Extended Abstract: K-Truss Implementation in Arkouda

Joseph Patchett
New Jersey Institute of Technology
Newark, New Jersey 07102
Email: jtp47@njit.edu

Zhihui Du
New Jersey Institute of Technology
Newark, New Jersey 07102
Email: zhihui.du@njit.edu

David Bader
New Jersey Institute of Technology
Newark, New Jersey 07102
Email: bader@njit.edu

1. Introduction

The k-Truss algorithm is the decomposition of a graph into cliques where every edge is incident to $k-2$ triangles. The results of this decomposition have implications in national security, social media, and community identification. These results can be used to interpret how connected a community is e.g. how interconnected a group on social media is. Higher values of k imply a higher degree of connection. The maximal k-truss is the subgraph for which there is the largest k. For certain graphs like those in social media the maximal k-Truss is not informative, therefore our algorithm allows different choices of k to tailor to the needs of each use case.

Our implementation of the k-Truss algorithm in Arkouda finds all maximal k-trusses in a graph. Arkouda [1], [2] is an open-source framework for large scale graph analytics that allows users to access a powerful server driven backend from their own personal computer. Real-world data and the graphs resulting from that data have become increasingly complex. Arkouda utilizes a powerful Chapel [3] back-end, ZeroMQ [4] for message communication, and a Python [5] front-end for users. Computations that require complex calculations and large networks will be handled on the back-end abstracted from the user. Our major contribution is extending the abilities of Arkouda and k-truss is a typical graph algorithm for such a framework.

2. Approach

We build off of previous work done on Arkouda, the inherent parallel methods in Chapel, and work on k-Truss implementations. Our algorithm works for undirected graphs that are maintained on an Arkouda server. We use the double index edge structure to efficiently iterate and maintain the triangle counts for each edge. This structure is a list of source and destination arrays and allows for a quick reference for each edge. For certain graphs like those in social media the maximal k-Truss is not informative, therefore our algorithm allows different choices of k. This allows all manners of users to work with and interpret the results.

The steps are as follows.

1) Using the double index edge structure, triangles incident to each edge are counted. Edges with fewer than $k-2$ triangles are passed into the frontier.
2) For each edge, $\langle u, v\rangle$, in the frontier, we decrement the triangle count for all edges, $\langle u, w\rangle$ and $\langle v, w\rangle$.
3) If those edges are now incident to a natural number fewer than $k-2$ triangles then these edges are added to the frontier.
4) This is continued until all edges incident to fewer than $k-2$ triangles have been iterated through and updated.
5) The subgraph with all edges incident to greater than or equal to $k-2$ is returned to the user.

For the user this approach has several benefits over traditional methods for k-truss: not only are all large scale calculations handled on the server side, but also actual graph data is maintained server side. Users can tackle graph data science problems orders of magnitude beyond what they could on their personal machines.

In further work we seek to expand the already impressive capabilities of Arkouda. We look to add additional functionality to handle directed graphs and give options to return the distinct residual subgraphs created by k-truss.

Acknowledgments

The authors would like to thank Michael Merrill and William Reus and the rest of the Arkouda Team. This research was funded in part by NSF grant number CCF-2109988.

References

[1] M. Merrill, W. Reus, and T. Neumann, "Arkouda: interactive data exploration backed by Chapel," in *Proceedings of the ACM SIGPLAN 6th on Chapel Implementers and Users Workshop*, 2019, pp. 28–28.

[2] W. Reus, "CHIUW 2020 Keynote: Arkouda: Chapel-Powered, Interactive Supercomputing for Data Science," in *Chapel Implementers and Users Workshop, 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2020, pp. 650–650.

[3] B. L. Chamberlain, E. Ronaghan, B. Albrecht, L. Duncan, M. Ferguson, B. Harshbarger, D. Iten, D. Keaton, V. Litvinov, P. Sahabu *et al.*, "Chapel comes of age: Making scalable programming productive," *Cray User Group*, 2018.

[4] P. Hintjens, *ZeroMQ: messaging for many applications*. O'Reilly Media, Inc., 2013.

[5] G. Rossum, *Python reference manual*. CWI (Centre for Mathematics and Computer Science), 1995.

978-1-6654-2369-4/21/$31.00 ©2021 IEEE