Fast Katz and Commuters: Efficient Estimation of Social Relatedness in Large Networks

Pooya Esfandiar[1], Francesco Bonchi[2], David F. Gleich[3],
Chen Greif[1], Laks V.S. Lakshmanan[1], and Byung-Won On[1]

[1] University of British Columbia, Vancouver BC, Canada
{pooyae,greif,laks,bwon}@cs.ubc.ca
[2] Yahoo! Research, Barcelona, Spain
bonchi@yahoo-inc.com
[3] Sandia National Laboratories*, Livermore CA, USA
dfgleic@sandia.gov

Abstract. Motivated by social network data mining problems such as link prediction and collaborative filtering, significant research effort has been devoted to computing topological measures including the Katz score and the commute time. Existing approaches typically approximate all pairwise relationships simultaneously. In this paper, we are interested in computing: the score for a single pair of nodes, and the top-k nodes with the best scores from a given source node. For the pairwise problem, we apply an iterative algorithm that computes upper and lower bounds for the measures we seek. This algorithm exploits a relationship between the Lanczos process and a quadrature rule. For the top-k problem, we propose an algorithm that only accesses a small portion of the graph and is related to techniques used in personalized PageRank computing. To test the scalability and accuracy of our algorithms we experiment with three real-world networks and find that these algorithms run in milliseconds to seconds without any preprocessing.

1 Introduction

The availability of large social networks and social interaction data (on movies, books, music, etc) have caused people to ask: what can we learn by mining this wealth of data? Measures of social relatedness play a fundamental role in answering this question. For example, Liben-Nowell and Kleinberg [13] identify a variety of topological measures as features for *link prediction*, the problem of predicting the likelihood of users/entities forming social ties in the future, given the current state of the network. The measures they studied fall into two categories – neighborhood-based measures and path-based measures. The former are cheaper to compute, yet the latter are more effective at link prediction. Katz

* Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

R. Kumar and D. Sivakumar (Eds.): WAW 2010, LNCS 6516, pp. 132–145, 2010.

scores [11] were among the best link predictors, and the commute time [6] also performed well. Other uses of Katz scores and commute time are anomalous link detection [18], recommendation [20], and clustering [19].

Katz scores measure the affinity between nodes via a weighted sum of the number of paths between them. Formally, the Katz score between node i and j is $K_{i,j} = \sum_{\ell=1}^{\infty} \alpha^\ell \text{paths}_\ell(x,y)$, where $\text{paths}_\ell(x,y)$ denotes the number of paths of length ℓ between i to j and $\alpha < 1$ is an attenuation parameter. Let A be the symmetric adjacency matrix, and recall that $(A^\ell)_{i,j}$ is the number of paths between node i and j. Then for all pairs of nodes,

$$K = \alpha A + \alpha^2 A^2 + \cdots = (I - \alpha A)^{-1} - I,$$

where the series converges if $\alpha < 1/\|A\|_2$.

The hitting time from node i to j is the expected number of steps for a random walk started at i to visit j, and the commute time between nodes is defined as the sum of hitting times from i to j and from j to i. The hitting time may be expressed using the row-stochastic transition matrix P with first-transition analysis: $H_{i,i} = 0$ and $H_{i,j} = 1 + \sum_k P_{i,k} H_{\ell,j}$. Unlike Katz, hitting time is not symmetric; but commute time is by definition, since $C = H + H^T$. Computing H and C via these definitions is not straightforward, and using the graph Laplacian, $L = D - A$ where D is the diagonal matrix of degrees, provides another means of computing the commute time. With the Laplacian, $C_{i,j} = \text{Vol}(G)(L^\dagger_{i,i} - 2L^\dagger_{i,j} + L^\dagger_{j,j})$ where $\text{Vol}(G)$ is the sum of elements in A and $L^\dagger$ is the pseudo-inverse of L [5].

Computing both of these measures between all pairs of nodes involves inverting a matrix, i.e. $(I - \alpha A)^{-1}$ or $L^\dagger$. Standard algorithms for a matrix inverse require $O(n^3)$ time and $O(n^2)$ memory and are inappropriate for a large network (see Section 2 for a brief survey of existing alternatives). Inspired by applications in anomalous link detection and recommendation [18,20], we focus on computing only a single Katz score or commute time and on computing the k most related nodes by Katz score.

In Section 3, we propose customized methods for the pairwise problems based on the Lanczos/Stieltjes procedure [8]. We specialize it for the Katz and commute time measures, providing a novel and useful application for the Lanczos/Stieltjes procedure. In Section 4, we present an algorithm to approximate the strongest ties between a given source node and its neighbors in terms of the Katz score (while we discuss the case of commute time in the conclusion section). This algorithm are inspired by a technique for personalized PageRank computing [14,2,3], though heavily adapted to the Katz score. We evaluate these methods on three real-world networks and report the results in Section 5. Our methods produce answers in seconds or milliseconds, whereas preprocessing techniques may often take over 10 minutes.

We have made our codes and data available for others to reproduce our results: `http://stanford.edu/~dgleich/publications/2010/codes/fast-katz/`.

2 Related Work

Most existing techniques to compute the Katz score and commute time determine the scores among all pairs of nodes simultaneously [1,24,20]. These methods tend to involve some preprocessing of the graph and a single, rather expensive, computation. In this paper instead we focus on quick estimates of these measures between a single pair of nodes and between a single node to all other nodes in the graph.

Standard techniques to approximate Katz scores include truncating the series expansion to paths of length less than $\ell_{\max}$ [4,24] and low-rank approximation [13,1]. (Note that computing these Katz scores between nodes is quite different from computing Katz's status index.) In general, these techniques for *all* the scores require more time and memory than our approach, and we do not compare against them for this reason.

Sarkar and Moore [20] proposed an interesting and efficient approach for finding approximate nearest neighbors with respect to a truncated version of the commute time measure. In [21], Sarkar et al. use their truncated commute time measure for link prediction over a collaboration graph and show that it outperforms personalized PageRank [15]. Spielman and Srivastava [22] show how to approximate the effective resistance of all edges (which is proportional to commute time) in $O(m \log n)$ time for a graph with m edges and n nodes. These procedures all involve some preprocessing.

Recently Li et. al. studied pairwise approximations of SimRank scores [12].

3 Algorithms for Pairwise Score

Consider the Katz score and commute time between a single pair of nodes: $K_{i,j} = e_i^T(I-\alpha A)^{-1}e_j - \delta_{i,j}$ and $C_{i,j} = \mathrm{Vol}(G)(e_i-e_j)^T L^\dagger (e_i-e_j)$. In these expressions, e_i and e_j are vectors of zeros with a 1 in the ith and jth position, respectively; and $\delta_{i,j}$ is the Kronecker delta function. A straightforward means of computing them is to solve the linear system $(I-\alpha A)x = e_j$ and $(L + \frac{1}{n}ee^T)y = e_i - e_j$. Then $K_{i,j} = e_i^T x - \delta_{i,j}$ and $C_{i,j} = \mathrm{Vol}(G)(e_i - e_j)^T y$. This form of commute time follow after substituting $L^\dagger = (L + \frac{1}{n}ee^T)^{-1} - \frac{1}{n}ee^T$ (see [19]). Solving these linear systems is an effective method to compute only the pairwise scores. In what follows, we show how a technique combining the Lanczos iteration and a quadrature rule produces the pairwise Katz score and commute time score, *as well as* upper and lower bounds on the estimate. Our technique is based on the methodology developed in [8,9], which we describe below.

Note that for $\alpha < 1/||A||_2$, $(I - \alpha A)$ is symmetric positive definite, as is $(L + \frac{1}{n}ee^T)$. Thus, the pairwise Katz score and the commute time score are related to the problem of computing the bilinear form $u^T f(E)v$ where E is a symmetric positive definite matrix. In the most general setting, u and v are given vectors and f is an analytic function on the interval containing the eigenvalues of E. In the application to Katz scores and commute time, $f(E) = E^{-1}$. Note we need only consider $u = v$ because

$$u^T f(E)v = \tfrac{1}{4}\Big[(u+v)^T f(E)(u+v) - (u-v)^T f(E)(u-v)\Big]. \tag{1}$$

Golub and Meurant [8,9] introduced techniques for evaluating such bilinear forms. They provided a solid mathematical framework and a rich collection of possible applications. These techniques are well known in the numerical linear algebra community, but they do not seem to have been used in data mining problems. We utilize this methodology to compute pairwise scores, which extends to a large-scale setting. The algorithm has two main components: Gauss-type quadrature rules for evaluating definite integrals, and the Lanczos algorithm for partial reduction to symmetric tridiagonal form.

Because E is symmetric positive definite, it has a unitary spectral decomposition, $E = Q\Lambda Q^T$, where Q is an orthogonal matrix whose columns are eigenvectors of E with unity 2-norm, and Λ is a diagonal matrix with the eigenvalues of E along its diagonal. We use this decomposition only for the derivation that follows – it is never explicitly computed in our algorithm. Given this decomposition, for any analytic function f,

$$u^T f(E)u = u^T Q f(\Lambda) Q^T u = \sum_{i=1}^{n} f(\lambda_i)\tilde{u}_i^T \tilde{u}_i,$$

where $\tilde{u} = Q^T u$. The last sum can be thought of as a quadrature rule for computing the Stieltjes integral

$$u^T f(E)u = \int_a^b f(\lambda) d\gamma(\lambda). \tag{2}$$

Here γ is a piecewise constant measure, which is monotonically increasing, and its values depend directly on the eigenvalues of E; λ denotes the set of all eigenvalues; γ is a discontinuous step function, each of whose pieces is a constant function. Specifically, $\gamma(\lambda)$ is identically zero if $\lambda < \min_i \lambda_i(E)$, is equal to $\sum_{j=1}^{i} \tilde{u}_j^2$ if $\lambda_i \leq \lambda < \lambda_{i+1}$, and is equal to $\sum_{j=1}^{n} \tilde{u}_j^2$ if $\lambda \geq \max_i \lambda_i(E)$.

The first of Golub and Meurant's key insights is that we can compute an approximation for an integral of the form (2) using a quadrature rule. The second insight is that the Lanczos procedure constructs a tridiagonal matrix whose eigenvalues are the quadrature nodes for the specific measure γ, and $u = e_i$. Since we use a quadrature rule, an estimate of the error is readily available. More importantly, we can use variants of the Gaussian integration formula to obtain both lower and upper bounds and "trap" the value of the element of the inverse that we seek between these bounds. The ability to estimate bounds for the value is powerful and provides effective stopping criteria for the algorithm. It is important to note that such component-wise bounds cannot be easily obtained if we were to extract the value of the element from a column of the inverse, by solving the corresponding linear system. Indeed, typically for the solution of a linear system, norm-wise bounds are available, but obtaining bounds pertaining to the components of the solution is significantly more challenging and results of this sort are harder to establish.

Algorithm 1 reproduces a consice procedure from [9] to estimate $u^T E^{-1} u$. The input is a matrix E, a vector u, estimates of extremal eigenvalues of E,

Algorithm 1. Computing Score Bounds

Input: $E, u, a < \lambda_{\min(E)}, b > \lambda_{\max(E)}, k$
Output: $\underline{b_k} \leq u^T E^{-1} u \leq \overline{b_k}$

1: **Initial step:** $h_1 = 0$, $h_0 = u$, $\omega_1 = u^T E u$, $\gamma_1 = ||(E - \omega_1 I)u||$, $b_1 = \omega_1^{-1}$, $d_1 = \omega_1$, $c_1 = 1$, $\overline{d_1} = \omega_1 - a$, $\underline{d_1} = \omega_1 - b$, $h_1 = \frac{(E-\omega_1 I)u}{\gamma_1}$.
2: **for** $j = 2, ...k$ **do**
3: $\quad \omega_j = h_{j-1}^T E h_{j-1}$
4: $\quad \widetilde{h_j} = (E - \omega_j I)h_{j-1} - \gamma_{j-1} h_{j-2}$
5: $\quad \gamma_j = \|\widetilde{h_j}\|$
6: $\quad h_j = \frac{\widetilde{h_j}}{\gamma_j}$
7: $\quad b_j = b_{j-1} + \frac{\gamma_{j-1}^2 c_{j-1}^2}{d_{j-1}(\omega_j d_{j-1} - \gamma_{j-1}^2)}$
8: $\quad d_j = \omega_j - \frac{\gamma_{j-1}^2}{d_{j-1}}$; $c_j = c_{j-1}\frac{\gamma_{j-1}}{d_{j-1}}$
9: $\quad \overline{d_j} = \omega_j - a - \frac{\gamma_{j-1}^2}{\overline{d_{j-1}}}$; $\underline{d_j} = \omega_j - b - \frac{\gamma_{j-1}^2}{\underline{d_{j-1}}}$
10: $\quad \overline{\omega_j} = a + \frac{\gamma_j^2}{\overline{d_j}}$; $\underline{\omega_j} = b + \frac{\gamma_j^2}{\underline{d_j}}$
11: $\quad \overline{b_j} = b_j + \frac{\gamma_j^2 c_j^2}{d_j(\overline{\omega_j} d_j - \gamma_j^2)}$; $\overline{b_j} = b_j + \frac{\gamma_j^2 c_j^2}{d_j(\underline{\omega_j} d_j - \gamma_j^2)}$

a and b, and a number of iterations k. In practice we can use the infinity norm of the original matrix as an estimate for b; this quantity is trivial to compute. The value of a is known to be small and positive, and in our experiments we have it set to 10^{-4}. (We note here that dynamically varying alternatives exist but these were not necessary in our experiments.) The algorithm computes $\underline{b_j}$ and $\overline{b_j}$, lower and bounds for $u^T E^{-1} u$. The core of the algorithm are steps 3–6, which are nothing but the Lanczos algorithm. In line 7 we apply the summation for the quadrature formula. The computation needs to be done for the upper bound as well as the lower bound; see lines 9 and 10. Line 11 computes the required bounds that "trap" the required quadratic form from above and below. For Katz we set $E = (I - \alpha A)$ and use (1) to get $e_i^T E^{-1} e_j$ by running the procedure twice and transposing the upper and lower bounds due to the subtraction. For commute time we approximate $(e_i - e_j)^T (L + (1/n)ee^T)^{-1}(e_i - e_j)$.

4 Top-k Algorithms

In this section, we show how to adapt techniques for rapid personalized PageRank computation [14,2,3] to the problem of computing the top-k largest Katz scores. These algorithms exploit the graph structure by accessing the edges of individual vertices, instead of accessing the graph via a matrix-vector product. They are "local" because they only access the outlinks of a small set of vertices and need not explore the majority of the graph. See the conclusions for a discussion of commute time and why we cannot utilize this procedure for that measure.

The basis of the algorithm is a variant on the Richardson stationary method for solving a linear system [23]. Given a linear system $Ax = b$, the Richardson iteration

is $x^{(k+1)} = x^{(k)} + \omega r^{(k)}$, where $r^{(k)} = b - Ax^{(k)}$ is the residual vector at the kth iteration and ω is an acceleration parameter. While updating $x^{(k+1)}$ is a linear time operation, computing the next residual requires another matrix-vector product. To take advantage of the graph structure, the personalized PageRank algorithms [14,2,3] propose the following change: do not update $x^{(k+1)}$ with the entire residual, and instead change only a single component of x. Formally, $x^{(k+1)} = x^{(k)} + \omega r_j^{(k)} e_j$, where $r_j^{(k)}$ is the jth component of the residual vector. Now, computing the next residual involves accessing a single column of the matrix A:

$$r^{(k+1)} = b - Ax^{(k+1)} = b - A(x^{(k)} + \omega r_j^{(k)} e_j) = r^{(k)} + \omega r_j^{(k)} Ae_j.$$

Suppose that r, x, and Ae_j are sparse, then this update introduces only a small number of new nonzeros into both x and the new residual r. Each column of A is sparse for most graphs, and thus keeping the solution and residual sparse is a natural goal for graph algorithms where the solution x is localized (i.e., many components of x can be rounded to 0 without dramatically changing the solution). By choosing the element j based on the largest entry in the sparse residual vector (maintained in a heap), this algorithm often finds a good approximation to the largest entries of the solution vector x while accessing only a small subset of the graph. Dropping the heap as in [2] yielded slightly worse localization and thus we did not use it in these experiments.

For a particular node i in the graph, the Katz scores to the other nodes are given by $k_i = [(I - \alpha A)^{-1} - I]e_i$. Let $(I - \alpha A)x = e_i$. Then $k_i = x - e_i$. We use the above process with $\omega = 1$ to compute x. For this system, x and r are always positive, and the residual converges to 0 geometrically if $\alpha < 1/\|E\|_1$. We observe convergence empirically for $1/\|E\|_1 < \alpha < 1/||E||_2$ and have a developed some theory to justify this result, but do not have space to present it here. To terminate our algorithm, we wait until the largest element in the residual is smaller than a specified tolerance, for example 10^{-4}.

5 Empirical Evaluation

Our experimental goals are: (i) to test the convergence speed; (ii) to measure the accuracy and scalability of our algorithms; and (iii) to compare our algorithms against the conjugate gradient (CG) method. Recall our setting: we only want a single score or top-k set. We use the CG iterative method as a reference point for our pairwise and top-k algorithms because it provides solutions in the large scale case without any preprocessing, just like our algorithms. As we previously mentioned, approaches based on preprocessing or simultaneously computing all the scores take considerably longer but provide more information. In the case of finding a small set of pairwise values, we leave finding the trade-off between our fast pairwise algorithms and the all-at-once approaches to future work.

Experiment settings. We implemented our methods in MATLAB and MATLAB mex codes. All computations and timings were done in Linux on a laptop with a Core2Duo T7200 processor (2 core, 2GHz) with 2GB of memory. We used

Table 1. Basic statistics about our datasets: number of nodes and edges, average degree, max singular value ($||A||_2$) and size of the 2-core in vertices

Graph	Nodes	Edges	Avg Degree	$\|\|A\|\|_2$	2-core Size
dblp	93,156	178,145	3.82	39.5753	76,578
arxiv	86,376	517,563	11.98	99.3319	45,342
flickr	513,969	3,190,452	12.41	663.3587	233,395

three real-world networks for our experiments: two citation-based networks based on publications databases, and one social network. The dataset[1] statistics are reported in Table 1.

Pairwise results. We begin by studying the accuracy of the pairwise algorithms for Katz scores and commute times. For this task, we first compute a highly accurate answer using the MINRES method [7] to solve the corresponding linear systems: $(I-\alpha A)x = e_i$ for Katz and $(L+\frac{1}{n}ee^T)x = (e_i - e_j)$ for commute time. We used a tolerance of 10^{-8} in these solutions. Next, we run our pairwise method. Recall that using Algorithm 1 requires a lower-bound on the smallest eigenvalue of the matrix E. We use 10^{-4} for this bound. We terminate our algorithms when the relative change in the upper and lower bounds is smaller than 10^{-4} or the upper and lower bounds cross each other. We evaluate the accuracy at each iteration of Algorithm 1. Because our approach to compute Katz scores requires two applications of Algorithm 1, the work at each iteration takes two matrix-vector products. As described in previous sections, our pairwise algorithm is closely related to iterative methods for linear systems, but with the added benefit of providing lower and upper bounds. As such, its convergence closely tracks that of the conjugate gradient method, a standard iterative method. We terminate conjugate gradient when the norm of the residual is smaller than 10^{-4}.

For convergence of the Katz scores, we use a value of α that makes $B = I - \alpha A$ nearly indefinite. Such a value produces the slowest convergence in our experience. The particular value we use is $\alpha = 1/(\|A\|_2 + 1)$. For a single pair of nodes in arxiv, we show how the upper and lower bounds "trap" the pairwise Katz scores in Figure 1 (top left). At iteration 13, the lower bound approaches the upper bound. Beyond this point the algorithm converges quickly. Similar convergence results are produced for the other two graphs. We show the convergence of both bounds to the exact solution in the bottom row. Both the lower and upper bounds converge similarly.

In comparison with the conjugate gradient method, our pairwise algorithm takes more matrix-vector products to converge. This happens because we must perform two applications of Algorithm 1. However, the conjugate gradient method does not provide upper and lower bounds on the element of the inverse, which our techniques do. The forthcoming experiments with commute time illustrate a case where it is difficult to terminate conjugate gradient early because of erratic convergence. For these problems, we also evaluated techniques

[1] In the interest of space we provide processing details of the datasets in our web page: http://stanford.edu/~dgleich/publications/2010/codes/fast-katz/

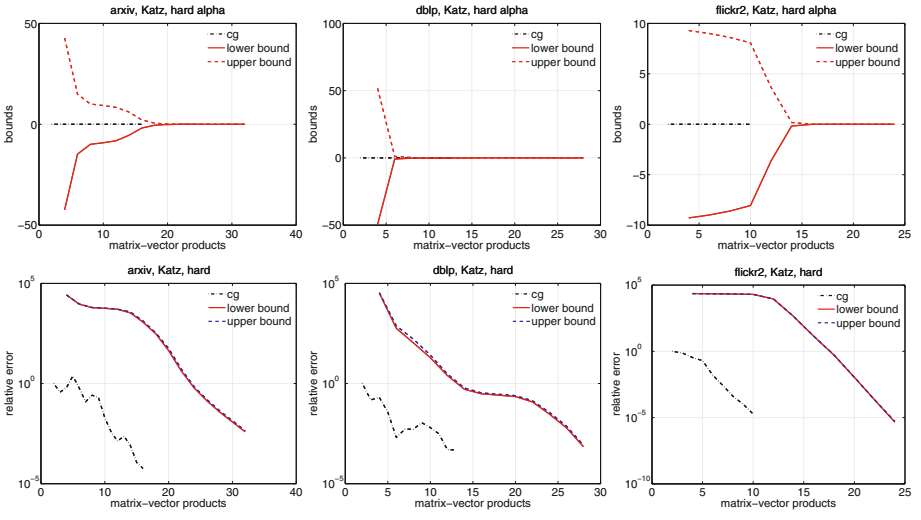

Fig. 1. Upper and lower bounds *(top)* and approximation error *(bottom)* for pairwise Katz on `arxiv` *(left)*, `dblp` *(center)*, and `flickr` *(right)*

based on the Neumann series for $I - \alpha A$, but those took over 100 times as many iterations as conjugate gradient or our pairwise approach. The Neumann series is the same algorithm used in [24] but customized for the linear system, not the matrix inverse, which is a more appropriate comparison for the pairwise case.

In Figure 2, we show how commute time converges for the same pairs of nodes. Again, the top row shows the convergence of the upper and lower bounds, and the bottom row shows the convergence of the error. While Katz took only a few iterations, computing pairwise commute times requires a few hundred iterations. A notable result is that the lower-bound from the quadrature rule provides a more accurate estimate of commute time than does the upper bound. See the curve of the lower bound in bottom row of Figure 2. This observation suggests that using the lower bound as an approximate solution is probably better for commute time.

Note that the relative error in the lower-bound produced by our algorithm is almost identical to the relative error from CG. This behavior is expected in cases where the largest eigenvalue of the matrix is well-separated from the remaining eigenvalues – a fact that holds for the Laplacians of our graphs. When this happens, the Lanczos procedure underlying both our technique and CG quickly produces an accurate estimate of the true largest eigenvalue, which in turn corrects the effect of our initial overestimate of the largest eigenvalue. (Recall from Algorithm 1 that the estimate of b is present in the computation of the lower-bound $\underline{b_j}$.)

Here, the conjugate gradient method suffers two problems. First, because CG does not provide bounds on the score, it is not possible to terminate it until the residual is small. Thus, the conjugate gradient method requires about twice as many iterations as our pairwise algorithms. Note, however, this result is simply a

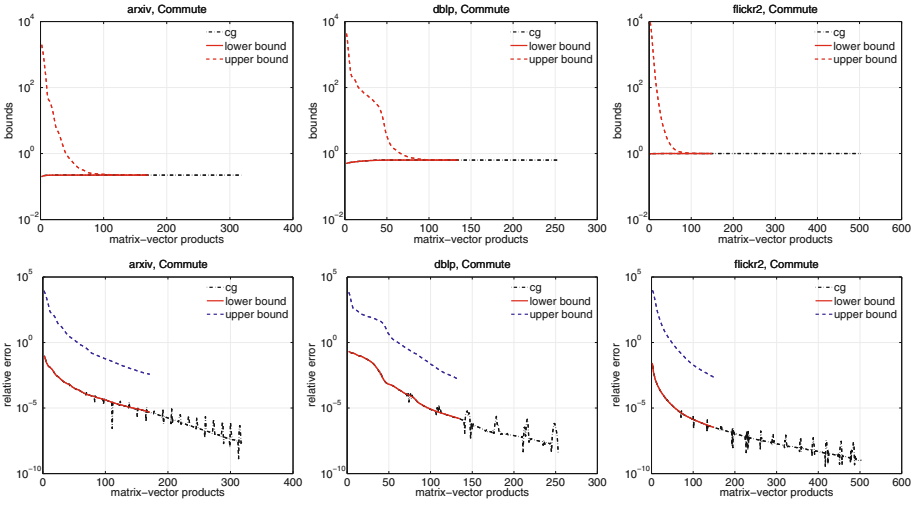

Fig. 2. Upper and lower bounds *(top)* and approximation error *(bottom)* for pairwise commute time scores on `arxiv` *(left)*, `dblp` *(center)*, and `flickr` *(right)*

matter of detecting when to stop – both conjugate gradient and our lower-bound produce similar relative errors for the same work. Second, the relative error for conjugate gradient displays erratic behavior. Such behavior is not unexpected, because conjugate gradient optimizes the A-norm of the solution error and it is not guaranteed to provide smooth convergence in the norm of the residual. These oscillations make early termination of the CG algorithm problematic, whereas no such issues occur for the upper and lower bounds from our pairwise algorithms.

Top-k results. We now proceed to a similar investigation of the top-k algorithms for Katz scores. In this section, we are concerned with the convergence of the set of top-k results. Thus, we evaluate each algorithm in terms of the precision between the top-k results generated by our algorithms and the exact top-k set produced by solving the linear system. Natural alternatives are other iterative methods and specialized direct methods that exploit sparsity. The latter – including approaches such as truncated commute time [20] – are beyond the scope of this work, since they require a different computational treatment in terms of caching and parallelization. Thus, we again use conjugate gradient (CG) as an example of iterative methods.

Let T_k^{alg} be the top-k set from our algorithm and T_k^* be the exact top-k set. The precision at k is $|T_k^{\text{alg}} \cap T_k^*|/k$, where $|\cdot|$ denotes cardinality. We also look at the Kendall-τ correlation coefficient between our algorithm's results and the exact top-k set. This experiment will let us evaluate whether the algorithm is ordering the true set of top-k results correctly. Let $x_{k^*}^{\text{alg}}$ be the scores from our algorithm on the exact top-k set, and let $x_{k^*}^*$ be the true top-k scores. The τ coefficients are computed between $x_{k^*}^{\text{alg}}$ and $x_{k^*}^*$. Both of these measures should tend to 1 as we increase the work in our algorithms. However, some of the exact

top-k results contain tied values. Our algorithm has trouble capturing precisely tied values and the effect is that our Kendall-τ score does not always tend to 1 exactly.

To compare with the pairwise results, we present the algorithm performance in *effective matrix-vector products*. An effective matrix-vector product corresponds to our algorithm examining the same number of edges as a matrix-vector product. In other words, suppose the algorithm accesses a total of 80 neighbors in a graph with 16 edges. Then this instance corresponds to $(80/16)/2 = 2.5$ effective matrix vector products.

For our first set of tests, we let the algorithm run for a prescribed number of steps and evaluate the results at the end. In Figure 3, we plot the convergence of the top-k set for $k = 10, 25, 100$, and 1000 for a single node. The top figures plot the precision at k, and the bottom figures plot the Kendall-τ correlation with the exact top-k set. Both of these measures trend to 1 quickly. In fact, the top-25 set is nearly converged after the equivalent of a single matrix-vector product – equivalent to just one iteration of the CG algorithm. We show results from the conjugate gradient method for the top-25 set after $2, 5, 10, 15, 25$, and 50 matrix-vector products.

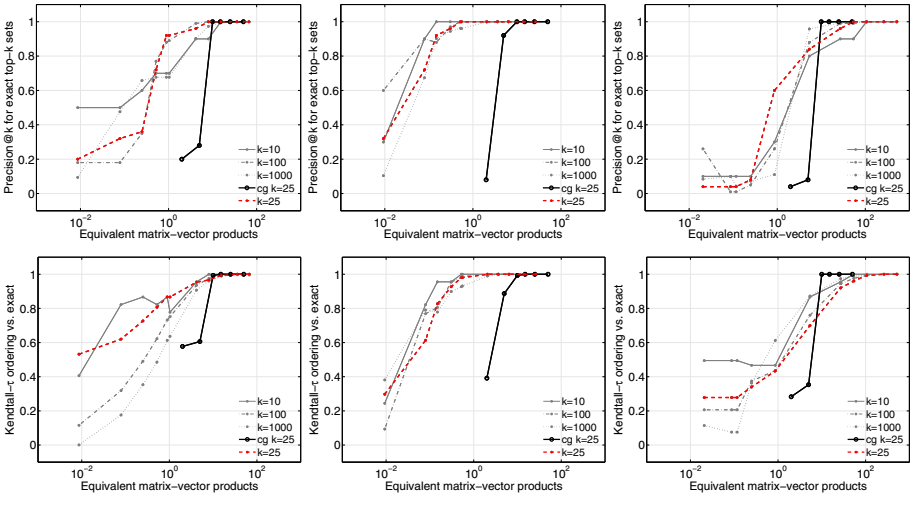

Fig. 3. Precision *(top)* and Kendall-τ correlation *(bottom)* for top-k Katz time scores on `arxiv` *(left)*, `dblp` *(center)*, and `flickr` *(right)*. We use the same value of α as Figure 1.

On the `dblp` graph, the top-k algorithm produces almost the exact Katz top-k set with just slightly more than 1 effective matrix-vector product. For `flickr`, we see a striking transition around 1 effective matrix-vector product, when it seems to suddenly "lock" the top-k sets, then slowly adjust their order. In all of the experiments, the CG algorithm does not provide any useful information until it converges. Our top-k algorithm produces useful partial information in much less work and time.

Runtime. Finally, we present the runtime of our pairwise and top-k methods in Table 2. We explore two values of α for Katz:

$$\begin{array}{ll} \text{easy-}\alpha & 1/(10\|A\|_1 + 10) \\ \text{hard-}\alpha & 1/(\max(\lambda(A)) + 1). \end{array}$$

The former should converge more quickly than the latter. In the pairwise case, we evaluate the the runtime on three pairs of nodes. These pairs were chosen such that there was a high degree-high degree pair, a high degree-low degree pair, and a low degree-low degree pair. For these, we use the shorthand high-high pair, etc. The results show the impact of these choices. As expected, the easy-α cases converged faster and commute time converged slower than either Katz score. In this small sample, the degree of the pairs played a role. On `flickr`, for example, the low-low pair converged fastest for Katz, whereas the high-low pair converged fastest for commute time. The solution tolerance was 10^{-4}. We do not report separate computation times for the conjugate gradient method, but note that the previous experiments suggest it should take about half the time for the Katz problems and about twice as long for the commute time experiments. In the top-k problems, we start the algorithm at one of the vertices among the same pairs of nodes. We terminate it when the largest element in the residual vector is smaller than $10^{-4}\alpha d_u$, where d_u is the degree of the source node. For most of the experiments, this setting produced a 2-norm residual smaller than 10^{-4}, which is the same convergence criterion for CG.

Table 2. Runtime (in seconds) of the pairwise *(left)* and top-k *(right)* algorithms for Katz scores and commute time. See the text for a description of the cases.

Graph	Pairs	Score		
		Katz		Commute
		easy-α	hard-α	
`arxiv`	High, high	0.6081	2.6902	24.8874
	High, low	0.6068	2.3689	19.7079
	Low, low	0.3619	0.5842	10.7421
`dblp`	High, high	0.3266	1.7044	10.3836
	High, low	0.3436	1.3010	8.8664
	Low, low	0.2133	0.5458	8.3463
`flickr`	High, high	5.1061	12.7508	227.2851
	High, low	4.2578	11.0659	82.0949
	Low, low	2.6037	3.4782	172.5125

Graph	Degree	Katz	
		easy-α	hard-α
`arxiv`	High	0.0027	0.2334
	Low	0.0003	0.2815
	Low	0.0004	0.5315
`dblp`	High	0.0012	0.0163
	Low	0.0011	0.0161
	Low	0.0007	0.0173
`flickr`	High	0.0741	0.0835
	Low	0.0036	36.2140
	Low	0.0040	0.0063

6 Conclusions and Future Work

Measures based on ensembles of paths such as the Katz score and the commute time have been found useful in several applications such as link prediction and collaborative filtering. In this paper, motivated by applications, we

focused on two problems related to fast approximations for these scores. First, for finding the score between a specified pair of nodes, we have proposed an efficient algorithm to compute it and also obtain upper and lower bounds, making use of a technique for computing bilinear forms. Second, for finding the top-k nodes that have the highest Katz scores with respect to a given source node, we have proposed a top-k algorithm based on a variant of the Richardson stationary method used in personalized PageRank.

We have conducted a set of experiments on three real-world datasets and obtained many encouraging results. Our experiments demonstrate the scalability of the proposed method to large networks, without giving up much accuracy with respect to the direct methods (that are infeasible on large networks).

There are many possible extensions of our techniques. For example, the algorithm we propose for computing the Katz and commute time between a given pair of nodes extends to the case where one wants to find the aggregate score between a node and a set of nodes. This could be useful in methods that find clusters using commute time [16,17,25]. In these cases, the commute time between a node and a group of nodes (e.g., a cluster) measures their affinity. We plan to explore this generalization in future work.

Furthermore, in link prediction, anomalous link detection, and recommendation, the underlying graph is dynamic and evolving in time. These tasks require almost real-time computation because the results should reflect the latest state of the network, not the results of an offline cached computation. Therefore, calculation of these metrics must be as fast as possible. We hope to evaluate our algorithms in such a dynamic setting, where we believe they should fit nicely because of the fast computation and preprocessing-free nature. An alternative is to combine some offline processing with techniques to get fast online estimates of the scores. These techniques invariably involve a compromise between scalability of the approach (e.g., computing a matrix factorization offline) and the complexity of implementation (see [10,3] for examples in personalized PageRank).

One key weakness of our current top-k algorithms is that they do no apply to estimating the closest commute time neighbors. This problem arises because the expression for all the commute times relative to a given node involves *all* of the diagonal entries of the matrix inverse, whereas the top-k algorithm only finds an approximation to a single linear system. We are currently investigating a diffusion-based measure that is inspired by commute time and can be used with our Richardson technique. Preliminary results show good agreement between the k closest nodes using commute time and the top-k set of the diffusion measure.

References

1. Acar, E., Dunlavy, D.M., Kolda, T.G.: Link prediction on evolving data using matrix and tensor factorizations. In: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops, ICDMW 2009, pp. 262–269. IEEE Computer Society, Los Alamitos (2009)
2. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using PageRank vectors. In: Proc. of the 47th Annual IEEE Sym. on Found. of Comp. Sci. (2006)

3. Berkhin, P.: Bookmark-coloring algorithm for personalized PageRank computing. Internet Math. 3(1), 41–62 (2007)
4. Foster, K.C., Muth, S.Q., Potterat, J.J., Rothenberg, R.B.: A faster Katz status score algorithm. Comput. & Math. Organ. Theo. 7(4), 275–285 (2001)
5. Fouss, F., Pirotte, A., Renders, J.-M., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans. Knowl. Data Eng. 19(3), 355–369 (2007)
6. Göbel, F., Jagers, A.A.: Random walks on graphs. Stochastic Processes and their Applications 2(4), 311–336 (1974)
7. Golub, G.H., Loan, C.F.V.: Matrix Computations, 3rd edn. Johns Hopkins Univ. Press, Baltimore (1996)
8. Golub, G.H., Meurant, G.: Matrices, moments and quadrature. In: Numerical analysis 1993 (Dundee, 1993). Pitman Res. Notes Math. Ser., vol. 303, pp. 105–156. Longman Sci. Tech., Harlow (1994)
9. Golub, G.H., Meurant, G.: Matrices, moments and quadrature ii; how to compute the norm of the error in iterative methods. BIT Num. Math. 37(3), 687–705 (1997)
10. Jeh, G., Widom, J.: Scaling personalized web search. In: Proceedings of the 12th International Conference on the World Wide Web, pp. 271–279. ACM, New York (2003)
11. Katz, L.: A new status index derived from sociometric analysis. Psychometrika 18, 39–43 (1953)
12. Li, P., Liu, H., Yu, J.X., He, J., Du, X.: Fast single-pair simrank computation. In: Proc. of the SIAM Intl. Conf. on Data Mining (SDM 2010), Columbus, OH (2010)
13. Liben-Nowell, D., Kleinberg, J.M.: The link prediction problem for social networks. In: Proc. of the ACM Intl. Conf. on Inform. and Knowlg. Manage. CIKM 2003 (2003)
14. McSherry, F.: A uniform approach to accelerated PageRank computation. In: Proc. of the 14th Intl. Conf. on the WWW, pp. 575–582. ACM Press, New York (2005)
15. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University (November 1999)
16. Qiu, H., Hancock, E.R.: Commute times for graph spectral clustering. In: Gagalowicz, A., Philips, W. (eds.) CAIP 2005. LNCS, vol. 3691, pp. 128–136. Springer, Heidelberg (2005)
17. Qiu, H., Hancock, E.R.: Clustering and embedding using commute times. IEEE Trans. Pattern Anal. Mach. Intell. 29(11), 1873–1890 (2007)
18. Rattigan, M.J., Jensen, D.: The case for anomalous link discovery. SIGKDD Explor. Newsl. 7(2), 41–47 (2005)
19. Saerens, M., Fouss, F., Yen, L., Dupont, P.: The principal components analysis of a graph, and its relationships to spectral clustering. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 371–383. Springer, Heidelberg (2004)
20. Sarkar, P., Moore, A.W.: A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In: Proc. of the 23rd Conf. on Uncert. in Art. Intell., UAI 2007 (2007)
21. Sarkar, P., Moore, A.W., Prakash, A.: Fast incremental proximity search in large graphs. In: Proc. of the 25th Intl. Conf. on Mach. Learn., ICML 2008 (2008)
22. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. In: Proc. of the 40th Ann. ACM Symp. on Theo. of Comput. (STOC 2008), pp. 563–568 (2008)

23. Varga, R.: Matrix Iterative Analysis. Prentice-Hall, Englewood Cliffs (1962)
24. Wang, C., Satuluri, V., Parthasarathy, S.: Local probabilistic models for link prediction. In: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, ICDM 2007, Washington, DC, USA, pp. 322–331. IEEE Computer Society, Los Alamitos (December 2007)
25. Yen, L., Fouss, F., Decaestecker, C., Francq, P., Saerens, M.: Graph nodes clustering based on the commute-time kernel. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 1037–1045. Springer, Heidelberg (2007)