OPEN ACCESS Freely available online

PLoS one

Fast Computing Betweenness Centrality with Virtual Nodes on Large Sparse Networks

Jing Yang*, Yingwu Chen

Department of Information Systems and Management, National University of Defense Technology, Changsha, China

Abstract

Betweenness centrality is an essential index for analysis of complex networks. However, the calculation of betweenness centrality is quite time-consuming and the fastest known algorithm uses $O(N(M+N\log N))$ time and $O(N+M)$ space for weighted networks, where N and M are the number of nodes and edges in the network, respectively. By inserting virtual nodes into the weighted edges and transforming the shortest path problem into a breadth-first search (BFS) problem, we propose an algorithm that can compute the betweenness centrality in $O(\bar{w}\bar{D}N^2)$ time for integer-weighted networks, where $\bar{w}$ is the average weight of edges and $\bar{D}$ is the average degree in the network. Considerable time can be saved with the proposed algorithm when $\bar{w}<\log N/\bar{D}+1$, indicating that it is suitable for lightly weighted large sparse networks. A similar concept of virtual node transformation can be used to calculate other shortest path based indices such as closeness centrality, graph centrality, stress centrality, and so on. Numerical simulations on various randomly generated networks reveal that it is feasible to use the proposed algorithm in large network analysis.

Citation: Yang J, Chen Y (2011) Fast Computing Betweenness Centrality with Virtual Nodes on Large Sparse Networks. PLoS ONE 6(7): e22557. doi:10.1371/journal.pone.0022557

Editor: Marco Tomassini, Université de Lausanne, Switzerland

Received March 27, 2011; **Accepted** June 24, 2011; **Published** July 27, 2011

Funding: This work is funded by the National Science Foundation (no. 70971131). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: jing_edu@sohu.com

Introduction

Networks, especially complex networks, have been extensively studied during the last decade [1–3]. Owing to the ability to gather and analyze large scale data using computers and communication networks, it is quite common to see studies on networks with millions of vertices (nodes) nowadays. The shift of studies from simple small graphs to large complex networks have increasingly contributed new findings of critical phenomena and development of theories in many fields, such as the scale-free distribution of network degrees [4,5], burstness of human behaviors [6], vulnerability of internet networks [7,8], and so on [1–3,9].

However, the computation of several network properties, such as the shortest paths, betweenness centrality and closeness centrality, are hindered by the large computation complexity [3,10]. As a result, many large-scale networks are regarded as unweighted when the above measures are reported [2,3]. Large efforts have been made to improve the efficiency of algorithms for calculating those network properties [10,11]. Take the betweenness centrality, for example [12,13]: for a weighted network G with N nodes and M edges, the naive algorithm requires $O(N^3)$ time and $O(N^2)$ storage, regardless of the algorithms implemented to find the shortest paths. A much faster algorithm proposed by Brandes [14], on the other hand, can calculate the betweenness centrality in $O(N(M+N\log N))$ time and $O(N+M)$ space when the shortest paths are calculated by Dijkstra's algorithm implemented with a Fibonacci heap. Parallel algorithms are also proposed to improve the efficiency for the calculation of betweenness centrality [10,11,15–21]: for example, Bader and Madduri [10] proposed a betweenness centrality algorithm on a high-end shared memory symmetric multiprocessor and multithreaded architectures, with which is "possible" to achieve the computation in $O(N(M+N\log N)/p)$ time with access conflicts, where p is the number of processors used. However, the parallel algorithms requires much more complex programming and are highly dependent on the hardwares: for example, in Bader and Madduri's study [10], they used an IBM p5 570 on 16 processors and utilized 20GB RAM. These equipments are obviously not adaptable for general network researchers.

To circumvent the difficulties in calculating betweenness centrality with large time complexity, we propose a new algorithm for integer-weighted networks in this paper. By replacing the weighted edges with connected virtual nodes, the new algorithm computes the betweenness centrality in $O(\bar{w}\bar{D}N^2)$ time and $O(N+(2\bar{w}-1)M)$ space, with $\bar{w}$ and $\bar{D}$ being the average edge weight and average degree of the network, respectively.

Methods

The Brandes' Algorithm

Given a network $G=(V,E)$, with $|V|=N$ the number of nodes and $|E|=M$ the number of edges, for the purpose of this study, we consider strongly connected networks [22] with no self loops (acyclic). Let $W=\{w_{ij},1\le i,j\le N\}$ be the weight matrix of G, where $w_{ij}>0$ is the weight on edge e_{ij}. In real practice, W can be distances between airports, information flows between computers, traffic loads between cities, etc.

Let σ_{st} denote the number of shortest paths from node s to t, and $\sigma_{st}(v)$ be the number of shortest paths from s to t that pass through $v\in V$, then the betweenness centrality of node v is defined

as [13,14]:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (1)$$

From the definition we can see that betweenness centrality is the sum of the fraction of shortest paths over all pairs of nodes passing through the node, high betweenness centrality indicates that a node can reach others (or be reached by others) with relatively short paths, or the node lies on considerable fraction of shortest paths connecting others. In many fields, the betweenness centrality can be regarded as a measure of the extent to which the node has control over information flowing between others, and it is thus a core index for evaluating the importance of nodes in the network [13,23]. For example, in the study of networks vulnerability to attacks, the removal of nodes with the highest betweenness centrality is shown to be one of the most harmful strategies that can break down the networks [8].

A straightforward way of calculating the betweenness centrality then use the following steps:

Step 1 Compute the length and number of shortest paths between all pairs of nodes;

Step 2 For each node v, calculate $\delta_{st}(v) = \sigma_{st}(v)/\sigma_{st}$ (*pair dependency*) for each pair and sum them up.

Obviously, the complexity of the naive algorithm is dominated by the second step which requires $O(N^3)$ time summation and $O(N^2)$ storage of *pair dependencies*. To introduce Brandes' algorithm, we first define the set of predecessors of node t on the shortest paths from s:

$$P_s(t) = \{u \in V : \{u,t\} \in E, d_G(s,t) = d_G(s,u) + d_G(u,t)\} \quad (2)$$

where $d_G(s,t)$ is the distance of the shortest path from s to t. Then the number of shortest paths from s to t can be calculated as:

$$\sigma_{st} = \sum_{u \in P_s(t)} \sigma_{su} \quad (3)$$

To eliminate the need for explicit summation of all *pair dependencies*, Brandes [14] defines the *dependency* of node v as:

$$\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v) \quad (4)$$

$\delta_{s\bullet}(v)$ has the recursive property that

$$\delta_{s\bullet}(v) = \sum_{w: v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s\bullet}(w)) \quad (5)$$

Note that $\delta_{s\bullet}(v)$ is merely a partial sum of Eq. (1), then the betweenness centrality can be expressed by:

$$C_B(v) = \sum_{s \neq v \in V} \delta_{s\bullet}(v) \quad (6)$$

The summation of *pair dependencies* is then reduced to accumulation of *dependencies* defined by Eq. (5). Specifically, given the shortest paths from $s \in V$ in G, the array storing $\delta_{s\bullet}(v)$ for all nodes can be recursively calculated according to Eq. (5), by traversing the nodes in non-increasing order of their distances from s. An illustrative algorithm is shown in Algorithm 1. We can see that the calculation for Step 2 is now in $O(M)$ time and $O(N+M)$ space, then the calculation complexity of betweenness centrality is determined by the shortest path algorithms used in Step 1. Using Dijkstra's algorithm implemented with Fibonacci heap [24], which requires $O(M + N \log N)$ time for the single source shortest path problem [25], the betweenness centrality can be computed by Brandes' algorithm in $O(NM + N^2 \log N)$ time and $O(N+M)$ space on weighted networks [14].

Computing Betweenness Centrality with Virtual Nodes

Brandes' algorithm has greatly reduced the computation burden for betweenness centrality, however, the time complexity is still too high for networks with millions of nodes since the shortest path algorithm would cost a lot of computation time anyway. In this section, we propose a new algorithm that can reduce the time complexity in Step 1, such that the betweenness centrality can be calculated within reasonable time under certain conditions.

Replacement of Weighted Edges. Our new algorithm originates from the idea that an integer-weighted network can be broken down into a simple unweighted network with virtual nodes, such that the calculation of shortest paths in Step 1 can be solved as a breadth-first search (BFS) problem.

Algorithm 1: Brandes' algorithm [14].

```
1  C_B[v] ← 0, v ∈ V;
2  for s ∈ V do
3    [P, σ, S] = single source shortest path algorithm()
     /*P[v] = set of predecessors for shortest paths from s to v ∈ V;*/
     /*σ[v] = array storing the number of shortest paths from s passing through v; */
     /*S = stack storing the distances of nodes from s in non-increasing order; */
     /*accumulate dependency from the most distant nodes */
4    δ[v] ← 0, v ∈ V;
5    while S not empty do
6      pop w ← S;
7      for v ∈ P[w] do δ[v] ← δ[v] + (σ[v]/σ[w])(1 + δ[w]);
8      if w ≠ s then C_B[w] ← C_B[w] + δ[w];
9    end
10 end
```

Figure 1 illustrates the representation of an undirected weighted network by an undirected unweighted network with three additional virtual nodes. We can see that edge e_{AC} and e_{BC} are replaced by 3 and 2 unit edge segments with two and one virtual nodes inserted, respectively. The number of virtual nodes to be inserted on a weighted edge e_{ij}, is then $w_{ij} - 1$.

Let $G_\bullet = (V_\bullet, E_\bullet)$ be the unweighted representation of $G = (V,E,W)$ with virtual nodes, where $V_\bullet = V \cup V'$ with V' the set of virtual nodes, then the number of virtual nodes in $G_\bullet$, is $|V_\bullet| = |W| - |E| = (\bar{w} - 1)M$, where $\bar{w}$ is the average edge weight.

Virtual Node Based Algorithm for Betweenness Centrality. Obviously, the insertion of virtual nodes does not change the distances between pairs of nodes in V and consequently the number of shortest paths between nodes is the same as in G. The calculation of shortest paths on $G_\bullet$ can then be solved by the BFS algorithm, instead of the traditional Dijkstra's algorithm.

However, before applying the BFS on $G_\bullet$ to calculate the betweenness centrality for nodes in G, there is at least one problem to be solved: to use the existing theories on summation of *pair dependency* in Algorithm 1, the predecessors of nodes in V recorded during the shortest path calculation in $G_\bullet$, should be kept as the same as if they were calculated by any shortest path algorithm in G. This can be achieved as follows: suppose the BFS finds a shortest path from s to v: $s \rightarrow \cdots v \rightarrow u'_1 \rightarrow u'_2 \rightarrow t$, where u'_1, u'_2 are two virtual nodes inserted on edge e_{vt}, then the predecessor of u'_1,

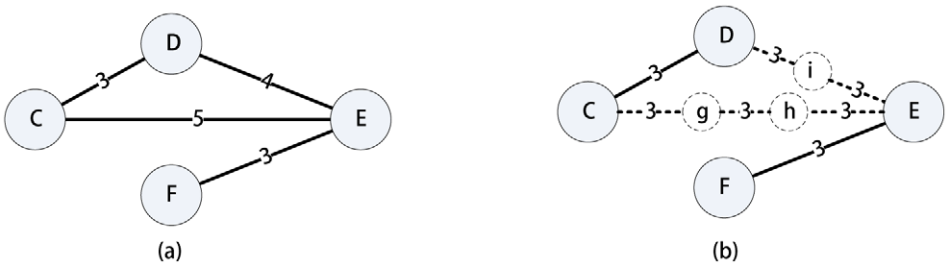

Figure 1. Illustration of representing the weighted network (a) by an unweighted network with virtual nodes (b).
doi:10.1371/journal.pone.0022557.g001

which is v, can be passed through u'_2 to the next non-virtual node t:

$$P[u'_1]=v;\ P[u'_2]=P[u'_1];\ P[t]=P[u'_2].$$

An implementation of the above process is presented in Algorithm 2, the steps for accumulation of *dependency* are identical as the Brandes' algorithm and thereby are omitted.

Algorithm 2: Virtual node algorithm for betweenness centrality

```
1   C_B[v]←0,v∈V;
2   for s∈V do
3      S←empty stack;
4      P[w]←empty list, w∈V_•;
5      σ[t]←0, t∈V_•; σ[s]←1;
6      d[t]←1, t∈V_•; d[s]←0;
7      Q←empty queue;
8      enqueue s→Q;
9      while Q not empty do
10        dequeue v←Q;
11        push v→S;
12        foreach neighbor w of v do
13           if d[w]<0 then /*visit w the first time*/
14              enqueue w→Q;
15              d[w]←d[v]+1;
16           end
17           if d[w]=d[v]+1 then
18              σ[w]←σ[w]+σ[v];
19              if v∈V then
20                 append v→P[w];
21                 else /*if v is a virtual node, retrieve the latest
                   non-virtual node as predecessor*/
22                 append P[v]→P[w];
23              end
24           end
25        end
26     end
27     accumulate dependency()/*as shown in Algorithm 1
28  end
```

Note that in Algorithm 2, we don't need to calculate shortest paths between virtual nodes. The BFS then requires $O(|V||E_\bullet|)=O(N\bar{w}M)$ time. For the sake of clarity, let $\bar{D}$ be the average degree of nodes in G such that $N\bar{D}=M$, then we have $N\bar{w}M=\bar{w}\bar{D}N^2$. The computation of betweenness centrality with virtual nodes (the VN algorithm), is dominated by the BFS and has a time complexity of $O(\bar{w}\bar{D}N^2)$, and needs $O(|V_\bullet|+|E_\bullet|)=O((N+(\bar{w}-1)M)+\bar{w}M)=O(N+(2\bar{w}-1)M)$ space.

Compared with Brandes' algorithm, we can see that the VN algorithm will perform better when $\bar{w}\bar{D}N^2<N^2(\bar{D}+\log N)$, that is, $\bar{w}<\log N/\bar{D}+1$. We henceforth denote $\bar{w}^*=\log N/\bar{D}+1$ as the critical threshold for the average edge weight on a network; if $\bar{w}<\bar{w}^*$, the VN algorithm will be able to calculate the betweenness centrality faster than Brandes' algorithm. Figure 2 shows the distribution of $\bar{w}^*$ over the domain of combinations of different network sizes and average degrees. We can see that the advantage of the VN algorithm becomes evident when the network is large and sparse, for example, when the network size is 1 million ($\approx 2^{20}$), and the average degree is 5, the VN algorithm would be faster for those with $\bar{w}\leq 5$; for the same average degree, $\bar{w}^*$ increases to 7 when the network size reaches 1 billion ($\approx 2^{30}$). For an average degree of 10, $\bar{w}^*$ lies beyond 3 for networks larger than 1 million. Note that many large-scale networks are reported to have rather small average degrees; for example, the mobile communication network reported in [26], contains 4.6 million nodes and an average of 3.04 edges. The Internet network [27], math co-authorship network [28], and power grid [29] reported in [1], are found to have average degrees of 3.5–4.1, 3.9 and 2.7, respectively. Networks with low integer weights are also reported in the literature; for example, the neural network of the Caenorhabditis elegans worm [29], the communication network of the online community [30], and the political support network of the US Senate [31], have average edge weights of 3.74, 2.95 and 3.74, respectively.

Results and Discussion

Numerical Experiments

To evaluate the algorithms, we generate scale-free networks [32] with different network sizes and edge weights, and the execution time of VN algorithm and Brandes' algorithm are then tested on these networks. Algorithms are coded in C and run on a PC with an Intel Core 2 Quad CPU (2.66 GHz, 6 Mb) and 6 Gb of RAM, all the following reported running times are the average of 100 simulations.

It is intuitive that when seldom edges in the network are weighted, the VN algorithm will calculate the betweenness centrality approximately as fast as the BFS, which is much faster than the Brandes' algorithm. For example, when the network size is 100,000 and we set the average degree as 2 and take 1000 edges to be weighted with random numbers generated from 1 to 10, the execution time for Brandes' algorithm is 8460 seconds, while the VN algorithm needs only 3830 seconds, which is around 1.3 hours faster than the Brandes' algorithm. Since when N becomes large, we have $\bar{w}\to 1$, more time can be expected to be saved in larger networks with fixed number of weighted edges. We calculated the VN and Brandes' algorithm on networks with 1% of edges being weighted as 2, and the execution times are presented in Figure 3(a). We can see that the difference in execution time become larger when the network size increases. When the network size is 50,000, the VN algorithm is 3 and 1.5 times faster than the Brandes' algorithm, for average network degrees of 2 and 10, respectively.

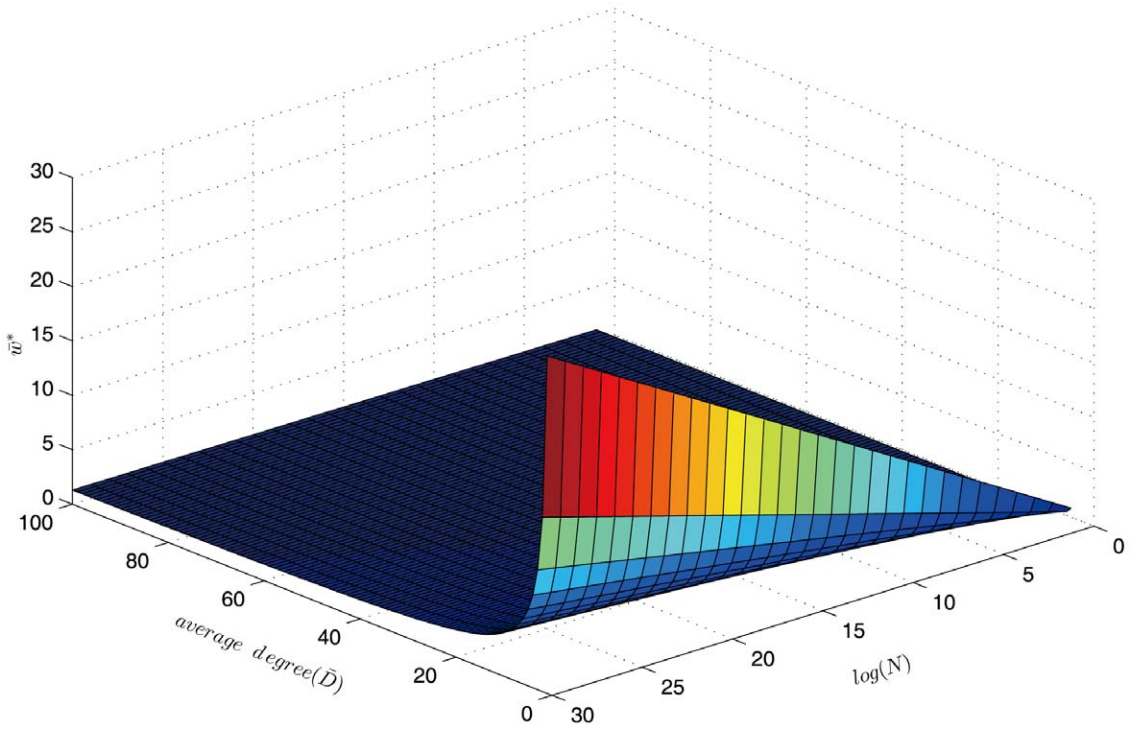

Figure 2. Critical threshold for average weights ($\bar{w}^*$) on networks with specified network size (N) and average degree ($\bar{D}$).
doi:10.1371/journal.pone.0022557.g002

The above results reveal that the VN algorithm is much faster on large sparse networks with limited number of weights. However, we should note that the VN algorithm is quite sensitive to the average degree and weight sum of the network, for any network with $N \leq 2^{\bar{D}}$, the VN algorithm will not outperform Brandes' algorithm as long as $\bar{w} > 2$. To illustrate the sensitivity of the VN algorithm, we run algorithms on networks with $\bar{D} = 2$ and $\bar{w} = 2$, and the difference in running times between the two algorithms decreases quickly as expected (Figure 3(b)).

Discussion

By replacing the weighted edges with connected virtual nodes, we propose the VN algorithm to calculate the betweenness centrality in weighted networks with the BFS rather than shortest path

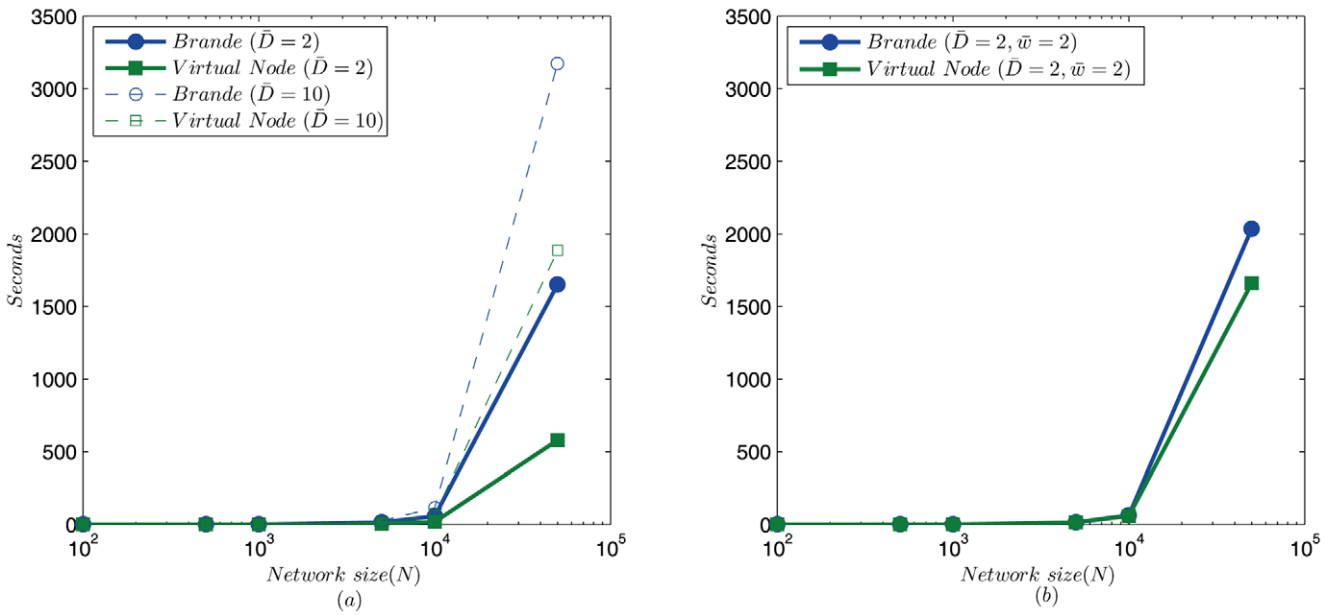

Figure 3. Running time of the VN algorithm and Brandes' algorithm. (a) Networks with average degree $\bar{D} = 2$ and $\bar{D} = 10$, 1% of the network edges are weighted with $\bar{w} = 2$; (b) Networks with average degree $\bar{D} = 2$, all edges are weighted with $\bar{w} = 2$.
doi:10.1371/journal.pone.0022557.g003

algorithms. The VN algorithm uses $O(\bar{w}\bar{D}N^2)$ time and $O(N+(2\bar{w}-1)M)$ space. Theoretically, the VN algorithm outperforms the Brandes' algorithm when $\bar{w} < \log N/\bar{D}+1$, indicating that when the average edge weight is low, considerable time can be saved on large sparse networks. The simulation study confirms that when $\bar{w} < \bar{w}^*$, more time can be saved when the network grows large.

We should note that the VN algorithm is quite sensitive to the density and weight of the networks, it can hardly outperform the Brandes' algorithm when the network is dense and weighted with large values. What's more, the theoretical threshold value $\bar{w}^*$, could be even lower in practice since the VN algorithm requires more space. Despite these limitations, given the evidences that large-scale networks in real life are mostly sparse, and the BFS is much easier to implement than the Fibonacci heap based shortest path algorithms, the VN algorithm is expected to be able to save analysis time in many scenarios. Moreover, the VN algorithm can easily be generalized to calculate other shortest path based network properties, such as closeness centrality [33], graph centrality [34], stress centrality [35], and so on. We henceforth recommend that network researchers to use the VN algorithm when the studied network is large, sparse, and lightly weighted, but continue to use the Brandes' algorithm otherwise.

Supporting Information

Both the Brandes' algorithm and the VN algorithm are written in C and are available upon request from the author.

Author Contributions

Conceived and designed the experiments: JY YC. Performed the experiments: JY YC. Analyzed the data: JY YC. Contributed reagents/materials/analysis tools: JY YC. Wrote the paper: JY YC.

References

1. Albert R, Barabasi AL (2002) Statistical mechanics of complex networks. Reviews of Modern Physics 74: 47–97.
2. Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU (2006) Complex networks: Structure and dynamics. Physics Reports 424: 175–308.
3. Newman MEJ (2003) The structure and function of complex networks. SIAM Review 45: 167–256.
4. Albert R, Jeong H, Barabasi AL (1999) Diameter of the world-wide web. Nature 401: 130–131.
5. Liljeros F, Edling CR, Nunes Amaral LA, Stanley HE, Aberg Y (2001) Social networks: The web of human sexual contacts. Nature 411: 907–908.
6. Barabasi AL (2005) The origin of bursts and heavy tails in human dynamics. Nature 435: 207–211.
7. Albert R, Jeong H, Barabasi AL (2000) Error and attack tolerance of complex networks. Nature 406: 378–382.
8. Holme P, Kim BJ, Yoon CN, Han SK (2002) Attack vulnerability of complex networks. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 65: 056109/1–056109/14.
9. Pastor-Satorras R, Vespignani A (2001) Epidemic spreading in scale-free networks. Physical Review Letters 86: 3200–3203.
10. Bader DA, Madduri K (2006) Parallel algorithms for evaluating centrality indices in real-world networks. In: Proceedings of the International Conference on Parallel Processing. pp 539–547.
11. Madduri K, Bader DA (2009) Compact graph representations and parallel connectivity algorithms for massive dynamic network analysis. In: IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium.
12. Linton CF (1977) A set of measures of centrality based on betweenness. Sociometry 40: 35–41.
13. Scott J (2000) Social Network Analysis: A Handbook SAGE Publications.
14. Brandes U (2001) A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25: 163–177.
15. Bader DA, Kintali S, Madduri K, Mihail M Approximating betweenness centrality.
16. Jiang K, Ediger D, Bader DA (2009) Generalizing k-betweenness centrality using short paths and a parallel multithreaded implementation. In: Proceedings of the International Conference on Parallel Processing. pp 542–549.
17. Madduri K, Ediger D, Jiang K, Bader DA, Chavarria-Miranda D (2009) A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In: IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium.
18. Puzis R, Elovici Y, Dolev S (2007) Fast algorithm for successive computation of group betweenness centrality. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 76.
19. Tan G, Sreedhar VC, Gao GR (2009) Analysis and performance results of computing betweenness centrality on ibm cyclops64. Journal of Supercomputing. pp 1–24.
20. Tan G, Tu D, Sun N (2009) A parallel algorithm for computing betweenness centrality. In: Proceedings of the International Conference on Parallel Processing. pp 340–347.
21. Tu D, Tan G (2009) Characterizing betweenness centrality algorithm on multi-core architectures. In: Proceedings - 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2009. pp 182–189.
22. Schwarte N, Cohen R, Ben-Avraham D, Barabasi AL, Havlin S (2002) Percolation in directed scalefree networks. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 66: 015104/1–015104/4.
23. Newman MEJ (2005) A measure of betweenness centrality based on random walks. Social Networks 27: 39–54.
24. Fredman ML, Tarjan RE (1987) Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34: 596–615.
25. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1: 269–271.
26. Onnela JP, Saramaki J, Hyvonen J, Szabo G, Lazer D, et al. (2007) Structure and tie strengths in mobile communication networks. Proceedings of the National Academy of Sciences of the United States of America 104: 7332–7336.
27. Yook SH, Jeong H, Barabasi AL (2002) Modeling the internet's large-scale topology. Proceedings of the National Academy of Sciences of the United States of America 99: 13382–13386.
28. Barabasi AL, Jeong H, Neda Z, Ravasz E, Schubert A, et al. (2002) Evolution of the social network of scientific collaborations. Physica A: Statistical Mechanics and its Applications 311: 590–614.
29. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393: 440–442.
30. Panzarasa P, Opsahl T, Carley KM (2009) Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. Journal of the American Society for Information Science and Technology 60: 911–932.
31. Skvoretz J, Carolina UOS (2003) Complexity theory and models for social networks. Complexity. pp 47–55.
32. Barabasi AL, Albert R (1999) Emergence of scaling in random networks. Science 286: 509–512.
33. Sabidussi G (1966) The centrality index of a graph. Psychometrika 31: 581–603.
34. Hage P, Harary F (1995) Eccentricity and centrality in networks. Social Networks 17: 57–63.
35. Shimbel A (1953) Structural parameters of communication networks. The Bulletin of Mathematical Biophysics 15: 501–507.