Procedia Computer Science

Volume 66, 2015, Pages 83–92

YSC 2015. 4th International Young Scientists Conference on Computational Science

Heuristic Algorithm for Approximation Betweenness Centrality Using Graph Coarsening

Mikhail Chernoskutov[1,2], Yves Ineichen[3], and Costas Bekas[3]

[1] Institute of Mathematics and Mechanics UB RAS, Yekaterinburg, Russia
[2] Ural Federal University, Yekaterinburg, Russia
[3] IBM Research, Zürich, Switzerland

Abstract

Nowadays, graph analytics are widely used in many research fields and applications. One important analytic that measures the influence of each vertex on flows through the network, is the betweenness centrality. It is used to analyze real-world networks like for example social networks and networks in computational biology. Unfortunately this centrality metric is rather expensive to compute and there is a number of studies devoted to approximate it. Here we focus on approximating the computation of betweenness centrality for dynamically changing graphs. We present a novel approach based on graph coarsening for approximating values of betweenness centrality, when new edges are inserted. Unlike other approaches, we reduce the cost (but not complexity) of the betweenness centrality computation step by working on a coarser graph. Our approach demonstrates more than 60% speedup compared to the exact recalculation of the betweenness centrality for dynamically changing graphs.

Keywords: graph algorithms, betweenness centrality, approximate computation

1 Introduction

Graph algorithms appear in many fields of science, industry and business, such as bioinformatics, social networks analysis, knowledge discovery, and many more. Nowadays, graph analytics are a typical "data intensive" task. Dealing with big real-world graphs poses many difficulties, such as irregular memory access patterns and workload imbalance because of the skewed degree distribution of vertices. Another important feature is that many useful graph algorithms have a rather high computational complexity. Betweenness centrality allows us to rank the vertices V in a graph by assigning some value to every each vertex. This metric originates from works of Bavelas, Sabidussi and Freeman [2, 13, 8].

Selection and peer-review under responsibility of the Scientific Programme Committee of YSC 2015

doi:10.1016/j.procs.2015.11.011

Betweenness centrality based on shortest-paths enumeration for some individual vertex v and defined as follows:

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

where σ_{st} denotes the number of shortest paths between vertices s and t, and $\sigma_{st}(v)$ is number of shortest paths which pass through v. From the definition we see that the betweenness centrality value of a vertex is relative to the number of shortest paths passing through the vertex. A vertex with high betweenness centrality is passed in many (shortest) paths and is therefore important for the flow in the network.

Betweenness centrality has $O(V^3)$ complexity (where V is number vertices) for unweighted graphs. In [3], Brandes introduced an algorithm with $O(VE)$ complexity. To the best of our knowledge this is currently the fastest sequential algorithm for exact computation of this centrality metric.

But how can we deal with big graphs consisting of millions of vertices and edges? Exact computations of betweenness centrality can take a lot of time even for Brandes algorithm. The obvious solution in this case is try to approximate all centrality computations. This trade off between accuracy and performance is very useful for real-world tasks.

Another important challenge, that we address in this paper, are graphs that change dynamically over time. Obliviously recomputing the centrality metric every time the graph is altered (addition or deletion of new vertex or edge) can get too expensive. In most cases, the changes in the values of the metric are relatively small. This property tempts to compute an approximation of the update reusing already computed values. In this study we present novel approach to approximate the computation of the betweenness centrality metric using graph coarsening. With the right coarsening strategy we can operate on a "similar" (in terms of centrality) but smaller graphs. The main idea is to condense the initial graph and subsequently assess changes in structure (adding or removing edges and vertices) on the "small" graph. Thanks to the smallness of the graph recomputing the centrality metric will be considerably cheaper. Finally the updates are extrapolate back to the initial graph.

This paper organized as follows. In Section 2 we briefly explore already existing studies, devoted to approximate computation of betweenness centrality metric for static and dynamically changing graphs. In Section 3 we describe our approach in detail and present the algorithm. Section 4 is devoted to describe the benchmarks and datasets which we used for testing the algorithm. Discussion about the performance with respect to accuracy and speedup are presented in Section 5. This paper ends with a conclusion in Section 6.

2 Related Works

There is a number of related works devoted to develop methods and approaches of approximate computation of the betweenness centrality metric. The first attempts by Eppstein et al. [6] traded time to solution for accuracy. They proposed sampling technique to approximate closeness centrality (which is also based on counting shortest paths). The high-level overview of this technique is summarized below:

1. Pick some vertex v_i in the graph and solve the single-source shortest path (SSSP) problem with v_i as the source;

2. Estimate the value of the centrality based on the previous computations of the SSSP problem;

3. Repeat procedure k times (until it converges).

In [4] Brandes et al. augmented the sampling technique of Eppstein et al. to approximate the betweenness centrality.

Bader et al. in [1] present adaptive sampling technique. It is termed adaptive, because the number of vertices used to solve the SSSP problem varies with the information obtained from each iteration of the algorithm. Extensive experimental study on real-world graph instances are presented in this study and showed good quality of approximation.

Two important improvements are proposed in [9] to speed up the computation of betweenness centrality in dynamically changing networks. First, authors empirically observed that the vertex with the highest centrality value stays invariant over time. Therefore, there is no need to recompute values for it every time the graph changes. Second, instead of computing the "usual" betweenness centrality, they used the k-betweenness centrality. This metric considers only the shortest paths with lengths at most equal to k. The authors conclude that the proposed approximation technique works quite efficient and accurate for scale-free graphs, while for random graphs it is poor due to their structural properties.

In [10] an improvement of Brandes [4] sampling technique is presented. Compared to the uniformed sampling in Brandes algorithm, the authors developed a distance-based sampling approach, where samples to estimate the betweenness centrality are chosen based on the length of shortest paths from a randomly chosen vertex. One important advantage of this approach is that proposed framework can be adopted to various sampling techniques.

In [12] probabilistic sampling techniques are used to further improve Brandes sampling technique. To that end an efficient random-sampling-based algorithm is used to estimate the betweenness centrality of the top-K vertices in the graph with high probability.

All presented approaches concentrates on processing the entire graph. Most of them are able to reduce the number of SSSP problems to estimate the betweenness centrality. Even so, these particular SSSP computations are performed on the full graph. In contrast, we present a novel approach based on reduction the size of initial graph. In this particular case, we can speedup the computation of the betweenness centrality by reducing the cost of computing the SSSP problems.

3 Approximation Betweenness Centrality

Our approach of approximating betweenness centrality consist of two main steps:

1. Condense the initial graph to a "smaller" graph using a coarsening procedure;

2. Extrapolate and apply changes to the initial graph.

Below we describe both steps in details.

3.1 Graph Coarsening

The goal of this step is to compute a "small" but still representative view of the initial graph G_0. In order to compute a condensed graph G_1 we use the betweenness centrality values of G_0 to select the top-k central vertices (hubs). The actual coarsening step collapses neighbors of top-k vertices to the central vertex. This process is visualized in Figure 1.

The high-level pseudocode for the graph coarsening is given in Algorithm 1. In line 2 we fill the *hubs_list* identifying vertices in initial graph with highest centrality values (hubs). The

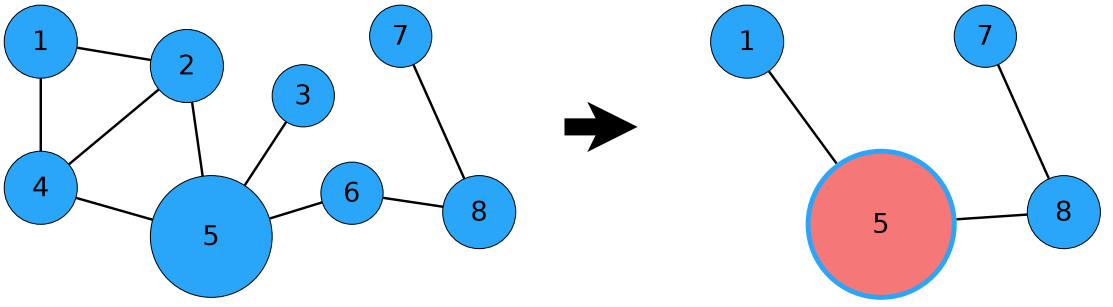

Figure 1: Initial graph before(left) and after (right) graph coarsening. Vertex 5 has highest betweenness centrality value, so it serve like a hub to adsorb its neighbors on it. Size of vertices corresponds to their centrality values

Input: G_0 – initial graph, *num_hubs* – number of vertices, which are considered as hubs
Output: G_1 – coarsened graph

```
1  G_1 ← G_0;
2  hubs_list ← get_highest_centrality_values(G_0, num_hubs);
3  foreach hub in hubs_list do
4  |   vertices_del ← get_neighbors(G_0, hub);
5  |   foreach vertex in vertices_del do
6  |   |   vertices_connect ← vertices_connect ∪ get_neighbors(G_0, vertex);
7  |   end
8  |   foreach vertex in vertices_del do
9  |   |   remove_vertex(G_1, vertex);
10 |   end
11 |   foreach vertex in vertices_connect do
12 |   |   add_edge(G_1, hub, vertex);
13 |   end
14 end
15 return G_1;
```

Algorithm 1: Graph coarsening

main loop starts by filling a list *vertices_del* with identifiers of vertices, which are adjacent to hubs (line 4). These vertices will be deleted in the coarsened representation of initial graph. In lines 5-7 we fill the list *vertices_connect* with identifiers of vertices, which will be connected to hubs (actually, these vertices are neighbors of neighbors of the hubs). Finally, we delete neighbors of hubs in lines 8-10 and make new edges in lines 11-13.

3.2 Applying Changes to the Initial Graph

One interesting feature of graph coarsening based on its betweenness centrality values is that it approximately preserves the order of shortest paths. In other words, if the shortest path between vertices u and v passed through some hub (or its neighbors), then after applying coarsening procedure it will still be on the path through the newly created super-vertex. An

Input: G_0 – initial graph, *bc_exact* – previously computed exact values of betweenness centrality metric of the initial graph, *num_hubs* – number of vertices, which are considered to be hubs, new_edge – edge to add to G_0
Output: *bc_approximate* – approximate values of betweenness centrality metric of the resulting graph

```
1  subgraph_before ← graph_coarsening(G_0, num_hubs);
2  subgraph_after ← subgraph_before.E ∪ {new_edge};
3  bc_subgraph_before ← exact_bc(subgraph_before);
4  bc_subgraph_after ← exact_bc(subgraph_after);
5  foreach vertex in subgraph_after do
6  |  α[vertex] ← bc_subgraph_after[vertex] / bc_subgraph_before[vertex];
7  end
8  foreach vertex in subgraph_after do
9  |  bc_approximate[vertex] ← α[vertex] × bc_exact[vertex];
10 end
11 return bc_approximate;
```

Algorithm 2: Approximate betweenness centrality computation for dynamically changing graph (adding a new random edge)

example of such a situation is visualized in Figure 2. Due to the preservation of shortest paths

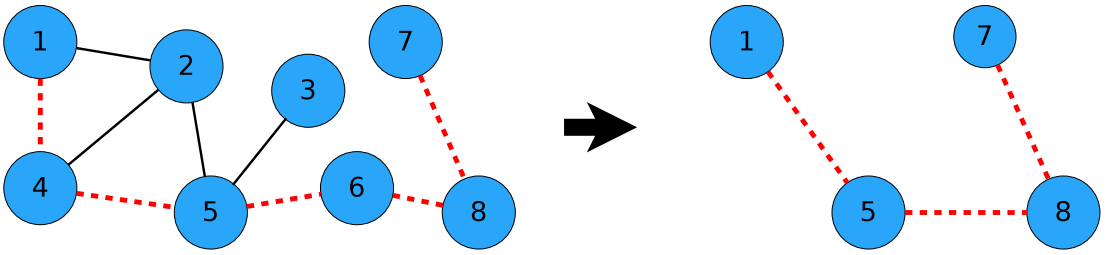

Figure 2: Representation of the shortest path (red dashed line) between vertices 1 and 7 before (left) and after (right) graph coarsening

in the condense graphs, we can use them as a replacement for the initial graph when computing the betweenness centrality. Therefore, in the case of dynamic graphs, we can compute an update for the relative change of betweenness centrality values on the coarsened graph after add some perturbation in graph and extrapolate it to the initial graph. Algorithm 2 shows the pseudocode for the algorithm approximating betweenness centrality (when adding random edge in graph). We start by creating two "small" representations of the initial graph (without and with new edge) in lines 1 and 2. Next, we compute the exact values of betweenness centrality metric for these two "small" graphs in lines 3 and 4. In the loop (lines 5-7) we compute correction coefficients α representing the relative change of the centrality metric in coarsened graphs. Finally, in lines 8-10, we update the betweenness centrality metric using the correction coefficients.

4 Benchmarks

For the purpose of testing our approach we use uniformly random Erdős-Rényi graphs [7] and RMAT graphs [5] (power law degree distribution). The benchmark uses three generated (fully connected) graphs of each type containing 1000, 1500 and 2000 vertices.

We implemented a prototype of our approach to approximating betweenness centrality (for dynamic graphs) with NetworkX citenetwork in Python. The experiments were carried out on workstation with Intel Core i7-2630QM and 4 GB RAM.

We observed that the betweenness centrality values before and after the addition of a single edge is very similar for both exact and approximate computations (using our approach). To compare we use the mean square error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{BC}_i - BC_i)^2,$$

where n is number of vertices in the graph, $\hat{BC}_i$ is the exact value, and BC_i is our approximate centrality value.

In the following we consider two mean square errors. First, we denote MSE_{EE} as the mean square "difference" between the exact values of betweenness centrality *before* and after the addition of a new edge. Second we denote MSE_{EA} as the error between the exact and approximate values *after* the addition of a new edge. We carried out experiments by adding a new edge to Erdős-Rényi and RMAT graphs for different graph sizes and numbers of hubs in the coarsening phase.

Initial graph		Hubs	Coarsened graph		MSE_{EE}	MSE_{EA}
Vertices	Edges		Vertices	Edges		
1000	4911	10	831	4545	1.06×10^{-10}	3.22×10^{-11}
		20	708	4246	4.06×10^{-10}	9.45×10^{-11}
		30	613	3972	2.64×10^{-10}	8.72×10^{-11}
		40	543	3811	4.31×10^{-10}	2.17×10^{-11}
		50	472	3636	4.09×10^{-10}	3.19×10^{-11}
1500	8287	15	1229	7669	6.90×10^{-10}	1.07×10^{-11}
		30	1026	7128	1.10×10^{-10}	1.28×10^{-11}
		45	870	6712	1.90×10^{-10}	2.70×10^{-11}
		60	733	6334	1.86×10^{-10}	1.77×10^{-11}
		75	647	6156	1.01×10^{-10}	4.69×10^{-11}
2000	9981	20	1653	9337	1.70×10^{-10}	8.43×10^{-11}
		40	1408	8835	4.78×10^{-11}	7.31×10^{-12}
		60	1225	8452	6.73×10^{-11}	7.01×10^{-12}
		80	1063	8097	5.71×10^{-11}	1.41×10^{-11}
		100	940	7894	5.10×10^{-11}	1.37×10^{-11}

Table 1: Approximation quality for different sizes and number hubs (Erdős-Rényi graphs)

The results in Table 1 show that for Erdős-Rényi graphs MSE_{EE} is strictly larger than MSE_{EA}, which indicates an acceptable accuracy of the approximation (approximate values are close to the exact ones). We choose the number of hubs such that the coarsest graph contains approximately half of the vertices of initial graph.

Initial graph		Hubs	Coarsened graph		MSE_{EE}	MSE_{EA}
Vertices	Edges		Vertices	Edges		
1000	2098	10	283	524	1.02×10^{-10}	9.00×10^{-11}
		20	212	555	1.20×10^{-9}	1.11×10^{-8}
		30	190	632	4.88×10^{-11}	4.08×10^{-11}
		40	180	675	5.80×10^{-11}	1.07×10^{-10}
		50	170	714	4.59×10^{-11}	6.29×10^{-11}
1500	3238	15	413	858	1.52×10^{-9}	1.15×10^{-9}
		30	350	935	1.60×10^{-10}	8.45×10^{-10}
		45	312	965	1.38×10^{-9}	2.78×10^{-9}
		60	294	1071	3.84×10^{-11}	2.34×10^{-9}
		75	275	1080	1.70×10^{-9}	8.88×10^{-6}
2000	4341	20	578	1223	1.74×10^{-9}	8.40×10^{-9}
		40	467	1356	2.99×10^{-11}	4.53×10^{-10}
		60	408	1483	9.38×10^{-11}	8.49×10^{-10}
		80	366	1536	1.67×10^{-9}	1.88×10^{-8}
		100	339	1573	3.61×10^{-9}	1.60×10^{-8}

Table 2: Approximation quality for different sizes and number hubs (RMAT graphs)

In contrast to the results for Erdős-Rényi, Table 2 shows that our approximation performs worse for RMAT graphs. MSE_{EE} is comparable with MSE_{EA} for some particular cases of small number of hubs (for instance, case of 10 hubs in graph with 1000 vertices and case of 15 hubs for graph with 1500 vertices). But most of experiments showed that MSE_{EE} smaller than MSE_{EA}. One of the reasons why this happens is that our method is sensitive to the degree distribution of the graph. Erdős-Rényi graphs have almost no imbalance in degree distribution, but RMAT graphs have few vertices with highest degrees and many vertices with small degrees. In terms of betweenness centrality it means that there are few vertices with high values of centrality metric and many vertices with zero values of centrality metric (with only one incident edge). Difference between values of centrality metric for both types of graphs showed in Figure 3. Probably, after coarsening step, degree distribution becomes more heavy-tailed because hubs adsorb its neighbors and make connections with many periphery vertices (with zero values of betweenness centrality). In current implementation of Algorithm 2 vertices with zero values of centrality not take into account. Therefore, because of such untracked nodes our approach demonstrates bad quality of approximation for RMAT graphs.

Additionally, we calculated the speedup of using our approximation approach in comparison to the exact computation for the case of Erdős-Rényi graphs. Results are presented in Figure 4. In the favorable case (large number of hubs) our approach attains a speedup of more than 60%. In the case of minimum and average number of hubs (3% and less of the total number of nodes) we obtained less (or even no) speedup. For example, whit 10 hubs in a graph of 1000 vertices the cost of computing the centrality for the two involved coarsened graphs ($subgraph\_before$ and $subgraph\_after$ in Algorithm 2) are higher than cost of computing the centrality for initial graph.

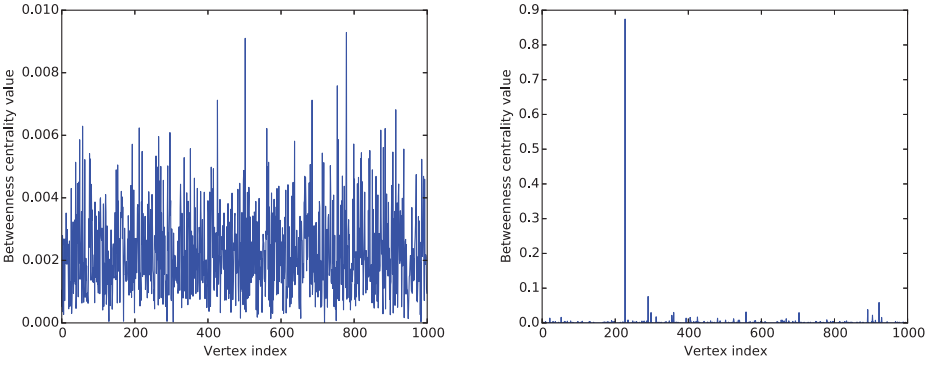

Figure 3: Exact values of betweenness centrality metric for Erdős-Rényi graph (left) and RMAT graph (right)

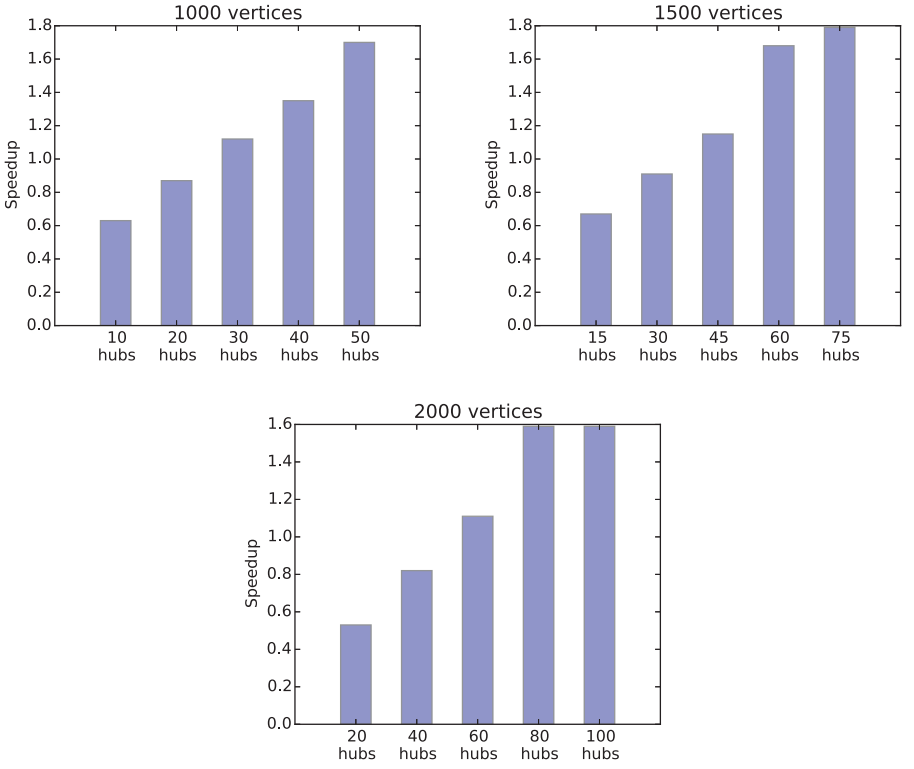

Figure 4: Speedup with respect to exact computation of betweenness centrality (for Erdős-Rényi graphs) with different number of hubs

5 Discussion

Despite the fact, that our approach demonstrates rather good approximation quality and speedup, it has some further work that we need to address:

1. α coefficients in Algorithm 2 are not computed for the neighbors of hubs. Therefore, during the approximation procedure, betweenness centrality values of these vertices does not change, which degrades quality of approximation;

2. According to Algorithm 2, α coefficients became indefinite for vertices, whose betweenness centrality values are zero on the "small" graph (see line 6 in Algorithm). In our approach betweenness centrality values of these vertices also stay unchanged to avoid division by zero;

3. Despite the fact that we achieve an acceptable speedup only in the case of many hubs (around 5% of the total number of nodes), the cases with smaller number of hubs can potentially be utilized when we continuously add a few edges in the same graph region, to improve coarsening performance;

4. Currently, our approach works well for uniformly random graph like Erdős-Rényi. It is crucial to adopt our approach for different types of graphs (i.e. RMAT) to apply it to a variety of real-world tasks;

5. Since the computation of betweenness centrality is well suited for parallelization, steps related to computing exact values of centrality metric in Algorithm 2 can be parallelized quite effectively (see in [11]);

6. Our approach can be fused with approaches exploiting sampling techniques. In this case, the exact computation of betweenness centrality values on the "small" graph can be potentially replaced with inexact sampling-based computations.

6 Conclusion

In this paper we present a novel approach to approximate the computation of betweenness centrality in the case of dynamically changing graphs. It consist of two steps: first we condense the initial graph to attain a "smaller" representation. Second we assess the changes on the "small" graph (adding a new edge) and extrapolate it to get the betweenness centrality estimation for the initial large graph.

Our approach demonstrates good approximation quality and good speedup (more than 60%) for uniformly random graphs with respect to the exact computation when coarsening tens of hubs.

Our further plans are split on two main directions – improving accuracy of approximation and parallelization of our approach as well. Also we interesting in tuning our approach for graphs with irregular inner structure and testing it on big real-world graph instances.

References

[1] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph*, WAW'07, pages 124–137, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] Alex Bavelas. A mathematical model for group structures. *Human Organizations*, 7:16–30, 1948.

[3] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[4] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318, 2007.

[5] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-mat: A recursive model for graph mining. In *In SDM*, 2004.

[6] David Eppstein and Joseph Wang. Fast approximation of centrality. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 228–229, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[7] P. Erdös and A. Rényi. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[8] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[9] Dimitra Gkorou, Johan Pouwelse, and Dick Epema. Efficient approximate computation of betweenness centrality. In *Proceedings of the 16th annual conf. on the Advanced School for Computing and Imaging*, ASCI 2010.

[10] Mostafa Haghir Chehreghani. An efficient algorithm for approximate betweenness centrality computation. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 1489–1492, New York, NY, USA, 2013. ACM.

[11] Kamesh Madduri, David Ediger, Karl Jiang, David A. Bader, and Daniel Chavarria-Miranda. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Matteo Riondato and Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 413–422, New York, NY, USA, 2014. ACM.

[13] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.