CS167: Reading in Algorithms
Triangle-Dense Graphs[*]

Tim Roughgarden[†]

April 9, 2014

1 Social Networks and Triangle Density

In these notes we discuss a paper of Gupta, Roughgarden, and Seshadhri, "Decompositions of Triangle-Dense Graphs" [1]. The motivation of the paper is to develop a theory of algorithms for social networks, like the graphs derived from Facebook or Twitter data mentioned in the first lecture. That is, the goal is to develop graph algorithms that work well on social networks, if not on worst-case graphs.

What's special about social networks? In the first lecture we mentioned three common properties: they are generally big, sparse, and have a skewed degree distribution. This paper discusses a different property: large *triangle density*. Intuitively, this is similar to having a large average clustering coefficient. Formally, the triangle density of a graph is the fraction of "filled in" 2-hop paths (aka "wedges"), which can also be written as

$$\frac{3 \cdot \text{number of triangles}}{\text{number of wedges}}.$$

Note the factor of "3" comes in because each triangle of a graph spawns three distinct wedges.

Every graph has a triangle density between 0 and 1. An acyclic graph has triangle density 0; so does a cycle of length at least 4. A triangle has a triangle density of 1, as does a clique, as does a disjoint union of cliques. A little thought shows the converse is also true: a graph has triangle density 1 only if it is the disjoint union of cliques.

The paper studies graphs with "large" triangle density, with the motivation that large social networks tend to have this property (among others). For example, let's compare the Facebook graph with a random graph. By a "random graph," we mean the following (called an "Erdős-Rényi graph"): for parameters n and $p \in [0,1]$, form a graph with n vertices by including each of the $\binom{n}{2}$ possible edges independently with probability p. We expect the

[†]Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: `tim@cs.stanford.edu`.

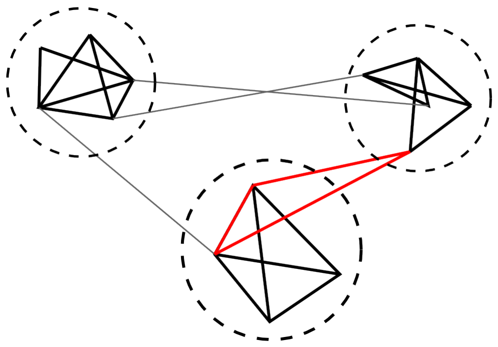


Figure 1: Main result. If a graph G has constant triangle density, then we can partition G into dense clusters such that not too many triangles are cut by the partition. The dotted circles denote such a partition, and the red triangle is cut by this partition.

triangle density of such a random graph to be roughly p — given that (u, v) and (v, w) are edges, the probability that (u, w) is also an edge is p. Thus, we expect the triangle density of a random graph to be at least p only if we also expect the number of edges to be at least $\approx pn^2/2$ — this means that only dense random graphs have high triangle density.

The Facebook graph, on the other hand, has close to 1 billion vertices and 100 billion edges (with an average degree close to 200). The corresponding edge density is little more than 10^{-6}. The triangle density of the Facebook graph, on the other hand, is roughly .16 [2] — five orders of magnitude larger than if it were a random graph with the same number of vertices and edge density.

2 Decomposing Triangle-Dense Graphs

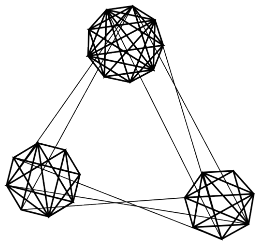
The paper [1] proposes triangle-dense graphs as a coarse model of social networks. The goal is then to design algorithms that work well on all triangle-dense graphs. There is a long history of designing graph algorithms for restricted classes of graphs — planar graphs, bounded-treewidth graphs, etc. Social networks are a relatively new application area, and triangle-dense graphs are a correspondingly new definition.

The primary contribution of the paper is to quantify the sense in which graphs with large triangle density are approximately a union of cliques. (Recall that a graph has triangle-density 1 if and only if it is a union of cliques.) More formally, the paper gives an algorithmic proof of the following result (Figure 1).

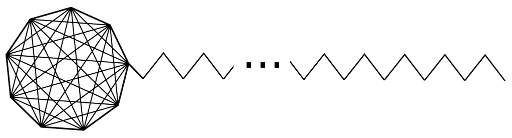
Theorem 2.1 *If the triangle density of $G = (V, E)$ is at least a constant, then there are disjoint subsets $V_1, \ldots, V_k$ of V such that:*

1. *Each induced subgraph $G[V_i]$ has radius at most 2 and is dense.*[1]

[1]A graph H has radius at most r if there is a vertex v of H such that every vertex w of H is at most r hops away from v. A graph H is dense if the number of edges is quadratic in the number of vertices — within a constant factor of the maximum possible.



(a) A motivating picture.



(b) A clique on $n^{1/3}$ vertices, attached to a path of length n.

Figure 2: Examples of triangle dense graphs.

2. The $G[V_i]$'s contain a constant fraction of the triangles of G.

A number of comments. First, we are suppressing the constant factors in both the hypothesis and the conclusions. The larger the triangle density of the graph G, the larger the edge density of the induced subgraphs $G[V_1], \dots, G[V_k]$ in the first property, and the larger the constant fraction of "saved" triangles in the second property. Second, while the V_i's are disjoint, they need not form a partition of V. Third, the most interesting case of Theorem 2.1 is when the graph G is sparse (but triangle-dense), as with most social networks. In this case, Theorem 2.1 states that the high triangle density must be the result of a number of disjoint dense "community-like" subgraphs. Fourth, the guarantee in second property is effectively identifying the "interesting content" of a social network with its triangles.

Let's get a better feel for Theorem 2.1 with some toy examples. The first picture that comes to mind is something like Figure 2a — a handful of near-cliques (missing a few edges each), sparsely connected to each other. But triangle-dense graphs can get pretty weird. Consider, for example, the "lollipop graph" of Figure 2b, with a clique on $n^{1/3}$ vertices and a path of n vertices. You should check that the number of edges and the number of triangles are both $\Theta(n)$, so this family of graphs has constant triangle density as $n \to \infty$.

Even though the lollipop graph does not resemble known social networks, Theorem 2.1 applies — so what do we expect the corresponding decomposition to look like? Recalling that the V_i's need not cover all of V, we can simply take V_1 equal to the clique. Observe that while all of the triangles of G are preserved, almost none of the $\approx n + n^{1/3}$ vertices are covered, and similarly almost none of the $\approx n + n^{2/3}/2$ edges are covered.

3 Interpretation of Theorem 2.1

There are two reasons to care about Theorem 2.1. The first is structural — it tells us what triangle-dense graphs must "look like." The second is algorithmic — the decomposition promised by Theorem 2.1 enables a divide-and-conquer approach to solving computational problems on triangle-dense networks.

As an example, let's consider the problem of counting the number of small cliques in a graph. Our first lecture considered the case of triangle-counting (i.e., 3-cliques); for concreteness, suppose we want to count the number of 4-cliques. All known algorithms for exactly counting r-cliques scale exponentially with r, so we'll consider algorithms that return approximate counts instead (say, up to 5% error).

Sampling is a standard approach for approximate counting. The most straightforward application of this to counting the 4-cliques of a graph G is the following. Pick 4 nodes at random from G, without replacement, and check if they form a clique in G. Note that if there are N_4 4-cliques of G, then the probability p that this random trial finds a 4-clique is exactly

$$\frac{24 \cdot N_4}{n(n-1)(n-2)(n-3)}, \tag{1}$$

since there are $\binom{n}{4}$ distinct 4-tuples of vertices (n is the number of vertices). Of course, a priori we don't know N_4 and hence don't know p. But we can estimate p via random trials: do as many independent random trials as time permits, and let $\hat{p}$ denote the fraction of these trials that are successful (i.e., find a 4-clique). From this estimate, we can use (1) to back out an estimate $\hat{N}_4$ of N_4.

How accurate is approximation counting via sampling? The answer depends on the true success probability p. If p is reasonably large, then the estimation approach above works well; if p is too small, then the estimate will suffer from large variance. More precisely, the approach will work well if p is at least $1000/M$ or so, where M is the number of trials.[2] So, when is the success probability p in our clique-counting experiment going to be high? Roughly speaking, we expect p to be tiny in sparse graphs, but not so tiny in dense graphs. That is, the accuracy of counting by sampling should increase with the edge density.

Returning to Theorem 2.1, here's how it enables a divide-and-conquer approach to approximate clique counting in triangle-dense graphs. The original triangle-dense graph G might well be sparse, so we can't directly estimate the number of 4-cliques by sampling. Instead:

1. Run the decomposition procedure of Theorem 2.1 to produce disjoint vertex subsets $V_1, \ldots, V_k$.

2. Separately for each (dense) induced subgraph $G[V_i]$, estimate the number of 4-cliques by sampling.

3. Return the sum of the estimates from all k $G[V_i]$'s.

Why might this work? First, Theorem 2.1 suggests that very few 4-cliques of G should be destroyed by the decomposition procedure — a typical 4-clique should wind up contained entirely in one of the V_i's. Second, Theorem 2.1 guarantees that each of the subgraphs $G[V_i]$ is dense — thus, as discussed earlier, a reasonable number of random trials should be enough to accurately estimate the number of 4-cliques in each of the subgraphs. Recent experiments have validated this approach [3].

[2] For example, suppose you know that a coin is either fair (50/50) or biased 60/40 toward heads. With 100 coin flips, you'll be able to distinguish between these two cases with high probability. If instead the probability of tails is either 10^{-5} or 10^{-10}, say, then the 100 coin flips will likely all be "heads" either way, giving you no information.

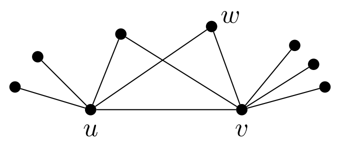


Figure 3: A graph with triangle density 2/9. Edges (u, v) and (v, w) have Jaccard similarity 2/7 and 1/5, respectively.

More generally, Theorem 2.1 gives a methodology for reducing a problem on triangle-dense graphs to the same problem on a collection of (edge-)dense graphs. This suggests that problems that are easy to solve on dense graphs should generally be easy also on (triangle-dense) social networks.

4 High-Level Description of Algorithm

The algorithm in [1] that proves Theorem 2.1 interleaves two different subroutines, the *cleaner* and the *extractor*. The key notion in the cleaner is the *Jaccard similarity* of an edge (u, v), which is a measure of the overlap in the neighborhoods of the two endpoints. Precisely, it is defined as

$$J(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)| - 2},$$

where $N(w)$ denotes the neighbors of a vertex w. The reason for the "-2" in the denominator is that we don't want to count u (which is in $N(v)$) or v (which is in $N(u)$). For example, the edge (u, v) in Figure 3 has Jaccard similarity $\frac{2}{7}$.

Why is Jaccard similarity an interesting statistic? Well, note that the triangles in which an edge (u, v) participates correspond exactly to the vertices w of $N(u) \cap N(v)$. Thus, edges with high Jaccard similarity belong to lots of triangles (relative to the sum of the endpoints' degrees), while edges with low Jaccard similarity play little role in the triangles of the graph.

The cleaner takes as input a graph and iteratively removes edges with Jaccard similarity less than a parameter ϵ, until no such edges remains.[3] In [1], ϵ is set to be roughly the same as the triangle density of the graph G. In the graphs in Figures 2a and 2b, this subroutine alone is enough to prune the irrelevant edges and produce a decomposition satisfying Theorem 2.1. In general, the intuition is that removing an arbitrary number of edges with low Jaccard similarity cannot destroy more than a small fraction of the graph's triangles.

The cleaner terminates with a graph in which every edge has Jaccard similarity at least ϵ. This is a good situation to be in, because all one-hop neighborhoods must then be edge-dense. To see this, consider an arbitrary vertex v and its neighborhood $N(v)$ (Figure 4). For every neighbor $w \in N(v)$, the edge (v, w) has Jaccard similarity at least ϵ. This implies that w is connected to at least a roughly ϵ fraction of the vertices in $N(v)$ (do you see

[3]Removing an edge changes the Jaccard similarity of the remaining edges. Suitable data structures can be used to efficiently track Jaccard similarities as edges are removed.

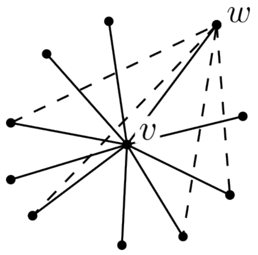


Figure 4: The neighborhood of $N(v)$, post-cleaning. Since edge (v, w) has Jaccard similarity at least ϵ, w must be connected to at least an ϵ fraction of $N(v)$.

why?). Iterating over all $w \in N(v)$, this argument shows that the set $\{v\} \cup N(v)$ induces an edge-dense subgraph. By construction, it has radius 1 (with v as the center).

The first idea for the extractor subroutine is to set V_1 equal to $\{v\} \cup N(v)$ for some vertex v, delete V_1 from G, and iterate. The discussion above shows that this algorithm would produce a decomposition satisfying the first property of Theorem 2.1. However, it could be a disaster for the second property. The problem is that there might be many more triangles that are partially contained (i.e., one or two vertices) in $\{v\} \cup N(v)$ than there are entirely contained in the set. (Exercise: check what happens when G is the complete tripartite graph $K_{n/3,n/3,n/3}$. What is the Jaccard similarity of every edge? What happens when you extract a 1-hop neighborhood?) The solution proposed in [1] is to extract $\{v\} \cup N(v)$ plus a greedily chosen subset of the 2-hop neighborhood of v, with a vertex of the two-hop neighborhood included if and only if its inclusion "saves" more triangles than it destroys. The right implementation of this idea ensures that the fraction of saved triangles is commensurate with the fraction of destroyed triangles, and establishes Theorem 2.1.

References

[1] Rishi Gupta, Tim Roughgarden, and C. Seshadhri. Decompositions of triangle-dense graphs. In *5th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 471–482, 2014.

[2] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. Technical Report 1111.4503, Arxiv, 2011.

[3] Joshua Wang, Rishi Gupta, Tim Roughgarden, and C. Seshadhri. Counting small cliques in social networks via triangle-preserving decompositions. Submitted, 2014.