Approximating Betweenness Centrality

David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail

College of Computing
Georgia Institute of Technology
{bader,kintali,kamesh,mihail}@cc.gatech.edu

Abstract. Betweenness is a centrality measure based on shortest paths, widely used in complex network analysis. It is computationally-expensive to exactly determine betweenness; currently the fastest-known algorithm by Brandes requires $O(nm)$ time for unweighted graphs and $O(nm + n^2 \log n)$ time for weighted graphs, where n is the number of vertices and m is the number of edges in the network. These are also the worst-case time bounds for computing the betweenness score of a single vertex. In this paper, we present a novel approximation algorithm for computing betweenness centrality of a given vertex, for both weighted and unweighted graphs. Our approximation algorithm is based on an adaptive sampling technique that significantly reduces the number of single-source shortest path computations for vertices with high centrality. We conduct an extensive experimental study on real-world graph instances, and observe that our random sampling algorithm gives very good betweenness approximations for biological networks, road networks and web crawls.

1 Introduction

One of the fundamental problems in network analysis is to determine the *importance* (or the *centrality*) of a particular vertex (or an edge) in a network. Some of the well-known metrics for computing centrality are closeness [1], stress [2] and betweenness [3, 4]. Of these indices, betweenness has been extensively used in recent years for the analysis of social-interaction networks, as well as other large-scale complex networks. Some applications include lethality in biological networks [5–7], study of sexual networks and AIDS [8], identifying key actors in terrorist networks [9, 10], organizational behavior [11], and supply chain management processes [12]. Betweenness is also used as the primary routine in popular algorithms for clustering and community identification [13] in real-world networks. For instance, the Girvan-Newman [14] algorithm iteratively partitions a network by identifying edges with high betweenness scores, removing them and recomputing centrality scores.

Betweenness is a global centrality metric that is based on shortest-path enumeration. Consider a graph $G = (V, E)$, where V is the set of vertices representing *actors* or *nodes* in the complex network, and E, the set of edges representing the relationships between the vertices. The number of vertices and edges are denoted by n and m respectively. The graphs can be directed or undirected.

We will assume that each edge $e \in E$ has a positive integer weight $w(e)$. For unweighted graphs, we use $w(e) = 1$. A *path* from vertex s to t is defined as a sequence of edges $\langle u_i, u_{i+1} \rangle$, $0 \leq i < l$, where $u_0 = s$ and $u_l = t$. The *length* of a path is the sum of the weights of edges. We use $d(s,t)$ to denote the distance between vertices s and t (the minimum length of any path connecting s and t in G). Let us denote the total number of shortest paths between vertices s and t by λ_{st}, and the number passing through vertex v by $\lambda_{st}(v)$. Let $\delta_{st}(v)$ denote the $fraction$ of shortest paths between s and t that pass through a particular vertex v i.e., $\delta_{st}(v) = \frac{\lambda_{st}(v)}{\lambda_{st}}$. We call $\delta_{st}(v)$ the *pair-dependency* of s, t on v.

Betweenness centrality [3, 4] of a vertex v is defined as

$$BC(v) = \sum_{s \neq v \neq t \in V} \delta_{st}(v)$$

Currently, the fastest known algorithm for exactly computing betweenness of all the vertices, designed by Brandes [15], requires at least $O(nm)$ time for unweighted graphs and $O(nm + n^2 \log n)$ time for weighted graphs, where n is the number of vertices and m is the number of edges. Thus, for large-scale graphs, exact centrality computation on current workstations is not practically viable. In prior work, we explored high performance computing techniques [16] that exploit the typical small-world graph topology to speed up exact centrality computation. We designed novel parallel algorithms to exactly compute various centrality metrics, optimized for real-world networks. We also demonstrate the capability to compute exact betweenness on several large-scale networks (vertices and edges in the order of millions) from the Internet and social interaction data; these networks are three orders of magnitude larger than instances that can be processed by current social network analysis packages.

Fast centrality estimation is thus an important problem, as a good approximation would be an acceptable alternative to exact scores. Currently the fastest exact algorithms for shortest path enumeration-based metrics require n shortest-path computations; however, it is possible to estimate centrality by extrapolating scores from a fewer number of path computations. Using a random sampling technique, Eppstein and Wang [17] show that the closeness centrality of all vertices in a weighted, undirected graph can be approximated with high probability in $O(\frac{\log n}{\epsilon^2}(n \log n + m))$ time, and an additive error of at most $\epsilon \Delta_G$ (ϵ is a fixed constant, and Δ_G is the diameter of the graph). However, betweenness centrality scores are harder to estimate, and the quality of approximation is found to be dependent on the vertices from which the shortest path computations are initiated from (in this paper, we will refer to them as the set of *source vertices* for the approximation algorithm). Recently, Brandes and Pich [18] presented centrality estimation heuristics, where they experimented with different strategies for selecting the source vertices. They observe that a random selection of source vertices is superior to deterministic strategies. In addition to exact parallel algorithms, we also discussed parallel techniques to compute approximate betweenness centrality in [16], using a random source selection strategy.

While prior approaches approximate centrality scores of *all* vertices in the graph, there are no known algorithms to compute the centrality of a single vertex in time faster than computing the betweenness of all vertices. In this paper, we present a novel *adaptive sampling*-based algorithm for approximately computing betweenness centrality of a given vertex. Our primary result is as follows:

Theorem : For $0 < \epsilon < 0.5$, if the centrality of a vertex v is n^2/t for some constant $t \geq 1$, then with probability $\geq 1 - 2\epsilon$ its centrality can be estimated to within a factor of $1/\epsilon$ with ϵt samples of source vertices.

The rest of this paper is organized as follows. We review the currently-known fastest sequential algorithm by Brandes in Section 2. We present our approximation algorithm based on adaptive sampling and its analysis in Section 3. Section 4 is an experimental study of our approximation technique on several real-world networks. We conclude with a summary of open problems in Section 5.

2 Exact computation of Betweenness Centrality

Brandes' algorithm [15] shows how to compute centrality scores of all the vertices in the graph in the same asymptotic time bounds as n SSSP computations.

2.1 Brandes' Algorithm

Define the *dependency* of a source vertex $s \in V$ on a vertex $v \in V$ as $\delta_{s*}(v) = \sum_{t \neq s \neq v \in V} \delta_{st}(v)$. Then the betweenness score of v can be then expressed as $BC(v) = \sum_{s \neq v \in V} \delta_{s*}(v)$. Also, let $P_s(v)$ denote the set of *predecessors* of a vertex v on shortest paths from s: $P_s(v) = \{u \in V : \langle u, v \rangle \in E, d(s,v) = d(s,u) + w(u,v)\}$. Brandes shows that the dependencies satisfy the following recursive relation, which is the most crucial step in the algorithm analysis.

Theorem 1. *The dependency of $s \in V$ on any $v \in V$ obeys*

$$\delta_{s*}(v) = \sum_{w : v \in P_s(w)} \frac{\lambda_{sv}}{\lambda_{sw}} (1 + \delta_{s*}(w))$$

First, n SSSP computations are done, one for each $s \in V$. The predecessor sets $P_s(v)$ are maintained during these computations. Next, for every $s \in V$, using the information from the shortest paths tree and predecessor sets along the paths, compute the dependencies $\delta_{s*}(v)$ for all other $v \in V$. To compute the centrality value of a vertex v, we finally compute the sum of all dependency values. The $O(n^2)$ space requirements can be reduced to $O(m+n)$ by maintaining a *running centrality score*. For additional details, we refer the reader to Brandes' paper [15].

3 Adaptive-sampling based approximation

The adaptive sampling technique was introduced by Lipton and Naughton [19] for estimating the size of the transitive closure of a digraph. Prior to their work, algorithms for estimating transitive closure were based on randomly sampling source-vertices, solving the single-source reachability problem for the sampled vertices, and using this information to estimate the size of the transitive closure. The Lipton-Naughton algorithm introduces adaptive sampling of source-vertices, that is, the number of samples varies with the information obtained from each sample.

In this section, we give an *adaptive sampling* algorithm for computing betweenness of a given vertex v. It is a sampling algorithm in that it estimates the centrality by sampling a subset of vertices and performing SSSP computations from these vertices. It is termed *adaptive*, because the number of samples required varies with the information obtained from each sample.

The following lemma is easy to see and the proof is omitted.

Lemma 1. *$BC(v)$ is zero iff its neighboring vertices induce a clique.*

Let a_i denote the dependency of the vertex v_i on v i.e., $a_i = \delta_{v_i *}(v)$. Let $A = \sum a_i = BC(v)$. It is easy to verify that $0 \leq a_i \leq n-2$ and $0 \leq A \leq (n-1)(n-2)/2$. The quantity we wish to estimate is A. Consider doing so with the following algorithm:

Algorithm 1 *Repeatedly sample a vertex $v_i \in V$; perform SSSP (using BFS or Dijkstra's algorithm) from v_i and maintain a running sum S of the dependency scores $\delta_{v_i *}(v)$. Sample until S is greater than cn for some constant $c \geq 2$. Let the total number of samples be k. The estimated betweenness centrality score of v, $BC(v)$ is given by $\frac{nS}{k}$.*

Let X_i be the random variable representing the dependency of a randomly sampled vertex on v. The probability of an event x is denoted by $\mathsf{Pr}[\, x \,]$. We establish the following lemmas to analyze the above algorithm.

Lemma 2. *Let $E[X_i]$ denote the expectation of X_i and $Var[X_i]$ denote the variance of X_i. Then, $E[X_i] = A/n$, $E[{X_i}^2] \leq A$, and $Var[X_i] \leq A$.*

The next lemma is useful in proving a lower bound on the expected number of samples made before stopping. The proof is presented in the Appendix.

Lemma 3. *Let $k = \epsilon n^2/A$. Then,*

$$\mathsf{Pr}[\, X_1 + X_2 + \cdots + X_k \geq cn \,] \leq \frac{\epsilon}{(c-\epsilon)^2}$$

Lemma 4. *Let $k \geq \epsilon n^2/A$ and $d > 0$. Then*

$$\mathsf{Pr}\left[\, |\frac{n}{k}\left(\sum_{i=1}^{k} X_i\right) - A| \geq dA \,\right] \leq \frac{1}{\epsilon d^2}$$

Theorem 2. *Let $\tilde{A}$ be the estimate of A in the above procedure and let $A > 0$. Then for $0 < \epsilon < 0.5$ with probability $\geq 1 - 2\epsilon$,* **Algorithm 1** *estimates A to within a factor of $1/\epsilon$.*

Proof. There are two ways that the algorithm can fail: (i) it can stop too early to guarantee a good error bound, (ii) it can stop after enough samples but with a bad estimate.

First we claim that the procedure is unlikely to stop with $k \leq n^2/A$. We have that

$$\mathsf{Pr}\,[\ (\exists j)(j \leq k) \wedge (X_1 + X_2 + \cdots + X_j \geq cn)\ ] \leq \mathsf{Pr}\,[\ X_1 + X_2 + \cdots + X_k \geq cn\ ]$$

where $k = \dfrac{\epsilon n^2}{A}$, because the event to the right of the inequality implies the event to the left. But by Lemma 3, the right side of this equation is at most $\epsilon/(c-\epsilon)^2$. Substituting $c = 2$ and noting that $0 < \epsilon < 0.5$, we get that this probability is less than ϵ.

Next we turn to the accuracy of the estimate. If $k = \epsilon n^2/A$, by Lemma 4 the estimate,

$$\tilde{A} = \frac{n}{k} \sum_{i=1}^{k} X_i$$

is within dA of A with probability $\geq 1/(\epsilon d^2)$. Letting $d = 1/\epsilon$, this is just ϵ.

Putting the two ways of failure together, we get that the total probability of failure is less than $\epsilon + (1-\epsilon)\epsilon$, which is less than 2ϵ. Finally, note that if $A > 0$, there must be at least one i such that $a_i > 0$, so the algorithm will terminate. The case when $A = 0$ (i.e., centrality of v is 0) can be detected using Lemma 1 (before running the algorithm).

An interesting aspect of our theorem is that the sampling is adaptive. usually such sampling procedures perform a fixed number of samples. Here it is critical that the algorithm adapts it behavior. Substituting $A = \dfrac{n^2}{t}$ in our analysis we get the following theorem.

Theorem 3. *For $0 < \epsilon < 0.5$, if the centrality of a vertex v is n^2/t for some constant $t \geq 1$, then with probability $\geq 1 - 2\epsilon$ its centrality can be estimated to within a factor of $1/\epsilon$ with ϵt samples of source vertices.*

Although our theoretical result is valid only for high centrality nodes, our experimental results (presented in the next section) show a similar behavior for all the vertices.

4 Experimental Study

We assess the quality of the sampling-based approximation algorithm on several real-world graph instances (see Table 1). We use the parallel centrality analysis

toolkit SNAP [20] to compute exact betweenness scores. Since the execution time and speedup achieved by the approximation approach are directly proportional to the number of BFS/shortest path computations, we do not report performance results in this section. For a detailed discussion of exact centrality computation in parallel, and optimizations for small-world graphs, please refer to [16].

Network data

Label	Network	n	m	Details	Source
`rand`	random graph	2000	7980	synthetic, undirected	[21]
`pref-attach`	preferential attachment	2000	7980	synthetic, undirected	[22]
`bio-pin`	human protein interactions	8503	32,191	undirected	[23]
`crawl`	web-crawl (`stanford.edu`)	9914	36,854	directed	[24]
`cite`	Lederberg citation network	8843	41,601	directed	[24, 25]
`road`	Rome, Italy road network	3353	4435	weighted, undirected	[26]

Table 1. Networks used in the experimental study

We experiment with two synthetic graph instances and four real networks in this study. `rand` is an unweighted, undirected random network of 2000 vertices and 7980 edges, generated using the Erdős–Rényi graph model [27]. This synthetic graph has a low diameter, low clustering, and a Gaussian degree distribution. `pref-attach` is a synthetic graph generated using the Preferential attachment model proposed by Barabási and Albert [28]. This model generates graphs with heavy-tailed degree distributions and scale-free properties. Vertices are added one at a time, and for each of them, we create a fixed number of edges connecting to existing vertices, with probability proportional to their degree. `bio-pin` is a biological network that represents interactions in the human proteome [29, 23]. This graph is undirected, unweighted and exhibits small-world characteristics. `crawl` corresponds to the `wb-cs-stanford` network in the UF sparse matrix collection [24]. It is a directed graph, where vertices correspond to pages in the Stanford Computer Science domain, and edges represent links. `cite` is a directed graph from the Pajek network collection [25]. It corresponds to papers by and citing J. Lederberg (1945-2002). `road` is a weighted graph of 3353 vertices and 4435 edges that corresponds to a large portion of the road network of Rome, Italy from 1999 [26]. Vertices correspond to intersections between roads, and edges correspond to roads or road segments. Edge weights are physical distances in metres. Road networks have more structure and a higher diameter than the other networks considered in this study.

Methodology

Our goal in this study is to quantify the approximation quality, and so we primarily compare the approximation results to exact scores. We first compute

exact centrality scores of all the networks in Table 1. In most data sets, we are interested in high-centrality vertices, as they are the critical entities and are used in further analysis. From the exact scores, we identify vertices whose centrality scores are an order of magnitude greater than the rest of the network. For these vertices, we study the trade-off between computation and approximation quality by varying the parameter c in Algorithm 1. We also show that it is easy to estimate scores of low-centrality vertices. We chose small networks for ease of analysis and visualization, but the approximation algorithm can be effectively applied to large networks as well (see, for instance, the networks considered in [16]).

Experiments

Figure 1 plots the distribution of exact and approximate betweenness scores for the six different test instances. Note that the synthetic networks, `rand` and `pref-attach` show significantly lower variation in exact centrality scores compared to the real instances. Also, there are a significant percentage of low-centrality vertices (scores less than, or close to, n) in `cite`, `crawl` and `bio-pin`.

We apply Algorithm 1 to estimate betweenness centrality scores of all the vertices in the test instances. In order to visualize the data better, we plot a smoothed curve of the estimated betweenness centrality data that is superimposed with the exact centrality score scatter-plot. We set the parameter c in Algorithm 1 to 5 for these experiments. In addition, we impose a cut-off of $\frac{n}{20}$ on the number of samples. Observe that in all the networks, the estimated centrality scores are very close to the exact ones, and we are guaranteed to cut down on the computation by a factor of nearly 20.

To further study the quality of approximation for high-centrality vertices, we select the top 1% of the vertices (about 30) ordered by exact centrality score in each network, and compute their estimated centrality scores using the adaptive-sampling algorithm. Since the source vertices in the adaptive approach are chosen randomly, we repeat the experiment five times for each vertex and report the mean and variance in approximation error. Figure 2 plots the mean percentage approximation error in the computed scores for these high centrality vertices, when the value of c (see Algorithm 3) is set to 5. The vertices are sorted by exact centrality score on the X-axis. The error bars in the charts indicate the variance in estimated score due to random runs, for each network. For the random graph instance, the average error is about 5%, while it is roughly around 10% for the rest of the networks. Except for a few anomalous vertices, the error variance is within reasonable bounds in all the graph classes.

Figure 3 plots the percentage of BFS/SSSP computations required for approximating the centrality scores, when c is set to 5. This algorithmic count is an indicator of the amount of work done by the approximation algorithm. The vertices are ordered again by their exact centrality scores from left to right, with the vertex with the least score to the left. A common trend we observe across all graph classes is that the percentage of source vertices decreases as the centrality score increases – this implies that the scores of high centrality vertices can

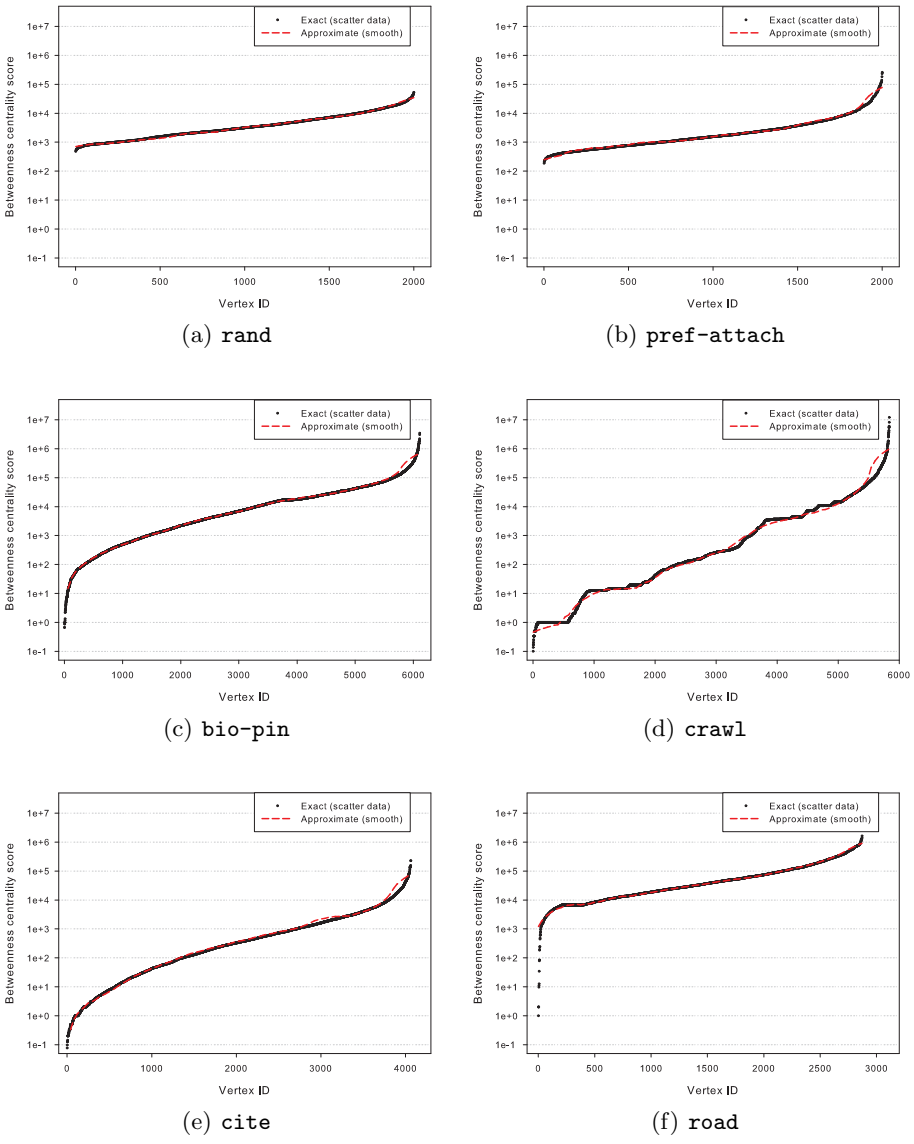

Fig. 1. A scatter plot of exact betweenness scores of all the vertices (in sorted order), and a line plot of their estimated betweenness scores (the approximate betweenness scatter data is smoothed by a local smoothing technique using polynomial regression)

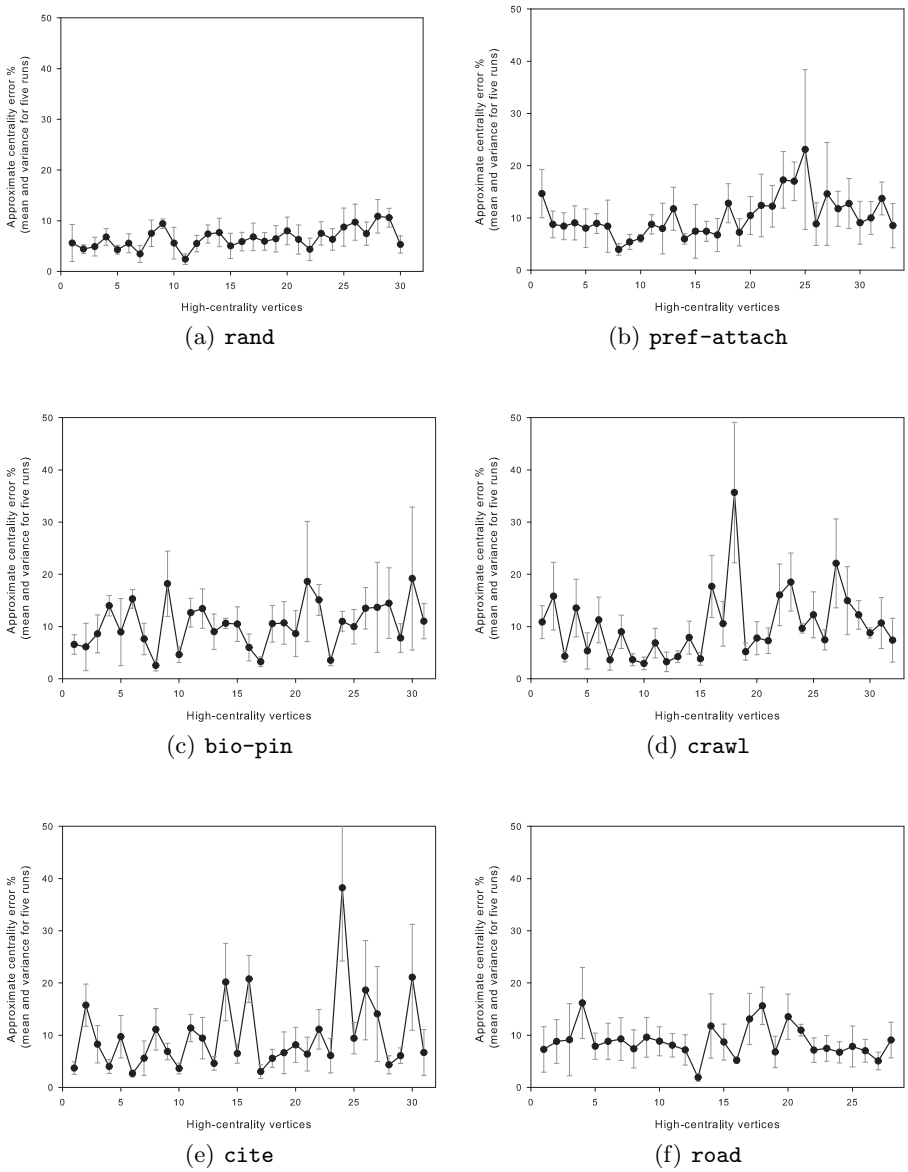

Fig. 2. Average estimated betweenness error percentage (in comparison to the exact centrality score) for multiple runs. The adaptive sampling parameter c is set to 5 for all experiments and the error bars indicate the variance.

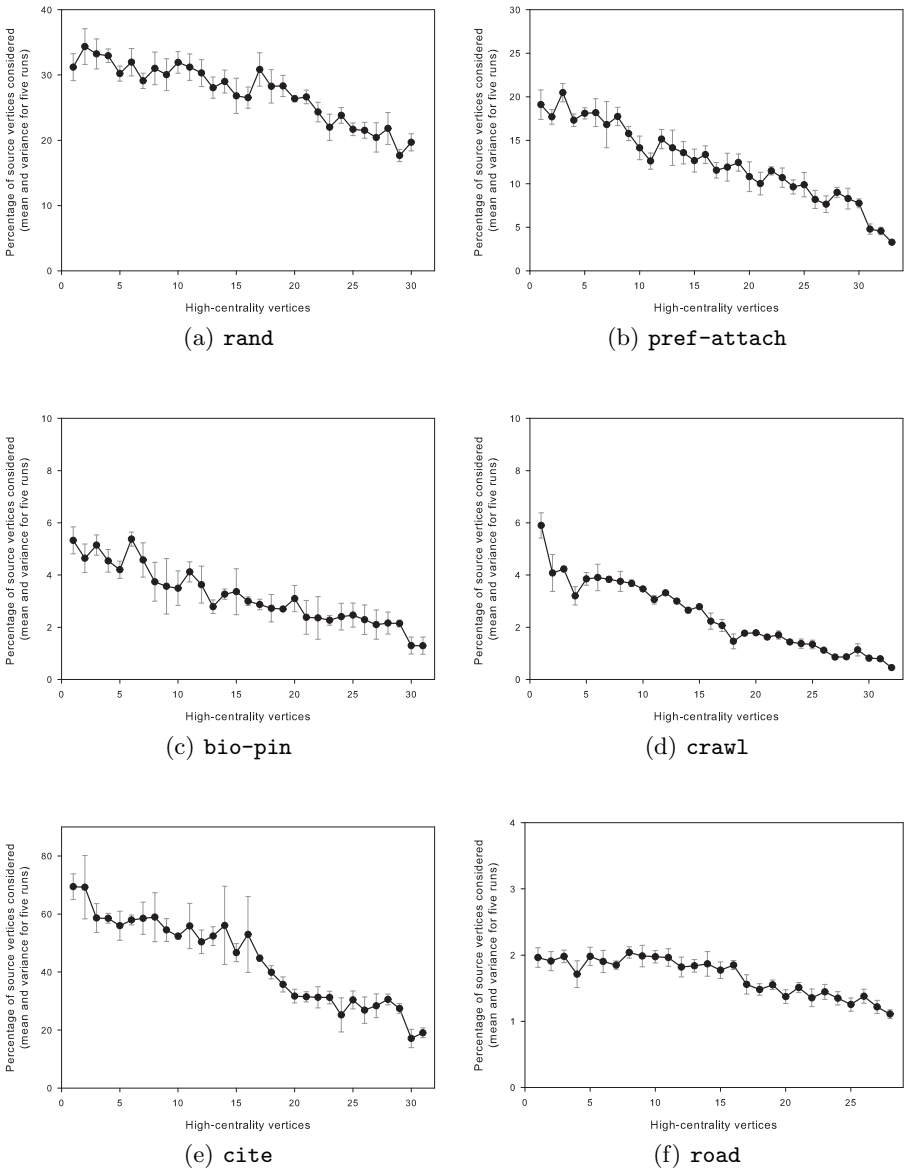

Fig. 3. The number of samples/SSSP computations as a fraction of n, the total number of vertices. This algorithmic count is an indicator of the amount of work done by the approximation algorithm. The adaptive sampling parameter c is set to 5, and the error bars indicate the variance from 5 runs.

be approximated with lesser work using the adaptive sampling approach. Also, this value is significantly lower for `crawl`, `bio-pin` and `road` compared to other graph classes.

We can also vary the parameter c, which affects both the percentage of BFS/SSSP computations and the approximation quality. Table 2 summarizes the average performance on each graph instance, for different values of c. Taking only high-centrality vertices into consideration, we report the mean approximation error and the number of samples for each graph instance. As expected, we find that the error decreases as the parameter c is increased, while the number of samples increases. Since the highest centrality value is around $10 * n$ for the citation network, a significant number of shortest path computations have to be done even for calculating scores with a reasonable accuracy. But for other graph instances, particularly the road network, web-crawl and the protein interaction network, $c = 5$ offers a good trade-off between computation and the approximation quality.

Network	`rand`	`pref-attach`	`bio-pin`	`crawl`	`cite`	`road`
t = 2						
Avg. error	16.28%	29.39%	46.72%	33.69%	32.51%	22.58%
Avg. $\frac{k}{n}$	11.31%	5.36%	1.30%	0.96%	17.00%	0.68 %
t = 5						
Avg. error	6.51%	10.28%	10.49%	10.31%	9.98%	8.79%
Avg. $\frac{k}{n}$	27.37%	12.38%	3.20%	2.42%	43.85%	1.68%
t = 10						
Avg. error	5.62%	6.13%	7.17%	7.04%	–	7.39%
Avg. $\frac{k}{n}$	54.51%	24.66%	6.33%	4.89%	–	3.29%

Table 2. Observed average-case algorithmic counts, as the value of the sampling parameter c is varied. The average error percentage is the deviation of the estimated score from the exact score, and the $\frac{k}{n}$ percentage indicates the number of samples/SSSP computations.

5 Conclusion and Open Problems

We presented a novel approximation algorithm for computing betweenness centrality, of a given vertex, in both weighted and unweighted graphs. Our approximation algorithm is based on an adaptive sampling technique that significantly reduces the number of single-source shortest path computations for vertices with high centrality. We conduct an extensive experimental study on real-world graph instances, and observe that the approximation algorithm performs well on web crawls, road networks and biological networks.

Approximating the centrality of *all* vertices in time less than $O(nm)$ for unweighted graphs and $O(nm + n^2 \log n)$ for weighted graphs is a challenging

open problem. Designing a fully dynamic algorithm for computing betweenness is very useful.

Acknowledgments

This work was supported in part by NSF Grants CAREER CCF-0611589, NSF DBI-0420513, ITR EF/BIO 03-31654, and DARPA Contract NBCH 30390004. Kamesh Madduri's work is supported in part by the NASA Graduate Student Researcher Program Fellowship (NASA NP-2005-07-375-HQ). We thank Richard Lipton for helpful discussions.

Appendix

Lemma 3

Let $k = \epsilon n^2/A$. Then,

$$\mathsf{Pr}\left[\, X_1 + X_2 + \cdots + X_k \geq cn \,\right] \leq \frac{\epsilon}{(c-\epsilon)^2}$$

Proof. We have

$$\begin{aligned}
\mathsf{Pr}\left[\, X_1 + \cdots + X_k \geq cn \,\right] &= \mathsf{Pr}\left[\left(X_1 - \frac{A}{n}\right) + \cdots + \left(X_k - \frac{A}{n}\right) \geq cn - \frac{kA}{n} \right] \\
&= \mathsf{Pr}\left[\left(X_1 - \frac{A}{n}\right) + \cdots + \left(X_k - \frac{A}{n}\right) \geq cn - \epsilon n \right] \\
&\leq \sum_i \mathsf{Pr}\left[X_i - \frac{A}{n} \geq (c-\epsilon)n \right] \\
&\leq \sum_i \frac{1}{(c-\epsilon)^2 n^2} Var[X_i] \\
&= \frac{1}{(c-\epsilon)^2 n^2} \sum_i Var[X_i] \\
&\leq \frac{1}{(c-\epsilon)^2 n^2} kA \\
&= \frac{\epsilon}{(c-\epsilon)^2}
\end{aligned}$$

Note that we have used Chebychev's inequality and union bounds in the above proof. We bound the error in the estimated value of A with the following lemma.

Lemma 4

Let $k \geq \epsilon n^2/A$ and $d > 0$. Then

$$\mathsf{Pr}\left[\left| \frac{n}{k} \left(\sum_{i=1}^{k} X_i \right) - A \right| \geq dA \right] \leq \frac{1}{\epsilon d^2}$$

Proof.

$$\begin{aligned} \mathsf{Pr}\left[\left| \frac{n}{k} \left(\sum_{i=1}^{k} X_i \right) - A \right| \geq t \right] &= \mathsf{Pr}\left[\left| \left(\sum_{i=1}^{k} X_i \right) - \frac{k}{n} A \right| \geq \frac{kt}{n} \right] \\ &= \mathsf{Pr}\left[\left| \left(\sum_{i=1}^{k} X_i - \frac{1}{n} A \right) \right| \geq \frac{kt}{n} \right] \\ &\leq \frac{n^2}{k^2 t^2} k \cdot Var[X_i] \end{aligned}$$

Let $k = \lambda \dfrac{n^2}{A}$, where $\lambda \geq \epsilon$. Then the above probability is less than or equal to

$$\frac{n^2}{k^2 t^2} k \cdot Var[X_i] \leq \frac{n^2}{\lambda \frac{n^2}{A} t^2} A$$

which is just $\dfrac{A^2}{\lambda t^2}$. Setting $Ad = t$ gives

$$\frac{1}{\lambda d^2} \leq \frac{1}{\epsilon d^2}$$

References

1. Sabidussi, G.: The centrality index of a graph. Psychometrika **31** (1966) 581–603
2. Shimbel, A.: Structural parameters of communication networks. Bulletin of Mathematical Biophysics **15** (1953) 501–507
3. Freeman, L.: A set of measures of centrality based on betweenness. Sociometry **40**(1) (1977) 35–41
4. Anthonisse, J.: The rush in a directed graph. Report BN9/71, Stichting Mathematisch Centrum, Amsterdam, Netherlands (1971)
5. Jeong, H., Mason, S., Barabási, A.L., Oltvai, Z.: Lethality and centrality in protein networks. Nature **411** (2001) 41–42
6. Pinney, J., McConkey, G., Westhead, D.: Decomposition of biological networks using betweenness centrality. In: Proc. 9th Ann. Int'l Conf. on Research in Computational Molecular Biology (RECOMB 2005), Cambridge, MA (May 2005) Poster session.
7. del Sol, A., Fujihashi, H., O'Meara, P.: Topology of small-world networks of protein-protein complex structures. Bioinformatics **21**(8) (2005) 1311–1315
8. Liljeros, F., Edling, C., Amaral, L., Stanley, H., Åberg, Y.: The web of human sexual contacts. Nature **411** (2001) 907–908

9. Krebs, V.: Mapping networks of terrorist cells. Connections **24**(3) (2002) 43–52
10. Coffman, T., Greenblatt, S., Marcus, S.: Graph-based technologies for intelligence analysis. Communications of the ACM **47**(3) (2004) 45–47
11. Buckley, N., van Alstyne, M.: Does email make white collar workers more productive? Technical report, University of Michigan (2004)
12. Cisic, D., Kesic, B., Jakomin, L.: Research of the power in the supply chain. International Trade, Economics Working Paper Archive EconWPA (April 2000)
13. Newman, M.: The structure and function of complex networks. SIAM Review **45**(2) (2003) 167–256
14. Girvan, M., Newman, M.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences USA **99**(12) (2002) 7821–7826
15. Brandes, U.: A faster algorithm for betweenness centrality. J. Mathematical Sociology **25**(2) (2001) 163–177
16. Bader, D., Madduri, K.: Parallel algorithms for evaluating centrality indices in real-world networks. In: Proc. 35th Int'l Conf. on Parallel Processing (ICPP), Columbus, OH, IEEE Computer Society (August 2006)
17. Eppstein, D., Wang, J.: Fast approximation of centrality. In: Proc. 12th Ann. Symp. Discrete Algorithms (SODA-01), Washington, DC (2001) 228–229
18. Brandes, U., Pich, C.: Centrality estimation in large networks. To appear in Intl. Journal of Bifurcation and Chaos, Special Issue on Complex Networks' Structure and Dynamics (2007)
19. Lipton, R., Naughton, J.: Estimating the size of generalized transitive closures. In: VLDB. (1989) 165–171
20. Madduri, K., Bader, D.: Small-world Network Analysis in Parallel: a toolkit for centrality analysis. `http://www.cc.gatech.edu/~kamesh` (2007)
21. Madduri, K., Bader, D.: GTgraph: A suite of synthetic graph generators. `http://www.cc.gatech.edu/~kamesh/GTgraph` (2006)
22. Barabási, A.L.: Network databases. `http://www.nd.edu/~networks/resources.htm` (2007)
23. Bader, D., Madduri, K.: A graph-theoretic analysis of the human protein interaction network using multicore parallel algorithms. In: Proc. 6th Workshop on High Performance Computational Biology (HiCOMB 2007), Long Beach, CA (March 2007)
24. Davis, T.: University of Florida Sparse Matrix Collection. `http://www.cise.ufl.edu/research/sparse/matrices` (2007)
25. Batagelj, V., Mrvar, A.: PAJEK datasets. `http://www.vlado.fmf.uni-lj.si/pub/networks/data/` (2006)
26. Demetrescu, C., Goldberg, A., Johnson, D.: 9th DIMACS implementation challenge – Shortest Paths. `http://www.dis.uniroma1.it/~challenge9/` (2006)
27. Erdős, P., Rényi, A.: On random graphs I. Publicationes Mathematicae **6** (290–297) 1959
28. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439) (1999) 509–512
29. Peri, S., *et al.*: Development of human protein reference database as an initial platform for approaching systems biology in humans. Genome Research **13** (2003) 2363–2371