Connectivity Determination in Heterogeneous Random Graphs

Project Report of Mobile Internet Course

Jiaru Zhang
515030910560

***Abstract*— This is the report of the project of course Mobile Internet. In this project, I studied the strategy to determine the connectivity in random graphs. I proposed two algorithms to deal with the global connectivity problem and K-vertices connectivity problem, with the method of finding smallest cut in the graphs. These two algorithms are useful and better than original ones under some conditions.**

I. Introduction

The study on random graphs has been started long time ago. A popolar model of random graphs is $G(n,p)$ random model, proposed by Gilbert in 1959 [1]. Erdos and Renyi, gave the equivalence between $G(m,n)$ graph and $G(n,p)$ model [2]. The probability of connectivity on such graphs has been studied thoroughly [3] [4].

However, there is still much to explore the strategy on how to determine the connectivity. In these problems, the edges usually exist with some probability p and the cost the detect is c. L. Fu studied the s-t connectivity when p and c are constants on each edges [5]. X. Fu generalized the problem to the condition where p and c are variable between edges [6]. However, the above works only focus on $s-t$ connectivity. The problem I focus is different with them. In this project report, I discuss a improved algorithm of determining the global connectivity on $G(1,0,p)$ random graph. I also propose a generalized random algorithm to determine the connectivity on K vertices rather than all vertices.

My contributions of the problem are summarized as two parts:

- I improved the original algorithm on determining the global connectivity of $G(1,0,p)$ random graphs. It can use information repeatedly rather than using only once. So the time complexity is better than before.
- I generalized the random algorithm from Karger [7] on finding the smallest cut on K vertices. So we can determine the connectivity of K vertices with better time complexity.

II. Related Works

Connectivity determination $G(p,c)$ graphs are some $G(n,p)$ graphs where the cost to detect one edge is a constant c. There is a efficient strategy to find a strategy on $G(p,c)$ graphs with the time complexity of $30log_2 n$ to judge which edge to detect [5].

Smallest cut. The algorithm of strategy to judge connectivity is closely related with smallest cut problem in a graph. A widely-known algorithm to find the global smallest cut is proposed by M.STOER and F.WAGNER [8]. It can find a global smallest cut definitely in $O(V^2E + V^3logV)$ time. Karger provided a random algorithm to find a global smallest cut [7]. The time complexity is $O(E)$ with success probability of over $\frac{1}{n^2}$. Another algorithm to find the smallest s-t cut is proposed by L. R. Ford, Jr. and D. R. Fulkerson [9].

III. Improved Algorithms

A. Basic Idea

I summarized three basic algorithms to determine connectivity in random graphs. They are shown in Algorithm 1, 2 and 3 and are all based on the equivalence property between determining which edge to detect and finding the smallest cut many times. However, the time complexities are too high and difficult to meet requirements. The algorithm proposed by L.Fu etal. [5] is an improvement of Algorithm 1. I will show an improved algorithm of Algorithm 2 and some discussions of Algorithm 3 in next sections.

B. Improved algorithm of Algorithm 2

The time complexity of Algorithm 2 is $O(V^2E + V^3logV)$, where $O(VE + V^2logV)$ is the time com-

Algorithm 1: s-t Connectivity

Input : $G(1, 0, p)$
Output: 0: unconnected
1: connected

repeat
  Merge all known connected vertices ;
  Find the smallest s-t cut (Ford - Fulkson Algorithm) ;
  for *each edge in the s-t cut* **do**
    Detect: if connected: turn to step 2
  end
  return 0 if all edges in the cut are unconnected
until *There is only one node in the graph*;
return 1

Algorithm 2: Global Connectivity

Input : $G(1, 0, p)$
s and t
Output: 0: unconnected
1: connected

repeat
  Merge all known connected vertices ;
  Find the global smallest cut (Stoer-Wagner Algorithm) ;
  for *each edge in the smallest cut* **do**
    Detect: if connected: turn to step 2
  end
  return 0 if all edges in the cut are unconnected
until *s and t are in the same super-vertex*;
return 1

plexity of Stoer-Wanger algorithm and it runs V times totally.

By carefully analyzing the running process of SW algorithm, I find there is so many extra operations in it. In each run of the algorithm, we can not only find the global smallest cut but also some Ss, Ts and their smallest $S-T$ cuts where Ss and Ts are sets of vertices. However, in next run the algorithm does not use this information at all.

Here is the concrete description of the algorithm. Concretely, we got $(n1)$ Ss and Ts in one SW algorithm where S and T are both sets of vertices, and correspond $S-T$ cut edges weight W. In next time, the only change between the two graphs is the mergence of

Algorithm 3: K-vertices Connectivity

Input : $G(1, 0, p)$
K vertices
Output: 0: unconnected
1: connected

repeat
  Merge all known connected vertices ;
  Find K-vertices smallest cut (($K-1$ times s-t cut)) ;
  for *each edge in the smallest cut* **do**
    Detect: if connected: turn to step 2
  end
  return 0 if all edges in the cut are unconnected
until *all of K vertices are in the same super-vertex*;
return 1

two vertices which are in the smallest $S-T$ cuts. And we know some edges in the smallest $S-T$ cut are unconnected. So we could use them directly without any detection.

For every known smallest $S-T$ cut, if they are in the same part of the last smallest $S-T$ cut, we can first detect all of $s-t$ cut in them, and the merge the two vertices set. The size of graph will decrease $|s|+|T|+1$. So

$$O(n) = O(n - |S| - |T| + 1) + \sum_{i=1}^{|S|}\sum_{j=1}^{|T|}\binom{|S|}{i}\binom{|T|}{j}O_{s-t}(n - |S| - |T| + 2)$$

where O_{s-t} means the time complexity of $s-t$ cut.

In particular, when $|S| = 1$ or $|T| = 1$,

$$O(n) = O(n - |T|) + \sum_{i=1}^{|T|}\binom{|T|}{j}O_{s-t}(n - |T| + 1)$$

More particularly, if $|S| = |T| = 1$, which means we have known a smallest $s-t$ cut, we have

$$O(n) = O(n - 1)$$

which means we can merge s and t directly without running S-W algorithm in the most ideal situation.

C. An improved random algorithm for K-vertices connectivity

To determine the connectivity of K vertices, a natural idea is to improve the algorithm to find smallest cut.

A natural idea of this problem is to use SW algorithm to find smallest global cut between K vertices. However, the algorithm does not keep effective even in the condition of $K = 2$. Recently, the only definite algorithm to solve the problem is to run $s-t$ algorithm for $(K-1)$ times, as shown in Algorithm 3. The total time complexity is

$$O(n) = (K-1)O_{s-t}(n)$$

Here I give a random algorithm to find smallest cut in K vertices shown in Algorithm 4 based on Karger algorithm [7].

Algorithm 4: Random algorithm for smallest cut of K vertices

Input : $G(1, 0, p)$
K vertices
Output: Smallest cut for K vertices

Denote set of other vertices as S_{nk} ;
For $\forall v \in S_{nk}$ with $Degree(v) = 1$ remove v ;
For $\forall v \in S_{nk}$ with $Degree(v) = 2$ replace v and its edges with the smaller one;
repeat
    Merge two vertices
until *all of K vertices are in two super-vertices*;
return smallest cut

The two operations are used to reduce the number of vertices in the graph. The principle of them is shown as follows.

Theorem 1 *The two operations do not change the result.*

proof. If the operation 1 changes the result, it means that the smallest cut contains the edge. However, if we remove the edge in the cut, it is also a cut of K vertices with smaller weights. This is a contradiction because we known the original one is the smallest cut.

If the operation 2 changes the result, it means the smallest cut contains another edge or both. If it contains both, we can remove the two edges. Or we can replace the edge with the smaller one. The new cut will also be a cut with smaller weights. This is a contradiction because we known the original one is the smallest cut. ■

Theorem 2 *The time complexity of the algorithm is $O(E)$ per run.*

proof. It is easy to prove. There are only E edges in the graph so the algorithm will run for at most E cycles. ■

Theorem 3 *It can find the smallest K-vertices cut with probability of over $(\frac{1}{K^2})$.*

proof. Suppose there are c edges in the smallest cut totally. So the degrees of K vertices are at least c because if not we can use the vertex as s and the cut will be its degree. So the total number of edges in the graph is at least $\frac{K \times c}{2}$. This verdict holds true in the procedure of merging because the degree of super-vertex is the degree of the set and other vertices. In the process we cannot choose the specific c edges because they are smallest cut. The probability of not choosing them is $P = 1 - \frac{c}{e} \geq 1 - \frac{2}{k}$. The number of vertices decrease one after merging. So the probability of not choosing all c edges in the whole procedure is

$$\begin{aligned} P &\geq (1 - \frac{2}{k}) \times (1 - \frac{2}{k-1}) \times \cdots \times (1 - \frac{2}{3}) \\ &\geq \frac{1}{K^2} \end{aligned}$$

■

The success rate is not high. However, if we run the algorithm t times, the total successful rate will be

$$Successrate = 1 - (1 - \frac{1}{K^2})^a$$

For example, if we run the algorithm K^2 times, the success rate will be $1 - \frac{1}{e}$. Furthermore, for $K^2 logK$ times run, the success rate can reach $1 - \frac{1}{K}$. Considering K is usually a large number in real problems, it is usually an acceptable rate for real application.

IV. Conclusion

In this project, I have studied the problem of optimally determining global connectivity and K-connectivity of random networks. These algorithms are based on a simple proposition: The equivalence property between determining which edge to detect and finding the smallest cut many times. Thus I derived the two useful algorithms.

References

[1] E. N. Gilbert, "Random graphs," *Ann. Math. Statist.*, vol. 30, no. 4, pp. 1141–1144, 12 1959. [Online]. Available: https://doi.org/10.1214/aoms/1177706098

[2] P. Erds and A. Rnyi, "On the evolution of random graphs," *Transactions of the American Mathematical Society*, vol. 286, no. 1, pp. 257–274, 2012.

[3] M. D. Penrose, "The longest edge of the random minimal spanning tree," *Annals of Applied Probability*, vol. 7, no. 2, pp. 340–361, 1997.
[4] E. N. Gilbert, "Random plane networks," *Journal of the Society for Industrial & Applied Mathematics*, vol. 9, no. 4, pp. 533–543, 1961.
[5] L. Fu, X. Wang, and P. R. Kumar, "Are We Connected? Optimal Determination of Source-Destination Connectivity in Random Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 751–764, 2017.
[6] X. Fu, Z. Xu, Q. Peng, L. Fu, and X. Wang, "Complexity vs. optimality: Unraveling source-destination connection in uncertain graphs," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.
[7] D. R. Karger, "Global Min-cuts in RN C , and Other Rami cations of a Simple Min-Cut Algorithm," 1992.
[8] M. Stoer, "A simple min-cut algorithm," *Journal of the Acm*, vol. 44, no. 4, pp. 585–591, 1997.
[9] L. R. Ford and D. R. Fulkerson, "A suggested computation for maximal multi-commodity network flows," *Management Science*, vol. 5, no. 1, pp. 97–101, 1958.