Parallel Algorithms for Evaluating Centrality for Weighted Graphs

Noor Mohammad Zahid
American International University-Bangladesh
Dhaka, Bangladesh

Badrun Nahar Khan
American International University-Bangladesh
Dhaka, Bangladesh

ABSTRACT
This paper discusses fast parallel algorithms for evaluating betweenness centrality in complex network analysis for weighted graphs. The previous studies on this topic mainly focused on unweighted graphs. Moreover, we will try to implement a shortest path algorithm which is the input of the parallel algorithm. These algorithms have been optimized to exploit properties typically observed in real-world large scale networks. The algorithm are implemented on real datasets such as the web graph, protein-interaction networks, movie-actor and citation networks, and report impressive parallel performance for evaluation of the computationally intensive centrality metrics on high-end shared memory symmetric multiprocessor and multithreaded architectures. For instance, we compute the exact betweenness centrality value for each vertex in a large US patent citation network (3 mil- lion patents, 16 million citations) in 42 minutes on 16 processors, utilizing 20GB RAM of the IBM p5 570. Current SNA packages on the other hand cannot handle graphs with more than hundred thousand edges.

General Terms
Parallel algorithm, centrality measurement, beweenness centrality

Keywords
Parallel algorithm, weighted graph, centrality, betweenneess centrality etc

1. INTRODUCTION
In any graph or network, the topology determines an influence structure among the nodes. Identifying the most important nodes in a network helps in explaining the network's dynamics, e.g., the distribution of power in exchange networks or migration in biological networks, as well as in designing optimal ways to externally influence the network, e.g., attack vulnerability of networks [1]. However, node importance is a rather vague concept and can be interpreted in various ways, giving rise to multiple coexisting centrality measures, the most common being degree closeness eigenvector and betweenness centrality. Finally, in betweenness centrality, the centrality of a node is given by the frequency of this node belonging to the shortest path between other two nodes in the network. Network analysis and modeling have received considerable attention in recent times, but algorithms are relatively less studied. Real-world networks are often very large in size, ranging from several hundreds of thousands to billions of vertices and edges. A space-efficient memory representation of such graphs is itself a big challenge, and dedicated algorithms have to be designed exploiting the unique characteristics of these networks. On single processor workstations, it is not possible to do exact in-core computations on large graphs due to the limited physical memory. Current high-end parallel computers have sufficient physical memory to handle large graphs, and a naıve in-core implementation of a graph theory problem is typically two orders of magnitude faster than the best external memory implementation [2]. Algorithm design is further simplified on parallel shared memory systems; due to the globally address memory space, there is no need to partition the graph, and we can avoid the overhead of message passing. However, attaining good performance is still a challenge, as a large class of graph algorithms are combinatorial in nature, and involve a significant number of non-contiguous, concurrent accesses to global data structures with low degrees of locality.

2. CENTRALITY MATRICS
One of the fundamental problems in network analysis is to determine the importance of a particular vertex or an edge in a network. Quantifying centrality and connectivity helps us identify portions of the network that may play interesting roles. Researchers have been proposing metrics for centrality for the past 50 years, and there is no single accepted definition. The metric of choice is dependent on the application and the network topology. Almost all metrics are empirical, and can be applied to element-level [3], grouplevel [4], or network-level analyses. We present a few commonly used indices in this section.

2.1 Preliminaries
Consider a graph G = (V,E), where V is the set of vertices representing actors or nodes in the social network, and E, the set of edges representing the relationships between the actors. The number of vertices and edges are denoted by n and m, respectively. The graphs can be directed or undirected. Let us assume that each edge e ∈ 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 (ui, ui+1), $0 \leq i \leq l$, where u0 = s and ul = 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 σst, and the number passing through vertex v by σst(v).

2.1.1 Degree Centrality
The degree centrality DC of a vertex v is simply the degree deg(v) for undirected graphs. For directed graphs, we can define two variants: in-degree centrality and out-degree centrality. This is a simple local measure, based on the notion of neighborhood. This index is useful in case of static graphs, for situations when we are interested in finding vertices that have the most direct connections to other vertices.

2.1.2 Closeness Centrality
This index measures the closeness, in terms of distance, of an actor to all other actors in the network. Vertices with a smaller total distance are considered more important. Several closeness-based metrics [5, 6, 7] have been developed by the SNA community. A commonly used definition is the

reciprocal of the total distance from a particular vertex to all other vertices:

$$CC(v) = 1 /(u \in V\ d(v, u))$$

Unlike degree centrality, this is a global metric. To calculate the closeness centrality of a vertex v, we may apply breadthfirst search (BFS, for unweighted graphs) or a single-source shortest paths (SSSP, for weighted graphs) algorithm from v.

2.1.3 Betweenness Centrality

Betweenness Centrality is another shortest paths enumeration-based metric, introduced by Freeman in [8]. Let δst(v) denote the pairwise dependency, or the fraction of shortest paths between s and t that pass through v:

$$\delta st(v) = \sigma st(v) /\sigma st$$

This metric can be thought of as normalized stress centrality. Betweenness centrality of a vertex measures the control a vertex has over communication in the network, and can be used to identify key actors in the network. High centrality indices indicate that a vertex can reach other vertices on relatively short paths, or that a vertex lies on a considerable fraction of shortest paths connecting pairs of other vertices. We discuss algorithms to compute this metric in detail in the next section. This index has been extensively used in recent years for analysis of social as well as other large scale complex networks. Some applications include biological networks [9, 10, 11], study of sexual networks and AIDS [12], identifying key actors in terrorist networks, organizational behavior, supply chain management, and transportation networks. There are a number of commercial and research software packages for SNA (e.g., Pajek [13], InFlow, UCINET) which can also be used to determine these centrality metrics. However, they can only be used to study comparatively small networks (in most cases, sparse graphs with less than 40,000 vertices). Our goal is to develop fast, high performance implementations of these metrics so that we can analyze large-scale real-world graphs of millions to billions of vertices.

3. BETWEENNESS CENTRALITY ALGORITHM

These two metrics require shortest paths enumeration and we design our parallel algorithm based on Brandes' [14] sequential algorithm for sparse graphs. Alg. 1 outlines the general approach for the case of unweighted graphs. On each BFS computation from s, the queue Q stores the current set of vertices to be visited, S contains all the vertices reachable from s, and P(v) is the predecessor set associated with each vertex v ∈ V . The arrays d and σ store the distance from s, and shortest path counts, respectively. The centrality values are computed in steps 22–25, by summing the dependencies δ(v), v ∈ V . The final scores need to be divided by two if the graph is undirected, as all shortest paths are counted twice. We observe that parallelism can be exploited at two levels:

Algorithm
Input: Graph G(V, E), the betweenness centrality values BC_G of G, and new edge e $\in$ V *V Output: The betweenness centrality values $BC_{G''}$ of graph G'' that is constructed by inserting edge e to graph G 1: Find the biconnected components of G'' 2: Let B'_e (V'_{Be}, E'_{Be}) be the biconnected component of G'' that edge e belongs to 3: Let B_e (V_{Be}, E_{Be}) be B'_e (V'_{Be}, E'_{Be} -{e}) 4: Let e =(v_1,v_2) 5: Perform a breadth-first search to compute the distance d_{v1s} between v_1 and s in B_e 6: Perform a breadth-first search to compute the distance d_{v2s} between v_2 and s in B_e 7: for all nodes s $\in$ V_{Be} do 8: if $d_{v1s} \neq d_{v2s}$ then 9: Add s to Q 10: for all nodes s $\in$ Q do 11: Find $\sigma_s[v]$ and $P_s[v]$ for v $\in$ V_{Be} (BFS from s) 12: $\delta_s[v] = 0$ for v $\in$ V_{Be} 13: $\delta_{Gs}[v] = 0$ for v $\in$ V_{Be} 14: for all nodes w $\in$ V_{Be} in reverse BFS order from s do 15: if s and w are articulation points then 16: $\delta_{Gs}[w]= \|V_{Gs}\|.\|V_{Gw}\|$ 17: for p $\in$ $P_s[w]$ do 18: $\delta_s[p] = {}_s[p] +(\sigma_s[p]/ \sigma_s[w]) . (1+ \delta_s[w])$ 19: if s is an articulation point then 20: $\delta_s[p] = \delta_s[p] + \delta_s[w] .(\sigma_s[p]/ \sigma_s[w])$ 21: if w $\neq$ s then 22: $BC_{G'}[w]= BC_{G'}[w]- \delta_s[w]/2.0$ 23: if s is an articulation point then 24: $BC_{G'}[w]= BC_{G'}[w]- \delta_s[w]. \|V_{Gs}\|$ 25: $BC_{G'}[w]= BC_{G'}[w]- \delta_{Gs}[w]/2.0$ 26: Find $\sigma'_s[v]$ and $P'_s[v]$ for V'_{Be} (partial BFS from s) 27: $\delta'_s[v] = 0$ for v $\in$ V'_{Be}

28: $\delta'_{Gs}[v] = 0$ for $v \in V'_{Be}$

29: for all nodes $w \in V'_{Be}$ in reverse BFS order from s do

30: if s and w are articulation points then

31: $\delta'_{Gs}[w] = |V_{Gs}|.|V_{Gw}|$

32: for $p \in P'_s[w]$ do

33: $\delta'_s[p] = {}'_s[p] +(\sigma'_s[p]/ \sigma'_s[w]) . (1+ \delta'_s[w])$

34: if s is an articulation point then

35: $\delta'_s[p] = \delta'_s[p] + \delta'_s[w] .(\sigma'_s[p]/ \sigma'_s[w])$

36: if $w \neq s$ then

37: $BC_{G'}[w]= BC_{G'}[w]+ \delta_s[w]/2.0$

38: if s is an articulation point then

39: $BC_{G'}[w]= BC_{G'}[w] + \delta_s[w]. |V_{Gs}|$

40: $BC_{G'}[w]= BC_{G'}[w] +\delta_{Gs}[w]/2.0$

41: return $BC_{G'}$

The algorithm works as follows: Line 1 decomposes the input graph G' into its biconnected components using Hopcroft and Tarjan algorithm. Then, Line 2 identifies biconnected component B'_e that is affected by the graph update. It performs the biconnected components decomposition on G' instead of G to support the general case when the inserted edge 'e' connects multiple biconnected components of G. Following, Lines 5–9 identify set Q by performing two breadth-first traversals in B_e. For all nodes in set Q, Lines 10–40 iteratively update the betweenness centrality of nodes in B'_e by performing a breadth-first and a reverse breadth-first search, respectively. The updates are done in two steps: subtracting the old source and external graph dependencies; i.e., A[v], B[v], and C[v] (Lines 11–25), and adding the new source and external graph dependencies; i.e., A'[v], B'[v], and C'[v] (Lines 26–40). Computing and adding A'[v], B'[v], and C'[v] are done similarly in Lines 26–40. Computing the source dependencies that encapsulate node pairs in A[v] is done with a direct application of Equation (4) in Line 18. These source dependencies are subtracted from the betweenness centrality values in Line 22. The pair dependencies in B[v] are encapsulated in $\delta_s[w]. |V_{Gs}|$, which is subtracted in Line 24. This is done only if 's' is an articulation. The case of pair dependencies in C[v] is more involved, and requires maintaining the structure $\delta_{Gs}[v]$to compute the external graph dependencies (Line 13). Computing $\delta_{Gs}[v]$ starts with the base case when both the source s and the node considered in the reverse breadth-first search (i.e, w) are articulation points. This is done in Line 16. The external graph dependency on node w is computed if the source s is an articulation point in Line 20. Note that $\delta_{Gs.}(w)$ is defined only if 's' is an articulation point, because that is the case when external graph dependencies are defined and needed to be propagated to nodes in B'e. Note that $\delta_{s.}(w)$and $\delta_{Gs.}(w)$ in Lines 22 and 25 to avoid counting the same path twice, since the input graph is undirected.

The alorithm utilizes the fact that many $\sigma_s v$ values remain unchanged in the breadth-first search DAG of s after inserting edge 'e', even when $s \in Q$. Let 'l' be the node of 'e' further from 's' in its breadth-first search DAG. To find the nodes for which $\sigma_s v$ changes, we only need to start a breadth-first search from 'l' in the breadth-first DAG of 's'. We refer to this optimized traversal as partial breadth-first search (Line 26).

4. CONCLUSION

We present parallel algorithms for evaluating several network indices, including betweenness centrality, optimized for Symmetric Multiprocessors and multithreaded architectures. To our knowledge, this is the first effort at parallelizing these widely-used social network analysis tools. Our implementations are designed to handle problem sizes in the order of billions of edges in the network, and this is three orders of magnitude larger than instances that can be handled by current social network analysis tools. We are currently working on improving the betweenness centrality implementation on the MTA-2, and also extend it to efficiently compute scores for weighted graphs. Previously, the works were based on unweighted graph. In future, we plan to implement and analyze performance of approximate algorithms for closeness and betweenness centrality detailed the paper, and also apply betweenness centrality values to solve harder problems like graph clustering.

5. ACKNOWLEDGMENTS

We would like to show our great honor to Dr. Saddam Hossain Mukta who has introduced with this topic and helped us to understand the things we could not understand.

6. REFERENCES

[1] Segarra, S., & Ribeiro, A. (2015). Stability and continuity of centrality measures in weighted graphs. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). doi:10.1109/icassp.2015.7178599

[2] Bader, D., & Madduri, K. (n.d.). Parallel Algorithms for Evaluating Centrality Indices in Real-world Networks. 2006 International Conference on Parallel Processing (ICPP06). doi:10.1109/icpp.2006.57

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

[4] P. Doreian and L. Albert. Partitioning political actor networks: Some quantitative tools for analyzing qualitative networks. Quantitative Anthropology, 161:279–291, 1989.

[5] A. Bavelas. Communication patterns in task oriented groups. J. Acoustical Soc. of America, 22:271–282, 1950

[6] G. Sabidussi. The centrality index of a graph. Psychometrika, 31:581–603, 1966.

[7] U. J. Nieminen. On the centrality in a directed graph. Social Science Research, 2:371–378, 1973

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

[9] H. Jeong, S. Mason, A.-L. Barabasi, and Z. Oltvai. Lethality and centrality in protein networks. Nature, 411:41, 2001

[10] J. Pinney, G. McConkey, and D. Westhead. Decomposition of biological networks using betweenness centrality. In Proc. Poster Session of the 9th Ann. Int'l Conf. on Research in Computational Molecular Biology (RECOMB 2004), Cambridge, MA, May 2005.

[11] A. del Sol, H. Fujihashi, and P. O'Meara. Topology of small-world networks of protein-protein complex structures. Bioinformatics, 21(8):1311–1315, 2005

[12] F. Liljeros, C. R. Edling, L. A. N. Amaral, H. E. Stanley, and Y. Aberg. The web of human sexual contacts. Nature, 411:907, 2001

[13] U. Brandes. A faster algorithm for betweenness centrality. J. Mathematical Sociology, 25(2):163–177, 2001.

[14] F. Jamour, S. Skiadopoulos and P. Kalnis, "Parallel Algorithm for Incremental Betweenness Centrality on Large Graphs", IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 3, pp. 659-672, 2018.