Chapter 11
Efficient Extraction of High-Betweenness Vertices from Heterogeneous Networks

Wen Haw Chong, Wei Shan Belinda Toh, and Loo Nin Teow

Abstract Centrality measures are crucial in quantifying the roles and positions of vertices in complex network analysis. An important and popular measure is betweenness centrality, which is computed based on the number of shortest paths that vertices fall on. However, betweenness is computationally expensive to derive, resulting in much research on efficient computation techniques. We note that in many applications, it is the set of vertices with high betweenness that is of key interest and that their betweenness rankings rather than the exact values is usually adequate for analysts to work with. Hence, we have developed a novel algorithm that efficiently returns the set of vertices with highest betweenness. The convergence criterion for our algorithm is based on the membership stability of the high-betweenness set. Through experiments on various artificial and real-world networks, we show that the algorithm is both efficient and accurate. From the experiments, we also demonstrated that the algorithm tends to perform better on networks with heterogeneous betweenness distributions.

11.1 Introduction

In the real world, a variety of networks exist and their statistical, mechanical and temporal properties are the subject of much research, known broadly as complex network analysis. Networks of interest include social groups, collaboration networks, computer networks, food webs, etc.

In complex networks, centrality measures play an important role in quantifying how vertices and edges are positioned within the network. Popular concepts include degree centrality, which measures the number of neighbors a vertex has; closeness centrality, which measures the proximity of a vertex to all other vertices in the

W.H. Chong (✉) · W.S.B. Toh · L.N. Teow
DSO National Laboratories, 20 Science Park Drive, Singapore, 118230, Singapore
e-mail: cwenhaw@dso.org.sg; tweishan@dso.org.sg; tloonin@dso.org.sg

T. Özyer et al. (eds.), *The Influence of Technology on Social Network Analysis and Mining*, Lecture Notes in Social Networks 6, DOI 10.1007/978-3-7091-1346-2_11,

network; betweenness centrality [1, 9], which is based on the number of shortest paths that a vertex or edge sits on as an intermediary; and eccentricity centrality, which measures the distance from a vertex to the vertex furthest from it in the network.

This paper focuses on betweenness centrality. Amongst the various centrality concepts, betweenness centrality is well known to be computationally demanding. Despite recent advances, betweenness computation for large networks remains expensive. The fastest known algorithm was developed by Brandes [5] and computes exact betweenness for unweighted graphs in $O(mn)$ time and weighted graphs in $O(mn + n^2 \log n)$ time, where n is the number of vertices and m is the number of edges. It is based on computing the Single Source Shortest Path (SSSP) from every vertex in the network. Partial sums are accumulated over these SSSPs to derive betweenness values. In the process, closeness and eccentricity can also be derived. For application on large networks, the algorithm can be easily parallelized by distributing the SSSP computations amongst multiple processors. Building on this algorithm, an approximation technique was developed in [6]. This computes SSSPs from a subset of vertices chosen through some selection scheme. Since the subset can be chosen to be much smaller than the network size, computation savings can be achieved at the expense of accuracy. The authors referred to the selected source vertices as pivots. Henceforth, we shall refer to the approximation technique in [6] as the pivot method.

In this paper, we specify that instead of computing the exact betweenness for all vertices in the network, we merely need to extract the k highest ranking vertices, where k can be any number smaller than n. This is effectively a simpler problem. In many practical applications, $k \ll n$. In addition, many such applications do not require the actual betweenness values for analysis purposes. Hence complete and exact derivation expends computational resources to solve a much harder problem than necessary. Typically, it is the set of vertices with high betweenness that is of interest. These vertices have various practical purposes, e.g. corresponding to key nodes or gateways in computer networks. They may also be crucial vertices with a large impact on average shortest path length in a network upon elimination. In the case of a social network, high-betweenness vertices may be interpreted as having a strong middleman role. In designing our algorithm, we walk the line advocated by Tarjan [18]: "the most efficient algorithms are generally those that compute exactly the information relevant to the problem situation". It is rare that vertices with low betweenness are of interest. In any case, simply selecting vertices with one neighbor, or whose neighbors induce a clique will already extract many such vertices. In addition, our algorithm will terminate even though the betweenness rankings of vertices below the top k may not have converged.

Theoretical error bounds for the convergence of both closeness and betweenness centralities were derived in [6] based on Hoeffding's theorem [11] (See Appendix 2). It has also been shown that the computation of betweenness is more difficult and unreliable than that of closeness. In their conclusion, the authors in [6] proposed an approach for extracting k vertices of highest closeness. This was investigated in [16]. The strategy is to first use the approximation technique to obtain k' vertices with highest (estimated) closeness, where $k' > k$. k' is chosen such that it

is guaranteed with a high probability that the k vertices (of highest exact closeness) is a subset of the k' vertices. Then, the exact closeness is computed for each of the k' vertices, to extract the final top-k vertices. To our knowledge, the problem of how to extract the highest ranking vertices in terms of betweenness has not been explored, nor has any strategy been proposed. Unfortunately, the framework in [16] is not extensible to extracting high-betweenness vertices due to the different nature of the metrics. While it is possible to obtain the exact closeness value of a single vertex via one SSSP computation with it as the root, it is necessary to compute multiple SSSPs for the betweenness of a single vertex. Another strategy is required for efficient extraction of high-betweenness vertices.

In Sect. 11.2, we formally describe betweenness centrality and the prior work that is directly relevant to this paper. We provide the basis for our algorithm design in Sect. 11.3, followed by a description of the algorithm itself. Section 11.4 presents our evaluation metrics and the various networks for experimentation. In Sect. 11.5, we compare our technique in terms of efficiency and accuracy against the exact algorithm and present the experimental results. In Sect. 11.6, we compare accuracies against another more recent competitive technique. We conclude in Sect. 11.7.

11.2 Preliminaries

Betweenness centrality is a metric based on the enumeration of shortest paths between vertex pairs in a network. Let σ_{st} denote the number of shortest paths between vertices s and t, and $\sigma_{st}(v)$ be the number that pass through the vertex v. Define the pair wise dependency as the fraction of shortest paths between s and t that pass through v:

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}} , \tag{11.1}$$

where $s \neq v \neq t$. The betweenness centrality of a vertex v accumulates the pair wise dependencies for all vertex pairs whose shortest paths pass through v:

$$\mathrm{BC}(v) = \sum_{st} \delta_{st}(v) . \tag{11.2}$$

Before describing the strategy for our algorithm, it is necessary to describe the workings of the exact algorithm by Brandes [5] and the pivot method [6].

11.2.1 Exact Algorithm

The one-sided dependency of a vertex s on another vertex v is derived by summation over the appropriate pair wise dependencies:

$$\delta_s(v) = \sum_{t} \delta_{st}(v) . \tag{11.3}$$

The betweennness of v can then be computed as

$$\mathrm{BC}(v) = \sum_{s} \delta_s(v) \ . \tag{11.4}$$

The algorithm is based on the following recursive relation for computing one-sided dependencies:

$$\delta_s(v) = \sum_{w:v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} \left(1 + \delta_s(w)\right) \ , \tag{11.5}$$

where $P_s(w)$ denotes the set of predecessors of a vertex w on the shortest path tree with s as the root. Essentially, an SSSP is computed for each vertex in the network, producing the shortest path tree from that vertex. For a given shortest path tree with s as the root, a backward pass then traverses through the vertices in order of non-increasing distance from s. During the traversal, the recursive relation in (11.5) is used to compute the dependencies of s on all other vertices in the tree. To compute the betweenness of any vertex, the appropriate dependencies arising from all SSSPs are summed up as in (11.4). For memory efficiency, a running sum can be maintained and updated for each vertex during the computations.

11.2.2 Pivot Method

The pivot method is very similar to the exact algorithm. The main difference is that instead of computing SSSPs for all vertices in the network, it selects a subset of vertices (pivots) to compute SSSPs and then extrapolates from the contributions of this smaller set to estimate betweenness values. Various pivot selection schemes have been tested [6] and it has been observed that random sampling of pivots consistently performs the best across different kinds of networks.

The efficiency and accuracy of the pivot method depends on the number of pivots used. Accuracy is largely monotonic with the number of pivots. Using fewer pivots leads to higher efficiency at the cost of accuracy. When all vertices in the network are used as pivots, the method is equivalent to the exact algorithm.

11.3 Proposed Algorithm

This section covers the basis for our algorithm design as well as its technical details.

11.3.1 Basis for Design

Through comprehensive experiments on various artificial and real-world networks [6], it was empirically shown that the following applies for the pivot method:

- Random selection of pivots is superior to more sophisticated selection schemes and performs consistently well on different kinds of networks;
- The inversion distance, i.e. the number of wrongly ordered pairs, decreases approximately monotonically with the number of pivots added.

The second point implies that as the number of pivots is increased, the ranking of each vertex approaches the true rank value. This observation forms the basis for our algorithm design. Given all the vertices in a network ranked in decreasing order of betweenness, we are only interested in the subset of k-highest ranked vertices. If a sufficient number of pivots is used to obtain the ranking, we expect that the membership of this subset would remain fairly consistent even if more pivots were added.

Formally, let S_k be the set of k vertices with highest exact betweenness. Denote $\hat{S}_k$ as the corresponding approximate set returned from the pivot method. Clearly, $\hat{S}_k$ is cheaper to compute than S_k. If the approximation is good, there is a large overlap between $\hat{S}_k$ and S_k. For discussion purpose, we introduce a subscript denoting the number of pivots such that $\hat{S}_{k,r}$ means that the estimated set is returned from the pivot method with r pivots. If q pivots are chosen in another trial of the pivot method, where $q > r$, it can be shown largely that $|S_k \cap \hat{S}_{k,q}| > |S_k \cap \hat{S}_{k,r}|$. Intuitively, this means that using a larger number of pivots leads to a more accurate approximation of S_k. Now consider the case where r is sufficiently large such that $|S_k \cap \hat{S}_{k,r}| = k$, i.e. the approximation is perfect. Note that r may be less than or equal to the network size n. Obviously, $r \ll n$ is desired and the smaller r is, the greater the computational savings. Given $r < n$, a larger set of pivots of size q can be selected such that $|S_k \cap \hat{S}_{k,q}| = |S_k \cap \hat{S}_{k,r}| = k$, i.e. $\hat{S}_{k,q} = \hat{S}_{k,r} = S_k$. Although we are not able to compute $|S_k \cap \hat{S}_{k,q}|$ and $|S_k \cap \hat{S}_{k,r}|$ since we do not have the ground truth S_k, we can directly and easily compare $\hat{S}_{k,q}$ and $\hat{S}_{k,r}$. This naturally leads to the design of an iterative algorithm where pivots are incrementally added to re-estimate $\hat{S}_k$, and convergence is formulated based on its membership stability across iterations. In accordance with the first observation, we use the random selection scheme for pivots. Next we describe the algorithm in detail.

11.3.2 Detailed Description

Our algorithm is essentially a modification of the pivot method. Instead of specifying an overall number of pivots at initialization, pivots are added incrementally in batches. At each iteration, we compute the SSSPs for the new batch of pivots, update all vertices' approximate betweenness values and extract the highest ranked set. Batches are made unique such that previously added pivots will not be re-added. Also note that normalization of the betweenness values is optional and does not affect the ranking. Let $\hat{S}_k(t)$ be the set of k highest ranked vertices extracted at iteration t. We compare it with the corresponding set extracted during previous iterations. If the membership of the sets remains largely stable according to some predefined criteria, the algorithm terminates.

Input: Network of vertices
Parameter: `max_pivots`
1: $t = 0$ //iteration step
2: `num_pivots` $= 0$ //number of pivots used so far
3: `counter` $= 0$ //termination counter
4: Order all vertices randomly into `pivot_set`(t)
5: **while** `num_pivots` $<$ `max_pivots`
6: `pivot_batch` $=$ `pivot_set`(t) //`pivot_set` for iteration t
7: `num_pivots` $=$ `num_pivots` $+ \Delta p$ //size of `pivot_batch`
8: Compute SSSPs for `pivot_batch` to update betweenness values for all vertices
9: Extract the k highest ranking vertices into $\hat{S}_k(t)$
10: **if** $\hat{S}_k(t) = \hat{S}_k(t-1)$ //current set is same as previous
11: `counter` $=$ `counter` $+ 1$ //increment
12: **else**
13: `counter` $= 0$ //reset
14: **end if**
15: **if** `counter` $= 3$ //same set for 3 consecutive iterations
16: Terminate with result set $\hat{S}_k = \hat{S}_k(t)$
17: **end if**
18: $t = t + 1$
19: **end while**
Output: Set of k highest betweenness vertices, $\hat{S}_k$

Fig. 11.1 Proposed algorithm

The pseudo code for the algorithm is shown in Fig. 11.1. The final set of k highest betweenness vertices is saved in $\hat{S}_k$. The parameter `max_pivots` specifies the maximum number of pivots to be used in the event that convergence does not occur. It can be set to less than or equal to the network size. Lines 1–3 correspond to initialization. In line 4, the `pivot_set` is created by randomly ordering all vertices in the network such that consecutive batches of pivots can be easily added in iterations. This is in accordance with the random pivot selection scheme. Line 5 starts the algorithm iterations. At the start of each iteration t, we take a new batch of Δp pivots from `pivot_set` (lines 6–7). The value of Δp determines the amount of additional computations that is done in an iteration. A large value means that the algorithm takes larger steps across iterations, but achieves coarser resolution. We have arbitrarily set Δp to 10 in our experiments. Following the method based on (11.5), SSSPs are computed over the newly added pivots to obtain the latest betweenness approximations (line 8). The current set of k highest ranking vertices $\hat{S}_k(t)$ is then extracted (line 9). This can be done via sorting or by any efficient order statistics algorithm. In lines 10–17, we evaluate the convergence criteria. This is based on the membership stability of $\hat{S}_k(t)$ across consecutive iterations. If the membership of this set remains unchanged over some number of consecutive iterations (we have used three in our experiments), we deem convergence to have occurred and terminate the algorithm. The speedup of the algorithm is directly proportional to the number of pivots used. The running time is $O(mp)$, where $p = \sum_t \Delta p$, i.e. the total number of pivots added over all iterations. We also point out that the

convergence criteria can be tweaked in many ways to achieve a tradeoff between efficiency and accuracy. A much stricter criterion would be to terminate only if the absolute order of vertex rankings in the sets are identical over multiple iterations. Another adjustment would be to increase the number of iterations for which the members of $\hat{S}_k(t)$ must remain unchanged. In our experiments, we have relaxed the convergence criterion somewhat to achieve greater computational efficiency. This has proven adequate to achieve good accuracy. One can think of special cases where the convergence criterion will not be met. For example, consider a network of vertices connected in a ring. All vertices have the same betweenness values and the membership of $\hat{S}_k(t)$ be unstable across iterations. However such contrived scenarios are generally rare in real networks. We deem it fruitful to investigate the performance of our algorithm on popular network models and real-world data.

11.4 Experiments

We describe our experiments, starting with the evaluation metrics and networks used, followed by an analysis of the results.

11.4.1 Evaluation Metrics

We term vertices in S_k as relevant vertices. Our estimated set $\hat{S}_k$ is also of size k. Accuracy is evaluated using precision, which measures the proportion of relevant vertices returned in $\hat{S}_k$:

$$\text{prec} = \frac{|\hat{S}_k \cap S_k|}{k} \,. \tag{11.6}$$

If all vertices in $\hat{S}_k$ correspond exactly to those in S_k, precision is 1. Where precision is less than 1, $\hat{S}_k$ contains irrelevant vertices. In such a case, it is desired that such vertices are ranked low in $\hat{S}_k$. We evaluate this using average precision, which is commonly used in document retrieval tasks and emphasizes returning relevant documents earlier. In our context, vertices from S_k are relevant and should be returned earlier, i.e. ranked higher. Average precision can be computed as

$$\text{ave\_prec} = \sum_{i=1}^{k} \text{prec}(i) \times \Delta\text{rec}(i) \,, \tag{11.7}$$

where $\text{prec}(i)$ is the precision at a cut-off rank position i of $\hat{S}_k$, i.e. $\text{prec}(i) = |\hat{S}_i \cap S_k|/i$. $\Delta\text{rec}(i)$ is the change in recall from position $i-1$ to i whereby recall measures the proportion of relevant vertices included in the result set, i.e. $\text{rec}(i) = |\hat{S}_i \cap S_k|/k$. Note that average precision is upper bounded by precision at position k.

Using the two measures above we only evaluate the extent that relevant vertices are included in $\hat{S}_k$ and if they are ranked above any irrelevant vertices that are also included. We do not directly compare the rankings of $\hat{S}_k$ with S_k using any rank correlation metrics, since $\hat{S}_k$ is not a closed list guaranteed to have the same set of members as S_k. The inversion distance is also not very meaningful since pairs of vertices can be ordered correctly in $\hat{S}_k$ even though one or both vertices in each pair may be irrelevant.

11.4.2 Heterogeneity Measure

We have implemented the entropy-based measure to quantify the heterogeneity of the actual betweenness distributions for our various tested networks. The motivation here is to check for any correlation between the performance of the algorithm and the heterogeneity of the underlying betweenness distribution. The entropy-based measure has previously been advocated by Wu et al. [20] to quantify the heterogeneity of degree distributions. To derive the measure for betweenness distribution, first construct a histogram of betweenness values. The histogram bins are uniform in width and cover the entire range of betweenness values from the distribution being evaluated. Let b_i be the probability that a randomly chosen vertex will have a betweenness value that is covered by the ith bin. The entropy-based measure H is then defined as

$$H = -\sum_{i=1}^{B} b_i \ln b_i \ . \tag{11.8}$$

Small H values are indicative of heterogeneous betweenness distribution. The maximum value of H occurs when $b_i = 1/B$ for all bins, which corresponds to a uniform distribution. In our experiments, we have used 1,000 bins across all networks for computing H.

11.4.3 Networks

We apply our algorithm on various artificial and real-world networks. For evaluation, it is necessary to compute exact betweenness and derive S_k for each network. This limits our experiments to networks of small to medium sizes. For all networks, we process them as undirected and unweighted networks. Loops and multiple edges between vertex pairs are excluded.

We examine three popular types of artificial networks. Each network generated is specified to have exactly 1,000 vertices and roughly 10,000 edges. The actual edge

count depends on the generation scheme specific to each network type. Efficient generation schemes are described in [3]. The network types are:

Random Such networks [10] are defined by the edge probability θ, where $0 < \theta < 1$. For each pair of vertices, an edge is independently formed with probability θ. In our experiments, we use $\theta = 0.02$ such that each random network has around 10,000 edges.

Small world (SW) This model was introduced by Watts and Strogatz [19]. The model starts with an initial ring of n vertices, with each vertex connected to its nearest 2d neighbors. Depending on the generation variant used, shortcut edges are then obtained by randomly and independently rewiring existing edges or by adding new random edges [15]. We use the latter. Five thousand random edges are added to the initial ring in which each vertex is connected to its nearest ten neighbors, i.e. $d = 5$.

Preferential Attachment (Pref. A.) Barabási and Albert [2] formulated a model for generating networks with heavy tailed degree distributions. In this model, vertices are added one at a time. The newly added vertex connects a fixed number of edges to existing vertices with probability proportional to the degree of the latter. We implement the model of Bollobás et al. [4].

In addition to artificial networks, we select several real-world networks of various origins and sizes for our experiments. We feel that this is a good representation of the complex networks popularly analyzed in the literature. The selected networks are:

Protein This is the protein interaction network for yeast. The data originates from Jeong et al. [12]. Each vertex corresponds to a protein while the edges represent protein-protein interactions.

Enron Enron emails [17] are used to construct a network of email users. Email users correspond to vertices. Two users are linked if they have communicated by email at least once.

Ticker Following the September 11th terrorist attacks, Corman et al. [8] compiled and transformed Reuters ticker news articles into a network text representation. Words appearing in noun phrases are represented as vertices. Words appearing together in the same noun phrase or consecutively within a sentence are linked via edges.

Internet Autonomous Systems (AS) An AS is effectively a set of routers under a single administration. Routing in a network of ASes is coordinated by the Border Gateway Protocol. There are several types of ASes, e.g. ISPs, end-users; and relationships between ASes, e.g. provider-customer. AS networks can be obtained from [7]. We have used a sample network from 1 January 2007.

DBLP DBLP [13] is an on-line database of computer science publications with the authorship and publication details of hundreds of thousands of articles. The records

Table 11.1 Vertex and edge counts for the selected real-world networks

	Protein	Enron	Ticker	AS	DBLP
Vertices	1,458	5,312	13,308	24,013	75,207
Edges	1,993	15,578	148,034	49,332	202,291

span many years, with information in the recent years being more complete. We have constructed a collaboration network for the year 2006. Authors are represented as vertices while edges are formed between all authors who have co-authored publication(s). For each real world network, we extract the largest connected component to work on. This ensures that all pivots contribute properly to betweenness computation and that none fall in isolated small components. Generally, when processing a network with multiple components, the proper procedure is to extract the components and process each individually. Table 11.1 shows the number of vertices and edges corresponding to the largest component of each network.

11.5 Results

11.5.1 Artificial Networks

Two scenarios are considered: extraction of the 10 and 20 highest ranked vertices in terms of betweenness. We specify `max_pivots` to be 1,000 and add pivots in batches of 10. For each scenario in each artificial network type, we conduct 20 trials whereby each trial consists of a realization of the network type followed by the application of the algorithm. Results are averaged over the 20 trials to obtain the mean precision (MP) and mean average precision (MAP), shown in Table 11.2.

The best performance is observed on the preferential attachment networks, where the MP is higher than 0.9 for both scenarios with almost equally high MAP. It is remarkable that these results are achieved using relatively few pivots. For example, to extract the 10 highest ranked vertices of preferential attachment networks, the algorithm requires just an average of 11 % of the vertices as pivots. This is well below the network size of 1,000 and implies substantial savings in computation time.

The algorithm performs less impressively on the small world and random networks, although accuracy and computational savings are still reasonable. Performance on these two types of networks is very similar. We note that as k is increased from 10 to 20, MP and MAP improvement is in the range of 10–20 %. However, this is at a cost of using many more pivots. This can be explained as follows: with larger k for these two network types, it becomes more difficult for the membership of $\hat{S}_k$ to stabilize early. Convergence is only achieved at larger iterations when more pivots have been added. The larger number of pivots in turn results in higher accuracy. Compared to the preferential attachment networks, both network types require many more pivots for the algorithm to converge: more than three times as many when $k = 10$ and more than four times when $k = 20$.

Table 11.2 Results for each scenario ($k = 10, 20$) in each artificial network type, with standard deviations enclosed in brackets. The best results for each k are in bold

k	Network type	% of vertices used as pivots	MP	MAP
10	Random	38.9 (17.8)	0.71 (0.13)	0.64 (0.16)
	SW	38.4 (15.9)	0.68 (0.16)	0.62 (0.19)
	Pref. A.	**11.0 (0.041)**	**0.91 (0.07)**	**0.91 (0.07)**
20	Random	65.5 (19.9)	0.81 (0.12)	0.79 (0.14)
	SW	69.7 (20.5)	0.85 (0.07)	0.82 (0.09)
	Pref. A.	**16.1 (0.069)**	**0.92 (0.04)**	**0.91 (0.05)**

For further analysis, we investigate how the betweenness distributions of the networks affect the difficulty of the extraction task. Figure 11.2 shows the histograms of true betweenness values from three sample networks, one from each network type. For numerical comparison of heterogeneity, the average H values of the true betweenness distribution over each network type are shown in Table 11.3.

For better visual comparison, the top 20 betweenness values for the preferential attachment network have been excluded in its histogram. The distributions of the random and small world networks appear fairly similar and Gaussian-like, while that of the preferential attachment network is heavily skewed with a tiny portion of vertices taking on very high betweenness values. Compared to the first two networks, the range of betweenness values is much more extreme (by comparing the horizontal scales). Analogous to the fact that extreme outliers are more easily detected in outlier detection problems, the extremity of the high-betweenness vertices results in them being extracted more readily by the algorithm. We can also arrive at the same insight by considering the earlier example of a network comprising of a ring of vertices. In such a network, all vertices have the same betweenness, i.e. a uniform betweenness distribution. It is clear that even with the maximum number of iterations of the algorithm, i.e. using all vertices in the network as pivots, higher betweenness vertices will not be surfaced since there are no extremities.

Table 11.3 provides a numerical quantification for Fig. 11.2. H values are significantly smaller for preferential attachment networks, indicating their betweenness distributions are much more heterogeneous than random and small world networks. The current results are intuitive. A heterogeneous distribution may have more extreme values, which leads to greater ease of extraction. However note that H is computed over the entire distribution, rather than just over the large betweenness values of interest. Hence extremity and heterogeneity does not have a perfect correlation.

Finally, we have ruled out the power-law distribution for the betweenness values of the preferential attachment network after plotting the cumulative distribution function on doubly logarithmic axes. The distribution type is left for future investigation.

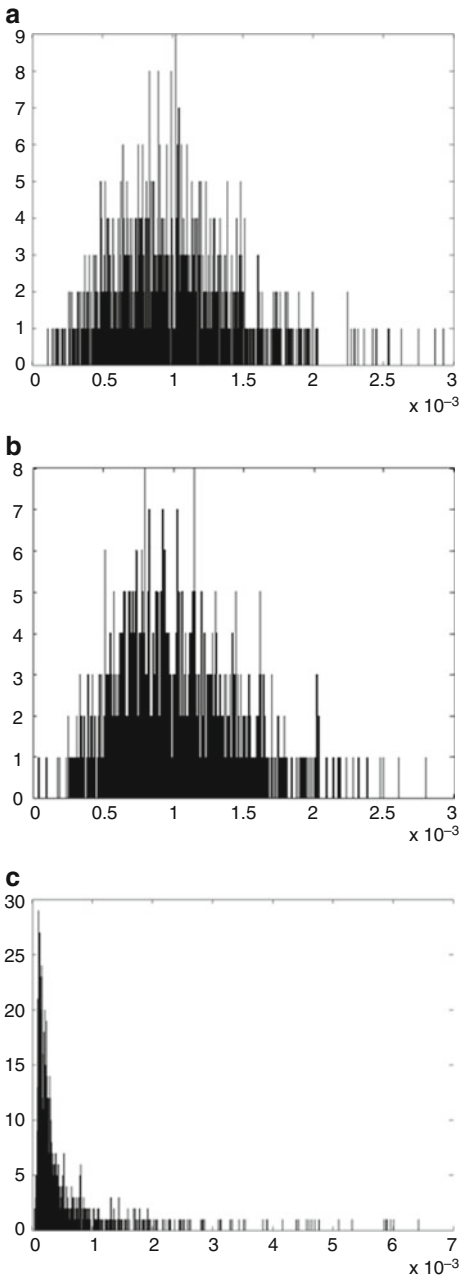

Fig. 11.2 Histograms of betweenness values for each artificial network type; that of the preferential attachment network appears significantly different. Networks: (**a**) Random, (**b**) Small world and (**c**) Preferential attachment

Table 11.3 Entropy-based heterogeneity values for each artificial network type (averaged over 20 trials) with standard deviations enclosed in brackets

Network type	H
Random	5.91 (0.067)
SW	5.93 (0.066)
Pref. A.	2.12 (0.13)

Table 11.4 Results for each scenario ($k = 10, 20$) in each real-world network, with standard deviations enclosed in brackets. The best results for each k are in bold

k	Network	% of vertices used as pivots	MP	MAP
10	Protein	6.79 (2.26)	0.96 (0.07)	0.96 (0.07)
	Enron	**1.62 (0.45)**	**0.98 (0.04)**	**0.97 (0.05)**
	Ticker	1.32 (0.34)	0.81 (0.10)	0.78 (0.13)
	AS	0.52 (0.32)	0.90 (0.05)	0.90 (0.05)
	DBLP	0.23 (7.31E − 4)	0.73 (0.12)	0.69 (0.14)
20	Protein	15.43 (4.73)	0.90 (0.07)	0.90 (0.07)
	Enron	**2.56 (0.77)**	**0.92 (0.03)**	**0.92 (0.04)**
	Ticker	1.71 (0.59)	0.86 (0.05)	0.85 (0.06)
	AS	0.67 (0.20)	0.92 (0.03)	0.91 (0.04)
	DBLP	0.34 (9.57e − 4)	0.83 (0.07)	0.80 (0.09)

11.5.2 Real-World Networks

We consider the same scenarios of extracting the top 10 and 20 vertices. For each network, ten trials of the algorithm are conducted whereby trials differ due to the random selection of pivots. Algorithm parameters are identical to those used for the artificial networks. Results are averaged over the ten trials and shown in Table 11.4. Excellent accuracies are obtained, with both MP and MAP above 0.8 for all network scenarios except the co-authorships network with $k = 10$. However, in this scenario the mean percentage of vertices used as pivots is extremely low compared to the network size, i.e. 0.23 %. More pivots can certainly be used to improve accuracy (this can be done by adjusting the convergence criterion). To illustrate this, we conduct separate experiments and plot the MP and MAP per iteration as the number of pivots is increased to 500. Figure 11.3 shows the plots for the DBLP co-authorships and Ticker news networks. All other networks exhibit fairly similar plots. The MP and MAP scores improve approximately monotonically as the number of pivots is increased. The approximate monotonicity is reflected in the kinks of the curves. The rate of improvement slows for larger number of pivots. At 500 pivots (0.66 % of network size), MP of 0.86 is achieved for the co-authorships network.

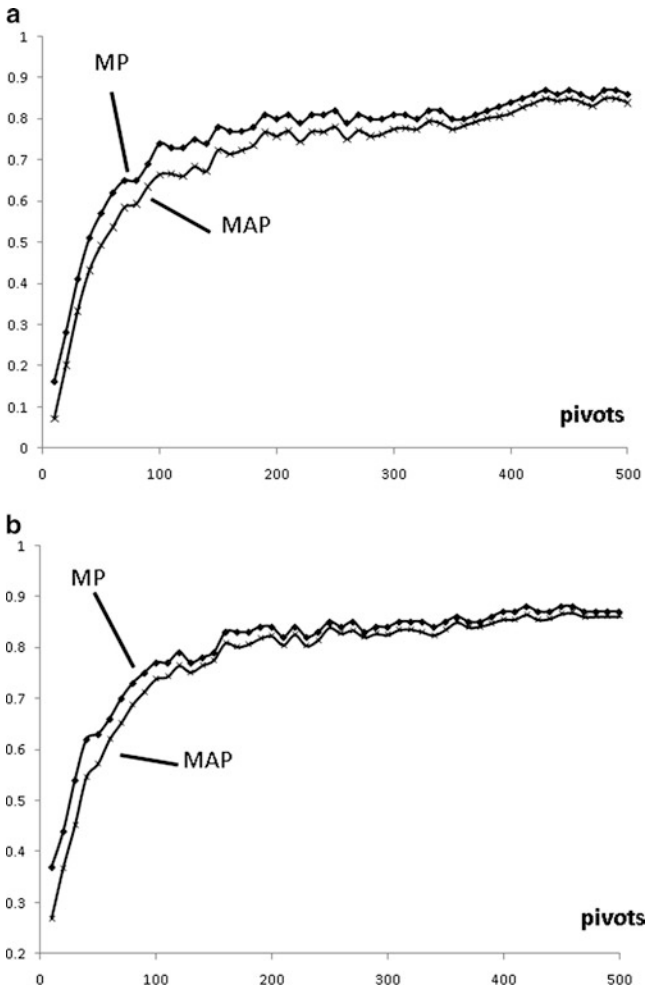

Fig. 11.3 MP/MAP of two real-world networks with increasing pivots, where $k = 10$. Each point is the respective MP/MAP score over ten trials. Networks: (**a**) DBLP and (**b**) Ticker

Returning to Table 11.4, it can be seen that for the protein interaction, Enron email and internet AS networks, the algorithm achieves MP and MAP scores of 0.9 and above. Across the board, these impressive results have been achieved at very low pivot counts relative to network size.

Similar to the case for the artificial networks, the number of pivots required for convergence increases at higher k, but the increase is generally less than linear, except for the protein interaction network. Further research can be conducted to explore the relationship between k and the number of pivots required for networks with different characteristics.

Table 11.5 Entropy-based heterogeneity values for each real-world network

Network type	H
Protein	2.57
Enron	1.26
Ticker	0.86
AS	0.38
DBLP	1.20

Table 11.5 indicates that the real world networks have various degrees of heterogeneity in their true betweenness distributions, although all H values are fairly low. Comparing Tables 11.4 and 11.5, there does not seem to be any obvious correlation between accuracies, efficiencies and heterogeneity measures. For example, while the protein network appears to have the least heterogeneous distribution, the algorithm still achieves good accuracies on it. However we note that the proportion of vertices required as pivots is the largest amongst all networks, with more than four times the proportion as the nearest competitor, the Enron network. Hence we are apt to conclude that the relative lack of heterogeneity is being compensated for with the use of more vertices.

11.6 Further Comparison

In previous experiments on each network, we have tabulated the percentage of vertices used as pivots by our algorithm. Implicitly, comparisons of both efficiency and accuracy are being made with the exact algorithm (akin to a baseline) described in Sect. 11.2.1. The baseline uses all vertices as pivots and achieves perfect accuracies at a high cost. For example, extracting the top 10 vertices from the Enron network requires just 1.62 % of vertices as pivots, and implies an efficiency improvement of more than 98 % over the baseline. This comes at a trade off of a 2 % drop in MP.

In this section, we conduct further experiments with real-world networks for explicit comparisons against a recent competitive method [14], which we term the sub-graph technique. This samples a large network greedily such that the extracted sub-graph contains the set of nodes with high betweennness and other centralities. To extract the high betweenness nodes, the exact betweenness computation algorithm is then applied on the sub-graph.

Denote n_g as the number of vertices and m_g as the number of edges in the sub-graph. For undirected networks, extracting S_k with the sub-graph technique then has a computational complexity of $O(g + n_g m_g)$ where the first term g represents the cost of sub-graph construction. Larger sub-graphs have higher probabilities of containing S_k and hence higher accuracies.

In this experiment, we compare the extraction accuracies given the same computation cost for our proposed and the sub-graph techniques. This is done by fixing the sub-graph size for each real world network such that the computational

Table 11.6 Average sub-graph sizes used

k	Network	n_g	m_g
10	Protein	403.1 (0.74)	489.9 (0.74)
	Enron	579 (0)	2,341.9 (2.85)
	Ticker	838 (0)	31,305.7 (27.77)
	AS	1,152 (0)	5,550.6 (0.7)
	DBLP	5,114 (0)	7,074.8 (3.71)
20	Protein	618.2 (0.42)	728.4 (1.26)
	Enron	780 (0)	2,719.5 (0.97)
	Ticker	998 (0)	33,993.4 (10.94)
	AS	1,344 (0)	6,077.7 (0.48)
	DBLP	6,091 (0)	8,510 (3.33)

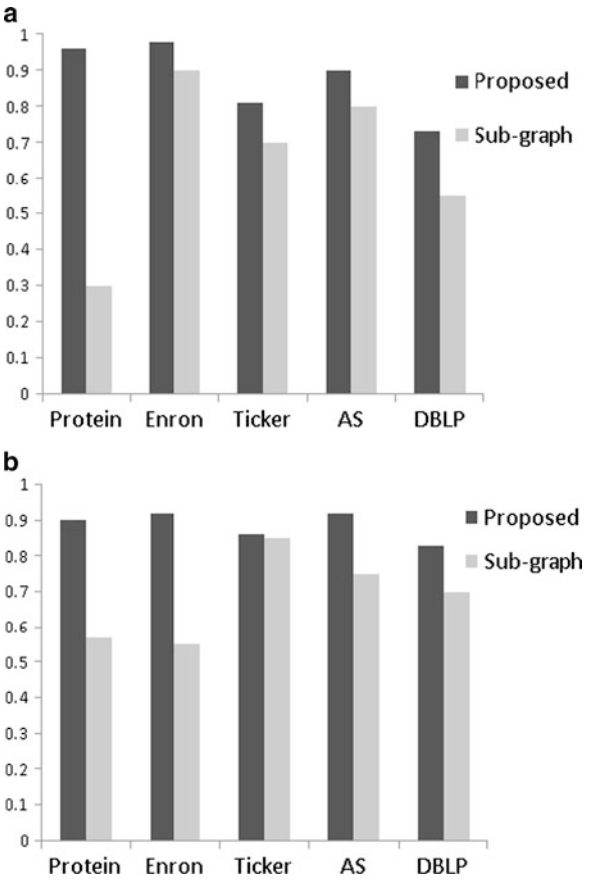

Fig. 11.4 Mean Precision (MP) over ten trials of our proposed and the sub-graph techniques given equal computation costs, for (**a**) $k = 10$ and (**b**) $k = 20$

cost equals that incurred by our algorithm in Sect. 11.5.2, i.e. $O(n_g m_g) = O(mp)$. We have ignored $O(g)$ which is small and hence given a slight advantage to the sub-graph technique. The average dimensions of the sub-graphs used are shown in Table 11.6 in Appendix 2. Figure 11.4 compares the mean precision over ten

Table 11.7 Detailed accuracy results for the sub-graph technique

k	Network	MP	MAP
10	Protein	0.3 (0)	0.3 (0)
	Enron	0.9 (0)	0.9 (0)
	Ticker	0.7 (0)	0.68 (2.3E − 3)
	AS	0.8 (0)	0.78 (0)
	DBLP	0.55 (0.05)	0.53 (0.03)
20	Protein	0.57 (0.05)	0.4 (0.04)
	Enron	0.55 (0)	0.4 (2.6E − 3)
	Ticker	0.85 (0)	0.79 (2.8E − 3)
	AS	0.75 (0)	0.74 (0)
	DBLP	0.7 (0)	0.58 (6.7E − 3)

trials for both techniques on each real world network. Detailed numerical results are shown in Table 11.7 in Appendix 2.

Figure 11.4 shows that our proposed technique consistently outperforms the sub-graph technique for each real-world network. We are apt to conclude that given the same computation cost, the former extracts S_k with higher precision. In particular, the difference in MP is largest for the protein network. For $k = 10$ on this network, the sub-graph technique performs poorly. Performance may have been impacted by the fact that the protein network is a much less heterogeneous network, as compared with the other networks.

Lastly, in terms of inclusion of high betweenness vertices, the sub-graph technique has been shown previously [14] to outperform sampling schemes based on depth-first search, breadth-first search and random walk. Hence by extension, our proposed technique outperforms these other techniques as well in terms of accuracy.

11.7 Conclusion

Betweenness computation is a notoriously expensive problem. Prior to the current work, it was not clear how the highest betweenness vertices of complex networks could be extracted accurately and efficiently in a systematic manner. Much work in the literature has focused on the efficient computation of betweenness values for vertices.

In this paper, we have developed a novel algorithm to accomplish the mentioned extraction task, using well known observations of the pivot method. The algorithm performs with excellent results on preferential attachment networks and various real-world networks. It performs less well, but still achieves reasonable results, on random and small world networks.

At the next stage, it is useful to investigate in detail the relationship between precision, number of pivots and the convergence criterion. The objective will be to provide hints to the potential user about the expected runtime and pivots required to obtain some required level of precision. This will enhance the utility of the algorithm in real applications.

Appendix 1: Hoeffding's Theorem

For independently identically distributed random variables $X_1, \ldots X_k$, where $0 \leq X_i \leq M$ for $i = 1 \ldots k$, and an arbitrary $\xi \geq 0$, the following probability bound applies:

$$\Pr\left(\left|\bar{X} - \mathrm{E}\left[\bar{X}\right]\right| \geq \xi\right) \leq \exp\left(-2k\left(\frac{\xi}{M}\right)^2\right), \tag{11.9}$$

where $\bar{X} = (X_1 + \ldots + X_k)/k$ and $\mathrm{E}\left[\bar{X}\right]$ is the expected value of $\bar{X}$.

Appendix 2: Detailed Results for Sect. 12.6

Table 11.6 shows the average number of vertices and edges (over ten trials) contained in the sampled sub-graph for each real-world network. Applying the exact algorithm on the sub-graph will incur a computation cost roughly equivalent to that incurred by our algorithm in Table 11.4. Standard deviations are enclosed in brackets. For each network, the required sub-graph dimensions increase with k as is intuitively expected.

Table 11.7 displays detailed MP and MAP results for the sub-graph technique. For each real-world network, ten trials are conducted. Standard deviations are enclosed in brackets.

References

1. Anthonisse, J.: The rush in a directed graph. Technical Reports BN 9/71, Stichting Mathematisch Centrum (1971)
2. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
3. Batagelj, V., Brandes, U.: Efficient generation of large random networks. Phys. Rev. E **71**(3), 036113 (2005)
4. Bollobás, B., Riordan, O., Spencer, J., Tusnády, G.: The degree sequence of a scale-free random graph process. RSA Random Struct. Algorithms **18**, 279–290 (2001)
5. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Sociol. **2**(25), 163–177 (2001)
6. Brandes, U., Pich, C.: Centrality estimation in large networks. Int. J. Bifurc. Chaos **7**(17), 2303–2318 (2007)
7. CAIDA: The CAIDA AS relationship dataset. http://www.caida.org/data/active/as-relationships (2007)
8. Corman, S.R., Kuhn, T., Mcphee, R.D., Dooley, K.J.: Studying complex discursive systems. Hum. Commun. Res. **28**(2), 157–206 (2002)
9. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35–41 (1977)
10. Gilbert, E.N.: Random graphs. Ann. Math. Stat. **30**, 1141–1144 (1959)

11. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **301**(58), 713–721 (1963)
12. Jeong, H., Mason, S.P., Barabasi, A.L., Oltvai, Z.N.: Lethality and centrality in protein networks. Nature **411**, 41–42 (2001)
13. Ley, M.: The DBLP computer science bibliography. http://www.informatik.uni-trier.de/~ley/db
14. Maiya, A.S., Berger-Wolf, T.Y.: Online sampling of high centrality individuals in social networks. In: Zaki, M.J., Yu, J.X., Ravindran, B. Pudi, V. (eds.) Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad. Proceedings. Part I, Lecture Notes in Computer Science, vol. 6118, pp. 91–98. Springer, Berlin (2010)
15. Newman, M.E.J., Watts, D.J.: Renormalization group analysis of the small-world network model. Phys. Lett. A **263**, 341–346 (1999)
16. Okamoto, K., Chen, W., yang Li, X.: Ranking of closeness centrality for large-scale social networks. In: Proceedings of the 2nd Annual International Workshop on Frontiers in Algorithmics, 2008, pp. 186–195. Springer-Verlag Berlin, Heidelberg (2008)
17. Shetty, J., Adibi, J.: Enron Dataset. http://www.isi.edu/~adibi/Enron/Enron.htm (2004)
18. Tarjan, R.E.: Data Structures and Network Algorithms. Society for Industrial and Applied Mathematics, Philadelphia (1983)
19. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**, 440–442 (1998)
20. Wu, J., Tan, Y.J., Deng, H.Z., Zhu, D.Z.: A new measure of heterogeneity of complex networks based on degree sequence. In: International Conference on Complex Systems. Springer, Berlin/New York (2006)