Fast Truss Community Query in Large-scale Dynamic Graphs

Zheng Lu
University of Tennessee
Electrical Engineering and Computer Science
zlu12@vols.utk.edu

Qing Cao
University of Tennessee
Electrical Engineering and Computer Science
cao@utk.edu

ABSTRACT

Recently, there has been significant interest in the study of the community search problem in social and information networks: given one or more query nodes, find densely connected communities containing the query nodes. However, most existing algorithms require linear computational time to the size of the found community for each specific K value. Therefore, state-of-the-art algorithms have limited scalability in large scale graphs, where communities grow to millions of edges.

In this paper, given an undirected graph G and a set of query nodes Q, we study community query using the k-truss based community model. We formulate our problem of finding a connected truss community, as finding a connected k-truss subgraph with all possible k that contains Q. The state-of-art approximation algorithm can achieve this goal with a time complexity of $O(n'm')$ where n' and m' are the size of the result truss community. For queries that only identity and exact size of communities are required, We construct an index structure that can retrieve there information of all connected k-truss communities that contain Q with all possible K values. The algorithm can run in $\sum_{u \in Q} d(u)$, where $d(u)$ is the degree of vertex u. We prove that this is the optimal time complexity for truss community query. Extensive experiments on real-world networks show the effectiveness and efficiency of our algorithms.

ACM Reference format:
Zheng Lu and Qing Cao. 2016. Fast Truss Community Query in Large-scale Dynamic Graphs. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 3 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Community structures naturally exist in many real-world networks such as social, biological, collaboration, and communication networks. The task of community detection is to identify all communities in a network, which is a fundamental and well-studied problem in the literature. Recently, several papers have studied a related but different problem called community search, which is to find the community containing a given set of query nodes. The need for community search naturally arises in many real application scenarios, where one is motivated by the discovery of the communities in which given query nodes participate. Since the communities defined by different nodes in a network may be quite different, community search with query nodes opens up the prospects of user-centered and personalized search, with the potential of the answers being more meaningful to a user [3]. As just one example, in a social network, the community formed by a person's high school classmates can be significantly different from the community formed by her family members which in turn can be quite different from the one formed by her colleagues [6].

Various community models have been proposed based on different dense subgraph structures such as k-core [2, 5, 8], k-truss [3], quasi-clique [1], weighted densest subgraph [10], to name a few major examples. Of these, the k-truss as a definition of cohesive subgraph of a graph G, requires that each edge be contained in at least (k - 2) triangles within this subgraph. It is well known that most of real-world social networks are triangle-based, which always have high local clustering coefficient. Triangles are known as the fundamental building blocks of networks [9]. In a social network, a triangle indicates two friends have a common friend, which shows a strong and stable relationship among three friends. Intuitively, the more common friends two people have, the stronger their relationship. In a k-truss, each pair of friends is "‘endorsed"’ by at least (k - 2) common friends. Thus, a k-truss with a large value of k signifies strong inner-connections between members of the subgraph. Huang et al. [3] proposed a community model based on the notion of k-truss as follows. Given one query node q and a parameter k, a k-truss community containing q is a maximal k-truss containing q, in which each edge is "‘triangle connected"’ with other edges. Triangle connectivity is strictly stronger than connectivity. The k-truss community model works well to find all overlapping communities containing a query node q. We extended this model for the case of multiple query nodes.

AS The time complexity of the state-of-art approximation algorithm can calculate the connected truss community containing all query vertices with the largest k with a time complexity of $O(n'm')$ where n' and m' are the size of the result truss community. Although the linear time complexity for retrieve the whole community is optimal, these algorithms have limited scalability in large scale graphs, where communities grow to millions of edges. In many applications, such as query if a set of users are involved in same community and how cohesive is the community, only the infomation, such as the identity, the k and the size, of the community are required rather than the detailed community itself. For these quries, We construct an index structure that can retrieve information of all connected k-truss communities that contain Q with all possible K values. The algorithm can run in $\sum_{u \in Q} d_u$, where d_u is the degree of vertex u. We prove that this is the optimal time complexity for truss community query. Note that if further details of found truss communities are required, our index structure can also retrieve the exact community with linear time complexity to the community size which is also the optimal time complexity.

The rest of this paper is organized as follows. In Section 2 we show previous works on community search and detection as well as dense graph mining. Section 3 provides notations and definitions used in this paper. We explain induced mst graph for truss community search in Section 4. Section 5 discusses index construction algorithm. The evaluations of our algorithm are in Section 6. We conclude our work in Section 7.

2 RELATED WORKS

The related work to our study include community search and detection, and dense subgraph mining.

Community Search and Detection.

Dense Subgraph Mining.

3 PRELIMINARIES

In our problem, we consider an undirected, unweighted graph $G = (V, E)$. We define the set of neighbors of a vertex v in G as $N(v) = u \in V : (v, u) \in E$, and the degree of v as $d(v) = |N(v)|$. We define a triangle in G as a cycle of length 3. Let $u, v, w \in V$ be the three vertices on the cycle, and we denote this triangle by $\triangle uvw$.

Edge support. The support of an edge $e(u, v) \in E$ is defined as $s(e, G) = |\triangle uvw : w \in V|$. When the context is obvious, we replace $s(e, G)$ by $s(e)$.

K-truss. Given a graph G and $k \geq 2$, $H \subseteq G$ is a k-truss if $\forall e \in E(H), s(e, H)$ft($k$fk??2).

Maximal k-truss H is a maximal k-truss if it is not a subgraph of another k-truss in G.

Trussness. The trussness of a subgraph $H \in G$ is the minimum support of edges in H, denoted by $\tau(H) = mins(e, H) : e \in E(H)$. The trussness of an edge $e \in E(G)$ is defined as $\tau(e) = max_{H \in G} \tau(H) : e \in E(H)$.

We use the same triangle adjacency and triangle connectivity definition as in [3].

Triangle adjacency: $\triangle_1$, $\triangle_2$ are adjacent if they share a common edge, denoted by $\triangle_1 \cap \triangle_2 \neq \emptyset$.

Triangle connectivity: $\triangle_1$, $\triangle_2$ are triangle connected if they can reach each other through a series of adjacent triangles, i.e., for $1 \leq i < n, \triangle_i \cap \triangle_{i+1} \neq \emptyset$.

K-truss community H is a k-truss community if H is a maximal k-truss and $\forall e1, e2 \in E(H), \exists \triangle_1, \triangle_2$ in H such that $e1 \in \triangle_1, e2 \in \triangle_2$, then either $\triangle_1 = \triangle_2$, or $\triangle_1$ is triangle connected with $\triangle_2$ in H;

Problem Definition. The problem of studied in this paper is defined as follows. Given a graph $G(V, E)$, a set of query vertices $Q \in V$, find all truss communities containing Q with maximum k, a specific k or any possible k.

4 K-TRUSS COMMUNITY SEARCH ON INDUCED MST GRAPH

This section describes the indeced graph we used for community search.

5 INDEX CONSTRUCTION AND QUERY

We propose to solve the K-Truss community query problem using a hierarchy index structure. This section explains how to construct such index structure from the indeced mst graph and how to query based on this index.

Table 1: Datasets

Dataset	Type	$\lvert V_{wcc} \rvert$	$\lvert E_{wcc} \rvert$	$\overline{\sigma}$
Wiki	Communication	2.4M	4.7M	3.9
Skitter	Internet	1.7M	11.1M	5.07
Livejournal	Social	4.8M	43.4M	5.6
Hollywood	Collaboration	1.1M	56.3M	3.83
Orkut	Social	3M	117M	4.21
Sinaweibo	Social	58.7M	261.3M	4.15
Webuk	Web	39.3M	796.4M	7.45
Friendster	Social	65M	1.8B	5.03

Datasets with the number of vertices and edges in the largest weakly connected components, and the average shortest distance $\overline{\sigma}$ of 100,000 vertex pairs.

6 EVALUATIONS

In this section, we show the results of experimental evaluation of truss community query on our index structure.

6.1 Datasets

We evaluate our algorithm on 8 graphs from different disciplines as shown in table 1. All graphs are complex networks that have power-law degree distributions and relatively small diameters. To simplify our experiments, we treat them as undirected, un-weighted graphs and only use the largest weakly connected component of each graph. All datasets are collected from Snap [4].

6.2 Experiment settings

We evaluate our algorithms, we use a Cloudlab [7] c8220 server with two 10-core 2.2GHz E5-2660 processors and 256GB memory. We base our algorithm on Snap [4]. All algorithms are implemented in C++.

6.3 Query time

We first compare the query time of truss community search based on our index structure with the state-of-art related work.

6.4 Index construction time and size

We show in this section that our index achieves the similar construction time and size as previous work.

6.5 Index update time

The index update time upon graph changes, i.e. edge deletion, is shown in ...

7 CONCLUSION

In this paper, we describe ...

REFERENCES

[1] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 277–288.

[2] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 991–1002.

[3] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 1311–1322.

[4] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[5] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. 2015. Influential community search in large networks. *Proceedings of the VLDB Endowment* 8, 5 (2015), 509–520.

[6] Julian J McAuley and Jure Leskovec. 2012. Learning to Discover Social Circles in Ego Networks.. In *NIPS*, Vol. 2012. 548–56.

[7] Robert Ricci, Eric Eide, and The CloudLab Team. 2014. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *USENIX ;login:* 39, 6 (Dec. 2014). https://www.usenix.org/publications/login/dec14/ricci

[8] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 939–948.

[9] Jia Wang and James Cheng. 2012. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment* 5, 9 (2012), 812–823.

[10] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.