RETRIEVING TOP WEIGHTED TRIANGLES IN GRAPHS

Raunak Kumar, Paul Liu*, Moses Charikar, Austin R. Benson*

SIAM Workshop on Network Science 2020
July 9–10 · Toronto

[1]

Abstract

Many real-world networks carry a natural notion of strength of connection between nodes, which are often modeled by a weighted graph, but existing scalable graph algorithms for pattern mining are designed for unweighted graphs. Here, we develop a suite of deterministic and random sampling algorithms that enable the fast discovery of the 3-cliques (triangles) with the largest weight in a graph, where weight is measured by a generalized mean of a triangle's edges. For example, one of our proposed algorithms can find the top-1000 weighted triangles of a weighted graph with billions of edges in thirty seconds on a commodity server, which is orders of magnitude faster than existing "fast" enumeration schemes.

Introduction

Small subgraph patterns, also called graphlets or network motifs, have proven fundamental for the understanding of the structure of complex networks [3]. One of the simplest non-trivial subgraph patterns is the triangle (3-clique), and the basic problem of triangle counting and enumeration has been studied extensively from theoretical and practical perspectives [10]. The focus on triangles is in part spurred by the widespread use of the pattern in graph mining applications, including community detection [7], network comparison [5], representation learning [8], and generative modeling [6]. In addition, triangle-based network statistics such as the clustering coefficient are used extensively in the social sciences [11].

Nearly all of the algorithmic literature on scalable counting or enumeration of triangles focuses on *unweighted* graphs. However, many real-world network datasets have a natural notion of *weight* attached to the edges of the graph [1]. However, edge weights can enrich the types of small subgraph patterns that are used in analysis. For instance, the network clustering coefficient has been generalized to account for edge weights [4]; in these cases, a triangle is given a weight derived from the weights of its constituent edges. All this being said, we still lack the algorithmic tools for fast analysis of modern large-scale weighted networks, especially in the area of weighted triangle listing and counting.

In applications of weighted triangles in this big data regime, it can often suffice to retrieve only the k triangles of largest weight for some suitable k. For example, in large online social networks, the weight of an edge could reflect how likely it is for users to communicate with each other, and top weighted triangles and cliques in this network could be used for group chat recommendations. In such a scenario, we would typically only be interested in a small number of triangles whose nodes are very likely to communicate with each other as opposed to finding *all* triangles in the graph.

Another application for finding top-weighted triangles appears in prediction tasks involving higher-order network interactions. The goal of the "higher-order link prediction" problem is to predict which new groups of nodes will simultaneously interact (such as which group of authors will co-author a scientific paper in the future) [2]. In this setting, existing algorithms first create a weighted graph where an edge weight is the number of prior interactions that involves the two end points and then predict that the top-weighted triangles in this weighted graph will appear as higher-order interactions in the future. Again, it is not necessary to find all triangles since only the top predictions will be acted upon. Existing triangle enumeration algorithms do not scale to massive graphs for these problems, and we need efficient algorithms for retrieving triangles in large weighted graphs.

In this work, we address the problem of enumerating the top-weighted triangles in a weighted graph. To be precise, let $G = (V, E, w)$ be a simple, undirected graph with positive edge weights w. We define the weight of a triangle in G be equal to the generalized p-mean; specifically, if a triangle (i, j, k) has edge weights w_{ij}, w_{jk}, and w_{ik}, then

[1]*Both authors contributed equally to this research.

the triangle weight is

$$m_p(i, j, k) := \left[\frac{1}{3}(w_{ij}^p + w_{jk}^p + w_{ik}^p)\right]^{1/p}. \tag{1}$$

Given G and an integer parameter k, we develop algorithms to extract the *top-k heaviest triangles* in G. Note that some special cases of the p-mean include arithmetic mean ($p = 1$), geometric mean ($p = 0$), harmonic mean ($p = -1$), minimum ($p = -\infty$) and maximum ($p = \infty$). This family of means is more general and includes those previously examined by Opsahl and Panzarasa [4] and Benson et al. [2].

Methods and Results

At a high level, we develop two families of algorithms for extracting top-weighted triangles. The first family of algorithms is deterministic and optimized for extracting top-k weighted triangles for small k (typically up to a few tens of thousands). These algorithms take advantage of the inherent heavy-tailed edge weight distribution common in real-world networks. In the most general case, we show that under a modified configuration model, these algorithms are even “distribution-oblivious”, in the sense that they can automatically compute optimal hyper-parameters to the algorithm for a wide range of input graph distributions. Additionally, the algorithmic analysis is done in a continuous sense (rather than discrete), which may be of independent interest. The second family of algorithms is randomized and aims to extract a large number of heavy triangles (not necessarily the top-k). We show that this family of sampling algorithms is closely connected to the prior sampling algorithms for *counting* triangles on *unweighted* graphs [9] and is easily parallelizable.

We find that a carefully tuned parallel implementation of our deterministic algorithm performs well across a broad range of large weighted graphs, even outperforming the fast random sampling algorithms that are not guaranteed to enumerate all of the top-weighted triangles. A parallel implementation of our algorithm running on a commodity server with 64 cores can find the top 1000 weighted triangles in 30 seconds on a graph with nearly two billion weighted edges. We compare this with the off-the-shelf alternative approach, which would be an intelligent triangle enumeration algorithm that maintains a heap of the top-weighted triangles. Our proposed algorithms are orders of magnitude faster than this standard approach (see Table 1).

Table 1: Summary statistics of datasets and time (s) for computing top-1000 triangles.

dataset	# nodes	# edges	edge weight	
			mean	max
Ethereum	38M	103M	2.8	1.9M
AMiner	93M	324M	1.3	13K
reddit-reply	8.4M	435M	1.5	165K
MAG	173M	545M	1.7	38K
Spotify	3.6M	1.9B	8.6	2.8M

dataset	brute force	sampling	deterministic
Ethereum	52.91	9.03	**6.94**
Aminer	243.75	**3.72**	12.36
reddit-reply	4047.62	5.19	**4.74**
MAG	512.24	**4.92**	20.89
Spotify	>86400	60.33	**30.79**

References

[1] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the national academy of sciences*, 101(11):3747–3752, 2004.

[2] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

[3] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[4] T. Opsahl and P. Panzarasa. Clustering in weighted networks. *Social networks*, 31(2):155–163, 2009.

[5] N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[6] P. Robles, S. Moreno, and J. Neville. Sampling of attributed networks from hierarchical generative models. In *KDD*, pages 1155–1164. ACM, 2016.

[7] K. Rohe and T. Qin. The blessing of transitivity in sparse and stochastic networks. *arXiv*, 2013.

[8] R. A. Rossi and N. K. Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, apr 2015.

[9] C. Seshadhri, A. Pinar, and T. G. Kolda. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Statistical Analysis and Data Mining*, 7(4):294–307, 2014.

[10] L. D. Stefani, A. Epasto, M. Riondato, and E. Upfal. TRIÈST: Counting Local and Global Triangles in Fully Dynamic Streams with Fixed Memory Size. *ACM TKDD*, 11(4):1–50, jun 2017.

[11] B. F. Welles, A. Van Devender, and N. Contractor. Is a” friend” a friend? investigating the structure of friendship networks in virtual worlds. In *CHI 2010*, pages 4027–4032, 2010.