2016 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2016, Yugawara, Japan, November 27th-30th, 2016

A Fast Method for Finding the Edge to be Added to Minimize Betweenness Centrality of a Specified Vertex

Toshiyuki Namba, Tatsuki Kohno and Norikazu Takahashi

Graduate School of Natural Science and Technology, Okayama University
3–1–1 Tsushima-naka, Kita-ku, Okayama 700–8530, Japan
Email: {nanba,kono}@momo.cs.okayama-u.ac.jp, takahashi@cs.okayama-u.ac.jp

Abstract—Betweenness centrality is a measure that represents the importance of each vertex in a graph. From the viewpoint of the robustness against attacks and failures, it is desired that all vertices take similar values of the betweenness centrality. In this paper, a fast algorithm for finding the edge to be added to minimize the betweenness centrality of a specified vertex is developed. The efficiency of the proposed algorithm is confirmed by some experiments on random graphs and scale-free graphs. It is also shown experimentally that the proposed algorithm is useful for improving the robustness of the graph by an edge addition.

1. Introduction

Since the seminal paper by Watts and Strogatz [1], complex networks have attracted a great deal of attention from researchers in physics, engineering, economics, sociology, and so on. When analyzing a network, it is often important to evaluate the importance of each node quantitatively. Various measures of the importance have been proposed in the literature. Among them, this paper focuses on the betweenness centrality [2] which is well known and widely used in the analysis of complex networks.

From the viewpoint of the robustness against attacks and failures, it is desired that all vertices take similar values of the betweenness centrality. A natural way to make a given network more robust is to add a small number of edges to decrease the largest betweenness centrality of the network as much as possible. As an example, let us consider the graph shown in Fig. 1 (a). If we add an edge to connect vertices 2 and 5, the largest betweenness centrality decreases from 8 to 4. However, finding edges to be added is computationally very expensive because, for each of the nonexisting edges, we have to compute the betweenness centrality for all vertices assuming that the edge is added.

In this paper, we consider the problem of finding the edge to be added in order to minimize the betweenness centrality of a specified vertex. This problem is closely related to the improvement of the robustness of a network mentioned above. If the betweenness centrality of the vertex with the largest betweenness centrality can be greatly decreased by an edge addition, it is expected that the largest betweenness centrality of the graph also decreases. We first provide a few theoretical results about the effect of an edge addition on the number and length of shortest paths between any pair of vertices. We next propose an algorithm, which is based on these theoretical results, for solving the above-mentioned problem. We finally examine the efficiency of the proposed algorithm by some experiments on random graphs and scale-free graphs, and show that the proposed algorithm is much faster than a simple method based on Brandes's algorithm [3].

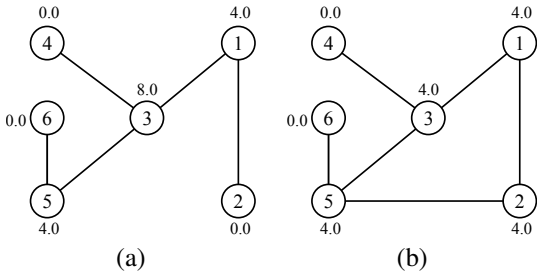

Figure 1: Effect of an edge addition on betweenness centrality. The number beside each vertex represents the betweenness centrality of the vertex.

2. Betweenness Centrality

2.1. Mathematical Expression of Networks

Throughout this paper, a network is expressed as a simple connected undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$ is the vertex set and $E = \{e_1, e_2, \ldots, e_M\}$ is the edge set. Because G is simple and undirected, each member of E is an unordered pair of distinct vertices.

The set of all shortest paths from vertex s to vertex $t\,(\neq s)$ in G is denoted by $G_{st} = (V_{st}, E_{st})$ where $V_{st} \subseteq V$ is the set of all vertices that appear in the shortest paths, and E_{st} is the set of all directed edges that appear in the shortest paths. Note that, unlike E, each member of E_{st} is an ordered pair of distinct vertices. The set of all shortest paths from vertex s to all other vertices in G is denoted by $G_s = (V, E_s)$ where E_s is the set of all edges that appear in the shortest paths. It is clear that E_{st} is a subset of E_s.

The number of shortest paths from vertex s to vertex t and the length of these paths are denoted by σ_{st} and d_{st}, respectively. Because G is undirected, $\sigma_{st} = \sigma_{ts}$ and $d_{st} = d_{ts}$ for all pairs of s and $t\,(\neq s)$. In the following discussions,

we assume for the sake of convenience that $\sigma_{ss} = 1$ and $d_{ss} = 0$ for all $s \in V$.

2.2. Betweenness Centrality

The betweenness centrality of vertex i, which is denoted by B_i, is defined by

$$B_i = \sum_{s \neq i} \sum_{t \neq i,s} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{1}$$

where $\sigma_{st}(i)$ is the number of shortest paths from vertex s to vertex t that pass through vertex i. In other words, the betweenness centrality of vertex i is the sum of the ratio of the number of shortest paths from s to t passing through i to the number of all shortest paths from s to t over all pairs of s and t. Therefore, a vertex with a high betweenness centrality is a key for many pairs of vertices because all or many of the shortest paths between these two vertices pass through the vertex.

2.3. Brandes's Algorithm for Computing Betweenness Centrality

A simple method for computing B_i for all $i \in V$ is as follows. First, for $s = 1, 2, \ldots, N$, one finds all shortest paths from vertex s to all other vertices by using, for example, the breadth-first search algorithm. During this process, d_{st} and σ_{st} can be found for all pairs of distinct vertices s and t, and $\sigma_{st}(i)$ can be found for all triples of distinct vertices s, t and i. Next, one computes B_i by using (1) for $i = 1, 2, \ldots, N$. The computational complexity of this method is $O(N^3)$, which becomes very large as the number of vertices increases.

The most widely used algorithm for computing the betweenness centrality is the one proposed by Brandes [3]. It is described as follows.

Algorithm 1 (Brandes's Algorithm)
Input: A simple connected undirected graph $G = (V, E)$
Output: $B_1, B_2, \ldots, B_N$

1. Set $s \leftarrow 1$ and $B_i \leftarrow 0$ for $i = 1, 2, \ldots, N$.
2. Find $G_s = (V, E_s)$ by using the breadth-first search. In so doing, find also d_{si} and σ_{si} for $i = 1, 2, \ldots, N$.
3. Set $\delta_s(i) \leftarrow 0$ for $i = 1, 2, \ldots, N$.
4. Update the value of $\delta_s(i)$ by
$$\delta_s(i) \leftarrow \sum_{j:(i,j) \in E_s} \frac{\sigma_{si}}{\sigma_{sj}}(1 + \delta_s(j)), \quad i = 1, 2, \ldots, s-1, s+1, s+2, \ldots, N.$$
5. Update the value of B_i by
$$B_i \leftarrow B_i + \delta_s(i), \quad i = 1, 2, \ldots, N.$$
6. If $s = N$ then return $B_1, B_2, \ldots, B_N$ and stop. Otherwise set $s \leftarrow s + 1$ and go to Step 3.

Let us consider the computational complexity of this algorithm. Step 2 can be done in $O(M)$ time, where M is the number of edges. Step 4 can be done in $O(M)$ too, because we can update the values of $\{\delta_s(i)\}_{i=1}^N$ while traversing the vertices once in non-increasing order of their distance from s [3]. Therefore, the total computational complexity of Algorithm 1 is $O(NM)$, which is in general lower than $O(N^3)$. In particular, the former is much lower than the latter if $M \ll N^2$, that is, the graph is sparse.

3. Theoretical Analysis of the Effect of One Edge Addition on Betweenness Centrality

Let $G' = (V, E')$ be the graph obtained from a graph $G = (V, E)$ by adding an edge $e = \{\alpha, \beta\} \notin E$. The set of all shortest paths from vertex s to vertex $t\,(\neq s)$ in G' is denoted by the directed graph $G'_{st} = (V'_{st}, E'_{st})$. The set of all shortest paths from s to all other vertices in G' is denoted by $G'_s = (V, E'_s)$. The number of shortest paths from s to t and the length of these paths are denoted by σ'_{st} and d'_{st}, respectively. The number of shortest paths from s to t that pass through i is denoted by $\sigma'_{st}(i)$.

In this section, we show that the values of σ'_{st}, $\sigma'_{st}(i)$ and d'_{st} can be computed from $\{\sigma_{pq}\}_{p,q=1}^N$ and $\{d_{pq}\}_{p,q=1}^N$. We hereafter assume without loss of generality that

$$d_{s\alpha} \leq d_{s\beta}. \tag{2}$$

We also assume for convenience that $V_{ss} = V'_{ss} = \{s\}$, $E_{ss} = E'_{ss} = \emptyset$, $\sigma_{ss} = \sigma'_{ss} = 1$ and $d_{ss} = d'_{ss} = 0$ for all $s \in V$.

Theorem 1 Under the assumption (2), the following statements hold true.

1. If $d_{s\alpha} < d_{s\beta}$ and $d_{s\alpha} + d_{\beta t} + 1 < d_{st}$ then i) G'_{st} consists of $G_{s\alpha}$, the directed edge (α, β) and $G_{\beta t}$, ii) $\sigma'_{st} = \sigma_{s\alpha}\sigma_{\beta t}$, and iii) $d'_{st} = d_{s\alpha} + 1 + d_{\beta t}$.
2. If $d_{s\alpha} < d_{s\beta}$ and $d_{s\alpha} + d_{\beta t} + 1 = d_{st}$ then i) G'_{st} consists of G_{st}, $G_{s\alpha}$, the directed edge (α, β), and $G_{\beta t}$, ii) $\sigma'_{st} = \sigma_{st} + \sigma_{s\alpha}\sigma_{\beta t}$, and iii) $d'_{st} = d_{st}$.
3. If $d_{s\alpha} = d_{s\beta}$ and $d_{s\alpha} + d_{\beta t} + 1 > d_{st}$ then i) $G'_{st} = G_{st}$, ii) $\sigma'_{st} = \sigma_{st}$, and iii) $d'_{st} = d_{st}$.

Theorem 2 Under the assumption (2), the following statements hold true.

1. If $d_{s\alpha} < d_{s\beta}$ and $d_{s\alpha} + d_{\beta t} + 1 < d_{st}$ then
$$\sigma'_{st}(i) = \begin{cases} \sigma_{si}\sigma_{i\alpha}\sigma_{\beta t}, & \text{if } i \in V_{s\alpha}, \\ \sigma_{s\alpha}\sigma_{\beta i}\sigma_{it}, & \text{if } i \in V_{\beta t}, \\ 0, & \text{if } i \notin V_{s\alpha} \cup V_{\beta t}. \end{cases}$$

2. If $d_{s\alpha} < d_{s\beta}$ and $d_{s\alpha} + d_{\beta t} + 1 = d_{st}$ then

$$\sigma'_{st}(i) = \begin{cases} \sigma_{st}(i) + \sigma_{si}\sigma_{i\alpha}\sigma_{\beta t}, & \text{if } i \in V_{s\alpha}, \\ \sigma_{st}(i) + \sigma_{s\alpha}\sigma_{\beta i}\sigma_{it}, & \text{if } i \in V_{\beta t}, \\ \sigma_{st}(i), & \text{if } i \notin V_{s\alpha} \cup V_{\beta t}. \end{cases}$$

3. If $d_{s\alpha} = d_{s\beta}$ or $d_{s\alpha} + d_{\beta t} + 1 > d_{st}$ then $\sigma'_{st}(i) = \sigma_{st}(i)$.

We omit the proofs of these theorems due to space constraints.

Note that we need to check the conditions $i \in V_{s\alpha}$ and $i \in V_{\beta t}$ in order to compute $\sigma'_{st}(i)$. However, this is easily done because it is well known that $i \in V_{pq}$ if and only if $d_{pq} = d_{pi} + d_{iq}$. Note also that the value of $\sigma_{st}(i)$ is required for the computation of $\sigma'_{st}(i)$. However, $\sigma_{st}(i)$ can be easily obtained by

$$\sigma_{st}(i) = \begin{cases} \sigma_{si}\sigma_{it}, & \text{if } d_{si} + d_{it} = d_{st}, \\ 0, & \text{if } d_{si} + d_{it} \neq d_{st}. \end{cases}$$

Thus $\sigma'_{st}(i)$ can be computed from $\{\sigma_{pq}\}_{p,q=1}^{N}$ and $\{d_{pq}\}_{p,q=1}^{N}$.

4. Proposed Algorithm

We consider in this section the problem of finding the edge to be added to minimize the betweenness centrality of a specified vertex. For this problem, the following algorithm is easily derived from the theoretical results in the previous section.

Algorithm 2
Input: $G = (V, E)$ and $i \in V$.
Output: Edge $\{\alpha, \beta\} \notin E$ to be added.

1. Compute $\{\sigma_{pq}\}_{p,q=1}^{N}$ and $\{d_{pq}\}_{p,q=1}^{N}$ for G by the breadth-first search algorithm.
2. Set $B_{\min} \leftarrow \infty$ and $\alpha \leftarrow 1$.
3. Set $\beta \leftarrow \alpha + 1$.
4. If $\{\alpha, \beta\} \in E$ then go to Step 7. Otherwise go to Step 5.
5. Compute B'_i by Algorithm 3 given below.
6. If $B'_i < B_{\min}$ then set $B_{\min} \leftarrow B'_i$, $\alpha_{\min} \leftarrow \alpha$ and $\beta_{\min} \leftarrow \beta$.
7. If $\beta = N$ then go to Step 8. Otherwise set $\beta \leftarrow \beta + 1$ and go to Step 4.
8. If $\alpha = N - 1$ then return $\{\alpha_{\min}, \beta_{\min}\}$ and stop. Otherwise set $\alpha \leftarrow \alpha + 1$ and go to Step 3.

Algorithm 3
Input: $G = (V, E)$, $\{\sigma_{pq}\}_{p,q=1}^{N}$, $\{d_{pq}\}_{p,q=1}^{N}$, $\{\alpha, \beta\} \notin E$ and $i \in V$
Output: B'_i

1. Set $s \leftarrow 1$ and $B'_i \leftarrow 0$.
2. If $s = i$ then go to Step 8. Otherwise go to Step 3.
3. Set $t \leftarrow 1$.
4. If $t \in \{i, s\}$ then go to Step 7. Otherwise go to Step 5.
5. Compute σ_{st} and $\sigma_{st}(i)$ by Theorem 1 and Theorem 2, respectively.
6. $B'_i \leftarrow B'_i + \sigma_{st}(i)/\sigma_{st}$
7. If $t = N$ then go to Step 8. Otherwise set $t \leftarrow t + 1$ and go to Step 4.
8. If $s = N$ then return B'_i and stop. Otherwise set $s \leftarrow s + 1$ and go to Step 2.

Let us examine the computational complexity of Algorithm 2. Step 1 can be done in $O(NM)$ time. Step 5, that is, Algorithm 3 can be accomplished in $O(N^2)$ time, and there are $N(N-1)/2 - M$ candidates for the edge to be added. Therefore, we can conclude that the computational complexity of Algorithm 2 is $O(N^4)$.

In Step 5 of Algorithm 2, we can employ Algorithm 1 instead of Algorithm 3. In this case, Step 1 can be skipped. Also, unnecessary computations in Steps 4 and 5 of Algorithm 1 can be skipped too. Nevertheless, the computational complexity of this simple algorithm is $O(N^3M)$ which is higher than $O(N^4)$.

Table 1: Computation time for random graphs.

N	M	Proposed (sec)	Simple (sec)
70	387	0.062	12.453
100	873	0.265	96.812
150	1867	1.281	986.25
200	3317	4.031	5910.172
250	5083	9.968	23525.422

Table 2: Computation time for scale-free graphs.

N	M	Proposed (sec)	Simple (sec)
70	204	0.094	7.203
100	294	0.328	40.891
150	444	1.609	302.077
200	594	4.968	1197.421
250	744	11.890	3520.593

5. Application of the Proposed Algorithm to Vertex with the Maximum Betweenness Centrality

In order to evaluate the efficiency of the proposed algorithm, the authors applied Algorithm 2 and the algorithm mentioned in the last paragraph of the previous section, which is hereafter called the simple algorithm, to random graphs [4] and scale-free graphs [5]. By a random graph, we mean a graph such that each pair of vertices is connected with probability p. In the experiments, the value

Table 3: The largest betweenness centrality before and after an edge addition to random graphs.

N	M	Before	After	Diff.
70	387	83.261	76.778	−6.483
100	873	110.542	107.266	−3.276
150	1867	132.943	130.914	−2.029
200	3317	183.125	181.772	−1.353
250	5083	243.350	241.127	−2.223

Table 4: The largest betweenness centrality before and after an edge addition to scale-free graphs.

N	M	Before	After	Diff.
70	204	1327.625	1307.093	−20.532
100	294	3330.202	3303.231	−26.971
150	444	8228.860	8178.460	−50.400
200	594	15834.116	15776.048	−58.068
250	744	25438.355	25367.843	−70.512

of p was set to 0.1666. Scale-free graphs were generated by the process proposed by Barabási and Albert [5]. To be more specific, we start with the complete graph with m_0 vertices, and then add new vertices one by one. When a new vertex is added, it is connected to $m\,(\leq m_0)$ existing vertices with a probability that is proportional to their degrees. In the experiments, the values of m_0 and m were set to 4 and 3, respectively. For each of the graphs mentioned above, the vertex with the largest betweenness centrality was chosen as the specified vertex. All algorithms were implemented with C language, complied with gcc version 5.3.0 and executed on a PC with Intel Core i5 4590 processor and 8GB RAM.

Computation time of the two algorithms is shown in Tables 1 and 2. In the case of random graphs, the proposed algorithm is faster than the simple one by a factor of 200 to 2360 as shown in Table 1. In the case of scale-free graphs, the proposed algorithm is faster than the simple one by a factor of 77 to 296 as shown in Table 2. These results suggest the effectiveness of the proposed algorithm.

If the edge found by the proposed algorithm is added to the graph, the betweenness centrality of the vertex having the largest betweenness centrality among all vertices is minimized. However, it is not guaranteed that the largest betweenness centrality of the graph decreases. If the betweenness centrality of some vertex other than i, which is the vertex having the largest betweenness centrality before the edge is added, becomes greater than B_i by the addition of the edge, the largest betweenness centrality increases. So let us examine whether or not the largest betweenness centrality is decreased by the edge addition. Results are shown in Tables 3 and 4, from which we see that the largest betweenness centrality decreases for all graphs used in the experiments. However, this is not always true. To see this, let us consider the graph shown in Fig. 2. Among

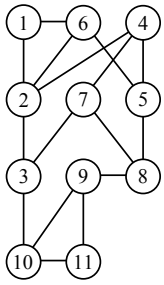

Figure 2: A graph with 11 vertices and 16 edges.

11 vertices of the graph, vertex 3 has the largest betweenness centrality, which is 11.50. If Algorithm 2 is applied to vertex 3 of this graph, it returns edge $\{2, 10\}$. In fact, this edge can decrease the betweenness centrality of vertex 3 to 2.95. However, it also increases the betweenness centrality of vertex 2 from 10.75 to 14.50.

6. Conclusion

The problem of finding the edge to be added that minimizes the betweenness centrality of a specified vertex of a graph has been studied in this paper. We first analyzed theoretically the effect of an edge addition on the number and length of the shortest paths between each pair of vertices. We then developed an algorithm, which is based on the theoretical analysis, for solving the problem and demonstrated the efficiency by experiments.

Acknowledgement

This work was partially supported by JSPS KAKENHI Grant Number JP15K00035.

References

[1] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol.393, pp.440–442, 1998.

[2] L. C. Freeman, “A set of measures of centrality based on betweenness,” *Sociometry*, vol.40, no.1, pp.35–41, 1977.

[3] U. Brandes, “A faster algorithm for betweenness centrality,” *Journal of Mathematical Sociology*, vol.25, no.2, pp.163–177, 2001.

[4] E. N. Gilbert, “Random graphs,” *The Annals of Mathematical Statistics*, vol.30, no.4, pp.1141–1144, 1959.

[5] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol.286, no.5439, pp.509–512, 1999.