Work Report:
Random Graph Connectivity Detection Algorithms &Power-law Random Graph Generation

Qianyang Peng

2016 年 6 月 10 日

This is a report introducing the work I have done this semester. This semester my experience in the social network middle group mainly contains two parts. The first is to listen to reports and read papers in order to get basic knowledge about the topic researchers cares about, and secondly I involved in some theoretical topics, helped members refine the feasibility of their algorithms and done the implementation work.

My implementation of the algorithms, in other word, the programs, contains mainly three works:

1. A NP algorithm for the optimal solution to detect the connectivity of both directed and undirected random graphs.
2. An approximation algorithm to calculate the approximate solution to detect the connectivity of a undirected random graph.
3. An intuitive linear algorithm to generate a power-law random graph.

The first two works is mainly based on the theoretical studies by *Fu Xinzhe* and *Xu Zhiying*, and the third work is based my own idea. I will next introduce the first two works. The third work's problem formulation is written separately in another document. As this is not a mature work yet, it is not contained in this version's report.

1 First work

Fu Xinzhe and *Xu Zhiying* is aiming on a nice algorithm to determine the connectivity between two nodes of a random graph. And they need a benchmark algorithm to detect the optimal algorithm which is having the minimum expected cost and to intuitively inspect what is happening during the determination of the optimal solution.

Initially, *Fu Xinzhe* proposes a algorithm base on dynamic programming, however it is having some difficulty on realization. The main difficulty lies on indexing graph into a list and transforming the former state to the latter. Basing on the same basic thought, I solved this problem using an algorithm combining recursion and dictionary.

Using python library *Networkx* and *Matplotlib*, the code can be implemented very laconic and efficiently. Graph of the result can be drawn with the help of *pyplot* and can be shown nice and clean. Moreover, this algorithm has a good extendibility and can be implemented into a version using a directed graph, without changing most part of the code.

2 Second work

My partners proposed approximate algorithm to this problem, which is the main contribution of them to this area of study, and I took the bound to make a realization to that algorithm. The three main difficulty in solving that problem has been introduced in the PPT slices. And here I introduce some details:

1. To calculate the number of paths and number of cuts between two nodes of a specified graph, some past studies and textbooks has given nice algorithms with relatively low complexity. My implementation of these two calculation is based on two references.[1][2]. Polynomial algorithms to these two basic problem guarantees out algorithm to have polynomial algorithm, and the efficiency of these basic problems is the bottleneck of the efficiency of the whole algorithm.

2. In this problem a graph is having three states, existing, undetected and nonexist. And basically the library *NetworkX* only support to build a graph with two states(exist or not). As we need to use many convenient features of *NetworkX*, I used a class to wrap two *NetworkX* graphs in it, and implemented algorithms in the wrapper class to realize the functions we need to solve this problem. Both the first and second work can use the same wrapper class and use the same API in them, which shows the high reusability of the code.

3 Future work

The future work lies in three parts:

1. Read papers in recent years, and find some application our newly formed algorithm can be applied on.

2. Further refine the algorithm of the second work, try to make it work in directed graphs. Currently the algorithm to count the number of cuts can not be applied to directed graphs.

3. Some more contrast algorithms are need to be implemented to compare their performance.

References

[1] R. Sedgewick, "Algorithms in c, part 5: Graph algorithms,"

[2] S. Tsukiyama, I. Shirakawa, and H. Ozaki, "An algorithm to enumerate all cutsets of a graph in linear time per cutset,"