A Short Introduction to Local Graph Clustering Methods and Software

Kimon Fountoulakis[1], David F. Gleich[2], and Michael W. Mahoney[3]

[1] University of Waterloo, Dept. of Computer Science, Waterloo ON N2L 3G1, Canada
[2] Purdue University, Dept. of Computer Science, West Lafayette, IN, USA
[3] University of California Berkeley, ICSI and Dept. of Statistics, Berkeley, CA, USA

Graph clustering has many important applications in computing, but due to the increasing sizes of graphs, even traditionally fast clustering methods can be computationally expensive for real-world graphs of interest. Scalability problems led to the development of local graph clustering algorithms that come with a variety of theoretical guarantees [1]. Rather than return a global clustering of the entire graph, local clustering algorithms return a single cluster around a given seed node or set of seed nodes. These algorithms improve scalability because they use time and memory resources that depend only on the size of the cluster returned, instead of the size of the input graph. Indeed, for many of them, their running time grows linearly with the size of the output.

In addition to scalability arguments, local graph clustering algorithms have proven to be very useful for identifying and interpreting small-scale and meso-scale structure in large-scale graphs [2, 3]. As opposed to heuristic operational procedures, this class of algorithms comes with strong algorithmic and statistical theory. These include statistical guarantees that prove they have *implicit* regularization properties [4, 5].

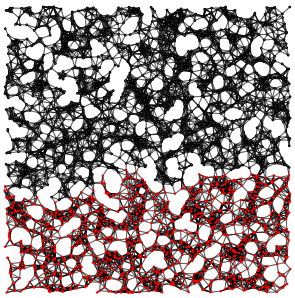
One of the challenges with the existing literature on these approaches is that they are published in a wide variety of areas, including theoretical computer science, statistics, data science, and mathematics. This has made it difficult to relate the various algorithms and ideas together into a cohesive whole. We have recently been working on unifying these diverse perspectives through the lens of optimization [6] as well as providing software to perform these computations in a cohesive fashion [7]. In this note, we provide a brief introduction to local graph clustering, we provide some representative examples of our perspective, and we introduce our software named Local Graph Clustering (LGC).

Local graph clustering

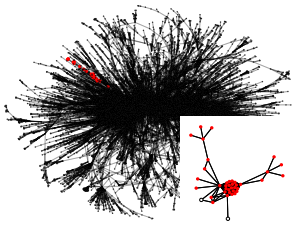
Given a seed node, or a seed set of nodes, the goal of local graph clustering is to compute a cluster "nearby" the seed that is related to the "best" cluster nearby the seed. Here, "best" and "nearby" are intentionally left under-specified, as they can be formalized in one of a few different but related ways. For example, "best" is usually related to a clustering score such as conductance. Formally, local graph clustering can be easily understood as a recovery problem. One assumes that there exists a target cluster in a given graph and the objective is to recover it from one or more example vertices inside the set. We can be more precise for a formulation involving conductance. Assume that there exists a target cluster B with conductance ϕ_T and we have one seed node in B, our objective is to find a cluster A that resembles B with conductance bounded by some function of ϕ_T, where resemblance is captured here by normalized precision and recall. Moreover, we want to do this in running time and memory proportional to the size of A.

COMPLEX NETWORKS

The 7^{th} International Conference on Complex Networks and Their Applications. 11 - 13 Dec., 2018, Cambridge (UK)



The near-optimal conductance solution for the random geometric graph bisects the graph into two large well-balanced pieces.



The near-optimal conductance solution for a typical data graph. (Inset. A zoomed view of the subgraph where two unfilled nodes are the border with the rest of the graph.)

Fig. 1. (Adapted from [6].) A motivation for local graph clustering. In graphs with an underlying geometry (left), a good partition is a near bisection. In most data graphs (right) [2, 8], the good clusters and communities are small. It is computationally ineffective to find the data-graph clusters using techniques whose runtime depends on the entire graph.

As a quick example of why local graph analysis is frequently useful in data science applications, we present in Figure 1 the results of finding a good partition of both a random geometric graph and a more typical graph from machine learning and data science. In the random geometric graph, the size of best cluster or community is about half the graph. In this case, an algorithm with runtime that scales with the size of the graph is reasonable. In the graph that is more typical of machine learning and data science applications, the best cluster or community derived from a conductance metric is an exceedingly small fraction of the network. This means that standard graph algorithms whose runtime depends on the size of the graph will do an enormous amount of work to return a tiny portion of the graph. Many other examples of this general phenomenon can be found [2, 3, 9, 10]. Local graph clustering techniques can find this cluster (and other similar clusters that happen not to be globally optimal) while touching not many more edges and nodes than are in the output cluster, greatly reducing the computation time.

Our Software

Local Graph Clustering (LGC) is a Python package that uses C++ routines and brings scalable graph analytics on your laptop. In particular, LGC provides methods that find local clusters, methods that improve a given cluster, tools to compute network community profiles, and multi-class label prediction. The software is on GitHub [7].

Methods in LGC. LGC implements seven local graph clustering methods. Three spectral methods, i.e., approximate PageRank [1], PageRank Nibble [1], ℓ_1-regularized PageRank [4], and four flow methods, i.e., Max-flow Quotient-cut Improvement [11], FlowImprove [12], SimpleLocal [13] and Capacity Releasing Diffusion [14].

Pipelines. In LGC one can find pipelines which employ the above methods to compute network community profiles (NCPs) [9]. An NCP is a plot that is defined as the the quality of the "best" community as a function of community size in a network. To measure the quality of a community we use conductance. Computing the NCP of a graph is an NP-hard problem, and therefore we compute an approximate version of it using

local graph clustering methods. The same approximation has been suggested also in [2, 9]. LGC also implements multi-class label prediction using local graph clustering as a workhorse [15].

Scalability. LGC offers routines to work with graphs that scale to the available memory of your system. We have used these routines to study graphs with billions of nodes on large memory machines, and graphs with 117 million edges on a laptop computer by using about 9.4 GB RAM. Examples can be found in the GitHub repository [7].

Acknowledgements. This material is based on research sponsored by DARPA and the Air Force Research Laboratory (AFRL) under agreement number FA8750-17-2-0122, as well as the Army Research Office (ARO). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and the AFRL or the U.S. Government.

References

1. Andersen, R., Chung, F., Lang, K.: Local Graph Partitioning using PageRank Vectors, Proc. of FOCS, 475–486 (2006)
2. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M. W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, Internet Mathematics, 6(1), 29–123 (2009)
3. Jeub, L. G. S., Balachandran, P., Porter, M. A., Mucha, P. J., Mahoney, M. W.: Think locally, act locally: Detection of small, medium-sized, and large communities in large networks, Phys. Rev. E, 91, 012821 (2015)
4. Fountoulakis, K., Roosta-Khorasani, K., Shun, J., Cheng, X., Mahoney, M. W., Variational Perspective on Local Graph Clustering, Mathematical Programming B, 1–21 (2017)
5. Gleich, D. F., Mahoney, M. W.: An Anti-differentiating approximation algorithms:A case study with min-cuts, spectral, and flow, Proc. of ICML, 1018–1025 (2014)
6. Fountoulakis, K., Gleich, D. F., Mahoney, M. W.: An Optimization Approach to Locally-Biased Graph Algorithms, Proceedings of the IEEE, 105(2), 256–272 (2017)
7. Fountoulakis, K., Meng, L., Gleich, D. F., Mahoney, M. W.: Local Graph Clustering (2018), GitHub repository https://github.com/kfoynt/LocalGraphClustering
8. Boguñá, M., Pastor-Satorras, R., Díaz-Guilera, A., Arenas, A.: Models of social networks based on social distance attachment, Phys. Rev. E, 70(5), 056122 (2004)
9. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M. W.: Statistical properties of community structure in large social and information networks, Proc. of WWW, 695–704 (2008)
10. Leskovec, J., Lang, A., Mahoney, M. W.: Empirical comparison of algorithms for network community detection, Proc. of WWW, 631–640 (2010)
11. Lang, K., Rao, S.: A Flow-Based Method for Improving the Expansion or Conductance of Graph Cuts, Proc. of IPCO, 3064 (2004)
12. Andersen, R., Lang, K.: An algorithm for improving graph partitions, Proc. of SODA, 651–660 (2008)
13. Veldt, N., Gleich, D. F., Mahoney, M. W.: A simple and strongly-local flow-based method for cut improvement, Proc. of ICML, 48, 1938–1947 (2016)
14. Wang, D., Fountoulakis, K., Henzinger, M., Mahoney, M. W., Rao, S.: Capacity Releasing Diffusion for Speed and Locality. Proc. of ICML, 3598–3607 (2017)
15. Gleich, D. F., Mahoney, M. W.: Using Local Spectral Methods to Robustify Graph-Based Learning Algorithms, Proc. of KDD, 359–368 (2015)
16. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth, Knowledge and Information Systems, 42(1), 181–213 (2013)