Improved Bounds for Matching in Random-Order Streams

Aaron Bernstein
Rutgers University, Department of Computer Science, New Brunswick, NJ, USA
https://aaronbernstein.cs.rutgers.edu
bernstei@gmail.com

Abstract

We study the problem of computing an approximate maximum cardinality matching in the semi-streaming model when edges arrive in a *random* order. In the semi-streaming model, the edges of the input graph $G = (V, E)$ are given as a stream $e_1, \ldots, e_m$, and the algorithm is allowed to make a single pass over this stream while using $O(n\text{polylog}(n))$ space ($m = |E|$ and $n = |V|$). If the order of edges is adversarial, a simple single-pass greedy algorithm yields a 1/2-approximation in $O(n)$ space; achieving a better approximation in adversarial streams remains an elusive open question.

A line of recent work shows that one can improve upon the 1/2-approximation if the edges of the stream arrive in a random order. The state of the art for this model is two-fold: Assadi et al. [SODA 2019] show how to compute a $\frac{2}{3}(\sim .66)$-approximate matching, but the space requirement is $O(n^{1.5}\text{polylog}(n))$. Very recently, Farhadi et al. [SODA 2020] presented an algorithm with the desired space usage of $O(n\text{polylog}(n))$, but a worse approximation ratio of $\frac{6}{11}(\sim .545)$, or $\frac{3}{5}(= .6)$ in bipartite graphs.

In this paper, we present an algorithm that computes a $\frac{2}{3}(\sim .66)$-approximate matching using only $O(n \log(n))$ space, improving upon both results above. We also note that for adversarial streams, a lower bound of Kapralov [SODA 2013] shows that any algorithm that achieves a $1 - \frac{1}{e}(\sim .63)$-approximation requires $(n^{1+\Omega(1/\log\log(n))})$ space. Our result for random-order streams is the first to go beyond the adversarial-order lower bound, thus establishing that computing a maximum matching is provably easier in random-order streams.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Graph Algorithms, Sublinear Algorithms, Matching, Streaming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.12

Category Track A: Algorithms, Complexity and Games

Funding *Aaron Bernstein*: NSF CAREER Grant 1942010 and Simons Collaboration on Algorithms and Geometry.

Acknowledgements I want to thank Sepehr Assadi for several very helpful discussions.

1 Introduction

Computing a maximum cardinality matching is a classical problem in combinatorial optimization, with a large number of algorithms and applications. Motivated by the rise of massive graphs, much of the recent research on this problem has focused on sub-linear algorithms that are able to compute a matching without storing the entire graph in memory. One of the standard sub-linear models for processing graphs is known as the *semi-streaming* model [17]: the algorithm has access to a sequence of edges (the stream), and is allowed to make a single pass over this sequence while only using only $O(n\text{polylog}(n))$ internal memory, where n is the number of vertices in the graph. Note that the memory used is still significantly smaller than the number of edges in the graph, and that $O(n)$ memory is also necessary if we want the algorithm to output the actual edges of the matching. (One typically assume

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 12; pp. 12:1–12:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


Table 1 Single-pass semi-streaming algorithms known for the maximum matching when edges arrive in a random order. The space bounds are expressed in terms of $O(\log(n))$-size words, though many existing results do not state the exact polylog(n) term. The result of Gamlath et al. [18] works in weighted graphs; all others are restricted to unweighted graphs.

	Approximation Factor		
	Bipartite graphs	**General graphs**	**Space**
Konrad et al. [28]	0.5005	0.5003	$O(n)$
Gamlath et al. [18]	0.512	0.506	$O(n \cdot \text{polylog}(n))$
Konrad [27]	0.539	-	$O(n \cdot \text{polylog}(n))$
Assadi et al. [3]	0.666	0.666	$O(n^{1.5} \cdot \text{polylog}(n))$
Farhadi et al. [16]	0.6	0.545	$O(n \cdot \text{polylog}(n))$
This paper	**0.666**	**0.666**	$O(n \log(n))$

$O(\log(n))$-size words, so that a single edge can be stored in $O(1)$ space; if one were to express space in terms of the number of bits, all the space bounds in this paper would increase by a $O(\log(n))$ factor.)

If the edges of the stream arrive in an arbitrary order, a simple greedy algorithm can compute a maximal matching – and hence a 1/2-approximate maximum matching – in a single streaming pass and $O(n)$ space. Going beyond a 1/2-approximation with a single pass is considered one of the main open problems in the area. The strongest lower bound is by Kapralov [23], who build upon an earlier lower bound of Goel et al. [20]: any algorithm with approximation ratio $\geq 1 - 1/e \sim .63$ requires $n^{1+\Omega(1/\log\log(n))}$ space [23]. But we still do not know where the right answer lies between 1/2 and $1 - 1/e$.

To make progress on this intriguing problem, several recent papers studied a more relaxed model, where the graph is still arbitrary, but the edges are assumed to arrive in a *uniformly random* order. Konrad et al. were the first to go beyond a 1/2-approximation in this setting: they showed that in random-order streams, there exists an $O(n)$-space algorithm that computes an .5003-approximate matching, or .5005-approximate for bipartite graphs [28]. This was later improved to .506 in general graphs [18] and .539 in bipartite graphs [27]. Assadi et al. then showed an algorithm with an approximation ratio of $(2/3 - \epsilon) \sim .66$, but their algorithm had a significantly larger space requirement of $O(n^{1.5}\text{polylog}(n))$ [3]. Finally, very recently (SODA 2020), Farhadi et al. achieved the current state of the art for $O(n\text{polylog}(n))$ space; their algorithm achieves an approximation ratio of $6/11 \sim .545$ for general graphs and $3/5 = .6$ for bipartite graphs [16]. A summary of these results can be found in Table 1.

Although this line of work suggests that computing a maximum matching might be fundamentally easier in random-order streams, we note that even in bipartite graphs, none of the previous results go beyond the best known lower bound for adversarial streams mentioned above [23]: the algorithm of Assadi et al. uses too much space ($n^{1.5} \gg n^{1+1/\log\log(n)}$), while the result of Farhadi et al. has an approximation ratio of $.6 < 1 - 1/e$.

Our result is the first to go beyond the adversarial-order lower bound, thus establishing that computing a matching is provably easier in random-order streams.

▶ **Theorem 1** (Our Result). *Given any (possibly non-bipartite) graph G and any approximation parameter $1 > \epsilon > 0$, there exists a deterministic single-pass streaming algorithm that with high probability computes a $(2/3 - \epsilon)$-approximate matching if the edges of G arrive in a uniformly random order. The space usage of the algorithm is $O(n \log(n) poly(\epsilon^{-1}))$.*

Our result significantly improves upon the space requirement of Assadi et al. [3] and the approximation ratio of Farhadi et al. [16]. In fact, our algorithm achieves the best of both those results (see Table 1). On top of that, our result is quite simple; given that it improves upon a sequence of previous results, we see this simplicity as a plus.

Related Work

If the only requirement is to return an approximate estimate of the *size* of the maximum matching, rather than the actual edges, a surprising result by Kapralov et al. shows that one can get away with very little space: given a single pass over a random-order stream, it is possible to estimate the size within a $1/\text{polylog}(n)$ factor using only $\text{polylog}(n)$ space [24]; a very recent improvement reduces the polylog factors to $O(\log^2(n))$ [25]. There is also a line of work that estimates the size of the matching in $o(n)$ space in *adversarial* streams for special classes of graphs as such planar graphs or low-arboricity graphs [14, 8, 30, 11, 31].

There are many one-pass streaming algorithms for computing a maximum matching in *weighted* graphs. For adversarial-order streaming, a long line of work culminated in a $(1/2-\epsilon)$-approximation using $O(n)$ space [17, 29, 13, 12, 32, 19]. Gamlath et al. recently showed that for random-order streams, one can achieve an approximation ratio of $1/2 + \Omega(1)$, [18]. See also other related work on weighted graphs in [8].

There are several results on upper and lower bounds for computing a maximum matching in dynamic streams (where edges can also be deleted) [26, 10, 6, 9, 5]. Finally, there are several results that are able to achieve better bounds by allowing the algorithm to make multiple passes over the stream: some results focus on just two or three passes [28, 15, 22, 27], while others seek to compute a $(1-\epsilon)$-approximate matching by allowing a large constant number (or even $\log(n)$) passes [29, 1, 21, 2].

Overview of Techniques

The basic greedy algorithm trivially achieves a 1/2-approximate matching in adversarial streams; in fact, Konrad et al later showed that the ratio remains $1/2 + o(1)$ even in random-order streams [28]. Existing algorithms for improving the 1/2 ratio in random-order streams generally fall into two categories. The algorithms in [28, 27, 18, 16] use the randomness of the stream to compute some fraction of short augmenting paths, thus going beyond the 1/2-approximation of a maximal matching. The result in [3] instead shows that one can obtain a large matching by constructing a subgraph that obeys certain degree-properties.

Our result follows the framework of [3]. Given any graph G, an earlier result of Bernstein and Stein for fully dynamic matching defined the notion of an edge-degree constrainted subgraph (denoted EDCS), which is a sparse subgraph $H \subseteq G$ that obeys certain degree-properties [7]. They showed that any EDCS H always contains a $(2/3-\epsilon)$-approximate matching. The streaming result of Assadi et al. [3] then showed that given a random-order stream, it is possible to compute an EDCS H in $O(n^{1.5})$ space; returning the maximum matching in H yields a $(2/3-\epsilon)$-approximate matching in G.

Our result also takes the EDCS as its starting point, but it is unclear how to compute an EDCS H of G using less than $O(n^{1.5})$ space. Our algorithm requires two new contributions. Firstly, we show that it is sufficient for H to satisfy a somewhat relaxed set of properties. Our main contribution is then to use an entirely different construction of this relaxed subgraph, which uses the randomness of the stream more aggressively to compute H using low space.

2 Notation and Preliminaries

Consider any graph $H = (V_H, E_H)$. We define $\deg_H(v)$ to be the degree of v in H and we define the degree of an edge (u, v) to be $\deg_H(u) + \deg_H(v)$. A matching M in H is a set of vertex-disjoint edges. All graphs in this paper are unweighted and undirected. We use $\mu(H)$ to denote the size of the maximum matching in H. Unless otherwise indicated, we let $G = (V, E)$ refer to the input graph and let $n = |V|$ and $m = |E|$. We note that every graph referred to in the paper has the same vertex V as the input graph; when we refer to subgraphs, we are always referring to a subset of edges on this same vertex set.

The input graph $G = (V, E)$ is given as a stream of edges $S = \langle e_1, \ldots, e_m \rangle$. We assume that the permutation $(e_1, \ldots, e_m)$ of the edges is chosen *uniformly at random* among all permutations of E. We use $S_{[i,j]}$ to denote the substream $\langle e_i, \ldots, e_j \rangle$, and we use $G_{>i} \subseteq G$ to denote the subgraph of G containing all edges in $\{e_{i+1}, \ldots, e_m\}$.

Our analysis will apply concentration bounds to segments $S_{[i,j]}$ of the stream. Observe that because the stream is a random permutation, any segment $S_{[i,j]}$ is equivalent to sampling $j - 1 + 1$ edges from the stream without replacement. We can thus apply the Chernoff bound for negatively associated variables (see e.g. the primer in [33]).

▶ **Theorem 2** (Chernoff). *Let $X_1, \ldots X_n$ be negatively associated random variables taking values in $[0, 1]$. Let $X = \sum X_i$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 < \delta < 1$ we have*

$$\Pr[X \leq \mu(1 - \delta)] \leq \exp(\frac{-\mu \cdot \delta^2}{2}),$$

and

$$\Pr[X \geq \mu(1 + \delta)] \leq \exp(\frac{-\mu \cdot \delta^2}{3})$$

The early and late sections of the stream

Our algorithm will use the first ϵm edges of the stream to learn about the graph and will effectively ignore them for the purposes of analyzing the maximum matching. Thus, we only approximate the maximum matching in the later $(1 - \epsilon)m$ edges of stream; because the stream is random, these edges still contain a large fraction of the maximum matching. We use the following definitions and lemmas to formalize this intuition.

▶ **Definition 3.** *We Let E^{early} denote the first ϵm edges of the stream, and E^{late} denote the rest: that is, $E^{early} = \{e_1, \ldots, e_{\epsilon m}\}$, and $E^{late} = \{e_{\epsilon m+1}, \ldots, e_m\}$. Define $G^{early} = (V, E^{early})$ and $G^{late} = (V, E^{late}) = G_{>\epsilon m}$.*

For the probability bounds to work out, we need to assume that $\mu(G) \geq 20 \log(n)\epsilon^{-2}$. We justify this assumption by observing that every graph G satisfies $m \leq 2n\mu(G)$, so if $\mu(G) < 20 \log(n)\epsilon^{-2}$, then the algorithm can trivially return an exact maximum matching by simply storing every edge using only $O(m) = O(\log(n)\epsilon^{-2})$ space. This justifies the following:

▷ Claim 4 (Assumption). We can assume for the rest of the paper that $\mu(G) \geq 20 \log(n)\epsilon^{-2}$.

Combining Claim 4 with Chernoff bound we get the following lemma, which allows us to focus our analysis on the edges in G^{late}.

▶ **Lemma 5.** *Assuming that $\epsilon < 1/2$, we have that $\Pr[\mu(G^{late}) \geq (1 - 2\epsilon)\mu(G)] \geq 1 - n^{-5}$.*

Proof. Fix some maximum matching $M = (f_1, ..., f_{\mu(G)})$ of G. Define X_i to be the indicator variable that edge $f_i \in M$ appears in G^{late}. Since the stream is random, and since G^{late} contains exactly $(1-\epsilon)m$ edges, we have that $\mathbb{E}[X_i] = (1-\epsilon)$ and $\sum \mathbb{E}[X_i] = (1-\epsilon)\mu(G)$. It is also easy to see that the X_i are negatively associated, since these variables correspond to sampling $(1-\epsilon)m$ edges without replacement. Recall from Claim 4 that we assume $\mu(G) \geq 20\log(n)\epsilon^{-2}$. Applying the Chernoff Bound in Theorem 2 completes the proof. ◀

Existing Work on EDCS

We now review the basic facts about the edge-degree constrained subgraph (EDCS), which was first introduced in [7].

▶ **Definition 6.** *Let $G = (V, E)$ be a graph, and $H = (V, E_H)$ a subgraph of G. Given any parameters $\beta \geq 2$ and $\lambda < 1$, we say that H is a (β, λ)-EDCS of G if H satisfies the following properties:*

- [Property P1]. *For any edge $(u, v) \in H$, $\deg_H(u) + \deg_H(v) \leq \beta$*
- [Property P2]. *For any edge $(u, v) \in G \setminus H$, $\deg_H(u) + \deg_H(v) \geq \beta(1-\lambda)$.*

The crucial fact about the EDCS is that it always contains a (almost) 2/3-approximate matching. The simplest proof of Lemma 7 below is in Lemma 3.2 of [4].

▶ **Lemma 7** ([4]). *Let $G(V, E)$ be any graph and $\epsilon < 1/2$ be some parameter. Let λ, β be parameters with $\lambda \leq \frac{\epsilon}{64}$, $\beta \geq 8\lambda^{-2}\log(1/\lambda)$. Then, for any $(\beta, \lambda) - EDCS$ H of G, we have that $\mu(H) \geq (\frac{2}{3} - \epsilon)\mu(G)$. (Note that the final guarantee is stated slightly differently than in Lemma 3.2 of [4], and to ensure the two are equivalent, we set λ to be a factor of two smaller than in Lemma 3.2 of [4].)*

3 Our Modified Subgraph

Unlike the algorithm of [3], we do not actually construct an EDCS of G, as we do not know how to do this in less than $O(n^{1.5})$ space. We instead rely on a more relaxed set of properties, which we analyze using Lemma 7 as a black-box. We now introduce some of the basic new tools used by our algorithm. Note that graph G in the lemma and definitions below crucially refers to any arbitrary graph G, and not necessarily the main input graph of the streaming algorithm.

▶ **Definition 8.** *We say that a graph H has bounded edge-degree β if for every edge $(u, v) \in H$, $\deg_H(u) + \deg_H(v) \leq \beta$.*

▶ **Definition 9.** *Let G be any graph, and let H be a subgraph of G with bounded edge-degree β. For any parameter $\lambda < 1$, we say that an edge $(u, v) \in G \setminus H$ is (G, H, β, λ)-underfull if $\deg_H(u) + \deg_H(v) < \beta(1-\lambda)$*

The two definitions above effectively separate the two EDCS properties: any subgraph H of G with bounded edge-degree β automatically satisfies property P1 of an EDCS, and underfull edges are then those that violate property P2. We now show that one can always construct a large matching from the combination of these two parts.

▶ **Lemma 10.** *Let $\epsilon < 1/2$ be any parameter, and let λ, β be parameters with $\lambda \leq \frac{\epsilon}{128}$, $\beta \geq 16\lambda^{-2}\log(1/\lambda)$. Consider any graph G, and any subgraph H with bounded edge-degree β. Let X contain all edges in $G \setminus H$ that are (G, H, β, λ)-underfull. Then $\mu(X \cup H) \geq (2/3 - \epsilon)\mu(G)$*

Proof. Note that it is NOT necessarily the case that $H \cup X$ is an EDCS of G, because adding the edges of X to H will increase vertex and edge degrees in H, so $H \cup X$ might not satisfy property P1 of an EDCS. We thus need a more careful argument.

Let M_G be the maximum matching in G, let $M_G^H = M_G \cap H$ and let $M_G^{G\setminus H} = M_G \cap (G \setminus H)$. Let $X^M = X \cap M_G^{G\setminus H}$. Note that by construction, $M_G \subseteq H \cup M_G^{G\setminus H}$, so $\mu(H \cup M_G^{G\setminus H}) = \mu(G)$.

We now complete the proof by showing that $H \cup X^M$ is a $(\beta+2, 2\lambda)$-EDCS of $H \cup M_G^{G\setminus H}$. Let us start by showing property P2. Recall that X contains all edges (u,v) in $G \setminus H$ for which $\deg_H(u) + \deg_H(v) < \beta(1-\lambda)$, so by construction X^M contains all such edges in $M_G^{G\setminus H}$. Thus, every edge $(u,v) \in (H \cup M_G^{G\setminus H}) \setminus (H \cup X^M) = M_G^{G\setminus H} \setminus X^M$ must have $\deg_H(u) + \deg_H(v) \geq \beta(1-\lambda) \geq (\beta+2)(1-2\lambda)$, where the last inequality is just rearranging the algebra to fit Property P2 for our new EDCS parameters of $\beta+2, 2\lambda$.

For property P1, note that $X^M \subseteq M_G^{G\setminus H}$ is a matching, so for every vertex v we have $\deg_H(v) \leq \deg_{H\cup X^M}(v) \leq \deg_H(v) + 1$. Now, for $(u,v) \in H$ we had $\deg_H(u) + \deg_H(v) \leq \beta$ (by property P1 of H), and for $(u,v) \in X^M \subseteq X$ we had $\deg_H(u) + \deg_H(v) < \beta$ (by definition of X). Thus, for every $(u,v) \in H \cup X^M$ we have that $\deg_{H\cup X^M}(u) + \deg_{H\cup X^M}(v) \leq \deg_H(u) + \deg_H(v) + 2 \leq \beta + 2$.

Note that because of how we set the parameters, $\beta' = \beta + 2 < 2\beta$ and $\lambda' = 2\lambda$ satisfy the requirements of Lemma 7. We thus have that $\mu(H \cup X) \geq \mu(H \cup X^M) \geq (2/3 - \epsilon)\mu(H \cup M_G^{G\setminus H}) = (2/3-\epsilon)\mu(G)$. ◀

4 The Algorithm

4.1 The Two Phases

Our algorithm will proceed in two phases. Once phase I terminates, the algorithm proceeds to phase II and never returns to phase I. The goal of phase I is to construct a suitable subgraph H of G. We now state the formal properties that will be guaranteed by phase I.

▶ **Definition 11** (parameters). *Throughout this section we use the following parameters. Let $\epsilon < 1/2$ be the final approximation parameter we are aiming for. Set $\lambda = \frac{\epsilon}{128}$ and set $\beta = 16\lambda^{-2}\log(1/\lambda)$; note that λ and β are $O(poly(1/\epsilon))$. Set $\alpha = \frac{\epsilon m}{n\beta^2+1} = O(\frac{m}{n} poly(1/\epsilon))$ and $\gamma = 5\log(n)\frac{m}{\alpha} = O(n\log(n) poly(1/\epsilon))$.*

▶ **Lemma 12.** *Phase I uses $O(n\beta) = O(npoly(1/\epsilon))$ space and constructs a subgraph H of G. The phase satisfies the following properties:*

1. *Phase I terminates within the first ϵm edges of the stream. That is, Phase I terminates at the end of processing some edge e_i with $i \leq \epsilon m$.*
2. *When Phase I terminates at the end of processing some edge e_i, the subgraph $H \subseteq G$ constructed during this phase satisfies the following properties:*
   a. *H has bounded edge-degree β. As a corollary, H has $O(n\beta)$ edges.*
   b. *With probability at least $1 - n^{-3}$, the total number of $(G_{>i}, H, \beta, \lambda)$-underfull edges in $G_{>i} \setminus H$ is at most γ. (Recall that $G_{>i}$ denotes the subgraph of G that contains all edges in $\{e_{i+1}, \ldots, e_m\}$.)*

We now show that if we can ensure the properties of Lemma 12, our main result follows.

Proof of Theorem 1. Let us say that Phase I terminates after edge e_i and let H be the subgraph constructed by Phase I. Phase II of the algorithm proceeds as follows. It initializes an empty set X. Then, for every edge (u,v) in $S_{[i+1,m]}$, if $\deg_H(u) + \deg_H(v) < \beta(1-\lambda)$ (that is, if (u,v) is $(G_{>i}, H, \beta, \lambda)$-underfull), the algorithm adds edge (u,v) to X. After the algorithm completes the stream, it then returns the maximum matching in $H \cup X$.

Let us now analyze the approximation ratio. By property 1 of Lemma 12, $G_{>i} \subseteq G^{late}$; thus, X contains all $(G^{late}, H, \beta, \lambda)$-underfull edges. By property 2a, H has bounded edge-degree β. Thus, applying Lemma 10, we have that $\mu(H) \geq (2/3 - \epsilon)\mu(G^{late})$. Combining this with Lemma 5, we get that $\mu(H) \geq (2/3 - \epsilon)(1 - 2\epsilon)\mu(G) \geq (2/3 - 3\epsilon)\mu(G)$; using $\epsilon' = \epsilon/3$ thus yields the desired approximation ratio.

For the space analysis, we know from Lemma 12 that Phase I requires $O(n\beta)$ space, which is the space needed to store subgraph H. By Property 2b, the size of X in Phase II is at most $O(n \log(n))$. The overall space is thus $O(n \log(n) + n\beta) = O(n \log(n) + n\text{poly}(1/\epsilon))$.

Finally, note that the only two probabilistic claims are Lemma 5 and Property 2b of Lemma 12, both of which hold with probability $\geq 1 - n^{-3}$. A union bound thus yields an overall probability of success $\geq 1 - 2n^{-3}$. ◀

4.2 Description of Phase I

All we have left is to describe Phase I and prove Lemma 12. See Algorithm 1 for pseudocode of the entire algorithm. Recall the parameters $\epsilon, \beta, \lambda, \alpha, \gamma$ from Definition 11. Phase I is split into epochs, each containing exactly α edges from the stream. So in epoch i, the algorithm looks at $S_{[(i-1)\alpha+1, i\alpha]}$.

Phase I initializes the graph $H = \emptyset$. In epoch i, the algorithm goes through the edges of $S_{[(i-1)\alpha+1, i\alpha]}$ one by one. For edge (u, v), if $\deg_H(u) + \deg_H(v) < (1 - \lambda)\beta$, then the algorithm adds edge (u, v) to H (Line 5). (Note that the algorithm changes H over time, so $\deg_H(u) + \deg_H(v)$ always refers to the degrees in H at the time edge (u, v) is being examined.) After each edge insertion to H, the algorithm runs procedure *RemoveOverfullEdges(H)* (Line 7); this procedure repeatedly picks an edge (x, y) with $\deg_H(x) + \deg_H(y) > \beta$ until no such edge remains. Note that as a result, our algorithm preserves the invariant that H always has bounded edge-degree β.

In each epoch, the algorithm also has a single boolean FoundUnderfull, which is set to True if the algorithm ever adds an edge to H during that epoch. At the end of the epoch, if FoundUnderfull is set to True, then the algorithm simply proceeds to the next epoch. If FoundUnderfull is False, then the algorithm permanently terminates Phase I and proceeds to Phase II. (The intuition is that since the ordering of the stream is random, if the algorithm failed to find an underfull edge in an entire epoch, then there must be relatively few underfull edges left in the stream, so Property 2b of Lemma 12 will be satisfied.)

Note that FoundUnderfull being false is the only way Phase I can terminate (Line 9); we prove in the analysis that this deterministically occurs within the first ϵm edges of the stream.

4.3 Analysis

We now turn to proving Lemma 12. The hardest part is proving Property 1. Observe that every epoch that doesn't terminate Phase I must add at least one edge to H. To prove Property 1, we use an auxiliary lemma that bounds the total number of changes made to H.

▶ **Lemma 13.** *Fix any parameter $\beta > 2$. Let $H = (V_H, E_H)$ be a graph, with E_H initially empty. Say that an adersary adds and removes edges from H using an arbitrary sequence of two possible moves*

- [Deletion Move]. *Remove an edge (u, v) from H for which $\deg_H(u) + \deg_H(v) > \beta$*
- [Insertion Move]. *Add an edge (u, v) to H for some pair $u, v \in V$ for which $\deg_H(u) + \deg_H(v) < \beta - 1$.*

Then, after $n\beta^2$ moves, no legal move remains.

Algorithm 1 The algorithm for computing a matching in a random-order stream. After initialization, the algorithm goes to Phase I. Once the algorithm exits Phase I, it moves on to Phase II and never returns to Phase I. Line 9 is the only place where the algorithm can exit Phase I.

Procedure *Initilization*
- Initialize $H = \emptyset$ `/* H is a global variable modified by Phase I */`
- Let $\epsilon < 1/2$ be the main approximation parameter
- Set $\lambda = \frac{\epsilon}{128}$, $\beta = 16\lambda^{-2}\log(1/\lambda)$, $\alpha = \frac{\epsilon m}{n\beta^2+1}$, $\gamma = 5\log(n)\frac{m}{\alpha}$ (Definition 11).
- **Go To** Phase I

Procedure *Phase I*
- **Do Until Termination** `/* each iteration corresponds to one epoch */`
  - (1) FoundUnderfull $\leftarrow$ FALSE
  - (2) **for** α *Iterations:* **do** `/* each epoch looks at exactly` α `edges. */`
    - (3) Let (u, v) be the next edge in the stream
    - (4) **if** $\deg_H(u) + \deg_H(v) < \beta(1-\lambda)$ **then**
      - (5) Add edge (u, v) to H `/* note: this increases` $\deg_H(u)$ `and` $\deg_H(v)$. `*/`
      - (6) FoundUnderfull $\leftarrow$ TRUE
      - (7) *RemoveOverfullEdges(H)*
  - (8) **if** *FoundUnderfull = FALSE* **then**
    - (9) **Go To** Phase II `/* permanently exit Phase I. */` ;
  - `/* Else, will move on to the next epoch of Phase I. */`

Procedure *RemoveOverfullEdges(H)*
- (1) **while** *there exists* $(u, v) \in H$ *such that* $\deg_H(u) + \deg_H(v) > \beta$ **do**
  - (2) Remove (u, v) from H `/* note: this decreases` $\deg_H(u)$ `and` $\deg_H(v)$ `*/`

`/* note: when the while loop terminates,` H `is guaranteed to have bounded edge-degree` β. `*/`

Procedure *Phase II*
- (1) Initialize $X \leftarrow \emptyset$ `/* all underfull edges will be added to` X `*/` ;
- (2) **foreach** *remaining edge* (u, v) *in the stream* **do**
  - (3) **if** $\deg_H(u) + \deg_H(v) < \beta(1-\lambda)$ **then**
    - (4) Add edge (u, v) to X `/* note: this does NOT change any` $\deg_H(v)$. `*/`
- (5) **Return** the maximum matching in $H \cup X$;

Proof. The proof is similar to that of Proposition 2.4 in [4]. Define the following potential functions $\Phi_1(H) = (\beta - 1/2) \cdot \sum_{v \in V_H} \deg_H(v)$, $\Phi_2(H) = \sum_{(u,v)\in E_H} \deg_H(u) + \deg_H(v)$, and the main potential function $\Phi(H) = \Phi_1(H) - \Phi_2(H)$. Note that initially H is empty so $\Phi(H) = 0$. We claim that at all times $\Phi(H) \le \Phi_1(H) \le n\beta^2$. To see this, note that every vertex $v \in V_H$ always has $\deg_H(v) \le \beta$, because as long as $\deg_H(v) = \beta$, the adversary cannot perform any insertion moves incident to v. In the rest of the proof, we show that every Insertion/Deletion move increases $\Phi(H)$ by at least 1; combined with the fact that at all times $0 \le \Phi(H) \le n\beta^2$, we get that there are at most $n\beta^2$ moves in total.

Consider any Deletion Move of edge (u, v). Clearly $\Phi_1(v)$ decreases by exactly $2\beta - 1$. We now show that $\Phi_2(v)$ decreases by at least 2β. One the one hand, $\Phi_2(v)$ decreases by at least $\beta + 1$ because edge (u, v) no longer participates in the sum, and $\deg_H(u) + \deg_H(v)$ was $> \beta$ before the deletion. But at the same time, since $\deg_H(u) + \deg_H(v) \geq \beta + 1$ before the deletion, there are at least $\beta - 1$ edges other than (u, v) incident to u or v, and each of their edge degrees decrease by 1 in the sum for $\Phi_2(H)$. Thus, $\Phi_2(H)$ decreases by at least $\beta + 1 + (\beta - 1) = 2\beta$, while $\Phi_1(H)$ decreases by exactly $2\beta - 1$, so overall $\Phi(H) = \Phi_1(H) - \Phi_2(H)$ increases by at least one.

Similarly, consider any Insertion Move of edge (u, v). Clearly $\Phi_1(v)$ increases by exactly $2\beta-1$. We now show that $\Phi_2(v)$ increases by at most $2\beta-2$. Recall that $\deg_H(u)+\deg_H(v) \leq \beta - 2$ before the insertion, so after the insertion we have that $\deg_H(u) + \deg_H(v) \leq \beta$, so the edge (u, v) itself contributes at most β to the sum in Φ_2. There are also at most $\beta - 2$ edges other than (u, v) incident to u or v, each of whose edge degrees increases by 1. Thus, overall, $\Phi_2(H)$ increases by at most $\beta + (\beta - 2) = 2\beta - 2$, so $\phi(H)$ increases by at least $(2\beta - 1) - (2\beta - 2) = 1$. ◀

Proof of Lemma 12. Property 2a is clearly satisfied by construction, because after any insertion to H the algorithm runs *RemoveOnderfullEdges(H)* (line 7) to ensure that H has bounded edge-degree β. As a result, we clearly have that every *vertex* degree is at most β, so Phase I needs only $O(n\beta)$ space to store H.

For the proof of Property 1, observe that any changes the algorithm makes to H follow the rules for Insertion/Deletion moves from Lemma 13, so Algorithm 1 makes at most $n\beta^2$ changes to H. (Line 5 of Phase I corresponds to deletion moves in Lemma 13, while line 2 of *RemoveOverfullEdges(H)* corresponds to insertion moves. Note that line 5 of phase I actually obeys an even stronger inequality than deletion moves, since $\beta(1 - \lambda) < \beta - 1$.) Each epoch that does not terminate Phase I makes at least one change to H, so phase I goes through at most $n\beta^2 + 1$ epochs before termination. Each epoch contains α edges, so overall Phase I goes through at most $\alpha(n\beta^2 + 1) = \epsilon m$ edges, as desired.

All that remains is to prove Property 2b. As mentioned above, the intuition is simple: the algorithm only exits Phase I if it fails to find a single underfull edge in the entire epoch (Line 9), and since the stream is random, such an event implies that there are probably relatively few underfull edges left in the stream. We now formalize this intuition.

Let $\mathcal{A}_i$ be the event that FoundUnderfull is set to FALSE in epoch i. Recall that epoch i ends on edge $e_{i\alpha}$; let $\mathcal{B}_i$ be the event that the number of $(G_{>i\alpha}, H, \beta, \lambda)$-underfull edges is more than γ. Note that Property 2b fails to hold if and only if we have $\mathcal{A}_i \wedge \mathcal{B}_i$ for some i, so we now upper bound $\Pr[A_i \wedge B_i]$. Our bound relies on the randomness of the stream. Let E_i^r contain all edges in the graph that have not yet appeared in the stream at the *beginning* of epoch i (r for remaining). Let E_i^e be the edges that appear in epoch i (e for epoch), and note that E_i^e is a subset of size α chosen uniformly at random from E_i^r. Define H_i to be the subgraph H at the beginning of epoch i, and define $E_i^u \subseteq E_i^r$ to be the set $\{(u, v) \in E_i^r \mid \deg_{H_i}(u) + \deg_{H_i}(v) < \beta(1 - \lambda)\}$ (u for underfull). Observe that because of event $\mathcal{A}_i$, the graph H does not change throughout epoch i, so an edge that is underfull at any point during the epoch will be underfull at the end as well. Thus, $\mathcal{A}_i \wedge \mathcal{B}_i$ is equivalent to the event that $|E_i^u| > \gamma$ but $E_i^u \cap E_i^e = \emptyset$.

Let $\mathcal{A}_i^k$ be the event that the kth edge of epoch i is not in E_i^u. We have that

$$\Pr[\mathcal{B}_i \wedge \mathcal{A}_i] \leq \Pr[\mathcal{A}_i \mid \mathcal{B}_i] = \Pr[\mathcal{A}_i^1 \mid \mathcal{B}_i] \prod_{k=2}^{\alpha} \Pr[\mathcal{A}_i^k \mid \mathcal{B}_i, \mathcal{A}_i^1, \ldots, \mathcal{A}_i^{k-1}].$$

Now, observe that

$$\Pr[\mathcal{A}_i^1 \mid \mathcal{B}_i] < 1 - \frac{\gamma}{m}$$

because the first edge of the epoch is chosen uniformly at random from the set of $\leq m$ remaining edges, and the event fails if the chosen edge is in E_i^u, where $|E_i^u| > \gamma$ by definition of $\mathcal{B}_i$. Similarly, for any k,

$$\Pr[\mathcal{A}_i^k \mid \mathcal{B}_i, \mathcal{A}_i^1, \ldots, \mathcal{A}_i^{k-1}] < 1 - \frac{\gamma}{m}$$

because conditioning on the previous events $\mathcal{A}_i^j$ implies that no edge from E_i^u has yet appeared in this epoch, so there are still at least γ edges from E_i^u left in the stream.

Recall from Definition 11 that $\gamma = 5\log(n) \cdot \frac{m}{\alpha}$. Combining the three above equations yields that $\Pr[\mathcal{B}_i \wedge \mathcal{A}_i] \leq (1 - \frac{\gamma}{m})^\alpha = (1 - \frac{5\log(n)}{\alpha})^\alpha \leq n^{-5}$. There are clearly at most n^2 epochs, so union bounding over all of them shows that Property 2b fails with probability at most n^{-3}, as desired. ◀

5 Open Problems

We presented a new single-pass streaming algorithm for computing a maximum matching in a *random-order* stream. The algorithm achieves a $(2/3 - \epsilon)$-approximation using $O(n\log(n))$ space; these bounds improve upon all previous results for the problem.

But while $2/3$ is a natural boundary, there is no reason to believe it is the best possible. Is there an algorithm with approximation ratio $2/3 + \Omega(1)$? Is it possible to compute a $(1-\epsilon)$-approximate matching in random-order streams? A lower bound of $1 - \Omega(1)$ in this setting would also be extremely interesting.

Another natural open problem is get improved bounds for weighted graphs. Gamlath et al. [18] recently broke through the barrier of $1/2$ and presented an algorithm for weighted graphs that computes a .506-approximation (or .512 in bipartite graphs) in random-order streams. Can we improve the approximation ratio to $2/3$ in weighted graphs? To $(1-\epsilon)$?

References

1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013.

2 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 202–211, 2015.

3 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635, 2019.

4 Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 11:1–11:20, 2019.

5 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, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742, 2017.

6 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364. SIAM, 2016.

7 Aaron Bernstein and Cliff Stein. Fully dynamic matching in bipartite graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 167–179, 2015.

8 Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 263–274, 2015.

9 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344, 2016.

10 Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1234–1251, 2015.

11 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 29:1–29:15, 2017.

12 Michael Crouch and Daniel S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 96–104, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.

13 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.

14 Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. *ACM Trans. Algorithms*, 14(4):48:1–48:23, 2018.

15 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016.

16 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1773–1785, 2020.

17 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.

18 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500. ACM, 2019.

19 Mohsen Ghaffari and David Wajc. Simplified and space-optimal semi-streaming (2+epsilon)-approximate matching. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd*

Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8–9, 2019 – San Diego, CA, USA, volume 69 of *OASICS*, pages 13:1–13:8. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2019.

20 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 468–485. SIAM, 2012. URL: http://dl.acm.org/citation.cfm?id=2095116.2095157.

21 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298, 2013.

22 Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

23 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013. doi:10.1137/1.9781611973105.121.

24 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014. doi:10.1137/1.9781611973402.55.

25 Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1753–1772, 2020.

26 Christian Konrad. Maximum matching in turnstile streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 840–852, 2015.

27 Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

28 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242, 2012.

29 Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 170–181, 2005. doi:10.1007/11538462_15.

30 Andrew McGregor and Sofya Vorotnikova. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 17:1–17:12, 2016.

31 Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 14:1–14:4, 2018.

32 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, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2153–2161, 2017.

33 David Wajc. Negative association: definition, properties, and applications. URL: `https://www.cs.cmu.edu/~dwajc/notes/Negative%20Association.pdf`.