New Streaming Algorithms for Counting Triangles in Graphs

Hossein Jowhari, Mohammad Ghodsi

Computer Engineering Department
Sharif University of Technology
Tehran, Iran
{jowhari@ce.,ghodsi@}sharif.edu

Abstract. We present three streaming algorithms that $(\epsilon, \delta)-$ approximate [1] the number of triangles in graphs. Similar to the previous algorithms [3], the space usage of presented algorithms are inversely proportional to the number of triangles while, for some families of graphs, the space usage is improved. We also prove a lower bound, based on the number of triangles, which indicates that our first algorithm behaves almost optimally on graphs with constant degrees.

1 Introduction

In this paper, we present streaming algorithms for counting triangles in massive graphs. In other words, let $G = (V, E)$ be an undirected graph with n vertices and m edges and let t be the number of triangles in G. we are interested in algorithms with sublinear space usage for $(\epsilon, \delta)-$approximating t while G is presented to the algorithm as a stream of edges. By sublinear space usage, we mean algorithms that use $o(m)$ bit space, and by stream of edges, we mean a sequence of edges that is an arbitrary permutation of E. In addition to the space usage, we restrict the algorithms to have only $O(1)$ passes over the stream and $o(m)$ per-edge processing time.

Bar-Yossef *et al* in [3] showed that every algorithm that decides the existence of a triangle, with probability at least 99/100, needs at least $\Omega(n^2)$ bit space. This fact results in a lower bound of $\Omega(n^2)$ for $(\epsilon, \delta)-$approximating t in general graphs, but Bar-Yossef *et al* showed that for graphs with considerably large number of triangles, it is possible to gain sublinear space. In other words, let T_i be the number of vertex triples that induce a subgraph with i edges in G. Based on this definition, $T_3 = t$. For $T_3 > 0$, they obtained a streaming algorithm with $O(1/\epsilon^3. \log 1/\delta.((T_1 + T_2 + T_3)/T_3)^3. \log n)$ space. Since $(T_1+T_2+T_3) = \Theta(mn)$, having an appropriate lower bound for T_3, one can use $o(m)$ space on graphs with $m^{2/3}n = o(T_3)$.

Our Contribution. We present three streaming algorithms for (ϵ, δ)-approximating T_3. Let d be the maximum degree and let C_i be the number of cycles of length i in

[1] Let $\epsilon, \delta > 0$ and let T_3 be the number of triangles. With probability at least $1 - \delta$, the algorithm outputs T_3' such that $(1 - \epsilon)T_3 \leq T_3' \leq (1 + \epsilon)T_3$.

the input graph. The first algorithm uses $O(1/\epsilon^2.\log(1/\delta).(md^2)/T_3.\log n)$ space and per-edge processing time and makes one pass over the stream (Theorem 1). The second algorithm uses $O(1/\epsilon^2.\log(1/\delta).(m^3 + mC_4 + C_6 + {T_3}^2)/{T_3}^2.\log n)$ space and per-edge processing time with one pass (Theorem 2) and the third algorithm uses $O(n + 1/\epsilon^2.\log(1/\delta).(T_2 + T_3)/T_3.\log n)$ while making three passes over the stream (Theorem 3). The first and second algorithm use the method of Alon *et al* in [2] and the third algorithm utilizes sampling to reduce the space usage. We also prove a lower bound of $\Omega(n/T_3)$ that indicates that our first algorithm behaves almost optimally when d is a constant (Theorem 4).

Related Work. After the seminal paper by Alon *et al* [2], Henzinger *et al* [7] formalized the streaming model and proved lower bounds for some graph problems. According to our knowledge, there are few attempts for solving graph problems in the streaming model. Recently, authors in [5, 6] have presented streaming algorithm for some graph problems such as maximum weighted matching and shortest path. In fact, most of the results for graph problems are impossibility results [4, 6, 5, 7].

2 Algorithms

First, we define some notations. For undirected graph $G = (V, E)$, let n, m and d be the number of vertices, edges and maximum degree respectively. Let Υ be the set of all vertex triples of G. We partition Υ to four parts Υ_i, $i = 0, \ldots, 3$, where Υ_i is the set of triples that induces a subgraph in G with i total edges and let $T_i = |\Upsilon_i|$. For $j = 4, \ldots, n$, let C_j be the number of cycles of length j in G .

2.1 One-pass algorithms

Here we present two algorithm that uses the method of Alon et al [2]. We define random variable X such that $E(X) = T_3$. By taking the average of an appropriate number of independent instances of X, we can reduce the variance so that by using Chebyshev's Inequality, we can obtain an approximation for T_3 with relative error at most ϵ. Hence, the space usage of the algorithm depend on $Var(X)$ and the space requirements for computing the random variable X.

First Estimator. The random variable X is computed as follows. Choose an edge (u, v), randomly and uniformly from the edges in the stream. Count the number of common neighbors of vertices u and v in the rest of the stream. Let c be the value of this counter at the end of the stream. We define $X = mc$.

Now, we compute the expectation and variance of X. Suppose we have an ordering on the triangles. For i-th triangle, we define indicator random variable Z_i as follows.

$$Z_i = \begin{cases} 1, \text{ if } i\text{-th triangle has been counted} \\ 0, \text{ otherwise} \end{cases}$$

By above definition, $X = m(\sum_{i=1}^{T_3} Z_i)$, and by the linearity of expectation, $E(X) = m(\sum_{i=1}^{T_3} E(Z_i))$. Since the probability of counting a specified triangle is $1/m$, we have $E(Z_i) = 1/m$, and consequently, $E(X) = m \times (T_3 \times 1/m) = T_3$. By definition of the variance,

$$Var(X) = m^2(Var(\sum_{i=1}^{T_3} Z_i) + \sum_{i \neq j} Cov(Z_i, Z_j)).$$

We bound $Cov(Z_i, Z_j)$ by $E(Z_i Z_j)$. $Z_i Z_j$ equals to 1 only when T_i and T_j have a common edge that has been picked by the algorithm. The probability of this event is $1/m$ and since there are at most $d-1$ triangles with a common edge, we have

$$Var(X) \leq m^2(\frac{1}{m}(T_3 + (d-2).T_3)) \leq m(d-1)T_3.$$

Now let Y be the average of $s = 8\frac{1}{\epsilon^2}\frac{md}{T_3}$ parallel instances of X. By Chebyshev's Inequality,

$$Pr(|Y - T_3| \geq \epsilon T_3) < \frac{Var(Y)}{\epsilon^2 E^2(Y)} = \frac{Var(X)/s}{\epsilon^2 {T_3}^2} < \frac{1}{8}.$$

Now for obtaining a (ϵ, δ)-approximation, we run $O(\log \frac{1}{\delta})$ independent estimators, each one succeeding to obtain an ϵ-relative approximation of T with probability at least $\frac{7}{8}$. We need $O(d. \log n)$ space for computing the random variable X and hence, the following result is obtained.

Theorem 1. *For $\epsilon, \delta > 0$, there is a streaming algorithm that outputs an (ϵ, δ)-approximation of T_3, with $T_3 > 0$, using $O(1/\epsilon^2. \log(1/\delta).(\frac{md^2}{T_3}). \log n)$ bit space and per-edge processing time.*

The space usage gets sublinear when $d^2 = o(T_3)$ and close to optimal when d is a constant (see Theorem 3). However the algorithm uses $O(d. \log n)$ bit space for computing the random variable that is poor for graphs with large degree. For our second estimator, we use a random variable that can be computed in $O(\log n)$ bit space.

Second Estimator. To compute the estimator, we need a family of uniform ± 1-valued random vectors of length n, which are 12-wise independent. As indicated in [2], this family can be constructed explicitly using the parity check matrices of BCH codes. These matrices can be constructed with only $O(\log n)$ bits (see [1] for the details). We pick a random vector v from this family (uniformly). Now, as the stream passes, we compute $Z = \sum_{(i,j) \in E} v(i)v(j)$. At the end of stream, we define $X = \frac{1}{6}Z^3$.

We now compute $E(X)$. Based on the definition,

$$E(X) = \frac{1}{6}E((\sum_{(i,j) \in E} v(i)v(j))^3).$$

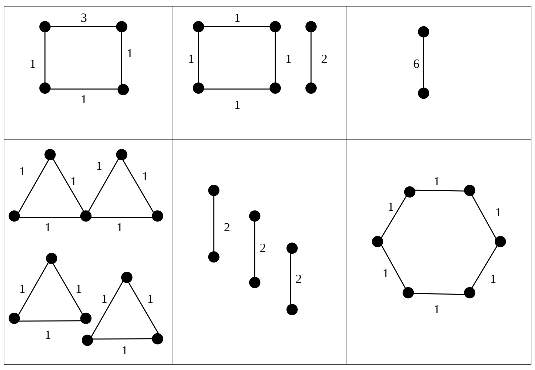

Fig. 1. Subgraphs that increase the variance of the second estimator

Consider that after expanding, each term in the summation corresponds to a specific subgraph of the input graph. By linearity of expectation and regarding the facts that $E(v^{2k+1}(i)) = 0$ and $v(i)$'s are 12-wise independent, the terms that have a variable with an odd power are evaluated to zero. Therefore, only the terms in form of $6v^2(i)v^2(j)v^2(k)$ are remained. These terms correspond to the triangles and thus,

$$E(X) = \frac{1}{6}(6 \times T_3) = T_3.$$

For variance, we have

$$Var(X) = E(X^2) - E^2(X) = \frac{1}{36}(E((\sum_{(i,j)\in E} v(i)v(j))^6) - T_3^2).$$

Similar to the computation of expectation, after expanding, we identify terms which are product of variables with an even power. The different subgraphs that correspond to the terms with an even power are depicted in Fig. 1. Note that the weight of edge (u_i, u_j) in each subgraph equals to the power of $(v(u_i)v(u_j))$ in the corresponding term. For each subgraph, the sum of the weight of edges, incident on each vertex, is even. Therefore according to the figure,

$$Var(X) = \frac{1}{36}(m + 120C_4 + 720{T_3}^2 + 360mC_4 + 720C_6 + 90m^3) - {T_3}^2$$
$$\leq 20({T_3}^2 + mC_4 + C_6 + m^3).$$

Similar to the previous algorithm, we run $s = 160\frac{1}{\epsilon^2}((mC_4 + C_6 + m^3)/{T_3}^2 + 1)$ parallel and independent instances of X and then we take the average of them. Let Y be the average. By Chebyshev's Inequality,

$$Pr(|Y - T_3| \geq \epsilon T_3) < \frac{Var(Y)}{\epsilon^2 E^2(Y)} = \frac{Var(X)/s}{\epsilon^2 {T_3}^2} < \frac{1}{8}.$$

As usual, by taking the median of $O(\log 1/\delta)$ independent and parallel instances of Y, an $(\epsilon, \delta)-$approximation is obtained.

Theorem 2. *For $\epsilon, \delta > 0$, there is a streaming algorithm that outputs an (ϵ, δ)-approximation of T_3, with $T_3 > 0$, using $O(1/\epsilon^2 . \log(1/\delta).(\frac{m^3+mC_4+C_6}{T_3^2})+1). \log n)$ bit space and per-edge processing time.*

2.2 Three-pass algorithm

A naive sampling algorithm, that picks samples of vertex triples, should pick at least $O(1/\epsilon^2 . \log(1/\delta).(\frac{T_0+T_1+T_2+T_3}{T_3}))$ random vertex triples. Here, we show how to decrease the number of required samples. The idea is to avoid sampling from triples in Υ_0 and Υ_1 by having the degrees of vertices.

Let d_i be degree of vertex u_i and let $D = \sum_{i=1}^{n} \binom{d_i}{2}$. For picking a random triple, first we pick a vertex randomly while a vertex with degree d_i is picked with probability $\binom{d_i}{2}/D$. Then, from neighbors of the picked vertex, we pick a pair of vertices randomly and uniformly. let a be the picked triple. Since $D = T_2 + 3T_3$, it is easy to see that

$$Pr(a \in \Upsilon_3) = \frac{3T_3}{T_2 + 3T_3}.$$

If we pick the triples based on the above procedure, $O(1/\epsilon^2 . \log(1/\delta).(\frac{T_2+3T_3}{T_3}))$ random triples suffices to $(\epsilon, \delta)-$approximate $|\Upsilon_3|$. Now we show how to implement the above sampling procedure with three passes over the stream. In the first pass, we compute d_i for each u_i. Then we pick the first vertex of the random triples. Let s_i be the number of occurrences of u_i in the sample set. In the second pass, for $i = 1, \ldots, n$, we pick s_i random pairs from neighbors of u_i. The random pairs and the starting vertex form the triples of the sample set. Finally, in the third pass, we determine the number of triangles in the sample set.

Theorem 3. *For $\epsilon, \delta > 0$, there is a streaming algorithm that uses three passes over the stream and produces an (ϵ, δ)-approximation of T_3, with $T_3 > 0$, using $O(n + \frac{1}{\epsilon^2} . \log \frac{1}{\delta} .(\frac{T_2}{T_3} + 1). \log n)$ bit space and per-edge processing time.*

3 Lower Bound

For lower bound, we use reduction to the *Bit-Vector Disjointness* problem in the communication complexity. In this problem, two parties A and B, each one has a binary vector with length of n. A and B want to devise an efficient communication protocol to decide whether their binary vectors are disjoint or not. Our lower bounds are based on a result obtained by Kalyanasundaram and Schnitger [8]. They showed that the length of any communication protocol, succeeding in distinguishing disjoint binary vectors with probability more than $\frac{1}{2}$, must be at least $\Omega(n)$ bits. Since the course of the communication is not restricted, the lower bound also holds for streaming algorithms with constant number of passes.

Theorem 4. *For $\epsilon < 1/3, \delta < 1/2$, every streaming algorithms that output an (ϵ, δ)-approximation of T_3, with $T_3 > 0$, requires at least $\Omega(n/T_3)$ bit space.*

Proof. We represent a pair of binary vectors by an undirected graph so that disjoint binary vectors can be distinguishable via approximating the number of triangles in the graph. Assume T_3 is even and divides n. Let G_1, G_2 and G_3 be n-vertex graphs, such that G_1 and G_2 has $T_3/2$ number of triangles and G_3 with no edges. In each graph, we partition the set of vertices into n/T_3 equal-size parts. For $j = 1, \cdots, n/T_3$, let P_{ij} be the partitions in G_i. Suppose we have an ordering on the set of vertices in each partition. For $k = 1, \cdots, T_3$, let v_{ijk} be the k-th vertex in partition P_{ij}. Consider the binary vectors B_1 and B_2, both with length of n/T_3 . For each j, we add the following edges.

1. For $k = 1, \cdots, T_3$, add an edge between v_{1jk} and v_{2jk}.
2. If $B_1(j) = 1$, for $k = 1, \cdots, T_3$, add an edge between v_{1jk} and v_{3jk}.
3. If $B_2(j) = 1$, For $k = 1, \cdots, T_3$, add an edge between v_{2jk} and v_{3jk}.

Let G be the resulted graph. It is easy to see that if B_1 and B_2 are disjoint, number of triangles in G would be T_3, otherwise number of triangles would be at least $2T_3$. By using an approximation algorithm with relative error less than $1/3$, we can distinguish between graphs with T_3 triangles and ones with at least $2T_3$ triangles. Consequently, we can distinguish disjoint bit-vectors by this algorithm. This completes the proof.

4 Summary

In this paper, we presented three streaming algorithms for counting triangles in graphs with $T_3 > 0$. In the following table, for each algorithm, we have shown when the space usage gets sublinear.

Algorithm	Sublinear space
Naive Sampling	$n^3 . \log n/m = o(T_3)$
[3]	$m^{2/3} n . \log n = o(T_3)$
1st Alg.	$d^2 . \log n = o(T_3)$
2nd Alg.	$\max\{m, \sqrt{C_4}, C_6/m\} . \log n = o(T_3)$
3rd Alg. (3 passes)	$\max\{n, T_2/T_3\} . \log n = o(T_3)$

However, due to the high dependency of the parameters in the space usage, a clear evaluation and comparison of the algorithms are still unknown to us. We presented a lower bound of $\Omega(n/T_3)$ that only helps in evaluating the first algorithm. A clear lower bound based on the terms T_3 and T_2 will be more useful to evaluate the algorithms. We guess a lower bound of $\Omega(T_2/T_3)$ could exist.

References

1. N. Alon, L. Babai, A. Atai. *A fast and simple randomized algorithm for maximum indepedent set problem.* J. Algorithms 7(1986), 567-583.

2. N. Alon, Y. Matias, M. Szegedy. *The space complexity of approximating the frequency moments.* STOC 96.
3. Z. Bar-Yossef, R. Kumar, S. Sivakumar. *Reduction in streaming algorithms with an application of counting triangles in graphs.* SODA 2002.
4. A.L Buchsbaum, R. Gianvarlo, and J.R Westbrook. *On finding common neighborhoods in massive graphs.* Thoretical Computer Science, 299 (1-3):707-718, 2003.
5. J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, *Graph distances in streaming model; the value of space.* Yale University Technical Report. 2004.
6. J. Feigenbaum, S.Kannan, A. McGregor, S. Suri, and J. Zhang, *On graph problems in a semi-streaming model.* To appear in the 31st International Colloquium on Automata, Languages and Programming, 2004.
7. M. R. Henzinger, P. Raghavan, and S. Rajagopalan, *Computing on data streams*, Technical Report 1998-001, DEC Systems Research Center .1998.
8. B. Kalyanasundaram and G. Schnitger, *The probabilistic communication complexity of set intersection.* SIAM Journal on Discrete Mathematics, 5 545-557, 1990.