The Sparse Awakens: Streaming Algorithms for Matching Size Estimation in Sparse Graphs

Graham Cormode[1], **Hossein Jowhari**[2], **Morteza Monemizadeh**[3], **and S. Muthukrishnan**[4]

1 **University of Warwick, UK.** `g.cormode@warwick.ac.uk.` [*]
2 **University of Warwick, UK.** `H.Jowhari@warwick.ac.uk.` [†]
3 **Amazon, Palo Alto, CA, USA.** `mmorteza@amazon.com.` [‡]
4 **Rutgers University, Piscataway, NJ, USA.** `muthu@cs.rutgers.edu.`

Abstract

Estimating the size of the maximum matching is a canonical problem in graph analysis, and one that has attracted extensive study over a range of different computational models. We present improved streaming algorithms for approximating the size of maximum matching with sparse (bounded arboricity) graphs.

- *(Insert-Only Streams)* We present a one-pass algorithm that takes $O(\alpha \log n)$ space and approximates the size of the maximum matching in graphs with arboricity α within a factor of $O(\alpha)$. This improves significantly upon the state-of-the-art $\tilde{O}(\alpha n^{2/3})$-space streaming algorithms, and is the first poly-logarithmic space algorithm for this problem.
- *(Dynamic Streams)* Given a dynamic graph stream (i.e., inserts and deletes) of edges of an underlying α-bounded arboricity graph, we present an one-pass algorithm that uses space $\tilde{O}(\alpha^{10/3} n^{2/3})$ and returns an $O(\alpha)$-estimator for the size of the maximum matching on the condition that the number edge deletions in the stream is bounded by $O(\alpha n)$. For this class of inputs, our algorithm improves the state-of-the-art $\tilde{O}(\alpha n^{4/5})$-space algorithms, where the $\tilde{O}(.)$ notation hides logarithmic in n dependencies.

In contrast to prior work, our results take more advantage of the streaming access to the input and characterize the matching size based on the ordering of the edges in the stream in addition to the degree distributions and structural properties of the sparse graphs.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases streaming algorithms; matching size

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.<1>

1 Introduction

In this paper, we address a core graph analysis question of finding the size of a maximum matching, using space asymptotically smaller than even the number of nodes. Graphs naturally capture relationships between entities, whether entities of the same type (simple graphs), of two types (bipartite graphs), or other combinations of types (encoded via multigraphs and hypergraphs). In modern applications, it is not uncommon to encounter

[*] Supported in part by European Research Council grant ERC-2014-CoG 647557, The Alan Turing Institute under the EPSRC grant EP/N510129/1, and a Royal Society Wolfson Research Merit Award.
[†] Supported by European Research Council grant ERC-2014-CoG 647557.
[‡] Work was done when the author was at Rutgers University, Piscataway, NJ, USA.

graphs with many millions or billions of nodes (capturing the huge number of entities that can interact), and billions to trillions of edges (enumerating the vast number of possible interactions). This has led to significant interest in addressing traditional graph analysis problems in novel computational models: external memory, parallel and streaming models.

Problems related to (maximum) matchings in graph have a long history in Computer Science. They arise in many contexts, from choosing which advertisements to display to online users [34], to characterizing properties of chemical compounds [42]. Stable matchings have a suite of applications, from assigning students to universities, to arranging organ donations [41]. These have been addressed in a variety of different computational models, from the traditional RAM model, to more recent sublinear (property testing [38]) and external memory/parallel (e.g. MapReduce [25]) models. Matching has also been studied for a number of classes of input graph, including general graphs, bipartite graphs, weighted graphs, and those with some sparsity structure.

Our work focuses on the streaming case, where each edge is seen once only, and we are restricted to space sublinear in the size of the graph (ie., the number of vertices). This captures the scenario when the number of edges is overwhelmingly large, such as when analyzing connections between a massive number of individuals in a communication or social network. Now the objective is to find (approximately) the *size* of the matching. That is, while we cannot hope to retrieve a full description of the matching in sublinear space, we can hope to estimate how big the matching is. Even here, results for general graphs are either weak or make assumptions about the input or the stream order. In this work, we seek to improve the guarantees by restricting to graphs that have some measure of sparsity – bounded arboricity, or bounded degree. This aligns with reality, where most massive graphs have asymptotically fewer than $\Theta(n^2)$ edges. For example, in graphs that arise in the context of social networks, most nodes have a degree that is less than a few hundred, as people can only maintain this number of active connections (Dunbar's number), although a few nodes ("celebrities") have very high (in-)degree in the multi-millions.

Estimating the matching size for graphs in the streaming model has been the subject of some study in the algorithms and data analysis community in recent years. Kapralov, Khanna, and Sudan [21] developed a streaming algorithm which computes an estimate of matching size for general graphs within a factor of $O(\text{polylog}(n))$ in the *random-order* streaming model using $O(\text{polylog}(n))$ space. In the random-order model, the input stream is assumed to be chosen uniformly at random from the set of all possible permutations of the edges. Esfandiari *et al.* [15] were the first to study streaming algorithms for estimating the size of matching in bounded arboricity graphs in the *adversarial-order* streaming model, where the algorithm is required to provide a good approximation for any ordering of edges. Graph arboricity is a measure to quantify the density of a given graph. A graph $G(V, E)$ has arboricity α if the set E of its edges can be partitioned into at most α forests. Since a forest on n nodes has at most $n-1$ edges, a graph with arboricity α can have at most $\alpha(n-1)$ edges. Indeed, by a result of Nash-Williams [36, 37] this holds for any subgraph of a α-bounded arboricity graph G. Formally, the Nash-Williams Theorem [36, 37] states that

$$\alpha = \max_{U \subseteq V} \frac{|E(U)|}{(|U|-1)},$$

where $|U|$ and $|E(U)|$ are the number of nodes and edges in the subgraph induced by the nodes U, respectively. Several important families of graphs have constant arboricity. Examples include planar graphs (that have arboricity $\alpha = 3$), bounded genus graphs, bounded treewidth graphs, and more generally, graphs that exclude a fixed minor.[1]

The important observation in [15] is that the size of matching in bounded arboricity graphs can be approximately characterized by the number of high degree vertices (vertices with degree above a fixed threshold) and the number of so-called *shallow edges* (edges with both low degree endpoints). This characterization allows for estimation of the matching size in sublinear space by taking samples from the vertices and edges of the graph. The work of [15] implements the characterization in $\tilde{O}(\alpha n^{2/3})$ space and gives a $O(\alpha)$ approximation of the matching size. Subsequent works [6, 30] consider alternative characterizations and improve upon the approximation factor however they do not result in major space improvements.

1.1 Our Contributions

We present major improvements in the space usage of streaming algorithms for sparse graphs (α-bounded Arboricity Graphs). Our main result is a polylog space algorithm that beats the n^{ε} space bound of prior algorithms. More precisely, we show:

▶ **Theorem 1.** *Let $G(V,E)$ be a graph with arboricity bounded by α. Let S be an (adversarial order) insertion-only stream of the edges of the underlying graph G. Let M^* be the size of the maximum matching of G (or S interchangeably). Then, there is a randomized 1-pass streaming algorithm that outputs a $(22.5\alpha + 6)(1+\varepsilon)$-approximation to M^* with probability at least $1-\delta$ and takes $O(\frac{\alpha}{\varepsilon^2}\log n)$ space.*

This result is notable, since it is the first demonstration that the polynomial space cost can be beaten for matching size estimation, and shows a that polylogarithmic space is sufficient for a constant factor approximation. Subsequent to our initial statement of results [10], McGregor and Vorotnikova have provided a new analysis of our algorithm to improve the constants in the approximation factor to achieve a $(\alpha+2)(1+\varepsilon)$ factor [32].

For the case of dynamic streams (i.e, streams of inserts and deletes of edges), we design a different algorithm using $\tilde{O}(\alpha^{10/3}n^{2/3})$ space which improves the $\tilde{O}(\alpha n^{4/5})$-space dynamic (insertion/deletion) streaming algorithms of [6, 7]. The following theorem states this result (proved in Section 3.3).

▶ **Theorem 2.** *Let $G(V,E)$ be a graph with the arboricity bounded by α. Let M^* be the size of the maximum matching of G. Let S be a dynamic stream of edge insertions and deletions of the underlying graph G of length at most $O(\alpha n)$. Let*

$$\beta = \mu\frac{(2\mu)}{(\mu - 2\alpha + 1) + 1} \quad \text{where } \mu > 2\alpha.$$

Then, there exists a streaming algorithm that takes $O(\frac{\beta^{4/3}(n\alpha)^{2/3}}{\varepsilon^{4/3}})$ space in expectation and outputs a $(1+\varepsilon)\beta$ approximation of M^* with probability at least 0.86.

Quite recently Assadi *et al.* [3] gave an $\Omega(\frac{n^{1/2}}{\alpha^{2.5}})$ space lower bound for getting a c-approximation of the matching size in dynamic graph streams with arboricity bounded by α. We obtain our $n^{2/3}$ bound by first defining an algorithm for insert-only streams with as $n^{1/2}$ behavior, which suggests that this could also be feasible in the dynamic setting.

Our algorithms for bounded arboricity graphs are based on two novel *streaming-friendly* characterizations of the maximum maching size. The first characterization is a modification

[1] For any H-minor-free graph, the arboricity is $O(h\sqrt{h})$ where h is the number of vertices of H. [24]

Reference	Graph class	Stream	Approx. Factor *	Space Bound**
[21]	General	Random Order	$O(\text{polylog}(n))$	$O(\text{polylog}(n))$
[15]	Arboricity $\leq \alpha$	Insert-Only	$5\alpha + 9$	$\tilde{O}(\alpha n^{2/3})$
[30]	Arboricity $\leq \alpha$	Insert-Only	$\alpha + 2$	$\tilde{O}(\alpha n^{2/3})$
[6, 7]	Arboricity $\leq \alpha$	Insert/Delete	$O(\alpha)$	$\tilde{O}(\alpha n^{4/5})$
This paper	Arboricity $\leq \alpha$	Insert-Only	$(2\alpha + 1)(2\alpha + 2)$	$\tilde{O}(\alpha^{2.5}\sqrt{n})$
This paper	Arboricity $\leq \alpha$	Insert/Delete	$(2\alpha + 1)(2\alpha + 2)$	$\tilde{O}(\alpha^{10/3}n^{2/3})$
This paper	Arboricity $\leq \alpha$	Insert-Only	$22.5\alpha + 6$	$\tilde{O}(\alpha \log n)$

Table 1 Known results for estimating the size of a maximum matching in data streams.
(*) In some entries, a $(1 + \varepsilon)$ multiplicative factor has been suppressed for concision.
(**) The $\tilde{O}(.)$ notation hides logarithmic in n dependencies.

of the characterization in [15] which approximates the size of the maximum matching by $h_\mu + s_\mu$ where h_μ is defined as the number of high degree vertices (vertices with degree more than a threshold μ) and s_μ is the number of shallow edges (edges with low degree endpoints). While h_μ can be easily approximated by sampling the vertices and checking if they are high degree or not, approximating s_μ in sublinear space is a challenge because in one pass we cannot determine if a sampled edge is shallow or not. The work of [15] resolves this issue by sampling the edges at a high rate and manages to implement their characterization in $\tilde{O}(\alpha n^{2/3})$ space for adversarial insert-only streams.

To bring the space usage down to $\tilde{O}(\alpha^{2.5}n^{1/2})$ (for insert-only streams), we modify the formulation of the above characterization. We still need to approximate h_μ but instead of s_μ we approximate n_μ the number of non-isolated vertices in the induced subgraph G_μ defined over the low degree vertices. Note that s_μ is the number of edges in G_μ. This subtle change of definition turns out to be immensely helpful. Similar to h_μ we only need to sample the nodes and check if their degrees are below a certain threshold or not. However we carry the additional constraint that we have to avoid counting the nodes in G_μ that are isolated (have only high degree nodes as neighbors). To satisfy this additional constraint, our algorithm stores the neighbors of the sampled vertices along with a counter for each that maintains their degree in the rest of the stream. Although we only obtain a lower bound on the degree of the neighbors, as it turns out the lower bound information on the degree is still useful because we can ensure the number of false positives that contribute to our estimate is within a certain limit. As a result, we can approximate $h_\mu + n_\mu$ using $\tilde{O}(\alpha n^{1/2})$ space which gives a $(2\alpha + 1)(2\alpha + 2)$ approximation of the maximum matching size after choosing appropriate values for μ and other parameters. This characterization is of particular importance, as it can be adapted to work under edge deletions as well as long as the number of deletions is bounded by $O(\alpha n)$. Details of the characterization and the associated algorithms are given in Lemma 4 and Section 3.2.

To obtain a polylog(n) space algorithm (and prove the claim of Theorem 1), we give a totally new characterization. This characterization, unlike the previous ones that only depend on the parameters of the graph, also takes the ordering of the edges in the stream into account. Roughly speaking, we characterize the size of a maximum matching by the number of edges in the stream that have few neighbor edges in the rest of the stream. To understand the connection with maximum matching, consider the following simplistic special case. Suppose the input graph G is a forest composed of k disjoint stars. Observe that the maximum matching on this graph is just to pick one edge from each star. We relate this to a combinatorial characterization that arises from the sequence of edges in the stream: no matter how we order the edges of G in the stream, from each star there is exactly one edge that has no neighboring edges in the remainder of the stream (in other words, the last edge of the star in the stream). Our characterization generalizes this idea to graphs with arboricity bounded by α by counting the *γ-good edges*, *i.e.* edges that have at most $\gamma = 6\alpha$ neighbors in the remainder of the stream. We prove this characterization gives an $O(\alpha)$ approximation of the maximum matching size. More important, a nice feature of this characterization is that it can be implemented in polylog(n) space if one allows a $1 + \varepsilon$ approximation. The implementation adapts an idea from the well-known L_0 sampling algorithm. It runs $O(\log n)$ parallel threads each sampling the stream at a different rate. At the end, a thread "wins" that has sampled roughly $\Theta(\log n)$ elements from the γ-good edges (samples the edges with a rate of $\frac{\log n}{k}$ where k is the number of γ-good edges). The threads that under-sample will end up with few edges or nothing while the ones that have oversampled will keep too many γ-good edges and will be terminated as soon as they hit a space threshold as a result. Table 1 summarizes the known and new results for estimating the size of a maximum matching.

1.2 Related Work

We discuss the most relevant work to matching size estimation earlier in the introduction. Here, we give an overview of related work by providing the context of matching and streaming algorithms in general, before focusing in on the most related works at the intersection. The problem of computing the maximum matching of G has been extensively studied in the classical offline model, where we assume we have enough space to store all vertices and edges of a graph $G = (V, E)$. The classical result in this model is the algorithm due to Micali and Vazirani [35] with running time $O(m\sqrt{n})$, where $n = |V|$ and $m = |E|$. Recent work has given improved results for sparse bipartite graphs [27]. A matching of size within $(1 - \varepsilon)$ factor of a maximum cardinality matching can be found in $O(m/\varepsilon)$ time [19, 35]. Recently, Duan and Pettie [12] developed a $(1 - \varepsilon)$-approximate maximum weighted matching algorithm in time $O(m/\varepsilon)$.

The model of streaming data analysis has received a similar level of scrutiny. A survey by McGregor [31] gives an overview of results in the graph streaming model. Many fundamental questions have been tackled: counting the number of occurrences of specific small subgraphs such as triangles [33]; estimating properties of neighborhoods [8]; and using 'sketch' techniques to track local and global properties of graphs like connectivity [2].

The question of *finding* an approximation to the maximum cardinality matching has been extensively studied in the streaming model. An $O(n)$-space greedy algorithm trivially obtains a maximal matching, which is a 2-approximation for the maximum cardinality matching [16]. A natural question is whether one can beat the approximation factor of the greedy algorithm with $O(n\,\text{polylog}(n))$ space. Recently, it was shown that obtaining an approximation factor better than $\frac{e}{e-1} \simeq 1.58$ in one pass requires $n^{1+\Omega(1/\log\log n)}$ space [17, 20], even in bipartite graphs and in the *vertex-arrival* model, where the vertices arrive in the stream together with their incident edges. This setting has also been studied in the context of *online algorithms*, where each arriving vertex has to be either matched or discarded irrevocably upon arrival. Seminal work due to Karp, Vazirani and Vazirani [22] gives an online algorithm with $\frac{e}{e-1}$ approximation factor in this model.

Closing the gap between the upper bound of 2 and the lower bound of $\frac{e}{e-1}$ remains one of the most appealing open problems in the graph streaming area (see [39]). The factor of 2 can be improved on if one either considers the random-order model or allows for two passes [23]. By allowing even more passes, the approximation factor can be improved to multiplicative $(1-\epsilon)$-approximation via finding and applying augmenting paths with successive passes [28, 29, 13, 1].

Another line of research [16, 28, 43, 14, 11] has explored the question of approximating the maximum-weight matching in one pass and $O(n\,\mathrm{polylog}(n))$ space. The latest result is that a $(2+\varepsilon)$ approximation factor is possible using an $O(n\log n)$ space deterministic algorithm, essentially meeting the unweighted matching case [40]. These results are for the insert-only case. Where deletions are allowed (the dynamic, or turnstile case), the problem is harder: $\Omega(n^{2-3\epsilon})$ space is needed to provide an $O(n^{\epsilon})$ approximation [5]; and $\Omega(n/\alpha^2)$ to provide an $O(\alpha)$ approximation [4]. However, our focus is on finding the size of the maximum matching without materializing it, and so our aim is for sublinear space algorithms.

2 Preliminaries and Notations

Let $G(V,E)$ be an undirected unweighted graph with $n=|V|$ vertices and $m=|E|$ edges. For a vertex $v\in V$, let $\deg_G(v)$ denote the degree of vertex v in G. A *matching* M of G is a set of pairwise non-adjacent edges, i.e., no two edges share a common vertex. Edges in M are called *matched* edges; the other edges are called *unmatched*. A *maximum matching* of graph $G(V,E)$ is a matching of maximum size. Throughout the paper, when we fix a maximum matching of $G(V,E)$, we denote it by M^*. A matching M of G is *maximal* if it is not a proper subset of any other matching in graph G. Abusing the notation, we sometimes use M^* and M for the size of the maximum and maximal matching, respectively. It is well-known (see for example [26]) that the size of a maximal matching is at least half of the size of a maximum matching, i.e., $M\geq M^*/2$. Thus, we say a maximal matching is a 2-approximation of a maximum matching of G. It is known [26] that the simple greedy algorithm, where we include each new edge if neither of its endpoints are already matched, returns a maximal matching.

3 Algorithms for Bounded Arboricity Graphs

Throughout this section, h_μ denotes the number of vertices in graph $G=(V,E)$ that have degree above μ. Let $G_\mu=(L,F)$ be the induced subgraph of G where $L=\{v|\deg_G(v)\leq\mu\}$ and $(u,v)\in F\subseteq E$ when u and v are both in L. Note that G_μ might have isolated vertices. In the following we let M_μ denote the size of maximum matching in G_μ.

3.1 Characterization lemmas

▶ **Lemma 3** ([15]). *For a α-bounded arboricity graph $G(V,E)$ and $\mu>2\alpha$, we have $h_\mu\leq\frac{2\mu}{\mu-2\alpha+1}M^*$.*

▶ **Lemma 4.** *For a α-bounded arboricity graph $G(V,E)$ and $\mu>2\alpha$, we have*

$$M^*\leq h_\mu+M_\mu\leq\left(\frac{2\mu}{\mu-2\alpha+1}+1\right)M^*\ .$$

Proof. The lower bound is easy to see: every edge of a maximum matching either has an endpoint with degree more than μ or both of its endpoints are vertices with degree at most μ. The number of matched edges of the first type are bounded by h_μ whereas the number of matched edges of the second type are bounded by M_μ.

To prove the upper bound, we use the fact $M_\mu\leq M^*$ and Lemma 3. ◀

▶ **Definition 5.** Let $S = (e_1, \ldots, e_m)$ be a sequence of edges. We say the edge $e_i = (u, v)$ is γ-*good* with respect to S if $\max\{d_i(u), d_i(v)\} \leq \gamma$ where $d_i(x)$ is defined as $|\{e_j | j > i, e_j = (x, w)\}|$, i.e. the number of edges incident on x that appear after the i-th edge in the stream. We write $E_\gamma(S)$ as the set of γ-good edges in S, and usually drop (S) in context.

To illustrate the power of this definition, we first consider the case of trees. Trees are a good test case for understanding matchings, since they can have widely varying matching sizes: from 1 (a star graph on n nodes) to $O(n)$ (a path of length n or a binary tree on n nodes). In fact the following lemma suggests that counting the number of 1-good edges gives a 2 factor approximation to the matching size on trees. (Due to the space limitations we have deferred the proof of this lemma to the full version of this paper.)

▶ **Lemma 6.** *For trees we have $M^* \leq |E_1| \leq 2M^*$.*

Our main result on γ-goodness is for general graphs with α-bounded arboricity.

▶ **Lemma 7.** *Let $\mu > 2\alpha$ be a (large enough) integer, and let E_γ be the set of γ-good edges in an edge stream for a graph with arboricity at most α. We have:*

$$\left(\frac{1}{2} - \frac{\alpha}{\mu+1}\right) M^* \;\leq\; |E_\gamma| \;\leq\; \left(\frac{5}{4}\gamma + 2\right) M^*,$$

where $\gamma = \max\{\mu - 1, \frac{4\alpha(\mu+1)}{\mu+1-2\alpha}\}$. In particular for $\mu = 6\alpha - 1$, we have

$$M^* \;\leq\; 3|E_{6\alpha}| \;\leq\; (22.5\alpha + 6)M^*$$

Proof. First we prove the lower bound on $|E_\gamma|$. In particular we show a relation involving the number of edges where both endpoints have low degree. Define $h_\mu = |\{v | v \in V, \deg_G(v) > \mu\}|$, and $s_\mu = |\{e = (u, v) | e \in E, \deg_G(u) \leq \mu, \deg_G(v) \leq \mu\}|$, i.e. the number of edges in the graph G_μ. Then:

$$\left(\frac{1}{2} - \frac{\alpha}{\mu+1}\right) h_\mu + s_\mu \;\leq\; |E_\gamma|.$$

The claim in the lemma follows from the relatively loose bound that $M^* \leq h_\mu + s_\mu$. Let H be the set of vertices in the graph with degree above μ and let $L = V \setminus H$. Recall that $h_\mu = |H|$. Let H_0 be the vertices in H that have no neighbor in L, and let $H_1 = H \setminus H_0$. First we notice that $|H_1| \geq (1 - \frac{2\alpha}{\mu})|H|$. To see this, let E' be the edges with at least one endpoint in H_0. By definition, every node in H_0 has degree at least $\mu + 1$, so we have $|E'| \geq \frac{\mu+1}{2}|H_0|$. At the same time, the total number of edges in the subgraph induced by the nodes H is at most $\alpha(|H| - 1)$, using the arboricity assumption. Therefore,

$$\alpha(|H| - 1) \geq |E'| \geq \tfrac{1}{2}(\mu + 1)|H_0|$$

It follows that $|H_0| \leq \frac{2\alpha}{\mu+1}(|H| - 1)$ which further implies that

$$|H_1| \geq (1 - \frac{2\alpha}{\mu+1})|H| = (1 - \frac{2\alpha}{\mu+1})h_\mu. \qquad (1)$$

Now let $\deg_H(v)$ be the degree of v in the subgraph induced by H. We have $\sum_{v \in H_1} \deg_H(v) \leq 2\alpha|H|$, again using the arboricity bound and the fact that summing over degrees counts each edge at most twice. Therefore, taking the average over nodes in H_1,

$$\overline{\deg_H}(v) \leq \frac{2\alpha}{1 - \frac{2\alpha}{\mu+1}}$$

```
Algorithm 1: Estimate-$M_\mu + h_\mu$
  Initialization: Each node is sampled to set $T$ with probability $p$ (determined below).
  Stream Processing:
  forall edges $e = (u, v)$ in the stream do
      if $u \in T$ or $v \in T$ then
          store $e$ in $H$;
          if $u \in T$ then increment $d(u)$ else increment $l(u)$;
          if $v \in T$ then increment $d(v)$ else increment $l(v)$;
  Post Processing:
  Let $T_1 = \{v \in T | d(v) \le \mu, \exists w \in \Gamma(v) : d(w) + l(w) \le \mu\}$
  Let $T_2 = \{v \in T | d(v) > \mu\}$
  return $s = (|T_1| + |T_2|)/p$
```

for $v \in H_1$. Consequently, at least half of the vertices in H_1 have their $\deg_H$ bounded by $\frac{4\alpha(\mu+1)}{\mu+1-2\alpha}$ (via the Markov inequality). Let H'_1 be those vertices. For each $v \in H'_1$ we find a γ-good edge. Let $e^* = (v, u)$ be the last edge in the stream where $u \in L$. Then, there cannot be too many edges that neighbor (v, u) and come after it in the stream: the total number of edges that share an endpoint with e^* in the rest of the stream is bounded by $\max\{\mu - 1, \frac{4\alpha(\mu+1)}{\mu+1-2\alpha}\}$. Consequently, for

$$\gamma = \max\{\mu - 1, \frac{4\alpha(\mu + 1)}{\mu + 1 - 2\alpha}\},$$

we have $|E_\gamma| \ge (\frac{1}{2} - \frac{\alpha}{\mu+1})h_\mu$, based on the set of $|H_1|/2$ edges connected to the vertices in H'_1 and using (1). For $\gamma \ge \mu$, E_γ also contains the disjoint set of edges from $L \times L$, which are all guaranteed to be γ-good since both their endpoints have degree bounded by μ. Therefore, as claimed,

$$|E_\gamma| \ge s_\mu + \left(\frac{1}{2} - \frac{\alpha}{\mu + 1}\right) h_\mu.$$

To prove the upper bound on $|E_\gamma|$, we notice that the subgraph containing only the edges in E_γ has degree at most $\gamma + 1$. Such a graph has a matching size of at least $\frac{4|E_\gamma|}{5(\gamma+1)+3}$ [18]. It follows that $|E_\gamma| \le \frac{5\gamma+8}{4} M^*$. This finishes the proof of the lemma. ◀

3.2 $\tilde{O}(\sqrt{n})$ space algorithm for insert-only streams

In this section, we present Algorithm 1 to estimate $M_\mu + h_\mu$ and prove the following theorem.

▶ **Theorem 8.** *Let $G(V, E)$ be a graph with the arboricity bounded by α. Let S be an (adversarial order) insertion-only stream of the edges of the underlying graph G. Let*

$$\beta = \mu((2\mu)/(\mu - 2\alpha + 1) + 1) \text{ where } \mu > 2\alpha.$$

Then, there exists a streaming algorithm (Algorithm 1) that processes S, takes $O(\frac{\beta\sqrt{\alpha n}}{\varepsilon}\log n)$ space in expectation and outputs a $(1 + \varepsilon)\beta$ approximation of M^* with probability at least 0.86, where M^* is a maximum matching of G.

For each w in $\Gamma(T)$ (the set of neighbors of nodes in T), the algorithm maintains $l(w)$, the number of occurrences of w observed since the first time a neighbor of w was added to T.

```
Algorithm 2: Estimate-$M^*$
  Initialization: Let $\varepsilon \in (0,1)$ and $t = \lceil \frac{\beta\sqrt{8nc}}{\varepsilon} \rceil$ where $\beta$ is as defined in Lemma 9.
  Stream Processing: Do the following tasks in parallel:
  (1) Greedily keep a maximal matching of size at most $r \leq t$ (and terminate this task if this
      size bound is exceeded).
  (2) Run the Estimate-$(M_\mu + h_\mu)$ procedure (Algorithm 1) with $p \geq \frac{8}{\lambda^2 t}$ where $\lambda = \frac{\varepsilon}{\beta}$.
  Post processing: If $r < t$ then output $2r$ as the estimate for $M^*$, otherwise output
  the result of the Estimate-$(M_\mu + h_\mu)$ procedure.
```

Note that in this algorithm, $l(w)$ is a lower bound on the degree of w. For the output, T_1 is the subset of nodes in T whose degree is bounded by μ and additionally for each node in T_1, there is a neighbor w whose observed degree ($d(w)$ or $l(w)$) is at most μ. Meanwhile, T_2 is the set of "high degree" nodes in T.

▶ **Lemma 9.** *Let $\varepsilon \in (0,1)$ and $\beta = \mu(\frac{2\mu}{\mu-2\alpha+1} + 1)$. With probability at least $1 - e^{\frac{-\varepsilon^2 M^* p}{4\beta^2}}$, Algorithm 1 outputs s where*

$$(1-\varepsilon)M^* \leq s \leq (1+\varepsilon)\beta M^*.$$

Proof. First we prove the following bounds on $\mathsf{E}(s)$.

$$M_\mu + h_\mu \leq \mathsf{E}(s) \leq \mu(M_\mu + h_\mu).$$

Let L be the set of vertices in G that have degree at most μ and let G_L be the induced graph on L. Let $H = V \setminus L$. Note that G_L might have isolated vertices. Let N be the non-isolated vertices in G_L. It is clear that if the algorithm samples $v \in N$, v will be in T_1. Likewise, if it samples a vertex $w \in H$, w will be in T_2. Given the fact that $|H| = h_\mu$ and $|N| \geq M_\mu$, this proves the lower bound on $\mathsf{E}(s)$.

The expectation may be above M_μ, as the algorithm may pick an isolated vertex in G_L (a vertex that is *only* connected to the high-degree vertices) and include it in T_1 because one of its high-degree neighbours w was identified as low degree, i.e., $w \in \Gamma(T)$ and $l(w) \leq \mu$ but $w \in H$. Let $u \in H$ and let $U = \{a_1, \ldots, a_\mu\}$ be the last μ neighbours of u according to the ordering of the edges in the stream. The algorithm can only identify u as low degree when it picks a sample from U and no samples from $\Gamma(u) \setminus U$. This restricts the number of *unwanted* isolated vertices to at most μh_μ. Together with the fact that $|N| \leq \mu M_\mu$, it establishes the upper bound on $\mathsf{E}(s)$. Now using a Chernoff bound,

$$\begin{aligned}\Pr\left[|s - \mathsf{E}(s)| \geq \lambda \mathsf{E}(s)\right] &= \Pr\left[|s.p - \mathsf{E}(s.p)| \geq \lambda \mathsf{E}(s.p)\right] \\ &\leq \exp(-\lambda^2(M_\mu + h_\mu)p/4) \leq \exp(-\lambda^2 M^* p/4).\end{aligned}$$

Therefore with probability at least $1 - e^{\frac{-\lambda^2 M^* p}{4}}$,

$$(M_\mu + h_\mu) - \lambda\mu(h_\mu + M_\mu) \leq s \leq \mu(1+\lambda)(M_\mu + h_\mu) \qquad (2)$$

Setting $\lambda = \frac{\varepsilon}{\beta}$ and combining with Lemma 4, we derive the statement of the lemma. ◀

Proof of Theorem 8: Suppose $M^* < t$. Clearly the size of the maximal matching r obtained by the first task will be less than t. In this case, $M^* \leq 2r \leq 2M^*$. Now suppose

$M^* \geq t$. By Lemma 4, we will have $M_\mu + h_\mu \geq t$ and hence by Lemma 9, with probability at least $1 - e^{-2} \geq 0.86$, the output of the algorithm will be within the promised bounds. The expected space of the algorithm is $O((t + pn\alpha)\log n)$. Setting $t = \beta\sqrt{8n\alpha}/\varepsilon$ to balance the space costs, the space complexity of the algorithm will be $O(\frac{\beta\sqrt{\alpha n}}{\varepsilon}\log n)$ as claimed. ◻

3.3 $O(n^{2/3})$ space algorithm for insertion/deletion streams

Algorithms 1 and 2 form the basis of our solution in the more general case where the stream contains deletions of edges as well. In the case of Algorithm 1, the algorithm has to maintain the induced subgraph on T and the edges of the cut $(T, \Gamma(T))$. However if we allow an arbitrary number of insertions and deletions, the size of the cut $(T, \Gamma(T))$ can grow as large as $O(n)$ even when $|T| = 1$. This is because each node at some intermediate point could become high degree and then lose its neighbours because of the subsequent deletion of edges. Therefore here in order to limit the space usage of the algorithm, we make the assumptions that number of deletions is bounded by $O(\alpha n)$. Since the processed graph has arboricity at most α this forces the number of insertions to be $O(\alpha n)$ as well. Under this assumption, if we pick a random vertex, still, in expectation the number of neighbours is bounded by $O(\alpha)$.

Another complication arises from the fact that, with edge deletions, a vertex added to $\Gamma(T)$ might become isolated at some point. In this case, we discard it from $\Gamma(T)$. Additionally for each vertex in $T \cup \Gamma(T)$, the counters $d(v)$ (or $l(v)$ depending on if it belongs to T or $\Gamma(T)$) can be maintained as before. The space complexity of the algorithm remains $O(pn\alpha \log n)$ in expectation as long as the arboricity factor remains within $O(\alpha)$ in the intermediate graphs. In the case of Algorithm 2, we need to keep a maximal matching of size $O(t)$. This can be done in $O(t^2)$ space using a randomized algorithm [7]. Setting t at $(\frac{8\beta n\alpha}{\varepsilon^2})^{1/3}$ to rebalance the space costs, we obtain the result of Theorem 2.

3.4 The polylog space algorithm for insert-only streams

In this section we present our polylog space algorithm by presenting an algorithm for estimating $|E_\gamma|$ within a $(1+\varepsilon)$ factor. Our algorithm is similar in spirit to the well-known L_0 sampling strategy [9]. We first describe it in terms of running $O(\log n)$ parallel threads each sampling the stream at a different rate. At the end, a thread "wins" that has sampled roughly $\Theta(\log n)$ elements from $|E_\gamma|$ (samples the edges with a rate of $\frac{\log n}{|E_\gamma|}$). The threads that under-sample will end up with few edges or nothing while the ones that have oversampled will keep too many elements of E_γ and will be aborted as result. Finally, we mention how a suitable implementation can reduce the space dependency to $O(\alpha \log n)$ (treating ϵ as constant).

First we give a simple subroutine (Algorithm 3) that is the building block of the algorithm. Given γ and an edge e in the stream, it simply counts up the number of subsequent edges that are incident on either endpoint of e, and consequently determines whether e is in E_γ. Our main algorithm (Algorithm 4) samples edges, and applies this subroutine to them. Multiple sampling rates p_i are used in parallel; however, if at any point the number of sampled edges in a level exceeds a threshold τ, the level is "terminated", and no further samples are taken at this level. This ensures that the space used remains bounded.

▶ **Lemma 10.** *With high probability, Algorithm 4 outputs a $1 \pm O(\varepsilon)$ approximation of $|E_\gamma|$ where γ is defined according to Lemma 7.*

Proof. Consider the sets of active γ-good tests at each level at the conclusion of the algorithm, X_i. First we observe that if $|X_0| \leq \tau$ then $X_0 = E_\gamma$ and the algorithm makes no error.

```
Algorithm 3: The $\gamma$-good test
  Initialization: given the edge $e = (u, v)$ in the stream, let $r(u) = 0$ and $r(v) = 0$.
  forall subsequent edges $e' = (t, w)$ do
      if $u \in \{t, w\}$ then increment $r(u)$;
      if $v \in \{t, w\}$ then increment $r(v)$;
      if $\max\{r(u), r(v)\} > \gamma$ then terminate and report NOT $\gamma$-good;
```

```
Algorithm 4: An algorithm for approximating $|E_\gamma|$
  Initialization:  $\forall i. X_i = \emptyset$   ▷ $X_i$ represents the current set of sampled $\gamma$-good edges.

  Stream Processing:
  forall levels $i \in \{0, 1, \ldots, [\lfloor \log_{1+\varepsilon} n^2 \rfloor\}$ in parallel do
      forall edges $e$ do
          Feed $e$ to the active $\gamma$-good tests and update $X_i$
          With probability $p_i = \frac{1}{(1+\varepsilon)^i}$ add $e$ to $X_i$ and start a $\gamma$-good test for $e$.
          Let $|X_i|$ be the number of active $\gamma$-good tests within this level.
          if $|X_i| > \tau = \frac{64\gamma^2 \log n}{\alpha\varepsilon^2}$ then  terminate level $i$;

  Post processing:
  if $|X_0| \leq \tau$ then
      return $|X_0|$
  else      ▷ $|X_0| > \tau$
      let $j$ be smallest integer s.t. $|X_j| \leq \frac{8 \log n(1+\varepsilon)}{\varepsilon^2}$ and $j$-th level was not terminated;
      if there is no such $j$ then return FAIL else return $\frac{|X_j|}{p_j}$;
```

In case $|X_i| > \tau$, we claim that $|E_\gamma| > \frac{\alpha}{2\gamma^2}\tau$. To prove this, let t be the time step where $|X_i|$ exceeds τ (i.e. when this level is terminated) and let $G_t = (V, E^{(t)})$ be the graph where $E^{(t)} = \{e_1, \ldots, e_t\}$. Clearly $M^*(G) \geq M^*(G_t)$ because the size of the matching only increases as new edges arrive. Abusing the notation, let $E_\gamma(G_t)$ denote the set of γ-good edges at time t. By Lemma 7 and definition of γ, we have

$$\tau < |E_\gamma(G_t)| \leq \left(\frac{5}{4}\gamma + 2\right) M^*(G_t) \leq 4\gamma M^*(G) \leq \frac{2(\mu+1)}{\mu - 2\alpha + 1} 4\gamma |E_\gamma| \leq \left(\frac{\gamma}{2\alpha}\right) 4\gamma |E_\gamma| \tag{3}$$

This proves the claim that $|E_\gamma| > \frac{\alpha}{2\gamma^2}\tau$ when $|X_i| > \tau$. Let $\tau' = \frac{8 \log n}{\varepsilon^2}$ and let i^* be the integer such that

$$(1+\varepsilon)^{i^*-1}\tau' \leq |E_\gamma| \leq (1+\varepsilon)^{i^*}\tau'.$$

Assuming the i^*-th level does not terminate before the end, we have $\frac{\tau'}{(1+\varepsilon)} \leq \mathsf{E}[|X_{i^*}|] \leq \tau'$. By a Chernoff bound, for each i we have (again assuming we do not terminate the corresponding level)

$$\Pr\left[\big||X_i| - \mathsf{E}(|X_i|)\big| \geq \varepsilon \mathsf{E}(|X_i|)\right] \leq \exp\left(-\frac{\varepsilon^2 p_i |E_\gamma|}{4}\right).$$

$$\text{So, } \Pr\left[\big||X_{i^*}| - \mathsf{E}(|X_{i^*}|)\big| \geq \varepsilon \mathsf{E}(|X_{i^*}|)\right] \leq \exp\left(-\frac{\varepsilon^2 |E_\gamma|}{2(1+\varepsilon)^{i^*}}\right) \leq \exp\left(\frac{2\log n}{1+\varepsilon}\right) \leq O(n^{-1}).$$

As a result, with high probability $|X_{i^*}| \leq \frac{8\log n(1+\varepsilon)}{\varepsilon^2}$. Moreover for all $i < i^* - 1$, the corresponding levels either terminate prematurely or in the end we will have $|X_i| > \frac{8\log n(1+\varepsilon)}{\varepsilon^2}$ with high probability. Consequently $j \in \{i^*, i^* - 1\}$. It remains to prove that runs corresponding to i^* and $i^* - 1$ will survive until the end with high probability. We prove this for i^*. The case of $i^* - 1$ is similar.

Consider a fixed time t in the stream and let $X_{i^*}^{(t)}$ be the set of sampled γ-good edges at time t corresponding to the i^*-th level. Note that $X_{i^*}^{(t)}$ contains the a subset of γ-good edges with respect to the stream $S_t = (e_1, \dots, e_t)$. From the definition of i^* and Inequality (3) we have

$$\mathsf{E}[|X_{i^*}^{(t)}|] = \frac{|E_\gamma(G_t)|}{(1+\varepsilon)^{i^*}} \leq \frac{2\gamma^2|E_\gamma|}{\alpha(1+\varepsilon)^{i^*}} \leq \frac{2\gamma^2\tau'}{\alpha}.$$

By the Chernoff inequality for $\delta \geq 1$,

$$\Pr\left[|X_{i^*}^{(t)}| \geq (1+\delta)E(|X_{i^*}^{(t)}|)\right] \leq \exp\left(\frac{-\delta}{3}E(|X_{i^*}^{(t)}|)\right).$$

From $\delta = \frac{\tau}{E(|X_{i^*}^{(t)}|)} - 1 = \frac{\tau(1+\varepsilon)^{i^*}}{|E_\gamma(G_t)|} - 1$, we get

$$\Pr\left[|X_{i^*}^{(t)}| \geq \tau\right] \leq \exp\left(\frac{-\tau}{3} + \frac{|E_\gamma(G_t)|}{(1+\varepsilon)^{i^*}}\right) \leq \exp\left(\frac{-\tau}{3} + \frac{2\gamma^2\tau'}{\alpha}\right)$$

For $\tau \geq \frac{8\gamma^2\tau'}{\alpha}$, the term inside the exponent is smaller than $-2\log n$. It also satisfies $\delta \geq 1$. After applying the union bound, for all t the size of $X_{i^*}^{(t)}$ is bounded by $\tau = \frac{64\gamma^2\log n}{\alpha\varepsilon^2}$ with high probability. This finishes the proof of the lemma. ◀

Next, putting everything together, we prove Theorem 1.

Proof of Theorem 1: The theorem follows from Lemmas 7 and 10 and taking $\gamma = \mu + 1 = 6\alpha$. Observe that the space cost of Algorithm 4 can be bounded: we have $\log_{1+\varepsilon} n^2$ levels where each level runs at most τ concurrent γ-good tests otherwise it will be terminated. Each γ-good test keeps an edge and two counters and as result it occupies $O(1)$ space. Consequently the space usage of the algorithm is bounded by $O(\tau \log_{1+\varepsilon} n)$. Using the fact that $\tau = O(\frac{\alpha}{\varepsilon^2}\log n)$ for $\gamma = 6\alpha$, we obtain a space bound of $O(\frac{\alpha}{\varepsilon^2}\log^2 n)$.

A simple implementation optimization is not to run multiple guesses of p in parallel, but instead to begin with $i = 1$ and $p_1 = 1$. Whenever $|X_i| > \tau$, then we increment i and uniformly sample elements from X_i into X_{i+1} with probability $\frac{1}{1+\epsilon}$. It is immediate that the resulting X_{i+1} corresponds to a sample of the γ-good edges in the stream so far with a sampling probability of $p_i = \frac{1}{(1+\epsilon)^i}$, by the principle of deferred decisions. Consequently, the space bound is reduced to $O(\frac{\alpha}{\epsilon^2}\log n)$. □

Acknowledgements

We thank Andrew McGregor and Jelani Nelson for some helpful conversations. We also thank the anonymous reviewers for their careful reading of the paper and helpful comments.

References

1 K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–467, 2012.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *ACM Principles of Database Systems*, pages 5–14, 2012. URL: `http://doi.acm.org/10.1145/2213556.2213560`, `doi:10.1145/2213556.2213560`.

3 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eigth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, January 2017*, 2017. URL: `http://www.seas.upenn.edu/~sassadi/pages/streaming_matching-size_2017.html`.

4 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742, 2017. `doi:10.1137/1.9781611974782.113`.

5 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1345–1364, 2016. `doi:10.1137/1.9781611974331.ch93`.

6 M. Bury and C. Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 263–274, 2015.

7 R. Chitnis, G. Cormode, H. Esfandiari, M.T. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1344, 2016.

8 Edith Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. In *ACM Principles of Database Systems*, pages 88–99, 2014. URL: `http://doi.acm.org/10.1145/2594538.2594546`, `doi:10.1145/2594538.2594546`.

9 Graham Cormode and Donatella Firmani. On unifying the space of ℓ_0-sampling algorithms. In *Algorithm Engineering and Experiments*, 2013.

10 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. Technical Report 1608.03118, ArXiv, 2016. URL: `http://arxiv.org/abs/1608.03118`.

11 M. Crouch and D. S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Proceedings of the 17th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 96–104, 2014.

12 R. Duan and S. Pettie. Linear-time approximation for maximum weight matchings. *Journal of the ACM*, 61(1):1–23, 2014.

13 S. Eggert, L. Kliemann, P. Munstermann, and A. Srivastav. Bipartite graph matchings in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.

14 Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.

15 H. Esfandiari, M.T. Hajiaghyi, V. Liaghat, M. Monemizadeh, and K. Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015.

16 J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207–216, 2005.

17 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–485, 2012.

18 Yijie Han. Matching for graphs of bounded degree. In *Frontiers in Algorithmics, Second Annual International Workshop, FAW 2008, Changsha, China, June 19-21, 2008, Proceeed-*

ings, pages 171–173, 2008. URL: `http://dx.doi.org/10.1007/978-3-540-69311-6_19`, `doi:10.1007/978-3-540-69311-6_19`.

19 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. URL: `http://dx.doi.org/10.1137/0202019`, `doi:10.1137/0202019`.

20 M. Kapralov. Better bounds for matchings in the streaming model. *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1679–1697, 2013.

21 M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014.

22 R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.

23 C. Konrad, F. Magniez, and C. Mathieu. Maximum matching in semi-streaming with few passes. In *Proceedings of the 11th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 231–242, 2012.

24 Alexandr V. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, 4(4):307–316, 1984.

25 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: A method for solving graph problems in mapreduce. In *Proceedings of the Twenty-third Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 85–94. ACM, 2011. URL: `http://doi.acm.org/10.1145/1989493.1989505`, `doi:10.1145/1989493.1989505`.

26 L. Lovasz and M.D. Plummer. Matching theory. In *North-Holland, Amsterdam-New York*, 1986.

27 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 253–262, 2013. URL: `http://dx.doi.org/10.1109/FOCS.2013.35`, `doi:10.1109/FOCS.2013.35`.

28 A. McGregor. Finding graph matchings in data streams. In *Proceedings of the 8th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 170–181, 2005.

29 A. McGregor. Graph mining on streams. In *Encyclopedia of Database Systems*, pages 1271–1275. Springer, 2009.

30 A. McGregor and S. Vorotnikova. Planar matching in streams revisited. In *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2016.

31 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014. URL: `http://doi.acm.org/10.1145/2627692.2627694`, `doi:10.1145/2627692.2627694`.

32 Andrew McGregor and Sofya Vorotnikova. A note on logarithmic space stream algorithms for matchings in low arboricity graphs. Technical Report 1612.02531, ArXiv, 2016.

33 Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better algorithms for counting triangles in data streams. In *ACM Principles of Database Systems*, pages 401–411, 2016. URL: `http://doi.acm.org/10.1145/2902251.2902283`, `doi:10.1145/2902251.2902283`.

34 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.

35 S. Micali and V. V. Vazirani. An $o(\sqrt{|V|}|e|)$ algorithm for finding maximum matching in general graphs. *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

36 C. St. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 36(1):445–450, 1961.

37 C. St. J. A. Nash-Williams. Decomposition of finite graphs into forests. *Journal of the London Mathematical Society*, 39(1):12, 1964.

38 Huy Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *IEEE Conference on Foundations of Computer Science*, 2008.

39 List of open problems in sublinear algorithms: Problem 60. `http://sublinear.info/60`.

40 Ami Paz and Gregory Schwartzman. A $(2 + \epsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2153–2161, 2017. `doi:10.1137/1.9781611974782.140`.

41 A. E. Roth and M. A. O. Sotomayor. *Two-sided matching: A study in game-theoretic modeling and analysis*. Cambridge University Press, 1990.

42 Nenad Trinajstic, Douglas J. Klein, and Milan Randic. On some solved and unsolved problems of chemical graph theory. *International Journal of Quantum Chemistry*, 30(S20):699–742, 1986.

43 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.