New Bounds for the CLIQUE-GAP Problem using Graph Decomposition Theory

Vladimir Braverman[1] ⋆, Zaoxing Liu[1] ⋆⋆, Tejasvam Singh[1],
N. V. Vinodchandran[2] ⋆ ⋆ ⋆, and Lin F. Yang[1] †

[1] Johns Hopkins University, Baltimore MD 21218, USA
[2] University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Abstract. Halldórsson, Sun, Szegedy, and Wang (*ICALP 2012*) [16] investigated the space complexity of the following problem CLIQUE-GAP(r, s): given a graph stream G, distinguish whether $\omega(G) \geq r$ or $\omega(G) \leq s$, where $\omega(G)$ is the clique-number of G. In particular, they give matching upper and lower bounds for CLIQUE-GAP(r, s) for any r and $s = c\log(n)$, for some constant c. The space complexity of the CLIQUE-GAP problem for smaller values of s is left as an open question. In this paper, we answer this open question. Specifically, for any r and for $s = \tilde{O}(\log(n))$, we prove that the space complexity of CLIQUE-GAP problem is $\tilde{\Theta}(\frac{ms^2}{r^2})$. Our lower bound is based on a new connection between graph decomposition theory (Chung, Erdös, and Spencer [11], and Chung [10]) and the multi-party set disjointness problem in communication complexity.

1 Introduction

Graphs are ubiquitous structures for representing real-world data in several scenarios. In particular, when the data involves relationships between entities it is natural to represent it as a graph $G = (V, E)$ where V represents entities and E represents the relationships between entities. Examples of such entity-relationship pairs include webpages-hyperlinks, papers-citations, IP addresses-network flows, and people-friendships. Such graphs are usually very large in size, e.g. the people-friendships "Facebook graph" [24] has 1 billion nodes. Because of the massive size of such graphs, analyzing them using classical algorithmic approaches is challenging and often infeasible. A natural way to handle such massive graphs is to process them under the *data streaming* model. When dealing with graph data, algorithms in this model have to process the input graph as a stream of edges. Such an algorithm is expected to produce an approximation of the required output while using only a limited amount of memory for any ordering of the edges. This streaming model has become one of the most widely accepted models for designing algorithms over large data sets and has found deep connections with a number of areas in theoretical computer science including communication complexity [3, 9] and compressed sensing [14].

While most of the work in the data streaming model is for processing numerical data, processing large graphs is emerging as one of the key topics in this area. Graph problems considered so far in this model include counting problems such as triangle counting [4, 6, 18, 17, 23, 12], MAX-CUT [19] and small graph minors [8], and classical graph problems such as bipartite matching [15], shortest path [13], and graph sparsification [1]. We refer the reader to a recent survey by McGregor for more details on streaming algorithms for graph problems [22]. Recently, Halldórsson, Sun, Szegedy, Wang [16] considered the problem of approximating the size of maximum clique in a graph stream. In particular, they introduced the CLIQUE-GAP(r, s) problem:

Definition 1. CLIQUE-GAP(r, s): *given a graph stream G, integer r and s with $0 \leq s \leq r$, output "1" if G has a r-clique or "0" if G has no $(s+1)$-clique. The output can be either* 0 *or* 1 *if the size of the max-clique $w(G)$ is in $[s+1, r]$.*

⋆ This material is based upon work supported in part by the National Science Foundation under Grant No. 1447639, by the Google Faculty Award and by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the authors and do not represent the official view of DARPA or the Department of Defense.

⋆⋆ This work is supported in part by DARPA grant N660001-1-2-4014.

⋆ ⋆ ⋆ Research supported in part by National Science Foundation grant CCF-1422668.

† This work is supported in part by NSF grant No. 1447639.

In this paper we further investigate the space complexity of the CLIQUE-GAP problem and its relation to other well studied topics including multiparty communication, graph decomposition theory, and counting triangles. We establish several new results including a solution to an open question raised in [16].

1.1 Our Results

In this paper, we establish a new connection between graph decomposition theory [10, 11] and the multiparty set disjointness problem of the communication complexity theory. Using this connection, we prove new lower bounds for for CLIQUE-GAP(r, s) when $s = O(\log n)$ and complement the results of [16]. Our main technical results are Theorems 1, 2, 3, and 4. We summarize our results below and defer the proofs to the later sections.

The Upper Bound : We give a one-pass streaming algorithm that solves CLIQUE-GAP(r, s) using $\tilde{O}(ms^2/r^2)$ space. Note that our results do not contradict the lower bounds in [16], since their results apply for dense graphs with $m = \Theta(n^2)$.

Theorem 1. *For any r and s where $r \geq 100s$, there is a one-pass streaming algorithm (Algorithm 1) that, on any streaming graph G with m edges and n vertices, answers* CLIQUE-GAP(r, s) *correctly with probability ≥ 0.99, using $\tilde{O}(ms^2/r^2)$ space.* [1]

Lower Bounds : We give a matching lower bound of $\tilde{\Omega}(ms^2/r^2)$ on the space complexity of CLIQUE-GAP(r, s) when $s = O(\log n)$.

Theorem 2. *For any $0 < \delta < 1/2$ there exists a global constant $c > 0$ such that for any $0 < s < r$, $M > 0$, there exists graph families $\mathcal{G}_1$ and $\mathcal{G}_2$ that satisfy the following:*

- *for all graph $G_1 \in \mathcal{G}_1$, $|E(G_1)| = m \geq M$, G_1 has a r-clique;*
- *for all graph $G_2 \in \mathcal{G}_2$; $|E(G_2)| = m \geq M$, G_2 has no $(s + 1)$-clique;*
- *any randomized one-pass streaming algorithm $\mathcal{A}$ that distinguishes whether $G \in \mathcal{G}_1$ or $G \in \mathcal{G}_2$ with probability at least $1 - \delta$ uses at least $cm/(r^2 \log_s^2 r)$ memory bits.*

For $s = O(\log n)$ our lower bound matches, up to polylogarithmic factors, the upper bound of Theorem 1. Using the terminology from graph decomposition theory [10, 11] we extend our results to a lower bound theorem for the general promise problem GAP$(\mathcal{P}, \mathcal{Q})$, which distinguishes between any two graph properties $\mathcal{P}$ and $\mathcal{Q}$ satisfying the following restrictions. Note that $\alpha_*(G_0, \mathcal{Q})$ is a parameter denotes the minimum decomposition of G_0 by graphs in $\mathcal{Q}$, first defined in [10]. Please refer to Equation 6 for details.

Theorem 3. *Let $\mathcal{P}, \mathcal{Q}$ be two graph properties such that*
- $\mathcal{P} \cap \mathcal{Q} = \emptyset$;
- If $G'' \in \mathcal{P}$ and G'' is a subgraph of G', then $G' \in \mathcal{P}$;
- If $G', G'' \in \mathcal{Q}$ and $V(G') \cap V(G'') = \emptyset$, then $\tilde{G} = (V(G') \cup V(G''), E(G') \cup E(G'')) \in \mathcal{Q}$;
Let G_0 be an arbitrary graph in $\mathcal{P}$. Given any graph G with m edges and n vertices, if a one-pass streaming algorithm $\mathcal{A}$ solves GAP$(\mathcal{P}, \mathcal{Q})$ *correctly with probability at least $3/4$, then $\mathcal{A}$ requires $\Omega(\frac{n}{|V(G_0)|} \frac{1}{\alpha_*^2(G_0, \mathcal{Q})})$ space in the worst case.*

We use the tools we develop for the CLIQUE-GAP problem to give a new two-pass algorithm to distinguish between graphs with at least T triangles and triangle-free graphs. For $T = n^{2+\beta}$, the space complexity of our algorithm is $o(m/\sqrt{T})$ for $\beta > 2/3$. Cormode and Jowhari [12] give a two-pass algorithm using $O(m/\sqrt{T})$ space. Also, for $T \leq n^2$ they provide a matching lower bound. Our results demonstrate that for some $T > n^2$, it might be possible to refine the lower bound of Cormode and Jowhari. We state our results in Theorem 4.

Theorem 4. *Let $\mathcal{G}_1$ be a class of graphs of n vertices that has at least $T = n^{2+\beta}$ triangles for some $\beta \in [0, 1]$. Let $\mathcal{G}_2$ be a class of graphs of n vertices and triangle-free. Given graph $G = (V, E)$ with n nodes and m edges, there is a two-pass streaming algorithm that distinguishes whether $G \in \mathcal{G}_1$ or $G \in \mathcal{G}_2$ with constant probability using $\tilde{O}(\frac{mn^{2-\beta}}{T})$ space. In particular, for $\beta > 2/3$, the algorithm uses $o(m/\sqrt{T})$ space.*

[1] In this and following theorems, the constants we choose are only for demonstrative convenience.

Incidence Model : We also give a new lower bound for the space complexity of CLIQUE-GAP$(r, 2)$ in the *incidence model* of graph streams (Theorem 5).

Theorem 5. *If one-pass streaming algorithm solves* CLIQUE-GAP$(r, 2)$ *in the incidences model for any G with m edges and n vertices with probability at least $3/4$, it requires $\Omega(m/r^3)$ space in the worst case.*

1.2 Related Work

Prior work that is closest to our work is the above-mentioned paper of Halldórsson et al [16]. They show that for any $\epsilon > 0$, any randomized streaming algorithm for approximating the size of the maximum clique with approximation ratio $cn^{1-\epsilon}/\log n$ requires $n^{2\epsilon}$ space (for some constant c). To prove this result they show a lower bound of $\Omega(n^2/r^2)$ for CLIQUE-GAP(r, s) (using the two-party communication complexity of the set disjointness problem) when $r = n^{1-\epsilon}$ and $s = 100 \cdot 2^{1/\epsilon} \log n$.

The problem related to cliques that has received the most attention in the streaming setting is approximately counting the number of triangles in a graph. Counting the number of triangles is usually an essential part of obtaining important statistics such as the clustering coefficient and transitivity coefficient [5, 20] of a social network. Starting with the work of Bar-Yossef, Kumar and Sivakumar [4], triangle counting in the streaming model has received sustained attention by researchers [18, 6, 23, 12]. Researchers have also considered counting other substructures such as $K_{3,3}$ subgraphs [7] and cycles [5, 21].

The problem of clique identification in a graph has also been considered in other models. For example, Alon, Krivelevich, and Sudakov [2] considered the problem of finding a large *hidden clique* in a random graph.

2 Definitions and Results

2.1 Notations and Definitions

We give notations and definitions that are necessary to explain our results. For a graph $G = (V, E)$ with vertex set V and edge set E, we use m to denote the number of edges, n to denote the number of vertices, T to denote the number of triangles in G, Δ to denote the maximum degree of G, and $\omega(G)$ to denote the size of the maximum clique (also known as the clique number). We use $\tilde{O}$ and $\tilde{\Omega}$ to suppress logarithmic factors in the asymptotics.

We consider *the adjacency streaming model* for processing graphs [4, 6]. In this model the graph G is presented as a stream of edges $\langle e_1, e_2, ..., e_m \rangle$. We process edges under the cash register model: edge deletion is not allowed.

A *k-pass* streaming algorithm can access the stream k times and should work correctly irrespective of the order in which the edges arrive (the ordering is fixed for all passes).

2.2 Lower Bound Techniques

To establish our lower bounds on the CLIQUE-GAP(r, s) problem for arbitrarily small s, we use the well known approach of reducing a communication complexity problem to CLIQUE-GAP(r, s). For the reduction, we make use of graph decomposition theory [10, 11]. The communication complexity problem we use is the set disjointness problem in the one-way multi-party communication model.

The set disjointness problem in the one-way k-party communication model, denoted by DISJ$_k^n$, is the following promise problem. The input to the problem is a collection of k sets $S_1, \ldots, S_k$ over a universe $[n]$, with the promise that either all the sets are pairwise disjoint or there is a *unique* intersection (that is there is a unique $a \in [n]$ so that $a \in S_i$ for all $1 \leq i \leq n$). There are k players with unlimited computational power and with access to randomness. Player i has the input S_i. Player i can only send information to Player $(i + 1)$. After all the communication between players, the last player (Player k) outputs “0” if the k sets are pairwise disjoint or outputs “1” if the sets uniquely intersect. For instances that do not meet the promise the last player can output “0” or “1” arbitrarily. The communication complexity of such a protocol is the total number of bits communicated by all players. This problem was first introduced by [3] to prove lower bounds on the space complexity of approximating the frequency moments. In [9], it is shown that the communication complexity of DISJ_k^n is $\Omega(n/k)$.

We review basics of graph decomposition [10, 11]. An $\mathcal{H}$-decomposition of graph G is a family of subgraphs $\{G_1, G_2, \ldots, G_t\}$ such that each edge of G is exactly in one of the G_is and each G_i belongs to a specified class of graphs $\mathcal{H}$. Let f be a nonnegative cost function on graphs. The cost of a decomposition with respect to f is defined as $\alpha_f(G, \mathcal{H}) \equiv \min_D \sum_{i=1}^{t} f(G_i)$, where $D = \{G_1, G_2, \ldots, G_t\}$ is an $\mathcal{H}$-decomposition of G. Two functions that have received attention are $f_0(G) \equiv 1$ and $f_1(G) \equiv |V(G)|$. The former one counts the minimum number of subgraphs among all decompositions; and the later one counts the total number of nodes in the minimum decomposition. Many interesting problems in graph theory are related to this framework. For example $\alpha_{f_0}(G, \mathcal{P})$ is the thickness of G, for $\mathcal{P}$ the set of planar graphs; $\alpha_{f_1}(G, \mathcal{B})$, where $\mathcal{B}$ is the set of of complete bipartite graphs, arises in the study of network contacts realizing certain symmetric monotone Boolean functions. Refer to [10, 11] for more details on graph decomposition.

We are interested in the cost function f_0. $\alpha_{f_0}(G, \mathcal{H})$ is typically denoted as $\alpha_*(G, \mathcal{P})$ which is what we use in this paper. For the class $\mathcal{B}$, the class of complete bipartite graphs, it is known that $\alpha_*(K_n, \mathcal{B}) = \lceil \log_2 n \rceil$ [10].

To illustrate the reduction, consider CLIQUE-GAP$(r, 2)$. Let $k = \lceil \log_2 r \rceil$. Let $\{H_1, H_2, \ldots, H_k\}$ be a decomposition of G so that H_i's are bipartite and $\cup H_i$ is K_r. We will reduce an instance $S_1, \ldots, S_k$ of $\mathrm{DISJ}_k^{n/r}$ to a graph G on n vertices as follows. The graph G has n/r groups of r vertices each. The players collectively and independently build the graph G as follows. Consider Player i and her input $S_i \subseteq [n/r]$. For an $a \in S_i$, Player i puts the graph H_i on r vertices of group a into the stream. It is clear that if S_is are disjoint then the graph G is a collection of disjoint bipartite graphs and if there is a unique intersection a, the group a forms $\cup H_i = K_r$. Using standard arguments, we can show that the space complexity of CLIQUE-GAP$(r, 2)$ is $\Omega(n/r \log_2^2 r)$. Details are given in Section 4.

This proof can be generalized. In particular, we prove Theorem 2 by choosing $\mathcal{H}$ as set of s-partite graphs and prove Theorem 5 by choosing $\mathcal{H}$ as set of k-star graphs.

3 An Upper Bound

In this section we give an algorithm for CLIQUE-GAP(r, s) that uses $\tilde{O}(ms^2/r^2)$ space. Note that for $s = \Omega(r)$, the trivial algorithm that stores the entire graph has the required space complexity. Hence we will assume $s = o(r)$.

Algorithm 1 Algorithm for CLIQUE-GAP(r, s)

```
Input:
    Graph edge stream $\langle e_1, e_2, \ldots, e_m \rangle$ of graph $G = (V, E)$, positive integers $r, s$.
Output:
    "1" if a clique of order $r$ is detected in $G$; "0" if $G$ is $(s+1)$-clique free.
Initialize:
    Set $p = 40(s+1)/r$.
    Set memory buffer $M$ empty.
    Compute $n$ pairwise independent bits $\{Q_v | \text{for all } v \in V\}$ using $O(\log n)$ space such that
    for each $v \in V$, $Pr[Q_v = 1] = p$.
while not the end of the stream do
    Read an edge $e = (a, b)$.
    Insert $e$ into $M$ if $Q_a = 1$ and $Q_b = 1$.
    If there is an $(s+1)$-clique in $M$, then output "1".
output "0".
```

Theorem 1. *For any r and s where $r \geq 100s$, there is a one-pass streaming algorithm (Algorithm 1) that, on any streaming graph G with m edges and n vertices, answers* CLIQUE-GAP(r, s) *correctly with probability ≥ 0.99, using $\tilde{O}(ms^2/r^2)$ space.* [1]

Proof. If $s < 2$, it is trivial to detect an edge. So let us assume $s \geq 2$. If the input graph G has no $(s+1)$-clique, the algorithm always outputs "0" since the algorithm outputs "1" only if there is an $(s+1)$-clique on a sampled subgraph of G. Consider the case where G has a r-clique. Let $K_r = (V_K, E_K)$ be such a clique. Let the random variable Z denote the number of nodes 'sampled' from V_K. That is, $Z = \sum_{v \in V_K} Q_v$. The probability that $Q_v = 1$ is p and $Var(Q_v) = p(1-p)$. Hence $E(Z) = rp$ and since each Q_v is pairwise independent, $Var(Z) = rp(1-p)$. Thus for $s \geq 2$, by Chebyshev's bound, we have

$$\begin{aligned} Pr(Z \leq s) =& Pr(Z - E(Z) < s + 1 - E(Z)) \\ \leq& Pr(|Z - E(Z)| \geq |s + 1 - E(Z)|) \\ \leq& \frac{Var(Z)}{(s+1-E(Z))^2} \\ =& \frac{rp(1-p)}{(s+1-rp)^2} \leq \frac{40(s+1)}{39^2(s+1)^2} \leq 1/100. \end{aligned} \tag{1}$$

The probability of sampling an edge (u, v) is p^2, given by the probability of sampling both u and v. Thus the expected memory used by the above algorithm is $\tilde{O}(ms^2/r^2)$.

4 Lower Bounds

In this section we present our lower bounds on the space complexity of the CLIQUE-GAP problem. Our main theorem is the following.

Theorem 2. *For any $0 < \delta < 1/2$ there exists a global constant $c > 0$ such that for any $0 < s < r$, $M > 0$, there exists graph families $\mathcal{G}_1$ and $\mathcal{G}_2$ that satisfy the following:*

- *for all graph $G_1 \in \mathcal{G}_1$, $|E(G_1)| = m \geq M$, G_1 has a r-clique;*
- *for all graph $G_2 \in \mathcal{G}_2$; $|E(G_2)| = m \geq M$, G_2 has no $(s+1)$-clique;*
- *any randomized one-pass streaming algorithm $\mathcal{A}$ that distinguishes whether $G \in \mathcal{G}_1$ or $G \in \mathcal{G}_2$ with probability at least $1 - \delta$ uses at least $cm/(r^2 \log_s^2 r)$ memory bits.*

For $s = O(\log n)$, this matches our $\tilde{O}(ms^2/r^2)$ upper bound up to poly-logarithmic factors and solves the open question of obtaining lower bounds for CLIQUE-GAP(r, s) for small values of s (from [16]). Our main technical contribution is a reduction from the multi-party set disjointness problem (DISJ$_k^n$) in communication complexity to the CLIQUE-GAP problem. The reduction employs efficient graph decompositions.

We use the following optimal bound on the communication complexity of DISJ$_k^n$ proved in [9].

Theorem 6 ([9]). *Any randomized one-way communication protocol that solves* DISJ$_k^n$ *correctly with probability $> 3/4$ requires $\Omega(n/k)$ bits of communication.*

Before we prove Theorem 2 in detail, we will give the construction for CLIQUE-GAP$(4, 2)$. The reduction is from DISJ$_2^{n/4}$ to CLIQUE-GAP$(4, 2)$ (for the general case it will be from DISJ$_{\lceil \log_s r \rceil}^{n/r}$ to CLIQUE-GAP(r, s)). For any instance of DISJ$_2^{n/4}$, where Player 1 holds a set $S_1 \subset [n/4]$ and Player 2 holds a set $S_2 \subset [n/4]$, we construct an instance G with n vertices of CLIQUE-GAP(4,2) as follows. The n vertices are denoted by $\{v_{i,j} | i = 1, 2, 3, \ldots, n/4, j = 0, 1, 2, 3\}$. This notation partitions the vertex set to be $n/4$ groups, each of size 4, denoting as $V_i \equiv \{v_{i,0}, v_{i,1}, v_{i,2}, v_{i,3}\}$ for $i = 1, 2, 3, \ldots, n/4$. We partition $V_i = V_{i,0} \cup V_{i,1}$, where $V_{i,0} = \{v_{i,0}, v_{i,1}\}$ and $V_{i,1} = \{v_{i,2}, v_{i,3}\}$. Further partition $V_{i,0} = V_{i,0,0} \cup V_{i,0,1}$ and $V_{i,1} = V_{i,1,0} \cup V_{i,1,1}$, where $V_{i,0,0} = \{v_{i,0}\}$, $V_{i,0,1} = \{v_{i,1}\}$, $V_{i,1,0} = \{v_{i,2}\}$ and $V_{i,1,1} = \{v_{i,3}\}$.

[1] In this and following theorems, the constants we choose are only for demonstrative convenience.

Player 1 places all edges of the complete bipartite graphs between $V_{i,0}$ and $V_{i,1}$ if $i \in S_1$.

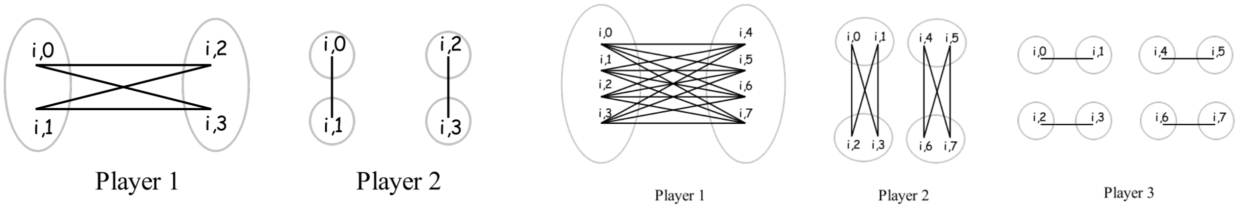
Player 2 places all edges between $V_{i,0,0}$ and $V_{i,0,1}$ and edges between $V_{i,1,0}$, $V_{i,1,1}$ if $i \in S_2$.

The edges and partitions are shown in Figure 1a.

If $S_1 \cap S_2 = \{i\}$, then there is a clique on vertex set V_i (which is of size 4). If $S_1 \cap S_2 = \emptyset$, since both Player 1 and Player 2 have only bipartite graph edges on disjoint vertex sets, the output graph is triangle free.

If there is a one-pass streaming algorithm A for CLIQUE-GAP$(4, 2)$ that distinguishes whether the input graph G has clique of size 4 or triangle-free, the players can use this algorithm to solve $\text{DISJ}_2^{n/4}$ as follows: Player 1 runs A on his edge set and communicates the content of the working memory at the end of his computation to Player 2. Player 2 continues to run the algorithm on his edge set and outputs the result of the algorithm as the answer of the DISJ problem. Hence if A uses space M, then total communication between players $\leq M$ (in general if there are k players we have the inequality: total communication $\leq (k-1)M$). This leads to the required lower bound.

The edge decomposition for the reduction from $\text{DISJ}_3^{n/8}$ to CLIQUE-GAP$(8, 2)$ is shown in Figure 1b.



(a)

(b)

Fig. 1: (a) The decomposition of K_4 to $\log_2 4 = 2$ bipartite graphs. (b) The decomposition of of K_8 to $\log_2 8 = 3$ bipartite graphs.

For obtaining a lower bound on the space complexity of CLIQUE-GAP(r, s), we will reduce $\text{DISJ}_{\lceil \log_s r \rceil}^{n/r}$ to CLIQUE-GAP(r, s) and use the lower bound stated in Theorem 6. For the reduction, we give an extension of the bipartite graph decomposition result. In particular, we show (implicitly) that $\alpha_*(K_r, \mathcal{H}) \leq \lceil \log_s r \rceil$ where $\mathcal{H}$ is the class of all s-partite graphs.

Proof. (***of Theorem 2***) We will reduce $\text{DISJ}_t^{n/r}$ to CLIQUE-GAP(r, s) where $t = \lceil \log r / \log s \rceil$. Consider an instance of $\text{DISJ}_t^{n/r}$, where Player l holds a set $S_l \subset [n/r]$ for $l = 1, 2, \ldots, t$. To construct an instance G on n vertices of CLIQUE-GAP(r, s), for $l = 1, \ldots, t$, Player l places an edge set E_l as described below.

The Construction of E_l: The construction follows the same pattern as in the figures above. To explain it precisely we need to structure the vertex set of the graph in certain way. W.l.o.g set $r = s^t$ and $n = 0$ mod r. We will denote an integer in $[r]$ by its s-ary representation using a t-tuple. We denote the n vertices by $V = \{v_{i,[j_1,j_2,\ldots,j_t]} | i = 1, 2, 3, \ldots, n/r, \text{ for all } j_1, j_2, \ldots, j_t \in [s]\}$ ($[j_1, j_2, \ldots, j_t]$ represents an integer in $[r]$ uniquely). This notation partitions the set V into n/r subsets, each of size r. We denote them as $V_1, V_2, \ldots, V_{n/r}$. That is, for each fixed $i = 1, 2, \ldots, n/r$, $V_i = \{v_{i,[j_1,j_2,\ldots,j_t]} | \text{for all } j_1, j_2, \ldots, j_t \in [s]\}$. Next we define a series of s partitions of each V_i where l^{th} partition is a refinement of the $(l-1)^{th}$ partition.

Partition 1: $V_i = V_{i,0} \cup V_{i,1} \ldots \cup V_{i,s-1}$, where for each fixed $j_1 \in [s]$

$$V_{i,j_1} \equiv \{v_{i,[j_1,j_2,j_3,\ldots,j_t]} | \text{ for all } j_2, j_3, \ldots, j_t \in [s]\}. \tag{2}$$

Partition l: For each set $V_{i,j_1,j_2,\ldots,j_{l-1}}$ in Partition $(l-1)$, partition $V_{i,j_1,j_2,\ldots,j_{l-1}} = V_{i,j_1,j_2,\ldots,j_{l-1},0} \cup V_{i,j_1,j_2,\ldots,j_{l-1},1} \ldots \cup V_{i,j_1,j_2,\ldots,j_{l-1},s-1}$ as s subsets, each of which is of size s^{t-l}. Here, for each fixed $i = 1, 2, \ldots, n/r$ and for each fixed $j_1, j_2, \ldots, j_l \in [s]$, we have

$$V_{i,j_1,j_2,\ldots,j_l} \equiv \{v_{i,[j_1,j_2,j_3,\ldots,j_l,j_{l+1},\ldots,j_t]} | \text{ for all } j_{l+1}, j_{l+2}, \ldots, j_t \in [s]\}. \tag{3}$$

With this structuring of vertices, we can now define E_l for each Player l. If an element i is in the set S_l, then for all $j_1, j_2, \ldots, j_{l-1} \in [s]$, Player l has all the s-partite graph edges between the s partitions of the vertex set $V_{i,j_1,j_2,\ldots,j_{l-1}}$, namely, $V_{i,j_1,j_2,\ldots,j_{l-1},0}$, $V_{i,j_1,j_2,\ldots,j_{l-1},1}$, $V_{i,j_1,j_2,\ldots,j_{l-1},3}, \ldots$ and $V_{i,j_1,j_2,\ldots,j_{l-1},s-1}$. Formally,

$$E_l = \cup_{i \in S_l} \cup_{j_1,j_2,\ldots,j_{l-1} \in [s]} E(i, j_1, j_2, \ldots, j_{l-1}), \tag{4}$$

where

$$E(i, j_1, j_2, \ldots, j_{l-1}) \equiv \cup_{j_l, j'_l \in [s], j_l \neq j'_l} \{(a, b) | \text{ for all } a \in V_{i,j_1,j_2,\ldots,j_{l-1},j_l}, b \in V_{i,j_1,j_2,\ldots,j_{l-1},j'_l}\}. \tag{5}$$

Note that each edge appears only in one of the edge set. **End of Construction of E_l.**
Correctness of the Reduction: On a negative instance, players' input sets $S_1, S_2 \ldots S_t$ are pairwise disjoint. The above construction builds all the s-partite graphs on disjoint sets of vertices, hence the output graph is s-partite and hence $(s+1)$-clique free.

On a positive instance, players' input sets have a unique intersection, $S_1 \cap S_2 \ldots \cap S_t = \{i\}$. For each Player l, the edge set E_l includes all the s-partite graph edges on each vertex set $V_{i,j_1,j_2,\ldots,j_{l-1}}$, i.e. $\cup_{j_1,j_2,\ldots,j_{l-1} \in [s]} E(i, j_1, j_2, \ldots, j_{l-1})$. We claim that there is a r-clique on vertex set V_i. Consider any two distinct vertices $u, v \in V_i$, where $u = v_{i,[j_1,j_2,\ldots,j_t]}$, $v = v_{i,[j'_1,j'_2,\ldots,j'_t]}$. Since $u \neq v$, $(j_1, j_2, \ldots, j_t) \neq (j'_1, j'_2, \ldots, j'_t)$. Let q be first integer such that $j_q \neq j'_q$. By the definition of the partitions, $u \in V_{i,j_1,j_2,\ldots,j_{q-1},j_q}$ and $v \in V_{j,j_1,j_2,\ldots,j_{q-1},j'_q}$. Therefore, there is an edge (u, v) in the edge set output by Player q.
Proof of the Bound: Suppose there is a one-pass streaming algorithm $\mathcal{A}$ that solves CLIQUE-GAP(r, s) in $M(n, r, s)$ space. Then consider the following one-way protocol for DISJ$_t^{n/r}$. For each $1 \leq l < t$, Player l simulates $\mathcal{A}$ on his edge set E_l and communicates the memory content to Player $(l+1)$. Finally Player t simulates $\mathcal{A}$ on E_t and outputs the result of $\mathcal{A}$. The total communication $\leq (t-1)M(n, r, s)$. Hence from the known lower bound on DISJ$_t^{n/r}$, we have that $M(n, r, s) = \Omega(n/rt^2) = \Omega(n/r \log_s^2 r)$. Now consider the hard instance of DISJ$_t^{n/r}$, any player holds a non-empty set (otherwise this is an easy instance). From the construction, for each hard instance we know $m = \Omega(r^2 \times n/r) = \Omega(nr)$. Hence any one-pass streaming algorithm that solves CLIQUE-GAP(r, s) requires $\Omega(m/r^2 \log_s^2 r)$ space. We further justify this argument by the following modification of the reduction.
Exchange Quantifier n to m: The above construction of a lower bound is based on the quantifier n. Suppose we are given m, r, s, we can construct a reduction graph for DISJ$_t^{m/2r}$ as following. We construct a graph on m/r vertices. Without loss of generality, assume $r = o(\sqrt{m})$, otherwise the bound is trivially $\Omega(1)$. The construction is the same for the first $n = m/2r$ vertices. For each player, in addition to sending the memory content of algorithm, the player also sends the number of edges in the current graph. By the above analysis, for the last player, the graph will have $m' \leq m/2$ edges. The last player add $(m - m') = O(m)$ edges to the last n vertices without creating an s-clique. This can be done since by Turán's theorem, an $(m/2r)$-vertices graph can have up to $(1 - 1/s)m^2/8r^2 = \omega(m)$ edges without creating an s-clique. The lower bound is the same with the previous analysis. By picking up graphs constructed for the hard instance for DISJ problem, we construct the graph classes as required in the theorem.

A Lower Bound to The General GAP Problem

Using the terminology from graph decomposition theory we prove a general lower bound theorem for the promise problem GAP$(\mathcal{P}, \mathcal{Q})$ which is defined as follows.

Definition 2. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two graph properties (equivalently, $\mathcal{P}$ and $\mathcal{Q}$ are two sets of graphs) such that $\mathcal{P} \cap \mathcal{Q} = \emptyset$. Given an input graph G, an algorithm for GAP$(\mathcal{P}, \mathcal{Q})$ should output "1" if $G \in \mathcal{P}$ and '0' if $G \in \mathcal{Q}$. For $G \notin \mathcal{P} \cup \mathcal{Q}$, the algorithm can output "1" or "0".*

We first recall the necessary definitions. Let $\mathcal{H}$ be a specified class of graphs. An $\mathcal{H}$-decomposition[3] of a graph G is the decomposition of G into subgraphs $G_1, G_2, \ldots, G_t$ such that any edge in G is an edge of

[3] Note that some papers define the decomposition on connected graph. We here use a more general statement.

exactly one of the G_i's and all G_is belong to $\mathcal{H}$. Define $\alpha_*(G, \mathcal{H})$ as:

$$\alpha_*(G, \mathcal{H}) \equiv \min_D |D| \tag{6}$$

where $D = \{G_1, G_2, \ldots, G_t\}$ is an $\mathcal{H}$-decomposition of G. For convenience, we define $\alpha_*(G, \mathcal{H}) = \infty$ if the $\mathcal{H}$-decomposition of G is not defined.

Theorem 3. *Let $\mathcal{P}, \mathcal{Q}$ be two graph properties such that*

- *$\mathcal{P} \cap \mathcal{Q} = \emptyset$;*
- *If $G'' \in \mathcal{P}$ and G'' is a subgraph of G', then $G' \in \mathcal{P}$;*
- *If $G', G'' \in \mathcal{Q}$ and $V(G') \cap V(G'') = \emptyset$, then $\tilde{G} = (V(G') \cup V(G''), E(G') \cup E(G'')) \in \mathcal{Q}$;*

Let G_0 be an arbitrary graph in $\mathcal{P}$. Given any graph G with m edges and n vertices, if a one-pass streaming algorithm $\mathcal{A}$ solves GAP$(\mathcal{P}, \mathcal{Q})$ *correctly with probability at least $3/4$, then $\mathcal{A}$ requires $\Omega(\frac{n}{|V(G_0)|}\frac{1}{\alpha_*^2(G_0, \mathcal{Q})})$ space in the worst case.*

Remark 1. We note that in the above statement G_0 is an arbitrary graph. To get the optimal bound, we can select a G_0 such that the denominator $|V_0|\alpha_*^2(G, \mathcal{Q})$ of the bound is minimized. We also note that this theorem is indeed a generalization of Theorem 2. Let $\mathcal{P} = \{G \mid G \text{ has a } r\text{-clique }\}$ and $\mathcal{Q} = \{G \mid G \text{ has no } (s+1)\text{-clique }\}$. In the proof of Theorem 2 we use $G_0 = K_r$ and shows that $\alpha_*(K_r, \mathcal{Q}) \leq \log_s r$ (in this case $m = O(nr)$).

Proof. (***of Theorem 3***). Denote $V_0 = V(G_0)$ and $E_0 = E(G_0)$. Suppose a streaming algorithm $\mathcal{A}$ solves GAP$(\mathcal{P}, \mathcal{Q})$ with probability at least $3/4$ using M bits of memory. We can use $\mathcal{A}$ to construct a communication protocol that solves $\text{DISJ}_t^{n/|V_0|}$, where $t = \alpha_*(G_0, \mathcal{Q})$.

The protocol works the same way as Theorem 2, except now each Player l is given a set $S_l \subset [n/|V_0|]$. We construct the input edge set E_l to the GAP problem of Player l as follows. Label the n vertices as $V = \{v_{i,j} | \text{for all } i \in [n/|V_0|], j \in V_0\}$. This notation partitions the vertices as $n/|V_0|$ subsets, $V = V_1 \cup V_2 \ldots \cup V_{n/|V_0|}$ each of which is of size $|V_0|$. For a fixed $i = 1, 2, \ldots, n/|V_0|$, denote $V_i = \{v_{i,j} | \text{ for all } j \in V_0\}$. Let $D = \{G_1^0, G_2^0, \ldots, G_t^0\}$ be the optimal $\mathcal{Q}$-decomposition of G_0 such that $|D| = \alpha_*(G_0, \mathcal{Q})$. Denote each G_l^0 as (V_l^0, E_l^0).

For Player l, if i is in her input set S_l, then she has the following edge set:

$$E_l(i) \equiv \cup_{(a,b) \in E_l^0} \{(v_{i,a}, v_{i,b})\}, \tag{7}$$

which is a copy of E_l^0 on vertices V_i. Let the set of all edges that Player l has be

$$E_l = \cup_{i \in S_l} E_l(i). \tag{8}$$

Clearly $\{E_1(i), E_2(i), \ldots, E_t(i)\}$ is a $\mathcal{Q}$-decomposition of the copy of G_0 on vertices V_i.

On a positive instance, Players' input sets uniquely intersect, $S_1 \cap S_2 \cap \ldots S_t = \{i\}$. Each Player l's edge set contains $E_l(i)$. The final stream contains a sub graph G_0' induced by $\cup_{l=0}^t E_l(i)$ on vertices V_i such that G_0' is a copy of G_0, hence $G_0' \in \mathcal{P}$. By definition, the constructed graph $G \in \mathcal{P}$.

On a negative instance, Players' input sets $S_1, S_2 \ldots S_t$ are pairwise disjoint, let $S' = S_1 \cup S_2 \ldots \cup S_t$. For each $i \in S'$, there exists an unique l such that $i \in S_l$. Therefore, only Player l outputs the edge sets $E_l(i)$, which induces a graph from $\mathcal{Q}$. The final graph is given by $\{\cup_{i \in S'} V_i, \cup_{i \in S'} E_l(i)\}$. The sub-graphs induced by the V_is are vertex disjoint, and therefore the constructed graph $G \in \mathcal{Q}$.

If $\mathcal{A}$ can decide whether $G \in \mathcal{P}$ or $G \in \mathcal{Q}$ with probability at least $3/4$, as in the the proof of Theorem 2, players can simulate $\mathcal{A}$ to solve any given instance of $\text{DISJ}_t^{n/|V_0|}$ with probability at least $3/4$, using the above reduction. If M is the memory used by A, then by Theorem 6, $(t-1)M \geq \Omega(n/(t|V_0|))$. Hence we have $M = \Omega(n/(|V_0|\alpha_*^2(G_0, \mathcal{Q})))$.

5 Relation to Triangle Counting

For triangle counting problem, given a graph G with at least T triangles, Cormode and Jowhari [12] give a two-pass algorithm using $O(m/\sqrt{T})$ space. Also, for $T \leq n^2$ they provide a matching lower bound. Pavan et al. [23] provide a one-pass streaming algorithm for triangle counting with space complexity of $O(m\Delta/T)$, where Δ is the max-degree of the graph G. We use the tools we develop for the CLIQUE-GAP problem to give a new two-pass algorithm to distinguish between graphs with at least T triangles and triangle-free graphs. For $T = n^{2+\beta}$, the space complexity of our algorithm is $o(m/\sqrt{T})$ for $\beta > 2/3$. Our results demonstrate that for some $T > n^2$, it might be possible to refine the lower bound of Cormode and Jowhari.

Algorithm 2 DETECT(G, p_1, p_2, s_1): Procedure of Detecting Triangles

```
Input:
    Graph edge stream $\langle e_1, e_2, \ldots, e_m \rangle$ of graph $G = (V, E)$. Real number $p_1, p_2 \in [0, 1]$,
    integer $s_1$.
Output:
    "1" if a triangle detected in $G$; "0" if not.
Initialize:
    Set memory buffer $M_i$ for $i = 1, 2, \ldots, s_1$ empty.
    Computes $s_1$ independent random binary size-$n$ vectors $Q_i = \{Q_{iv} | \text{for all } v \in V\}$ for
    $i = 1, 2, \ldots s_1$ using $O(s_1 \log n)$ space such that for a fixed $i$, each $Q_{iv}$ is pairwise
    independent and $Pr[Q_{iv} = 1] = p_1$.
while not the end of the stream do
    Read an edge $e = (u, v)$.
    for $i = 1, 2, \ldots, s_1$ do
        Draw a bit $c_e$ from $\{0, 1\}$ independently, such that $Pr[c_e = 1] = p_2$.
        If $c_e = 1$ and either $Q_{iv} = 1$ or $Q_{iu} = 1$ , then insert $e$ to $M_i$.
        If $e$ completes a triangle with 2 other edges in $M_i$, then output "1".
Output "0".
```

Theorem 4. *Let $\mathcal{G}_1$ be a class of graphs of n vertices that has at least $T = n^{2+\beta}$ triangles for some $\beta \in [0, 1]$. Let $\mathcal{G}_2$ be a class of graphs of n vertices and triangle-free. Given graph $G = (V, E)$ with n nodes and m edges, there is a two-pass streaming algorithm that distinguishes whether $G \in \mathcal{G}_1$ or $G \in \mathcal{G}_2$ with constant probability using $\tilde{O}(\frac{mn^{2-\beta}}{T})$ space. In particular, for $\beta > 2/3$, the algorithm uses $o(m/\sqrt{T})$ space.*

To show our bound, we need the following notation.

Let $G = (V, E)$ be a graph with T triangles. For each $u \in V$, $\tau(u)$ is the number of triangles that have u as a node. Let $\tilde{V} \subseteq V$ be the set of vertices that are nodes of at least one triangle. Partition $\tilde{V}$ into $t = O(\log |\tilde{V}|)$ sets as $\tilde{V} = S_0 \cup S_1 \cup S_2 \ldots \cup S_t$ where each $S_i = \{a \in V | 2^i \leq \tau(a) < 2^{i+1}\}$.

Claim. There exists an i such that $|S_i| \cdot 2^{i+1} > \frac{3T}{\log n}$.

Proof. This follows from the following observation,

$$3T < \sum_{i=1}^{t} |S_i| \cdot 2^{i+1} \leq 6T \tag{9}$$

since each triangle counts 3 times.

Definition 3. *Define $i(G) = \min\{i : |S_i| 2^{i+1} \geq \frac{3T}{\log n}\}$, the* significant index *of graph G.*

Lemma 1. *Let $\mathcal{G}_1$ be a class of graphs of n vertices that has at least $T = n^{2+\beta}$ triangles for some $\beta \in [0,1]$ and there is an integer $i_0 \leq 2\log n - 1$ such that $i(G) \leq i_0$ for all $G \in \mathcal{G}_1$. Let $\mathcal{G}_2$ be a class of graphs of n vertices and triangle-free. Given graph $G = (V,E)$ with n nodes and m edges, there is a two-pass streaming algorithm that distinguishes whether $G \in \mathcal{G}_1$ or $G \in \mathcal{G}_2$ with constant probability using $\tilde{O}(\frac{mn^2 2^{i_0+1}}{T^2})$ space.*

Proof. Let $\hat{T}(G)$ be the number of triangles in G. Suppose $G \in \mathcal{G}_1$ Denote $n^\alpha = \frac{\hat{T}(G)}{2^{i_0+1}\log n}$ for some $\alpha > 0$, by definition of significant index, we have

$$\frac{3\hat{T}(G)}{\log n \cdot |S_{i(G)}|} \leq 2^{i(G)+1} \leq 2^{i_0+1} = O(n^2). \tag{10}$$

Hence

$$\begin{aligned} n \geq |S_{i(G)}| \geq \frac{3\hat{T}(G)}{2^{i(G)+1}\log n} \geq \frac{3\hat{T}(G)}{2^{i_0+1}\log n} &= 3n^\alpha \\ &= \tilde{\Omega}(\frac{T}{n^2}) = \tilde{\Omega}(n^\beta). \end{aligned} \tag{11}$$

We have $\beta \leq \alpha \leq 1$. On the other hand for any $u \in S_{i(G)}$,

$$\tau(u) = \Theta(2^{i(G)}) = \tilde{\Omega}\left(\frac{\hat{T}(G)}{n}\right) = \tilde{\Omega}(n^{1+\beta}). \tag{12}$$

We now construct an algorithm that distinguishes whether $\mathcal{G} \in \mathcal{G}_1$ or $\mathcal{G} \in \mathcal{G}_2$ using Algorithm 2 as follows. Let $p_1 = \Theta(1/n^\alpha)$, $p_2 = \Theta(1/n^\beta)$, and s_1 be some sufficiently large positive integer. Make the first pass over the stream using Algorithm 2 and keep the memory. If a triangle is detected, halt the algorithm, output "1". If not we make another pass to check if any edge can complete a triangle with the edges we have already stored in the memory.

If $G \in \mathcal{G}_2$, the algorithm will be guaranteed output "0", since no triangle will be sampled.

If $G \in \mathcal{G}_1$, the guarantee of the first pass is:
In the node sampling step of the algorithm, with constant probability, we sample a node u from $S_{i(G)}$;
In the edge sampling step of the algorithm, we claim the algorithm samples a 2 edges of a triangle sharing the same node u from G with constant probability by following. Let $T(u)$ be the set of triangles that have u as a node. Let X be the minimum edge set that any triangle $t \in T(u)$ touches an edge in X. We claim $|X| = \Omega(n^\beta)$ by

$$2\tau(u) \leq \sum_{(u,v)\in X} |T(u,v)| \leq n|X|, \tag{13}$$

where $T(u,v)$ is the set of triangles that have nodes u, v, hence of size at most n. We now partition $X = X_0 \cup X_1 \ldots \cup X_l$ as l sets where $l = \Theta(\log n)$, such that each X_a is defined as $\{(u,v) : 2^a \leq |T(u,v)| < 2^{a+1}\}$. By similar argument, there exists an a_0 such that $|X_{a_0}|2^{a_0+1} > \frac{3|\hat{T}(G)|}{\log n}$. Since $n \geq 2^{a_0} \geq \frac{|\hat{T}(G)|}{|X_{a_0}|\log n}$, $|X_{a_0}| = \tilde{\Omega}(n^{1+\beta})$ and $|T(u,v)| = \Theta(2^{a_0}) = \tilde{\Omega}(\frac{\hat{T}(G)}{n^2}) = \tilde{\Omega}(n^\beta)$ for each $(u,v) \in X_{a_0}$, where we use $|X_a| \leq \binom{n}{2}$. Therefore, with $p_2 \geq 1/n^\beta$, there's $\Omega(1)$ probability sampling and edge from X_{a_0} and a constant probability of sampling another edge from its neighbor edges that form a triangle.

The probability sampling an edge is

$$p_1 p_2 = \Theta\left(\frac{1}{n^\alpha n^\beta}\right) = \Theta\left(\frac{n^{2-\alpha}}{T}\right). \tag{14}$$

The expected space used in this algorithm is $O\left(\frac{mn^{2-\alpha}}{T}\right) = \tilde{O}(\frac{mn^2 2^{i_0+1}}{T^2})$.

Proof (Of Theorem 4). The theorem follows from using the algorithm in Lemma 1 and set $\alpha = \beta$.

For $\alpha + \beta/2 > 1$ (e.g. $\beta > 2/3$), we have $n^{2-\alpha} = o(n^{1+\beta/2}) = o(\sqrt{T})$, the algorithm provided by Theorem 4 obtains a space bound $o(\frac{m}{\sqrt{T}})$ for the triangle distinguish problem.

6 Incidence Model

In designing algorithms for graph streams, researchers have also considered the *incidence model*. This model assumes that the graph $G = (V, E)$ is presented as a stream of incidence lists $\{(v, E_v)\}_{v \in V}$ where E_v is the set of edges incident on the vertex v. This is a valid assumption since in many situations it is natural to store a graph as an array of incidence lists.

Since the incidence model is a restriction of the adjacency stream model, our upper bound of $O(ms^2/r^2)$ for CLIQUE-GAP(r, s) holds in this model also. Here we prove a lower bound for CLIQUE-GAP$(r, 2)$ in the incidence model.

Theorem 5. *If one-pass streaming algorithm solves* CLIQUE-GAP$(r, 2)$ *in the incidences model for any G with m edges and n vertices with probability at least $3/4$, it requires $\Omega(m/r^3)$ space in the worst case.*

Proof. We will reduce DISJ$_r^{n/r}$ to CLIQUE-GAP$(r, 2)$. Given an instance of DISJ$_r^{n/r}$, construct an instance $G = (V, E)$ of CLIQUE-GAP$(r, 2)$ as follows. We label the vertices in V as $v_{i,j}$ with each $i \in [n/r], j \in [r]$. Assuming each Player $j = 1, 2, \ldots, r$ is given a set $S_j \subset [n/r]$, Player j has the set of edges $E_j = \{(v_{i,j}, v_{i,l}) | \text{for all } i \in S_j, l \in [r], \text{s.t. } l \neq j\}$ (a set of $(r-1)$-stars). Note that each edge only appears in one of these sets. Since for each vertex, all edges incident to that vertex is known by the players, the players can output the edges in a incidence list form.

Let G be the graph induced by $E_1 \cup E_2 \ldots \cup E_r$. On a negative instance, $S_1, S_2 \ldots S_r$ are pairwise disjoint, and hence G contains only $(r-1)$-stars. On a positive instance, $S_1 \cap S_2 \cap \ldots \cap S_r = \{i\}$, and hence G contains a r-clique on vertices $v_{i,1}, v_{i,2}, \ldots, v_{i,r}$. Therefore, using arguments similar to our other lower bound arguments, if there an algorithm for CLIQUE-GAP$(r, 2)$ that uses M space, by Theorem 6, $M = \Omega(n/r^3)$. In the cases of positive and negative instances, the number of edges $m = O(n)$ and $m = O((r-1)n/r + r^2/2)$, respectively. Therefore any one-pass algorithm in the incidence model for CLIQUE-GAP$(r, 2)$ requires $\Omega(m/r^3)$ space.

References

1. Ahn, K.J., Guha, S.: Graph sparsification in the semi-streaming model. In: In Proceedings of the 36th International Colloquium on Automata,Languages and Programming: Part II. pp. 328–338. ICALP, Springer-Verlag (2009)
2. Alon, N., Krivelevich, M., Sudakov, B.: Finding a large hidden clique in a random graph. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 594–598. ACM/SIAM (1998)
3. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 20–29. ACM (1996)
4. 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. pp. 623–632. SODA, Society for Industrial and Applied Mathematics (2002)
5. Buriol, L.S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: Computing clustering coefficients in data streams. In: European Conference on Complex Systems (ECCS) (2006)
6. Buriol, L.S., 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. PODS, ACM (2006)
7. Buriol, L.S., Frahling, G., Leonardi, S., Sohler, C.: Estimating clustering indexes in data streams. In: ESA. Lecture Notes in Computer Science, vol. 4698, pp. 618–632. Springer (2007)
8. Buriol, L.S., Frahling, G., Leonardi, S., Spaccamela, A.M., Sohler, C.: Counting graph minors in data streams. Tech. rep., DELIS – Dynamically Evolving, Large-Scale Information Systems (2005)
9. Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In: IEEE Conference on Computational Complexity. pp. 107–117. IEEE Computer Society (2003)
10. Chung, F.: On the decomposition of graphs. SIAM Journal on Algebraic Discrete Methods 2(1), 1–12 (1981)
11. Chung, F., Erdős, P., Spencer, J.: On the decomposition of graphs into complete bipartite subgraphs. In: Studies in Pure Mathematics, pp. 95–101. Birkhäuser Basel (1983)

12. Cormode, G., Jowhari, H.: A second look at counting triangles in graph streams. Theoretical Computer Science 552(0), 44 – 51 (2014)
13. Demetrescu, C., Finocchi, I., Ribichini, A.: Trading off space for passes in graph streaming problems. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms. pp. 714–723. ACM (2006)
14. Donoho, D.L.: Compressed sensing. IEEE Transactions on Information Theory 52, 1289–1306 (2006)
15. Goel, A., Kapralov, M., Khanna, S.: On the communication and streaming complexity of maximum bipartite matching. In: Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 468–485. SODA, SIAM (2012)
16. Halldórsson, M.M., Sun, X., Szegedy, M., Wang, C.: Streaming and communication complexity of clique approximation. In: ICALP Proceedings of the 39th international colloquium conference on Automata, Languages, and Programming - Volume Part I. pp. 449–460. Springer (2012)
17. Jha, M., Seshadhri, C., Pinar, A.: When a graph is not so simple: Counting triangles in multigraph streams. CoRR abs/1310.7665 (2013), http://arxiv.org/abs/1310.7665
18. Jowhari, H., Ghodsi, M.: New streaming algorithms for counting triangles in graphs. In: COCOON. Lecture Notes in Computer Science, vol. 3595, pp. 710–716. Springer (2005)
19. Kapralov, M., Khanna, S., Sudan, M.: Streaming lower bounds for approximating MAX-CUT. CoRR abs/1409.2138 (2014), http://arxiv.org/abs/1409.2138
20. Kutzkov, K., Pagh, R.: On the streaming complexity of computing local clustering coefficients. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. pp. 677–686. WSDM, ACM (2013)
21. Manjunath, M., Mehlhorn, K., Panagiotou, K., Sun, H.: Approximate counting of cycles in streams. In: Proceedings of the 19th European Conference on Algorithms. pp. 677–688. ESA, Springer-Verlag (2011)
22. McGregor, A.: Graph stream algorithms: A survey. SIGMOD Rec. 43(1), 9–20 (2014)
23. Pavan, A., Tangwongsan, K., Tirthapura, S., Wu, K.L.: Counting and sampling triangles from a graph stream. Proceedings of the VLDB Endowment 6(14), 1870–1881 (2013)
24. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the facebook social graph. CoRR abs/1111.4503 (2011), http://arxiv.org/abs/1111.4503