Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 154172, 6 pages
http://dx.doi.org/10.1155/2014/154172

Research Article

A Multilevel Simplification Algorithm for Computing the Average Shortest-Path Length of Scale-Free Complex Network

Guoyong Mao[1] and Ning Zhang[2]

[1] *Department of Electronic Information and Electric Engineering, Changzhou Institute of Technology, Changzhou 213002, China*
[2] *Business School, University of Shanghai for Science and Technology, Shanghai 200093, China*

Correspondence should be addressed to Guoyong Mao; gymao@mail.shu.edu.cn

Received 20 March 2014; Accepted 18 May 2014; Published 2 June 2014

Academic Editor: Roberto Barrio

Computing the average shortest-path length (ASPL) of a large scale-free network needs much memory space and computation time. Based on the feature of scale-free network, we present a simplification algorithm by cutting the suspension points and the connected edges; the ASPL of the original network can be computed through that of the simplified network. We also present a multilevel simplification algorithm to get ASPL of the original network directly from that of the multisimplified network. Our experiment shows that these algorithms require less memory space and time in computing the ASPL of scale-free network, which makes it possible to analyze large networks that were previously impossible due to memory limitations.

1. Introduction

The research on complex network is developing very fast and significant achievements have been made in the past decade [1–4]. Two well-known and much studied classes of complex networks are scale-free networks [5, 6] and small-world networks. A scale-free network is a complex network or a connected graph with the property that the number of links originating from a given node exhibits a power law distribution. In a scale-free network, very few vertices have a huge number of connections, while a large proportion of vertices have only one input edge. Many networks are conjectured to be scale-free, including World Wide Web links, biological networks, and social networks.

Like other networks, specific structural features can characterize a scale-free network. The most important 3 features are degree distribution, clustering coefficient, and average shortest-path length (ASPL). Average shortest-path length is a concept in network topology that is defined as the average number of steps along the shortest paths for all possible pairs of network nodes. It is a measure of the efficiency of information or mass transport on a network. Some examples are the average number of clicks which will lead you from one website to another or the number of people you will have to communicate through on average, to contact a complete stranger.

The average shortest-path length is defined as follows. Consider an unweighted network G with the set of vertices V. Let $d(v_1, v_2)$ denote the shortest distance between v_1 and v_2 ($v_1, v_2 \in V$). Assume that $\text{dist}(v_1, v_2) = 0$ if $v_1 = v_2$ or v_2 cannot be reached from v_1, $\text{has\_path}(v_1, v_2) = 0$ if $v_1 = v_2$ or if there is no shortest path from v_1 to v_2, and $\text{has\_path}(v_1, v_2) = 1$ if there is a shortest path from v_1 to v_2; then the average shortest-path length ASPL(G) is

$$\text{ASPL}(G) = \frac{\sum_{i,j}^{N} \text{dist}\left(v_i, v_j\right)}{\sum_{i,j}^{N} \text{has\_path}\left(v_i, v_j\right)}, \tag{1}$$

where N is the number of vertices in G, $\sum_{i,j}^{N} \text{dist}(v_i, v_j)$ is the value of all-pairs-shortest-path length of graph G, and $\sum_{i,j}^{n} \text{has\_path}(v_i, v_j)$ is the number of shortest paths that exist in graph G.

For unweighted directed network, the time is $O(N * (N + E))$ for computing the all-pairs-shortest-path length using breadth-first-search algorithm [7]. If the network is represented by an adjacency list, then it occupies $O(N + E)$ space in memory, while an adjacency matrix representation

occupies $O(N^2)$, where E is the number of edges in G. If the network has millions of nodes or edges, then the time needed in computing will be unbearable; the memory requirement is also beyond the limit of most computers.

For the accurate computation of ASPL of large-scale network, people used to seek help from parallel computing [8, 9]. For example, from (1), we know that all-pairs-shortest-path length is the sum of single-source-shortest-path length (SSSPL) of each node in G; suppose there are N nodes in G and P processors available for computing; then one processor is responsible for the computing of SSSPL of N/P nodes. However, the parallel computing of SSSPL means that all nodes in G should be considered as source nodes or target nodes and the computation has to be done on the full network. In other words, G has to be fully loaded into the memory of each computing processor, which means that parallel computing cannot be applied if the network is too big to be fitted into memory.

In this paper, we describe how the computation of ASPL of a scale-free network can be done using only the simplified network. In our approach, the original network can be simplified multiple times to reduce the scale of network, which makes it possible to analyze large networks that were previously impossible due to memory limitations.

The remainder of this paper is organized as follows. The idea of the simplification algorithm is discussed in Section 2 followed by the implementation of the simplification algorithm in Section 3. In Section 4, we discuss the multilevel simplification algorithm followed by the analysis of the simplification algorithm in Section 5. Finally, in Section 6, we summarize our findings and the main contributions.

2. Idea of Simplification

As mentioned in the previous section, the scale-free network has the feature that very few vertices have a huge number of connections, while a large proportion of vertices have only one input edge and no output edge. These vertices are called suspension vertices or suspension nodes. For example, the China Research and Education Network (CERNET) is a scale-free network; it has 366406 nodes and 540750 edges in 2005 [10, 11]; among them 308958 nodes are suspension nodes, accounting for about 84% of the total number of nodes. If all these suspension nodes and the edges connecting them are removed, then there are only 57448 nodes and 231792 edges left; the network will be significantly simplified. If we can get the ASPL of the original network from that of the simplified network, then the memory requirement to load the network will be significantly reduced.

Let us illustrate the idea of the simplification algorithm using Figure 1 and Table 1. In Figure 1, node 5, node 6, and node 8 are suspension nodes; node 2 is the only node that connects node 5 and node 6. Node 7 is the only node that connects node 8. The SSSPL of each node is listed in Table 1. There is a shortest path that starts from node i to node j if the shortest path length from node i to node j is bigger than 0. The ASPL of the whole network is the sum of the SSSPL of each node divided by the sum of the number of the shortest paths started from each node. For example, the sum of SSSPL of each node in the original network G is 63; the sum of the shortest paths starting from each node in the original network is 29.

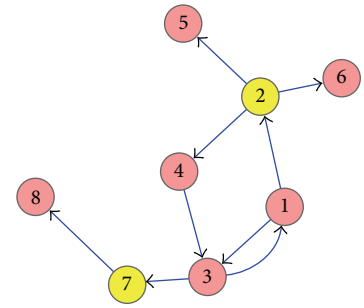

Figure 1: A simple directed network with 3 suspension nodes and 2 connectors.

We name the nodes that connect the suspension nodes as connectors. If all 3 suspension nodes in Figure 1 are removed, then the 3 edges starting from the connector to these nodes are also removed which means the shortest paths starting from other nodes to these suspension nodes are removed. The 6th, 7th, and the 9th columns are deleted from Table 1 if the corresponding node 5, node 6, and node 8 are removed. The all-pairs-shortest path is changed into 30 and the number of shortest paths is changed into 16 as shown in the right 2 columns in Table 1.

For each connector c, if the suspension node that it connects and the edge starting from the connector to the suspension node are deleted, then the all-pairs-shortest-path length will be reduced by

$$\sum_{(k\in G', k<>c)} (\text{length}(k,c) + i) + 1, \tag{2}$$

where $i = 1$ if length$(k,c) > 0$, $i = 0$ if length$(k,c) = 0$, and length(k,c) is the shortest path length between k and c. G' is the simplified network.

Suppose there are M suspension nodes in G and the value of k is $N - M - 1$, which means all suspension nodes and the connector c are not used as the start node, because there is no shortest path from one node to itself and there is no shortest path originated from the suspension nodes.

Formula (2) can be explained as follows. For one suspension node, the shortest paths that lead to it must pass through a connector. If there is a shortest path that leads to this connector, then there must be a shortest path that leads to this suspension node. Therefore, the length of the shortest path that leads to a suspension node is the length from the connector to this suspension node, which is 1, plus the sum of length that started from the other nodes that belong to G' to this suspension node. The length to a connector plus 1 is the length to the suspension node if a shortest path exists between the node in G' and this connector, because one further step is needed to get to the suspension node from this connector; the length is 0 if there is no path existing between the node in G' and the connector.

Table 1: The SSSPL of each node and the ASPL of the original and simplified networks.

Node	1	2	3	4	5	6	7	8	SSSPL of node in G	Number of shortest paths in G	SSSPL of node in G'	Number of shortest paths in G'
1	—	1	1	2	2	2	2	3	13	7	6	4
2	3	—	2	1	1	1	3	4	15	7	9	4
3	1	2	—	3	3	3	1	2	15	7	7	4
4	2	3	1	—	4	4	2	3	19	7	8	4
5	0	0	0	0	—	0	0	0	0	0	—	
6	0	0	0	0	0	—	0	0	0	0	—	
7	0	0	0	0	0	0	—	1	1	1	0	
8	0	0	0	0	0	0		—	0	0	—	
Sum	Shortest path between nodes								63	29	30	16

Table 2: Length and number of shortest path reduced in simplification.

Node	2	2	7
1	1	1	2
2	—	—	3
3	2	2	1
4	3	3	2
7	0	0	—
$\sum_{(k\in G',k<>c)}$ length(k,c)	6	6	8
$\sum_{(k\in G',k<>c)} i$	3	3	4
$\sum_{(k\in G',k<>c)}$ (length$(k,c)+i)+1$	10	10	13
$\sum_{(k\in G',k<>c)} (i)+1$	4	4	5

For each connector c, if the suspension node that it connects and the edge starting from the connector to the suspension node are deleted, the number of all-pairs-shortest paths will be reduced by

$$\sum_{(k\in G',k<>c)} (i)+1, \tag{3}$$

where $i = 1$ if length$(k,c) > 0$, $i = 0$ if length$(k,c) = 0$, and length(k,c) is the shortest path length between k and c.

Formula (3) can be explained similar to formula (2). The only difference between these 2 formulas is that if there is a shortest path that starts from node k in G' to connector c, which means length$(k,c) > 0$, then the number of shortest paths between these 2 nodes is 1 and the number of shortest paths between k and the suspension node is also 1, as there is always a path between the connector c and the suspension node it connects. Therefore, the number of shortest paths that start from c to the suspension node is always 1.

From formulas (2) and (3) we can get the all-pairs-shortest-path length and the number of shortest paths. Figure 2 is the network with all suspension nodes removed from Figure 1; the all-pairs-shortest-path length of G' is 30; the number of shortest paths is 16. The reduced length and number of shortest paths by cutting the suspension nodes connected to each connector $(2, 2, 7)$ are illustrated in Table 2.

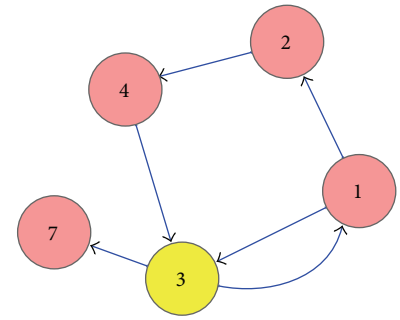

Figure 2: The network simplified from Figure 1.

From Table 2 we can see that the length of shortest paths that are reduced in simplification is 33(10 + 10 + 13), the number of shortest paths that are reduced is 13(4 + 4 + 5), and connector 2 is calculated twice as it connects 2 suspension nodes. Hence, the all-pairs-shortest-path length of G is 63; the number of shortest paths of G is 29.

3. Simplification Algorithm

Following the idea in Section 2, we simplify the original network G into G' by removing all the suspension nodes and edges, as listed in Algorithm 1.

In Algorithm 1, Length_of_G is the all-pairs-shortest-path length of G and Length_of_G' is the all-pairs-shortest-path length of G'. Number_of_path_of_G is the number of shortest paths of G, Number_of_Path_of_G' is the number of shortest paths of G', and Number_of_s_nodes is the number of suspension nodes in G. For the network as shown in Figure 1, the number is 3. Number_of_suspension_edges is the number of suspension edges that connect a connector and the suspension nodes. For connector 2, the number is 2.

```
(1)  for i in G'
(2)    for j in G':
(3)      if (i <> j):
(4)        Number_of_Path_of_G'++
(5)        Length = Shortest_Path_Length(i, j)
(6)        Length_of_G' += Length
(7)        if is_connector(j):
(8)          Number_of_Path_to_Connector += Number_of_suspension_edges_of(j)
(9)          Length_to_Connector += Length * Number_of_suspension_edges_of(j)
(10)  Length_of_G = Length_of_G' + Length_to_Connector + Number_of_Path_to_Connector + Number_of_s_nodes
(11)  Number_of_path_of_G = Number_of_Path_of_G' + Number_of_Path_to_Connector + Number_of_s_nodes
```

Algorithm 1: Calculate the ASPL of G from the simplified graph G'.

```
(1) depth = 1
(2) if is_connector(j):
(3)   Length_Reduced += Length_to_Connector + Number_of_Path_to_Connector + 1
(4)   Number_of_Path_Reduced += Number_of_Path_to_Connector + 1
(5)   if (connector → next == leaf) return
(6)   else
(7)     while (connector → next <> leaf)
(8)       depth++
(9)       Length_Reduced += Length_to_Connector + depth * Number_of_Path_to_Connector + depth + 1
(10)      Number_of_Path_Reduced += Number_of_Path_to_Connector + depth
(11)      connector = connector → next
```

Algorithm 2: Restore the ASPL of G from the multisimplified graph.

4. Multilevel Simplification

It is necessary to simplify G' again if there is still a lot of suspension nodes in G'. For example, node 7 is a suspension node in G', as shown in Figure 2. We can get G'' if we simplify G'. There are no suspension nodes in G''. There are two ways to get ASPL(G) from G''. One way is to work on G'' to get ASPL(G') from ASPL(G'') first and then work on G' to get ASPL(G) from ASPL(G'); the other way is to work on G'' only to get ASPL(G) directly from ASPL(G''). Obviously, the latter computation method is more efficient both in time and in space because G'' has less nodes and edges than G'. Therefore, we use the second method for simplification.

After the simplification of the first round, new suspension nodes will appear. Obviously, only the connectors can be the suspension nodes after the previous simplification, because we only remove the suspension nodes and the edges starting from the connectors to them. This implies that if a connector becomes a suspension node after simplification, then there is no shortest path that starts from this connector to other nodes before simplification, except the suspension node it connects. For example, node 7 will become a suspension node after the simplification of the first round; it has no shortest path to other nodes except node 8, which is a suspension node it connects. With this regular pattern, we can use the multisimplified network to get ASPL of the original network.

Let us still use the network shown in Figure 1 as an example. Node 5, node 6, and node 8 are removed in the first round of simplification; only node 7 is removed in the second round of simplification. We should remember that all connectors are in G'' now, because only G'' can be used in the second method. Therefore, node 3 is the only connector in the second round of simplification. If the suspension node that node 3 connects is removed, then the length of the shortest path is reduced by

$$\begin{aligned}&\text{Length\_Reduced}^{(2)}\\&\quad= \text{Length\_to\_Connector}\\&\qquad+ \text{Number\_of\_Path\_to\_Connector} + 1,\end{aligned} \tag{4}$$

where $\text{Length\_Reduced}^{(2)}$ is the length of shortest path reduced in the second round.

Node 7 is the suspension node in the second round; it is a connector in the first round. If the suspension node that node 7 connects is removed in the first round, then the length is reduced by $\text{Length\_Reduced}^{(1)}$, where

$$\begin{aligned}\text{Length\_Reduced}^{(1)} = {}&\text{Length\_Reduced}^{(2)}\\&+ \text{Number\_of\_Path\_to\_Connector}\\&+ 1 + 1.\end{aligned} \tag{5}$$

If we replace Length_Reduced$^{(2)}$ with formula (4), we get

$$\begin{aligned}&\text{Length\_Reduced}^{(1)}\\&\quad=\text{Length\_to\_Connector}\\&\qquad+2*\text{Number\_of\_Path\_to\_Connector}+3.\end{aligned}\tag{6}$$

Actually, formula (5) is the same as formula (4) because Length_Reduced$^{(2)}$ is the length to the node that will become the connector in the first round or the length to the new connector. In Figure 1, this node is node 7. The same reason can be given to explain Number_of_Path_to_Connector + 1, because one further step is needed from the connector in the second round to the node that will become the connector in the first round.

From formula (5) we know if a suspension node is removed in the second round, then the number of shortest path is reduced by Number_of_path_Reduced$^{(2)}$, where

$$\begin{aligned}&\text{Number\_of\_path\_Reduced}^{(2)}\\&\quad=\text{Number\_of\_path\_to\_connector}+1.\end{aligned}\tag{7}$$

The number of shortest paths is reduced by Number_of_path_Reduced$^{(1)}$ in the first round if the suspension node connected by the suspension node in the second round is removed, where

$$\begin{aligned}&\text{Number\_of\_path\_Reduced}^{(1)}\\&\quad=\text{Number\_of\_path\_Reduced}^{(2)}+1\end{aligned}\tag{8}$$

or

$$\begin{aligned}&\text{Number\_of\_path\_Reduced}^{(1)}\\&\quad=\text{Number\_of\_path\_to\_connector}+2.\end{aligned}\tag{9}$$

Formula (8) and formula (9) only apply to two-level simplification. For multilevel, the formulas are

$$\begin{aligned}&\text{Length\_Reduced}^{(1)}\\&\quad=\text{Length\_to\_Connector}\\&\qquad+i*\text{Number\_of\_Path\_to\_Connector}\\&\qquad+i+1,\\&\text{Number\_of\_path\_Reduced}^{(1)}\\&\quad=\text{Number\_of\_path\_to\_connector}+i,\end{aligned}\tag{10}$$

where i is the number of simplification levels.

The implementation of the multilevel simplification is done on the network after multiple simplifications; the connectors of different levels should be saved and processed separately. While in one-level simplification, only the connectors of the upper level are saved. We use an adjacency list to save multilevel connectors; the connectors of the lowest level are the start points in the list and they point to the connectors in the upper level that they connect. Here the lowest level connectors refer to those connectors that cannot be simplified again. For the network listed in Figure 1, the adjacency list is shown in Figure 3. The restore of ASPL of G from G'' started from the connectors in the lowest level, which are node 2, node 2, and node 3.

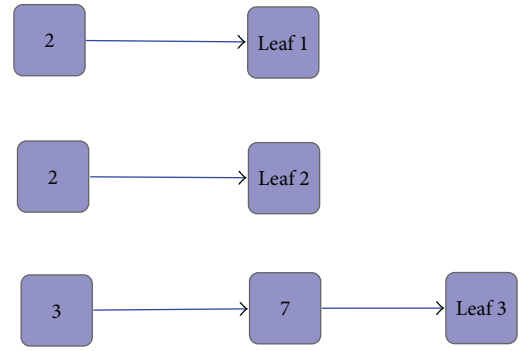

Figure 3: The adjacency list corresponding to Figure 1.

Suppose we already get the values of Number_of_Path_to_Connector and Length_to_Connector from the multisimplified graph, depth is the number of levels in the adjacency list and the restore algorithm is listed in Algorithm 2.

5. Analysis of Algorithms

From Algorithms 1 and 2 we can see that the computation of ASPL can be implemented fully on the simplified network. Suppose the number of nodes is reduced from N to n and the number of edges from E to e in the simplification. If adjacency matrix is used to store the network, then the space complexity is reduced by

$$\left(1-\frac{N^2-n^2}{N^2}\right)*100\%.\tag{11}$$

If adjacency list is used, then the space complexity is reduced by

$$\left(1-\frac{N*E-n*e}{N*E}\right)*100\%.\tag{12}$$

If 90% of the nodes and 50% of the edges are reduced, then the space complexity is reduced by 99% for the adjacency matrix storage format and 95% for the adjacency list storage format.

The time complexity is reduced from $O(N*(N+E))$ to $O(n*(n+e))$. Though extra computation is needed to restore ASPL of G from that of G' (from line 7 to line 11 in Algorithm 1), these computations are basic addition or multiplication operations that are negligible in time compared with that of computing the shortest path length between two nodes (line 5 in Algorithm 1).

6. Conclusion

The main contribution of this paper is the multilevel simplification algorithm to reduce the time and space complexities in

the computation of ASPL of scale-free networks; the property of Algorithm 2 can guarantee the efficiency for all scale-free networks.

In this paper, we presented the reason for network simplification, analyzed the feature of scale-free network, and illustrated the relation between the ASPL of the original network and that of the simplified network; detailed implementation algorithm is given to compute ASPL directly from the multiple-simplified networks so that the time and space complexities are significantly reduced.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by Natural Science Fund of china (70971089), Shanghai Leading Academic Discipline Project under Grant no. XTKX2012, and Jiangsu Overseas Research & Training Programs for University Prominent Young & Middle-Aged Teachers and Presidents. The authors are grateful to researches in German Research School for Simulation Science and the reviewers for their valuable comments and suggestions to improve the presentation of this paper.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási, "Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

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

[3] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–91, 2002.

[4] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[5] A.-L. Barabási, R. Albert, and H. Jeong, "Mean-field theory for scale-free random networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 272, no. 1, pp. 173–187, 1999.

[6] X. F. Wang and G. Chen, "Complex networks: small-world, scale-free and beyond," *IEEE Circuits and Systems Magazine*, vol. 3, no. 1, pp. 6–20, 2003.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[8] K. Madduri, D. A. Bader, W. B. Jonathan, and J. R. Crobak, "An experimental study of a parallel shortest path algorithm for solving large-scale graph instances," in *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments*, pp. 23–35, New Orleans, La, USA, January 2007.

[9] G. Mao and N. Zhang, "Analysis of average shortest-path length of scale-free network," *Journal of Applied Mathematics*, vol. 2013, Article ID 865643, 5 pages, 2013.

[10] N. Zhang, "Demonstration of complex network: CERNET," *Journal of Systems Engineering*, vol. 21, no. 4, pp. 337–340, 2006.

[11] G. Mao and N. Zhang, "Study of evolution model of China education and research network," *Mathematical Problem in Engineering*, vol. 2013, Article ID 808901, 6 pages, 2013.