2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

k-Truss Network Community Detection

Ian McCulloh
Accenture
Washington, DC, United States
Ian.McCulloh@accenturefederal.com

Onur Savas
Accenture
Washington, DC, United States
Onur.Savas@accenturefederal.com

***Abstract*— We review the k-truss algorithm for community detection in networks. The k-truss is an efficient clustering algorithm that holds advantageous properties for many network applications. In this paper, we compare the k-truss performance against other, more well-known community detection algorithms. The k-truss is uniformly more computationally efficient than the Louvain and Clauset-Newman-Moore algorithms in terms of speed and memory with comparable modularity. Potential applications are discussed.**

Keywords—Truss, Clustering, Community Detection, Modularity, Louvain, Newman.

I. Introduction

Computational methods in social network analysis (SNA) and network science (NS) can be broadly classified in to two categories: measurement of central actors and measurement of group or community structure. Both of these approaches face challenges with increasing network size, due in part to the emergence of large network data sets such as online social networks. In this paper we focus on efficient methods for measuring community structure in large networks.

Network community detection approaches can be further classified into three general approaches: minimize the number of edges removed; maximize the modularity; or exploit a local graph property. Minimizing the number of edges to remove is NP-hard and most often approximated with spectral clustering [1]. Spectral clustering uses the eigenvectors of the Laplacian matrix of the network. Unfortunately, the number of desired groups must be specified in advance. This is particularly problematic with online networks, where skewed degree distribution and missing edge data may result in many small network structures intermixed with larger components.

Modularity maximization is NP-complete [2]. Several heuristics are used to approximate modularity maximization. The most well-known is the Girvan-Newman grouping [3-5] which calculates the betweenness centrality [6] of each edge, removes the most central edge, and iterates this process, stopping when modularity reaches a maximum. This heuristic runs in $O(n^3)$ computational cost, where n is the number of nodes in the network. The complexity can be improved using a the faster Brandes betweenness centrality computation [7] or using an improvement introduced by Clauset, Newman, and Moore [6] that runs in $O(n \log n)$.

The most computationally efficient community detection approach in the literature is the Louvain method [8]. The Louvain method uses local properties to evaluate the ability of nodes and edges to propagate information through simulation. While the procedure is more complicated than the Clauset-Newman-Moore approach, their use of local properties significantly reduces the computational complexity and the composite method still outperforms the Clauset-Newman-Moore method in most situations.

Local properties of the network can also be used for community detection. A common example is the clique [9], which is a maximal sub network where all nodes are connected to all other nodes in the sub network. The clique can be a very strict criteria, excluding nodes that should be included in communities. Several relaxations of this criteria have been proposed such as the n-clique [10], k-plex [11], or n-clan [12]. In practice, however, these structures are often dispersed throughout the network and when they occur may overlap with each other. It is therefore problematic to use this approach for identifying distinct cohesive communities within a network.

In this paper, we focus on the k-truss for locating cohesive communities within a network [13]. While the k-truss has been introduced in the literature [13-15], there has been no formal comparison of the truss to other common community detection approaches. The background section will formally define the truss. The method section will describe a computational experiment to explore the computational efficiency of the truss against the Louvain and Clauset-Newman-Moore clustering approaches. Findings and conclusions will be presented.

II. Background

The k-truss is defined by triangles (3-clique) in the network. The triangle is the essential building block for social network analysis as well as many other network applications [16]. The k-truss requires all edges in the network to be contained within k or more triangles. In a social network, this finds practical application as a result of one of the six social forces, transitivity, where the likelihood of an edge between to vertices is increased with more connected vertices they share between them [17]. This differs from the clique, n-clique, k-plex, n-clan, or k-core in that these other methods rely on a simple dichotomous edge connection. Figure 1 illustrates this difference using a medium sized e-mail network collected at a small liberal arts college in North America. The network on the left highlights the 3-truss cluster, meaning each node is connected to an edge shared by at

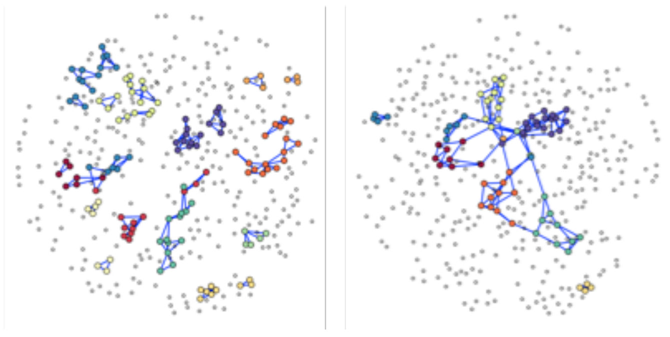

Figure 1. 3-Truss (left) and 3-Core (right)

least k-1 or 1 triangle. The network on the right highlights the 3-core, meaning all nodes have a degree of at least 3. The nodes are colored by their academic department. It can be seen that the k-core method can often identify overlapping clusters, which can make it unclear as to which group a given node belongs with. When dealing with large networks, it is often desired to achieve complexity reduction for visualization which is made problematic when using the clique or k-core approach. The k-truss tends to locate disparate communities.

The truss may include cliques, as seen in Figure 1. It is important to note that a k-truss is always a (k-1)-core, but a (k-1)-core is not always a k-truss. The clique or k-core approach may include vertices that are not as tightly bound to the community. By requiring each edge to include at least (k-2) triangles, the truss achieves greater complexity reduction, while still preserving the community.

The truss also has the property that any k-truss is completely embedded within a k-1 truss. For example, for an edge to share 5 triangles in common, they must also share 4 triangles in common. This allows the truss to locate hierarchical subgraphs that represent communities at different levels of resolution.

We formally define the truss as follows. Given an undirected, unweighted graph, G, which contains a set of vertices, V, and a set of edges, E, for any three vertices, u, v, w a triangle T_{uvw} exists if the three vertices are fully connected, such that $\{(u, v), (u, w), (v, w) \in E\}$.

Definition 1 (SUPPORT). The support of an edge, $e = (u, v)$, in graph, G, is defined as $|T_{uvw}|\ \forall\ w$ in V, and denoted as $sup(e)$.

The support of an edge is the number of triangles that contain that edge.

Definition 2 (k-TRUSS). The k-truss of G, denoted as C_k, is the largest subgraph of G such that $\forall\ e$ in C_k requires $sup(e) \geqslant (k-2)$.

The computational complexity of the truss algorithm occurs in the step to compute the support, which uses $O(|E|^{1.5})$ time, where $|E|$ is the cardinality of the edge set. This is made more efficient with an initial step to remove nodes with degree less than k-1, which eliminates pendant vertices (degree equal to 1) from the network.

Louvain is a *heuristic* algorithm that finds high modularity (defined below) partitions of large networks and that unfolds a complete hierarchical community structure for the network, thereby giving access to different resolutions of community detection [18]. The Louvain algorithm is divided in two phases that are repeated iteratively. The method consists of repeated application of two steps. The first step is a "greedy" assignment of nodes to communities, favoring local optimizations of modularity. The second step is the definition of a new coarse-grained network, based on the communities found in the first step. These two steps are repeated until no further modularity-increasing reassignments of communities are possible. One of the advantages of the Louvain algorithm is its ability to decompose the network into communities for different levels of organization.

Clauset-Newman-Moore is a hierarchical agglomeration algorithm for detecting community structure [5]. The operation of the algorithm involves finding the changes of a function of modularity (defined below) that would result from the amalgamation of each pair of communities, choosing the largest of them, and performing the corresponding amalgamation. One way to envisage (and implement) this process is to think of network as a multigraph, in which a whole community is represented by a vertex, bundles of edges connect one vertex to another, and edges internal to communities are represented by self-edges. Its running time on a network with $|V|$ vertices and $|E|$ edges is $O(|E|d\ \log|V|)$ where d is the depth of the dendrogram describing the community structure.

III. Method

We demonstrate the computational efficiency of the k-truss with an experiment using six publicly available data sets downloaded from the SuiteSparse Matrix Collection [19]. The networks were of varying size and complexity with number of vertices ranging from 3,783 to 317,080 and edges ranging from 14,124 to 1,049,866.

The performance of the k-truss is compared with the Louvain and Clauset-Newman-Moore clustering methods. We use the Wang and Cheng [15] algorithm for computing k-truss, De Meo et al [8] algorithm for Louvain, and the Clauset-Newman-Moore [5] algorithm. The algorithms were run using Python 3.7 in a Jupyter Notebook. While there are implementations of these algorithms in more efficient native languages, such as C++, the Python implementation provided an easy to use and understand implementation that was consistent across all three algorithms and allowed for a fair experiment for computational efficiency. The experiment is conducted on a computer using an i5 core processor with 8GB of RAM.

The k-truss algorithm that we use begins with $k = 2$ and continues until the maximum k is reached, beyond which there is no more k-truss. The efficiency of this algorithm can therefore be improved depending on the application. For example, if a top-down approach [15] were used to only locate the most core sub-graphs, this would avoid applying the algorithm to much of the network and achieve even faster performance at lower memory cost.

The computational efficiency is assessed using three criteria: clock time, memory consumption, and modularity. The clock time is simply the time in seconds that it takes to complete calculations on the respective data set. The memory consumption is calculated as the maximum increase in the Python heap space while running the community detection algorithm. The modularity is the resulting modularity, or quality of the clustering result, for the final clustered graph.

Modularity, in a nutshell, is one of the measures of the strength of division of a network into modules (also called groups, clusters or communities). We first define the community structure and then modularity below.

Definition 3 (Community Structure). Community structure (CS) is a division of the vertices in V into a collection of disjoint L subsets of vertices $\mathcal{C} = \{C_1, C_2, \dots, C_L\}$ such that $\bigcup_{i=1,2,\dots,L} C_i = V$.

Definition 4 (Modularity). Modularity of a community structure $\mathcal{C}$ is defined as

$$Q(\mathcal{C}) = \frac{1}{2M} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2M} \right) \delta_{ij},$$

where M is the total edge weights, d_i and d_j are the degree of nodes i and j respectively, $A=[A_{ij}]$ is the square adjacency matrix of dimension $|V|$ such that

$$A_{ij} = \begin{cases} 1, & if\ (i,j) \in E, \\ 0, & o.w. \end{cases}$$

and δ_{ij} is an indicator function such that

$$\delta_{ij} = \begin{cases} 1, & if\ i\ and\ j\ are\ in\ the\ same\ community, \\ 0, & o.w. \end{cases}$$

The aforementioned Louvain and Clauset-Newman-Moore algorithms aim to maximize this value because it is widely believed that higher modularity values indicate stronger community structure.

IV. Findings

The *k*-truss is uniformly more efficient than both the Clauset-Newman-Moore and the Louvain clustering algorithms. Figure 2 displays the clock time in seconds along the y-axis, while the number of edges is represented along the x-axis. A plot of vertices versus clock time provides a similar plot. It can be seen that the *k*-truss computes community detection more efficiently in terms of clock time than either the Louvain or Clauset-Newman-Moore.

The *k*-truss uses less memory when computing communities across all six data sets. Figure 3 displays the memory usage along the y-axis, while the number of edges is represented along the x-axis.

The modularity obtained with the *k*-truss algorithm is comparable to those of the Louvain and Clauset-Newman-Moore algorithms. The modularity of the *k*-truss algorithm is computed by first finding the *k,* for which the modularity is maximized. Then after removing the *k*-truss component from the original graph, the connected components of the rest of the graph are declared as other communities. Tables 1-3 display the numeric results from the computational experiment. The "NaN" indicates that Louvain failed to execute for these data. While the modularity is lower for the *k*-truss, this is a result of disregarding weakly connected vertices, such as the uncolored vertices in

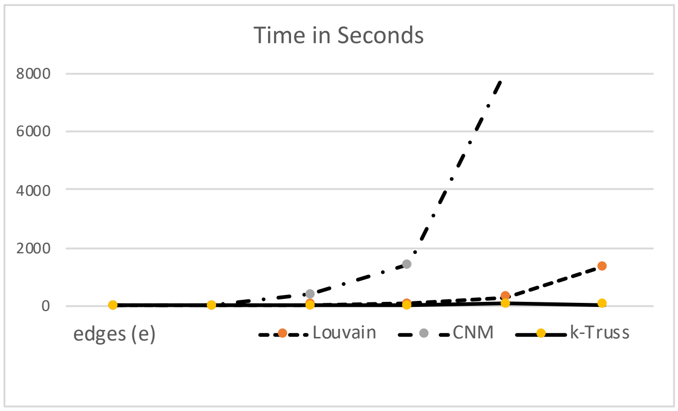

Figure 2. Computational efficiency as function of number of edges

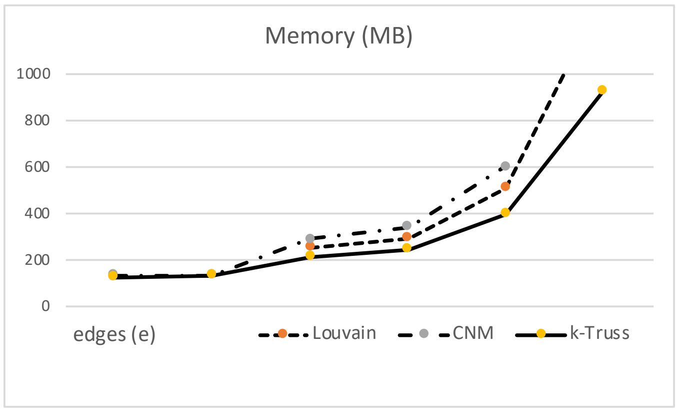

Figure 3. Memory usage as function of number of edges

Table 1. Louvain Results

LOUVAIN					
	nodes(N)	edges	time(sec)	memory	modularity
ca-Gr-Qc	5242	14496	4.75	128.55	0.8611
soc-bitcoin	3783	14124	NaN	NaN	NaN
wiki-RfA	11380	181073	30.09	254.81	0.4942
email-Enron	36692	183831	63.99	293.01	0.6168
soc-Epinions1	7588	405740	308.22	509.16	0.4381
com-DBLP	317080	1049866	1336.50	1337.53	0.8219

Table 2. Clauset-Newman-Moore Results

CLAUSET-NEWMAN-MOORE					
	nodes(N)	edges	time(sec)	memory	modularity
ca-Gr-Qc	5242	14496	7.46	132.84	0.8196
soc-bitcoin	3783	14124	21.44	134.06	0.4770
wiki-RfA	11380	181073	398.47	289.53	0.3972
email-Enron	36692	183831	1412.78	342.84	0.5202
soc-Epinions1	7588	405740	8072.50	602.17	0.3930
com-DBLP	317080	1049866	-	-	-

Table 3. *k*-Truss Results

K-TRUSS (runs from *k*=2 until *k* max)					
	nodes(N)	**edges**	**time(sec)**	**memory**	**modularity**
ca-Gr-Qc	5242	14496	0.97	123.50	0.8471
soc-bitcoin	3783	14124	1.78	131.51	0.4542
wiki-RfA	11380	181073	12.55	212.31	0.4679
email-Enron	36692	183831	14.39	246.87	0.4820
soc-Epinions1	7588	405740	58.09	395.81	0.3671
com-DBLP	317080	1049866	32.86	927.86	0.7754

Figure 1. While these vertices may affect the modularity, it is not clear that is a negative outcome. In fact, the hierarchical property, where a k-truss is completely embedded in a (k-1)-truss is possibly a more attractive clustering property.

V. Conclusion

As social network analysis is increasingly applied to larger and larger data, it becomes necessary to efficiently locate cohesive communities and understand how those communities interact with those adjacent to them. This reality has led to a wide range of methods for cohesive community detection. In this paper we conduct a computational experiment to evaluate the performance of leading community detection algorithms and identify a uniformly more efficient approach with the *k*-truss. The *k*-truss scales in polynomial time with the number of edges, and that can be improved with a top-down selection of clusters, focused on the most highly connected cores.

The *k*-truss has the added benefit of a hierarchical clustering property such that a *k*-truss is completely embedded in a (*k*-1)-truss. This property allows for the construction of hierarchical reduced graphs, where the graph can be easily expanded to reveal increasing levels of *k*. This may have advantages for driving network visualization layouts. The reduced graph from the *k*-maximal truss can be used to seed improved starting locations for graph layout algorithms to drive improved performance in network visualization. This provides an approach to create better displays of large online networks.

While the truss has existed in the literature for some time, it has not been widely used. This is surprising given the clear advantages in computational performance and the hierarchical property. It also provides interesting structural information about the network. The novel contribution of this paper is the *k*-truss comparison to other leading community detection approaches and demonstration of its computational efficiency.

References

[1] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, 2001.

[2] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On finding graph clusterings with maximum modularity," in *Graph-Theoretic Concepts in Computer Science*, 2007, pp. 121–132.

[3] M. Girvan and M. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, p. 7821, 2002.

[4] M. Newman and M. Girvan, "Finding and evaluating com- munity structure in networks," *Physical Review E*, vol. 69, no. 2, p. 26113, 2004.

[5] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 66111, 2004.

[6] L. Freeman, "A set of measures of centrality based on betweenness." Sociometry (1977): 35-41.

[7] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163– 177, 2001.

[8] P. De Meo, E. Ferrara, G. Fiumara, A. Provetti, "Generalized Louvain method for community detection in large networks," arXiv:1108.1502v2. 2012.

[9] R. Luce and A. Perry. "A method of matrix analysis of group structure," *Psychometrika*, 14(2):95–116, June 1949.

[10] R. D. Luce. "Connectivity and generalized cliques in sociometric group structure," *Psychometrika*, 15(2):169–190, 1950.

[11] S. B. Seidman and B. L. Foster. "A graph-theoretic generalization of the clique concept," *Journal of Mathematical Sociology*, 6:139–154, 1978.

[12] R. J. Mokken. "Cliques, clubs and clans," *Quality & Quantity*, 13(2):161–173, Apr. 1979.

[13] J. Cohen. "Trusses: Cohesive Subgraphs for Social Network Analysis," *NSA:Technical report*, 2008.

[14] E. Akbas and P. Zhao. "Truss-based community search: a truss-equivalence based indexing approach." *Proceedings of the VLDB Endowment* 10, no. 11 (2017): 1298-1309.

[15] J. Wang and J. Cheng. "Truss decomposition in massive networks." *Proceedings of the VLDB Endowment* 5.9 (2012): 812-823.

[16] S. Wasserman and K. Faust. "Social network analysis: Methods and applications," *Cambridge University Press*, 1994.

[17] I. McCulloh, H. Armstrong, and A. Johnson. *Social network analysis with applications*. John Wiley & Sons, 2013.

[18] V. D. Blondel, J.-L. Guillaume1, R. Lambiotte, and E. Lefebvre. "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics Theory and Experiment*, 2008.

[19] SuiteSparse Matrix Collection, https://sparse.tamu.edu/SNAP. Retrieved 15 April 2019.