Triangle Core Decomposition and Maximum Cliques

Ryan A. Rossi
Purdue University

David F. Gleich
Purdue University

Assefaw H. Gebremedhin
Purdue University

Consider a graph $G = (V, E)$. A k-core of G is a maximal induced subgraph of G where each vertex has degree at least k. There is a linear time $O(|E| + |V|)$ time algorithm to compute the maximum k such that a vertex is in a k-core for all vertices in the graph [1]. Because of this efficient algorithm, and a simple, but often powerful, interpretation, k-cores are frequently utilized to study modern networks. Important k-core related quantities include the size of the 2-core compared with the graph, the distribution of k-core sizes, the largest k-core, and many similar quantities. In particular, the maximum value of k such that there is a $k-1$-core in G is known as the *coloring number* of a graph and provides an upper-bound on the largest clique in G. In this abstract, we will discuss a recently proposed generalization of k-cores to a triangle motif [2], refine a procedure to compute them to enable computations on large graphs, and use the relationship between largest triangle core to confirm that a heuristic maximum clique finder discovers the optimal solution for many social and information networks.

An equivalent definition of a k-core is a maximal induced subgraph of G where each vertex is incident on at least k edges. This definition then generalizes to any motif, and in particular, a triangle k-core is the maximal induced subgraph of G for which each edge $(u, v) \in E$ participates in at least k triangles [2]. The *triangle core number* $T(u, v)$ of an edge $(u, v) \in E$ is the highest order core that contains that edge. Similarly, the maximum triangle core of G is denoted $T(G)$. There is also a polynomial time algorithm for triangle cores, which is $O(|T|) = O(|E|^{3/2})$ in the worst case [2].

This algorithm to compute triangle k-cores proceeds in the same fashion as an edge k-core method. Let us review: to compute an edge k-core, we repeatedly remove all vertices of degree less than k. For triangle k-cores, then, we repeatedly remove all edges that participate in fewer than k-triangles. The insight that led to the $O(|V| + |E|)$ algorithm for edge k-cores, is that we can run this procedure for all k iteratively because the cores are nested [1]. The same fact is true of triangle cores [2]. In that triangle k-core algorithm, however, they explicitly store an array of triangles in order to check whether or not a triangle has been processed. This storage limits scalability as graphs may have billions of triangles with only millions of edges – see the Hollywood graph in Table 1. In the algorithm we present here, we

Algorithm 1 Fast Triangle Core Decomposition

```
procedure TriangleCores(G = (V, E))
    for each (u, v) ∈ E do
        Set T(u, v) to be the number of triangles involving (u, v)
    Bucket sort the edges by increasing triangle count
    while there are remaining edges do
        Let (u, v) be the edge with smallest T(u, v)
        Index all neighbors of u via remaining edges
        for each edge (v, w) that remains with w in the index do
            Subtract 1 from T(u, w) and T(v, w)
            and update the bucket sort
        Mark (u, v) removed.
```

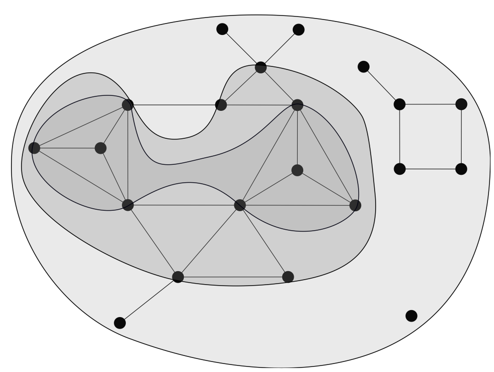

Figure 1: Triangle cores 0, 1, and 2.

Table 1: $d_{\max}$ and d_{avg} are the largest and average degrees, respectively. Also, $\bar{\kappa}$ is the mean clustering coefficient; $|T|$ is the total triangles, $T_{\max}$ and T_{avg} are the maximum number and average number of triangles incident on a vertex; K is an upper bound on ω from edge k-cores, T is an upper bound on ω from triangle k-cores, and $\tilde{\omega}$ is the size of a clique from our heuristic. The timings measure how long our heuristic, triangle counts, and triangle cores took, respectively.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	$d_{\max}$	d_{avg}	$\bar{\kappa}$	$T_{\max}$	T_{avg}	K	T	$\tilde{\omega}$	$\tilde{\omega}$ sec	Δ sec	T sec
DBLP-2010	226k	716k	4.8M	238	6	0.64	5.9k	21	75	75	75	0.02	0.10	0.29
CITESEER	227k	814k	8.1M	1.4k	7	0.68	5.4k	35	87	87	87	0.05	0.09	0.43
HOLLYWOOD	1.1M	56M	15B	11k	105	0.77	4.0M	13k	2209	2209	2209	1.57	38.93	727.2
YOUTUBE	496k	1.9M	7.3M	25k	7	0.11	151k	14	50	19	14	0.42	0.71	15.92
ORKUT	3.0M	106M	1.6B	27k	70	0.17	1.3M	525	231	75	45	13.8	38.73	770.5
LIVEJOUR	4.0M	28M	251M	2.7k	13	0.26	80k	62	214	214	214	3.2	3.92	54.3
FRIENDSTER	65M	1.8B	12.5B	5.2k	55	0.16	190	158k	305	129*	129	561	2616	45247
TEXAS84	36k	1.6M	34M	6.3k	87	0.19	141k	922	82	62	48	0.070	0.34	4.80
PENN94	42k	1.4M	22M	4.4k	65	0.21	68k	520	63	48	44	0.05	0.22	3.27
P2P-GNUT	63k	148k	6.1k	95	4	0.01	17	0.1	7	4*	4	0.01	0.01	0.02
AS-SKITTER	1.7M	11M	86M	35k	13	0.26	565k	50	112	68	64	0.37	4.33	70.6
UK-05	130k	12M	2.5B	850	181	0.99	124k	19k	500	500	500	0.05	2.52	12.40
WIKIPEDIA	1.9M	4.5M	6.7M	2.6k	4	0.16	12k	3	67	31*	31	0.66	0.54	3.54

can accomplish the same type of check implicitly by carefully ordering and indexing. Eliminating this storage reduces the runtime and memory requirements considerably: 443 seconds to 60 seconds and 7GB of memory to 500MB (LiveJournal graph from [2]).

In Algorithm 1, we present the refinement of the algorithm to remove the global triangle marks. At the conclusion, the array $T(u, v)$ contains the triangles core numbers for each edge. In the loop over all edges, we treat each edge only once, and remove it once we have visited all of its triangles. The inner-loop only visits a triangle (u, v, w) once because, after that loop, edge (u, v) is removed from the graph. This algorithm works by removing edges (u, v) with minimal triangle core number, and thus, once we remove all edges, we'll have the maximum triangle core numbers for each edge.

Now, we discuss the implications of triangle cores on clique finding. As noted in ref. [2], if a graph has a clique of size k, then it must have a triangle core of size $k-2$. Hence, if T is the maximum triangle core, then $T + 2$ is an upper bound on the size of the maximum clique ω in G. Thus, we have the following bounds: $\omega \leq T + 2 \leq K + 1 \leq d_{max}$ where K is the maximum k-core and d_{max} is the maximum degree in G. We have also found that the triangle core bound is sometimes tighter than an approximate chromatic number $\tilde{\chi}$ from a greedy coloring that uses k-core ordering. We expect that using the triangle cores in a greedy coloring scheme ought to improve a coloring bound further, but in practice, we find that the largest triangle core is often close to the maximum clique size. See the table.

Given that we have an upper bound on the largest clique size, we use a heuristic procedure to find a large clique in the graph. Our heuristic prunes vertices and their neighborhoods by edge k-cores and only searches vertices with an edge k-core bigger than the largest clique found thus far. At each step of the heuristic clique growing procedure, vertices are selected by greedily choosing the vertex with largest edge k-core number among the search candidates. We find this produces large cliques quickly.

In Table 1, we use the heuristic to compute $\tilde{\omega}$, and use the triangle cores to compute T, an upper bound on the maximum clique size. We can compute the triangle cores relatively quickly, it takes minutes and hours to handle graph with hundreds of millions or billions of edges, but the memory load is feasible on a modern desktop.

In the future, we plan to study ways to use the triangle cores to get better upper-bounds on the clique size by coloring the graph.

[1] V. Batagelj and M. Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[2] Y. Zhang and S. Parthasarathy. Extracting analyzing and visualizing triangle k-core motifs within networks. In *ICDE*, pages 1049–1060, 2012.