Streaming Algorithms Measured in Terms of the Computed Quantity*

Shengyu Zhang

California Institute of Technology
Computer Science Department and Institute for Quantum Information,
1200 E California Bl, MC 107-81, Pasadena, CA 91125, USA
shengyu@caltech.edu

Abstract. The last decade witnessed the extensive studies of algorithms for data streams. In this model, the input is given as a sequence of items passing only once or a few times, and we are required to compute (often approximately) some statistical quantity using a small amount of space. While many lower bounds on the space complexity have been proved for various tasks, almost all of them were done by reducing the problems to the cases where the desired statistical quantity is at one extreme end. For example, the lower bound of triangle-approximating was showed by reducing the problem to distinguishing between graphs without triangle and graphs with only one triangle.

However, data in many practical applications are not in the extreme, and/or usually we are interested in computing the statistical quantity only if it is in some range (and otherwise reporting "too large" or "too small"). This paper takes this practical relaxation into account by putting the computed quantity itself into the measure of space complexity. It turns out that all three possible types of dependence of the space complexity on the computed quantity exist: as the quantity goes from one end to the other, the space complexity can goes from max to min, remains at max, or goes to somewhere between.

1 Introduction

Data stream is a very natural and important model for massive data sets in many applications, where the input data is given as a stream of items with only one or a few passes, and usually we want to determine or approximate some statistical quantity of the input stream. See [16] for an excellent and comprehensive survey.

Many algorithms are designed and many lower bounds on the space complexities are proven for various types of problems such as, just to name a few, frequency moments [1, 7, 14, 3, 6], vector distance [11, 17], and some graph problems [2, 4, 9, 13]. Almost all lower bounds were proved by reducing the problem to the cases where the desired statistical quantity is at an extreme end (and this

* This work was mostly done when the author was a graduate student in Computer Science Department at Princeton University, supported in part by NSF grants CCR-0310466 and CCF-0426582.

G. Lin (Ed.): COCOON 2007, LNCS 4598, pp. 338–348, 2007.

is further reduced to the communication complexity of some related problems). For example, the lower bound for approximating the number of triangles was proved by a reduction to distinguishing between the graph containing 0 and 1 triangle; the lower bound for the infinity frequency moment F_∞^* was proved by reducing the problem to distinguishing between $F_\infty^* = 1$ and $F_\infty^* = 2$.

Despite its theoretical correctness, this reduction to extreme cases can be misleading for many practical applications for at least the following two reasons. First, the extreme case may not happen at all in practice. For example, the number of triangles in a graph has many implications in various applications. In a social network, the number of triangles characterizes the average strength of the ties in the community [8, 18]. But note that in most (if not all) practical communities, there are a large number of triangles, and those extreme cases (0 and 1 triangle) are never the case. As another example, in many applications such as data mining, the number of common neighbors of two vertices in a graph shows the amount of common interest. A canonical example is that if two commodities have a large number of common buyers, then putting these two commodities close to each other in a supermarket will make more sales for both of them. Similar to the triangle example, the maximal number of common neighbors from data in practice is always large.

The second reason is from the user side. Even if some data happen to have the quantity at extreme, we are not interested in it in this case. For example, if two commodities have a very small number (such as one) of common buyers, then it barely means anything because that buyer may just happen to buy them. Therefore, we have a *threshold range* in mind within which we care about the quantity; if the quantity is outside the range, we will be satisfied if the algorithm can report "too low" or "too high".

Due to these two reasons, it is natural to ask the following question: is the hardness of a problem essentially due to the extreme cases? To answer this question, we study the space complexity in terms of both input size and the threshold range. In particular, for any input size n and any possible quantity value $q(n)$, the stream space complexity $s(n, q(n))$ is, roughly speaking, the minimal space used to compute $f(x)$ for all inputs in $\{x : f(x) = \Theta(q(n))\}$.

This question has been occasionally studied implicitly. In [2], Bar-Yosseff, Kumar, and Sivakumar initialized the study of graph problems in the adjacency stream model, where the graph is given by a sequence of edges (i, j) in an arbitrary order[1]. In particular they studied the problem of approximating the number of triangles, giving a one-pass algorithm using $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta}(1 + \frac{T_1+T_2}{T_3})^3 \log n)$ space, where T_i is the number of unordered triples containing i edges. Unfortunately, they could not show when it is better than the naive sampling algorithm (which uses $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta}(1 + \frac{T_0+T_1+T_2}{T_3}))$ space), and they asked this as an open problem. They also gave an $\Omega(n^2)$ lower bound for general graph, by reducing the problem at one extreme end (distinguishing between graphs with no triangle *vs.*

[1] They also proposed the incidence stream model, where each item in the stream is a vertex with all its neighbors. In this paper we only study the adjacency stream model.

with one triangle) to the communication complexity of some Boolean function. They then ask as another open problem for a lower bound in terms of T_1, T_2, T_3 [2]. Jowhari and Ghodsi [13] later proved a lower bound of $\Omega(n/T_3)$. In this paper we will show that their algorithm is always asymptotically worse than the naive sampling algorithm (for any graph) by proving $\left(1+\frac{T_1+T_2}{T_3}\right)^3 \geq \Omega\left(1+\frac{T_0+T_1+T_2}{T_3}\right)$ using algebraic graph theoretical arguments. Also, we prove a lower bound of $\min\{\Omega(n^3/T_3), \Omega(n^2)\}$, which matches the naive sampling algorithms, and the proof is much simpler than the previous (weaker) one [2]. It should be noted that subsequent papers [13, 5] improve the upper bound and finally [5] achieve $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\cdot(1+\frac{T_1+T_2}{T_3}))$. So our lower bound does not mean that the naive sampling algorithm is always the best, but that it is the best if the algorithm aims at dealing with all the graphs with T_3 in the known range.

For the problem of computing the maximal number of common neighbors, previously Buchsbaum, Giancarlo and Westbrook [4] gave a lower bound of $\Omega(n^{3/2}\sqrt{c})$ to compute the exact value, where c is the max number of the common neighbors. In this paper, after observing a matching upper bound, we consider the approximation version of the problem, showing that approximating the number needs $\tilde{\Theta}(n^{3/2}/\sqrt{c})$ space. Compared to the triangle counting example where the space complexity $\Theta(\min\{n^3/T_3, n^2\})$ drops from the maximum possible value ($\Theta(n^2)$) to the minimum possible value (constant), in this common neighbor example, the space complexity drops from some large value $\Theta(n^{3/2})$ to some small value $\Theta(n)$. Note that the $\tilde{\Theta}(n)$ space capacity is a well-studied model (called the semi-stream model) for graph problems, which is interesting [9] partly because $\tilde{\Theta}(n)$ is affordable in some Internet applications but higher space is not.

Not surprisingly, there are also many other problems whose space complexity, though first proved by considering extreme inputs, remains hard even if the computed quantity is not at extreme. We will give a simple example in this category too.

2 Preliminaries and Definitions

We say that an algorithm $\mathcal{A}$ (ϵ, δ)-approximates the function f on input x if $\mathbf{Pr}[|\mathcal{A}(x) - f(x)| \leq \epsilon f(x)] \geq 1-\delta$. In this paper, we will think of ϵ and δ as small constants. A graph $G = (V, E)$ is given in the adjacency streaming model if the input is a sequence of edges $(i, j) \in E$ in an arbitrary order.

2.1 Formulation of the Notion

The most naive way to formulate the notion in Section 1 is to define the space complexity $s(n, q)$ to be the minimum space needed to compute the quantity

[2] A lower bound in terms of all T_1, T_2 and T_3 does not seem quite justified: after all, for an unknown given graph, we do not know what T_i's are. But a lower bound in terms of mere T_3 is well-justified for the two reasons mentioned earlier.

$f(x)$ for all those x satisfying $f(x) = q$. However, this is obviously a useless definition because if we already know that $f(x) = q$ then we do not need any computation. Thus we need to be a little more careful about the definition.

Definition 1. *A function f has stream space complexity $\Theta(s(n, q(n)))$ if for any constants $c_2 > c_1 > 0$, the best algorithm (ϵ, δ)-approximating f on any input in $\{x : c_1 q(n) \leq f(x) \leq c_2 q(n)\}$ uses space $\Theta(s(n, q(n)))$.*

Several comments are in order. First, as mentioned in Section 1, we may desire that for those inputs that are not in the range, the algorithm outputs "too high" or "too low". Actually, in most (if not all) cases, the algorithm working for Definition 1 can be easily modified (with a multiplicative constant factor cost added) such that for any constants d_1 and d_2 with $c_1 < d_1 < d_2 < c_2$ and $d_2 - d_1 \geq 2\epsilon$, the algorithm has the following additional property: it outputs "too low" if $f(x) < d_1 q(n)$ and "too high" if $f(x) > d_2 q(n)$; for those inputs x with $c_1 q(n) \leq f(x) \leq d_1 f(x)$ or $d_2 q(n) \leq f(x) \leq c_2 f(x)$, either an ϵ-approximation or a "too low/high" is considered correct. Second, distinguishing between $f(x) \leq (1-\epsilon)cq$ and $f(x) \geq (1+\epsilon)cq$ with success probability $1-\delta$ is clearly a relaxation of the above task for any constant c (since we can let $d_1 = (1-\epsilon)c$ and $d_2 = (1+\epsilon)c$). The lower bounds showed in this paper apply to this easier task.

A basic fact that will be used in the proofs is as follows. The problem Index is a streaming problem where the input is an n-bit string x followed by an index $i \in [n]$, and the task is to output x_i with success probability at least $1 - \delta$.

Fact 1. *The Index problem needs $(1 - 2\delta)n$ bits of memory.*

A generalization of the fact is to consider k bits instead of just one bit. In the problem k-Index, the input is an n-bit string x followed by k indices $i_1, ..., i_k \in [n]$. The task is to distinguish between "$x_i = 1, \forall i = i_1, ..., i_k$" and "$x_i = 0, \forall i = i_1, ..., i_k$" with success probability at least $1 - \delta$.

Fact 2. *The k-Index problem needs $(1 - 2\delta)n/k$ bits of memory.*

This is easy to see by repeating each bit in Fact 1 k times. Also, a simple random sampling argument shows an $O(\frac{n}{k} \log \frac{1}{2\delta})$ upper bound for the number of memory cells.

3 Three Types of Dependence of the Space Complexity on the Computed Quantity

In this section, we will show three types of dependence of space complexity $s(n, q(n))$ on $q(n)$. In Section 3.1, we show a dependence which is the strongest possible: as $q(n)$ goes from one end (constant) to the other ($\Theta(n^3)$), the space complexity $s(n, q(n))$ drops from the maximal possible value ($\Theta(n^2)$) to the minimal possible value (constant). In Section 3.2, we show a weaker dependence: $s(n, q(n))$ drops from $\Theta(n^{3/2})$ to $\Theta(n)$. In Section 3.3, we show one example in which $s(n, q(n))$ is independent of $q(n)$.

3.1 Strong Dependence

The first problem that we study is triangle counting: Given a graph in the adjacency streaming model, (ϵ, δ)-approximate the number of triangles in the graph. Recall that T_i is the number of unordered triples of vertices with i edges, and thus T_3 is the number of triangles. The following theorem gives lower bounds that match the naive upper bounds: $O(n^3/T_3)$ for $T_3 \geq \frac{n}{3(1+\epsilon)}$ (by random sampling) and $O(n^2)$ space for $T_3 < \frac{n}{3(1+\epsilon)}$ (by storing all the edges).

Theorem 1. *Any streaming algorithm distinguishing between $T_3 \leq (1-\epsilon)t$ and $T_3 \geq (1+\epsilon)t$ with error probability $\delta = 1/3$ needs $\Omega(n^3/t)$ space for $t \geq \frac{n}{3(1+\epsilon)}$ and $\Omega(n^2)$ space for $t < \frac{n}{3(1+\epsilon)}$.*

Proof. Consider the case $t \geq \frac{n}{3(1+\epsilon)}$ first. Let the input graph G consist of 2 parts. One part is an $(n/3, n/3)$-bipartite graph $H = (L, R, E_H)$ (where L and R are left and right side vertex sets), and another part $J = (V_J, E_J)$ contains $n/3$ vertices. Partition L into $n/3k$ blocks $L_1, ..., L_{n/3k}$, each of size $k = \sqrt{3(1+\epsilon)t/n}$; similarly partition $R = R_1 \cup ... \cup R_{n/3k}$. (See Figure 1.) Denote by $H_{i,j}$ the subgraph $(L_i, R_j, E_H|_{L_i \times R_j})$. Now let the stream first give the graph H, with the promise that each subgraph $H_{i,j}$ is either empty or complete. Clearly it needs $(n/3k)^2$ bits of information to specify H. We claim that the streaming algorithm needs to basically keep all these $(n/3k)^2$ bits of information in order to approximate the number of triangles in the whole graph.

Actually, we claim that for any (i, j), by choosing the rest of the graph in an appropriate way, we can know whether $H_{i,j}$ is empty or complete with probability $1 - \delta$. Suppose we want to know whether $H_{i,j}$ is empty or complete, we let the remaining stream contain all edges in $\{(a, b) : a \in L_i \cup R_j, b \in V_J\}$. If $H_{i,j}$ is

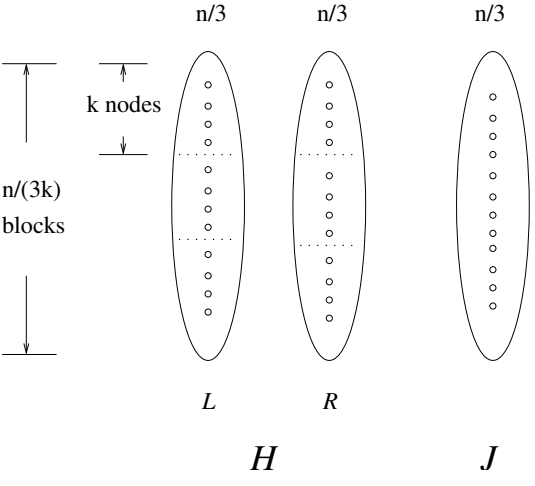

Fig. 1. A graph for illustration of the proof of triangle counting problem

complete, then G contains $k^2n/3 = (1+\epsilon)t$ triangles; if H_i is empty, then G contains no triangle. Since the algorithm can distinguish between $T_3 \leq (1-\epsilon)t$ and $T_3 \geq (1+\epsilon)t$ with probability $1-\delta$, it follows that after the first half of the stream (that specifies H) has passed, we can extract, for any (i,j), the one bit information about whether $H_{i,j}$ is empty or complete with probability $1-\delta$. Therefore by Fact 1, we need

$$(1-2\delta)\left(\frac{n}{3k}\right)^2 = \frac{(1-2\delta)n^3}{27(1+\epsilon)t} = \Omega\left(n^3/t\right) \tag{1}$$

bits of memory.

Note that in the above analysis, we implicitly require that block size $k \geq 1$ and the number of blocks $n/(3k) \geq 1$, for which we need $\frac{n}{3(1+\epsilon)} \leq t \leq \frac{(1/2-\delta)n^3}{27(1+\epsilon)}$. For $t > \frac{(1/2-\delta)n^3}{27(1+\epsilon)}$, the lower bound is trivially true. For $t < \frac{n}{3(1+\epsilon)}$, let $k=1$ and then the graph has $n/3$ triangles if $H_{i,j}$ is complete. Similar arguments give the lower bound of $(1/2-\delta)n^2/9 = \Omega(n^2)$, which completes our proof.

Another open question asked in [2] is about the comparison of their algorithm and the naive random sampling one. Their algorithm uses $O(\frac{1}{\epsilon^3}\log\frac{1}{\delta}(1+\frac{T_1+T_2}{T_3})^3\log n)$ space, and they asked when the algorithm is better than the naive sampling algorithm which uses $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}(1+\frac{T_0+T_1+T_2}{T_3}))$ space. We now show by some simple algebraic graph theory arguments that the algorithm in [2] is always no better than the naive random sampling one.

Proposition 1. *For any graph we have*

$$\left(1+\frac{T_1+T_2}{T_3}\right)^3 \geq \Omega\left(1+\frac{T_0+T_1+T_2}{T_3}\right), \tag{2}$$

Proof. First observe that $T_0+T_1+T_2+T_3 = \binom{n}{3}$ and that $T_1+2T_2+3T_3 = m(n-2)$ where m is the number of edges in the graph. The latter implies that $T_1+T_2+T_3 = \Theta(mn)$. Thus it is enough to prove that $T_3^2 = O(m^3)$, which can be easily done using algebraic arguments as follows. Suppose A is the adjacency matrix of the graph, then $Tr(A^3) = 6T_3$. Now notice that $Tr(A^3) = \sum_i\sum_k a_{ik}b_{ki}$ where $B = [b_{ij}]_{ij} = A^2$. It is easy to see that B is also symmetric, so $Tr(A^3) = \sum_{ik} a_{ik}b_{ik} \leq \sqrt{(\sum_{ik} a_{ik}^2)(\sum_{ik} b_{ik}^2)}$. Note that $\sum_{ik} a_{ik}^2 = \sum_{ik} a_{ik} = 2m$, and $\sqrt{\sum_{ik} b_{ik}^2} = \|B\|_2 = \|A\cdot A\|_2 \leq \|A\|_2^2 = 2m$, we thus have $T_3 \leq (2m)^{3/2}/6$, as desired.

As mentioned in Section 1, new algorithms are known ([13, 5]) which are better than the naive sampling algorithm for some graphs. So the above proposition is mainly of discrete math interest.

3.2 Weak Dependence

The second problem that we study is max common neighbor counting: Given a graph in the adjacency stream model, (ϵ,δ)-approximate the maximum number

of common neighbors, *i.e.* $mcn(G) = \max_{u,v} |\{w : (u,w) \in E, (v,w) \in E\}|$. In [4], it is showed that computing the *exact* value of $mcn(G)$ needs $\Omega(n^{3/2}\sqrt{c})$ space, where c is the max number of common neighbors. In this paper, after observing a matching upper bound for this exact counting problem, we consider the approximate version of the problem and show that the space complexity of approximating $mcn(G)$ is $\tilde{\Theta}(n^{3/2}/\sqrt{c})$.

In both the upper and lower bounds, we will use the following theorem in extremal graph theory. Denote by $ex(n, H)$ is the maximal number of edges that an n-vertex graph can have without containing H as a subgraph. The upper bound is by Kovari, Sos and Turan [15], and the lower bound is by Furedi [12].

Theorem 2. $\frac{1}{2}\sqrt{t}n^{3/2} - O(n^{4/3}) \leq ex(n, K_{2,t+1}) \leq \frac{1}{2}\sqrt{t}n^{3/2} + n/4$.

Now there is a very easy algorithm: keep all the edge information until the number of edges exceeds $\frac{1}{2}\sqrt{t}n^{3/2} + n/4$, in which case the graph contains $K_{2,t+1}$ for sure; otherwise, use the kept edge information to decide whether $mcn(G) \geq t$.

Now we give the algorithm to approximate $mcn(G)$ as in the Algorithm **Approx-mcn(G)** box. Its analysis is given by the theorem below.

Algorithm **Approx-mcn(G)**
Input: a data stream of edges $(i, j) \in E$ of a graph G in arbitrary order, two constants $a \in (0, 1)$ and $b > 1$.
Output: an (ϵ, δ)-approximate of $mcn(G)$ if $mcn(G) \in [ac, bc]$.

1. Use a counter to count the total number m of edges. Stop and output "$mcn(G) > bc$" if $m > M \equiv \frac{1}{2}n^{3/2}\sqrt{bc} + n/4$.
2. Randomly pick (with replacement) $t = \frac{1}{a\epsilon^2}(\log \frac{3n^2}{\delta})\frac{2n}{c}$ vertices $v_1, ..., v_t$. Denote this multi-set by T.
3. Keep all edges incident to T. (If the number exceeds $\frac{6Mt}{\delta n}$, output FAIL.)
4. Use the kept edge information to get

$$c' = \max_{u,v \in V-T} \sum_{i=1}^{t} \mathbf{1}[v_i \text{ is a common neighbor of } u \text{ and } v]$$

   where $\mathbf{1}[\phi]$ is the indicator variable for the event ϕ.
5. Output $c'n/t$ as estimate to $mcn(G)$. If $c' = 0$, output $mcn(G) < ac$.

Theorem 3. *For any c and any constants $a \in (0, 1)$ and $b > 1$, Algorithm **Approx-mcn(G)** (ϵ, δ)-approximates $mcn(G)$ for those G with $mcn(G) \in [ac, bc]$, and the algorithm uses space $O(n^{3/2} \log^2 n/\sqrt{c})$.*

Proof. First it is obvious that if the number of edge exceeds M which is larger than $ex(n, K_{2,bc})$, then $mcn(G) > bc$ for sure. Now consider $m = |E| < M$. By Markov's Inequality, the total degree of vertices in T is at most

$$\frac{3}{\delta}\frac{2mt}{n} \leq \frac{6Mt}{\delta n} = O\left(\frac{n^{3/2}(\log n + \log\frac{1}{\delta})}{\epsilon^2\delta\sqrt{c}}\right) \tag{3}$$

with probability $1-\delta/3$. Now assume $mcn(G) = s \in [ac, bc]$, then $\exists\ u_0, v_0$ sharing a set S_0 of s common neighbors. Fix u_0, v_0 and S_0. Since

$$\mathbf{Pr}[u_0 \in T \text{ or } v_0 \in T] \leq 2t/n \leq \delta/3 \tag{4}$$

if $c \geq \frac{12}{a\delta\epsilon^2}\log\frac{3n^2}{\delta}$. Let $X_i(u,v)$ be the indicator random variable for the event "the i-th vertex picked is a common neighbor of u and v", and let $X(u,v) = \sum_{i=1}^{t} X_i(u,v)$. Then under the condition that $u_0, v_0 \notin T$, we have $X(u_0,v_0) \leq c'$. Now by Chernoff's bound,

$$\mathbf{Pr}\left[X(u_0,v_0) < \frac{(1-\epsilon)ts}{n}\right] < e^{-\frac{\epsilon^2 ts}{2n}} \leq e^{-\frac{\epsilon^2 tac}{2n}} = \delta/(3n^2). \tag{5}$$

Therefore, $\mathbf{Pr}[c'n/t < (1-\epsilon)s] \leq \delta/(3n^2) < \delta/3$. On the other hand, by the definition of c', we have

$$\mathbf{Pr}[c'n/t > (1+\epsilon)s] = \mathbf{Pr}[c' > (1+\epsilon)st/n] \tag{6}$$

$$= \mathbf{Pr}[\exists u, v \in V - T, s.t.\ X(u,v) > (1+\epsilon)st/n] \tag{7}$$

$$\leq n^2 \cdot \mathbf{Pr}[X(u,v) > (1+\epsilon)st/n \mid u, v \notin T] \tag{8}$$

$$\leq n^2 \cdot \delta/(3n^2) = \delta/3. \tag{9}$$

Putting all things together, the algorithm outputs an ϵ-approximation with probability at least $1-\delta$.

The analysis of space that the algorithm uses is as follows. It needs $O(\log n)$ to store an vertex v or an edge (u,v), and the algorithm needs to store t vertices and $6MT/(\delta n)$ edges. Step 4 is space efficient since we can reuse space to check each pair (u,v). Thus the total number of bits used in the algorithm is $O(\frac{Mt}{\delta n}\log n) = O\left(\frac{n^{3/2}(\log n+\log\frac{1}{\delta})}{\epsilon^2\delta\sqrt{c}}\log n\right)$, which is $O\left(\frac{n^{3/2}\log^2 n}{\sqrt{c}}\right)$ if ϵ and δ are constants.

We can also prove a matching lower bound as follows.

Theorem 4. *Distinguishing between $mcn(G) \leq (1-\epsilon)c$ and $mcn(G) \geq (1+\epsilon)c$ with small constant error probability δ needs $\Omega(n^{3/2}/\sqrt{c})$ space for small constant ϵ (say $\epsilon = 0.1$).*

Proof. Consider the extremal graph H with $n-(1-\epsilon)c-1$ vertices and no $K_{2,(1-\epsilon)c+1}$ as a subgraph. For each vertex, partition its neighbors into subsets of size $2\epsilon c$, with possibly one subset of smaller size. The total number of edges in the regular (*i.e.* not smaller) subsets is at least

$$\sqrt{(1-\epsilon)c}(n-(1-\epsilon)c-1)^{3/2}/2-(n-(1-\epsilon)c-1)2\epsilon c-O(n^{4/3}) = \Omega(\sqrt{c}n^{3/2}) \tag{10}$$

if $\epsilon \leq 0.1$. Now consider the graph G with n vertices, $n-(1-\epsilon)c-1$ of which are for the graph H, and the rest $(1-\epsilon)c+1$ vertices are denoted by u and S with $|S| = (1-\epsilon)c$. (See Figure 2.)

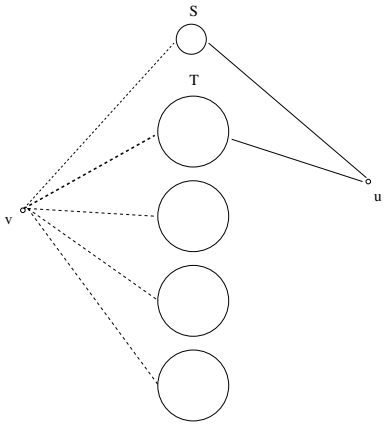

Fig. 2. A graph for illustration of the proof of max common neighbor counting problem

We will use Fact 2 to show the lower bound. Let the streaming first provide (a subgraph of) H, where for any $v \in H$ and each of its $2\epsilon c$-subsets T, the edges from v to T may all exist or all not. Fix the content of the memory of the algorithm. Then for any fixed $v \in H$ and any of its $2\epsilon c$-subsets T, we can know whether v and T are connected or not by providing the remaining stream (and running the streaming algorithm) in the following way. Connect v and S, u and $S \cup T$. Now note that if there is no edge between v and T, then an easy case-by-case study shows that there are no two vertices in the whole graph G sharing $(1-\epsilon)c+1$ common neighbors. If, on the other hand, v connects to all points in T, then clearly u and v share $(1+\epsilon)c$ common neighbors. Thus if we can distinguish the $mcn(G) \leq (1-\epsilon)c$ and $mcn(G) \geq (1+\epsilon)c$ with probability $1-\delta$, then we can distinguish between the two cases that the edges between v and T all exist and all of them do not exist with success probability $1-\delta$, which need $\Omega(n^{3/2}\sqrt{c})/2\epsilon c = \Omega(n^{3/2}/\sqrt{c})$ space by Fact 2.

3.3 Independence

Not surprisingly, there are also many problems whose hardness is not due to the extreme case in nature, though the previous lower bounds were proved by considering the extreme cases. We just mention one simple example here to end this section. The problem is to estimate the distance between two vertices on a graph: For two fixed vertices u, v on a graph G which is given in the adjacency stream model, (ϵ, δ)-approximate $d(u, v)$, the distance between u and v on the graph G. It is not hard to see that distinguishing between $d(u, v) \leq 3$ and $d(u, v) \geq n/2$ needs $\Omega(n)$ bits of memory. Actually, consider the graph consisting of two parts. One is a $n/2$-long path connecting u and v, and another part contains $n/4-1$ disjoint edges $(u_1, v_1), ..., (u_{n/4-1}, v_{n/4-1})$. We first stream in these two parts, but each edge (u_i, v_i) may or may not exist. Then to know whether a particular edge (u_i, v_i) exists or not, we connect (u, u_i) and (v, v_i). If

(u_i, v_i) exists, then the $d(u, v) = 3$; otherwise it is $n/2$. Thus we need $n/4-1$ bits of memory to distinguish these two cases. Note that in [10], a $2t + 1$ spanner is constructed using $O(tn^{1+1/t} \log^2 n)$ space thus the graph distance problem can be approximated up to a factor of $2t + 1$ by using the same amount of space, which implies that the $\Omega(n)$ lower bound is almost optimal for large constant approximation.

4 Discussions

Previous research on streaming algorithms mainly focused on designing space-efficient algorithms for important tasks; usually, log or even constant space complexity is desired. There may be more problems that, though may be very important in practical applications, did not get well studied theoretically simply because a high lower bound can be easily shown (by considering extreme inputs). This paper studies some problems which are hard for general case but easy if the computed quantity is within some range that we care about and/or the practical data are actually in.

Clearly, the same question can be asked for general algorithms. And within the domain of streaming algorithms, the problems studied in this paper happen to be those on graphs, but we believe that there are many more other problems having the same interesting phenomena.

Acknowledgement

The author thanks Sanjeev Arora, Moses Charikar, Yaoyun Shi and Martin Strauss for listening to the results and giving valuable comments.

References

[1] Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. Journal of Computer and System Sciences 58(1), 137–147 (1999)

[2] Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an application to counting triangles in graphs. In: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms (SODA), pp. 623–632. ACM Press, New York (2002)

[3] Bar-Yossef, Z., Jayram, T., Kumar, R., Sivakumar, D.: Information statistics approach to data stream and communication complexity. In: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, pp. 209–218. IEEE Computer Society Press, Los Alamitos (2002)

[4] Buchsbaum, A., Giancarlo, R., Westbrook, J.: On finding common neighborhoods in massive graphs. Theoretical Computer Science 299, 707–718 (2003)

[5] Buriol, L., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: Counting Triangles in Data Streams. In: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 253–262. ACM Press, New York (2006)

[6] Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multi-party communication coplexity of set disjointness. In: Proceedings of the 18th IEEE Conference on Computational Complexity, pp. 107–117. IEEE Computer Society Press, Los Alamitos (2003)
[7] Coppersmith, D., Kumar, R.: An improved data stream algorithm for frequency moments. In: Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 151–156. ACM Press, New York (2004)
[8] Derenyi, I., Palla, G., Vicsek, T.: Clique percolation in random networks. Physical Review Letters 94, 160–202 (2005)
[9] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 531–543. Springer, Heidelberg (2004)
[10] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph Distances in the Streaming Model: The Value of Space. In: Proceedings of the 16th Symposium on Discrete Algorithms (SODA), pp. 745–754 (2005)
[11] Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: An approximate L1 difference algorithm for massive data streams. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 501–511 (1999)
[12] Furedi, Z.: New asymptotics for bipartite Turan numbers. Journal of Combinatorial Theory, Series A 75, 141–144 (1996)
[13] Jowhari, H., Ghodsi, M.: New Streaming Algorithms for Counting Triangles in Graphs. In: Proceedings of the Eleventh International Computing and Combinatorics Conference, pp. 710–716 (2005)
[14] Indyk, P., Woodruff, D.: Optimal approximations of the frequency moments of data streams. In: Proceedings of the 37th ACM Symposium on Theory of Computing, pp. 202–208. ACM Press, New York (2005)
[15] Kovari, T., Sos, V.T., Turan, P.: On a problem of K. Zarankiewicz. Colloq. Math, vol. 3, pp. 50–57 (1954)
[16] Muthukrishnan, S.: Data Streams: Algorithms and Applications. Roundations and Trends in Theoretical Computer Science, vol. 1(2), pp. 117–236 (2005)
[17] Saks, M., Sun, X.: Space lower bounds for distance approximation in the data stream model. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, pp. 360–369. ACM Press, New York (2002)
[18] Shi, X., Adamic, L., Strauss, M.: Networks of strong ties. Physica A 378(1), 33–47 (2007)