of the cluster adjacent to at most an α-fraction of the vertices in the cluster. In this paper, we adopt a slightly different definition. Given a graph $G = (V, E)$ with self-loops added to all vertices, a subset $C \subseteq V$ is called an (α, β)-*community* when each vertex in C is connected to at least β vertices of C (self-loop counted) and each vertex outside of C is connected to at most α vertices of C ($\alpha < \beta$). Similarly to that of [9], this definition acknowledges the importance of self-loops: although a maximal clique should intuitively be a community, this cannot be guaranteed without self-loops. An (α, β)-community in a graph G is called *proper* if it corresponds to a non-empty proper subgraph of G.

Given a subset $S \subseteq V$, for a vertex $v \notin S$, $\alpha(v)$ is defined as the number of edges between v and vertices of S. Similarly, for a vertex $w \in S$, $\beta(w)$ is defined as the number of edges between w and vertices of S (self-loop counted). Then, $\alpha(S) = \max\{\alpha(v) | v \notin S\}$ and $\beta(S) = \min\{\beta(w) | w \in S\}$.

A maximal clique is guaranteed to be an (α, β)-community since self-loops are counted by the definition. Every graph that is not a clique must contain an (α, β)-community (or, a maximal clique) as a proper subgraph. Starting with any vertex, it is either a proper (α, β)-community (i.e. a maximal clique) or there must be another vertex connected to it (i.e. not a maximal clique). Then, a pair of two vertices connected by an edge is either a proper (α, β)-community (i.e. a maximal clique) or there must be a third vertex connected to both (i.e. not a maximal clique). Continue this argument until a proper (α, β)-community is

found or all vertices are included in a clique, contradicting the assumption that the graph is not a clique. Thus, we have the following theorem:

Theorem 1. *Every graph other than a clique contains a proper* (α, β)*-community.*

Proof. Omitted due to space limitations (see [6]).

Algorithm 1. (α, β)-COMMUNITY$(G = (V, E), k)$

$S \leftarrow$ a random subset of V of k vertices
while $\beta(S) \leqslant \alpha(S)$ **do**
  $S \leftarrow$ SWAPPING(G, S)
  $A \leftarrow \{v \notin S \mid \alpha(v) = \alpha(S)\}$
  $B \leftarrow \{v \in S \mid \beta(v) = \beta(S)\}$
  if $\{(a_i, b_j) \notin E \mid a_i \in A, b_j \in B\} \neq \emptyset$ **then**
    pick such a pair of vertices (a_i, b_j)
    $S \leftarrow (S - \{b_j\}) \cup \{a_i\}$
  else if $\{a_i \in A \mid (a_i, a_k) \notin E, \forall a_k \in A, k \neq i\} \neq \emptyset$ **then**
    pick such a vertex a_i
    $S \leftarrow S \cup \{a_i\}$
  else if $\{b_j \in B \mid (b_j, b_k) \notin E, \forall b_k \in B, k \neq j\} \neq \emptyset$ **then**
    pick such a vertex b_j
    $S \leftarrow S - \{b_j\}$
  else
    $S \leftarrow S \cup A$
  end if
end while
return S

Algorithm 2. SWAPPING$(G = (V, E), S)$

while $\beta(S) < \alpha(S)$ **do**
  $A \leftarrow \{v \notin S \mid \alpha(v) = \alpha(S)\}$
  $B \leftarrow \{v \in S \mid \beta(v) = \beta(S)\}$
  pick a vertex $a \in A$ and a vertex $b \in B$
  $S \leftarrow (S - \{b\}) \cup \{a\}$
end while
return S

Given a graph G with a self-loop added to each vertex and an integer k, define COMMUNITY as the problem of finding an (α, β)-community of size k in G. Given a graph G and an integer k, define CLIQUE as the problem of determining whether there exists a clique of size k in G.

Theorem 2. *The* COMMUNITY *problem is* NP*-hard.*

Proof. Omitted due to space limitations (see [6]).

Next, we give a heuristic algorithm for finding an (α, β)-community of size k in a graph $G = (V, E)$. A mathematical description of this (α, β)-COMMUNITY algorithm, along with a subroutine called SWAPPING, is given above. See [6] for more details.

3 Experimental Results

3.1 Social Graphs

Twitter. The Twitter dataset [1, 2] corresponds to a directed friendship graph among a subset of Twitter user accounts. Each vertex represents an individual Twitter user account, and each edge represents a following relation from one user to another. For simplicity, we convert this directed graph into an undirected graph, ignoring the direction of the edges and combining multiple edges with the same pair of endpoints. Further, we iteratively remove from the graph the isolated and degree-one vertices in order to get rid of the insignificant outliers. This effectively reduces the number of vertices and edges, resulting in a smaller graph with 112,957 vertices and 481,591 edges. Then, the average degree of the Twitter graph is 8.52. Finally, self-loops are added to this graph in accordance with the definition of (α, β)-community.

For a given size k, the heuristic (α, β)-COMMUNITY algorithm is applied to the Twitter graph for finding (α, β)-communities starting with a number of (e.g. 500) random subsets of size k. Theoretically, the algorithm is not guaranteed to terminate within a reasonable period of running time, thus we specify an upper bound (e.g. 1,000) on the number of iterations the algorithm can execute. However, from what we have observed in the experiments, the case of not finding any (α, β)-community within 1,000 iterations is extremely rare. In other words, 500 (α, β)-communities are obtained most of the time with 500 runs of the algorithm.

(a) Core (b) Chain

Fig. 1. The overlapping structure

To shed a light on how many (α, β)-communities there are in the Twitter graph, 45,361 runs of the algorithm are performed for $k = 200$ and 6,912 distinct (α, β)-communities are obtained. Surprisingly, many (α, β)-communities are observed to massively overlap with each other and differ only by a few vertices. Moreover, such a great number of (α, β)-communities are all clustered into

Table 1. Cores of the Twitter graph

k	25	50	100	150	200	250	300	350	400	450	500
number of cores	221	94	19	9	4	4	4	3	3	3	3
average core size	23	45	73	112	151	216	276	332	364	402	440

a small number of disjoint groups. Specifically, every pair of (α, β)-communities from the same group shares a resemblance higher than 0.9, while every pair of (α, β)-communities from distinct groups shares a resemblance lower than 0.06. Here, the pairwise resemblance $r(A, B)$ between two sets A and B is defined as:

$$r(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

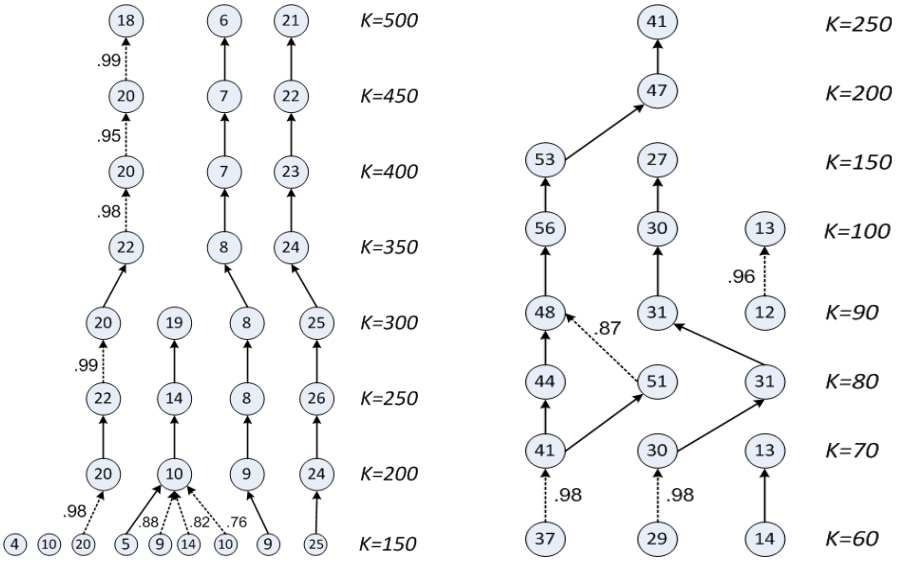
The overlapping (α, β)-communities form a "core" structure rather than a "chain" structure, as illustrated in Fig. 1. The intersection of all (α, β)-communities in each group bears an over 75% resemblance with every single (α, β)-community in that group. For $k = 200$, all 6,912 (α, β)-communities found by the 45,361 runs of the algorithm cluster into four "cores". The "cores" correspond to dense regions of the graph while being exclusively disjoint, and in contrast to what we would have expected, there are no isolated (α, β)-communities scattered between these densely-clustered "cores".

For a group of pairwise similar (α, β)-communities, we formally define the *core* to be the intersection of those (α, β)-communities. The number of cores can be determined by computing the resemblance matrix of all obtained (α, β)-communities. Intuitively, (α, β)-communities can be categorized according to the resemblance matrix in a way that every pair of (α, β)-communities in the same category is similar to each other, i.e. the pairwise resemblance is large. A pairwise resemblance is considered to be sufficiently large if it is greater than 0.6, while in practice we frequently observe resemblance greater than 0.9. Based on each category, a core is formed by taking the intersection of all member (α, β)-communities. Therefore, the number of cores is equal to that of such intersections, i.e. the number of blocks along the diagonal of the resemblance matrix. The number and average size of cores in the Twitter graph as functions of the community size k are given in Table 1.

The number of cores decreases as the size k increases. This number is relatively small when k becomes large and will eventually decrease to one as k further increases, indicating that (α, β)-communities are well clustered into a small number of cores before gradually merging into one large core. For example, the (α, β)-communities are clustered into 9 cores for $k = 150$ and 4 cores for $k = 200$, where in both cases the cores are disjoint from each other. As the size k increases, the cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. A layered tree diagram is constructed to illustrate this phenomenon in Fig. 2(a).

Each level of the diagram, indexed by the size k, consists of cores extracted from collections of pairwise similar (α, β)-communities by taking their respective

intersections. For each pair of cores in adjacent levels, a directed edge is added from the lower level to the upper level if the fraction of overlap is significant, i.e. a substantial fraction (e.g. 60%) of vertices in the core of the lower level is contained in the core of the upper level. If the fraction of overlap is smaller than one, a dotted arrow with this fraction labeled is added to represent a partial merge. Otherwise, a solid arrow with the label "1" omitted is added to represent a full merge. As shown in Fig. 2(a), the fraction of overlap is close to one as we move up the levels, that is, a core of some lower level is (almost) entirely merged into a core of the next higher level.



(a) Twitter (b) Slashdot

Fig. 2. Tree diagrams indexed by the size k. (Each circle represents a core obtained for a given size, in which the integer denotes the β-value of the core. Each dotted arrow represents a partial merge with the fraction of overlap labeled, and each solid arrow represents a full merge).

The definition of (α, β)-community does not prevent a community from having more edges connecting it to the rest of the graph than those connecting within the community itself. Empirically, there are many more vertices outside of an (α, β)-community, and the edges connecting the community to the rest of the graph are almost always more than those connecting within itself. This definition gives an intuitive criterion as to whether to classify a subset of vertices as a community, i.e. the number of edges connecting each vertex in the community to vertices of the community should be strictly greater than that connecting any vertex outside of the community to vertices of the community. Moreover, by

taking the intersection of a number of massively overlapping (α, β)-communities, the set of (α, β)-communities which differ only by a few vertices is reduced to an underlying core. Thus, each (α, β)-community consists of one of a small number of cores and a few random peripheral vertices, and these peripheral vertices are what gives rise to such a large number of (α, β)-communities.

Before proceeding to our experiments on other social networks, we provide a detailed discussion on the core structure. There might be a generic bias in the (α, β)-COMMUNITY algorithm which is attracted to dense regions of the graph, and thus it is possible that (α, β)-communities located in sparse regions of the graph are never found by the algorithm.

A natural question is what causes the Twitter graph to display this core structure, and further, why the graph shows only a small number of disjoint cores for a large size k. As we will show later, this is due to the fact that a definite social structure, as opposed to randomness, exists in the Twitter network. To take a closer look into this, we simplify the Twitter graph by removing low-degree vertices, i.e. vertices of degree lower than 19, and then obtain a smaller graph with 4,144 vertices and 99,345 edges. The smallest β-value for most (α, β)-communities is given by 19, thus removing vertices of degree lower than 19 will get rid of insignificant low-degree vertices without destroying the fundamental structure of the graph. Again, the (α, β)-COMMUNITY algorithm is applied to this graph with minimum degree 19 for $k = 200, 250, 300, 350, 400$, and exactly two disjoint cores are obtained in each case. Between any two adjacent levels in the corresponding tree diagram, the two cores of the lower level are completely contained in those of the upper level. One possible reason for such a small number of cores could be that the vertices of the cores are more "powerful" in pulling other vertices toward them. If we remove the two cores from the graph and repeat the experiment for $k = 200$, the returned (α, β)-communities are no longer clustered and form a large number of scattered communities.

Another question is why there are exactly two distinct cores in the simplified Twitter graph. For instance, define C_1 and C_2 as the two cores obtained for $k = 200$. C_1 corresponds to a fairly dense subgraph with 156 vertices and 3,029 edges, where the minimum degree is 23 and the average degree is 38.8. C_2 has 159 vertices and 2,577 edges, where the minimum degree is 19 and the average degree is 32.4. Consider the bipartite graph with the two sets of vertices being the vertices of C_1 and the vertices of C_2. Surprisingly, there are only 105 cross edges between C_1 and C_2, where 110 (70%) vertices of C_1 and 100 (63%) vertices of C_2 are not associated with any cross edges. Thus, the cores C_1 and C_2 correspond to two subsets of vertices that are densely connected internally but sparsely connected with each other. As a result, they are returned by the algorithm as the cores of two groups of massively overlapping (α, β)-communities.

It is observed that, in addition to the merging of cores, some cores existing for a smaller k simply disappear from the tree diagram as k increases. In other words, few vertices of these disappearing cores are contained in the cores of the next higher level. The cores take on more vertices as the community size k increases, and there may be two cores taking on the same set of vertices. Thus, the

SWAPPING algorithm should run into one of the following two situations: either 1) most vertices of the two cores merge into a new core with some peripheral vertices discarded, or 2) most vertices of one core plus a small fraction of the other form a new core with the latter one disappearing. To verify that one of the above two cases happens, consider the two cores obtained for $k = 150$ that later disappear for $k = 200$, as shown in Fig. 2(a). Let C be one of the two disappearing cores, and recursively perform the following process: enlarge C by adding a random vertex $v \notin C$, run the (α, β)-COMMUNITY algorithm on this enlarged C to find an (α, β)-community of one size larger, and update C to be this obtained (α, β)-community. This process is repeated a number of times until the size of C is increased to 200. Empirically, any obtained (α, β)-community of size 200 contains only a small fraction of vertices of the initial core, while the initial core was completely contained in the (α, β)-communities of size up to about 170. A core may fracture when merging into some other core of a larger size. What happens is that, as vertices are added to one core A, they are also well connected to another core B. As k further increases, these vertices of A will be included in a larger core C that completely contains B, leading to the disappearance of A. If we continue to increase k, the vertices that have disappeared may reappear in a larger core that completely contains C, since they are well connected to the rest of that core.

A *bridge* between two (α, β)-communities or two cores S_1 and S_m is a sequence of intermediate (α, β)-communities $S_2, \cdots, S_{m-1}$, where the pairwise resemblance is large between adjacent subsets but small between the first and last subsets, i.e. $r(S_1, S_m) < 0.3$ and $r(S_i, S_{i+1}) > 0.6$ for all $i \in \{1, 2, \cdots, m-1\}$. The *length* of the bridge is thus given by $m - 1$. Recall that for $k = 200$, (α, β)-communities are all clustered into four disjoint cores, and there is little overlap between any two (α, β)-communities from distinct cores. It is possible that there exists a bridge in the Twitter graph, but the bias of our algorithm may prevent it from being found. Thus, although no bridge is detected in this experiment, a subsequent question is whether the graph contains any bridge between two cores at all.

Next, the following experiment is designed to determine whether there exists a bridge between two cores. Pick any two cores obtained for $k = 200$ and recursively perform the following steps: randomly choose r vertices from one core and $200 - r$ vertices from the other to form an initial subset of size 200, and apply the (α, β)-COMMUNITY algorithm to this subset. If every iteration returns an (α, β)-community that substantially overlaps with one core but is disjoint from the other, then it implies that there does not exist any bridge between the two cores. During 100 runs of the algorithm, 99 of them return such an (α, β)-community that significantly overlaps with one core but is disjoint from the other. Only one trial returns an (α, β)-community C that contains 95.54% of one core A and 26.22% of the other core B. However, no other intermediate (α, β)-communities can be found between B and C using the above method, which demonstrates the non-existence of a bridge.

Table 2. Cores of the Slashdot graph

k	30	40	50	60	70	80	90	100	150	200	250
number of cores	29	10	3	3	3	3	3	3	2	1	1
average core size	25	33	41	53	62	72	85	97	148	197	244

Another approach to finding a bridge is to search for (α, β)-communities that fall between cores. Generate random subsets of size 200 and run the (α, β)-Community algorithm recursively. As we have seen before, four disjoint cores are obtained with 500 runs of the algorithm, and for another 45,361 runs, the (α, β)-community obtained at the end of each iteration is compared with the four cores to check whether it is an intermediate (α, β)-community. This approach is also useful for estimating the total number of (α, β)-communities of a given size. Among the 45,361 runs, no intermediate (α, β)-communities are found, however, only 6,912 distinct (α, β)-communities are returned, which indicates a relatively small number of (α, β)-communities of size 200 and/or a generic bias of our algorithm towards some particular communities over others.

Overall, the above experiments have suggested that there is no bridge between cores, that is, there is not likely to exist a sequence of intermediate (α, β)-communities that connects two cores with substantial overlap between adjacent pairs. The non-existence of such a bridge demonstrates the underlying social structure of the Twitter network with (α, β)-communities neatly categorized into a few densely-clustered disjoint cores.

Slashdot. Slashdot is a technology-related news website known for its professional user community. The website features contemporary technology-oriented news submitted by users and evaluated by editors. In 2002, Slashdot introduced the Slashdot Zoo feature, which allows users to tag each other as friends or foes. The social network based on the common interest shared among Slashdot users was obtained and released by Leskovec et al. [8] in February 2009.

The Slashdot graph contains 82,168 vertices and 504,230 edges, with an average degree of 12.3. Similarly, the (α, β)-Community algorithm is applied to this dataset and the statistics are given in Table 2. The number of cores decreases as the community size k increases and becomes relatively small for large k, behaving the same as it did in the Twitter graph. The cores returned by the algorithm are almost disjoint from each other and correspond to dense regions of the graph, with few edges connecting the bipartite graph induced by the vertices of each pair of cores. This indicates that (α, β)-communities are well clustered into a small number of cores for large k, which correspond to dense regions of the graph and share little overlap among them. For example, the (α, β)-communities are clustered into three nearly disjoint cores for $k = 100$, where only 171 edges connect the two cores of size 93 and 100 that have 2,142 and 1,105 internal edges, respectively. Before eventually merging into one large core as k further increases, these densely-clustered cores emerge as the underlying social structure displayed

by the Slashdot network. It is also observed that, as k increases, the cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. A layered tree diagram is constructed to illustrate this phenomenon in the Slashdot graph, as shown in Fig. 2(b).

arXiv hep-ph Coauthor and Citation. Similar patterns of the core structure can also be observed in the arXiv hep-ph Coauthor and Citation graphs. See [6] for detailed experimental results.

3.2 Random Graphs

To demonstrate that the structure we have found in social graphs is not merely a random artifact, a similar set of experiments is carried out for random graphs. Random graph models do not produce clusters as social graphs do. The cores obtained by the (α, β)-Community algorithm usually have significant overlap among them, and correspond to dense regions due to the way the graph was generated. This contrast between social graphs and random graphs again verifies the existence of community structure in various large-scale social networks. See [6] for more details.

4 Conclusion

In social networks, the (α, β)-communities returned by the (α, β)-Community algorithm for a given size k are well clustered into a small number of disjoint cores, each of which is the intersection of a group of massively overlapping (α, β)-communities. Two (α, β)-communities from the same group share a significant overlap and differ by only a few vertices, while the pairwise resemblance of two (α, β)-communities from different groups is extremely small. The number of cores decreases as k increases and becomes relatively small for large k. The cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. Further, the cores correspond to dense regions of the graph. There are no isolated (α, β)-communities scattered between these densely-clustered cores, nor bridges of (α, β)-communities connecting one core to another. Various large-scale social graphs have been explored thoroughly, all of which display the core structure rather than the chain structure.

By constructing random graphs with a power-law degree distribution or the same degree distribution as the social graphs, it is demonstrated that neither high-degree vertices nor a particular degree distribution can result in the core structure displayed in large-scale social graphs. The cores found by the (α, β)-Community algorithm in random graphs usually have significant overlap among them and are increasingly scattered across the graph as the size k increases, which implies the non-existence of well-defined clusters in random graphs and verifies the existence of underlying structure in various social networks.

References

1. Choudhury, M.D., Lin, Y.-R., Sundaram, H., Candan, K., Xie, L., Kelliher, A.: How does the sampling strategy impact the discovery of information diffusion in social media? In: Proc. 4th Int'l AAAI Conf. Weblogs and Social Media, ICWSM (2010)
2. Choudhury, M.D., Sundaram, H., John, A., Seligmann, D.D., Kelliher, A.: Birds of a feather: does attribute homophily impact information diffusion on social media? (under review)
3. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70, 06111 (2004)
4. Gaertler, M.: Clustering. In: Brandes, U., Erlebach, T. (eds.) Network Analysis. LNCS, vol. 3418, pp. 178–215. Springer, Heidelberg (2005)
5. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99(12), 7821–7826 (2002)
6. He, J., Hopcroft, J.E., Liang, H., Supasorn, S., Wang, L.: Detecting the structure of social networks using (α, β)-communities. Tech. rep., Cornell University (2011), http://hdl.handle.net/1813/22415
7. Lang, K., Rao, S.: A flow-based method for improving the expansion or conductance of graph cuts. In: Bienstock, D., Nemhauser, G.L. (eds.) IPCO 2004. LNCS, vol. 3064, pp. 325–337. Springer, Heidelberg (2004)
8. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.: Statistical properties of community structure in large social and information networks. In: Proc. 18th Int'l World Wide Web Conf. WWW (2008)
9. Mishra, N., Schreiber, R., Stanton, I., Tarjan, R.E.: Finding strongly-knit clusters in social networks. Internet Mathematics 5(1-2), 155–174 (2009)
10. Newman, M.E.J.: Detecting community structure in networks. The European Physical J. B 38, 321–330 (2004)
11. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 066133 (2004)
12. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74, 036104 (2006)
13. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA 103(23), 8577–8582 (2006)
14. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004)
15. Schaeffer, S.E.: Graph clustering. Computer Science Review 1(1), 27–64 (2007)

Latent Clustering on Graphs with Multiple Edge Types*

Matthew Rocklin[1] and Ali Pinar[2]

[1] Department of Computer Science, University of Chicago
mrocklin@cs.uchicago.edu
[2] Sandia National Laboratories
apinar@sandia.gov

Abstract. We study clustering on graphs with multiple edge types. Our main motivation is that similarities between objects can be measured in many different metrics, and so allowing graphs with multivariate edges significantly increases modeling power. In this context the clustering problem becomes more challenging. Each edge/metric provides only partial information about the data; recovering full information requires aggregation of all the similarity metrics. We generalize the concept of clustering in single-edge graphs to multi-edged graphs and discuss how this generates a space of clusterings. We describe a meta-clustering structure on this space and propose methods to compactly represent the meta-clustering structure. Experimental results on real and synthetic data are presented.

1 Introduction

Graphs are widely recognized as the standard modeling language to represent relations between entities of a complex system. Entities in the data are represented as nodes while relationships between entities are represented as edges between nodes. For instance, an email network would have email accounts as nodes, and the email exchanges between two accounts form an edge between the two nodes. Proteins (nodes) are connected in a protein interaction network by an edge if the proteins are part of the same system function.

In many real-world problems, connections or similarities between entities can be defined by many different relationships, where connections/similarities are quantified by boolean (a connection exists or not), or continuous variables. For example, similarity between two scientific articles can be defined based on authors, citations to, citations from, keywords, titles, where they are published, text similarity, etc.... Relationships between people can be based on the nature of the relationship (e.g., business, family, friendships) or the means of communication (e.g., email, phone, personal meetings). Electronic files can be grouped by their type (Latex, C, html), names, the time

* This work was funded by the applied mathematics program at the United States Department of Energy and performed at Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 38–49, 2011.

they are created, or the pattern they are accessed. In these examples, there are multiple graphs that define relationships between the subjects. In sociology these graphs are called "graphs with multiple relations, multivariate graphs, or multiplexed graphs."[5] For brevity we use "multiweighted graphs." These multiweighted graphs differ from traditional multigraphs. In our case we have a fixed number of labeled edges rather than a multigraph which has a variable number of unlabeled edges.

This paper studies the community detection problem on networks with multiple edges-types/relations. Clustering is a method to reduce the complexity of a singly-weighted graph while still retaining much of its information. Groups of vertices (clusters) are formed which are well connected within the cluster and sparsely connected between clusters. This technique is a critical enabler in unsupervised machine learning and continues to be a very active area of research. Almost all methods however, require a singly-weighted graph. It is convenient to aggregate multi-weighted edges to a single composite edge. However, the choice of the aggregation function should be done cleverly, and we should be able to analyze the inevitable loss of information in the results.

Consider the situation where several edge types share redundant information yet as an ensemble combine to form some broader structure. For example scientific journal articles can be connected by text similarity, abstract similarity, keywords, shared authors, cross-citations, etc.... Many of these edge types reflect the *topic* of the document while others are also influenced by the *location* of the work. Text, abstract, and keyword similarity are likely to be redundant in conveying topic information (physics, math, biology) while shared authorship (two articles sharing a common author) is likely to convey both topic and location information because we tend to work with both those in our same field and with those in nearby institutions. We say that the topic and location attributes are *latent* because they do not exist explicitly in the data. We can represent much of the variation in the data by two relatively independent clusterings based on the topic of documents and their location. This compression of information from five edge types to two meaningful clusterings is the goal of this paper.

1.1 Contributions

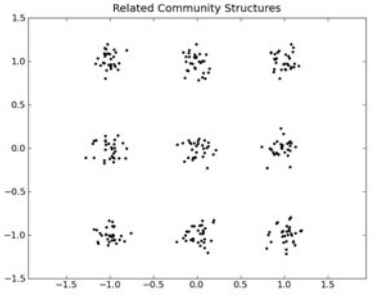
The community detection problem on networks with multiple edge types bears many interesting problems. In our earlier work we studied how to compute an aggregation scheme that best resonates with the ground-truth data, when such data was available [12]. In this work we study the following questions: Is there a meta-clustering structure, (i.e., are the clusterings clustered) and if so how do we find it? How do we find significantly different clusterings for the same data? Our main contributions in this paper are as follows.

- We describe how the space of clusterings can be searched using sampling methods, and investigate the structure of this space. We introduce the meta-clusters: while the clusterings vary with how we aggregate various similarity measures, these clusterings gather around a small number of clusters. That is clusterings are nicely clustered.
- We propose methods to efficiently represent the space of clusterings with minimal loss of information. More specifically, if we can produce a handful of clusterings

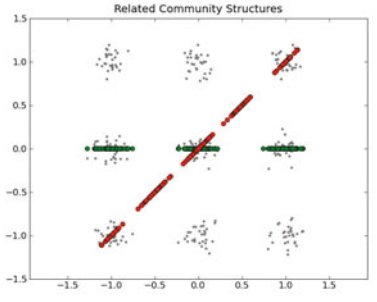
that represent the meta-clusters, then these small number of clusters can be used for data analysis, providing a more accurate and thorough information of the data, at a reasonable increase in processing times.
– We apply our proposed techniques to a data set collected from scientific articles in the arXiv database, and show that or proposed techniques can be successfully adopted for analysis of real data.

1.2 An Illustrative Problem

We construct a simple multiweighted network to demonstrate latent classes. For illustration, we assume our graph is perfectly embedded in $\mathbb{R}^2$ as seen in Fig. 1a. In this example each point on the plane represents a vertex, and two vertices are connected by an edge if they are close in distance. The similarity/weight for each edge is inversely proportional to the Euclidean distance. We see visually that there are nine natural clusters. More interestingly we see that these clusters are arranged symmetrically along two axes. These clusters have more structure than the set $\{1, 2, 3, ..., 9\}$. Instead they have the structure $\{1, 2, 3\} \times \{1, 2, 3\}$. An example of such a structure would be the separation of academic papers along two factors, {Physics, Mathematics, Biology} and {West Coast, Midwest, East Coast}. The nine clusters (with examples like physics articles from the West or biology articles from the Midwest) have underlying structure.



(a) 270 vertices arranged in nine clusters on the plane. Edges exist between vertices so that close points are well connected and distant points are poorly connected.



(b) Two 1D graphs arranged to suggest their relationship to the underlying 3x3 community structure. Both have clear community structures that are related but not entirely descriptive of the underlying 3x3 communities.

Fig. 1. Illustrating clusters (a) underlying structure and (b) low-dimensiona/partial views

Our data sets do not directly provide this information. For instance with journal articles we can collect information about authors, where the articles are published, and their citations. Each of these aspects provides only a partial view of the underlying structure. Analogous to our geometric example above we could consider features of the data as projections of the points to one dimensional subspaces. Distances/similarities between

the points in a projection have only partial information. This is depicted pictorially in Fig. 1b. For instance, the green projection represents a metric that clearly distinguishes between columns but cannot differentiate between different communities on the same column. The red projection on the other hand provides a diagonal sweep, capturing partial information about columns and partial information about rows. Neither of the two metrics can provide the full information for the underlying data. However when considered as an ensemble they do provide a complete picture. Our goal is to be able to tease out the latent factors of data from a given set of partial views.

In this paper, we will use this 3×3 example for conceptual purposes and for illustrations. Our approach is construct many multi-weighted graphs by using combinations of the partial views of the data. We will cluster these graphs and analyze these clusters to recover the latent structure.

2 Background

A weighted graph is represented as a tuple $G = (V, E)$, V a set of vertices and E a set of edges. Each edge e_i is a tuple $e_i = \{v_a, v_b, w_i \mid v_a, v_b \in V, w_i \in \mathbb{R}\}$ representing a connection between vertices v_a and v_b with weight w_i. In this work we replace $w_i \in \mathbb{R}$ with $\boldsymbol{w_i} \in \mathbb{R}^k$ with k being the number of edge types. We will construct functions that map multiweighted edges $\boldsymbol{w_i} \in \mathbb{R}^k$ to *composite edge types* $f(\boldsymbol{w_i}) = \omega_i \in \mathbb{R}$. In this paper f will be linear $\omega_i = \sum \alpha_i w_i$.

2.1 Clustering

Intuitively, the goal of clustering is to break down the graph into smaller groups such that vertices in each group are tightly coupled among themselves and loosely coupled with the remainder of the network. Both the translation of this intuition into a well-defined mathematical formula and design of associated algorithms pose big challenges. Despite the high quality and the high volume of the literature, the area continues to draw a lot of interest due to the growing importance of the problem and the challenges posed by the size and mathematical variety of the subject graphs.

Our goal here is to extend the concept of clustering to graphs with multiple edge types without getting into the details of clustering algorithms and formulations, since such a detailed study will be well beyond the scope of this paper. In this paper, we used *Graclus*, developed by Dhillon et al[3], which uses the top-down approach that recursively splits the graph into smaller pieces and *FastCommunity* developed by Clauset et al[2] which uses an agglomerative approach which optimizes the modularity metric. For further information on clustering see Lancichinetti et al.[6].

2.2 Variation of Information of Clusterings

At the core of most of our discussions will be similarity between two clusterings. Several metrics and methods have been proposed for comparing clusterings, such as *variation of information* [9], *scaled coverage measure* [13], *classification error* [7,8,9], and *Mirkin's metric* [10]. Out of these, we have used the variation of information metric in our experiments.

Let $C_0 = \langle C_0^1, C_0^2, \ldots, C_0^K \rangle$ and $C_1 = \langle C_1^1, C_1^2, \ldots, C_1^K \rangle$ be two clusterings of the same node set. Let n be the total number of nodes, and $P(C,k) = \frac{|C^k|}{n}$ be the probability that a node is in cluster C^k in a clustering C. Similarly the probability that a node is in cluster C^k in clustering C_i and in cluster C^l in clustering C_j is $P(C_i, C_j, k, l) = \frac{|C_i^k \cap C_j^l|}{n}$. The *entropy of information* or expectation value of learned information in C_i is defined

$$H(C_i) = -\sum_{k=1}^{K} P(C_i, k) \log P(C_i, k)$$

the mutual information shared by C_i and C_j is

$$I(C_i, C_j) = \sum_{k=1}^{K} \sum_{l=1}^{K\prime} P(C_i, C_j, k, l) \log P(C_i, C_j, k, l),$$

Given these two quantities Meila defines the variation of information metric by

$$d_{VI}(C_i, C_j) = H(C_i) + H(C_j) - 2I(C_i, C_j). \tag{1}$$

Meila [9] explains the intuition behind this metric a follows. $H(C_i)$ denotes the average uncertainty of the position of a node in clustering C_i. If, however, we are given C_j, $I(C_i, C_j)$ denotes average reduction in uncertainty of where a node is located in C_i. If we rewrite Equation (1) as

$$d_{VI}(C_i, C_j) = (H(C_i) - I(C_i, C_j)) \quad + \quad (H(C_j) - I(C_i, C_j)),$$

the first term measures the information lost if C_j is the true clustering and we know instead C_i, and the second term is the opposite.

The variation of information metric can be computed in $O(n)$ time.

2.3 Previous Work

Similar problems have been approached in previous work. Mucha et al.[11] looked at community detection when multiple edge types are sampled in time and strongly correlated. Dunlavy et al. [4] described this problem as a three dimensional Tensor and used a PARAFAC decomposition (SVD generalization) to identify dominant factors.

3 Searching the Space of Clusterings

From a multiweighted graph $G = (V, E)$ with edges $e_i \in E = (v_a, v_b, \langle w_i^0, w_i^1, \ldots, w_i^k \rangle)$ we can build a composite edge-type $\omega_i = \sum_j \alpha_j w_i^j$. This composite edge-type along with the vertex set V define a graph $G_{\boldsymbol{\alpha_j}}$ indexed by the vector α_j. We may apply a traditional clustering algorithm $\mathcal{C}$ to this graph to obtain a clustering $\mathcal{C}(G_{\boldsymbol{\alpha_j}}) = C_{\boldsymbol{\alpha_j}}$. This process identifies with each point $\boldsymbol{\alpha_j} \in \mathbb{R}^k$ a clustering $C_{\boldsymbol{\alpha_j}}$. Thus a multiweighted graph is imbued with a *space* of clusterings.

We expect that different regions of this space will have different clusterings. How drastic these differences are will depend on the particular multiweighted graph. How can we characterize this space of clusterings? Are there homogeneous regions, easily identifiable boundaries, groups of similar clusterings, etc...? We investigate the existence of a meta-clustering structure. That is we search for whether or not several clusterings in this space exhibit community structure themselves. In this section, we present our methods for these questions on the 3×3 data. We will later provide results on a larger data set.

3.1 Sampling the Clustering Space

To inspect the space of clusterings we sample in a Monte Carlo fashion. We take points $\boldsymbol{\alpha_i} \in \mathbb{R}^k$ such that $|\boldsymbol{\alpha_i}| = 1$, and compute the appropriate graph and clustering at each point. We may then compare these clusterings in aggregate.

As our first experiment, we take 16 random one-dimensional projections of the points laid out in the plane shown in Fig. 1 and consider the projected-point-wise distances in aggregate as a multiweighted graph. From this multiweighted graph we take 800 samples of the linear space of clusterings. These 800 clusterings approximate the clustering structure of the multiweighted graph.

The results of these experiments are presented in Figure 2(a). In this figure each row and column corresponds to a clustering of the graph. Entries in the matrix represent the variation of information distance between two clusterings. Therefore dark regions in this matrix are sets of clusterings that are highly similar. White bands show informational independence between regions. The rows/columns of this matrix have been ordered to have more similar clusterings closer to each other so as to highlight the clusters of clusterings detected.

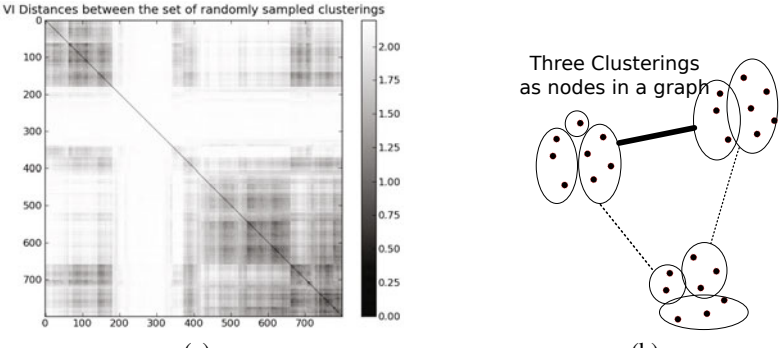


Fig. 2. The Meta-clustering information (a) VI distances between 800 sampled clusterings. Vertices are ordered to show optimal clustering of this graph. Dark blocks on the diagonal represent clusters. The white band is a group of completely independent clusterings. (b) Three Clusterings treated as nodes in a graph. Similar clusterings (top two) are connected with high-weighted edges. Distant clusterings are connected with low-weighted edges.

3.2 Meta-clusters: Clusters of Clusterings

While it is interesting to know that significantly different clusterings can be found, the lack of stable clustering structure is not helpful for applications of clustering such as for unsupervised learning. We need to reduce this set of clusterings further. We approach this problem by applying the idea of clustering onto this set of clusterings. We call this problem the *meta-clustering* problem.

We represent the clusterings as nodes in a graph and connect them with edge-weights determined by the inverse of the variation of information metric [9]. We inspect this graph to see if it contains clusters. That is, we *cluster the graph of clusterings* to see if there exist some tightly coupled clusters of clusterings within the larger space. For instance in Fig. 2(b) the top two clusterings differ only in the position of a single vertex and thus are highly similar. In contrast the bottom clustering is different from both and is weakly connected.

Figure 2(a) reveals the meta-clustering structure in our experiments. The dark blocks around the diagonal correspond to meta-clusters. We can see two big blocks in the upper left and lower right corners. Furthermore, there is a hierarchical clustering structure within these blocks, as we see smaller blocks within the larger blocks. In this experiment, we were able to observe meta-clusters. As usual, results depend on the particular problem instance. While we do not claim that one can always find such meta-clusters, we expect that they will exist in many multi-weighted graphs, and exploiting the meta-clustering structure can enable efficiently handling this space, which is the topic of the next section.

4 Efficient Representation of the Clusterings

In this section we study how to efficiently represent the meta-clustering structure. First we will study how to reduce a cluster of clusterings into a single averaged or representative clustering. Then, we will study how to select and order a small number of meta-clusters to cover the clustering space efficiently.

4.1 Averaging Clusterings within a Cluster

To increase the human accessibility of this information we reduce each cluster of clusterings into a single representative clustering. We use the "Cluster-based Similarity Partitioning Algorithm" (CSPA) proposed by Strehl et. al [14] to combine several clusterings into a single average. In this algorithm each pair of vertices is connected with an edge with weight equal to the number of clusters in which they co-occur. If v_a and v_b are in the same cluster in k of the clusterings then in this new graph they are connected with weight k. If they are never in the same cluster then they are not connected. We then cluster this graph and use the resultant clustering as the representative. In Fig. 3 we depict the addition of three clusterings to form an average graph which can then be clustered.

We perform this process on the clusters of clusterings found in section 3.2 and presented in Fig. 2(a) to obtain the *representative-clusterings* in Fig. 4. We see that the product of the first two representative-clusterings identifies the original nine clusterings

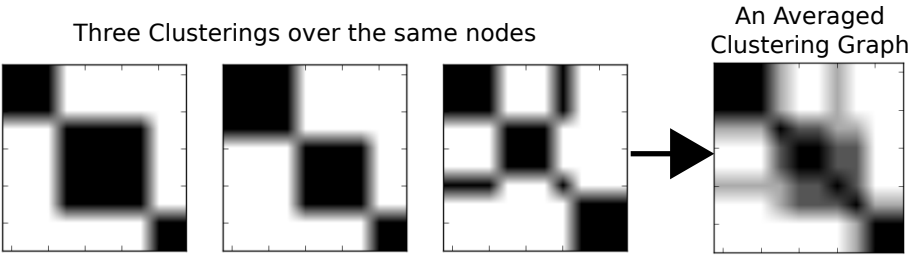


Fig. 3. Showing the CSPA [14] averaging procedure for clusterings. Each clustering is displayed as a block diagonal graph (or permutation) with two nodes connected if and only if they are in the same cluster. Then an aggregate graph (right) is formed by the addition of these graphs. This graph on the right is then clustered using a traditional algorithm. This clustering is returned as the representative-clustering.

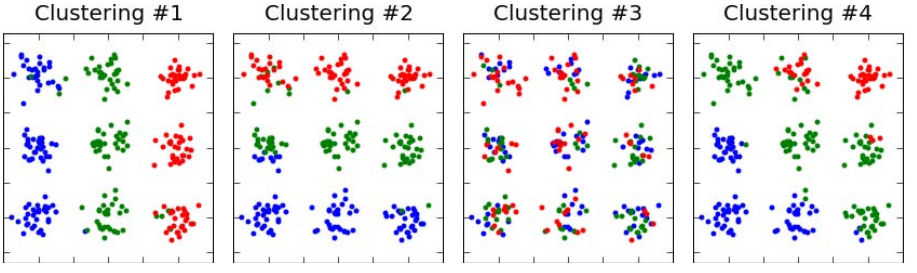


Fig. 4. Representative-Clusterings of the four dominant clusters-of-clusterings from Fig. 2(a). Clusterings are displayed as colorings of the original points in the 2-d plane. These are ordered to maximize cumulative set-wise information. Notice how the first two representative-clusterings recover the original nine clusterings exactly.

with little error. We see also that the two factors are identified perfectly by each of these clusterings individually.

4.2 Ordering by Set-Wise Information Content

In Fig. 4, the original 3x3 community structure can be reconstructed using only the first two representative-clusterings. Why are these two chosen first? Selecting the third and fourth representative-clusterings would not have had this pleasant result. How should we order the set of representative-clusterings?

We may judge a set of representative-clusterings by a number of factors: *(i)* How many of our samples ascribe to the associated meta-clusters, what fraction of the space of clusterings do they cover? *(ii)* How much information do the clusterings cover as a set? *(iii)* How redundant are the clusterings? How much informational overlap is present? We would like to maximize information while minimizing redundancy. In Fig. 4 we ordered the representative-clusterings to maximize setwise information. Minimizing redundancy came as a fortunate side-effect. Notice how each of the clusterings in

order is independent from the preceding ones. Knowing that a vertex is red in the first image tells you nothing about the color of the vertex in the second. The second therefore brings only novel information and no redundancy.

To compute the information content of a set of clusterings we extend the Variation of Information metric in a natural way. In section 2.2 we introduced the mutual information of two clusterings as follows:

$$I(C_i, C_j) = \sum_{k=1}^{K} \sum_{l=1}^{K\prime} P(C_i, C_j, k, l) \log P(C_i, C_j, k, l),$$

where $P()$ is the probability that a randomly selected node was in the specified clusters. This is equivalent to the self-information of the Cartesian product of the two clusterings. Its extension to a set of clusterings $I(C_\alpha, C_\beta, \ldots, C_\omega)$ is

$$\sum_{a=1}^{K} \sum_{b=1}^{K'} \cdots \sum_{z=1}^{K'''} P(C_\alpha, C_\beta, \ldots, C_\omega, a, b, \ldots, z) \log P(C_\alpha, C_\beta, \ldots, C_\omega, a, b, \ldots, z).$$

For a large number of clusterings or large K this quickly becomes inconvenient. In these cases we order the clusterings by adding new clusterings to the set based on maximizing the minimum pairwise distance to every other clustering currently in the set. This process is seeded with the informationally maximal pair within the set. This does not avoid triple-wise information overlap but works well in practice.

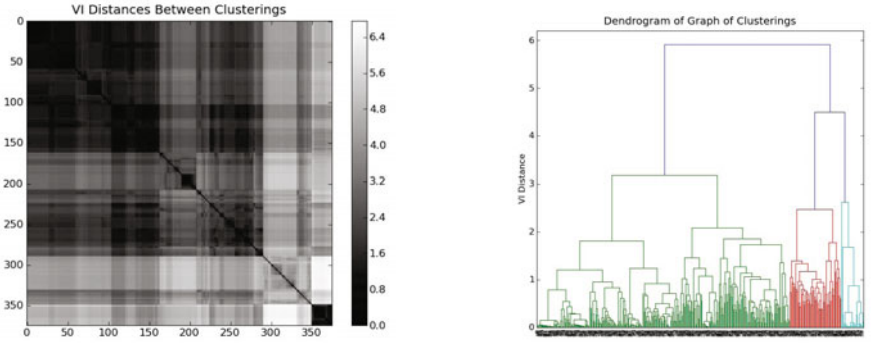
5 Physics Articles from arXiv.org

ArXiv.org releases convenient metadata (title, authors, etc...) for all articles in their database. Additionally, a special set of 30 000 high energy physics articles are released with abstracts and citation networks. We apply our process to this network of papers with edge types *Titles, Authors Abstracts and Citations*.

Articles are connected by *title* or *abstract* based on the cosine similarity of the text (using the bag of words model[1]). Two articles are connected by *author* by the number of authors that the two articles have in common. Two articles are connected by *citation* if either article cites the other (undirected). We inspect this system with the following process discussed in greater detail above.

These graphs are normalized by the L_2 norm and then the space of composite edge types is sampled uniformly. That is $\omega_j = \sum_{i=1}^{4} \alpha_i w_i$, where $\alpha_i \in (-1, 1)$, $w_i \in \{$*titles, abstract, authors, citation*$\}$. The resulting graphs are then clustered using Clauset et al's FastModularity[2] algorithm. The resulting clusterings are compared in a graph which is then clustered to produce clusters of clusterings. The clusters of clusterings are averaged [14] and we inspect the resultant representative-clusterings.

The similarity matrix of the graph of clusterings is shown in Fig. 5(a). The presence of blocks on the diagonal imply clusters of clusterings. From this process we obtain representative-clusterings. The various partitionings of the original set of papers vary considerably (large VI distance) yet exhibit high modularity scores implying a variety of high-quality clusterings within the dataset.



(a) (b)

Fig. 5. (a) The pairwise distances between the sampled clusterings form a graph. Note the dark blocks along the diagonal. These are indicative of tightly knit clusters. (b) A dendrogram of this graph. We use the ordering of the vertices picked out by the dendrogram to optimally highlight the blocks in the left image.

Table 1. Commonly appearing words (stemmed) in two distinct representative-clusterings. Clusters within each clustering correspond to well known subfields in High-Energy Physics (Traditional Field Theory/Lattice QCD, Cosmology/GR, Supersymmetry/String Theory). This data however does not show a strong distinction between the clusterings. Furher investigation is warranted.

Cluster	Statistically Significant Words in Clustering 1
1	quantum, algebra, integr, equat, model, chern-simon, lattic, particl, affin
2	potenti, casimir, self-dual, dilaton, induc, cosmolog, brane, anomali, scalar
3	black, hole, brane, supergrav, cosmolog, ads/cft, sitter, world, entropi
4	cosmolog, black, hole, dilaton, graviti, entropi, dirac, 2d, univers
5	d-brane, tachyon, string, matrix, theori, noncommut, dualiti, supersymmetr, n=2

Cluster	Statistically Significant Words in Clustering 2
1	potenti, casimir, self-dual, dilaton, induc, energi, scalar, cosmolog, gravit
2	integr, model, toda, equat, function, fermion, casimir, affin, dirac
3	tachyon, d-brane, string, orbifold, n=2, n=1, dualiti, type, supersymmetr
4	black, hole, noncommut, supergrav, brane, sitter, entropi, cosmolog, graviti

Analysis of this dataset is challenging and still in progress. We can look at articles in a clustering and inspect attributes like the country (by submitting e-mail's country code), or words which occur more often than statistically expected given the corpus. Most clusterings found show a separation into various topics identifyable by domain experts (example in Table 1) however a distinction between clusterings has not yet been found. While the VI distance between metaclusterings presented in Fig. 5(a) is large it has so far proven difficult to identify the qualitative distinction for the quantitative difference. More in depth inspection by a domain expert may be necessary.

6 Conclusion and Future Work

We investigated clustering in the context of network data with multiple relationships between nodes. We found that a rich clustering structure can exist with clusters of clusterings. In an example we found that by reducing this clustering structure we uncovered latent classes which explained the underlying graph very compactly. We presented a simple method that works well on simple cases. In the future it will be interesting to apply these methods to more challenging problems and see which aspects become interesting. There is much room for growth in this topic. Ongoing work includes more intelligent sampling (intentionally finding distinct clusterings), effects of adding nonlinear combinations of edge-types, and searching the space for clusterings with desired attributes.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3(4-5), 993–1022 (2003), doi:10.1162/jmlr.2003.3.4-5.993
2. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics 70(6 Pt. 2), 066111 (2004), http://www.ncbi.nlm.nih.gov/pubmed/15697438
3. Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(11), 1944–1957 (2007), http://www.ncbi.nlm.nih.gov/pubmed/17848776, doi:10.1109/TPAMI.2007.1115
4. Dunlavy, D.M., Kolda, T.G., Philip Kegelmeyer,W.: Multilinear Algebra For Analyzing Data With Multiple Linkages (2006)
5. Fienberg, S.E., Meyer, M.M., Wasserman, S.S.: Statistical Analysis of Multiple Sociometric Relations. Journal of the American Statistical Association 80(389), 51–67 (1985), http://www.jstor.org/stable/2288040
6. Lancichinetti, A., Fortunato, S.: Community detection algorithms: A comparative analysis. Physical Review E 80(5) (November 2009), http://pre.aps.org.proxy.uchicago.edu/abstract/PRE/v80/i5/e056117, doi:10.1103/PhysRevE.80.056117
7. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions, neural computation. Neural Computation 16, 1299–1323 (2004)
8. Luo, X.: On coreference resolution performance metrics. In: Proc. Human Language Technology Conf. and Conf. Empirical Methods in Natural Language Processing, pp. 25–32. Association for Computational Linguistics, Vancouver (2005)
9. Meila, M.: Comparing Clusterings by the Variation of Information. Technical Report, pp. 173–187 (2003)
10. Mirkin, B.: Mathematical Classification and Clustering. Kluwer Academic Press, Dordrecht (1996)
11. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.P.: Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. Science 328(5980), 876–878 (2010), http://www.sciencemag.org/content/328/5980/876.full, doi:10.1126/science.1184819

12. Rocklin, M., Pinar, A.: Computing an aggregate edge-weight function for clustering graphs with multiple edge types. In: Kumar, R., Sivakumar, D. (eds.) WAW 2010. LNCS, vol. 6516, pp. 25–35. Springer, Heidelberg (2010)
13. Stichting, C., Centrum, M., Dongen, S.V.: Performance criteria for graph clustering and markov cluster experiments. Technical Report INS-R0012, Centre for Mathematics and Computer Science (2000)
14. Strehl, A., Ghosh, J.: Cluster Ensembles A Knowledge Reuse Framework for Combining Multiple Partitions. Journal of Machine Learning Research 3(3), 583–617 (2003), http://www.crossref.org/jmlr_DOI.html, doi:10.1162/153244303321897735

Quick Detection of Top-k Personalized PageRank Lists

Konstantin Avrachenkov[1], Nelly Litvak[2], Danil Nemirovsky[1], Elena Smirnova[1], and Marina Sokol[1]

[1] INRIA Sophia Antipolis
{k.avrachenkov,dnemirov,esmirnov,msokol}@sophia.inria.fr
[2] University of Twente
n.litvak@ewi.utwente.nl

Abstract. We study a problem of quick detection of top-k Personalized PageRank (PPR) lists. This problem has a number of important applications such as finding local cuts in large graphs, estimation of similarity distance and person name disambiguation. We argue that two observations are important when finding top-k PPR lists. Firstly, it is crucial that we detect fast the top-k most important neighbors of a node, while the exact order in the top-k list and the exact values of PPR are by far not so crucial. Secondly, by allowing a small number of "wrong" elements in top-k lists, we achieve great computational savings, in fact, without degrading the quality of the results. Based on these ideas, we propose Monte Carlo methods for quick detection of top-k PPR lists. We demonstrate the effectiveness of these methods on the Web and Wikipedia graphs, provide performance evaluation and supply stopping criteria.

1 Introduction

Personalized PageRank (PPR) or Topic-Sensitive PageRank [15] is a generalization of PageRank [10], and is a stationary distribution of a random walk on an entity graph, with random restart from a given personalization distribution. Originally designed for personalization of the Web search results [15], PPR found a large number of network applications, e.g., in finding related entities [11], graph clustering and finding local cuts [1,4], link predictions in social networks [20] and protein-protein interaction networks [23]. The recent application of PPR to the person name disambiguation problem lead to the first official place in the WePS 2010 challenge [21]. In most of applications, e.g., in name disambiguation, one is mainly interested in detecting top-k elements with the largest PPR. This work on detecting top-k elements is driven by the following two key observations:

Observation 1: Often it is extremely important to detect fast the top-k elements with the largest PPR, while the exact order in the top-k list as well as the exact values of the PPR are by far not so important. Application examples are given in the above mentioned references.

Observation 2: We may apply a relaxation that allows a small number of elements to be placed erroneously in the top-k list. If the PPR values of these

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 50–61, 2011.

elements are of a similar order of magnitude as in the top-k list, then such relaxation does not affect applications, but it enables us to take advantage of the generic "80/20 rule": 80% of the result is achieved with 20% of efforts.

We argue that the Monte Carlo approach naturally takes into account the two key observations. In [9] this approach was proposed for the computation of the standard PageRank. The estimation of the convergence rate in [9] was very pessimistic. The implementation of the Monte Carlo approach was improved in [13] and also applied there to PPR. Both [9] and [13] only use end points of the random walks to compute the PageRank values. Moreover, [13] requires extensive precomputation efforts and is very demanding in storage resource. In [5] the authors have further improved the realization of the Monte Carlo method [13]. In [2] it is shown that Monte Carlo estimation for large PageRank values requires about the same number of operations as one iteration of the power iteration method. In this paper we show that the Monte Carlo algorithms require an incomparably smaller number of operations when our goal is to detect a top-k list with k not large. In our test on the Wikipedia entity graph with about 2 million nodes typically few thousands of operations are enough to detect the top-10 list with just two or three erroneous elements. Hence, we obtain a relaxation of the top-10 list with just about 1-5% of operations required by one power iteration. Experimental results on the Web graph appear to be even more striking. In the present work we provide theoretical justifications for such remarkable efficiency. We would like to emphasize that the Monte Carlo approach allows easy online and parallel implementation and does not require the knowledge of the complete graph.

We consider the present work as an initiation to a new line of research on quick detection of top-k ranked network central elements. A number of interesting questions will be addressed in the future research: What is the difference in performance between the randomized algorithms, like the presented Monte Carlo algorithms, and the non-randomized algorithms, like algorithms in [1] and [7]? What are efficient practical stopping criteria for the randomized algorithms? What is the effect of the graph structure on the performance of the randomized algorithms?

2 Monte Carlo Methods

Given a directed or undirected graph connecting some entities, the PPR $\pi(s, c)$ with a seed node s and a damping parameter c is defined as a solution of the following equations

$$\pi(s,c) = c\pi(s,c)P + (1-c)\mathbf{1}_s^T, \qquad \sum_{j=1}^{n} \pi_j(s,c) = 1,$$

where $\mathbf{1}_s^T$ is a row unit vector with one in the s^{th} entry and all the other elements equal to zero, P is the transition matrix associated with the entity graph and n is the number of entities. Equivalently, PPR can be given by [19]

$$\pi(s,c) = (1-c)\mathbf{1}_s^T[I - cP]^{-1}. \tag{1}$$

When the values of s and c are clear from the context we shall simply write π.

We note that PPR is often defined with a general distribution v in place of $\mathbf{1}_s^T$. However, typically v has a small support. Then, due to linearity, the problem of PPR with distribution v reduces to computing PPR with distribution $\mathbf{1}_s^T$ [16].

In this work we consider two Monte Carlo algorithms. The first algorithm is inspired by the following observation. Consider a random walk $\{X_t\}_{t\geq 0}$ that starts from node s, i.e, $X_0 = s$. Let at each step the random walk terminate with probability $1 - c$ and make a transition according to the matrix P with probability c. Then, the end-points of such a random walk has the distribution $\pi(s, c)$.

Algorithm 1 (MC End Point). *Simulate m runs of the random walk $\{X_t\}_{t\geq 0}$ initiated at node s. Evaluate π_j as a fraction of m random walks which end at node $j \in 1, \ldots, n$.*

Next, we exploit the fact that the element (s, j) of the matrix $[I - cP]^{-1}$ equals to the expected number of visits to node j by the random walk initiated at state s with the run time geometrically distributed with parameter c [2]. Thus, the formula (1) suggests the following estimator for the PPR

$$\hat{\pi}_j(s, c) = (1 - c)\frac{1}{m}\sum_{r=1}^{m} N_j(s, r), \tag{2}$$

where $N_j(s, r)$ is the number of visits to state j during the run r of the random walk initiated at node s. This leads to our second Monte Carlo algorithm.

Algorithm 2 (MC Complete Path). *Simulate m runs of the random walk $\{X_t\}_{t\geq 0}$ initiated at node s. Evaluate π_j as the total number of visits to node j multiplied by $(1 - c)/m$.*

As outputs of the proposed algorithms we would like to obtain with high probability either a *top-k list* of nodes or a *top-k basket* of nodes.

Definition 1. *The* top-k list *of nodes is a list of k nodes with largest PPR values arranged in a descending order of their PPR values.*

Definition 2. *The* top-k basket *of nodes is a set of k nodes with largest PPR values with no ordering required.*

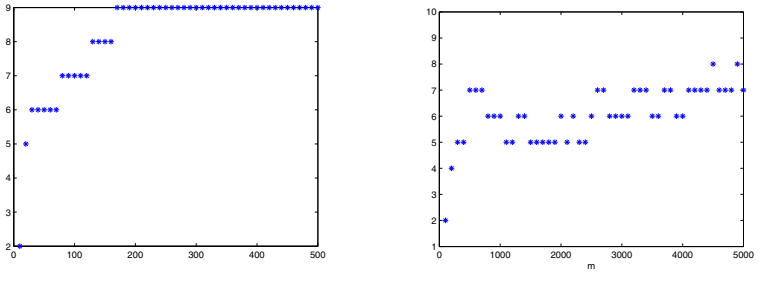
It turns out that it is beneficial to relax our goal and to obtain a top-k basket with a small number of erroneous elements.

Definition 3. *We call* relaxation-l *top-k basket a realization when we allow at most l erroneous elements from top-k basket.*

In the present work we aim to estimate the numbers of random walk runs m sufficient for obtaining top-k list or top-k basket or relaxation-l top-k basket with high probability. In particular, we demonstrate that ranking converges considerably faster than the values of PPR and that a relaxation-l with quite small l helps significantly.

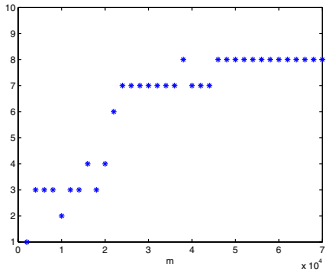
Throughout the paper we illustrate the theoretical analysis with the help of experiments on two large graphs: the Wikipedia entity graph and the Web graph. There is a number of reasons why we have chosen the Wikipedia entity graph. Firstly, all elements of PPR can be computed with high precision for the Wikipedia entity graph with the help of BVGraph/WebGraph framework [8]. Secondly, the Wikipedia graph has already been used in several applications related to finding top-k semantically related entities. Thirdly, since the Wikipedia entity graph has a very small average distance [24], it represents a very challenging test for the Monte Carlo methods. In just 3-4 steps the random walk can be very far from the starting node. Since the Monte Carlo approach does not require the knowledge of a complete graph, we can apply our algorithms to the actual Web graph. However, computing the exact values for the Pesonalized PageRank of web pages is infeasible in our experiments. We can only obtain correct top-k lists by Monte Carlo methods with very high probability as in [2,13] using an ample number of crawls.

Illustrating example with Wikipedia: Following our recent work [21] we illustrate PPR by application to the person name disambiguation problem. One of the most common English names is Jackson. We have selected three Jacksons who have entries in Wikipedia: Jim Jackson (ice hockey), Jim Jackson (sportscaster) and Michael Jackson. Two Jacksons have even a common given name and both worked in ice hockey, one as an ice hockey player and another as an ice hockey sportscaster. In [3] we provide the exact lists of top-10 Wikipedia articles arranged according to PPR vectors. We observe that an exact top-10 list identifies quite well its seed node. Next, we run the Monte Carlo End Point method starting from each seed node. Notice that to obtain a relaxed top-10 list with two or three erroneous elements we need different number of runs for different seed nodes (50000 runs for `Michael Jackson` vs. 500 runs for `Jim Jackson (ice hockey)`). Intuitively, the more immediate neighbours a node has, the larger number of Monte Carlo steps is required. Indeed, if a seed node has many

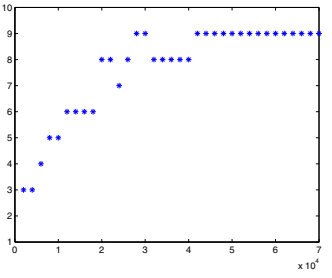


(a) Seed node `Jim Jackson (ice hockey)`. (b) Seed node `Jim Jackson (sportscaster)`.

Fig. 1. The number of correctly detected elements by MC End Point

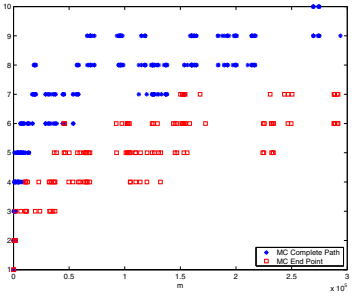


(a) Seed node `Michael Jackson`.

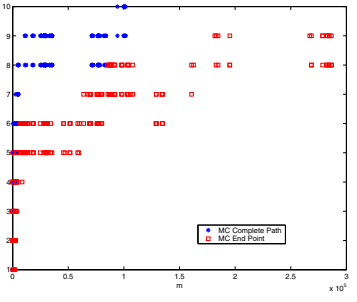


(b) Seed node `Michael Jackson`.

Fig. 2. The number of correctly detected elements by MC End Point (a) and MC Complete Path (b)



(a) Seed node Michael Jackson Web page.



(b) Seed node Jim Jackson Web page.

Fig. 3. The number of correctly detected elements by MC End Point

immediate neighbours then the Monte Carlo method easily drifts away. In Figures 1-2.(a) we present examples of typical runs of the Monte Carlo End Point method for the three different seed nodes. An example of the Monte Carlo Complete Path method for the seed node `Michael Jackson` is given in Figure 2.(b). As expected, it outperforms the Monte Carlo End Point method. In the following sections we shall quantify all the above qualitative observations.

Illustrating example with the Web: We have also tested our two Monte Carlo methods on the Web. To see the difference in comparison with a "smaller" Wikipedia graph we have chosen the official Web page of Michael Jackson `http://www.michaeljackson.com` and the Web page of the hockey player statistics Jim Jackson hosted at `http://www.hockeydb.com`. In Figures 3.(a) and 3.(b) we present examples of typical runs of the Monte Carlo Complete Path and End Point methods for, respectively, the Michael Jackson Web page and the Jim Jackson Web page as a seed node. We have performed enough steps (6×10^5) to make sure that the top-k lists of nodes are stabilized so

that we could say with very high certainty that we know the correct top-k lists. We observe that in comparison to the Wikipedia graph we need longer runs. However, the amount of computational saving is still very impressive. Indeed, according to even modest estimates, the size of the Web is more than 10^{10} pages. However, to get a good top-k list for the Michael Jackson page we need about 10^5 steps with MC Complete Path. Thus, we are using only 10^{-5} fraction of computational resources which are needed for just one power iteration!

3 Variance Based Performance Comparison and CLT Approximations

In the MC End Point algorithm the distribution of end points is multinomial [17]. Namely, if we denote by L_j the number of paths that end at node j after m runs, then we have

$$P\{L_1 = l_1, L_2 = l_2, \ldots, L_n = l_n\} = \frac{m!}{l_1! l_2! \cdots l_n!} \pi_1^{l_1} \pi_2^{l_2} \cdots \pi_n^{l_n}. \tag{3}$$

Thus, the standard deviation of the MC End Point estimator for the k^{th} element is given by

$$\sigma(\hat{\pi}_k) = \sigma(L_k/m) = \frac{1}{\sqrt{m}} \sqrt{\pi_k(1-\pi_k)}. \tag{4}$$

An expression for the standard deviation of the MC Complete Path is more complicated. Define the matrix $Z = (z_{ij}) = [I - cP]^{-1}$ and let N_j be the number of visits to node j by the random walk with the run time geometrically distributed with parameter c. Further, denote by $E_i(\cdot)$ a conditional expectation provided that the random walk starts at $i = 1, \ldots, n$. From (2), it follows that

$$\sigma(\hat{\pi}_k) = \frac{(1-c)}{\sqrt{m}} \sigma(N_k) = \frac{(1-c)}{\sqrt{m}} \sqrt{E_s\{N_k^2\} - E_i\{N_k\}^2}. \tag{5}$$

First, we recall that

$$E_s\{N_k\} = z_{sk} = \pi_k(s)/(1-c). \tag{6}$$

Then, from [18], it is known that $E_s\{N_k^2\} = [Z(2Z_{dg} - I)]_{sk}$, where Z_{dg} is a diagonal matrix having as its diagonal the diagonal of matrix Z and $[A]_{ik}$ is the $(i,k)^{\text{th}}$ element of matrix A. Thus, we write

$$\begin{aligned} E_s\{N_k^2\} &= \mathbf{1}_s^T Z(2Z_{dg} - I)\mathbf{1}_k = \frac{1}{1-c}\pi(s)(2Z_{dg} - I)\mathbf{1}_k \\ &= \frac{1}{1-c}\left(\frac{1}{1-c}\pi_k(s)\pi_k(k) - \pi_k(s)\right). \end{aligned} \tag{7}$$

Substituting (6) and (7) into (5), we obtain

$$\sigma(\hat{\pi}_k) = \frac{1}{\sqrt{m}} \sqrt{\pi_k(s)(2\pi_k(k) - (1-c) - \pi_k(s))}. \tag{8}$$

Since $\pi_k(k) \approx 1 - c$, we can approximate $\sigma(\hat{\pi}_k)$ with

$$\sigma(\hat{\pi}_k) \approx \frac{1}{\sqrt{m}} \sqrt{\pi_k(s)((1-c) - \pi_k(s))}.$$

Comparing the latter expression with (4), we see that MC End Point requires approximately $1/(1-c)$ walks more than MC Complete Path. This was expected as MC End Point uses only information from end points of the random walks. We would like to emphasize that $1/(1-c)$ can be a significant coefficient. For instance, if $c = 0.85$, then $1/(1-c) \approx 6.7$.

Now, for the MC End Point we can use CLT-type result given e.g. in [22]:

Theorem 1. *[22] For large m and $\sum_{i=1}^{n} l_i = m$, a multivariate normal density approximation to the multinomial distribution (3) is given by*

$$f(l_1, l_2, \ldots, l_n) = \left(\frac{1}{2\pi m}\right)^{(n-1)/2} \times \left(\frac{1}{n\pi_1\pi_2\cdots\pi_n}\right)^{1/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}\frac{(l_i - m\pi_i)^2}{m\pi_i}\right\}. \tag{9}$$

For the MC Complete Path, we note that $N(s,r) = (N_1(s,r), \ldots, N_n(s,r))$, $r = 1, 2, \ldots$, form a sequence of i.i.d. random vectors. Hence, we can apply the multivariate central limit theorem. Denote

$$\hat{N}(s,m) = \frac{1}{m}\sum_{r=1}^{m} N(s,r). \tag{10}$$

Theorem 2. *Let m go to infinity. Then, we have the following convergence in distribution to a multivariate normal distribution*

$$\sqrt{m}\left(\hat{N}(s,m) - \bar{N}\right) \xrightarrow{D} \mathcal{N}(0, \Sigma(s)),$$

where $\bar{N}(s) = \mathbf{1}_s^T Z$ and $\Sigma(s) = E\{N^T(s,r)N(s,r)\} - \bar{N}^T(s)\bar{N}(s)$ is a covariance matrix, which can be expressed as

$$\Sigma(s) = \Omega(s)\, Z + Z^T \Omega(s) - \Omega(s) - Z^T \mathbf{1}_s \mathbf{1}_s^T Z. \tag{11}$$

where the matrix $\Omega(s) = \{\omega_{jk}(s)\}$ is defined by

$$\omega_{jk}(s) = \begin{cases} z_{sj}, \text{ if } j = k, \\ 0, \quad \text{otherwise.} \end{cases}$$

Proof. See [3].

We would like to note that in both cases we obtain the convergence to rank deficient (singular) multivariate normal distributions.

Illustrating example with the Web (cont.): In Table 1 we provide means and standard deviations for the number of hits of MC End Point for top-10

Table 1. MC End Point for the Jim Jackson Web page: means and Standard Deviations

nr. runs	10000		100000	
rank	mean	std	mean	std
1	0.79104	0.012531	0.79748	0.004834
2	0.006329	0.001622	0.006202	0.000569
3	0.005766	0.001075	0.005885	0.000354
4	0.006365	0.002103	0.006561	0.000704
5	0.005183	0.001664	0.005518	0.00055
6	0.005541	0.003766	0.005801	0.001257
7	0.007617	0.003633	0.006243	0.001266
8	0.007566	0.012384	0.005854	0.003969
9	0.006186	0.001468	0.006182	0.000547
10	0.00672	0.003492	0.006223	0.001132

nodes with the Jim Jackson Web page as the seed node for the number of runs $m = 10^4$ and $m = 10^5$. We observe that the means are very close to each other and the standard deviations are significant with respect to the values of the means. This shows that a direct application of the central limit theorem and the confidence intervals technique will lead to inadequate stopping criteria. In the ensuing sections we discuss metrics and stopping criteria which are much more efficient for the present problem.

4 Convergence Based on Order

For the two introduced Monte Carlo methods we aim to calculate or estimate a probability that after a given number of steps we correctly obtain top-k list or top-k basket. These are the probabilities $P\{L_1 > \cdots > L_k > L_j, \forall j > k\}$ and $P\{L_i > L_j, \forall i, j : i \leq k < j\}$ respectively, where L_k, $k \in 1, \ldots, n$, can be either the Monte Carlo estimates or the ranked elements or their CLT approximations. We refer to these probabilities as the ranking probabilities and we refer to complementary probabilities as misranking probabilities [6]. Because of combinatorial explosion, exact calculation of these probabilities is infeasible in non-trivial cases. Thus, we propose estimation methods based on Bonferroni inequality. This approach works for reasonably large values of m.

Drawing correctly the top-k basket is defined by the event $\bigcap_{i \leq k < j} \{L_i > L_j\}$. Applying the Bonferroni inequality $P\{\bigcap_s A_s\} \geq 1 - \sum_s P\left\{\bar{A}_s\right\}$ to this event, we obtain $P\left\{\bigcap_{i \leq k < j} \{L_i > L_j\}\right\} \geq 1 - \sum_{i \leq k < j} P\left\{\overline{\{L_i > L_j\}}\right\}$. Equivalently, we can write the following upper bound for the misranking probability

$$1 - P\left\{\bigcap_{i \leq k < j} \{L_i > L_j\}\right\} \leq \sum_{i \leq k < j} P\{L_i \leq L_j\}. \tag{12}$$

We note that the upper bound for the misranking probability is very useful, because it will provide a guarantee on the performance of our algorithms. Since in the MC End Point method the distribution of end points is multinomial (see (3)), for small m we can directly use the formula

$$P\{L_i \leq L_j\} = \sum_{l_i + l_j \leq m,\ l_i \leq l_j} \frac{m!}{l_i! l_j! (m - l_i - l_j)!} \pi_i^{l_i} \pi_j^{l_j} (1 - \pi_i - \pi_j)^{m - l_i - l_j}. \tag{13}$$

For large m it is computationally intractable. Hence, we now turn to the CLT approximations for the both MC methods. Denote by L_j the original number of hits at node j and by Y_j its CLT approximation. First, we obtain a CLT based expression for the misranking probability for two nodes $P\{Y_i \le Y_j\}$. Since the event $\{Y_i \le Y_j\}$ coincides with the event $\{Y_i - Y_j \le 0\}$ and a difference of two normal random variables is again a normal random variable, we obtain $P\{Y_i \le Y_j\} = P\{Y_i - Y_j \le 0\} = 1 - \Phi(\sqrt{m}\rho_{ij})$, where $\Phi(\cdot)$ is the cumulative distribution function for the standard normal random variable and

$$\rho_{ij} = \frac{E[Y_i] - E[Y_j]}{\sqrt{\sigma^2(Y_i) - 2\mathrm{cov}(Y_i, Y_j) + \sigma^2(Y_j)}}.$$

For large m, the above expression can be bounded by $P\{Y_i \le Y_j\} \le \frac{1}{\sqrt{2\pi}} e^{-\frac{\rho_{ij}^2}{2}m}$. Since the misranking probability for two nodes $P\{Y_i \le Y_j\}$ decreases when j increases, we can write

$$1 - P\left\{\bigcap_{i \le k < j} \{Y_i > Y_j\}\right\} \le \sum_{i=1}^{k} \left(\sum_{j=k+1}^{j^*} P\{Y_i \le Y_j\} + \sum_{j=j^*+1}^{n} P\{Y_i \le Y_{j^*}\} \right),$$

for some j^*. This gives the following upper bound

$$1 - P\left\{\bigcap_{i \le k < j} \{Y_i > Y_j\}\right\} \le \sum_{i=1}^{k} \sum_{j=k+1}^{j^*} (1 - \Phi(\sqrt{m}\rho_{ij})) + \frac{n - j^*}{\sqrt{2\pi}} \sum_{i=1}^{k} e^{-\frac{\rho_{ij^*}^2}{2}m}. \quad (14)$$

Since we have a finite number of terms in the right hand side of expression (14), we conclude that

Theorem 3. *The misranking probability of the top-k basket goes to zero with geometric rate,* $1 - P\left\{\bigcap_{i \le k < j} \{Y_i > Y_j\}\right\} \le Ca^m$, *for some* $C > 0$, $a \in (0, 1)$.

We note that the multinomial distribution, ρ_{ij} has a simple expression

$$\rho_{ij} = \frac{\pi_i - \pi_j}{\sqrt{\pi_i(1 - \pi_i) + 2\pi_i\pi_j + \pi_j(1 - \pi_j)}}.$$

For MC Complete Path $\sigma^2(Y_i) = \Sigma_{ii}(s)$ and $\mathrm{cov}(Y_i, Y_j) = \Sigma_{ij}(s)$ where $\Sigma_{ii}(s)$ and $\Sigma_{ij}(s)$ can be calculated by (11). Similarly the Bonferroni inequality can be applied to the top-k list (see [3]).

5 Solution Relaxation

In this section we analytically evaluate the relation between the number of experiments m and the average number of correctly identified top-k nodes. We use the relaxation by allowing the latter number to be smaller than k. We aim

to mathematically justify the observed "80/20 behavior" of the algorithm: 80 percent of the top-k nodes are identified correctly in a very short time.

Let M_0 be a number of correctly identified elements in the top-k basket. In addition, denote by K_i the number of nodes ranked not lower than i. Formally, $K_i = \sum_{j \neq i} 1\{L_j \geq L_i\}, \quad i = 1, \ldots, k$, where $1\{\cdot\}$ is an indicator function. Placing node i in the top-k basket is equivalent to the event $\{K_i < k\}$, and thus $E(M_0) = E\left(\sum_{i=1}^{k} 1\{K_i < k\}\right) = \sum_{i=1}^{k} P(K_i < k)$. Direct evaluation of $P(K_i < k)$ is computationally intractable in realistic scenarios, even with Markov chain representation techniques [12]. Thus, we use *approximation* and *Poissonisation*.

The End Point algorithm is merely an occupancy scheme where each independent experiment (random walk) results in placing one ball (visit) to an urn (node of the graph). Under Poissonisation [14], we assume that the number of random walks is a Poisson random variable M with given mean m. Because the number of hits in the Poissonised model is different from the number of original hits, we use the notation Y_i instead of L_j for the number of visits to page j. Note that Y_j is a Poisson random variable with parameter $m\pi_j$ and is independent of Y_i for $i \neq j$. The imposed independence of Y_j's greatly simplifies the analysis.

Next to Poissonisation, we also apply *approximation* of M_0 by a closely related measure M_1: $M_1 = k - \sum_{i=1}^{k}(K_i'/k)$, where K_i' denotes the number of pages outside the top-k list that are ranked higher than node $i = 1, \ldots, k$. Note that K_i' is the number of mistakes with respect to node i that lead to errors in the identified top-k list. Then the sum in the definition of M_1 is simply the average number of such mistakes with respect to each of the top-k nodes.

The measure M_1 is more tractable than M_0 because its average value $E(M_1) = k - \frac{1}{k}\sum_{i=1}^{k} E(K_i')$ involves only the average values of K_i' and not their distributions, and because K_i' depends only on the nodes outside the top-k list. Then, we can make use of the following convenient measure $\mu(y)$:

$$\mu(y) := E(K_i'|Y_i = y) = \sum_{j=k+1}^{n} P(Y_j \geq y), \; i = 1, \ldots, k,$$

which implies $E(K_i') = \sum_{y=0}^{\infty} P(Y_i = y)\mu(y), \; i = 1, \ldots, k$. Therefore, we obtain the following expression for $E(M_1)$:

$$E(M_1) = k - \frac{1}{k}\sum_{y=0}^{\infty} \mu(y) \sum_{i=1}^{k} P(Y_i = y). \tag{15}$$

Illustrating example with Wikipedia (cont.): Let us calculate $E(M_1)$ for the top-10 basket corresponding to the seed node `Jim Jackson (ice hockey)`. Using formula (15), for $m = 8 \times 10^3; 10 \times 10^3; 15 \times 10^3$ we obtain $E(M_1) = 7.75; 9.36; 9.53$. It took 2000 runs to move from $E(M_1) = 7.75$ to $E(M_1) = 9.36$, but then 5000 runs is needed to advance from $E(M_1) = 9.36$ to $E(M_1) = 9.53$. We see that we obtain quickly 2-relaxation or 1-relaxation of the top-10 basket but then we need to spend a significant amount of effort to get the complete basket. This is indeed in agreement with the Monte Carlo runs (see

e.g., Figure 1). In the next theorem we explain this "80/20 behavior" and provide indication for the choice of m.

Theorem 4. *In the Poisonized End Point Monte Carlo algorithm, if all top-k nodes receive at least $y = ma > 1$ visits and $\pi_{k+1} = (1-\varepsilon)a$, $\varepsilon > 1/y$, then*

(i) to satisfy $E(M_1) > (1-\alpha)k$ it is sufficient to have

$$\sum_{j=k+1}^{n} \frac{(m\pi_j)^y}{y!} e^{-m\pi_j} \left[1 + \sum_{l=1}^{\infty} \frac{(m\pi_j)^l}{(y+1)\cdots(y+l)} \right] < \alpha k.$$

(ii) Statement (i) is always satisfied if $m > 2a^{-1}\varepsilon^{-2}[-\log(\varepsilon\pi_{k+1}\alpha k)]$.

Proof. See [3].

From (i) we can already see that the 80/20 behavior of $E(M_1)$ (and, respectively, $E(M_0)$) can be explained mainly by the fact that $\mu(y)$ drops drastically with y because the Poisson probabilities decrease faster than exponentially.

The bound in (ii) shows that m should be roughly of the order $1/\pi_k$. The term ε^{-2} is not defining since ε does not need to be small. For instance, by choosing $\varepsilon = 1/2$ we can filter out the nodes with PPR not higher than $\pi_k/2$. This often may be sufficient in applications. Obviously, the logarithmic term is of a smaller order of magnitude.

We note that the bound in (ii) is quite rough because in its derivation (see [3]) we replaced π_j, $j > k$, by their maximum value π_{k+1}. In realistic examples, m can be chosen much smaller than in (ii) of Theorem 4. In fact, in our examples good top-k baskets are obtained if the algorithm is terminated at the point when for some y, each node in the current top-k basket has received at least y visits while the rest of the nodes have received at most $y - d$ visits, where d is a small number, say $d = 2$. Such choice of m satisfies (i) with reasonably small α. Without a formal justification, this stopping rule can be understood since we have $m\pi_{k+1} = ma(1-\varepsilon) \approx ma - d$, which results in a small value of $\mu(y)$.

Acknowledgments. We would like to thank Brigitte Trousse for her very helpful remarks and suggestions.

References

1. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: Proceedings of FOCS 2006, pp. 475–486 (2006)
2. Avrachenkov, K., Litvak, N., Nemirovsky, D., Osipova, N.: Monte Carlo methods in PageRank computation: When one iteration is sufficient. SIAM Journal on Numerical Analysis 45(2), 890–904 (2007)
3. Avrachenkov, K., Litvak, N., Nemirovsky, D., Smirnova, E., Sokol, M.: Monte Carlo Methods for Top-k Personalized PageRank Lists and Name Disambiguation, INRIA Research Report no.7367 (2010)
4. Avrachenkov, K., Dobrynin, V., Nemirovsky, D., Pham, S.K., Smirnova, E.: PageRank Based Clustering of Hypertext Document Collections. In: Proceedings of ACM SIGIR 2008, pp. 873–874 (2008)

5. Bahmani, B., Chowdhury, A., Goel, A.: Fast Incremental and Personalized PageRank. In: Proceedings of VLDB Endow. (2010)
6. Barakat, C., Iannaccone, G., Diot, C.: Ranking flows from sampled traffic. In: Proceedings of CoNEXT 2005 (2005)
7. Berkhin, P.: Bookmark-Coloring Algorithm for Personalized PageRank Computing. Internet Mathematics 3, 41–62 (2006)
8. Boldi, P., Vigna, S.: The WebGraph framework I: Compression techniques. In: Proceedings of the 13th International World Wide Web Conference (WWW 2004), pp. 595–601 (2004)
9. Breyer, L.A.: Markovian Page Ranking distributions: Some theory and simulations, Technical Report (2002), http://www.lbreyer.com/preprints.html
10. Brin, S., Page, L., Motwami, R., Winograd, T.: The PageRank citation ranking: bringing order to the Web, Stanford University Technical Report (1998)
11. Chakrabarti, S.: Dynamic Personalized PageRank in entity-relation graphs. In: Proceedings of WWW 2007 (2007)
12. Corrado, C.J.: The exact joint distribution for the multinomial maximum and minimum and the exact distribution for the multinomial range, SSRN Research Report (2007)
13. Fogaras, D., Rácz, B., Csalogány, K., Sarlós, T.: Towards scaling fully personalized Pagerank: Algorithms, lower bounds, and experiments. Internet Mathematics 2(3), 333–358 (2005)
14. Gnedin, A., Hansen, B., Pitman, J.: Notes on the occupancy problem with infinitely many boxes: general asymptotics and power laws. Probability Survyes 4, 146–171 (2007)
15. Haveliwala, T.: Topic-Sensitive PageRank. In: Proceedings of WWW 2002 (2002)
16. Jeh, G., Widom, J.: Scaling personalized web search. In: Proceedings of WWW 2003 (2003)
17. Johnson, K.L., Kotz, S., Balakrishnan, N.: Discrete Multivariate Distributions. Wiley, New York (1997)
18. Kemeny, J., Snell, J.: Finite Markov Chains. Springer, Heidelberg (1976)
19. Langville, A.N., Meyer, C.D.: Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, Princeton (2006)
20. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: Proceedings of CIKM 2003 (2003)
21. Smirnova, E., Avrachenkov, K., Trousse, B.: Using Web Graph Structure for Person Name Disambiguation. In: Proceedings of CLEF/WEPS 2010 (2010)
22. Tanabe, K., Sagae, M.: An exact Cholesky Decomposition and the generalized inverse of the variance-covariance matrix of the multinomial distribution, with applications. Journal of the Royal Statistical Society (Series B) 54(1), 211–219 (1992)
23. Voevodski, K., Teng, S.H., Xia, Y.: Spectral affinity in protein networks. BMC Systems Biology 3, 112 (2009)
24. Zlatic, V., Bozicevic, M., Stefancic, H., Domazet, M.: Wikipedias: Collaborative web-based encyclopedias as complex networks. Phys. Rev. E 74 (2006)

Rank-Based Models of Network Structure and the Discovery of Content

Adam Douglas Henry and Paweł Prałat

[1] Division of Public Administration, West Virginia University, Morgantown, WV 26506-6322, USA
[2] Department of Mathematics, West Virginia University, Morgantown, WV 26506-6310, USA

Abstract. Research on self-organizing networks, especially in the context of the Web graph, holds great promise to understand the complexity that underlies many social systems. We argue that models of social network structure should begin to consider how structure arises from the "content" of networks, a term we use to describe attributes of network actors that are independent of their structural position, such as skill, intelligence, or wealth. We propose a rank model of how content (operationalized as attribute rank relative to other individuals) may change amongst agents over time within a stochastic system. We then propose a model of network self-organization based on this rank model. Finally, we demonstrate how one may make inferences about the content of networks when attributes are unobserved, but network structures are readily measured. This approach holds promise to enhance our study of social interactions within the Web graph and in complex social networks in general.

1 Why Network Content Matters

Research on the Web graph has been very influential in social science research regarding the structure and function of complex social networks. While the structure and emergence of networks has been a long-standing theme in disciplines such as sociology [1] [2], political science [3] [4], and economics [5], coupling theoretical models with rigorous models of network self-organization (e.g., [6] and [7]) is still an emerging area of research. Indeed, research on complex networks, especially in the context of the Web graph, has broad applicability in the social sciences and can help to inform methods to unpack the complexity that characterizes many social systems.

At the same time, the social sciences can also contribute to modeling work in mathematics and computer science, since it offers concrete theories about the factors that drive network relationships. Thus, social science theory can help to discipline researchers' focus on particular models that are likely to be more realistic in particular contexts. In research on the Web graph it is important to focus on social drivers of network structure since the Web is, after all, a self-organizing network created and manipulated by human beings. Interactions within the Web

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 62–73, 2011.

graph are both a direct reflection of human behavior (e.g., when organizations decide to reference one another due to shared interests or resources), but also hold promise as indicators for latent forms of socially-relevant relations such as trust or agreement [8].

In this paper we argue that modeling work on social networks should take seriously the role of network content—meaning the inherent attributes of network actors [9]—in driving network self-organization. Some research has begun to do this by asking, for example, how network structures are influenced by the fitness of actors [10], strategies [11], or the spatial positioning of agents [12]. These types of "content" models of network structure are important supplements to classical modeling approaches that emphasize the importance of structural drivers such as node degree or other measures of centrality. This is because many social science theories are ultimately concerned with attributes of individual actors—why are some powerful and others marginalized, why do political organizations behave the way they do, and how are behaviors, norms, or beliefs learned from others within a network. Thus, to understand complex social networks one must consider structure, but also how structure is dependent upon, and co-evolves with, network content. This will allow researchers to move towards coherent theories of emergent behaviors within social networks.

We contribute to this endeavor in two ways. First, this paper posits a simple, mathematically tractable, yet reasonable model of network self-organization that accounts for the ways in which network content drives network structure. This is a contribution in of itself, and builds heavily upon earlier modeling work in this area by Łuczak, Prałat, Wormald (e.g., [13], [14], [15], [16]) and especially by Prałat and Janssen (e.g., [17] [18]). The model outlined here is a "rank" model where link formation probabilities are based on externally-determined prestige labels relative to other agents in the system; this general approach was first proposed by Fortunato, Flammini and Menczer in [19]. Thus, this paper is concerned with at least preliminary models of network self-organization.

Second, and more importantly, we investigate how this network model may be used to estimate network content—that is, the rank of nodes—based on observed structure alone. This is an important area for research, since in many applications of social network analysis we may know the structure of the network (for example, if networks amongst organizations are measured using hyperlink data), but attributes of actors remain a latent, unobserved variable. Our research builds on prior work to estimate node attributes from observed structure [12], although this research involved a different model and was focused on predicting distances between nodes rather than the attributes of the nodes themselves. We find that making inferences about node ranks is eminently doable, and this research establishes a baseline for methods of statistically inferring node attributes from network structure only. We illustrate the use of this approach through computational simulation, which provides a starting point for future work emphasizing mathematical proof.

The progression of this paper is as follows. We first discuss how the content of a network may be thought of in terms of the rank of vertices—while this is

just one of many possible approaches, it has direct applicability to models of the Web graph and of attendant social network structures as well. The model we consider is stochastic, involving the random entry into and exit from the system of vertices over time, and we present some essential results regarding the shifting of ranks over time using the differential equations method [20]. We then overlay a network model on top of the basic ranking model, which provides a starting point for thinking about how network relations are chosen based on rank. We then present the results of simulations that show how we may determine the content (rank) of vertices based on observations of the structure only. While simulations are used here justify the essential prediction method, future versions of this paper will present rigorous results through mathematical proof. Moreover, due to space limitations, proofs of theorems stated in this paper have been omitted but will be included in a future version.

2 A Rank Model of Content

In this section, we formally define a ranking model that reflects the "content" of actors within a hypothetical social system. This model not only specifies the process by which attributes are assigned to individual agents, but also specifies the way in which these attributes shift over time as actors enter or exit the system. Our focus is on modeling systems where the total number of actors is large but fixed (for example, if at each time step an agent is removed uniformly at random and immediately replaced by a new one). This type of behavior is most consistent for well-established systems. Such stochastic systems are also usually more challenging to model than, say, systems that are "young" or "middle-aged" and hence growing over time, with agents being added to the system at a faster rate than they are removed.

2.1 Model Overview

At each time t, we have exactly n objects in a set V_t. Moreover, at each time t, each object $v \in V_t$ has rank $r_t(v) \in [n]$ (we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$). In order to obtain a proper ranking, the rank function $r_t : V_t \to [n]$ is a bijection for all t, so every object has a unique rank. In agreement with the common use of the word "rank", high rank refers to a object v for which $r_t(v)$ is small: the highest ranked object is ranked number one, so has rank equal to 1; the lowest ranked object has rank n. The initialization and update of the ranking is done according to a *ranking scheme*. Various ranking schemes can be considered, and might lead to different behavior. We first give the general model, and then list a few natural ranking schemes.

The model produces a sequence $\{(V_t, r_t)\}_{t=0}^{\infty}$ of sets V_t of n objects and ranking functions r_t, where t denotes time. To initialize the model, let V_0 be any set of n objects and let r_0 be any initial rank function $r_0 : V_0 \to [n]$ which is consistent with the ranking scheme. For $t \geq 1$ we form (V_t, r_t) from (V_{t-1}, r_{t-1}) according to the following rules:

(i) Choose uniformly at random an object $u_t \in V_{t-1}$ and delete it.
(ii) Add a new object v. (We refer to the time step t in which object v was added as time in which v was born.)
(iii) Assign an initial rank to v, update V_t and the ranking function $r_t : V_t \to [n]$ according to the ranking scheme.

One can define a number of different ranking schemes. In this paper, we focus on the random initial rank scheme but the concept of the ranking by age will also be important. Therefore, let us define the following two schemes. In order to distinguish them, we will use a_t for the ranking by age and r_t for the random initial rank.

(i) **Ranking by age:** The newly added object v obtains an initial rank n; its rank decreases by one each time an object with smaller rank is removed. Formally, for each $v \in V_{t-1} \setminus \{u_t\}$, $a_t(v) = a_{t-1}(v) - \gamma$, where $\gamma = 1$ if the rank of the object deleted in step t is smaller than $a_{t-1}(v)$, and 0 otherwise.
(ii) **Random initial rank:** The object added at time t obtains an initial rank R_t which is randomly chosen from $[n]$ according to a prescribed distribution. Ranks of all objects are adjusted accordingly. Formally, for each $v \in V_{t-1} \setminus \{u_t\}$, $r_t(v) = r_{t-1}(v) + \delta - \gamma$, where $\delta = 1$ if $r_{t-1}(v) > R_t$ and 0 otherwise, and $\gamma = 1$ if where the rank of u_t, the object deleted in step t, is smaller than $r_{t-1}(v)$, and 0 otherwise.

The results are generally about the behavior of ranking functions, where the asymptotics are based on n tending to infinity. We say that an event holds *asymptotically almost surely* (*aas*), if it holds with probability tending to one as $n \to \infty$. We will sometimes use the stronger notion of *wep* in favour of the more commonly used *aas*, since it simplifies some of our proofs. We say that an event holds *with extreme probability* (*wep*), if it holds with probability at least $1 - \exp(-\Theta(\log^2 n))$ as $n \to \infty$. Thus, if we consider a polynomial number of events that each holds *wep*, then *wep* all events hold. To combine this notion with asymptotic notations such as $O()$ and $o()$, we follow the conventions in [21].

The coupon collector problem can give us insight into when all objects from the initial set V_0 will be deleted. Namely, let $L = n(\log n + \omega(n))$ where $\omega(n)$ is any function tending to infinity with n. It is a well-known result that *aas* after L steps all original objects will have been deleted.

2.2 Ranking by Age

To understand the influence of age, we need to understand the behavior of the age rank function $a_t(v)$ defined before (in short, $a_t(v) - 1$ equals the number of objects in V_t that were born earlier than v). We assume (without loss of generality) that v was born at time 0, so $a_0(v) = n$. For $t > 0$, $a_t(v)$ decreases by one precisely when in time step $t+1$, the object u which is deleted was older than v, so $a_t(u) < a_t(v)$. We obtain that

$$\mathbb{E}(a_{t+1}(v) - a_t(v) \mid G_t) = -\frac{a_t(v) - 1}{n - 1},$$

conditional on the fact that v is not deleted. To analyze this random variable, we use the differential equations method. Defining a real function $z(x)$ to model the behaviour of $a_{xn}(v)/n$, the above relation implies the following differential equation

$$z'(x) = -z(x)$$

with the initial condition $z(0) = 1$.

The general solution is $z(x) = \exp(-x+C)$, $C \in \mathbb{R}$ and the particular solution is $z(x) = \exp(-x)$. This *suggests* that a random variable $a_t(v)$ should be close to the deterministic function $n\exp(-t/n)$. The following theorem precisely states the conditions under which this holds. This theorem is proved in [17].

Theorem 1. *Let $a_t(v)$ be the age rank of object v at time t. Then* wep*, for every t in the range $0 \le t \le t_f = \frac{1}{2}n\log n - 2n\log\log n$, we have*

$$a_t(v) = n\exp(-t/n)(1 + O(\log^{-1/2} n))$$

conditional upon the object v surviving until time t_f.

2.3 Randomly Chosen Initial Rank

In this section, we consider the case where the rank R_i of the object v added at time i is chosen at random from $[n]$. The ranks of existing objects are adjusted accordingly. We make the assumption that all initial ranks are chosen according to the same distribution. In particular, we fix a continuous bijective function $F : [0,1] \to [0,1]$, and for all integers $1 \le k \le n$, we let

$$\mathbb{P}(R_i \le k) = F\left(\frac{k}{n}\right).$$

Thus, F represents the limit, for n going to infinity, of the cumulative distribution functions of the variables R_i. To simplify the calculations while exploring a wide array of possibilities for F, we assume F to be of the form

$$F(x) = \begin{cases} (2x)^s/2 & \text{if } 0 \le x \le 1/2 \\ 1 - (2(1-x))^s/2 & \text{if } 1/2 < x \le 1 \end{cases}, \text{ where } s \ge 1.$$

This distribution has the advantage of allowing us to generalize our results to a broad class of realistic initial distributions, ranging from situations where initial ranks are distributed uniformly at random (when $s = 1$) to situations where agents enter the system with a mediocre rank with higher probability (when $s > 1$; in this case the highest probability rank is $n/2$). See Figure 1 (a) to see the differences in distributions across $s = 1.0$, $s = 1.2$, and $s = 1.5$. This functional form is reasonable because it reflects the notion that many types of attributes follow a Normal distribution in social systems; it tends to be unlikely

that new agents will be "born" into the system with a very low rank or a very high rank. If rank represents a type of dynamic fitness where agents compete for better ranks, entering agents are unlikely to have very poor ranks because then they may not be able to enter the system at all, and they are also unlikely to enter with very good ranks, which are obtained only through a history of competition in the system. Our functional form for $F(x)$ reflects these possibilities.

Case $s = 1$: The case $s = 1$ represents the uniform distribution of the R_i. The random variable $r_t(v)$ is sharply concentrated around the initial rank R_i. The following result was obtained in [17].

Lemma 1. *Suppose that object v obtained an initial rank $R \geq \sqrt{n}\log^2 n$. Then,* wep

$$r_t(v) = R(1 + O(\log^{-1/2} n))$$

to the end of its life.

Case $s > 1$: In this case, the initial rank is biased towards the middle range ranks. The rank function exhibits more complex behaviour in this case. Due to the symmetry of the function $F(x)$, without loss of generality we can assume that an initial rank is at most $\frac{n}{2}$. For ranks close to $\frac{n}{2}$ we clearly cannot predict the behaviour; the final rank can be bigger or smaller than the initial rank. However, if the initial rank is separated a bit from the middle rank, then we get a concentration.

Theorem 2. *Suppose that an object v obtained an initial rank*

$$r_0(v) = R < \frac{n}{2} - \sqrt{n}\log^2 n$$

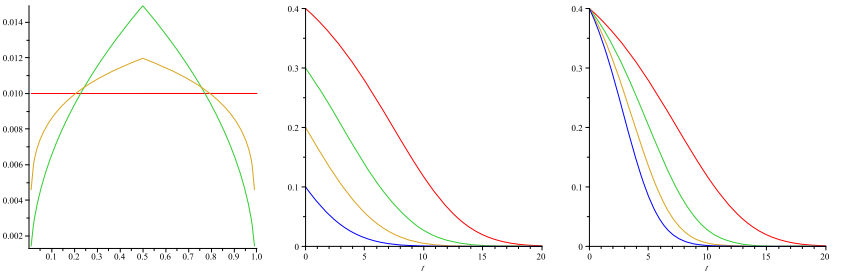
at time 0. *Then* wep, *for every t in the range $0 \leq t \leq t_f = \frac{1}{2}n\log n - 2n\log\log n$ conditional upon the object v surviving until time t,*

$$r(v,t) = \frac{n}{2}\left(\left(\left(\frac{2R}{n}\right)^{1-s} - 1\right)e^{(s-1)t/n} + 1\right)^{\frac{1}{1-s}}(1 + O(\log^{-1/2} n)) \quad (1)$$

provided

$$\frac{n}{2}\left(\left(\left(\frac{2R}{n}\right)^{1-s} - 1\right)e^{(s-1)t/n} + 1\right)^{\frac{1}{1-s}} \geq \sqrt{n}\log^2 n\,.$$

Figure 1 (b),(c) presents the behaviour of different initial ranks for one specific value of $s = 1.2$ as well as the behaviour of one specific initial rank $R = 0.4n$ for different values of s. (Both rank and time is scaled by n.)



(a) $s = 1.0, 1.2, 1.5$ (b) $R = 0.1, 0.2, 0.3, 0.4$ (c) $s = 1.2, 1.3, 1.4, 1.5$

Fig. 1. (a) Different distributions: $f(x) = \mathbb{P}\left(x \leq R/n \leq x + \frac{1}{100}\right)$; (b) The behaviour for different initial ranks ($s = 1.2$); (c) The behaviour for different values of s ($R = 0.4$)

3 A Rank Model of Network Structure

In this section, we introduce the network on top of the process discussed in previous sections. We need two more parameters, the *attachment strength* $\alpha \in (0, 1)$ and *initial degree* $d \in \mathbb{N}$. This time, the model produces a sequence $\{(G_t, r_t)\}_{t=0}^{\infty}$ of graphs $G_t = (V_t, E_t)$ on n vertices and ranking functions $r_t : V_t \to [n]$. To initialize the model, let G_0 be any graph on n vertices and let r_0 be any initial rank function $r_0 : V_0 \to [n]$ which is consistent with the ranking scheme. For $t \geq 1$ we form G_t from G_{t-1} according to the following rules:

(i) Choose uniformly at random a vertex $u_t \in V_{t-1}$ and delete it.
(ii) Add a new vertex v_t together with d edges from v_t to existing vertices chosen randomly with weighted probabilities. The edges are added in d substeps. In each substep, one edge is added, and the probability that v_i is chosen as its endpoint (the link probability), is proportional to $r_{t-1}(v_i)^{-\alpha}$.
(iii) Assign an initial rank to v_t, update V_t and the ranking function $r_t : V_t \to [n]$ according to the ranking scheme.

In [17], it has been shown that the uniform distribution for the initial rank (that is, the specific case of $s = 1$ in our model) generates *wep* a power-law degree distribution with exponent $1 + 1/\alpha$. Here, we will show that it is also the case for $s > 1$. However, there is a constant factor difference. Let Z_k denote the number of vertices of degree k, and $Z_{\geq k} = \sum_{l \geq k} Z_l$.

Theorem 3. *Let* $0 < \alpha < 1$ *and* $d \in \mathbb{N}$, $\log^4 n \leq k \leq n^{\alpha/2} \log^{-3\alpha} n$. *Then* wep

$$Z_{\geq k} = (1 + o(1))2^{1-s} \left(\frac{d(1-\alpha)}{k(1+\alpha)}\right)^{1/\alpha} n.$$

The proof is a consequence of the following result.

Theorem 4. *Let* $0 < \alpha < 1$, $d \in \mathbb{N}$, $i = i(n) \in [n]$, *and let* v_i *be the vertex whose age rank at time* L *equals* $a(v_i, L) = i = xn$. *Let* R *be the initial rank of* v_i, *and assume that* $\sqrt{n}\log^2 n < R < \frac{n}{2} - \sqrt{n}\log^2 n$. *Then the expected degree of* v_i *is given by*

$$\mathbb{E}\deg(v_i, L) = (1 + O(\log^{-1/2} n))\frac{d(1-\alpha)2^\alpha}{1+\alpha}\left(\left(\frac{2R}{n}\right)^{1-s} - 1\right)^{\frac{-\alpha}{1-s}}\left(x^{-\alpha} - x\right),$$

provided $x = o(1)$ *or* $R/n = o(1)$; *otherwise* $\mathbb{E}\deg(v_i, L) = O(1)$. *Moreover, if* $\mathbb{E}\deg(v_i, L) \geq \log^4 n$, *then* wep

$$\deg(v_i, L) = \mathbb{E}\deg(v_i, L) + O(\sqrt{\mathbb{E}\deg(v_i, L)}\log n),$$

and if $\mathbb{E}\deg(v_i, L) < \log^4 n$, *then* wep $\deg(v_i, L) = O(\log^4 n)$.

4 The Discovery of Content through Structure

While the Web graph is a useful platform for social networks research, the notion that networks self-organize as a function of network content suggests the need to observe both structure as well as attributes of the nodes. While structures may be observed directly, for example through hyperlink data, in most cases attributes (ranks) of agents embedded in the network are latent, unobserved variables. However, given a realistic model of the process by which the network was generated, it is possible to infer likely attributes of network agents.

Consider, for example, the degree of a given node. Given the model outlined here, this degree is a function of two factors related to content: first, the length of time the node has been in the system, and second, the initial rank assigned to the node when it was "born" into the network. Agents with smaller initial ranks tend to have larger degrees, and older vertices also tend to have larger degrees. Despite this correlation, however, the true relationship is quite complicated and it would seem to be a lost cause to try to infer only one of these attributes (age or rank) based on degree only. Fig 3 presents the relation between age and degree for vertices of degree at least $d/2$ when networks are simulated according to the model described here ($n = 20,000$, $d = 100$, $s = 1.5$, and $\alpha = 0.8$). Young vertices have small degree (there is no time to accumulate neighbours, even if the initial rank is good) but old vertices can still have small degree (because they have an unattractive rank).

As noted above, networks generated according to this rank model are characterized by power-law degree distributions, which is readily observed within simulated networks. Fig 2 presents the cumulative degree distribution on a log-log scale: $y(x)$ is the number of vertices of degree at least x.

It turns out, however, that it becomes feasible to estimate these properties when we broaden our focus from the degree of a single agent to properties of their second neighborhood. Consider, for example, the following coefficient defined

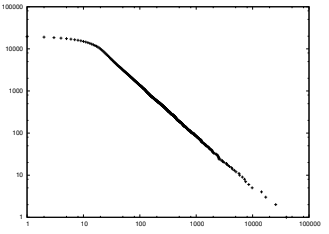
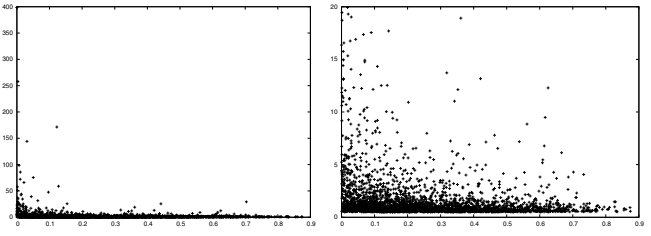


Fig. 2. Power-law degree distribution generated by the rank model



(a) (b)

Fig. 3. Degree of v vs. the age rank of v (rescaled)

for a vertex v of non-zero degree that is proportional to the average degree of neighbours of v:

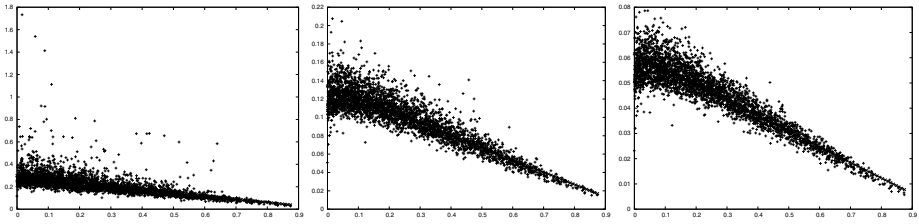
$$c_2(v) = \frac{\sum_{u \sim v} \deg(u)}{\deg(v)}.$$

We put $c_2(v) = 0$ if $\deg(v) = 0$. Clearly old nodes have more old neighbours compared to younger nodes. In other words, there is a correlation between the age of v and ages of its neighbours. On the other hand, ranks are generated independently, so a distribution of ranks of the neighbours of v should be similar to the distribution we use in the model. The more neighbours v has, the better correlation we should see. Older vertices should have larger coefficients $c_2(v)$'s. See Fig. 4(a) for the relation for vertices of degree at least $d/2$.

This process can even be carried further to develop even more finely-tuned estimates of agents' unobserved attributes. We can take a look at third, fourth, and higher-order neighborhoods by defining, recursively, for $i \geq 3$

$$c_i(v) = \frac{\sum_{u \sim v} c_{i-1}(u)}{\deg(v)},$$

provided that $\deg(v) > 0$; otherwise, $c_i(v) = 0$. Again, in this case older vertices should have larger coefficients and the error should decrease for, say, $i = 3$ and $i = 4$. See Fig. 4(b-c) for the results for $c_3(x)$ and $c_4(x)$.



(a) $c_2(x)$ (b) $c_3(x)$ (c) $c_4(x)$

Fig. 4. $c_i(x)$ vs. the age rank of v

Even upon casual examination, these scatterplots reveal a strong, nearly linear, relationship between the average degree of neighbors (or higher-order c_i coefficients) and the age of node. Inferring rank rather than age may be accomplished in a similar way.

One way of viewing the increasing predictive power of these structural characteristics is to perform an simple OLS linear regression with node age as the dependent variable (the unobserved variable to be inferred in real-world applications) and degree or various c_i measures as possible independent variables. Using this approach, we find that predicting age as a linear function of average degree becomes more precise as we move to higher-order neighborhoods. For example, the R^2 statistic when age is predicted using degree only is 0.01, meaning that node degree explains only 1% of the variance in actual node age. When the average degree of neighbor (c_2) is used as an independent variable, a linear model explains 35% of the variance on age ($R^2 = 0.35$). R^2 jumps to 0.77 for c_3, and 0.83 for c_4.

These regression models provide at least heuristic evidence that one can achieve fairly accurate predictions of age when one examines the degree of neighbors, and neighbors of neighbors, and so forth. And while these linear models are suggestive of strong patterns, the scatterplots also make it clear that the accuracy with which we can predict age depends on the degree of the node. In particular, it seems that for low-degree nodes age may be predicted with fair accuracy (in particular because, having just entered the system, the number of relationships is a more direct result of initial rank) while the relationship between c_i and age for high-degree nodes is less precise.

It is also interesting to note that going from the second neighborhood to the third neighborhood provides a smaller marginal benefit in terms of predictive power, as measured by the R^2 values. While examination of the third neighborhood provides the strongest inferences regarding age, of course there will be an upper bound on the "depth" of neighborhoods that may be examined, plus there is likely to be an optimal neighborhood to examine in terms of maximizing the predictive power of this method. These issues, along with the strength of predictions that may be made for small- versus high-degree agents, will be sorted out through mathematical proof in a journal version of this paper.

5 Conclusion

This paper outlined a model of network self-organization that is driven by the "ranks" of individual agents in terms of an arbitrary attributes that are inherently individual phenomenon, such as wealth, power, beliefs, skills, or any other actor-level variables that are likely to play an important role in networking behavior. This is a stochastic model involving the formation and deletion of network ties, and adjustment in ranks, as actors dynamically enter and exit the system over time.

This research builds upon prior work in ranking and associated models of network self-organization, and continues the enterprise of linking these network models to enhance our understanding of the dynamics of real-world social systems. An important area for future research is to carefully consider how network structure evolves as a function of network content. Of course, this not only requires models of networks per se, but also requires models of attributes of individuals and how these attributes are manifest in network structure. The World Wide Web provides an excellent platform for the study of such networks because it yields large-scale, high-quality network data that contains traces of real-world interactions amongst social or political agents.

On the other hand, network content is often exceedingly difficult to observe and can be a limiting factor on our ability to study complex, self-organizing social networks. However, given realistic models of how network structure is driven by content, it seems that we are able to make reasonable inferences regarding the attributes of individuals based on structure only. This research provides a platform for more research, emphasizing analytical proof, the exploits the potential "reversibility" of mathematical models to infer latent, unobserved variables that are crucial to the development of network theory in the social sciences.

References

1. Podolny, J.M., Page, K.L.: Network Forms of Organization. Annual Review of Sociology 24, 57–76 (1998)
2. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
3. Thatcher, M.: The Development of Policy Network Analysis: From Modest Origins to Overarching Frameworks. Journal of Theoretical Politics 10(4), 389–416 (1998)
4. Laumann, E.O., Knocke, D.: The Organizational State: Social Choice in National Policy Domains. University of Wisconsin Press, Madison (1987)
5. Jackson, M.O.: Social and Economic Networks. Princeton University Press, Princeton (2008)
6. Bonato, A.: A Course on the Web graph, Providence, Rhode Island. American Mathematical Society Graduate Studies Series in Mathematics (2008)
7. Chung, F.R.K., Lu, L.: Complex graphs and networks. American Mathematical Society, U.S.A (2004)
8. Henry, A.D.: Measuring Social Networks Using the World Wide Web. In: American Association for the Advancement of Science Annual Meeting, Chicago, Illinois (2009)

9. Ingold, K.M.: How Network Structures Influence Policy Outputs. Policy Studies Journal (forthcoming)
10. Borgs, C., Chayes, J.T., Daskalakis, C., Roch, S.: First to Market is not Everything: an Analysis of Preferential Attachment with Fitness. In: Proceedings of the 39rd Annual ACM Symposium on the Theory of Computing (STOC), pp. 135–144 (2007)
11. Skyrms, B., Pemantle, R.: A Dynamic Model of Social Network Formation. Proceedings of the National Academy of Sciences 97(16), 9340–9346 (2000)
12. Janssen, J., Prałat, P., Wilson, R.: Estimating node similarity from co-citation in a spatial graph model. In: Proceedings of the 2010 ACM Symposium on Applied Computing (SAC) – Special Track on Self-organizing Complex Systems, pp. 1329–1333 (2010)
13. Łuczak, T., Prałat, P.: Protean graphs. Internet Mathematics 3, 21–40 (2006)
14. Prałat, P.: A note on the diameter of protean graphs. Discrete Mathematics 308, 3399–3406 (2008)
15. Pralat, P.: Connectivity threshold and recovery time in rank-based models for complex networks. To appear in Discrete Mathematics
16. Prałat, P., Wormald, N.: Growing protean graphs. Internet Mathematics 4, 1–16 (2009)
17. Janssen, J., Prałat, P.: Protean graphs with a variety of ranking schemes. Theoretical Computer Science 410, 5491–5504 (2009)
18. Janssen, J., Prałat, P.: Rank-based attachment leads to power law graphs. SIAM Journal on Discrete Mathematics 24, 420–440 (2010)
19. Fortunato, S., Flammini, A., Menczer, F.: Scale-free network growth by ranking. Phys. Rev. Lett. 96(21), 218701 (2006)
20. Wormald, N.: The differential equation method for random graph processes and greedy algorithms. In: Karoński, M., Prömel, H.J. (eds.) Lectures on Approximation and Randomized Algorithms, pp. 73–155. PWN, Warsaw (1999)
21. Wormald, N.C.: Random graphs and asymptotics. In: Gross, J.L., Yellen, J. (eds.) Handbook of Graph Theory, Section 8.2, pp. 817–836. CRC, Boca Raton (2004)
22. Janson, S., Łuczak, T., Ruciński, A.: Random Graphs. Wiley, New York (2000)
23. Pittel, B., Spencer, J., Wormald, N.: Sudden emergence of a giant k-core in a random graph. J. Combinatorial Theory, Series B 67, 111–151 (1996)

1-Local 33/24-Competitive Algorithm for Multicoloring Hexagonal Graphs

Rafał Witkowski[1,⋆] and Janez Žerovnik[2,⋆⋆]

[1] Adam Mickiewicz University,
ul. Umultowska 87
61-614 Poznań, Poland
rmiw@amu.edu.pl
[2] University of Ljubljana,
Aškerčeva 6,
SI-1000 Ljubljana, Slovenia
janez.zerovnik@fs.uni-lj.si, janez.zerovnik@imfm.si

Abstract. In the frequency allocation problem, we are given a cellular telephone network whose geographical coverage area is divided into cells, where phone calls are serviced by assigned frequencies, so that none of the pairs of calls emanating from the same or neighboring cells is assigned the same frequency. The problem is to use the frequencies efficiently, i.e. minimize the span of frequencies used. The frequency allocation problem can be regarded as a multicoloring problem on a weighted hexagonal graph, where each vertex knows its position in the graph. We present a 1-local 33/24-competitive distributed algorithm for multicoloring a hexagonal graph, thereby improving the previous 1-local 7/5-competitive algorithm.

1 Introduction

A fundamental problem concerning cellular networks is to assign sets of frequencies (colors) to transmitters (vertices) in order to avoid unacceptable interferences [2]. The number of frequencies demanded at a transmitter may vary between transmitters. In a usual cellular model, the transmitters are the centers of hexagonal cells and the corresponding adjacency graph is an induced subgraph of the infinite triangular lattice. An integer $d(v)$ is assigned to each vertex of the triangular lattice and will be called the *demand* (or *weight*) of the vertex v. The vertex weighted graph induced by the subset of the triangular lattice of vertices of positive demand is called a (vertex weighted) *hexagonal graph*. The phenomenon of hexagonal graphs and its multicoloring arises naturally in studies of cellular networks.

A *proper multicoloring* of G is a mapping f from $V(G)$ to subsets of integers such that $|f(v)| = d(v)$ for any vertex $v \in V(G)$ and $f(v) \cap f(u) = \emptyset$ for any pair

⋆ This work was supported by grant N206 017 32/2452 for years 2007-2010 and N206 1842 33 for years 2011-2014.
⋆⋆ Supported in part by ARRS, the research agency of Slovenia.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 74–84, 2011.

of adjacent vertices u and v in the weighted graph G. The minimum number of colors needed for a proper multicoloring of G, $\chi_m(G)$, is called the *multichromatic number*. Another invariant of interest in this context is the *(weighted) clique number*, $\omega(G)$, see formal definition in Section 2. It is well known that $\chi_m(G) \geq \omega(G)$ and that it is NP-complete problem to decide whether $\chi_m(G) = \omega(G)$ [6]. There was a lot of work done on finding approximation algorithms that imply upper bounds for the multichromatic number of hexagonal graphs.

An assumption that naturally arises in hexagonal graphs which model cellular networks is that each vertex is aware of its location. In this paper so called k-local algorithms for multicoloring hexagonal graphs are studied. An algorithm is *k-local* if the computation at any vertex v uses only the information about the demands of vertices at distance at most k from v.

A framework for studying distributed online assignment in cellular networks was developed in [5]. A distinction between *online* and *offline* algorithms was introduced and the definition of *p-competitive algorithm* was given as well. In the offline version of the problem the demands are fixed and known in advance while in the online version the demands may change over time, motivated by the fact that the number of calls within the cell in the network changes. Here we will consider only the offline version as we develop an algorithm that multicolors a hexagonal graph with known demands at vertices. Online version is only mentioned in the corollary that follows directly using result of [5]. Finally, an algorithm is p-competitive if it uses at most p times as many colors (frequencies) overall as the optimal offline algorithm would. In the same paper [5], a 3/2-competitive 1-local, 17/12-competitive 2-local and 4/3-competitive 4-local algorithms were outlined. Later, a 4/3-competitive 2-local algorithm was developed in [9]. The best ratio for 1-local case was first improved to 13/9 [1], and later to 17/12 [12] and to 7/5 [13]. In this paper we develop a new 1-local algorithm which uses no more than $\frac{33}{24}\omega(G) + O(1)$ colors, implying the existence of a 33/24-competitive algorithm.

It may be worth mentioning that the approximation bound for multicoloring algorithms on hexagonal graphs $\chi_m(G) \leq (4/3)\omega(G) + O(1)$ [6,8,9] is still the best known, both for distributed and not distributed models of computation. In view of this one can naturally take 4/3 as (maybe too ambitious) goal ratio for 1-local algorithms. With this assumption, our improvement from 7/5 to 33/24 can be calculated by the following formula

$$\frac{\frac{7}{5} - \frac{33}{24}}{\frac{7}{5} - \frac{4}{3}} = \frac{3}{8} = 37.5\%$$

which is a considerable improvement. Evaluating the previous improvements from 3/2 to 13/9 to 17/12 to 7/5 by the same formula gives 33.3%, 25% and 20%, respectively.

Our algorithm substantially differs from the algorithms in [1] and [9] which are composed of two stages. At the first stage, a triangle-free hexagonal graph with weighted clique number no larger than $\lceil \omega(G)/3 \rceil$ is constructed from G, while at the second stage an algorithm for multicoloring a triangle-free hexagonal graph

is used (see [1], [4], [10], [14]). Our improvement is based on the idea to borrow some colors used in the first stage and to use them for the demands of the second stage (see [13]). This in particular implies that the second stage of our algorithm cannot be applied as a stand-alone algorithm for the multicoloring arbitrary triangle-free hexagonal graphs.

The main result of this paper is

Theorem 1. *There is a 1-local distributed approximation algorithm for multicoloring hexagonal graphs which uses at most* $\frac{33}{24}\omega(G) + O(1)$ *colors. Time complexity of the algorithm at each vertex is constant.*

In [5] it was proved that a k-local c-approximate offline algorithm can be easily converted to a k-local c-competitive online algorithm, so we have:

Corollary 1. *There is a 1-local 33/24-competitive online algorithm for multicoloring hexagonal graphs.*

The paper is organized as follows: in the next section we formally define some basic terminology. In Section 3 we present an overview of the algorithm, while in Section 4 we provide a proof of Theorem 1.

2 Basic Definition and Useful Facts

A vertex weighted graph is given by a triple $G(E, V, d)$, where V is the set of vertices, E is the set of edges and $d : V \to \mathbb{N}$ is a weight function assigning (non-negative) integer demands to vertices of G.

Following the notation from [6], the vertices of the triangular lattice T can be described as follows: the position of each vertex is an integer linear combination $x\boldsymbol{p}+y\boldsymbol{q}$ of two vectors $\boldsymbol{p} = (1, 0)$ and $\boldsymbol{q} = (\frac{1}{2}, \frac{\sqrt{3}}{2})$. Thus vertices of the triangular lattice may be identified with pairs (x, y) of integers. Given the vertex v we will refer to its coordinates as $x(v)$ and $y(v)$. Two vertices are adjacent when the Euclidean distance between them is one. Therefore each vertex (x, y) has six neighbors: $(x-1, y), (x-1, y+1), (x, y+1), (x+1, y), (x+1, y-1), (x, y-1)$. For simplicity we refer to the neighbors as: *left*, *up-left*, *up-right*, *right*, *down-right* and *down-left*.

Assume that we are given a weight function $d : V \to \{0, 1, 2, \ldots\}$ on vertices of triangular lattice. We define a *weighted hexagonal graph* $G = (V, E, d)$ as an induced subgraph by vertices of positive demand on the triangular lattice, (see Figure 1). Sometimes we want to consider (unweighted) hexagonal graphs $G = (V, E)$ that can be defined as induced by subsets of vertices of the triangular lattice. In both cases we can assume that every vertex of hexagonal graph G knows its coordinates (x, y) in the triangular lattice.

Let us recall that a *proper multicoloring* of $G = (V, E, d)$ is a mapping f from $V(G)$ to subsets of integers such that $|f(v)| = d(v)$ for any vertex $v \in V(G)$ and $f(v) \cap f(u) = \emptyset$ for any pair of adjacent vertices u and v in the weighted graph G. The minimum number of colors needed for a proper multicoloring of

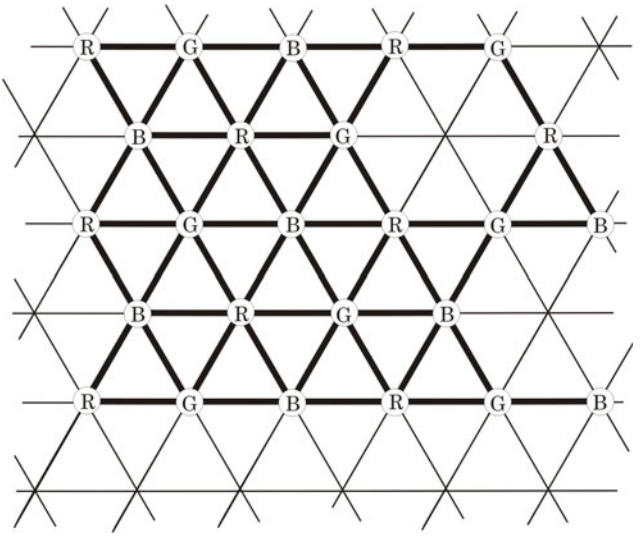


Fig. 1. An example of a hexagonal graph (with base coloring)

G, $\chi_m(G)$, is called the *multichromatic number*. The *(weighted) clique number*, $\omega(G)$, is the maximal clique weight on G, where the weight of a clique is the sum of demands on its vertices. As cliques in hexagonal graphs have at most three vertices, the weighted clique number is the maximum weight over weights of all triangles, edges and weights of isolated vertices.

There exists an obvious 3-coloring of the infinite triangular lattice which gives partition of the vertex set of any hexagonal graph into three independent sets. Let us denote a color of any vertex v in this 3-coloring by $bc(v)$ and call it a *base color* (for simplicity we will use *red*, *green* and *blue* as the base colors and their arrangement is given in Figure 1), i.e. $bc(v) \in \{R, G, B\}$.

An induced subgraph of the triangular lattice without 3-clique is called a *triangle-free hexagonal graph*. A *corner* in a triangle-free hexagonal graph is a vertex which has at least two neighbors and none of which are at angle π. A vertex is a *right corner* if it has an up-right or a down-right neighbor, otherwise it is a *left corner* (see Figure 2). A vertex which is not a corner is called a *non-corner*.

Lemma 1. *[11] Consider a 3-coloring (R,G,B) of the triangular lattice. Every odd cycle of the triangle-free hexagonal graph G contains at least one non-corner vertex of every color.*

As the elegant proof of Sudeep and Vishwanathan [11] is very short, we recall it for completeness and for future reference.

Proof. Assume without loss of generality that there exists an odd cycle in the graph which does not have a non-corner vertex colored red. Notice that in the

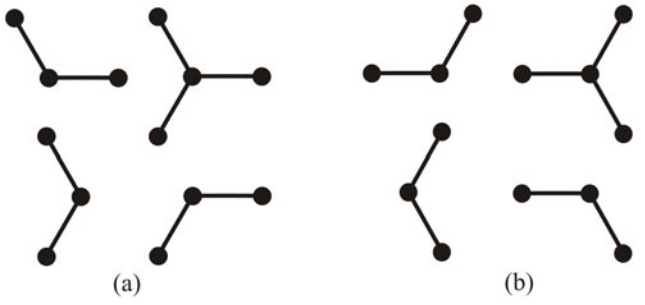


Fig. 2. All possibilities for: (a) - left corners, (b) - right corners

3-coloring of the triangular lattice, a corner has all its neighbors colored by the same color (they are at the angle $2\pi/3$). Hence, if all neighbors of a red colored corner are blue, we can recolor this corner by green color and vice-versa. That gives a valid 2-coloring of an odd cycle, a contradiction. □

Notice that a graph G after removing non-corners of one color is bipartite. For any weighted bipartite graph H, $\chi_m(H) = \omega(H)$ (see [8]), and it can be optimally multicolored by the following 1-local procedure.

Procedure 1. *Let $H = (V, E, d)$ be a weighted bipartite graph and let a bipartition $V = V' \cup V''$ be given. We get an optimal multicoloring of H if we assign to each vertex $v \in V'$ a set of colors $\{1, 2, \ldots, d(v)\}$, and with each vertex $v \in V''$ we associate a set of colors $\{m(v)+1, m(v)+2, \ldots, m(v)+d(v)\}$, where $m(v) = \max\{d(u) : \{u, v\} \in E\}$.*

Proof. The procedure is 1-local, because each vertex v uses only its weight function $d(v)$ or calculates value $m(v)$ which is taken from its neighbors. From definition of $m(v)$ we can clearly see that no conflict occurs in this multicoloring. Since in a bipartite graph the only cliques are edges and isolated vertices, the largest number of color used is

$$\max\{\max\{d(v) : v \in V'\}, \max\{d(v) + m(v) : v \in V''\}\} =$$
$$= \max\{\max\{d(v) : v \in V'\}, \max\{d(v) + \max\{d(u) : \{u, v\} \in E\} : v \in V''\}\} =$$
$$= \max\{\max\{d(v) : v \in V'\}, \max\{d(v) + d(u) : \{u, v\} \in E\}\} = \omega(G).$$

□

Notice that in any weighted hexagonal graph G, a subgraph of the triangular lattice T induced by vertices with positive demands $d(v)$, the only cliques are triangles, edges and isolated vertices. Recall that by definition all vertices of T which are not in G must have demand $d(v) = 0$. Therefore, the weighted clique number of G can be computed as follows:

$$\omega(G) = \max\{d(u) + d(v) + d(t) : \{u, v, t\} \in \tau(T)\},$$

where $\tau(T)$ is the set of all triangles of T.

For each vertex $v \in G$, define *base function* κ as

$$\kappa(v) = \max\{a(v,u,t) : \{v,u,t\} \in \tau(T)\},$$

where

$$a(u,v,t) = \left\lceil \frac{d(u)+d(v)+d(t)}{3} \right\rceil,$$

is an average weight of the triangle $\{u,v,t\} \in \tau(T)$. Clearly, the following fact holds.

Fact 1. *For each* $v \in G$,

$$\kappa(v) \leq \left\lceil \frac{\omega(G)}{3} \right\rceil$$

We call vertex v *heavy* if $d(v) > \kappa(v)$, otherwise we call it *light*. If $d(v) > 2\kappa(v)$ we say that the vertex is v *very heavy*.

To color vertices of G we use colors from an appropriate *pallet*. For a given color c, its palette is defined as a set of pairs $\{(c,i)\}_{i\in\mathbb{N}}$. A palette is called a *base color palette* if $c \in \{R,G,B\}$ is one of the base colors, and it is called *additional color palette* if $c \notin \{R,G,B\}$. In algorithm we will use 5 additional color pallets of different size (without explicitly naming the colors).

Very important invariant in the algorithm will be the parity of the coordinates. Let $p(x) = x \bmod 2$ be the parity function, which we will use for coordinates.

In our 1-local model of computation we assume that each vertex knows its coordinates as well as its own demand (weight) and demands of all its neighbors. In the next section, we will demonstrate how each vertex can color itself properly in constant time, using only this information.

3 Algorithm

Our algorithm consists of three main phases. In the first phase (Step 1 and 2 below) vertices take $\kappa(v)$ colors from its base color palette, so use no more than $\omega(G)$ colors. After this phase, all light vertices in G are fully colored, i.e. every light vertex $v \in V(G)$ already received all needed $d(v)$ colors. The vertices that are heavy but not very heavy induce a triangle-free hexagonal graph with weighted clique number not exceeding $\lceil \omega(G)/3 \rceil$. Very heavy vertices in G are isolated in the remaining graph and therefore are easily and fully colored. However, they have to be treated separately (Step 2).

In the second phase we construct two types of bipartite graphs. First (Step 3 below) we construct three bipartite graphs by removing each color types of noncorners and use Procedure 1 for optimal satisfying 1/8 of demands in the remaining graph. Next (Step 4 below) we divide the vertices in the triangular lattice into two sets of well separated parallel lines. Finally (Step 5 and 6 below), by using this partition we construct two bipartite graphs and use Procedure 1 for optimal satisfying 3/8 of demands of all vertices except some corners which have to be satisfied in a separate way – by using free colors from base color palettes.

More precisely, our algorithm consists of the following steps:

Algorithm

Input: Weighted hexagonal graph $G = (V, E, d)$, where all vertices know its position in the graph.

Output: A proper multicoloring of G, using at most $33/24 \cdot \omega(G) + O(1)$ colors.

Step 0 For each vertex $v \in V$ compute its base color $bc(v)$

$$bc(v) = \begin{cases} R & \text{if} \quad (x(v) + 2y(v)) \bmod 3 = 0 \\ G & \text{if} \quad (x(v) + 2y(v)) \bmod 3 = 1 \\ B & \text{if} \quad (x(v) + 2y(v)) \bmod 3 = 2 \end{cases},$$

and its base function value

$$\kappa(v) = \max\left\{ \left\lceil \frac{d(u) + d(v) + d(t)}{3} \right\rceil : \{v, u, t\} \in \tau(T) \right\}.$$

Step 1. For each vertex $v \in V$ assign to v $\min\{\kappa(v), d(v)\}$ colors from its base color palette. Construct a new weighted triangle-free hexagonal graph $G_1 = (V_1, E_1, d_1)$ where $d_1(v) = \max\{d(v) - \kappa(v), 0\}$, $V_1 \subseteq V$ is the set of vertices with $d_1(v) > 0$ (heavy vertices in G) and $E_1 \subseteq E$ is the set of all edges in G with both endpoints from V_1 (G_1 is induced by V_1).

Step 2. For each vertex $v \in V_1$ with $d_1(v) > \kappa(v)$ (very heavy vertices in G) assign the first unused $\kappa(v)$ colors of the base color palettes of its neighbors in T. Construct a new graph $G_2 = (V_2, E_2, d_2)$ where $d_2(v)$ is the difference between $d_1(v)$ and the number of colors assigned in this step, $V_2 \subseteq V_1$ is the set of vertices with $d_2(v) > 0$ and $E_2 \subseteq E_1$ is the set of all edges in G_1 with both endpoints from V_2 (G_2 is induced by V_2).

Step 3. For each class of noncorners (red, green, blue) do as follows:

- construct bipartite graph $G_{nR} = (V_{nR}, E_{nR}, d_{nR})$ where $d_{nR}(v) = d_2(v)/8$, $V_{nR} \subseteq V_2$ is the set of vertices from G_2 without red noncorners, and $E_{nR} \subseteq E_2$ is the set of all edges in G_{nR} with both endpoints in V_{nR} (E_{nR} is induced by V_{nR})
- construct bipartition of graph G_{nR}:
  - V'_{nR} is the set of blue vertices and red corners with all neighbors green,
  - V''_{nR} is the set of green vertices and red corners with all neighbors blue.
- Apply Procedure 1 for weighted graph G_{nR} by using colors from new additional color palette.

Do analogous for graph G_{nB} (G_2 without blue noncorners) and G_{nG} (G_2 without green noncorners).

Step 4. Divide vertices of infinite triangular lattice into two sets (two sets of horizontal lines):

$$V_e = \{v \in \tau(T) : p(x(v)) = p(y(v))\}$$
$$V_o = \{v \in \tau(T) : p(x(v)) \neq p(y(v))\}$$

Step 5. Using sets V_e and V_o do as follows:

5a.
- Construct a bipartite graph $\tilde{G} = (\tilde{V}, \tilde{E}, \tilde{d})$ where $\tilde{d}(v) = \frac{3d_2(v)}{8}$, $\tilde{V} \subseteq V_2$ consist of vertices from V_e and noncorners from V_o, and $\tilde{E} \subseteq E_2$ is the set of all edges in G_2 with both endpoints in $\tilde{V}$ ($\tilde{E}$ is induced by $\tilde{V}$).
- Construct bipartition of graph $\tilde{G}$:
  - $\tilde{V}'$ is the set of vertices with both coordinates even ($\{v \in \tilde{V} : p(x(v)) = p(y(v)) = 0\}$), noncorners with neighbors with both coordinates odd, and noncorners u with both neighbors in V_o and $p(x(u)) = 0$.
  - $\tilde{V}''$ is the set of vertices with both coordinates odd ($\{v \in \tilde{V} : p(x(v)) = p(y(v)) = 1\}$), noncorners with neighbors with both coordinates even, and noncorners u with both neighbors in V_o and $p(x(u)) = 1$.
- Apply Procedure 1 for weighted graph $\tilde{G}$ by using colors from new additional color palette.

5b. For each corner $v \in V_o$ assign $\frac{3d_2(v)}{8}$ colors from the free base color palettes. Left corners take unused odd colors and right corners take unused even colors.

Step 6. Switch the sets V_e and V_o and do the same as in Step 5.

4 Correctness Proof

Due to space limitations we omit a detailed proof and give in part only informal sketch of it.

At the very beginning of the algorithm there is a 1-local communication when each vertex finds out about the demands of all its neighbors. >From now on, no further communication will be needed. Recall that each vertex knows its position (x, y) on the triangular lattice T.

In Step 0 there is nothing to prove.

In Step 1 each heavy vertex v in G is assigned $\kappa(v)$ colors from its base color palette, while each light vertex u is assigned $d(u)$ colors from its base color palette. Hence the remaining weight of each vertex $v \in G_1$ is

$$d_1(v) = d(v) - \kappa(v).$$

Note that G_1 consists only of heavy vertices in G, therefore G_1 is a triangle-free hexagonal graph, and the proof of that can be found in [1].

In Step 2 only vertices with $d_1(v) > \kappa(v)$ (very heavy vertices in G) are colored. Each very heavy vertex in G has enough unused colors in its neighborhood to be finally multicolored. If vertex v is very heavy in G then it is isolated in G_1 (all its neighbors are light in G) (see [13]). Without loss of generality we may assume that $bc(v) = R$. Denote by

$$D_G(v) = \min\{\kappa(v) - d(u) : \{u, v\} \in T, bc(u) = G\},$$

$$D_B(v) = \min\{\kappa(v) - d(u) : \{u, v\} \in T, bc(u) = B\}.$$

Obviously, $D_G(v), D_B(v) > 0$ for very heavy vertices v in G. Since in Step 1 each light vertex t uses exactly $d(t)$ colors from its base color palette, we have at least $D_G(v)$ free colors from the green base color palette and at least $D_B(v)$ free colors from the blue base color palette. If vertex v could assign those colors to itself, we would have G_2 with $\omega(G_2) \leq \lceil \omega(G)/3 \rceil$. Formally it can be proved that in G_1 for every edge $\{v, u\} \in E_1$ holds:

$$d_1(v) + d_1(u) \leq \kappa(v), \quad d_1(u) + d_1(v) \leq \kappa(u),$$

and the proof can be found in [12].

From these facts we can observe that

$$\omega(G_2) \leq \left\lceil \frac{\omega(G)}{3} \right\rceil.$$

In Step 3 each vertex v has to decide whether it is a corner in G_2 or not. Only heavy neighbors of v can still exist in G_2. Unfortunately, in 1-local model v does not know which of his neighbors are heavy (and still exist in G_2) and which were light in G. Vertex v knows only where its neighbors u with $d(u) \leq a(v, u, t_1)$ and $d(u) \leq a(v, u, t_2)$, are located. We call these vertices *slight neighbors* of v. They must be light in G and as such they are fully colored in Step 1. Therefore, v knows where it cannot have neighbors in G_2 and presumes that all its neighbors which are not slight, still exist in G_2. Based on this knowledge, it can decide whether it is a corner or not. In each triangle in $\tau(T)$ containing v at least one neighbor of v is slight, so v has at least three such neighbors. If vertex v has more than four slight neighbors, then it is a non-corner. If vertex v has four slight neighbors, then the remaining two are not slight. In this case if an angle between those two are π, then v is a non-corner, otherwise it is a corner – a right corner if its down-left, up-left and right neighbors are slight, and a left corner if its down-right, up-right and left neighbors are slight. If vertex v has three slight neighbors, then it is a corner and the distinction between left and right is determined in the same way as above.

According to proof of Lemma 1 it is obvious that V'_{nR} and V''_{nR} are the bipartition of G_{nR}.

The problem is that, under 1-locality assumption, vertices cannot calculate value of d' of their neighbors, which is needed in Procedure 1 to calculate value $m(v) = \max\{\lceil d'(u)/8 \rceil : \{u, v\} \in E'\}$. However, this procedure can work fine in this model of calculation, by using $d'_v(u)$ instead of $d'(u)$, which is the number of expected demands on vertex u in vertex v after Step 2. The proof of this fact can be found e.g. in [13].

In Step 4 there is nothing to prove.

In Step 5a it is easy to check that $\tilde{V}'$ and $\tilde{V}''$ gives the bipartition of $\tilde{G}$. The problem with 1-locality assumption in Procedure 1 we can solve as in Step 3.

In Step 5b we take the corners and use again the base color palettes. If vertex $v \in G_2$ is a corner, it means that it has three slight neighbors with the same base color. Without loss of generality, assume that $bc(v) = R$ and its slight neighbors' base color is blue. Recall function D_B from Step 2 – we have $D_B(v)$ free colors from blue base color palette. We claim that if v is a corner in G_2 with three slight neighbors colored blue, then $d_2(v) \leq D_B(v)$ (see proof in [13]). Therefore, vertex v has as much as $d_2(v)$ free colors from the blue base color palette at his disposal. Since left corners take unused odd colors and right corners take unused even colors, no conflict occurs, because no two left corners can be connected.

During Step 3 each noncorner participate in exactly two from three rounds and receives $\lceil 2d_2(v)/8 \rceil$ colors while each corner participates in all three rounds and receives $\lceil 3d_2(v)/8 \rceil$ colors. During Steps 5 and 6 each vertex receives $\lceil 3d_2(v)/8 \rceil$ colors twice, so in both Steps it receives $\lceil 6d_2(v)/8 \rceil$ colors. Combining all Steps each noncorner receives $\lceil 8d_2(v)/8 \rceil$, so as many as it needs, and each corner receives $\lceil 9d_2(v)/8 \rceil$, so even more than it needs. Therefore, at the end, all of the demands are satisfied.

Ratio

We claim that during the first phase (Steps 1 and 2) our algorithm uses at most $\omega(G) + 2$ colors. To see this, notice that in Step 1 each vertex v uses at most $\kappa(v)$ colors from its base color palette and, by Fact 1 and the fact that there are three base colors, we know that no more than $3\lceil \omega(G)/3 \rceil \leq \omega(G) + 2$ colors are used. Note also that in Step 5b we use only those colors from base color palettes which were not used in Step 1, so altogether no more than $\omega(G) + 2$ colors from base color palettes are used in total in the first and second phase.
To count the number of colors used in the second phase (Steps 3-6) notice that we can divide the demands of each vertex in G_2 into eight equal parts. In Step 3 we introduce three new palettes that contain $\lceil \omega(G_2)/8 \rceil$ colors each. In Step 5 we introduce next palette that contain $\lceil 3\omega(G_2)/8 \rceil$ colors, and repeat this operation in Step 6.

Let $A(G)$ denote the number of colors used by our algorithm for the graph G. Thus, since $\omega(G_2) \leq \lceil \omega(G)/3 \rceil \leq \omega(G)/3 + 1$, the total number of colors used by our algorithm is at most

$$A(G) \leq \omega(G) + 2 + 3\left(\frac{\omega(G_2)}{8} + 1\right) + \left(\frac{3\omega(G_2)}{8} + 1\right) + \left(\frac{3\omega(G_2)}{8} + 1\right) =$$

$$= \omega(G) + 2 + \frac{9\omega(G_2)}{8} + 3 < \omega(G) + \frac{9\omega(G)}{24} + 2 + 5 = \frac{33}{24}\omega(G) + 7.$$

The performance ratio for our strategy is 33/24, hence we arrived at the statement of Theorem 1.

5 Conclusion

We have given a 1-local 33/24-approximation algorithm for multicoloring hexagonal graphs. This implies a 33/24-competitive solution for the online frequency allocation problem, which involves servicing calls in each cell in a cellular network. The distributed algorithm is practical in the sense that the frequency allocation for each base station is done locally, based on the information about itself and its neighbors only, and the time complexity is constant.

According to goal ratio 4/3, the improvement of the result is the biggest since paper [5] in 2000. Present authors strongly believe that with this kind of method the competitive ratio achieved here cannot be improved, since McDiarmid and Reed conjecture implies that 9/8 is the best possible competitive ratio on triangular-free hexagonal graphs. There has to be found a significant new idea to solve this problem with better ratio.

References

1. Chin, F.Y.L., Zhang, Y., Zhu, H.: A 1-local 13/9-competitive Algorithm for Multicoloring Hexagonal Graphs. Algorithmica 54, 557–567 (2009)
2. Hale, W.K.: Frequency assignment: theory and applications. Proceedings of the IEEE 68(12), 1497–1514 (1980)
3. Havet, F.: Channel assignment and multicoloring of the induced subgraphs of the triangular lattice. Discrete Mathematics 233, 219–231 (2001)
4. Havet, F., Žerovnik, J.: Finding a five bicolouring of a triangle-free subgraph of the triangular lattice. Discrete Mathematics 244, 103–108 (2002)
5. Janssen, D., Narayanan, L., Shende, S.: Distributed Online Frequency Assignment in Cellular Network. Journal of Algorithms 36(2), 119–151 (2000)
6. McDiarmid, C., Reed, B.: Channel assignment and weighted coloring. Networks 36(2), 114–117 (2000)
7. Narayanan, L.: Channel assignment and graph multicoloring. In: Handbook of Wireless Networks and Mobile Computing, pp. 71–94. Wiley, New York (2002)
8. Narayanan, L., Shende, S.M.: Static frequency assignment in cellular networks. Algorithmica 29(3), 396–409 (2001)
9. Šparl, P., Žerovnik, J.: 2-local 4/3-competitive Algorithm for Multicoloring Hexagonal Graphs. Journal of Algorithms 55(1), 29–41 (2005)
10. Šparl, P., Žerovnik, J.: 2-local 5/4-competitive algorithm for multicoloring triangle-free hexagonal graphs. Information Processing Letters 90(5), 239–246 (2004)
11. Sudeep, K.S., Vishwanathan, S.: A technique for multicoloring triangle-free hexagonal graphs. Discrete Mathematics 300, 256–259 (2005)
12. Witkowski, R.: A 1-local 17/12-competitive algorithm for multicoloring hexagonal graphs. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 346–356. Springer, Heidelberg (2009)
13. Witkowski, R., Žerovnik, J.: 1-local 7/5-competitive Algorithm for Multicoloring Hexagonal Graphs. Electronic Notes in Discrete Mathematics, vol. 36, pp. 375–382 (2010)
14. Žerovnik, J.: A distributed 6/5-competitive algorithm for multicoloring triangle-free hexagonal graphs. International Journal of Pure and Applied Mathematics 23(2), 141–156 (2005)

Modeling Social Networks through User Background and Behavior

Ilias Foudalis[1], Kamal Jain[2], Christos Papadimitriou[3], and Martha Sideri[4]

[1] Georgia Institute of Technology
fountalis@gatech.edu
[2] Microsoft Research, Redmond
kamalj@microsoft.com
[3] EECS Department, UC Berkeley
christos@cs.berkeley.edu
[4] Department of Informatics, Athens University of Economics and Business
sideri@aueb.gr

Abstract. We propose a generative model for social networks, both undirected and directed, that takes into account two fundamental characteristics of the user: background (specifically, the real world groups to which the user belongs); and behavior (namely, the ways in which the user engages in surfing activity and occasionally adds links to other users encountered this way). Our experiments show that networks generated by our model compare very well with data from a host of actual social networks with respect to a battery of standard metrics such as degree distribution and assortativity, and verify well known predictions about social networks such as densification and shrinking diameter. We also propose a new metric for social networks intended to gauge the level of surfing activity, namely the correlation between degree and Page rank.

1 Introduction

Digital social networks are important for several reasons: They have become crucial human resources; they are growing in ubiquity and economic importance; they constitute a fascinating and productive research subject; and they can teach us much about human behavior and relationships. Regarding this last point, it would be desirable to go the other way around, namely to use rudimentary models of human behavior in order to understand social networks. In this paper we develop a generative model for social networks based on a minimalistic set of assumptions about the nature of human relations and behavior.

Our premise is that social networks reflect two basic aspects of human life: *identity* — or "background" — and *behavior*. There are of course many well known and useful models of social networks (see the subsection on related work below); however, we are not aware of any one that explicitly evokes categories related to identity and behavior. We capture identity by the association of humans with groups: cities, neighborhoods, professions, sports clubs, religious groups, fan clubs, bars frequented, etc. We are not thinking of these as digital groups, but as a description of an individual and his/her relationship with the real world. Note that certain groups constitute a partition of the

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 85–102, 2011.

universe of potential users (e.g., countries) while others (e.g. fan clubs) may overlap and fail to cover everybody. That is, we think of the underlying universe as the union of cliques, and we call the resulting undirected graph *the world*. We emphasize again that this is *not* the social network we are interested in, but the ground truth underlying it. User behavior in the formation of a social network entails the process of acquiring new relationships, that is, adding links.

The social network is created in discrete steps. At each step one new user/node arrives (this is the clock whereby time is measured in the network creation process). Upon arrival, the node is connected to a subset of the nodes already in the network, determined by a random process as follows: first, a node is connected only to people with whom it is connected in the world; second, the smaller the groups two nodes share, the higher the connection probability — for example, two fans of an esoteric band, or two Alaskans, have a higher probability of being connected than two scientists or two Indians. During each such step, another important activity takes place: Every node, with some small probability, takes a random walk by "surfing" the existing network, and adds a link (a directed or undirected edge, depending on the network being modeled) to one of the other nodes encountered; the probability of taking a random walk at a step decreases with the node's age. To model social networks, such as Facebook, in which remote surfing is disabled or problematic, the length of the random walk can be restricted to two. Each node has two characteristics, *extroversion*, which determines the intensity of adding links to others, and *quality*, which affects the probability that a visiting node will add a link to it; we assume that these are uniformly distributed (perhaps the distribution least likely to seed the statistical behavior of our model).

We validate our model by comparing the networks it generates to a diverse suite of social networks. We use a wide range of metrics, most of them proposed in the literature and some new. We find very satisfactory agreement, and make a few observations about the structure of social networks that have not been mentioned before. In particular, the generated networks are assortative and have high clustering coefficient, in agreement with data and expectation. With respect to clustering, we observe (both in data and our networks) that clustering diminishes in high-degree nodes. Furthermore, we notice that our model verifies certain well known predictions about the evolution of social networks, such as decreasing effective diameter and densification [17]. We also find agreement in other metrics, such as degree distribution and betweenness centrality. Finally, we propose a new metric for directed networks whose intention is to capture the degree to which links are the result of surfing activity by the nodes: the correlation between in-degree and Page rank of the nodes, with high correlation indicating high probability that links are the result of surfing activity. We believe that this metric is of some interest and use in the study of social networks — for example, it sharply differentiates between Twitter and Flickr. We also compare our model and its agreement with data to that of other models, such as the *forest fire model* of [17]. Finally, we also derive some analytical results that help

Related Work. Non-digital social networks had been studied for a long time in the social sciences, see the recent textbook [9], Milgram's "small world" experiment [20] and more recent work on small worlds [11]. With the advent of digital social networks numerous researchers have studied and measured social networks [21],[14],[6] and have

identified several statistical characteristics, such as degree of clustering [10] and the presence of assortativity [25]. Also, researchers have analyzed the temporal evolution of social networks and have made several interesting observations, such as the shrinking of the diameter and the densification of the network [17]. With respect to models of social networks, Watts and Strogatz [29] presented a simple "rewiring" model that exhibits small world characteristics (with small diameter and big clustering coefficients). Other models choose graphs that conform to a given degree distribution, as in the configuration model described in [23], or by preferential attachment [13],[1]. Leskovec et al [17] proposed a forest-fire model to explain the decreasing diameter phenomenon that they observed at citation graphs. Leskovec et al [19] proposed a mathematically tractable model based on Kronecker multiplication and presenting many of the observed properties of social networks. Kumar et al [14] analyzed the evolution of social networks and categorized users according to connectivity properties. A different approach was proposed in [27] where the authors presented a game-theoretic approach on modeling social networks. Another interesting work is presented in [15] where a mathematically tractable model based on ideas that are rooted in sociology is proposed, albeit without a comparison with real data. In [10] there was presented a dynamic model of network formation where each node would connect uniformly at random to another pre-existing node and then it would connect to some of the neighbors of the pre-existing node. Also in a very recent study [3] postulates that nodes have "zones of influence" and are connected if these overlap, in a manner quite reminiscent of the cliques of the World in this paper. For mathematical analyses of some of the above models see to [2],[12] and [22],[24].

2 The Model

We start by describing informally, and motivating, our model. The nature of relationships in the real world depends on people's *identity/background*, as well as their *behavior*. For us, identity is captured by the groups to which a person belongs — groups of geography, professional affiliation, interests, hobbies, etc. Certain of these groups form a partition of the population (e.g., neighborhoods), while others overlap and do not exhaust the population (e.g., bars frequented). Note that there are digital systems of this sort, usually called *affiliation networks* [15],[30]; here, we do not think of these groups as digital networks, but instead as the underlying truth about the world, the attribute combinations of the people who will join our network. Once a new person joins a social network, it is natural that s/he will be connected with people s/he knows from the various groups to which s/he belongs. It is natural to assume that the smaller the group, the higher the probability of knowing that person well enough to link to them (and this is captured in Step 5 of our algorithm, see Algorithm 1 below).

But after somebody has joined a social network, they are likely to explore it and, depending on how extroverted they are, and on how interesting or high-quality the other nodes they visit appear to them, they may initiate links to these other nodes. But it is natural to assume that the interest in creating new links wanes as time goes by (see, for example, the study of the Cyworld social network [7]). These two aspects are both captured in the first and second probabilities used in Step 6 of our algorithm below. And

these are the two sources of links in our network: Initial links coming from connections in the World, and links acquired while the user is surfing the network.

In our algorithm (see Algorithm 1 below), there is a first phase, during which we construct the World, a set of groups (cliques) over the users. We start by generating random groups which partition the users (say, into cities, then into professions, etc.); the number of partitions is a parameter (Step 1). We also generate a few more random groups/cliques, which however do not constitute partitions of the universe (Step 2). These groups will determine, once the network generation process has started, how newly arriving nodes get initially connected to others that have previously arrived. The sizes of these cliques are power-law distributed, with power-law exponent γ (taken in our experiments to be between 2 and 3).

Upon arrival (Step 4), a node is assigned two values, both sampled uniformly from $[0, 1]$: *extroversion* and *quality*; these will influence how often links will be created from and to (respectively) the node. Then the node (Step 5) is linked to some of the previously arrived nodes with whom s/he is connected in The World via a common group, with the probability of connection decreasing with the size of the common group.

Finally, new links are created at each step by each node taking random walks in the network, and linking to the nodes encountered with probability influenced by the visited node's quality and the visitor's own extroversion. This is a crucial part of our model, for many reasons. It is here that user behavior influences the network. Since the intensity of the process is inversely proportional to the node's age, it not hard to see that about $N \log N$ edges are added this way. Finally, in directed networks, this process has the effect of bringing the in-degree of a node in line with its Page rank [5]: recall that the Page rank is precisely the frequency with which a node is visited in a particular random walk, and this is both verified and captured by a metric that we propose.

Our algorithm is based on the following parameters (shown in bold):

- **N** is the number of nodes.
- **F** is the number of partitions generated in Step 1 below.
- γ is the power-law exponent of the distribution of cliques in The World.
- **r** is a parameter regulating the expected number of random walks that a node will perform at a given time.
- **d** is the expected depth of each random walk.

Algortithm 1

- *First Phase: Generation of the World*
  1. *Repeat* **F** *times: generate a random partition of the universe* $\mathbf{N} = \{0, 1, \ldots, \mathbf{N}-1\}$ *into sets whose sizes are power-law distributed with exponent* γ.
  2. *Suppose the previous step generated* s *sets; generate* s *random subsets of the universe with power-law distributed sizes.*
- *Second Phase: Generation of the Network*
  3. *Repeat for* $t = 0, 1, \ldots, \mathbf{N} - 1$*:*
  4. *A node* t *joins the network, with extroversion* x_t *and quality* q_t*, both generated uniformly at random from* $[0, 1]$.
  5. *For* $i = 0, \ldots, t-1$ *add a link (directed or undirected, depending on the nature of the network desired) from* t *to* i *with probability* $1 - \frac{\min}{\max}$*, where* $\min$ *is the*

size of the smallest clique the two nodes have in common, and max *is the size of the maximum clique in The World. If there is no common clique, the probability is* 0.

6 *For each node* $i < t$ *repeat* $\mathbf{r}$ *times: start a random walk with probability* $\frac{1}{(t-i+1)}$. *If during the random walk node* j *is visited, a link from* i *to* j *is added with probability* $x_i * q_j$, *and the random walk halts with probability* $(1 - \frac{1}{\mathbf{d}})$ *(Note: In the* depth-two version *motivated by networks such as Facebook, the random walk is made to stop deterministically at depth two.)*

In our experiments we choose parameters to best fit the comparison data. The typical parameter values where, $F = 1, \gamma = -2.35, r = 10$ and $d = 10$. It is interesting that, while the power law exponent γ does not affect much the network structure, varying F results in huge differences.

3 Metrics

The following are metrics used widely in the literature for measuring and comparing networks:

Degree distribution: The distribution of the network's degrees (in the case of directed networks, of both in- and out-degrees) is an important characteristic of a network. As most such distributions in practice are power-law distributed, this distribution is typically summarized by the effective exponent of the power-law γ, found by regression.

Diameter and average path length: The diameter of a network is the largest distance between any two connected nodes. In Internet-like networks it grows slowly, while for social networks it has been found to *shrink* very slowly [17]. We also measure, in addition to the diameter, the average shortest path length, and the effective diameter as defined in [17].

Clustering: Clustering measures how likely it is, given that ij and jk are links, that ik is also a link (the network could be either undirected or directed). Social networks tend to be highly clustered, more so than Internet-like graphs (since, arguably, clustering is a social network's *raison d' être*).

Betweenness Centrality: *Betweenness centrality* captures a node's importance in terms of connecting other nodes; it was first proposed by Freeman [8]:

$$Ce_i^B(g) = \sum_{k \neq j : i \neq (j,k)} \frac{P_i(kj)/P(kj)}{(N-1)(N-2)/2} \quad . \tag{1}$$

where $P_i(kj)$ is the number of shortest paths between k and j that go through i. In order to make calculations on large graphs we have implemented the betweenness centrality algorithm proposed in [4].

Assortativity coefficient and neighbor degrees: The *assortativity coefficient* [25] captures correlations between the degrees of neighboring nodes:

$$\frac{\sum_{\text{link}(i,j)}(d_i - \bar{d})(d_j - \bar{d})}{\sum_{i \epsilon N}(d_i - \bar{d})^2} \quad . \tag{2}$$

where d_i is the degree of node i and $\bar{d}$ is the average degree. A related measure is the average degree of the neighbors of a vertex of degree d, $K_{\text{nn}}(d)$ (also termed as average neighbour connectivity):

$$K_{\text{nn}}(d) = \sum_{i}^{maxdegree} P(i|d) \quad . \tag{3}$$

where $P(i|d)$ is the conditional probability that given a node with degree d will be connected with a node of degree i. Average neighbor degree of the whole graph is the average of (3) for all the degrees. Positive assortativity implies an increasing $K_{\text{nn}}(d)$ function, that is, high-degree nodes tend to connect to other high-degree nodes, and similarly for low-degree nodes.

Explicit Page rank: We introduce a new metric for directed networks, *Page rank explicitness*. Page rank [5] is a well known and useful measure of the "importance" of a document/node in a web-like directed graph, roughly the steady-state distribution of a restarting random walk. In social networks one would expect that, since user behavior is random walk-like and link addition is easier, a node's in-degree (d_{in}) would be highly correlated than usual with Page rank (r). In our model, Step 6 clearly has the effect of bringing closer the in-degree and the Page rank. We call this effect *explicit Page rank*, and we measure it in terms of the Pearson correlation coefficient of the in-degrees and Page ranks of the nodes; in Page rank calculations we use a damping factor $d = 0.85$. Thus $ExplicitPagerank = \frac{\sum_{i=1}^{N}(d_{in}(i)-\bar{d_{in}})(r(i)-\bar{r})}{\sqrt{\Sigma_{i=1}^{N}(d_{in}(i)-\bar{d_{in}})^2 \Sigma_{i=1}^{N}(r(i)-\bar{r})^2}}$. This measure is between -1 and 1, with 1 standing for perfect correlation (positive multiple). In most directed networks this correlation is expected to be high (it is theoretically predicted to be very high in expectation for random graphs, for example), but differences of, say, between .7 and .95, can be revealing. We propose this measure as an indication of the degree to which surfing behavior affects the structure of the network (but one should keep in mind that it is also very high in random graphs).

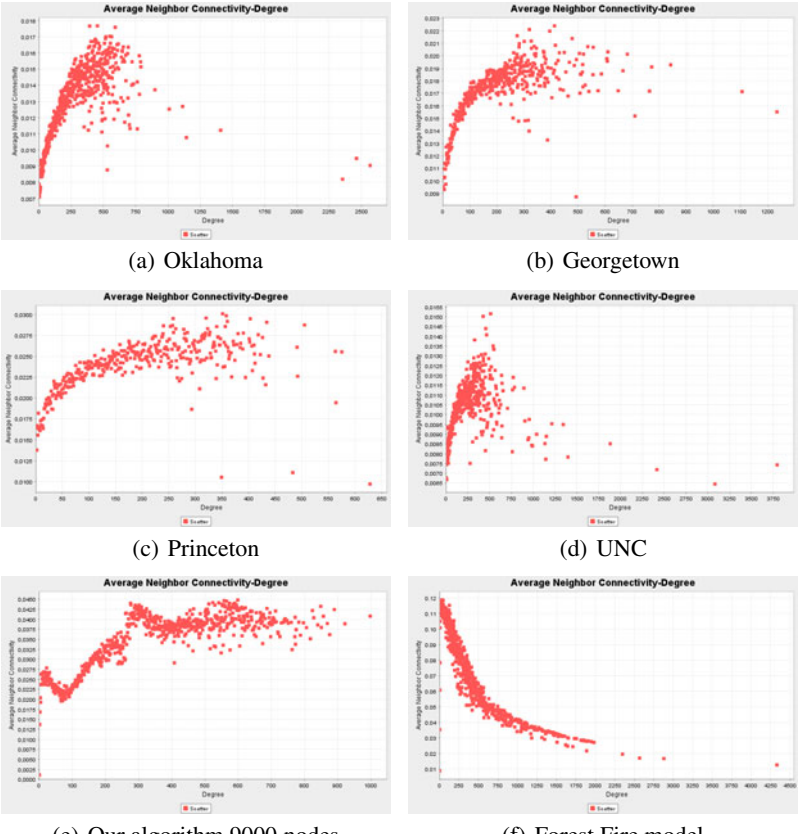
4 Results: Undirected Networks

Table 1 below summarizes the results of experiments with our algorithm, and comparisons with data from Facebook — in particular, networks capturing the relationships between community members at four large U.S. universities (UNC, Princeton, Oklahoma and Georgetown), obtained from [28].

Overall, we see remarkable consistency in the data, and good agreement with the results of our algorithm. Regarding assortativity, It has been observed that social networks have positive assortativity (in contrast to scientific networks which have negative

Table 1. Comparisons of Facebook graphs and our algorithm

	Princeton	Georgetown	Oklahoma	UNC	our model (1)	our model (2)	Forest Fire
Number of nodes	6596	9414	17425	18163	9000	18000	9000
Number of edges	293320	425638	892528	766800	394512	917512	300130
Average degree	88.93	90.42	102.442	84.43	87.66	100.275	66.69
Max Degree	628	1235	2568	3795	847	1313	4330
Degree exponent (γ)	-1.13	-1.26	-1.40	-1.46	-1.19	-1.34	-0.99
Average neighbour degree exponent	2.62	2.47	2.22	2.77	1.97	2.17	1.265
Assortativity coefficient	0.091	0.075	0.073	0.0007	0.066	0.085	-0.34
Avg Node Betweenness	2.525E-4	1.856E-4	1.01E-4	9.913E-5	1.031E-4	6.002E-5	1.856E-4
Betweenness Slope	4.652E-6	4.49E-6	2.77E-6	6.051E-6	2.751E-6	1.306E-6	4.49E-6
Clustering coefficient	0.244	0.231	0.235	0.206	0.21	0.256	0.561
Diameter	9	11	9	7	7	7	10
Average path length	2.67	2.75	2.767	2.801	2.27	2.42	2.67



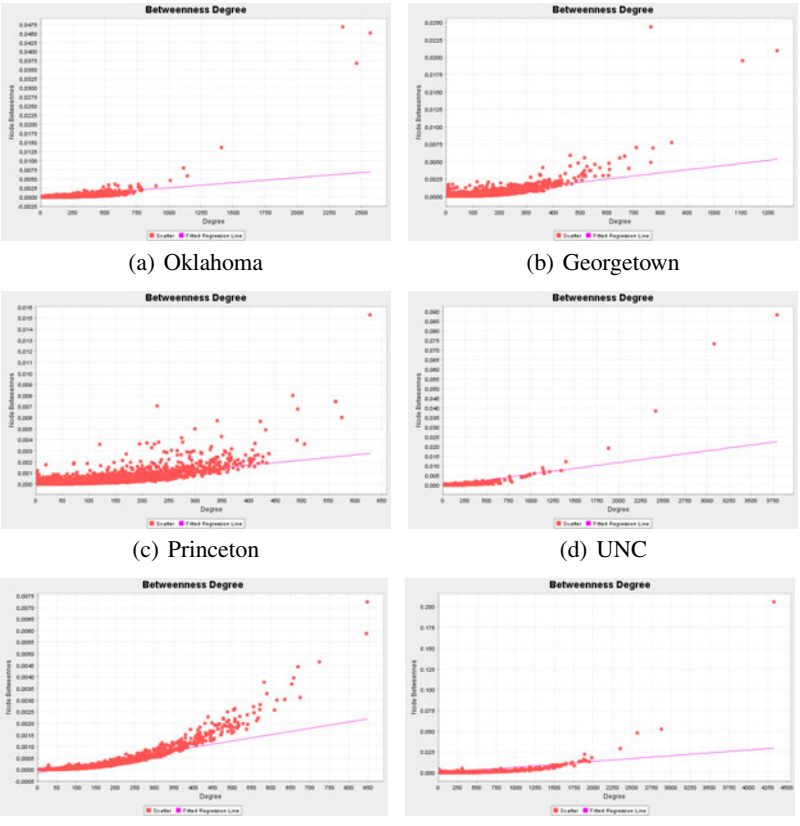
(a) Oklahoma (b) Georgetown

(c) Princeton (d) UNC

(e) Our algorithm 9000 nodes (f) Forest Fire model

Fig. 1. Average degree of the nearest neighbours as a function of the degree d ($K_{\mathrm{nn}}(d)$)

assortativity) [25], and we can see that this is true for the Facebook data and also for our model. **Although assortativity values are close to** 0 **we can verify the assortative nature of the networks by observing (Fig. 1) the increasing** $K_{\mathrm{nn}}(d)$ **function.**

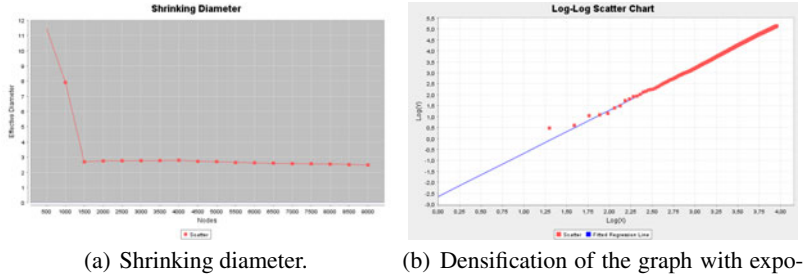


(a) Oklahoma (b) Georgetown

(c) Princeton (d) UNC

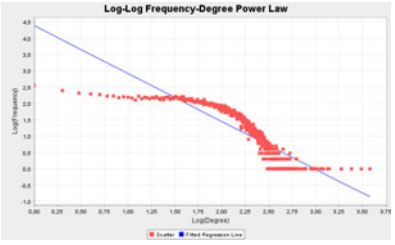
(e) Our algorithm 9000 nodes (f) Forest Fire model

Fig. 2. Betweenness vs. Degree

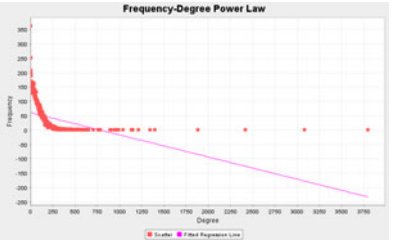


(a) Shrinking diameter. (b) Densification of the graph with exponent $\gamma = 2$

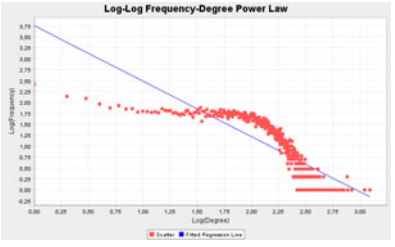
Fig. 3. Temporal graph evolution plots. Shrinking diameter and densification laws.



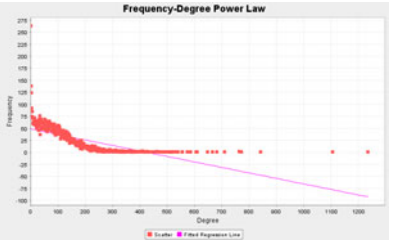
(a) UNC log-log



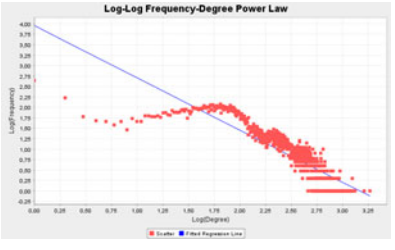
(b) UNC normal



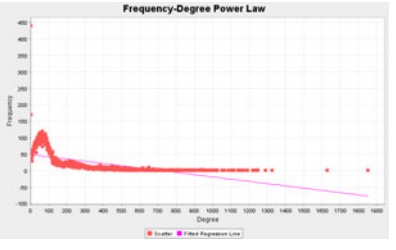
(c) Georgetown log-log



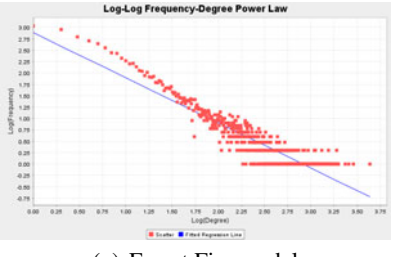
(d) Georgetown normal



(e) Our algorithm 18000 log-log



(f) Our algorithm 18000 normal



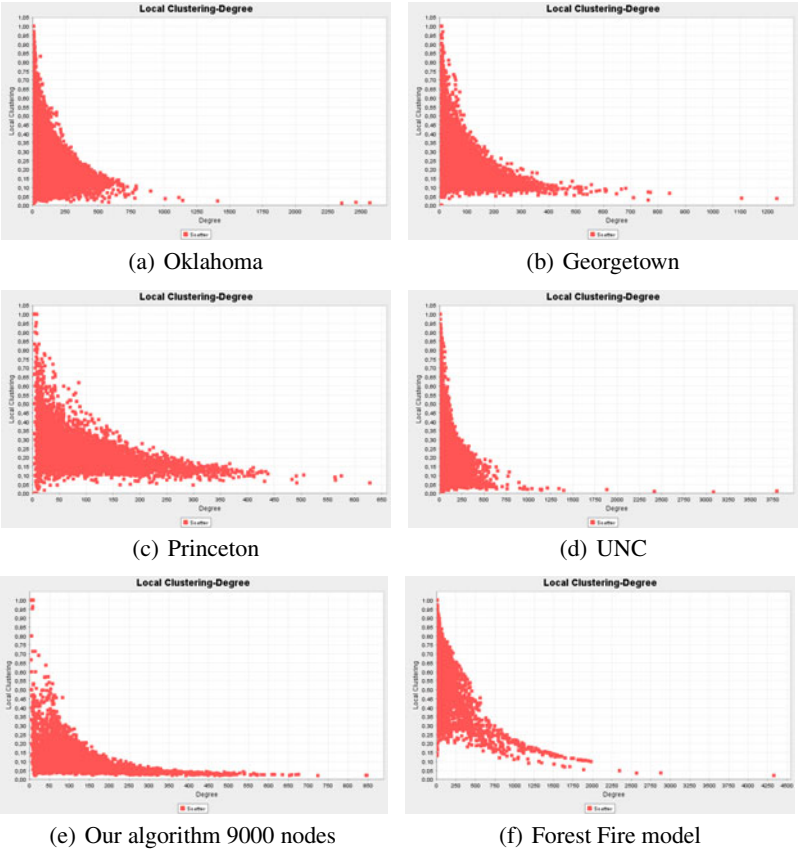
(g) Forest Fire model

Fig. 4. Frequency vs Degree

An interesting observation that can be seen from Fig. 2 is that both in Facebook graphs and the graphs produced by our model a small part of the higher degree nodes can be considered as central, and there exists no clear increasing trend between a nodes degree and a nodes betweenness centrality. A node with a high betweenness centality value is expected to be the mediator of the information transfer in a network. We observe that very few nodes take that role.

It has been also observed that social networks present high clustering coefficients (compared to clustering coefficients exhibited by a pure random network). Our data and the generated graphs do exhibit such behavior. For example, the clustering coefficient of our Georgetown data set is many times higher than the value $90.42/425638 = 0.00002$, the clustering coefficient of a random graph with the same number of nodes. The same is true for all other data sets. Another important observation we make in both the graphs generated by our model and the Facebook data sets is that clustering correlates negatively with degree; that is, two neighbors of high degree nodes are less likely to be connected to each other than two neighbors of a low degree node, and this again makes sense for social networks.

Social networks usually have a very small diameter, and in fact it has been also observed that a social network's diameter shrinks as the network grows, up to a point, and from that point on it is stable. As we can see from Fig. 3(a), the same phenomenon is present in the graphs generated by our model. We can see from our datasets that all Facebook networks have a small diameter. Also we have observed that the networks

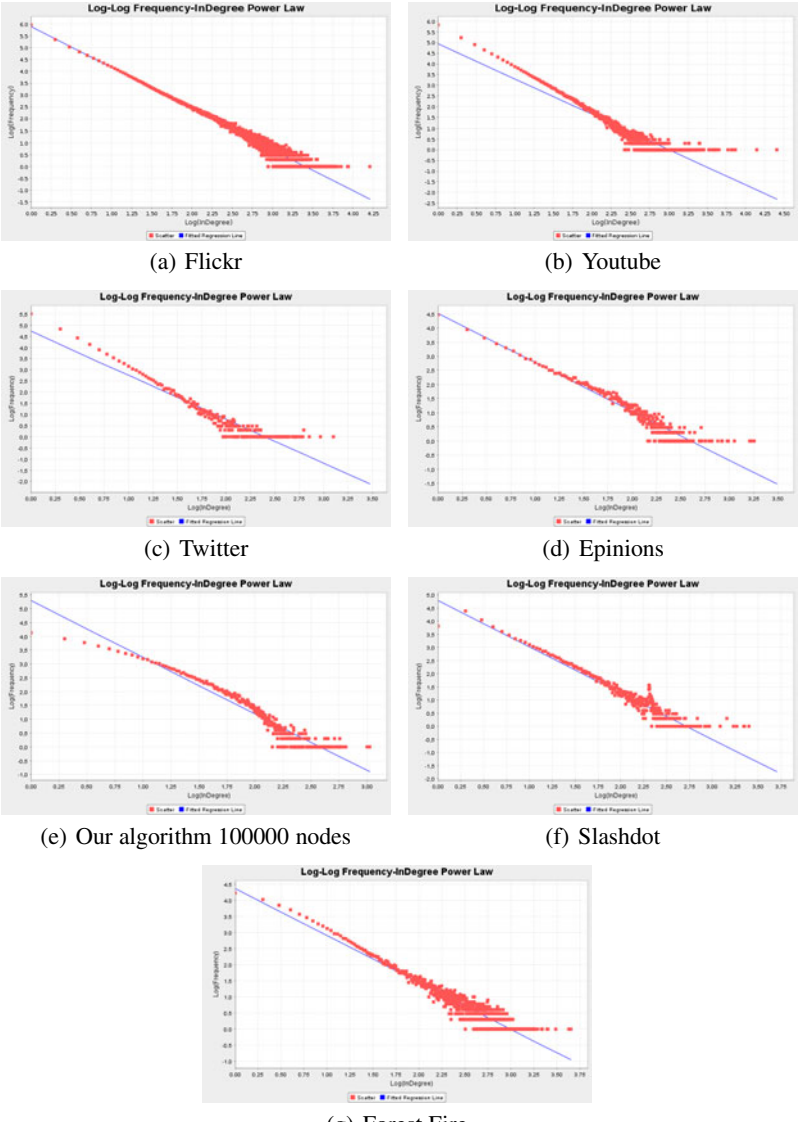


(a) Oklahoma (b) Georgetown

(c) Princeton (d) UNC

(e) Our algorithm 9000 nodes (f) Forest Fire model

Fig. 5. Clustering vs. Degree distributions



(a) Flickr (b) Youtube

(c) Twitter (d) Epinions

(e) Our algorithm 100000 nodes (f) Slashdot

(g) Forest Fire

Fig. 6. Log-log frequency vs. in-degree plots for our directed graphs

produced by our algorithm have a stable and small diameter. Also, both the networks produced by our algorithm and the Facebook data sets have a rather small average path length of about 2.7.

The frequency vs. degree distributions shown on Fig. 4 for the Facebook graphs are very similar to the ones produced from our algorithm. Even though there is no clear

power law distribution, there are fat tails in a log-log scale diagram, that is, the majority of the nodes have small degrees.

It has also been observed [17] that as social networks grow, they become more and more dense. As we can see from Fig. 3(b), the same effect is apparent in our model, and the graph follows a relation $e(t) \propto n(t)^{\alpha}$ where $e(t)$ and $n(t)$ are the number of edges and nodes at time t and α is an exponent which in our case is close to 2.

5 Results: Directed Networks

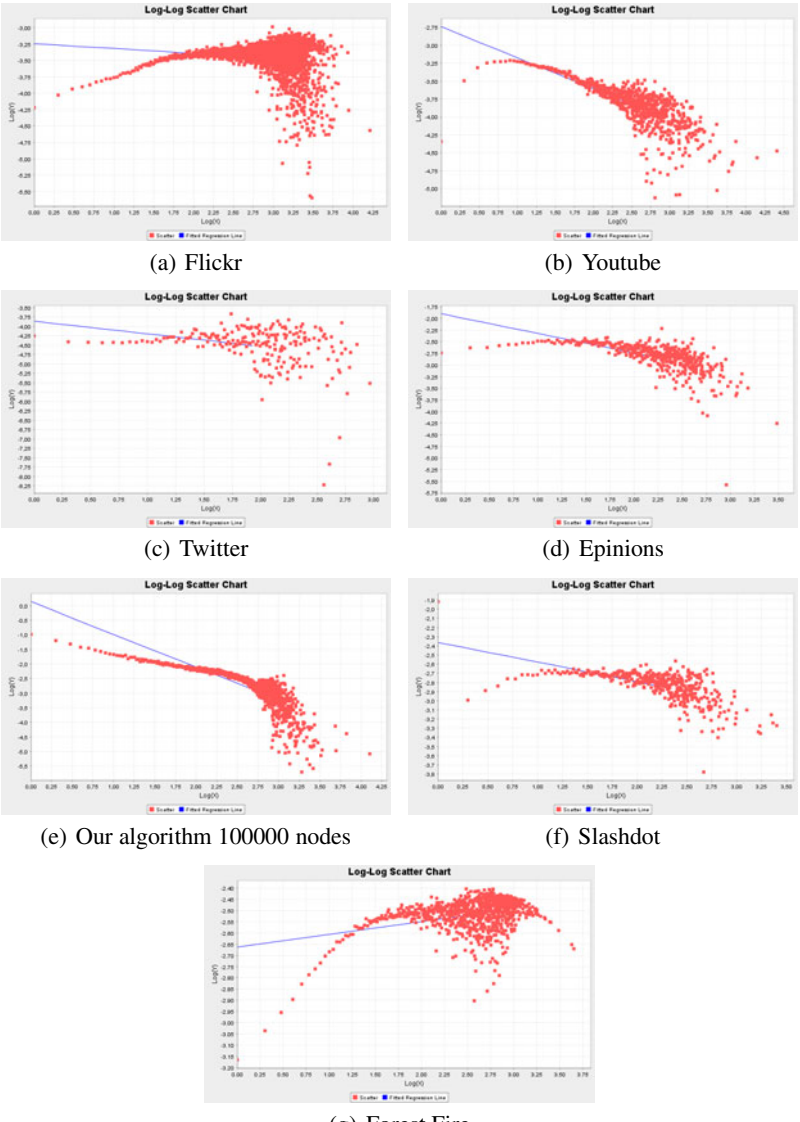
Our algorithm seems to generate equally well undirected and directed graphs. To validate our algorithm on directed social networks ("information networks") we obtained data from Flickr [21], Youtube [21], Epinions [18], Slashdot [26] and Twitter (the latter obtained from public accessible user profiles). In order to compute the efecctive diameter [17] we used the SNAP library [16]. Table 2 summarizes the experiments.

Table 2. Comparisons of Directed graphs and our algorithm

	Flickr	Youtube	Twitter	Epinions	Slashdot	our model	Forest Fire
Number of nodes	1846198	1157827	460622	75879	82170	100000	100000
Number of edges	22613981	4945382	950046	508837	948465	1734653	1881680
Average degree	12.24	4.29	2.06	6.7	11.54	17.34	18.81
In-Degree exponent (γ)	-1.56	-1.52	-1.97	-1.72	-1.75	-1.89	-1.453
Assortativity coefficient	0.004	-0.03	-0.03	-0.01	-0.04	-0.06	0.033
Ratio of nodes in the largest scc	0.69	0.44	0.62	0.42	0.86	0.89	1.0
Diameter	12	12	14	10	9	12	13
Effective Diameter	7.53	7.57	7.91	5.88	5.671	6.98	7.25
Page rank correlation	0.7	0.97	0.92	0.95	0.97	0.92	0.623

In all directed networks examined, in-degrees seem to be distributed by some power law (see the figures). It has been noted in [10] that information networks are expected to have much smaller average degree than undirected social networks, because in an information network (that is, networks where the edges represent information flow and not social relations) very few nodes have high degrees. This fact can be verified from Table 2. Also as expected all the networks present small diameter and effective diameter. On all networks, (both data and generated) a very large fraction of the nodes belongs to the "giant" strongly connected component.

In all the directed social networks except Flickr, we observe a high correlation between Page rank and in-degree, and this agrees with the networks generated by our algorithm (this could be expected, since in our algorithm many of the edges are shortcuts of random walks). What is the difference between Flickr and the other networks considered here, which could explain this discrepancy? In networks such as Youtube, Slashdot, Twitter, and Epinions we believe that the users add new edges as they surf the network. For example whilst surfing, a Youtube user may observe an uploader with many videos of his taste and consequently decide to follow him. On the other hand in



(a) Flickr (b) Youtube

(c) Twitter (d) Epinions

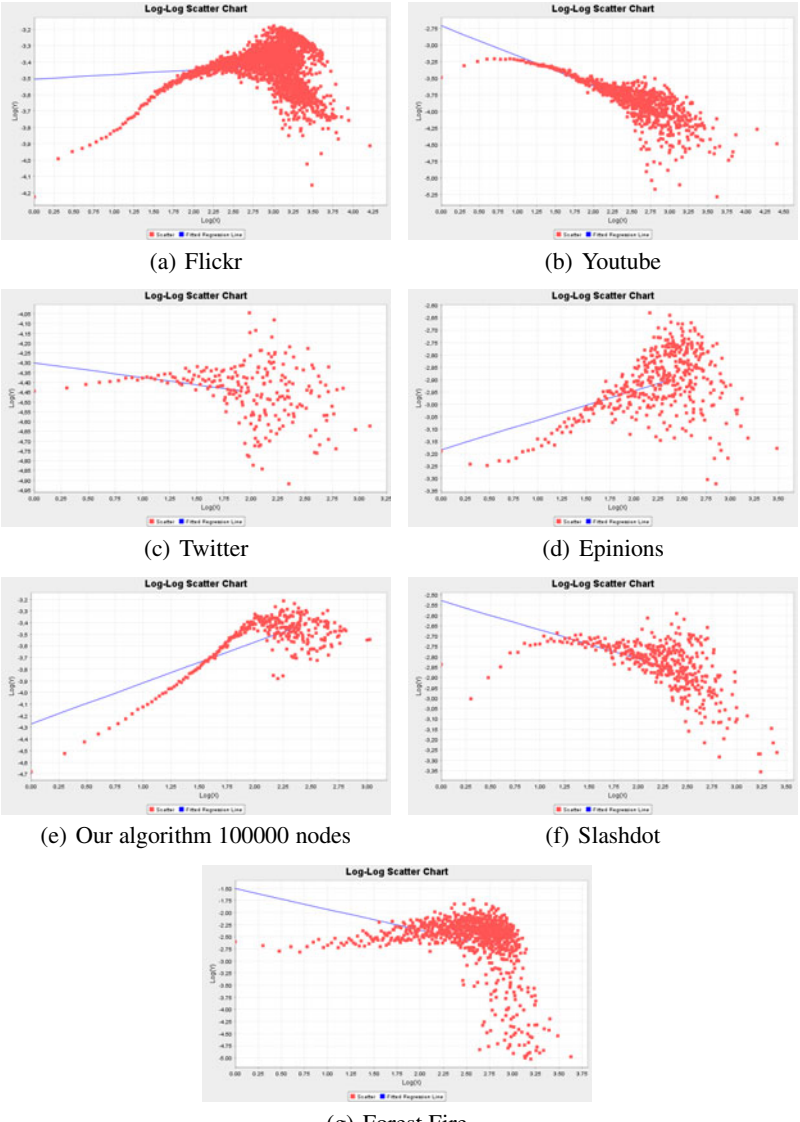
(e) Our algorithm 100000 nodes (f) Slashdot

(g) Forest Fire

Fig. 7. Log-log correlation between in-degree of a node and in-degree of the nodes that the node is pointed by

Flickr users share photos with friends, and thus we could have expected that the random walk behavior is a little less prevalent.

Because of the directed nature of networks we had to expand the definition of the $K_{\mathrm{nn}}(d)$ distribution. Thus we have considered two different measures: The correlation of a node's in-degree and the in-degree of the nodes that point to it $K_{\mathrm{nn\,in}}(d)$; and the



(a) Flickr (b) Youtube

(c) Twitter (d) Epinions

(e) Our algorithm 100000 nodes (f) Slashdot

(g) Forest Fire

Fig. 8. Log-log correlation between in-degree of a node and in-degree of nodes that the node points

correlation of a node's in-degree and the in-degree of the nodes that the node points to, $K_{\text{nn out}}(d)$. These two measures are shown in Fig. 7 and Fig. 8. As we can see from Fig. 7 in all networks there is a tendency for low in-degree nodes to be pointed by low in-degree ones (decreasing $K_{\text{nn in}}(d)$). In Fig. 8, three kinds of networks (Flickr,

Epinions and the networks generated by our model) have an increasing $K_{\text{nn out}}(d)$ function contrary to the rest of the networks.Regarding assortativity we found that, surprisingly, all of the networks except for Flickr tend to be disassortative. This is contrary to the widely held belief that all social networks present positive assortativity.

6 Comparison with the Forest Fire Model

We have seen that our model largely succeeds in capturing many of the salient characteristics of the data sets. We further test it by comparing its agreement with data to that of the Forest Fire model of [17], one of the best known models of social networks. In the Forest Fire model once a node t is added to the network it is connected to an "ambassador" node w chosen uniformly at random. Then x edges out of w and y edges into w are chosen at random, where x and y are drawn from given geometric distributions (whose parameters are the main inputs to the generation algorithm), and the other endpoints of those edges are joined with t in the same way. Then the process is repeated recursively with those x new nodes, skipping any node we have seen before so no node is ever visited twice, until the recursion dies out for lack of fresh nodes.

We implemented the Forest Fire model and, through extensive experimentation, chose parameters that maximize the model's fit with our Facebook data sets. Since these data sets are undirected, we treat the Forest Fire model's edges as undirected.

In Table 1 we summarize the results of this comparison. We notice that, like our algorithm, the Forest Fire model also produces a network with high clustering coefficients, and also captures the decreasing trend of the clustering coefficient with the degree (Fig. 5(f)). Also similarly to our model, networks generated by Forest Fire have short average path lengths and effective diameter. The authors of the Forest Fire model also note that it exhibits densification and shrinking diameter over time. We can also observe from Fig. 2(f) that the betweeness centrality values in the Forest Fire graphs are small and only a small part of the higher degree nodes have large values, in accordance both with our model and the Facebook datasets.

So far the Forest Fire model is in good agreement both with our model and the Facebook networks. In other aspects, Forest Fire fares distinctly worse than our model. The graphs it produces have a decreasing $K_{nn}(d)$ function (Fig.1(f)) and a very negative assortativity coefficient, whereas social networks are by nature assortative (where high-degree nodes tend to connect with other high-degree nodes), and our model captures this aspect. Also, the degree distribution in the Forest Fire model (Fig. 4(g)) clearly follows a power law, a contrast with the Facebook datasets (and our model). .

The results of the comparison between the directed versions of our model and the Forest Fire model are presented in Table 2. On the negative side, the nodes of the Forest Fire network form a giant strongly connected component (in contrast to both our model and our data), and the correlation between page rank and in-degree is lower than the one that in our model and most data sets. However, the Forest Fire model comes especially close to the Flickr network both in terms of the k_{nn} distributions and positive assortativity, and also in terms of low page rank correlations.

7 Some Analytical Observations

Our model is conceptually simple and one can therefore predict analytically some of its behavior.

Degree distribution. Consider the i-th node in the directed version of our model. Its in-degree and out-degree, call them a_i and b_i, are the sum of two parts: $a_i = a'_i + a''_i$, and similarly for $b_i = b'_i + b''_i$, where by a''_i we denote the part of the in-degree due to edges of the random walk. It is easy to see that a'_i (b'_i) is the sum of $n - i$ (respectively, $i - 1$) Bernoulli variables that are not independent, but have dependence of the variety (selection without replacement) that can be analyzed by Hoeffding's inequality. Therefore, a'_i is exponentially concentrated with mean $c(n-i)$, and similarly for b'_i with mean $c(i - 1)$, where c is a random variable depending only on the World (ignoring, for simplicity, the partition aspect of the sets comprising the World). Therefore, the a'_i's (seen now as an ensemble) are power-law distributed, and so are the $b - i$'s; however, for the undirected version the part of the degrees that is not due to the random walk, $a'_i + b'_i$, is exponentially concentrated with the same mean $c(n - 1)$.

Now for the random walk edges making up a''_i and b''_i: Out of i we start a total of $r(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i+1}) \approx r \log n$ random walks in expectation, and therefore draw $x_i \cdots \log(n - i)/2d$ edges out of i, again in expectation. A very similar analysis holds for the a''_i; however, the quantity π_i enters the formula, the stationary distribution of the graph at various times t, seen as a Markov chain. This is the reason why one expects the in-degrees to be aligned with Page rank in the directed case.

Densification and shrinking diameter. It is easy to see that the expected number of edges added at step i is a superlinear function of i, and hence densification is expected. As for shrinking diameter, let us consider the undirected case. At step t, we have a graph whose degrees are highly concentrated around the expectation ct for some $c < 1$. Treating this as a regular graph of degree ct, we can postulate that its diameter grows as $d(t) = \frac{\log t}{\log ct}$. For $c < 1$, this is a slowly decreasing function.

8 Conclusions and Open Problems

We have presented a new generative model for social networks that attempts seriously to take into account characteristics of user behavior and identity. The networks generated by our model agree reasonably well with actual social and information networks, both directed and undirected, with respect to the measures studied in the literature, and confirm empirical predictions about the growth of certain key parameters (diameter and density). We also introduced a new measure, correlation between degree and Page rank, which we believe is a useful way of gauging the extent to which the structure of the network is the result of surfing behaviour.

One obvious open problem is to formalize more the distinction between "social networks" and "information networks", with an eye of providing analytical evidence of their different natures.

References

1. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks, vol. 286, pp. 509–512 (October 1999)
2. Bollobas, B.: Mathematical results on scale-free random graphs. In: Handbook of Graphs and Networks, pp. 1–34. Wiley, Chichester (2003)
3. Bonato, A., Janssen, J., Pralat, P.: A Geometric Model for On-line Social Networks. In: Proceedings of the 3rd Workshop on Online Social Networks (WOSN 2010), Boston, MA, USA (June 2010), http://www.usenix.org/events/wosn10/tech/full_papers/Bonato.pdf
4. Brandes, U.: A faster algorithm for betweenness centrality (2001), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.2024
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Computer Networks and ISDN Systems, pp. 107–117 (1998)
6. Cha, M., Mislove, A., Gummadi, K.P.: A measurement-driven analysis of information propagation in the flickr social network. In: WWW 2009: Proceedings of the 18th International Conference on World Wide Web, pp. 721–730. ACM Press, New York (2009), http://dx.doi.org/10.1145/1526709.1526806, doi:10.1145/1526709.1526806
7. Chun, H., Kwak, H., Eom, Y.-H., Ahn, Y.-Y., Moon, S., Jeong, H.: Comparison of online social relations in volume vs interaction: a case study of cyworld. In: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, IMC 2008, Vouliagmeni, Greece, pp. 57–70. ACM, New York (2008), http://doi.acm.org/10.1145/1452520.1452528, doi:10.1145/1452520.1452528
8. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry 40(1), 35–41 (1977), http://links.jstor.org/sici?sici=0038-0431%28197703%2940%3A1%3C35%3AASOMOC%3E2.0.CO%3B2-H
9. Jackson, M.O.: Social and Economic Networks. Princeton University Press, Princeton (2008), http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0691134405
10. Jackson, M.O., Rogers, B.W.: Meeting strangers and friends of friends: How random are social networks? American Economic Review 97(3), 890–915 (2007), http://ideas.repec.org/a/aea/aecrev/v97y2007i3p890-915.html
11. Kleinberg, J.: The small-world phenomenon: An algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing, pp. 163–170 (2000), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.4366
12. Kleinberg, J.: Complex networks and decentralized search algorithms. In: Proceedings of the International Congress of Mathematicians, ICM (2006)
13. Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., Upfal, E.: Stochastic models for the web graph. In: FOCS 2000: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, p. 57. IEEE Computer Society, Washington, DC, USA (2000)
14. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: KDD 2006: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 611–617. ACM, New York (2006), http://portal.acm.org/citation.cfm?id=1150402.1150476, doi:10.1145/1150402.1150476
15. Lattanzi, S., Sivakumar, D.: Affiliation networks. In: STOC 2009: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, pp. 427–434. ACM, New York (2009), doi:10.1145/1536414.1536474

16. Leskovec, J.: Snap graph library, http://snap.stanford.edu
17. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: KDD 2005: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, Illinois, USA, pp. 177–187. ACM, New York (2005), doi:10.1145/1081870.1081893
18. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. CoRR abs/0810.1355 (2008)
19. Leskovec, J., Chakrabarti, D., Kleinberg, J.M., Faloutsos, C.: Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 133–145. Springer, Heidelberg (2005), http://dx.doi.org/10.1007/11564126_17, doi:10.1007/11564126_17
20. Milgram, S.: The small world problem. Psychology Today 2, 60–67 (1967)
21. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: IMC 2007: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, California, USA, pp. 29–42. ACM, New York (2007), doi:10.1145/1298306.1298311
22. Mitzenmacher, M.: A brief history of generative models for power law and lognormal distributions. Internet Mathematics 1(2) (2003)
23. Molloy, M., Reed, B.: A critical point for random graphs with a given degree sequence. In: Random Graphs 93: Proceedings of the Sixth International Seminar on Random Graphs and Probabilistic Methods in Combinatorics and Computer Science, pp. 161–179. John Wiley & Sons, Inc., New York (1995)
24. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003), http://arxiv.org/abs/cond-mat/0303516
25. Newman, M.E.J., Park, J.: Why social networks are different from other types of networks. Physical Review E 68(3), 36122 (2003)
26. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 351–368. Springer, Heidelberg (2003), http://www.metapress.com/content/DM0N8R10RL4U430Q
27. Skyrms, B., Pemantle, R.: A dynamic model of social network formation. Proc. Natl. Acad. Sci. U. S. A. 97(16), 9340–9346 (2000)
28. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: Community structure in online collegiate social networks. Tech. Rep. arXiv:0809.0690 (September 2008), comments: 38 pages, 14 figure files, some of which have been compressed for this posting (higher-resolution pdf available from http://www.amath.unc.edu/Faculty/mucha/Reprints/facebook.pdf)
29. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393(6684), 440–442 (1998), http://dx.doi.org/10.1038/30918, doi:10.1038/30918
30. Zheleva, E., Sharara, H., Getoor, L.: Co-evolution of social and affiliation networks. In: KDD 2009: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1007–1016. ACM, New York (2009), http://dx.doi.org/10.1145/1557019.1557128, doi:10.1145/1557019.1557128

Dirichlet PageRank and Trust-Based Ranking Algorithms

Fan Chung*, Alexander Tsiatas, and Wensong Xu

Department of Computer Science and Engineering
University of California, San Diego
{fan,atsiatas,w4xu}@cs.ucsd.edu

Abstract. Motivated by numerous models of representing trust and distrust within a graph ranking system, we examine a quantitative vertex ranking with consideration of the influence of a subset of nodes. An efficient algorithm is given for computing *Dirichlet PageRank* vectors subject to Dirichlet boundary conditions on a subset of nodes. We then give several algorithms for various trust-based ranking problems using Dirichlet PageRank with boundary conditions, showing several applications of our algorithms.

1 Introduction

PageRank has proven to be a useful tool for ranking nodes in a graph in many contexts, but it is clear that many refinements can be made for specific purposes. For example, PageRank can be manipulated by link spammers to boost certain nodes' ranking, and it interprets all links between nodes as positive votes for importance even if some links are meant to show distrust.

As an illustrative example, consider the following problem. Suppose an agent v in a social network wants to compute a quantitative ranking among all nodes, but v has a subset of neighbors X who v trusts. v wants to compute a personal ranking of the nodes in the network, but v wants to make sure that there is no large disparity with its trusted neighbors.

Another trust that ranking systems do not take into account arises from a distinction between types of social networks. Some networks, such as Facebook, model closer relationships between people: a social friendship where edges form presumably only between people who know each other personally. This is in direct contrast with systems such as Twitter where the act of "following" someone indicates no such connection. The close-knit network indicates a higher degree of trust.

When agents participate in both networks, it is useful to be able to rank the nodes of the larger network based on the smaller. A node v can compute a ranking among its own personal trust network and then use this to calculate a ranking of nodes in the larger network, many of which do not appear in the

* Research supported in part by ONR MURI N000140810747, and AFOR's complex networks program.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 103–114, 2011.

more personal network. Current ranking mechanisms such as PageRank compute a global ranking and therefore are not suitable for this situation.

In this paper, we consider a tool that can model these and many other scenarios: Dirichlet PageRank vectors with boundary conditions. Given a graph G and a subset of nodes S, we first compute a PageRank vector pr subject to Dirichlet boundary conditions $\mathrm{pr}(v) = 0$ for vertices v on the boundary of S. For example, Dirichlet boundary conditions can be used to model and propagate the distrust of specified vertices in the graph. We then generalize Dirichlet PageRank to use arbitrary boundary conditions $\mathrm{pr}(v) = \sigma(v)$ for boundary nodes v, providing a general framework for a variety of ranking models. Our algorithms are efficient, giving approximate solutions.

1.1 Related Work

The idea of ranking nodes in a graph has a rich history starting from the introduction of PageRank by Brin and Page [5]. The original PageRank was meant for Web Search, but many researchers have developed more tailored ranking systems such as personalized PageRank [13,14], giving a ranking relative to some specified starting distribution $\boldsymbol{s}$.

One pitfall with PageRank as a ranking system is the fact that all edges contribute positively. In practice, an edge such as a link from one Web page to another can also represent a negative interaction or distrust between the nodes. Several related mathematical models of propagating trust and distrust in a network ranking system are given [11]. There are numerous empirical results. Another algorithm [12] relies on a small hand-picked set of trusted nodes, but one must be careful not to allow malicious nodes to be included.

There are many other algorithms derived from PageRank that use specific tweaks to model trust or distrust in ranking schemes. [1] considers axioms that a ranking system should satisfy and develops several ranking systems accordingly. [4] and [16] systematically model distrust by modifying the PageRank equations to consider negatively-weighted edges, and [15] gives an algorithm with a similar flavor using random walks. Many of these algorithms are closely related, but rigorous analysis is desired for capturing specific phenomena. We will show that these related models can be represented by Dirichlet PageRank with appropriate boundary conditions.

Another area of research concerns spam nodes when they are identified. It has been shown that if agents can collude [2] or easily create pseudonyms [6], they can artificially boost their ranking in PageRank and other ranking systems. There is some work done in how to penalize these vertices [3], and our Dirichlet PageRank can be efficiently used to achieve the same goal.

1.2 Results in This Paper

Motivated by the continual development of new PageRank-based algorithms and analysis of Dirichlet eigenvectors in [9], we develop and analyze Dirichlet PageRank algorithms. For a connected graph G, we examine a Dirichlet PageRank

equation and compute the unique solution with Dirichlet boundary conditions $\mathrm{pr}(v) = 0$ for vertices v on the boundary of a specified vertex subset S.

After giving the algorithm for computing Dirichlet PageRank vectors, we generalize the boundary conditions to arbitrary values $\mathrm{pr}(v) = \sigma(v)$ for boundary vertices v. We give an efficient algorithm ApproxDirichPR to compute approximate Dirichlet PageRank vectors with any boundary condition σ. We also give a full analysis leading to the following theorem, where $|\cdot|$ is the L_1-norm, and $\mathrm{vol}(S)$ is twice the number of edges in the specified vertex subset. (Detailed definitions are given in Section 2.)

Theorem 1. *For any $\epsilon \in (0,1)$ and any jumping constant $\alpha \in (0,1)$, the algorithm ApproxDirichPR outputs an ϵ-approximate Dirichlet PageRank vector $\widetilde{\mathrm{pr}}_S$ in time $O(\frac{\mathrm{vol}(S)\log\frac{1}{\epsilon}}{\alpha})$, which, compared to the exact Dirichlet PageRank pr_S, satisfies:*

$$\widetilde{\mathrm{pr}}_S(v) \leq \mathrm{pr}_S(v), \forall v \in S$$
$$|\mathrm{pr}_S - \widetilde{\mathrm{pr}}_S| < \frac{\epsilon \mathrm{vol}(S)}{\alpha}.$$

We illustrate several applications of Dirichlet PageRank with boundary conditions below. Many of the specific PageRank variations are covered by this general framework, and we will show how its use can allow the efficient consideration of several models in [1,3,4].

1.3 Several Applications of Dirichlet PageRank

• **Diminishing known spammers' influence.** A first application concerns the adjustment of PageRank in the presence of known spammers. For example, on the Web, many web pages can be identified as spammers based on content or user reports. We would like a network ranking scheme to take this into account, penalizing these nodes and others that heavily link to them. We will show that Dirichlet PageRank is useful for this problem.

• **Considering trusted friends' opinions.** While adjusting PageRank to take into account known spammers is relatively simplistic, there are plently of more subtle or complex problems that Dirichlet PageRank can handle well. For example, a single node in a graph may want to compute a personalized ranking of nodes, but it has a set of trusted friends or neighbors whose own rankings are to be valued accordingly. While previously-studied personalized PageRank vectors do not take this effect into account, we can use Dirichlet PageRank with boundary conditions to compute a more informed ranking. We will present an algorithm PRTrustedFriends for this problem.

• **Validating rank for newly-created nodes.** The previous application of adjusting PageRank to account for trusted friends can be logically extended to validating a new vertex's ranking within a larger network. Suppose that a new person enters a social network but is unsure about who to trust and distrust within the network. Personalized PageRank is a useful tool for quantitative information, but it raises the question of whether or not this ranking is susceptible

to unknown spammers. Without a specific set of trusted friends, it may seem hopeless for the newcomer, but the new vertex can use Dirichlet PageRank with boundary conditions to validate and adjust its own ranking with a randomly-selected pool of established nodes within the network. We will give the details in an algorithm PRValidation.

• **Reconciling rank in personal and global social networks.** Additional interesting questions arise when analyzing different types of social networks. While social networks are often treated on their own, there are many different types of networks with wide participation. Some, like Facebook, offer a more personal viewpoint on network structure, where edges are formed only by mutual consent between two people who usually know each other. This is in contrast with a network such as Twitter, where connections are much more impersonal. Here, people "follow" each other based not only on friendship, but also interest in subject matter, celebrity appeal, advertising, and many conceivable other reasons.

With the vast array of information available on a network such as Twitter, it is important for a user to know who is trustworthy or worth following. This is a difficult problem, but a user does have some information at hand; its own, more close-knit, smaller social network or even a trusted subgraph of the larger network. Using Dirichlet PageRank, a user can compute a ranking on the smaller network, and then use boundary conditions appropriately to infer a ranking on the remaining nodes in the larger network, taking its personal associations into account. We will develop an algorithm PRTrustNetwork for this problem.

Finally, we can use similar ideas to tackle the problem in reverse: suppose a global ranking of the nodes in a larger, loose social network such as Twitter is known, and a user wants to develop a personalized ranking for a small subgraph or its own trusted network taking the global ranking into account. We again can use Dirichlet PageRank with appropriate boundary conditions, as outlined in an algorithm PRInferRanking.

The rest of the paper proceeds as follows. In Section 2, we outline necessary background on PageRank and Dirichlet boundary conditions. Section 3 develops the theory of PageRank with Dirichlet boundary conditions, and Section 4 extends this theory for arbitrary boundary conditions σ. We develop and analyze the ApproxDirichPR algorithm in Section 5 and give algorithms for the previously-discussed applications in Section 6.

2 Preliminaries

For a connected undirected graph $G = (V, E)$ with n vertices and m edges, let A be the *adjacency matrix* and D be the *diagonal degree matrix* where D_{ii} is the degree of the i-th vertex. The typical random walk on G is defined by the *transition probability matrix* $D^{-1}A$.

The *normalized Laplacian* $\mathcal{L}$ is defined as:

$$\mathcal{L} = D^{-1/2}(D - A)D^{-1/2} = I - D^{-1/2}AD^{-1/2}.$$

Let S denote a subset of V consisting of vertices in G. The *vertex boundary* δS is defined as: $\delta S = \{v | v \notin S, (u,v) \in E, \text{where } u \in S\}$, and the *edge boundary* ∂S is defined as: $\partial S = \{(u,v) | u \in S, v \notin S\}$. The *volume* $\text{vol}(S)$ denotes the sum of the degrees of vertices in S.

The *restricted Laplacian* $\mathcal{L}_S$ is the submatrix of $\mathcal{L}$ restricted to $S \times S$. And the *restricted Green's function* $\mathcal{G}_{S,\beta}$ is defined as: $\mathcal{G}_{S,\beta}(\beta I_S + \mathcal{L}_S) = I_S$, where $\beta \geq 0$. Note that $\mathcal{L}_S$ is positive definite [9], so $\mathcal{G}_{S,\beta}$ is well-defined. The *PageRank vector* pr is defined as:

$$\text{pr} = \alpha s + (1-\alpha)\text{pr}W,$$

where α is called the *jumping constant*, s is the *seed vector*, and $W = \frac{1}{2}(I + D^{-1}A)$ is called the *lazy random walk transition matrix*. PageRank was first introduced in [5] to measure the importance of Web pages, and recently it has been applied to many fields, such as measuring trust in social networks [1].

3 PageRank with Dirichlet Boundary Conditions

Let S be a subset of G; for a function (or vector) $f : V \to \mathbb{R}$, we say f satisfies the Dirichlet boundary condition if $f(v) = 0$ for all $v \in \delta S$.

The PageRank vector satisfying the Dirichlet boundary conditions is defined by the equations:

$$\text{pr}(v) = \begin{cases} \alpha s(v) + (1-\alpha)\sum_{u \in V} \text{pr}(u) W_{uv} & \text{if } v \in S \\ 0 & \text{otherwise .} \end{cases} \tag{1}$$

Let pr_S, s_S denote the vectors pr and s restricted to S, and W_S, D_S, A_S denote the respective matrices restricted to $S \times S$. Dirichlet boundary conditions are also applied to PageRank vectors in [7].

Theorem 2. *For a connected graph, the above PageRank equation (1) has one and only one solution. With* $\beta = \frac{2\alpha}{1-\alpha}$*, it is given by*

$$\text{pr}_S = \beta s_S D_S^{-1/2} \mathcal{G}_{S,\beta} D_S^{1/2}.$$

Proof. Since $\text{pr}(v) = 0$ when $v \notin S$, the PageRank equation (1) is equivalent to

$$\text{pr}_S = \alpha s_S + (1-\alpha)\text{pr}_S W_S.$$

And since

$$W = \frac{1}{2}(I + D^{-1}A) = I - \frac{1}{2}(D^{-1/2}\mathcal{L}D^{1/2}),$$

and D is diagonal matrix, we have

$$\text{pr}_S = \alpha s_S + (1-\alpha)\text{pr}_S(I - \frac{1}{2}(D_S^{-1/2}\mathcal{L}_S D_S^{1/2})).$$

Solving for pr_S gives the theorem. □

Let pr' be the original PageRank vector with no boundary conditions. Suppose G_S is the subgraph of G consisting of vertices of S and only those edges between vertices in S. Let pr'' be the PageRank of G_S.

It is easy to see that for every $v \in S$, $\mathrm{pr}(v) \leq \mathrm{pr}'(v)$; however, since we only care about the relative ranking of the vertices and not the value itself, and the L_1-norm of these vectors is arbitrary (depending on the boundary condition), it is more interesting to compare the relative magnitudes of the PageRank values of different sets of nodes among these 3 definitions of PageRank on the subset S.

Define

$$S_o = \{v \in S | \exists u \notin S : (u, v) \in E\}, S_i = S \setminus S_o,$$

and assume that neither of these two sets are empty.

Let W_{ii} be W restricted to $S_i \times S_i$, W_{0i} be W restricted to $S_o \times S_i$, and

$$w_0 = (1-\alpha)\mathbf{1}W_{0i}^T, w_i = \mathbf{1} - (1-\alpha)\mathbf{1}W_{ii}^T.$$

Lemma 1

$$\frac{\mathrm{pr}'_{S_o} w_0^T}{\mathrm{pr}'_{S_i} w_i^T} \geq \frac{\mathrm{pr}_{S_o} w_0^T}{\mathrm{pr}_{S_i} w_i^T} \geq \frac{\mathrm{pr}''_{S_o} w_0^T}{\mathrm{pr}''_{S_i} w_i^T}$$

Proof. Let W'' be the lazy random walk transition probability matrix of G_S; then the following equations hold:

$$\mathrm{pr}'_{S_i} = \alpha s_{S_i} + (1-\alpha)\left(\mathrm{pr}'_{S_i} W_{ii} + \mathrm{pr}'_{S_o} W_{0i}\right), \tag{2}$$

$$\mathrm{pr}_{S_i} = \alpha s_{S_i} + (1-\alpha)\left(\mathrm{pr}_{S_i} W_{ii} + \mathrm{pr}_{S_o} W_{0i}\right), \tag{3}$$

$$\mathrm{pr}''_{S_i} = \alpha s_{S_i} + (1-\alpha)\left(\mathrm{pr}''_{S_i} W''_{ii} + \mathrm{pr}''_{S_o} W''_{0i}\right). \tag{4}$$

Let $c_1 = \alpha s_{S_i} \mathbf{1}^T$. Then we have

$$\frac{\mathrm{pr}_{S_o} w_0^T}{\mathrm{pr}_{S_i} w_i^T} = \frac{\mathrm{pr}_{S_i} w_i^T - c_1}{\mathrm{pr}_{S_i} w_i^T}.$$

(2) - (3) gives

$$\left(\mathrm{pr}'_{S_i} - \mathrm{pr}_{S_i}\right)\left(I - (1-\alpha)W_{ii}\right) = \left(\mathrm{pr}'_{S_o} - \mathrm{pr}_{S_o}\right)\left((1-\alpha)W_{0i}\right)$$

Let $c_2 = \left(\mathrm{pr}'_{S_i} - \mathrm{pr}_{S_i}\right)\left(I - (1-\alpha)W_{ii}\right)\mathbf{1}^T$. Since $\forall v \in S, \mathrm{pr}(v) \leq \mathrm{pr}'(v)$ implies that $\mathrm{pr}'_{S_i} - \mathrm{pr}_{S_i}$ is nonnegative, c_2 is also nonnegative. Then we have

$$\frac{\mathrm{pr}'_{S_o} w_0^T}{\mathrm{pr}'_{S_i} w_i^T} = \frac{\mathrm{pr}'_{S_i} w_i^T - c_1}{\mathrm{pr}'_{S_i} w_i^T} = \frac{\mathrm{pr}_{S_i} w_i^T + c_2 - c_1}{\mathrm{pr}_{S_i} w_i^T + c_2}.$$

Since for every $v \in S_o$, the degree decreases by at least 1 from G to G_S, $W''_{v,u} \geq W_{v,u}(1 + \frac{1}{d})$, where d is the maximum degree of vertices in S_o of G_S. And for $v \in S_i$, there is no change in degree from G to G_S, so $W_{ii} = W''_{ii}$. Hence we have

$$\begin{aligned}\mathrm{pr}''_{S_i}\left(I - (1-\alpha)W_{ii}\right) &= \alpha s_{S_i} + \mathrm{pr}''_{S_o}\left((1-\alpha)W''_{0i}\right) \\ &\geq \alpha s_{S_i} + \mathrm{pr}''_{S_o}\left((1-\alpha)W_{0i}\right)\left(1 + \frac{1}{d}\right).\end{aligned}$$

So

$$\frac{\mathrm{pr}''_{S_o} w_0^T}{\mathrm{pr}''_{S_i} w_i^T} \leq \frac{\mathrm{pr}_{S_i} w_i^T - c_1}{(1+\frac{1}{d})\mathrm{pr}_{S_i} w_i^T},$$

and the lemma follows from

$$\frac{\mathrm{pr}_{S_i} w_i^T + c_2 - c_1}{\mathrm{pr}_{S_i} w_i^T + c_2} \geq \frac{\mathrm{pr}_{S_i} w_i^T - c_1}{\mathrm{pr}_{S_i} w_i^T} \geq \frac{\mathrm{pr}_{S_i} w_i^T - c_1}{(1+\frac{1}{d})\mathrm{pr}_{S_i} w_i^T} \qquad \square$$

Intuitively, since pr'' ignores all the boundary edges, pr''_{S_o} is inaccurate and underestimated. On the other hand, we want to decrease the influence of boundary nodes, so we want the result PageRank on S_o not to be overestimated compared with pr'_{S_o}. Lemma 1 shows that pr_{S_o} is bounded between pr''_{S_o} and pr'_{S_o}; therefore it is preferred in such cases.

4 Dirichlet PageRank with Given Boundary Conditions

In some cases, we want to further decrease or increase the influence of the boundary nodes, or we already have some estimation of the PageRank on the boundary of some vertex set S and want to approximate the PageRank of S very quickly. Then instead of setting $p(v) = 0$ for all $v \notin S$, we can set them according to arbitrary boundary conditions σ.

The Dirichlet PageRank with given boundary conditions σ is defined by the equations:

$$\mathrm{pr}(v) = \begin{cases} \alpha s(v) + (1-\alpha)\sum_{u\in V} \mathrm{pr}(u) W_{uv} & \text{if } v \in S \\ \sigma(v) & \text{otherwise} \end{cases}, \tag{5}$$

where $\sigma(v) \geq 0, \forall v$ and $|\sigma| \leq 1$.

Let $W_{\delta S}$ denote W restricted to $\delta S \times S$.

Theorem 3. *For a connected graph, the above PageRank equation (5) has one and only one solution. With $\beta = \frac{2\alpha}{1-\alpha}$, it is given by*

$$\mathrm{pr}_S = (\beta s_S + 2\sigma_{\delta S} W_{\delta S}) D_S^{-1/2} \mathcal{G}_{S,\beta} D_S^{1/2}.$$

Proof. Notice that

$$\begin{aligned} \mathrm{pr}_S &= \alpha s_S + (1-\alpha)\left(\mathrm{pr}_S W_S + \sigma_{\delta S} W_{\delta S}\right) \\ &= \frac{1-\alpha}{2}(\beta s_S + 2\sigma_{\delta S} W_{\delta S}) + (1-\alpha)\mathrm{pr}_S W_S. \end{aligned} \tag{6}$$

Then it is straightforward to see the theorem holds by following the steps as in the proof of Theorem 2. $\square$

5 Algorithms and Analysis

To solve the PageRank equations (1, 5) with boundary conditions, as implied by Theorem 2 and Theorem 3, all it requires are vector-matrix multiplication and solving a linear system:

$$x(\beta I_S + \mathcal{L}_S) = y.$$

Since D is diagonal, the complexity of solving the PageRank is just the complexity of solving the linear system. And since the matrix $\beta I_S + \mathcal{L}_S$ is diagonally dominant, it can be solved approximately in nearly linear time with a Spielman-Teng Solver [17]. Here, we show a simpler algorithm ApproxDirichPR to solve the PageRank equations with boundary conditions approximately, which is faster and has a better approximation ratio as long as α is not too small.

The basic outline of our algorithm ApproxDirichPR is as follows: we initialize pr_S as $\mathbf{0}$ and maintain a residue r, which is the difference between the right side and left side of equation 6. Then we gradually move mass from r to pr_S while maintaining the following invariant:

$$\mathrm{pr}_S + r = \alpha s_S + (1-\alpha)\left(\mathrm{pr}_S W_S + \sigma_{\delta S} W_{\delta S}\right)$$

until for every $v \in S$, $r(v) \le \epsilon' d_v$. In the beginning, we set $\epsilon' = 1$; after every iteration, we decrease ϵ' by half until $\epsilon' \le \epsilon$ which is the given desired approximation ratio.

Algorithm 1. ApproxDirichPR

Input: G, S, α, s, σ, ϵ
Output: pr_S

```
pr_S ⇐ 0, ε' ⇐ 1, r ⇐ αs_S + (1 − α)σ_{δS} W_{δS}
while ε' > ε do
    while r(v) ≥ ε' d_v for some v do
        pr_S(v) ⇐ pr_S(v) + r(v)
        For each neighbor u of v, r(u) ⇐ r(u) + (1 − α)r(v)/2d_v
        r(v) ⇐ (1 − α)r(v)/2
    end while
    ε' ⇐ ε'/2
end while
```

Now that we have presented our algorithm, we will prove Theorem 1.

Proof. (of Theorem 1) Since a FIFO queue can be used to store every vertex v such that $r(v) \ge \epsilon' d_v$, each iteration of the inner loop can be done in $O(d_v)$ time. For each iteration of the outer loop, since $|r| \le 2\epsilon' \mathrm{vol}(S)$ at the beginning, and $r(v)$ will decrease at least $\alpha\epsilon' d_v$, let T be the number of iterations and $v_i, 1 \le i \le T$ be the vertex selected at the i-th step, we have

$$\sum_{i=1}^{T} \alpha\epsilon' d_{v_i} \le 2\epsilon' \mathrm{vol}(S),$$

so

$$\sum_{i=1}^{T} d_{v_i} \leq \frac{2\text{vol}(S)}{\alpha}.$$

There are $\log \frac{1}{\epsilon}$ iterations of the outer loop, so the running time is $O(\frac{\text{vol}(S) \log \frac{1}{\epsilon}}{\alpha})$.

The output $\widetilde{\text{pr}}_S$ satisfies:

$$\widetilde{\text{pr}}_S + r = \alpha s_S + (1-\alpha)\left(\widetilde{\text{pr}}_S W_S + \sigma_{\delta S} W_{\delta S}\right),$$

where $0 \leq r(v) < \epsilon d_v \forall v \in S$, and the exact solution pr_S satisfies:

$$\text{pr}_S = \alpha s_S + (1-\alpha)\left(\text{pr}_S W_S + \sigma_{\delta S} W_{\delta S}\right).$$

Subtracting these two equations gives:

$$\text{pr}_S - \widetilde{\text{pr}}_S = \alpha \frac{r}{\alpha} + (1-\alpha)\left((\text{pr}_S - \widetilde{\text{pr}}_S) W_S\right),$$

which is also a PageRank equation. Since r is nonnegative,

$$\widetilde{\text{pr}}_S(v) \leq \text{pr}_S(v), \forall v \in S.$$

And by the properties of PageRank that $|\text{pr}_{\alpha,s}| \leq |s|$ and $|\text{pr}_S| \leq |\text{pr}|$, we have

$$|\text{pr}_S - \widetilde{\text{pr}}_S| \leq |\frac{r}{\alpha}| < \frac{\epsilon \text{vol}(S)}{\alpha}.$$ □

6 Applications of Dirichlet PageRank

6.1 Adjusting Spammers' Influence

One downfall of pure link-based ranking systems such as PageRank is that they interpret all nodes as honest agents and all links as votes or validation between nodes. However, real-world networks such as the World Wide Web often contain malicious nodes or spammers. It then becomes an important question to find ranking systems that better represent the true, honest ranking of nodes in the graph.

There are many schemes developed to try to combat this problem [1,3,4,6,11,12,15,16], but it turns out that many of them can be modeled using Dirichlet PageRank with different boundary conditions. This will allow for the efficient consideration of many different models by simply considering different boundary conditions. For example, [3] outlines an algorithm SpamRank which penalizes spam nodes. Using Dirichlet boundary conditions, we can penalize known spammers v by enforcing the condition $\text{pr}(v) = 0$. One can adjust their ranking even further by enforcing $\text{pr}(v) = -1$.

Another paper [4] concerns propagating trust and distrust within a network, using a weighted random walk W with a trusted seed vertex s. Here, the authors start by assigning rank $\text{pr}(s) = 1$, a condition covered by Dirichlet PageRank with the boundary condition $\sigma(s) = 1$. There is a subtle difference in the way distrust is handled (the original algorithm does not allow for the propagation of trust scores less than 0), but it should be clear that Dirichlet PageRank allows us to efficiently consider these and many other models.

6.2 Adjusting Rank Based on Trust

While it is interesting to be able to devise ranking systems that take known spammers into account, it is also important to calculate a ranking based on various concepts of trust in a network. There are numerous scenarios to consider, and Dirichlet PageRank with boundary conditions will be a useful algorithmic tool.

Consider the following problem: in a network G, node v wants to compute a personalized ranking of the nodes, but v trusts its own friends and wants its ranking on the top ρ fraction of nodes to be similar to its friends'. Presumably one's friends' actions carry a lot of weight in one's own decisions. Vertex v can efficiently compute a personalized PageRank vector as its ranking function using algorithms from [10], but PageRank alone will not take into account the implied trust between v and its friends. But using Dirichlet PageRank with boundary conditions, we can take v's trusted friends into account. We illustrate this in the algorithm PRTrustedFriends.

Algorithm 2. PRTrustedFriends

Input: $G = (V, E)$, v, α, ρ, ϵ
Output: $\boldsymbol{p}$

$\boldsymbol{p} \Leftarrow$ SharpApproximatePR(v, α, ϵ) [10]
$\boldsymbol{p'} \Leftarrow \frac{1}{\sum_{u \sim v} p(u)} \sum_{u \sim v} p(u)$SharpApproximatePR$(u, \alpha, \epsilon)$
$S \Leftarrow \arg\max_{S \subseteq V, |S| \leq \rho |V|} \sum_{s \in S} p(s)$
$\boldsymbol{p} \Leftarrow$ ApproxDirichPR$(G, V \setminus S, \alpha, v, \boldsymbol{p'}, \epsilon)$

A natural extension of PRTrustedFriends is a similar problem where v is a newcomer to a network and is therefore unsure about what other nodes are trustworthy. In such a scenario, the only available information to v is the network itself. For ranking purposes, v can select a small sample of nodes to compare with its own ranking; if these nodes are well distributed, they provide a good control to ensure that v's own ranking function is too distorted by the presence of nearby spam or malicious nodes. We give the algorithm PRValidation.

Algorithm 3. PRValidation

Input: $G = (V, E)$, v, k, α, ρ, ϵ
Output: $\boldsymbol{p}$

$\boldsymbol{p} \Leftarrow$ SharpApproximatePR(v, α, ϵ) [10]
$v_1, \ldots, v_k \Leftarrow$ i.i.d. samples from V according to $\boldsymbol{p}$
$\boldsymbol{p'} \Leftarrow \frac{1}{\sum_{i=1}^{k} p(v_k)} \sum_{i=1}^{k} p(v_k)$SharpApproximatePR$(u, \alpha, \epsilon)$
$S \Leftarrow \arg\max_{S \subseteq V, |S| \leq \rho |V|} \sum_{s \in S} p(s)$
$\boldsymbol{p} \Leftarrow$ ApproxDirichPR$(G, V \setminus S, \alpha, v, \boldsymbol{p'}, \epsilon)$

A third, more complex situation arises in the context of different types of social networks. Although the problem setup here appears rather complicated,

it is a natural model for a common social phenomenon: a distinction between different types of social graphs.

Suppose that a vertex v is part of two networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \subseteq V_2$. We interpret G_1 as a closely-knit social network where edges represent a deeper connection with the implication that the endpoints share mutual trust for one another. G_2 is a larger network where nodes form edges for looser reasons; for example, acquaintanceship or curiosity. We assume that v does not know much about the many sources of information present in G_2. An important question for v is: which nodes in G_2 are trustworthy? Is there some way to rank the vertices of G_2?

One effective way of finding such a ranking of vertices in G_2 for a node v is by computing the ranking on G_1 and then computing Dirichlet PageRank on G_2 using G_1's ranking as the boundary condition. This is outlined in the algorithm PRTrustNetwork.

Algorithm 4. PRTrustNetwork

Input: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, $v \in V_1 \cap V_2$, α, ϵ
Output: $\boldsymbol{q}$

$\boldsymbol{p} \Leftarrow$ SharpApproximatePR(v, α, ϵ) [10] for G_1
$\boldsymbol{q} \Leftarrow$ ApproxDirichPR$(G_2, V_2 \setminus V_1, \alpha, v, \boldsymbol{p}, \epsilon)$

Dirichlet PageRank can also be used to solve a related problem if a global ranking for G_2 is already known or pre-computed. Suppose that such a ranking exists, and $v \in G_2$ wants to be able to rank its own more personal network or neighborhood G_1 taking this into account. One way to do this is to compute a Dirichlet PageRank vector for G_1, but using the nodes adjacent to G_1 as a boundary with rank given by the global ranking on G_2. This procedure is given in the following algorithm PRInferRanking.

Algorithm 5. PRInferRanking

Input: $G_2 = (V_2, E_2)$, $G_1 = (V_1, E_1) \subseteq G_2$, $v \in V_1$, $\boldsymbol{p}$, α, ϵ
Output: $\boldsymbol{q}$

$\partial E_1 \Leftarrow \{(w, x) \in E_2 | w \in V_1, x \notin V_1\}$
$\partial V_1 \Leftarrow \{w \in V_2 \setminus V_1 | w$ is an endpoint of an $e \in \partial E_1\}$
$\boldsymbol{q} \Leftarrow$ ApproxDirichPR$((V_1 \cup \partial V_1, E_1 \cup \partial E_1), V_1, \alpha, v, \boldsymbol{p}, \epsilon)$

From the examples above, we see that Dirichlet PageRank with boundary conditions is a useful tool, especially in modeling trust and distrust in a network ranking system. It is of interest to further take advantage of the efficient computation and approximation of Dirichlet PageRank vectors. More applications and directions remain to be explored.

References

1. Andersen, R., Borgs, C., Chayes, J., Feige, U., Flaxman, A., Kalai, A., Mirrokni, V., Tennenholtz, M.: Trust-based recommendation systems: an axiomatic approach. In: WWW 2008 (2008)
2. Baeza-Yates, R., Castillo, C., López, V.: PageRank increase under different collusion topologies. In: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (2005)
3. Benczur, A., Csalogany, K., Sarlos, T., Uher, M.: SpamRank — fully automatic link spam detection. In: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (2005)
4. Borgs, C., Chayes, J., Kalai, A.T., Malekian, A., Tennenholtz, M.: A novel approach to propagating distrust. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 87–105. Springer, Heidelberg (2010)
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30(1-7), 107–117 (1998)
6. Cheng, A., Friedman, E.: Sybilproof reputation mechanisms. In: Proceedings of Third Workshop on Economics of Peer-to-Peer Systems (2005)
7. Chung, F.: PageRank as a discrete Green's function. Geometry and Analysis I ALM 17, 285–302 (2010)
8. Chung, F.: Spectral Graph Theory. AMS Publications, Providence (1997)
9. Chung, F., Yau, S.-T.: Discrete Green's functions. J. Combinatorial Theory (A) 91, 191–214 (2000)
10. Chung, F., Zhao, W.: A sharp PageRank algorithm with applications to edge ranking and graph sparsification. In: Kumar, R., Sivakumar, D. (eds.) WAW 2010. LNCS, vol. 6516, pp. 2–14. Springer, Heidelberg (2010)
11. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: WWW 2004 (2004)
12. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating Web spam with TrustRank. In: VLDB 2004 (2004)
13. Haveliwala, T.: Topic-sensitive PageRank: A context-sensitive ranking algorithm for Web search. IEEE Transactions on Knowledge and Data Engineering 15, 784–796 (2004)
14. Jeh, G., Widom, J.: Scaling personalized Web search. In: WWW 2003 (2003)
15. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: WWW 2003 (2003)
16. de Kerchove, C., Dooren, P.: The PageTrust algorithm: how to rank Web pages when negative links are allowed? In: Proceedings of the SIAM International Conference on Data Mining (2008)
17. Spielman, D.A., Teng, S.-H.: Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems (2008), http://arxiv.org/abs/cs.NA/0607105

Efficient Generation of Networks with Given Expected Degrees

Joel C. Miller[1,2] and Aric Hagberg[3]

[1] Center for Communicable Disease Dynamics, Harvard School of Public Health, Boston, MA 02115, USA
[2] Fogarty International Center, National Institute of Health, Bethesda, MD 20892, USA
joel.c.miller.research@gmail.com
[3] Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
hagberg@lanl.gov

Abstract. We present an efficient algorithm to generate random graphs with a given sequence of expected degrees. Existing algorithms run in $\mathcal{O}(N^2)$ time where N is the number of nodes. We prove that our algorithm runs in $\mathcal{O}(N+M)$ expected time where M is the expected number of edges. If the expected degrees are chosen from a distribution with finite mean, this is $\mathcal{O}(N)$ as $N \to \infty$.

1 Introduction

Random graph models are regularly used for studying random processes on networks. Such processes include rumor spreading or the spread of epidemics on social networks [15,13,9], and web searchers seeking information on the World Wide Web [4,14]. Capturing the degree distribution is one of the most important goals in creating a model and some well-understood graph models have been developed where the degree distribution is controlled. In this paper we focus on the model of Chung and Lu [5] which generates a random graph with a given sequence of expected degrees.

Because application networks are often very large, efficient random graph generation is important to evaluate the models and processes. Although many theoretical results are known about the Chung-Lu model [2,12,5,6,7,8], the algorithms used for generating such graphs are inefficient. In this paper we introduce a new algorithm which generates Chung-Lu random graphs with expected runtime that is $\mathcal{O}(N + M)$ where N the number of nodes and M is the expected number of edges. For sequences with finite average degree $M = \mathcal{O}(N)$ and the expected runtime is $\mathcal{O}(N)$, a significant improvement over previous algorithms that require $\mathcal{O}(N^2)$ runtime.

We begin by showing a fast algorithm for generating Erdős–Rényi graphs. This approach is by itself not a significant result (indeed closely related algorithms exist already [1]), but it serves as an example to motivate our algorithm for generating Chung-Lu graphs which uses an additional rejection sampling step.

A. Frieze, P. Horn, and P. Prałat (Eds.): WAW 2011, LNCS 6732, pp. 115–126, 2011.

Once we introduce the main algorithm, we prove that it generates Chung-Lu graphs and that its expected runtime is $\mathcal{O}(N+M)$. We conclude with a discussion of other applications and implications of this algorithm.

1.1 Model Description

The basic random graph model is the Erdős–Rényi graph $G(N,p)$ with N nodes $0, 1, \ldots, N-1$. Each pair of nodes has an edge with probability p, and edges between a pair of nodes are assigned independently of one another. The expected degree of a node is $\kappa = p(N-1)$. Typically we consider graphs for which κ is $\mathcal{O}(1)$ as $N \to \infty$, so p is small and the graph is sparse. Under such conditions, the expected degree approaches pN as $N \to \infty$. The obvious algorithm to generate $G(N,p)$ graphs requires considering each of the $\mathcal{O}(N^2)$ pairs of nodes independently and assigning an edge with probability p. However, it has been shown that the runtime can be reduced to $\mathcal{O}(N+M)$ where N is the number of nodes and M the number of edges [1].

The degree distribution resulting from the Erdős–Rényi model is binomial, and in the large N, constant pN limit it approaches a Poisson distribution with mean pN. Many real world graphs have much more pronounced heterogeneity in degrees [16,2]. This has led to models which attempt to incorporate more heterogeneity. Some of these models retain independence of edges but allow nodes to have different expected degrees.

Of these models the most prominent was introduced by Chung and Lu [6]. In the Chung-Lu model each node u is assigned a weight w_u which we can assume is chosen from a distribution with density ρ. We do not need to restrict w_u to be an integer. We define $\overline{w} = \sum_u w_u/N$ to be the average weight. Two nodes u and v with weights w_u and w_v are then joined by an edge with probability $p_{u,v} = w_u w_v/N\overline{w}$. Looking at all nodes $v \neq u$, we anticipate that the expected degree of u is $\sum_{v\neq u} w_u w_v/N\overline{w} = w_u - w_u^2/N\overline{w}$. In a large graph this typically converges to w_u. So the weight w_u represents the expected degree of node u in a large graph. A number of theoretical results are known about this model [2,12,6,8,7]. For a Chung-Lu model graph, we will use $M = N\overline{w}/2$ as our estimate for the expected number of edges. This is an upper bound, but usually this is only a small effect.

We note that it is possible that $w_u w_v/N\overline{w} > 1$. In this case, we set $p_{u,v} = 1$, and the expected degree of u will be less than $w_u - w_u^2/N\overline{w}$. Other approaches have been introduced to produce related graphs that avoid this difficulty. For example, we could define $p_{u,v} = 1 - \exp(-w_u w_v/N\overline{w})$ or $p_{u,v} = w_u w_v/(N\overline{w} + w_u w_v)$ [17,3,12]. Typically as N grows the difference between these approaches is negligible because $w_u w_v/N \to 0$ and thus the leading order terms for $p_{u,v}$ are the same.[1] We will focus our attention on the Chung-Lu graphs, though our algorithm can be easily translated to the others. A fast algorithm to generate approximate Chung-Lu graphs in the bipartite case appears in Ref. [10].

[1] In distributions for which the average squared weight is infinite, but the average weight is finite, $w_u w_v/N$ may not tend to zero for the highest weight nodes, in which case these models differ.

2 Erdős–Rényi Case

We begin by describing our algorithm in a simpler context. The Erdős–Rényi network is a special case of Chung-Lu networks in which all nodes have the same weight w and therefore $p_{u,v} = p = w/N$ for all pairs u and v. We first describe the obvious inefficient $\mathcal{O}(N^2)$ algorithm, and show how it can be naturally sped up to an algorithm that is $\mathcal{O}(N + M)$.

Let the the nodes be numbered $0, 1, \ldots, N-1$ and begin by setting $u = 0$. Then for each $v = 1, 2, \ldots, N-1$ we generate a random number r. If $r < p$, then we place an edge from u to v. Once all possible choices for v have been considered, we set $u = 1$, and then consider each $v = 2, 3, \ldots, N-1$. We continue this process until all possible choices for u have been considered. This algorithm is $\mathcal{O}(N^2)$ because it considers each pair of nodes separately.

This algorithm is slow because considerable effort is spent on node pairs which never form edges. The algorithm can be sped up by skipping these pairs. Returning to the first pass through the algorithm described above with $u = 0$, let v_1 be the first neighbor with which u forms an edge. The value of v_1 is $u + 1 + \delta$ where δ is the number of pairs considered that do not form edges before the first successful edge formation. Similarly, the second neighbor v_2 is $v_1 + 1 + \delta$ where δ is the new number of pairs that do not form edges. The probability of a particular value of δ is $(1-p)^\delta p$ (in fact, δ is geometrically distributed). Thus, rather than considering every node after u as above, we can find the next neighbor in a single step by choosing r uniformly in $(0, 1)$ and setting $\delta = \lfloor \ln r / \ln(1-p) \rfloor$, taking $\delta = 0$ if $p = 1$. The full procedure for $p < 1$ is presented in Algorithm 1.

Algorithm 1. $G(N, p)$ Graph

Input: number of nodes N, and probability $0 < p < 1$
Output: $G(N, p)$ graph $G(V, E)$ with $V = \{0, \ldots, N-1\}$

```
E ← ∅
for u = 0 to N − 2 do
    v ← u + 1
    while v < N do
        choose r ∈ (0, 1) uniformly at random
        v ← v + ⌊log(r) / log(1 − p)⌋
        if v < N then
            E ← E ∪ {u, v}
            v ← v + 1
```

Theorem 1. *Algorithm 1 runs in $\mathcal{O}(N + M)$ time where M is the number of edges in the output graph and N the number of nodes.*

Proof. The proof of this theorem is relatively straightforward. We simply count the number of times that each loop executes.

The outer loop is executed $N-1$ times. To calculate the number of executions of the inner loop we separate those "successful" iterations where the new v is at most $N-1$ from those "unsuccessful" iterations with $v \geq N$. In each pass through the outer loop, the inner loop executes once unsuccessfully. The total number of successful executions of the inner loop is exactly the number of edges that are generated which is $\mathcal{O}(M)$. Combining the total number of passes through the outer loop with the number of successful and unsuccessful passes through the inner loop the runtime is $\mathcal{O}(M+N)$. □

This runtime is exact in the sense that the time taken to generate a given graph is $\mathcal{O}(M+N)$ where M is the actual number of edges in the graph produced: in a worse-case scenario the bound above remains correct, but M is large. For the Chung-Lu graphs our results are less precise: we bound the expected runtime in terms of the expected number of edges.

An algorithm similar to Algorithm 1 is described in [1] which avoids the unsuccessful iterations of the inner loop. However, our approach lends itself to the generalization we describe below.

3 Chung-Lu Case

Having set the framework with the Erdős–Rényi case we now describe an algorithm to create Chung-Lu networks where not all nodes have the same weight. As before, we want to skip as many nodes as possible, but this is more difficult because the probabilities that any pair of nodes have an edge are not fixed. To simplify this, we assume that we have a list W of the weights in the network, and that this list is sorted in descending order.

The obvious $\mathcal{O}(N^2)$ algorithm considers each pair u and v and assigns an edge with probability $p_{u,v} = w_u w_v / N\overline{w}$. We present a slightly different $\mathcal{O}(N^2)$ algorithm, which is easily modified the same way we altered the Erdős–Rényi algorithm to create an $\mathcal{O}(M+N)$ algorithm.

Starting with $u = 0$, we consider every $v = 1, 2, \ldots, N-1$ in turn. As v increases, $p_{u,v}$ decreases monotonically, so we can avoid recalculating p for each v by setting $p = p_{u,u+1} = w_u w_{u+1} / N\overline{w}$ initially and discarding each v with probability $1-p$. When we arrive at the first node v_1 which is not discarded, we call v_1 a *potential neighbor*. We have selected v_1 with probability p, but the probability of an edge between v_1 and u is actually $q \leq p$. We calculate $q = p_{u,v_1}$, and then assign an edge with probability q/p. We then set $p = q$ and continue on, discarding nodes with probability $1-p$ until we have considered all possible nodes. We then increase u by 1 and repeat. This algorithm is $\mathcal{O}(N^2)$.

In the algorithm just described, p is fixed at each step until a potential neighbor is identified. The same method used in the Erdős–Rényi approach can quickly identify the potential neighbors v_i without considering each intermediate v in turn. Starting with $u = 0$ and using $p = p_{u,u+1}$, we choose a random number r uniformly in $(0,1)$ and find the first potential neighbor $v_1 = u + 1 + \delta$ where $\delta = \lfloor \ln r / \ln(1-p) \rfloor$. If $p = 1$, we take $\delta = 0$. Once v_1 is identified, we assign an edge between u and v_1 with probability q/p where $q = p_{u,v_1}$. We then set $p = q$

and continue, jumping immediately to the next potential neighbor v_2, possibly placing an edge. Again resetting p, we continue until there are no more nodes to consider. We then increase u by 1 and repeat. Ultimately, u takes all possible values, and the set of edges is complete. Note that for given u the value of p decreases monotonically, so the expected value of δ increases monotonically.

The Chung-Lu graph generating procedure is presented in Algorithm 2.

Algorithm 2. Chung-Lu Graph

Input: list of N weights, $W = w_0, \ldots, w_{N-1}$, sorted in decreasing order
Output: Chung-Lu graph $G(V, E)$ with $V = \{0, \ldots, N-1\}$

```
E ← ∅
S ← Σ_u w_u
for u = 0 to N − 2 do
    v ← u + 1
    p ← min(w_u w_v / S, 1)
    while v < N and p > 0 do
        if p ≠ 1 then
            choose r ∈ (0, 1) uniformly at random
            v ← v + ⌊log(r) / log(1 − p)⌋
        if v < N then
            q ← min(w_u w_v / S, 1)
            choose r ∈ (0, 1) uniformly at random
            if r < q/p then
                E ← E ∪ {u, v}
            p ← q
            v ← v + 1
```

Proposition 1. *Algorithm 2 generates Chung-Lu graphs.*

Proof. We need to prove that an edge is assigned from node u to node v with probability $p_{u,v}$ that is independent of other edges.

Consider a given u and $v > u$. Let $\hat{p}$ represent the value of p when the inner loop reaches (or passes) v while assigning edges for u. This is the probability with which v is selected as a potential neighbor. The value of $\hat{p}$ is influenced by other edges which have been assigned to u. Our goal is to show that regardless of the value of $\hat{p}$, an edge is placed between u and v with probability $p_{u,v}$.

Because of the ordering of the weights we know that $\hat{p} \geq p_{u,v}$. If v is selected as a potential neighbor, then with probability $q = p_{u,v}/\hat{p} \leq 1$, v will become an actual neighbor of u. Thus the probability that an edge exists between u and v is $\hat{p}q = p_{u,v}$. This is independent of $\hat{p}$, so it is independent of any previous edges that have been assigned. By similar argument it has no influence on what later edges will be assigned. □

3.1 Efficiency

For the Chung-Lu algorithm it is more difficult to bound the number of steps because of the rejection sampling: in some cases a potential neighbor v_i is

identified, but upon closer inspection no edge is placed to v_i. We refer to these as *excess potential neighbors*. Let m be the number of actual neighbors and l the number of excess potential neighbors generated in a particular realization. Let L be the expected value of l. We define $M = N\overline{w}/2$, and because each node's expected degree is at most its weight we know that the expected value of m is at most M. For many relevant weight distributions, the expected value of m approaches M as $N \to \infty$. L is more difficult to bound. We will prove that $L = \mathcal{O}(N + M)$, and so the algorithm executes in $\mathcal{O}(N + M)$ expected time.

Theorem 2. *Algorithm 2 executes in $\mathcal{O}(N + M)$ expected time where $M = N\overline{w}/2$ and N is the number of nodes.*

Proof. We follow a similar argument to the Erdős–Rényi case, and conclude that the runtime is $\mathcal{O}(N+m+l)$ where m is the actual number of edges created and l the number of excess potential neighbors. We will show that the expected value of l is $L = \mathcal{O}(N + M)$, and since the expectation of m is at most M, the expected runtime is $\mathcal{O}(N + M)$. However, the calculation of L is considerably more technical, and is the focus of our proof. We will bound the probability that each node is selected as an excess neighbor of another node. Summing this gives a bound on L.

Consider a given u and $v > u$. Let $\rho_{u,v}(\hat{p})$ be the *a priori* probability that the value of p is $\hat{p}$ when the inner loop reaches (or passes) v while assigning edges for u. Define $P_{u,v} = \sum_{\hat{p}} \rho_{u,v}(\hat{p})\hat{p}$, the *a priori* probability that v will be chosen as a potential neighbor of u. The probability that v will be selected as an excess potential neighbor is $P_{u,v} - p_{u,v}$. We seek to calculate $P_{u,v}$.

We know that $P_{u,u+1} = p_{u,u+1}$. We look to find $P_{u,v+1} - P_{u,v}$ for all v. This requires calculating the change in $\rho_{u,v}(\hat{p})$ from v to $v + 1$. If v is not chosen as a potential neighbor, there is no change to p, but if v is chosen, then p changes from $\hat{p}$ to $p_{u,v}$ and so the change in p is $p_{u,v} - \hat{p}$. This occurs with probability p. So the change in $P_{u,v}$ is the expected change in p, which we define to be $\Delta P_{u,v} = P_{u,v+1} - P_{u,v}$. To make $\Delta P_{u,N-1}$ defined, it is convenient to set $P_{u,N}$ to be the value P would take for node N if the node list were extended by adding an additional node with weight 0. Note that $\Delta P_{u,v} \leq 0$. We have

$$\begin{aligned}
\Delta P_{u,v} &= P_{u,v+1} - P_{u,v} \,, \\
&= \sum_{\hat{p}} \rho_{u,v}(\hat{p})\hat{p}\,(p_{u,v} - \hat{p}) \,, \\
&= P_{u,v}p_{u,v} - \sum_{\hat{p}} \rho_{u,v}(\hat{p})\hat{p}^2 \,, \\
&\leq P_{u,v}\,(p_{u,v} - P_{u,v}) \,, \qquad (1)
\end{aligned}$$

using Jensen's inequality to say that the expectation of p^2 is at least the square of the expectation of p. Note that $P_{u,v}$ decreases monotonically with v and that $P_{u,v}$ cannot be less than $p_{u,v}$.

We now define $\zeta(u)$ to be the number of excess potential neighbors node u is expected to have,

$$\zeta(u) = \sum_{v=u+1}^{N-1} P_{u,v} - p_{u,v} \,. \tag{2}$$

From our bound (1) for $\Delta P_{u,v}$, we can bound $\zeta(u)$ as

$$\zeta(u) \leq \sum_{v=u+1}^{N-1} \frac{-\Delta P_{u,v}}{P_{u,v}} \,.$$

By analogy with the integral $-\int_a^b \phi'(x)/\phi(x)\,\mathrm{d}x$, we anticipate that this summation behaves like a logarithm, and we use this to bound the sum. We use the inequality for $x > -1$ that $\ln(1+x) \leq x$, implying $-x \leq -\ln(1+x)$. For simplicity, we extend the range to include $x = -1$ by allowing for infinity. Then

$$\begin{aligned}
\frac{-\Delta P_{u,v}}{P_{u,v}} &\leq -\ln\left(1 + \frac{\Delta P_{u,v}}{P_{u,v}}\right) \,, \\
&\leq -\ln \frac{P_{u,v} + \Delta P_{u,v}}{P_{u,v}} \,, \\
&\leq -\ln \frac{P_{u,v+1}}{P_{u,v}} \,, \\
&\leq \ln P_{u,v} - \ln P_{u,v+1} \,.
\end{aligned}$$

So $\zeta(u)$ is bounded by a telescoping summation,

$$\zeta(u) \leq \ln P_{u,u+1} - \ln P_{u,N} \,.$$

It is difficult to bound $P_{u,N}$ away from zero. So instead we break the sum in Eq. (2) into terms for which P can be easily bounded away from zero and those for which it is more difficult. Set x to be the first node such that $w_x < 1$, so $w_{x-1} \geq 1$. Assume for now $u < x$. We have

$$\zeta(u) \leq \sum_{v=u+1}^{x-1} \frac{-\Delta P_{u,v}}{P_{u,v}} + \sum_{v=x}^{N-1} P_{u,v} - p_{u,v} \,.$$

The first summation is at most

$$\ln P_{u,u+1} - \ln P_{u,x} = \ln[P_{u,u+1}/P_{u,x}] \,.$$

We have

$$P_{u,u+1} = p_{u,u+1} = \min(w_u w_{u+1}/N\overline{w}, 1) \leq w_u w_{u+1}/N\overline{w} \,,$$

while

$$P_{u,x} \geq \min(w_u w_{x-1}/N\overline{w}, 1) \geq w_u/N\overline{w} \,.$$

Thus $P_{u,u+1}/P_{u,x} \leq w_{u+1}$. The first summation is at most $\ln w_{u+1}$, which in turn is at most $\ln w_u$.

The second summation can be bounded by observing that the expected number of excess potential neighbors in $[x, N-1]$ is at most the expected number of potential neighbors in $[x, N-1]$. Assuming that u has at least one potential neighbor $v \geq x$, the probability for any later node to be a potential neighbor is at most $w_u w_v / N\overline{w} < w_u / N\overline{w}$. There are $N-1-x$ nodes in this region, which is bounded by N, so u has at most $1 + N w_u / N\overline{w}$ expected further potential neighbors in $[x, N-1]$. This gives an upper bound on the second sum. So if $u < x$, $\zeta(u) \leq 1 + \ln w_u + w_u/\overline{w}$.

If $u \geq x$, then the approach used above for the second summation gives an upper bound of $\zeta(u) \leq 1 + w_u N / N\overline{w}$.

Both bounds for $u < x$ and $u \geq x$ can be replaced by

$$\zeta(u) \leq 1 + w_u + \frac{w_u}{\overline{w}} ,$$

for any u. This is a significant overestimate, but will not weaken our final bound, and we discuss it further in the next section. We sum $\zeta(u)$ over all nodes and get

$$\begin{aligned} L &= \sum_{u=0}^{N-1} \zeta(u) , \\ &\leq \sum_{u=0}^{N-1} 1 + w_u + \frac{w_u}{\overline{w}} , \\ &\leq N + N\overline{w} + \frac{N\overline{w}}{\overline{w}} , \\ &\leq 2M + 2N . \end{aligned}$$

Therefore $L = \mathcal{O}(N+M)$, and we have that $\mathcal{O}(N+M+L) = \mathcal{O}(N+M)$. So the algorithm executes in $\mathcal{O}(N+M)$ expected time. □

4 Examples

We demonstrate the runtime of Algorithm 2 using three different weight distributions. The first has weights chosen uniformly in $(1, 50)$, giving an average of 25.5. The second has all weights equal to 25. The third has weights chosen from a power law distribution with exponent $\gamma = -2.1$, and every weight above 100 is set to 100, giving an average of about 4.7. We generate graphs on N nodes where the weights are chosen from each of these distributions. In Fig. 1 we show that the execution time is $\mathcal{O}(M+N)$. In contrast, the standard algorithm for generating these scales like $\mathcal{O}(N^2)$.

The proof of Theorem 2 shows that the total expected excess neighbors is $L = \mathcal{O}(M+N)$. Reducing L reduces the runtime so we now look at L more closely. The bound used for L is fairly crude. It was derived by replacing $\ln w_{u+1}$ with w_u and assuming that every node would have at least one neighbor of weight $w < 1$.

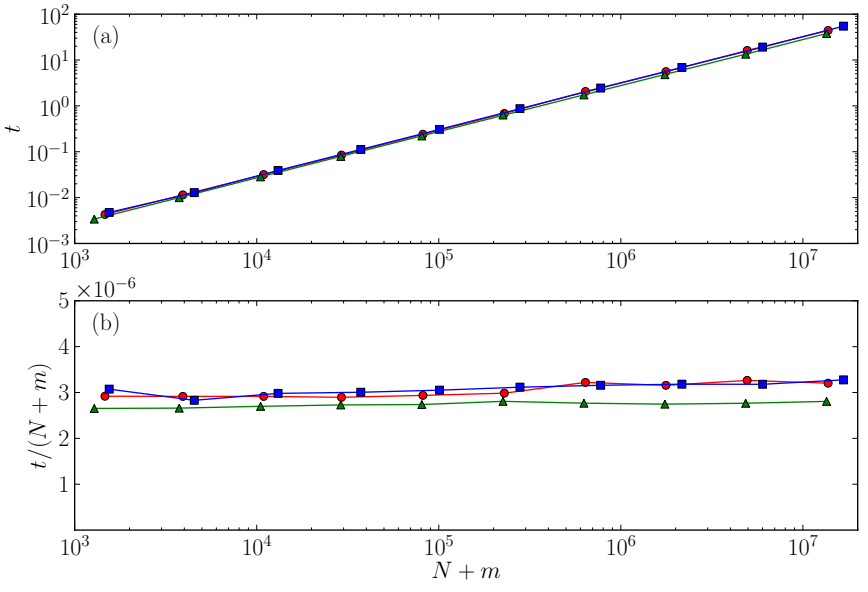


Fig. 1. Performance of Algorithm 2 showing linear scaling in the number of nodes and edges $N+m$. The 3 curves are data for weights chosen from the following distributions (red circles) uniform on (1,50); (green triangles) constant $w = 25$; and (blue squares) power law with exponent $\gamma = -2.1$. (a) Running time vs number of nodes and edges. (b) Estimate of the coefficient.

Analyzing the algorithm closer, we see that to have larger L, there is a competition: a node is more likely to be an excess potential neighbor if the weights before it are large, but its weight is small. However, such a node reduces the probability that nodes after it become excess potential neighbors. Thus heterogeneity in weights plays a role in determining the number of excess potential neighbors.

To investigate this further, we show the total number of excess potential neighbors generated by the algorithm in Fig. 2. In Fig. 2(a) we consider the same three distributions as before. We find that L/M, the number of excess potential neighbors created for each edge is largely independent of the number of nodes, and that if all nodes have the same weight $L = 0$. Although we have proven that $L \leq 2M + 2N$, the numerical experiments suggest that stronger bounds are possible and depend on the heterogeneity in weights.

In Fig. 2(b) we show the interplay of heterogeneity and average weight more closely. We considered two classes of distributions. In one, w is chosen uniformly in $(\overline{w}-5, \overline{w}+5)$, and in the other w is chosen uniformly in $(1, 2\overline{w}-1)$. We see that the number of excess potential neighbors generated per node L/N decreases in the first case and increases slowly in the second case. In both cases M increaseslinearly with $\overline{w}$, so L/M decreases to zero. The excess potential neighbors become a negligible contribution to runtime as graph size increases.

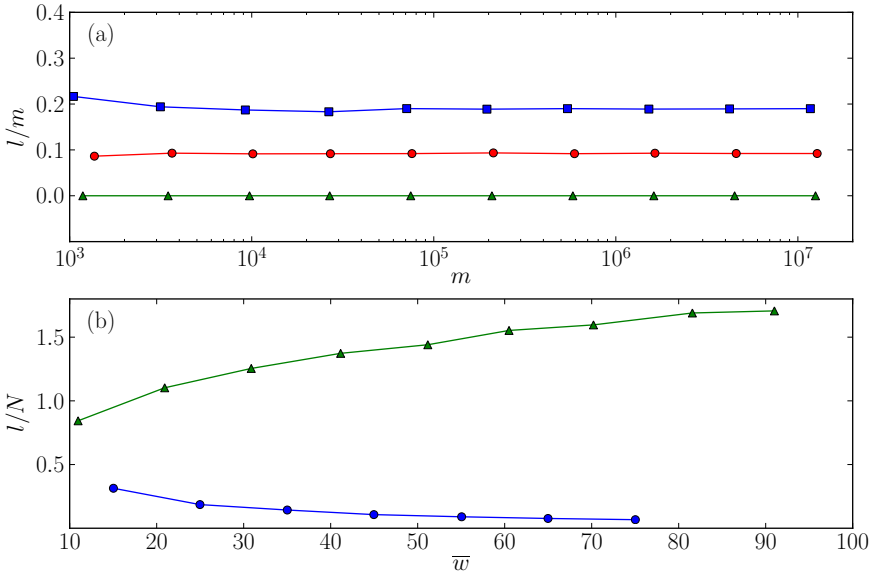


Fig. 2. Fraction of "excess potential neighbors" l/m generated by Algorithm 2. (a) The fraction l/m vs total number of edges in the graph m for weights from the three distributions of Fig. 1. (b) The fraction l/m for uniformly distributed sequences with fixed variance: w chosen uniformly from $(\overline{w}-5, \overline{w}+5)$ (blue circles), and with fixed coefficient of variance $\mathrm{std}(W)/\overline{w}$: w chosen uniformly from $(1, 2\overline{w}-1)$ (green triangles).

5 Discussion

We have developed an algorithm for creating Chung-Lu random graphs in $\mathcal{O}(M+N)$ expected runtime. Our algorithm may be generalized to other contexts. In particular, it may also be used to generate the random graphs introduced in Refs. [17,3]. This algorithm requires first that there is a single parameter w assigned to each node which determines the probability that any two nodes share an edge, and second that the nodes may be ordered in such a way that if u appears before v_1 which appears before v_2, then the probability that u has an edge with v_1 is at least the probability that u has an edge with v_2. It is possible to generate many other graph models in this manner, including models which have assortative mixing (nodes with similar weights are preferentially connected).

Some graph models, such as the configuration model, do not assign edges independently, and so have somewhat different generation algorithms. In the configuration model each node is assigned a degree *a priori*, and then nodes are wired together subject to the assigned degrees as a constraint. An efficient algorithm to do this begins by placing nodes into a list once for each edge the node will have. The list is then shuffled, and adjacent nodes are joined by an edge. Consequently edges may be repeated, nodes may have edges to themselves, and if the sum of degrees is odd, a node is left unpaired. Typically the number

of such edges is small compared to the number of nodes, so these are simply discarded. It is possible to avoid these cases, but even the most efficient known algorithms that produce true graphs with the imposed degree distribution are substantially slower than $\mathcal{O}(N + M)$ [11].

Acknowledgments

JCM was supported by 1) the RAPIDD program of the Science and Technology Directorate, Department of Homeland Security and the Fogarty International Center, National Institutes of Health and 2) the Center for Communicable Disease Dynamics, Department of Epidemiology, Harvard School of Public Health under Award Number U54GM088558 from the National Institute Of General Medical Sciences. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute Of General Medical Sciences or the National Institutes of Health. Part of this work was funded by the Department of Energy at Los Alamos National Laboratory under contract DE-AC52-06NA25396, and the DOE Office of Science Advanced Computing Research (ASCR) program in Applied Mathematics.

References

1. Batagelj, V., Brandes, U.: Efficient generation of large random networks. Physical Review E 71, 036113 (2005)
2. Bollobás, B., Janson, S., Riordan, O.: The phase transition in inhomogeneous random graphs. Random Structures and Algorithms 31(1), 3 (2007)
3. Britton, T., Deijfen, M., Martin-Löf, A.: Generating simple random graphs with prescribed degree distribution. Journal of Statistical Physics 124(6), 1377–1397 (2006)
4. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the Web. In: WWW9: 9th International World Wide Web Conference, vol. 33, pp. 309–320 (2000)
5. Chung, F., Lu, L.: Connected components in random graphs with given expected degree sequences. Annals of Combinatorics 6(2), 125–145 (2002)
6. Chung, F., Lu, L.: The average distance in a random graph with given expected degrees. Internet Mathematics 1(1), 91–113 (2004)
7. Chung, F., Lu, L.: The volume of the giant component of a random graph with given expected degrees. SIAM Journal on Discrete Mathematics 20(2), 395–411 (2007)
8. Chung, F., Lu, L., Vu, V.: The spectra of random graphs with given expected degrees. Internet Mathematics 1(3), 257–275 (2004)
9. Draief, M., Ganesh, A.: A random walk model for infection on graphs: spread of epidemics & rumours with mobile agents. Discrete Event Dynamic Systems 21, 41–61 (2011)
10. Eubank, S., Kumar, V.S., Marathe, M.V., Srinivasan, A., Wang, N.: Structural and algorithmic aspects of massive social networks. In: Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 718–727 (2004)

11. del Genio, C.I., Kim, H., Toroczkai, Z., Bassler, K.E.: Efficient and exact sampling of simple graphs with given arbitrary degree sequence. PloS ONE 5(4), e10012 (2010)
12. van der Hofstad, R.: Critical behavior in inhomogeneous random graphs, arXiv:0902.0216v2 [math.PR] (2010)
13. Keeling, M.J., Eames, K.T.D.: Networks and epidemic models. Journal of The Royal Society Interface 2(4), 295–307 (2005), doi:10.1098/rsif.2005.0051
14. Kleinberg, J.: Complex networks and decentralized search algorithms. In: Proceedings of the International Congress of Mathematicians, ICM (2006)
15. Newman, M.E.J.: Spread of epidemic disease on networks. Physical Review E 66(1), 016128 (2002)
16. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
17. Norros, I., Reittu, H.: On a conditionally Poissonian graph process. Advances in Applied Probability 38(1), 59–75 (2006)

Author Index