A brief survey of PageRank algorithms

Fan Chung [*]

Abstract

We examine several PageRank approximation algorithms. Quantitative analyses are provided to illustrate the extraordinary effectiveness of the PageRank computation.

1 Introduction

In the past decade, the landscape of information technology has dramatically changed. One of the driving forces is the super robust and incredibly fast Web search algorithms which are capable of handling massive data sets of enormous size.

In this paper, we examine a number of PageRank algorithms, ranging from the simple iterative methods to the efficient random walk based algorithms. We will give a pseudo code together with quantitative analysis concerning computational complexity and error bounds for each approximation algorithms, if any.

It is worth pointing out the analysis of PageRank algorithms is quite different from the usual algorithmic analysis, which can be summarized as follows.

- For PageRank algorithms, the graphs under consideration are usually of enormous size, for example, often having more than 10^{10} nodes. The number of vertices n could be so large so that a 'sweep' of nodes should not be carried out lightly. The typical PageRank algorithms have computational complexity of order $\log n$ or $\log |S|$ where S is some subset of the nodes, for example, representing a local community.

[*]University of California, San Diego. Research supported in part by AFSOR FA9550-09-1-0090

- The error estimate for the ranking vector (or any vector involved in computation) is quite different. The usual notion for estimating errors or analyzing the rate of convergence for random walks is the so-called variation distance. Basically, the variation distance between two vectors is the accumulated sum of the errors over all coordinates (or half of the L_1 distance). For ranking algorithms dealing with massive graphs, the approximation bound is much weaker in the sense that we are not concerned with accumulated errors. In fact, all vectors involved in computation should have finite support (e. g. of size independent of n). Namely, all but a finite number of nodes can have been assigned zero. As we will see later, we only require non-trival values falling within an additive or multiplicative error term from the true value.

- In graph theory the usual notion of distance is the length of a shortest path joining two nodes. However, real-world graphs exhibit the so-called "small world phenomenon". Any pair of vertices are connected through a very short path (as happens in random graphs [10]). Therefore the usual notion of graph distance is no longer very useful. Instead, we need a quantitative and precise formulation to differentiate between nodes that are 'local' from 'global' and 'alike' from 'dissimilar'. This is exactly what PageRank is meant to achieve.

We note that there are a large number of research papers on PageRank and search algorithms. Here we focus on several selected PageRank algorithms and the choices are far from comprehensive or complete. The reader is referred to several books and surveys on this topic [27, 28, 31, 32].

2 Defining PageRank

In 1998, Brin and Page [8] introduced the notion of PageRank for Google's Web search algorithm. Different from the usual methods in pattern matching previously used in data retrieval, the novelty of PageRank relies entirely on the underlying Webgraph to determine the 'importance' of a Webpage.

Although PageRank is originally designed for the Webgraph, the concept and definitions work well for any graph. Indeed, PageRank has become a valuable tool for examining the correlations of pairs of vertices (or pairs of subsets) in any given graph and hence leads to many applications in graph theory.

The starting point of the PageRank is a typical random walk on a graph G with edge weights w_{uv} for edges $\{u, v\}$. The probability transition matrix

P is defined by: $P(u,v) = \frac{w_{uv}}{d_u}$ where the degree of u is defined by $d_u = \sum_v w_{u,v}$. For an initial probability distribution f_0, the distribution after one step is exactly $f_0 P$ where f_0 is taken to be a row vector. In general, after k steps, the distribution at v is $f_0 P^k(v)$. In addition to the definitions, a table is included to summarize the notation in Table 1.

Symbol	Name
d_v	the degree of a vertex v
G	an undirected graph
$\tilde{G}$	a sparsifier of G
g	the Green value
$h(S)$	the Cheeger ratio of S
P	the transition probability matrix of a random walk
$pr_{\alpha,s}$	the PageRank vector with a jumping constant α and preference vector s
w_{uv}	the edge weight of u and v
W	the transition probability matrix for a lazy walk
$\mathrm{vol}(S)$	the volume of S
χ_v	the characteristic function of a vertex v

Table 1: Notations

We consider the lazy walk W defined by

$$W = \frac{I+P}{2}.$$

For a *preference vector* s, and a *jumping constant* α where $0 < \alpha < 1$, the PageRank, denoted by $pr_{\alpha,s}$ as a row vector, satisfies the following recurrence relation:

$$pr_{\alpha,s} = \alpha s + (1-\alpha) pr_{\alpha,s} W. \tag{1}$$

Equivalently, $pr_{\alpha,s}$ can be expressed as a series of random walks as follows:

$$pr_{\alpha,s} = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k s W^k. \tag{2}$$

The preference vector s can be a characteristic function χ_u for some vertex u, (i.e., $\chi_u(v) = 1$ if $u = v$ and 0 otherwise.) In general, s can be some chosen probability distribution. When $s = \chi_u$, we write $pr_{\alpha,\chi_u} = pr_{\alpha,u}$.

In the original definition of Brin and Page [8], s is taken to be the constant function with value $1/n$ at every vertex, motivated by modeling the behavior of a typical surfer who moves to a random page with probability α and clicks a linked page with probability $1-\alpha$. Let $pr_\alpha = pr_{\alpha,\vec{1}/n}$ denote the original PageRank vector where $\vec{1}$ denotes the all 1's function. It is easily checked that for every vertex v, we have

$$pr_\alpha(v) = \frac{1}{n}\sum_{u\in V} pr_{\alpha,u}(v). \tag{3}$$

The equation in (1) was introduced by Jeh and Widom [22], called "personalized page rank". A similar approach was used by Berkhin [6].

Remark 1 Kleinberg [25] proposed a different method for ranking webpages around the same time as the PageRank was introduced. The linking structure of the webpages is modeled by a directed graph. A directed edge from u to v can be used for conferring the 'authority' of v or asserting the importance of u as a 'hub'. For each webpage, denoted by a node v, let x_v denote the 'importance' as an'authority' and y_v denote the 'importance' as a'hub'. The importance of a webpage as an authority is proportional to the sum of the importances of webpages as hubs incident to v. Namely, we can write, for each vertex v,

$$x_v = \rho_1 \sum_{u\to v} y_u \text{ and } y_v = \rho_2 \sum_{w\leftarrow v} x_w.$$

This can be expressed by using vectors $\mathbf{x} = (x_v)$ and $\mathbf{y} = (y_v)$ as follows:

$$\mathbf{x} = \rho_1 A\mathbf{y} \text{ and } \mathbf{y} = \rho_2 A^T\mathbf{x}.$$

This implies

$$\mathbf{x} = \rho AA^T\mathbf{x} \text{ and } \mathbf{y} = \rho' A^T A\mathbf{y}.$$

Kleinberg's HITS algorithm (Hyperlink Induced Topic Search algorithm) involves finding the principal eigenvectors of AA^T and A^TA.

3 Approximate pagerank algorithms

3.1 The approximate pagerank algorithm *by iterations*

From the recurrence in (1), a straightforward iterated method follows, which was essentially the original PageRank algorithm used by Google as described in [26].

```
IteratedPR ($\alpha, s, \epsilon$):
    Let $p \leftarrow s$
loop:
    $p_{i+1} \leftarrow \alpha s + (1-\alpha) p_i W$
    $\delta \leftarrow \|p_{i+1} - p_i\|_1$
while $\delta > \epsilon$
return $p$
```

Some partial iterated algorithms for computing PageRank vectors were given in [20, 22] (also see [27]). For iterated algorithms, a main issue concerns termination and convergence. The 1999 paper [26] on Google's Web search algorithms contains extensive discussions on the implementation of the iterated algorithm for computing PageRank . It was stated, "*PageRank scales very well even for extremely large collections as the scaling factor is roughly linear in* $\log n$*.*"

The effectiveness of the iterated PageRank algorithms can be reasoned along the following lines:
(i) The assumption that Web graphs are random-like with expansion properties.
(ii) The fact that random walks on expander graphs are rapidly mixing.

The following justification was given in [26]: *"The fact that the PageRank computation terminates in logarithmic time is equivalent to saying that the random walk is rapidly mixing or that the underlying graph has a good expansion factor."*

3.2 The approximate pagerank algorithm *by pushes*

The following approximate PageRank algorithm is adapted from [2]. This algorithm maintains a pair of vectors p and r, starting with the trivial approximation $p = \vec{0}$ and $r = s$. A series of push operations move probability from r to p while maintaining the invariant $p = pr_{\alpha,s-r}$. Each push operation takes the probability from r at a single vertex u, moves an α fraction of this probability to $p(u)$, and then spreads the remain ing $(1-\alpha)$ fraction within r by applying a lazy random walk step to the vector $(1-\alpha)r(u)\chi_u$.

```
push $(\alpha, u)$:
Let $p' = p$ and $r' = r$, except for these changes:
    1. $p'(u) = p(u) + \alpha r(u)$.
    2. $r'(u) = (1-\alpha) r(u)/2$.
    3. For each vertex $v$ such that $(u, v) \in E$:
       $r'(v) = r(v) + (1-\alpha) r(u)/(2d_u)$.
```

Suppose p' and r' are the result of performing push(u) on p and r. Then it follows that

$$p = pr_{\alpha, s-r} \quad \Longrightarrow \quad p' = pr_{\alpha, s-r'}.$$

During each push operation, some probability is moved from r to p, where it remains. This algorithm performs pushes only on vertices where $r(u) \geq \epsilon d_u$, which ensures that a significant amount of probability is moved at each step, and allows us to bound the number of pushes required to compute an ϵ-approximate PageRank vector.

```
ApproximatePR $(s, \alpha, \epsilon)$:
    1. Let $p = \vec{0}$, and $r = s$.
    2. While $r(u) \geq \epsilon d_u$ for some vertex $u$:
        (a) Pick any vertex $u$ where $r(u) \geq \epsilon d_u$.
        (b) Apply push $(u)$.
    3. Return $p$ and $r$.
```

This algorithm can be implemented by maintaining a queue containing those vertices u satisfying $r(u) \geq \epsilon d_u$. At each step, a push operation is performed on the first vertex u in the queue. If $r(u)$ is still at least ϵd_u after the push is performed, then u is placed at the back of the queue; otherwise u is removed from the queue. If a push operation raises the value of $r(x)$ above ϵd_x for some neighbor x of u, then x is added to the back of the queue. This continues until the queue is empty, at which point all vertices satisfy $r(u) < \epsilon d_u$. The algorithm computes an ϵ-approximate PageRank vector p for $pr_{\alpha,s}$. The support of p satisfies $\text{vol}(\text{Supp}(p)) \leq \frac{2}{(1-\alpha)\epsilon}$, and the running time of the algorithm is $O(\frac{1}{\epsilon\alpha})$.

One way to validate the effectiveness of PageRank concerns identifying a local community from a given seed (or preference vector). A quantative measure for a subset S of vertices is the Cheeger ratio:

$$h(S) = \frac{|E(S, \bar{S})|}{\mathrm{vol}(S)}$$

where $E(S, \bar{S})$ denotes the set of edges leaving S and $\mathrm{vol}(S) = \sum_{v \in S} d_v$.

Suppose C is a subset with Cheeger ratio $h(C) = h$. It was shown in [2] that by choosing α to be $10h$, for at least half of the vertices u in C, the ϵ-approximate PageRank $pr_{\alpha,u}$ can be used to find a a subset S satisfying the following:
(i) $h(S) \le 2\sqrt{h \log(\mathrm{vol}(C))}$.
(ii) $\mathrm{vol}(S \cap C) \ge \mathrm{vol}(S)/2$.
(iii) S is determined by a sweep of the ϵ-approximate PageRank $p = pr_{\alpha,u}$. Namely, if we order the vertices so that $p(u_1) \ge p(u_2) \ge \ldots \ge p(u_k) \ge \ldots$ then $S = \{v : p(v) \ge p(u_j)$ for some $j \le \mathrm{vol}(C)/\epsilon\}$.
This gives a quantitative and rigorous validation of the power of PageRank.

3.3 A sharp approximate pagerank algorithm

For some applications, the error bound ϵ is required to be quite sharp. For example, in order to derive an effective ranking for edges in a graph (see [19]), ϵ is taken to be of order $O(n^{-2})$. For such small ϵ, instead of having the factor $1/\epsilon$ in the running time, it is desirable to only allow terms such as $\log(1/\epsilon)$.

The estimate error bound for the algorithm ApproximatePR can be further improved by the following iterated process:

```
SharpApproximatePR $(s, \alpha, \epsilon)$:

  1. Let $\epsilon' = 1$, $r = s$ and $p = \vec{0}$..

  2. 2. While $\epsilon' > \epsilon$ :

      (a) Set $\epsilon' \leftarrow \epsilon'/2$.
      (b) Let $p'$ and $r'$ be the output of ApproximatePR($r$, $\alpha$, $\epsilon'$).
      (c) Let $p \leftarrow p + p'$ and $r \leftarrow r'$.

  3. Return $p$ and $r$.
```

It was shown in [19] that the algorithm SharpApproximatePR(s, α, ϵ) computes approximate PageRank vector $p = pr_{\alpha, s-r}$ such that the residual vector r satisfies $|r(v)/d_v| \leq \epsilon$ for all vertices v. The running time of the algorithm is $O(\frac{m}{\alpha}\log(1/\epsilon))$, where m is the number of edges in the graph.

As an immediate consequence, by taking ϵ to be the inverse of a power of n, the algorithm SharpApproximatePR(s, α, ϵ) computes an approximate PageRank vector $p = pr_{\alpha,s-r}$ such that the residual vector r satisfies $|r(v)/d_v| \leq n^{-k}$ for all vertices v. The running time of the algorithm is $O(\frac{m}{\alpha}\log n)$.

There is a large literature [5, 23, 24, 33, 34] on graph sparsification. The goal of sparsification is to approximate a given graph G by a sparse graph $\tilde{G}$ on the same set of vertices while the sparse graph $\tilde{G}$ preserves the Cheeger ratios of every subset of vertices to within a factor of $1 + \epsilon$.

The main step in any sparsification algorithm is to choose an appropriate probability distribution for random sampling the edges in a way that Cheeger ratios of subsets change little. A sparsification algorithm in [19] is a sampling process using probabilities proportional to the PageRank for edges. The derivation of the PageRank for edges is quite similar to the *effective resistance* of edges in electrical network theory. In a way, the PageRank for edges can be viewed as the generalized effective resistance with an additional parameter α which can be controlled. The edge-PageRank also has a connection with Green's function [11] and we call the edge-PageRank the *Green values* g_α, defined on edges of the graphs as follows.

For each edge $e = \{u, v\} \in E$, we define the *Green value* $g_\alpha(u, v)$ of e to be a combination of four terms in PageRank vectors :

$$g_\alpha(u,v) \quad = \quad \frac{pr_{\alpha,u}(u)}{d_u} - \frac{pr_{\alpha,u}(v)}{d_v} + \frac{pr_{\alpha,v}(v)}{d_v} - \frac{pr_{\alpha,v}(u)}{d_u}. \qquad (4)$$

Since the Green values are relatively small (e.g., of order $1/n^c$, for some positive constant c), we need very sharp approximations, to be within a factor of $1 + n^{-c}$ of the exact values in the analysis of the performance bound for the graph sparsification algorithms in [19].

Let $\Delta = \max_{v \in V} d(v)$ denote the maximum degree. It was shown in [19] that for given any constant $\epsilon > 0$ and any pair $(u, v) \in V \times V$, one can compute the quantity $\tilde{g}_\alpha(u, v)$ in $O(\frac{\Delta}{\alpha^2\epsilon})$ time such that

$$|g_\alpha(u,v) - \tilde{g}_\alpha(u,v)| \leq \epsilon g_\alpha(u,v).$$

In particular, after $O(\frac{\Delta n}{\alpha^2\epsilon})$ preprocessing time, for each edge $\{u, v\}$, one can compute such $\tilde{g}_\alpha(u, v)$ by using a constant number of queries.

The graph sparsification algorithm [19] is just to do q rounds of sampling of edges e with probability p_e proportional to $w(e)\tilde{g}_\alpha(e)$ where $w(e)$ denotes the weight of e and α, q are appropriately chosen (e. g., $\alpha = 1/10$ and $q = c(n \log n)/\epsilon^2$) for some absolute constant c. In each round, we add e to $\tilde{G}$ with weight $w(e)/(qp_e)$ and sum the weights if an edge is chosen more than once. It can then be shown [19] that the graph $\tilde{G}$ with $cn \log n/\epsilon^2$ edges is a sparsifier in the sense that for all vertex subset S, the Cheeger ratio of S is preserved within the following error bound:

$$|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S).$$

3.4 Approximate pagerank algorithm *by random walks*

The PageRank algorithm given in [7] is mainly based on the following observation:
$pr_{\alpha,u}(v)$ is equal to the success probability that a random walk starting at u and independently terminating at each time step with probability α, hits v just before termination.

In [7], the notion of an approximate PageRank is further weakened in a *probabilistic sparse-and-approximate row access model with additive and multiplicative errors*. The input of the PageRank approximate algorithm, called ApproximatePRaccess, has inputs including a specified vertex u, an additive error bound ϵ, a multiplicative error bound δ and a success probability η. The output p' of the algorithm is an approximation for $p = pr_{\alpha,u}$ in the following sense:

1. With probability η, for every vertex v,

$$(1-\delta)p(v) - \epsilon \leq p'(v) \leq (1+\delta)p(v) + \epsilon.$$

2. With probability $1 - \eta$, p' can be any sparse vector.

```
ApproximatePRaccess $(u, \epsilon, \delta, \eta)$:

 1. Set $t = \lceil \log_{\frac{1}{(1-\alpha)}} \left(\frac{4}{\epsilon}\right) \rceil$.
 2. Set $r = \lceil \frac{1}{\epsilon\delta^2} \cdot 4\ln(n/\eta) \rceil$.
 3. for $r$ rounds do
 4.     Run one realization of a restarting random walk from $u$.
          Stop the walk after $t$ steps if it has not terminated already.
 5.     if the walk visited a node $v$ just before a termination step
        then $p'(v) \leftarrow p'(v) + 1/r$ .
 6.     end if
 7. end for.
 8. Return $p'$.
```

It was shown [7] that the running time for the algorithm ApproximatePRaccess can be bounded above by $O(\frac{\ln^2(n)\ln(1/\eta)\ln(\epsilon^{-1})}{\epsilon\delta^2})$. The approximate Pagerank is then used to address the *Significant PageRank problem.*
Significant PageRank problem
For a given graph (or directed graph) $G = (V, E)$, a threshold value $1/n \leq \Delta \leq 1$ and a positive constant $c > 1$, compute a subset $S \subseteq V$ with the property that S contains all vertices v of PageRank $pr_\alpha(v) \geq \Delta/n$ where pr_α is as defined in (3). By using the algorithm ApproximatePRaccess as a subroutine, the significant PageRank problem was solved [7] with cost $\tilde{O}(1/\Delta)$.

For empirical simulations, the reader is referred to an extensive survey by Leskovec et al [31] in which PageRank algorithms are favorable in comparison with a number of partition algorithms for community detection using examples from 40 different networks.

4 Applications and generalization of PageRank

Because of the fundamental nature of PageRank for quantifying the relationship among vertices and subsets of nodes, there are numerous applications using PageRank vectors in a wide range of areas. In addition to local partitioning, graph sparsification and identifying significant nodes, many applications of PageRank including clustering/visualization [17], containing epidemic deceases [15], multi-commodity allocation [14], trust-based ranking [1, 18] , and gene identification through metabolic and in protein interaction network databases [21, 29], to name a few.

There are several variations and extensions of PageRank, which can be used to deal with information networks arising in a variety of scenarios. For example, the *Kronecker PageRank* can be used to treat networks with multiple attributes [14, 30], the *connection PageRank* is useful for geometrical graphs in high dimensions and the *heat kernel PageRank* which is expressed as an exponential sum of random walks, leads to improved local partitioning algorithms [12, 13, 16]. Since many real-world information networks are directed graphs, a modified PageRank algorithm for directed graphs can be found in [4]. Numerous problems dealing with information networks can possibly take advantage of PageRank and its variations, and the full implications of these ideas remain to be explored.

References

[1] R. Andersen, C. Borg, J. Chayes, U. Feige, A. flaxman, A. Kalai, V. Mirrokni and M. Tennenholtz, Trust-based recommendation system: an axiomatic approach, *WWW2008*, *Proccedings of the 17th International Conference on World Wide Web* (2008), 199-208.

[2] R. Andersen, F. Chung and K. Lang, Local graph partitioning using pagerank vectors, *Proceedings of the 47th Annual IEEE Symposium on Founation of Computer Science (FOCS 2006)*, 475–486.

[3] R. Andersen, F. Chung and K. Lang, Detecting sharp drops in PageRank and a simplified local partitioning algorithm, *Theory and Applications of Models of Computation, Proceedings of TAMC 2007*, 1–12.

[4] R. Andersen, F. Chung and K. Lang, local partitioning for directed graphs using PageRank, *Internet Math.*, **5** (2008), 3–22.

[5] A. A. Benczúr and D. R. Karger, Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time, *STOC 96*, 47–55.

[6] P. Berkhin, Bookmark-coloring approach to personalized pagerank computing, *Internet Math.*, **3**, (2006), 41–62.

[7] C. Borgs, M. Brautbar, J. Chayes and S.-H. Teng, Multi-scale matrix sampling and subliinear-time Pagerank computation, *WAW* (2012), 41–53.

[8] S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine, *Computer Networks and ISDN Systems*, **30 (1-7)**, (1998), 107–117.

[9] F. Chung, *Spectral Graph Theory*, AMS Publications, 1997.

[10] F. Chung, A whirlwind tour of random graphs, *Encyclopedia of Complex Systems*, Springer, 2008.

[11] F. Chung, PageRank as a discrete Green's function, *Geometry and Analysis, I, ALM* **17**, (2010), 285–302.

[12] F. Chung, The heat kernel as the pagerank of a graph, *Proc. Nat. Acad. Sciences*, **105** (50), (2007), 19735–19740.

[13] F. Chung, A local graph partitioning algorithm using heat kernel PagePank, *WAW 2009*, *Lecture Notes in Computer Science*, vol. 5427, Springer, (2009), 62–75.

[14] F. Chung, J. Hughes and P. Horn, Multi-commodity allocation for dynamic demands using PageRank vectors , *WAW2012, LNCS 7323* (2012), 138–152.

[15] F. Chung, P. Horn and A. Tsiatas, Distributing antidote using PageRank vectors , *Internet Mathematics*, **6**, (2009), 237–254.

[16] F. Chung and O. Simpson, Solving linear systems with boundary conditions using heat kernel pagerank , *WAW 2013*, 203–219.

[17] F. Chung and A. Tsiatas, Finding and visualizing graph clusters using PageRank optimization, *WAW 2010*, *Lecture Notes in Computer Science*, vol. 6516, Springer, (2010), 86–97.

[18] F. Chung and A. Tsiatas, Dirichlet PageRank and trust-based ranking algorithms, *WAW 2011*, *LNCS 6732*, (2011), 103–114.

[19] F. Chung and W. Zhao, A sharp PageRank algorithm with applications to edge ranking and graph sparsification, *Proceedings of Workshop on Algorithms and Models for the Web Graph (WAW)*, (2010), 2–14.

[20] H. Haveliwala, Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search, *IEEE Trans. Knowl. Data Eng.*, **15(4)**, (2003), 784–796.

[21] G. Iván and V. Grolmusz, When the web meets the cell: using personalized PageRank for analyzing protein interaction networks, *Bioinformatics*, **27**, no. 3 (2011), 405–407.j

[22] G. Jeh and J. Widom, Scaling personalized web search, *Proceedings of the 12th World Wide Web Conference WWW*, (2003), 271–279.

[23] David R. Karger. Using randomized sparsification to approximate minimum cuts, *SODA 94*, 424–432.

[24] David R. Karger. Minimum cuts in near-linear time, *JACM*, 47 (2000), 46–76..

[25] J. Kleinberg, Authoritative sources in a hyperlinked environment, *JACM*, **46**, (1999), 604–632.

[26] L. Page, S. Brin, R. Motwani and T. Winograd, The PageRank citation ranking: Bringing order to the Web, Technical report, Stanford InfoLab (1999).

[27] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings*, Princeton University Press, (2011).

[28] A. N. Langville and C. D. Meyer, Deeper inside PageRank, *Internet Math.* **1** (2004), 335–380.

[29] I. Lee, U. M. Blom, P. I. Wang, J. E. Shim and E. M. Marcotte, Prioritizing candidate disease genes by network-based boosting of genome-wide association data, *Genome Res.* **21** (2011), 1109–1121

[30] J. Leskovec, C. Faloutsos, Scalable modeling of real graphs using Kronecker multiplication, *International Conference on Machine Learning (ICML)* (2007), 497–504.

[31] J. Leskovec, K. J. Lang and M. Mahoney, Empirical comparison of algorithms for network community detection, *Proceedings of the 19th International Conference on World Wide Web* 2010, 631–640.

[32] B. Liu and P. Yu, PageRank, a chapter in *The Top Ten Algorithms in Data Mining*, (X. Wu and V. Kumar eds.) CRC Press, Taylor & Francis Group, Boca Raton, FL, 2008.

[33] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear

systems. Available at http://www.arxiv.org/abs/cs.NA/0607105, 2006.

[34] D. A. Spielman and N. Srivastava, Graph sparsification by effective resistances. *STOC 2008*, 563–568.