Fast Hierarchy Construction for Dense Subgraphs

Ahmet Erdem Sarıyüce[*] Ali Pinar[*]

Abstract

Discovering dense subgraphs and understanding the relations among them is a fundamental problem in graph mining. We want to not only identify dense subgraphs, but also build a hierarchy among them (e.g., larger but sparser subgraphs formed by two smaller dense subgraphs). Peeling algorithms (k-core, k-truss, and nucleus decomposition) have been effective to locate many dense subgraphs. However, constructing a hierarchical representation of density structure, even correctly computing the connected k-cores and k-trusses, have been mostly overlooked. Keeping track of connected components during peeling requires an additional traversal operation, which is as expensive as the peeling process. We propose efficient and generic algorithms to construct the hierarchy of dense subgraphs for k-core, k-truss, or any nucleus decomposition. Our algorithms leverage the disjoint-set forest data structure to efficiently construct the hierarchy during traversal. Furthermore, we introduce a new idea to avoid traversal. We construct the subgraphs while visiting neighborhoods in the peeling process, and build the relations to previously constructed subgraphs. We also consider an existing idea to find the k-core hierarchy and adapt for our objectives efficiently. Experiments on different types of large scale real-world networks show significant speedups over naive algorithms and existing alternatives.

1 Dense Subgraphs

Graphs are used to model relationships in many applications such as sociology, the WWW, cybersecurity, bioinformatics, and infrastructure. Although the real-world graphs are sparse ($|E| << |V|^2$), vertex neighborhoods are dense [9]. Clustering coefficients [21], and transitivity [20] of real-world networks are also high and suggest the micro-scale dense structures. Literature is abundant with the benefits of dense subgraph discovery for various applications [12, 8].

k-core [18, 14], k-truss [16, 4, 24, 19, 5, 25, 10], and their generic variant for larger cliques, nucleus decomposition [17], (peeling algorithms in general) are deterministic algorithms which are effective and efficient solutions to find dense subgraphs and creating hierarchical relations among them. Hierarchy has been shown to be a central organizing principle of complex networks, which is useful to relate communities of a graph and can offer insight into many network phenomena [3]. Peeling algorithms do not aim to find a single optimum dense subgraph, but rather gives many dense subgraphs with varying sizes and densities, and ***hierarchy*** among them, if supported by a post-processing traversal step [14, 17].

2 Problem, Misconception and Challenges

We focus on undirected, unattributed graphs. Hierarchy of dense subgraphs is represented as the tree structure where each node is a subgraph, each edge shows a containment relation, and the root node is the entire graph. The aim is to efficiently find the hierarchy by using peeling algorithms.

Misconception in the literature: Recent studies on peeling algorithms has interestingly overlooked the connectivity condition of k-cores and k-trusses. In the original definition of k-core, Seidman states that k-core is the maximal and connected subgraph where any vertex has at least degree k [18]. However, almost all the recent papers on k-core algorithms [2, 6, 7, 1, 15, 13, 11, 23, 22, 13] did not mention that k-core is a connected subgraph although they cite Seidman's seminal work [18]. On the k-truss side, the idea is introduced independently by Saito *et al.* [16] (as k-dense), Cohen [4] (as k-truss), Zhang and Parthasarathy [24] (as triangle k-core), and Verma and Butenko [19] (as k-community). They all define k-truss as a subgraph where any edge is involved in at least k triangles. Regarding the connectivity, Cohen [4], and Verma and Butenko [19] defined the k-truss as a single component subgraph, while others [16, 24] ignored the connectivity.

Finding k-cores requires traversal on the graph after the peeling process, where maximum k-core values of vertices are found. It is same for k-truss and nucleus decompositions where the traversal is done on higher order structures. Constructing the hierarchy is only possible after that. However, it is not easy to track nested structure of subgraphs during a single traversal over entire graph. Traversing k-cores is cheap by a simple breadth-first search (BFS) in $O(|E|)$ time. When

[*]Sandia National Laboratories, Livermore, CA, USA. {asariyu,apinar}@sandia.gov

	k-core	k-truss			$(3,4)$ nucleus	
	NAIVE	HYPO	NAIVE	TCP*[10]	HYPO	NAIVE*
`Stanford3`	25.50x	1.10x	12.58x	3.41x	1.48x	1321.89x
`twitter-higgs`	27.89x	1.33x	16.24x	3.27x	1.78x	38.96x
`uk-2005`	58.02x	1.68x	90.50x	11.07x	1.24x	1.98x

Table 1: Speedups with our best algorithms for each decomposition. Starred columns (*) show lower bounds, when the other algorithm did not finish in 2 days. Best k-truss and (3,4) algorithms are significantly faster than alternatives, and also more efficient than the hypothetically best possible algorithm (HYPO) that does traversal to find the hierarchy.

it comes to k-truss and higher order peeling algorithms, however, traversal becomes much costly due to the larger clique connectivity constraints.

3 Contributions

- **Hierarchy construction by disjoint-set forest:** We propose to use disjoint-set forest data structure (DF) to track the disconnected substructures that appear in the same node of the hierarchy tree. Disjoint-set forest is incorporated into the hierarchy tree by selectively processing the subgraphs in a particular order. We show that our algorithm is generic, i.e., works for *any* peeling algorithm.
- **Avoiding traversal:** We introduce a new idea to build the hierarchy without traversal. In the peeling process, we construct the subgraphs while visiting neighborhoods and bookkeep the relations to previously constructed subgraphs. Applying a lightweight post-processing operation to those tracked relations gives us all the hierarchy, and it works for *any* peeling algorithm.
- **Experimental evaluation:** All the algorithms we proposed are implemented for k-core, k-truss and $(3,4)$-nucleus decompositions, in which peeling is done on triangles and the four-clique involvements. Furthermore, we bring out an idea from Matula and Beck's work [14], and adapt and implement it for our needs to solve the k-core hierarchy problem more efficiently. Table 1 gives a summary of the speedups we get for each decomposition. Our k-core hierarchy algorithm adaptation outperforms naive baseline by 58 times on `uk-2005` graph. The best k-truss and $(3,4)$ algorithms are significantly faster than alternatives. They also beat the hypothetically best possible algorithm (HYPO) that does traversal to find hierarchy. It is a striking result to show the benefit of our traversal avoiding idea.

Acknowledgments:
This work was funded by the Applied Mathematics Program at the U.S. Department of Energy and Sandia's Laboratory Directed Research & Development (LDRD) program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *Proc. of the 20th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pages 1316–1325, 2014.

[2] J. Cheng, Y. Ke, S. Chu, and M. T. Ozsu. Efficient core decomposition in massive networks. In *Proc. of the IEEE 27th International Conf. on Data Engineering*, ICDE, pages 51–62, 2011.

[3] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[4] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report, 2008.

[5] P. C. de Simon, M. Serrano, M. G. Beiro, J. I. Alvarez-Hamelin, and M. Boguna. Deciphering the global organization of clustering in real complex networks. *Scientific Reports*, 3(2517), 2013.

[6] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *International Conf. on Advances in Social Network Analysis and Mining (ASONAM)*, pages 87–93, 2011.

[7] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. D-cores: measuring collaboration of directed graphs based on degeneracy. *Knowl. Inf. Syst.*, 35(2):311–343, 2013.

[8] A. Gionis and C. E. Tsourakakis. Dense subgraph discovery: KDD'15 tutorial. In *Proc. of the 21th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pages 2313–2314, 2015. (`densesubgraphdiscovery.wordpress.com/tutorial`).

[9] D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proc. of the 18th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, KDD, pages 597–605, 2012.

[10] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In *Proc. of the ACM SIGMOD International Conf. on Management of Data*, pages 1311–1322, 2014.

[11] W. Khaouid, M. Barsky, S. Venkatesh, and A. Thomo. K-core decomposition of large networks on a single PC. *PVLDB*, 9(1):13–23, 2015.

[12] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, volume 40. 2010.

[13] R. Li, J. X. Yu, and R. Mao. Efficient core maintenance in large dynamic graphs. *IEEE Trans. Knowl. Data Eng.*, 26(10):2453–2465, 2014.

[14] D. Matula and L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of ACM*, 30(3):417–427, 1983.

[15] M. P. O'Brien and B. D. Sullivan. Locally estimating core numbers. In *IEEE International Conf. on Data Mining, ICDM*, pages 460–469, 2014.

[16] K. Saito and T. Yamada. Extracting communities from complex networks by the k-dense method. In *IEEE International Conf. on Data Mining Workshops, ICDMW*, pages 300–304, 2006.

[17] A. E. Sarıyüce, C. Seshadhri, A. Pınar, and Ü. V. Çatalyürek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proc. of the International Conf. on World Wide Web (WWW)*, pages 927–937, 2015.

[18] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

[19] A. Verma and S. Butenko. Network clustering via clique relaxations: A community based approach. In *Graph Partitioning and Clustering, DIMACS Workshop*, pages 129–140, 2012.

[20] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications.* Cambridge Univ Press, 1994.

[21] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[22] D. Wen, L. Qin, Y. Zhang, X. Lin, and J. X. Yu. I/o efficient core graph decomposition at web scale. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 133–144, 2016.

[23] H. Wu, J. Cheng, Y. Lu, Y. Ke, Y. Huang, D. Yan, and H. Wu. Core decomposition in large temporal graphs. In *Big Data, IEEE International Conf. on*, pages 649–658, 2015.

[24] Y. Zhang and S. Parthasarathy. Extracting analyzing and visualizing triangle k-core motifs within networks. In *Proc. of the IEEE International Conf. on Data Engineering*, ICDE, pages 1049–1060, 2012.

[25] F. Zhao and A. K. H. Tung. Large scale cohesive subgraphs discovery for social network visual analysis. In *Proc. VLDB Endow.*, pages 85–96, 2013.