On Regularity Lemma and Barriers in Streaming and Dynamic Matching*

Sepehr Assadi
sepehr.assadi@rutgers.edu
Rutgers University
New Brunswick, New Jersey, USA

Soheil Behnezhad
s.behnezhad@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

Sanjeev Khanna
sanjeev@cis.upenn.edu
University of Pennsylvania
Philadelphia, Pennsylvania, USA

Huan Li
huanli@cis.upenn.edu
University of Pennsylvania
Philadelphia, Pennsylvania, USA

ABSTRACT

We present a new approach for finding matchings in dense graphs by building on Szemerédi's celebrated Regularity Lemma. This allows us to obtain non-trivial albeit slight improvements over longstanding bounds for matchings in streaming and dynamic graphs. In particular, we establish the following results for n-vertex graphs:

(i) A deterministic **single-pass streaming** algorithm that finds a $(1-o(1))$-approximate matching in $o(n^2)$ bits of space. This constitutes the first single-pass algorithm for this problem in sublinear space that improves over the $1/2$-approximation of the greedy algorithm.

(ii) A randomized **fully dynamic** algorithm that with high probability maintains a $(1-o(1))$-approximate matching in $o(n)$ worst-case update time per each edge insertion or deletion. The algorithm works even against an adaptive adversary. This is the first $o(n)$ update-time dynamic algorithm with approximation guarantee arbitrarily close to one.

Given the use of regularity lemma, the improvement obtained by our algorithms over trivial bounds is only by some $(\log^* n)^{\Theta(1)}$ factor. Nevertheless, in each case, they show that the "right" answer to the problem is not what is dictated by the previous bounds.

Finally, in the streaming model, we also present a randomized $(1-o(1))$-approximation algorithm whose space can be upper bounded by the density of certain Ruzsa-Szemerédi (RS) graphs. While RS graphs by now have been used extensively to prove streaming lower bounds, ours is the first to use them as an upper bound tool for desigining improved streaming algorithms.

*Sepehr Assadi is supported in part by an NSF CAREER Grant CCF-2047061, a Sloan Research Fellowship, a Google Research gift, and a Fulcrum award from Rutgers Research Council. Sanjeev Khanna is supported in part by NSF awards CCF-1934876 and CCF-2008305. Huan Li is supported in part by NSF award CCF-2008305.

STOC '23, June 20–23, 2023, Orlando, FL, USA

ACM ISBN 978-1-4503-9913-5/23/06…$15.00
https://doi.org/10.1145/3564246.3585110

CCS CONCEPTS

• **Theory of computation** → **Streaming models**; *Communication complexity*;

KEYWORDS

Maximum Matching, Streaming Algorithms, Dynamic Algorithms, Regularity Lemma

ACM Reference Format:

Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. 2023. On Regularity Lemma and Barriers in Streaming and Dynamic Matching. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC '23), June 20–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3564246.3585110

1 INTRODUCTION

Given a graph $G = (V, E)$, a matching M in G is any collection of edges that share no endpoints. Finding maximum matchings has been a cornerstone of algorithm design starting from the work of König [64] over a century ago. Nevertheless, many fundamental questions regarding the complexity of this problem have remained unresolved, specifically in modern models of computations such as streaming or dynamic graphs. Indeed, in both mentioned models, despite significant attention, there has been no improvement in certain key cases over longstanding barriers that have remained in place since the introduction of the model itself.

In this paper, we make an ever so slight improvement over these barriers, showing that the *right* answer to the problem must be different than what is dictated by prior bounds. Our results combine tools from extremal combinatorics, primarily Szemerédi's Regularity Lemma [80] and its extensions, with multiple ideas (old and new) tailored to each model specifically. To put our results in more context, we start with the history of the problem in each model separately.

Graph Streaming. In this model, edges of an n-vertex graph appear one by one in a stream in an arbitrary order. The algorithm can read the edges in the arrival order while using a limited memory smaller than the input size, and output the solution at the end of the stream. The holy grail of algorithms here is a one that uses $O(n \cdot \text{polylog}(n))$ memory and a single pass over the stream. The study of graph streaming algorithms were initiated by Feigenbaum et al. [43] who already observed that there is a straightforward $1/2$-approximation algorithm for matching in $O(n \log n)$ space[1]: greedily maintain a maximal matching in the stream. They further proved that finding an exact maximum matching requires $\Omega(n^2)$ space, which matches the trivial algorithm that stores the entire input via its adjacency matrix.

Almost two decades since [43], there are still no better algorithms for matchings than these two straightforward solutions. On the lower bound front, a series of work by Goel et al. [50] and Kapralov [59] culminated in a recent work of Kapralov [60] that rules out better than $1/(1+\ln 2) \approx 0.59$ approximation in $n^{1+o(1/\log\log n)}$ space. This lack of progress has led researchers to consider various relaxations of the problem, in particular by allowing a few more passes over the input (e.g., in [9, 44, 58, 66–68]) or assuming random arrival of edges in the stream (e.g., in [10, 11, 25, 42, 67])[2]. At this point, beating $1/2$-approximation factor of the greedy algorithm in $O(n \cdot \text{polylog}(n))$ space, or even much larger than that, has become one of the most central open questions of the graph streaming literature; see, e.g., [44, 60, 67, 71, 82] for various references to this question.

(Fully) Dynamic Graphs. In this model, we have an n-vertex graph that undergoes an arbitrary sequence of edge insertions and deletions. The goal is to maintain the solution to the problem, say an approximate maximum matching of the graph, with a quick update time per each insertion or deletion. Dynamic algorithms for matchings were studied first in this model by Ivkovic and Lloyd [57] in 1993 and continue to be a highly active area of research (see, e.g. [8, 19, 20, 22–24, 26, 27, 29–33, 37, 52, 53, 63, 72, 73, 76, 78, 83] and references therein).

There is a folklore algorithm that for any $\varepsilon > 0$, maintains a $(1-\varepsilon)$-approximate matching in $O(n/\varepsilon^2)$ (amortized) update time: Assume inductively that we have a $(1-\varepsilon/2)$-approximate matching M of the current graph; (i) for the next $(\varepsilon/2) \cdot |M|$ updates do nothing and return M still as the answer; after that, (ii) compute a $(1-\varepsilon/2)$-approximate matching of the current graph in $O(m/\varepsilon)$ time using the Hopcroft-Karp algorithm [56] where m is the number of edges in the graph and repeat from step (i). Since $m = O(n \cdot |M|)$ in any graph with maximum matching size bounded by $O(|M|)$, the amortized update time will be $O(n/\varepsilon^2)$, and the correctness can be easily verified. This algorithm can also be deamortized using standard batching ideas.

For sparser graphs, this folklore algorithm was improved by Gupta and Peng [53] to achieve an $O(\sqrt{m}/\varepsilon^2)$ update time where m denotes the (dynamic) number of edges. Faster algorithms are only known for smaller approximations between $1/2$ and $2/3$ which can respectively be maintained in $O(1)$ [79] (see also [19]) and $O(\sqrt{n})$ update-times [29]. See also a recent result of [23] for update-time/approximation trade-offs between $1/2$ and $2/3$, and the recent breakthroughs of [21, 34] in beating $1/2$-approximation in polylogarithmic time for estimating size of maximum matching. Yet, for the original $(1-\varepsilon)$-approximation question, raised e.g. in [53], an $O(n)$ update time still remains a barrier in general.

1.1 Our Contributions

We present the first algorithms that beat the aforementioned barriers for finding matchings in streaming and dynamic graphs with non-trivial albeit quite small factors:

Result 1 (Formalized in Theorem 3). *There is a randomized $(1-o(1))$-approximate matching algorithm in single-pass streams with adversarial order of edge arrivals in $n^2/(\log^* n)^{\Omega(1)}$ space and polynomial time.*

This is the first $o(n^2)$-space algorithm for matchings in adversarial-order streams with better than $1/2$-approximation guarantee. In fact, it was not known previously how to achieve a $(1-o(1))$-approximation in $o(n^2)$ space even on random-arrival streams and even if we allow any constant number passes over the input (see [1, 2, 13, 17, 46, 70] for representative examples of multi-pass streaming matching algorithms[3]). Moreover, combined with the lower bound of $\Omega(n^2)$ space by [43] for computing exact matchings, Result 1 shows the first provable separation between the space complexity of computing nearly-optimal versus exact-optimal matchings in single-pass streams.

Result 2 (Formalized in Theorem 2). *There is a randomized $(1-o(1))$-approximate matching algorithm in fully dynamic graphs against an adaptive adversary with $n/(\log^* n)^{\Omega(1)}$ worst case update time.*

This is the first algorithm for matchings in fully dynamic graphs that achieves $o(n)$ update time for all densities with close to one approximation guarantee (this was not known before even for oblivious adversaries).

The key idea behind both these results is to maintain a *matching cover*—introduced by Goel et al. [50] in spirit of cut/spectral sparsifiers—that is a "sparse" subgraph which approximately preserve matchings in each induced subgraph of the input graph. We present a polynomial time algorithm for constructing $o(n^2)$-size matching covers using Szemerédi's Regularity Lemma [80] and along the way extend them to general graphs ([50] only proves their existence and for bipartite graphs). We then show this new construction can be maintained in streaming and dynamic graphs using several new ideas combined with standard tools from prior work specific to each model. We elaborate more on our techniques in Section 2.

We also present a third result specific to the graph streaming model. *All* previous lower bounds for approximating matchings in

[1] Throughout, we always measure the space of streaming algorithms in *bits*.

[2] See the papers of Feldman and Szarf [44] and Assadi and Behnezhad [11], respectively, for the state of the art in each case, and more details on previous work on each relaxation.

[3] The state-of-the-art is the $O_\varepsilon(n^{1+1/p})$-space $O(p/\varepsilon)$-pass algorithm by Ahn and Guha [2] and $O_\varepsilon(n \cdot \text{polylog}(n))$-space $\text{poly}(1/\varepsilon)$-pass by Fischer et al. [46] (see also algorithms by Assadi et al. [13] and Assadi et al. [17] with improved bounds for bipartite graphs).

graph streams in a single pass [14, 50, 59, 60], multi-pass [9, 18, 38], or random-order streams [11] rely on constructions based on Ruzsa-Szemerédi (RS) graphs [77]. These are graphs whose edges can be partitioned into "large" *induced* matchings (see Section 3.3 for details). We present a converse approach by developing a streaming algorithm for matchings whose space can be *upper bounded* by the density of (certain) RS graphs. In particular,

Result 3. (Formalized in Theorem 4) *For any $k \geqslant 1$, there is a randomized $(1 - o(1))$-approximate matching algorithm in single-pass streams with adversarial order of edge arrivals in*

$$\left(n^2/k + RS(n, o(n/k))\right) \cdot \text{polylog}\,(k)$$

space and exponential time; here, $RS(n, r)$ denotes the largest number of edges in any n-vertex graph whose edges can be partitioned into induced matchings of size r. The algorithm can be made deterministic if the goal is an additive $o(n)$ approximation instead.

Result 3 builds on and generalize the RS graph based communication protocol of Goel et al. [50] to the streaming model (and from bipartite to general graphs).

To put this result in more context, notice that RS graphs are naturally becoming denser and denser by reducing the size of their induced matchings[4], leading to a tradeoff between the two terms in the space guarantee of Result 3. Unfortunately, proving tight bounds on the density of RS graphs is a notoriously difficult problem in combinatorics (see, e.g., [40, 48, 51]). As such, the space complexity of the algorithm in Result 3 as purely a function of n is not clear at this point. However, using Result 3 combined with Fox's triangle-removal lemma [47] that, to our knowledge, provides the best approach currently for bounding density of RS graphs with $o(n)$-size induced matchings, we can obtain the following result:

- **A corollary of Result 3** (Formalized in Corollary 6.6)**.** There is a deterministic $(1 - o(1))$-approximate matching algorithm in single-pass streams with adversarial order of edge arrivals using $n^2/2^{\Theta(\log^* n)}$ space and exponential time.

This corollary improves upon our algorithm in Result 1 based on the regularity lemma in terms of approximation ratio and being deterministic at the cost of taking exponential time. Moreover, by a result of Goel et al. [50] on lower bounds for streaming matching via RS graphs, obtaining streaming algorithms with better space complexity than this corollary, namely, beating n^2 by more than a $2^{\Theta(\log^* n)}$ factor, immediately implies improved RS graph upper bounds; in other words, **improving upon our algorithm at the very least requires proving better RS graph upper bounds than currently known bounds** (see [49] for why this can be challenging).

Finally, given the current state of knowledge about RS graphs (see [6, 48]), it is possible that the space of the algorithm in Result 3 can be improved to $n^2/2^{\Theta(\sqrt{\log n})}$—thus more than any polylog(n) factor shaving in the space over n^2—assuming that the currently best construction of dense RS graphs in [77] (see also [6]) with induced matchings of size $n/2^{\Theta(\sqrt{\log n})}$ cannot be improved substantially to larger induced matching sizes.

[4]Any (simple) graph can be seen as an RS graph with induced matchings of size one.

In conclusion, our paper shows that these longstanding barriers in computing large matchings in streaming and dynamic graphs can at least be broken by some non-trivial albeit quite small factors. Moreover, these algorithms rely on techniques and ideas that are vastly different from prior approaches used in these two models. We hope our work paves the path toward further progress on these longstanding open questions.

2 TECHNICAL OVERVIEW

Matching sparsifiers, which loosely speaking, are sparse subgraphs that approximately preserve the maximum matching have long been known to be an important tool for fully dynamic and (variants of) streaming algorithms. Some prominent examples include edge-degree constrained subgraphs (EDCS) [28, 29] and its generalizations [12, 23], kernels [8, 26, 30, 31], and matching skeletons [50]. One of our main contributions, and the key to both Result 1 and Result 2, is a new matching sparsifier based on Szemerédi's Regularity Lemma.

Our matching sparsifier, more strongly, is a *matching cover*—à la Goel et al. [50]—which not only preserves an approximate *maximum* matching of the graph, but rather "covers" smaller matchings of it as well. Let us formalize this. For a given graph G, we write $\mu(G)$ to denote the maximum matching size of G, and write $G[A, B]$ to denote the bipartite subgraph of G between some disjoint vertex subsets A, B. We say a subgraph H of G is an **α-matching cover** for $\alpha \in (0, 1)$ if for any disjoint subsets of vertices (A, B) in G, $\mu(H[A, B]) \geqslant \mu(G[A, B]) - \alpha \cdot n$. That is, H preserves the largest matching in the induced bipartite subgraph $G[A, B]$ to within an additive $\alpha \cdot n$ factor. While from an information theoretic perspective, *existence* of an $o(n^2)$-edge $o(1)$-matching cover for bipartite graphs was proved in the original paper of Goel et al. [50], it was not known up until now whether one can find such matching covers efficiently, say in polynomial time. Note that this is specially important, for instance, for applications in dynamic algorithms where the goal is to optimize the update time.

In this paper, we prove that there is an $\tilde{O}(n^\omega)$-time[5] offline algorithm that computes an $o(n^2)$-edge $o(1)$-matching cover of any n-vertex graph (not necessarily bipartite). Our algorithm builds on Szemerédi's Regularity Lemma (and its algorithmic version due to Alon et al. [5]). We first explain how our offline algorithm for obtaining a matching cover works, and then outline its use in obtaining improved dynamic matching and streaming algorithms.

2.1 Matching Covers via Regularity Lemma

Roughly speaking, Szemerédi's regularity lemma [80] says that the vertices of any graph can be partitioned into a small *irregular* part C_0 with $|C_0| = o(n)$, plus k other equal-size parts $C_1, \ldots, C_k$ for some $k \in [\omega(1), \log n]$. The latter k parts have the property that all but $o(1)$-fraction of the C_i, C_j pairs are *regular*: for any pair of subsets $X \subseteq C_i, Y \subseteq C_j$ with large enough size, the edge density between X, Y is similar to that of C_i, C_j. Therefore, if the edges

[5]Here and throughout, $\omega \approx 2.37286$ is the matrix multiplication exponent with current best bounds achieved by Alman and Williams [3].

between C_i, C_j are dense to start with, the density will also be high between every large enough $X \subseteq C_i, Y \subseteq C_j$ pair.

It is not difficult to see that by regularity, any large matching between a dense regular pair C_i, C_j can be mostly preserved if we subsample edges between them at a sufficiently high rate $p = o(1)$. In particular, the subsampled graph will be a matching cover of the graph induced by edges between the regular pair C_i, C_j. This suggests a natural strategy for building an $o(1)$-matching cover with $o(n^2)$ edges: *subsample* the edges between dense regular C_i, C_j pairs at rate $p = o(1)$ and take *all* other edges. We would like to show that this is an α-matching cover for some $\alpha = o(1)$.

This idea runs into the following problem. Suppose we have an (αn)-size matching M whose edges are evenly distributed across all $\binom{k}{2}$ pairs of C_i, C_j, then the number of edges of M between each C_i, C_j pair is only $O(\alpha \cdot n/k^2)$. This means that only an $O(\alpha/k)$ fraction of vertices in C_i, C_j are matched to each other – this is unfortunately way too small to invoke the regularity property.

We get around this issue by first focusing on solving an *α-hitting set* problem: find one edge between endpoints of any (αn)-size matching – we will show later on using a similar argument as in [50] that this is sufficient for obtaining an α-matching cover. Now to fix our problem about an (αn)-size matching whose edges are distributed across many pairs, we present a strategy for *consolidating the support* of a matching over different pairs. This consolidation argument shows that whenever there is a large matching M between dense regular C_i, C_j pairs, there must also exist another (almost as) large matching M' that is supported on the same set of vertices $V(M)$ but only uses edges between a small number of such C_i, C_j pairs. As a result, there must exist one pair of C_i, C_j where a substantial fraction of vertices are matched to each other, to which we are now able to apply regularity to prove the existence of an edge between them in the subsampled graph (which solves our α-hitting set problem). At a high level, our argument for consolidating the support of the matching is proved by (i) viewing the matching M as a *fractional* matching in a meta graph obtained by contracting each C_i into a supernode; and (ii) rounding the fractional matching by an edge sampling process.

All in all, using the algorithm of [5] for finding the regularity lemma partition in $\widetilde{O}(n^\omega)$ time, and a direct sampling algorithm between dense regular pairs, this step gives us an $\widetilde{O}(n^\omega)$ time and $\widetilde{O}(n)$ space algorithm for finding an α-matching cover of size $n^2/(\log^* n)^{\Omega(1)}$ for some $\alpha = 1/(\log^* n)^{\Omega(1)}$.

2.2 Applications of Matching Cover

A fully dynamic matching algorithm. The matching cover algorithm above is offline. But observe that since the algorithm takes $\tilde{O}(n^\omega) = n^{3-\Omega(1)}$ time, the time spent *per edge* in a dense instance is sublinear in n. This gives hope that perhaps such a matching cover can be maintained in $o(n)$ time, and indeed we show this to be the case.

Our algorithm roughly proceeds by re-computing an $o(1)$-matching cover every $\tilde{\Theta}(n^{\omega-1})$ updates, and then using the $O(\sqrt{m})$-update time data structure by Gupta and Peng [53] to maintain a nearly optimal matching in the matching cover through the subsequent $\tilde{\Theta}(n^{\omega-1})$ updates. Since the matching cover only has $o(n^2)$ edges, we immediately get an update time of $o(n)$ for the Gupta-Peng algorithm. To argue the correctness, we show that the matching cover found by our offline algorithm has the additional feature that it is *robust to edge updates*: not only is it an $o(1)$-matching cover of the graph at the time we compute it, but it remains an $o(1)$-matching cover throughout any arbitrary sequence of $n^{2-o(1)}$ updates. This suffices to show that our algorithm can dynamically maintain an approximate matching with an additive error $o(n)$.

When the number of edges is close to n^2, this additive approximation becomes a $(1 - o(1))$-multiplicative approximation, since the maximum matching size is itself $\Omega(n)$. On the other hand, when the number of edges is $o(n^2)$, directly applying the Gupta-Peng data structure gives us a nearly-optimal matching in $o(n)$ update time. Our final algorithm then balances the dense and the sparse regimes together to maintain a $(1 - o(1))$-approximate matching in $o(n)$ update time.

Streaming algorithms. Our streaming algorithm in Result 1 is also based on using matching covers as a natural sparsifier for matchings. The algorithm works through a series of buffers of edges $B_1, B_2, \ldots,$. The first buffer B_1 reads the edges from the input until it gets "full", i.e., receives some $o(n^2)$ edges (which is some constant factor larger than the size of our matching cover). At that point we compute a matching cover of the edges in the buffer using an offline/non-streaming algorithm and send its edges to the buffer B_2; then, we "restart" B_1 by emptying all its current edges and letting it collect more edges from the stream. The same approach is repeated across all other buffers as well. The number of these buffers can be bounded as only a constant fraction of edges in one buffer can make their way to the next one, eventually reaching a buffer that never gets full. This also implies that fewer edges will be be further "sparsified" in each matching cover, thus the error occurred due to the approximation guarantee of the matching cover does not get amplified "too much". Thus, using this algorithm along with our matching cover algorithm for regularity lemma, leads to an $o(n^2)$-space $(1 - o(1))$-approximation algorithm for single-pass streaming matchings.

The strategy we outlined above works for *any* choice of matching cover (as long as we can compute it in a small space offline). Thus, we can alternatively implement the matching cover subroutine by simply enumerating all subgraphs of the input (in exponential time) and the *optimal* one. An argument due to Goel et al. [50]—extended in our paper to general graphs—shows that density of optimal matching covers can be bounded by the density of certain RS graphs. To obtain Result 3, we also need to turn the additive approximation guarantee of the matching cover into a multiplicative bound. This is done using vertex-sparsification methods of Assadi et al. [16] and Chitnis et al. [39] (as specified in [15]) that reduce the number of vertices in the graph down to its maximum matching size without reducing the matching size by much. This turns the additive guarantee of the matching cover into a multiplicative one, giving us Result 3 as well.

Finally, one key step in making the above algorithms work is to store the $o(n^2)$ edges they have in the buffers more efficiently than

spending $\Theta(\log n)$ bits per each (which is prohibitive for us given the extremely small improvement in the space the algorithms get over the trivial $O(n^2)$ bound). This is done by storing the edges via the succinct dynamic dictionary of Raman and Rao [75] (see Section 3.4) and then performing all the computation in this compressed space instead.

3 PRELIMINARIES

Notation. For any integer $t \geqslant s \geqslant 1$, we let $[t] := \{1, \ldots, t\}$ and let $[s, t] = \{s, \ldots, t\}$. We use the term with high probability, abbreviated w.h.p., to imply probability at least $1 - 1/n^c$ for any desirably large constant $c \geqslant 1$ (that might affect the hidden constants in our statements).

For a graph $G = (V, E)$, we use $V(G) = V$ to denote the set of vertices and $E(G) = E$ to denote the edges. For any subsets of edges $F \subseteq E$ and disjoint subsets of vertices $X, Y \subseteq V$, we use $X(F)$ and $Y(F)$ to denote the edges of F incident on X and Y, respectively, and $F(X, Y)$ to denote the edges of F going between X and Y. Similarly, we use $G[X]$ and $G[X, Y]$ to respectively denote the subgraph of G induced on vertices X, and the bipartite subgraph of G between vertices X and Y. For any $p \in [0, 1]$, we use $G[p]$ to denote a random subgraph of G that includes each edge of G independently with probability p.

For any graph G, $\mu(G)$ denotes the size of the maximum matching in G. We have,

Fact 3.1. *Any graph G has at most $2n \cdot \mu(G)$ edges.*

The proof of Fact 3.1 is simply based on picking an arbitrary edge of the graph and adding to a matching, removing at most $2n$ edges incident on this edge, and repeating until the graph is empty.

We will also need the following version of Hall's theorem.

Proposition 3.2 (Extended Hall's marriage theorem; cf. [54])**.** *Let $G = (L, R, E)$ be any bipartite graph with $|L| = |R| = n$. Then $\max(|A| - |N_G(A)|) = n - \mu(G)$, where A ranges over all subsets of L and R, and $N_G(A)$ denotes the neighbors of A in G.*

3.1 Szemerédi's Regularity Lemma

Szemerédi's Regularity Lemma [80] is a powerful tool in extremal combinatorics. Loosely speaking, it says that every dense graph can be well-approximated by a "small" collection of random-like subgraphs. To formally state the lemma, we need a few definitions.

Let $G = (V, E)$ be any given graph, and $A, B \subseteq V$ be any disjoint vertex subsets. We write $e(A, B)$ to denote the number of edges between A, B. If $A, B \neq \emptyset$, we define the **density** of edges between A and B by:

$$d(A, B) := \frac{e(A, B)}{|A||B|}.$$

For a parameter $\gamma \in (0, 1)$, we say (A, B) is γ**-regular** if for every $X \subseteq A$ and $Y \subseteq B$ satisfying $|X| \geqslant \gamma \cdot |A|$ and $|Y| \geqslant \gamma \cdot |B|$, we have $|d(A, B) - d(X, Y)| < \gamma$.

Let $C_0, C_1, \ldots, C_k$ be a partition of the vertex set V. We say this partition is **equitable** if the classes $C_1, \ldots, C_k$ all have the same size. We will call C_0 the **exceptional class**. We say this partition is γ-regular if both of the following statements are true:

(1) It is equitable and $|C_0| \leqslant \gamma n$.
(2) All but at most $\gamma \binom{k}{2}$ of the pairs C_i, C_j for $1 \leqslant i < j \leqslant k$ are γ-regular.

Instead of the original formulation of Szemerédi's Regularity Lemma in [80], we state an algorithmic version of it due to Alon et al. [5].

Proposition 3.3 ([5])**.** *There exists a function $Q : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ satisfying $\log^* Q(x, y) \leqslant \text{poly}(x, y)$ for all x, y, such that, given any n-vertex graph $G = (V, E)$ and $\gamma \in (0, 1), t \geqslant 1$, one can find in $n^\omega \cdot Q(t, 1/\gamma)$ time a γ-regular partition of V into $k + 1$ classes such that $t \leqslant k \leqslant Q(t, 1/\gamma)$.*

The algorithm in Proposition 3.3 can also be implemented in a space-efficient manner (which is needed for our streaming algorithms). The proof is deferred to the full version.

Proposition 3.4. *Given query access to the adjacency matrix, the algorithm in Proposition 3.3 can be implemented in $O(n \cdot Q(t, 1/\gamma) \log n)$ space and $\text{poly}(n, Q(t, 1/\gamma))$ time.*

3.2 Fox's Triangle Removal Lemma

Similar to the Regularity Lemma, the Triangle Removal Lemma is another highly useful tool in extremal combinatorics. While original proofs of this lemma were based on the regularity lemma, Fox [47] presented a proof that bypasses regularity lemma and thus obtains stronger bounds. We will use this result also in one of our streaming algorithms.

Proposition 3.5 ([47])**.** *There exists an absolute constant $b > 1$ such that the following is true. For any $\gamma \in (0, 1)$ let $\delta := \delta(\gamma)$ be inverse of the tower of twos of height $b \cdot \log(1/\gamma)$, i.e., $\delta^{-1} = 2 \uparrow\uparrow b \cdot \log(1/\gamma)$. Then, any n-vertex graph with at most $\delta \cdot n^3$ triangles can be made triangle-free by removing at most $\gamma \cdot n^2$ edges.*

3.3 Ruzsa-Szemerédi Graphs

A matching M in a graph G is called an **induced matching** iff the subgraph of G induced on vertices of M only contains the edges of M itself; in other words, there are no other edges between the vertices of this matching.

Definition 3.6. *For integers $r, t \geqslant 1$, a graph $G = (V, E)$ is called an (r, t)-*Ruzsa-Szemerédi graph *(RS graph for short) iff its edge-set E can be partitioned into t induced matchings $M_1, \ldots, M_t$, each of size r. For any integer $n \geqslant 1$ and parameter $\beta \in (0, 1/2)$, we use $\text{RS}(n, \beta)$ to denote the maximum number of edges in any n-vertex RS graph with induced matchings of size $\beta \cdot n$.*

RS graphs have been extensively studied as they arise naturally in property testing, PCP constructions, additive combinatorics, streaming algorithms, graph sparsification, etc. (see, e.g., [4, 6, 7, 12, 35, 45, 49, 50, 55, 61, 81]). In particular, a line of work initiated by Goel et al. [50] have used different constructions of these graphs to prove communication complexity and streaming lower bounds for graph streaming algorithms [11, 14, 16, 18, 38, 41, 50, 59, 60, 65]. In this

work however, we shall use them as an upper bound tool. The only other upper bound application of these graphs in a similar context that we are aware of is the communication protocols of [50]: they show that to obtain a one-way communication protocol for $\varepsilon \cdot n$-additive approximation of matchings, roughly $O(\text{RS}(n, \varepsilon))$ communication is sufficient and also necessary.

We establish a simple property of the $\text{RS}(n, \beta)$ function in Definition 3.6 that relates density of different RS graphs with similar parameters (see full version for the proof).

Claim 3.7. *For any integer $n \geqslant 1$ and real $0 < \beta < 1$, $\text{RS}(2n, 3\beta) \leqslant O(1) \cdot \text{RS}(n, \beta)$.*

3.4 Succinct Dynamic Dictionaries

We need to use succinct dynamic dictionaries from prior work in [36, 74, 75]. For concreteness, we use the construction of [75] although the other ones work as fine also for us.

Proposition 3.8 (c.f. [75])**.** *There exists a dynamic data structure $\mathbf{D}$ for maintaining a subset S of size at most s from a universe U of size u that supports the following operations:*

- $\mathbf{D}.\texttt{insert}(a)$*: Inserts an element $a \in U$ to the set S;*
- $\mathbf{D}.\texttt{member}(a)$*: Returns whether the given element $a \in U$ belongs to U or not;*

The data structure requires $(1 + o(1)) \cdot \log \binom{u}{s}$ bits of space to store S and answers each query in $O(1)$ amortized expected time or $O(s)$ worst-case deterministic time.

4 A MATCHING COVER VIA REGULARITY LEMMA

In this section, we give a polynomial time algorithm for constructing an *matching cover* of size $o(n^2)$. We use the algorithm of this section both in the streaming model and the dynamic model. Most of the proofs in this section are deferred to the full version.

Let us start by formally defining matching covers.

Definition 4.1 ([50])**.** *A subgraph H of an n-vertex graph G is an α-matching cover of G if for any disjoint subsets of vertices (A, B) in G, we have $\mu(H[A, B]) \geqslant \mu(G[A, B]) - \alpha \cdot n$.*

The following theorem is our main result of this section.

Theorem 1. *Given any n-vertex graph G, for some $\alpha = (\log^* n)^{-\Omega(1)}$, there is an $O(n^{\omega} \log n)$ time algorithm, which is formalized below as Algorithm 1, for finding an α-matching cover H of G with at most $n^2/(\log^* n)^{\Omega(1)}$ edges.*

Even though existence of $o(n^2)$ size $o(1)$-matching covers for bipartite graphs was already proved by Goel et al. [50], it was not known whether it is possible to find one in polynomial time (nor whether they also exist for general, not necessarily bipartite, graphs).

4.1 First Step: A Hitting Set Argument

In this section, we give an algorithm for finding an *α-hitting set*, defined below. We later show in Section 4.2 that this can be turned into a matching cover.

Definition 4.2. *A subgraph H of an n-vertex graph G is an α-hitting set of G if for any disjoint subsets of vertices (A, B) in G satisfying $|A| = |B| = \alpha n$ and $\mu(G[A, B]) = \alpha n$, there is at least one edge between A and B in H.*

The following lemma is our main guarantee of this section.

Lemma 4.3. *Given any n-vertex graph G, for some $\alpha = (\log^* n)^{-\Omega(1)}$, there is an $O(n^{\omega} \log n)$ time algorithm, formalized below as Algorithm 1, for finding an α-hitting set H of G with at most $n^2/(\log^* n)^{\Omega(1)}$ edges.*

It is not hard to see that Algorithm 1 outputs a subgraph with $O(\gamma n^2) = o(n^2)$ edges, since essentially the dense parts of the decomposition are subsampled and there are 'few' other edges in the graph. The following claim formalizes this.

Claim 4.4. *The output F of Algorithm 1, w.h.p., has at most $O(\gamma n^2)$ edges.*

The harder part of the proof, is to show that the sparse subgraph returned by Algorithm 1 is indeed a matching cover. We continue with the following claim.

Claim 4.5. *W.h.p., it holds for all $X \subseteq C_i$, $Y \subseteq C_j$ such that (C_i, C_j) a good pair, $|X| \geqslant \gamma |C_i|$, and $|Y| \geqslant \gamma |C_j|$ that $|F_3(X, Y)| > n^2/\log^6 n$.*

Next, we prove the following lemma on *consolidating the support* of an arbitrary fractional matching so that each non-zero variable takes a sufficiently large value.

Algorithm 1. The construction of the matching cover for Theorem 1.

Let $t \leftarrow (\log^* n)^{\delta}$, $\gamma \leftarrow (\log^* n)^{-\delta}$ for some constant $\delta \in (0, 1)$ such that $Q(t, 1/\gamma) \leqslant \log n$.

(i) Run the algorithm in Proposition 3.3 to obtain a γ-regular partition $C_0, \ldots, C_k$ with $t = (\log^* n)^{\delta} \leqslant k \leqslant Q(t, 1/\gamma) \leqslant \log n$.

(ii) Let (C_i, C_j) for $i \neq j$. We say (C_i, C_j) is *good* if (i) $i, j \neq 0$, (ii) it is γ-regular, and (iii) $d(C_i, C_j) \geqslant 8\gamma$. Otherwise, we say (C_i, C_j) is *bad*.

(iii) Let $F \subseteq E$ be a subset of edges obtained by $F := F_1 \cup F_2 \cup F_3$ where
  (a) F_1 contains the edges between the bad pairs; i.e. $F_1 := \bigcup_{\text{bad } C_i, C_j} E \cap (C_i \times C_j)$.
  (b) F_2 contains the edges within each class; i.e. $F_2 := \bigcup_{0 \leqslant i \leqslant k} E \cap (C_i \times C_i)$.
  (c) F_3 is obtained by sampling the edges between good pairs with probability $p := \frac{10}{\log n}$; i.e. $F_3 = \bigcup_{\text{good } C_i, C_j} (E \cap (C_i \times C_j))[p]$.

(iv) Return F.

Lemma 4.6. *Let* $\mathbf{x}$ *be any fractional matching (not necessarily in the matching polytope). For any* $\varepsilon \in (0,1]$*, there is a fractional matching* $\mathbf{y}$ *such that all the following hold:*

(1) For any vertex v*,* $y_v \leqslant x_v$*, where here* $y_v := \sum_{e \ni v} y_e$ *and* $x_v := \sum_{e \ni v} x_e$*.*

(2) $\mathrm{supp}(\mathbf{y}) \subseteq \mathrm{supp}(\mathbf{x})$*. That is, if* $y_e > 0$ *for some edge* e*, then* $x_e > 0$*.*

(3) For any edge e*, either* $y_e = 0$ *or* $y_e \geqslant \frac{\varepsilon^3}{12\ln(1/\varepsilon)}$*.*

(4) $|\mathbf{y}| \geqslant |\mathbf{x}| - 2\varepsilon n$*, where here* $|\mathbf{y}| := \sum_e y_e$ *and* $|\mathbf{x}| := \sum_e x_e$*.*

We are now ready to prove that Algorithm 1 returns a $o(1)$-matching-cover w.h.p.

Lemma 4.7. *The output of Algorithm 1 is, w.h.p., an* α*-hitting set of* G *for* $\alpha = \Theta((\gamma \log(1/\gamma))^{1/3}) = (\log^* n)^{-\Omega(1)}$*.*

4.2 Second Step: From Hitting Set to Matching Cover

We now prove that any subgraph satisfying the hitting set requirement (Definition 4.2) is also a matching cover (Definition 4.1). This will follow from Hall's theorem (Proposition 3.2), and the proof similar to that of Lemma 9.3 in [50].

Lemma 4.8 (From Hitting Set to Matching Cover)**.** *Let* $G = (V, E)$ *be any graph that is not necessarily bipartite. Then any subgraph* H *of* G *that is an* α*-hitting set is also an* α*-matching cover of* G*.*

We are now ready to prove Theorem 1.

Proof of Theorem 1. The output of Algorithm 1 being an $o(1)$-hitting set was proved in Lemma 4.7. By Lemma 4.8, the output subgraph is an $o(1)$-matching cover. This matching cover having at most $O(\gamma n^2) = n^2/(\log^* n)^{\Omega(1)}$ edges was proved in Claim 4.4. Finally, the running time follows from the algorithm of Proposition 3.3 for finding the regularity decomposition, and the fact that $Q(t, 1/\gamma) \leqslant \log n$ in Algorithm 1. □

5 A FULLY DYNAMIC ALGORITHM VIA MATCHING COVERS

In this section, we show that the matching cover of Section 4 can be used to prove the following result in the fully dynamic model.

Theorem 2. *There is a randomized, fully dynamic algorithm that maintains with high probability a* $(1 - o(1))$*-approximate matching under (possibly adversarial) edge updates. The algorithm has initialization time* $O(n^\omega \log n)$ *and worst-case update time* $n/(\log^* n)^{\Omega(1)}$*.*

We start by giving an overview of our algorithm. We first describe a strategy that enables us to maintain an approximate matching with additive error $o(n)$, and latter explain how to make the approximation guarantee multiplicative. We re-compute an $o(1)$-matching cover of the current graph every $\Theta(n^{\omega-1}\log^2 n)$ updates, and then pretend as if the matching cover is the entire graph, and use the $O(\sqrt{m})$ update-time algorithm of Gupta and Peng [53], stated below as Proposition 5.1, to maintain a nearly optimal matching.

Algorithm 2. A fully dynamic algorithm for Theorem 2.

Input: An n-vertex fully dynamic graph G subject to edge insertions and deletions.

Output: A (dynamically changing) $(1-\varepsilon)$-approximate maximum of G.

Parameters: We set t, γ, δ as in Algorithm 1, and set $\varepsilon \leftarrow 10(\log^* n)^{-\delta/64}$.

Sparse regime:

(1) Whenever the number of edges in G exceeds $n^2/(\log^* n)^{\delta/8}$, switch to the dense regime.
(2) Use Proposition 5.1 on the whole graph G to maintain a $(1-\varepsilon)$-approximation.

Dense regime:

(1) Whenever the number of edges in graph G falls below $n^2/(2(\log^* n)^{\delta/8})$, restart the algorithm of Proposition 5.1 on the whole graph G to maintain a $(1-\varepsilon)$-approximate matching of it, and switch to the sparse regime.
(2) Do the following every $n^{\omega-1}\log^2 n$ updates (including before the first update):
  (a) Use Algorithm 1 to construct an $o(1)$-matching cover F of the graph G in $O(n^\omega \log n)$ time (see Theorem 1).
  (b) Restart the algorithm of Proposition 5.1 for maintaining a $(1-\varepsilon)$-approximation of subgraph F.
(3) Upon insertion of an edge e, let $F \leftarrow F \cup \{e\}$ and trigger an edge insertion to the algorithm of Proposition 5.1 we use on F.
(4) Upon deletion of an edge e, if $e \in F$, let $F \leftarrow F - \{e\}$ and trigger an edge deletion to the algorithm of Proposition 5.1 we use on F; otherwise ignore the deletion.

First, it is easy to see that the amortized update time of this strategy is $o(n)$, as the computation time $O(n^\omega \log n)$ of the matching cover gets amortized over $\Theta(n^{\omega-1}\log^2 n)$ updates to $o(n)$, and the number of edges in the matching cover is $o(n^2)$. Then to argue the correctness, we have to show that the matching cover found by our offline algorithm is *robust to edge updates* - that is, it remains an $o(1)$-matching cover throughout the following $\Theta(n^{\omega-1}\log^2 n)$ updates. This is indeed a feature of our offline algorithm: in particular, the number of edges between each pair of large enough $X \subseteq C_i, Y \subseteq C_j$ for dense, regular C_i, C_j pairs is $\tilde{\Omega}(n^2)$, which means that the hitting set property will be preserved as long as $\ll n^2$ edges have been deleted, and as a result the subgraph obtained by our algorithm remains an $o(1)$-matching cover throughout the following $\Theta(n^{\omega-1}\log^2 n) \ll n^2$ updates, as desired.

To turn the additive approximation guarantee to a multiplicative one, we will deal with "sparse" and "dense" regimes separately. Specifically, when the number of edges is at most $n^2/(\log^* n)^{\Omega(1)}$, we simply use the Gupta-Peng algorithm to maintain a $(1-\varepsilon)$-approximation in $n/(\log^* n)^{\Omega(1)}$ update time. On the other hand, when the graph is dense, we first use the matching cover of Theorem 1 to sparsify the graph while preserving its maximum matching,

then run Proposition 5.1 on this sparse graph to maintain a $(1-\varepsilon)$-approximate matching of it in $n/(\log^* n)^{\Omega(1)}$ update-time. We also set up a "buffer zone" in the thresholds for switching between the two algorithms so that we do not pay the switching overhead too often.

We now present our algorithm. Here, we only show an algorithm with initialization time $O(n^\omega \log n)$ and *amortized* update time $n/(\log^* n)^{\Omega(1)}$, which we believe suffices to demonstrate the main idea. We defer the discussion on how to make the update time worst-case in the full version.

Proposition 5.1 ([53]). *There is a deterministic, fully dynamic algorithm for maintaining a $(1-\varepsilon)$-approximate matching with initialization time $O(m_0\varepsilon^{-1})$ and worst-case update time $O(\sqrt{m}\varepsilon^{-2})$, where m_0 is the number of edges in the initial graph, and m is the maximum number of edges in the graph throughout the updates.*

Our algorithm is formally presented in Algorithm 2. We now turn to analyze Algorithm 2. First, we prove that it has our desired update-time via amortization. As discuss, we will later show how the algorithm can be deamortized. The proof is deferred to the full version.

Claim 5.2. *The amortized update-time of Algorithm 2 is $n/(\log^* n)^{\Omega(1)}$.*

Next, we prove that Algorithm 2 maintains a $(1-o(1))$-approximate matching w.h.p.

Claim 5.3. *At any point, the output of Algorithm 2 is w.h.p. a $(1-o(1))$-approximate maximum matching of G. This holds, in particular, against an adaptive adversary that is aware of both the output and the state of the algorithm.*

Proof. For the sparse regime, this directly follows from the correctness of Proposition 5.1 since we run it on the entire graph G. We thus focus on the dense regime.

First, note that in the dense regime there are at least $m \geqslant n^2/(2(\log^* n)^{\delta/8})$ edges in the graph. Observe that any n-vertex m-edge graph has a matching of size at least $m/(2n-1)$: iteratively pick an arbitrary free edge, add it to the matching, and remove its endpoints from the graph; each step only removes at most $(2n-1)$ edges, thus the matching must have size at least $m/(2n-1)$. From this, we get that whenever the algorithm is in the dense regime, there is a matching of size at least $\mu(G) \geqslant n/(4(\log^* n)^{\delta/8})$ in it.

Next, we claim that at any point in the dense regime, F is an α-matching cover of G (Definition 4.1), where as defined in Lemma 4.7,

$$\begin{aligned}\alpha =&\Theta((\gamma \log(1/\gamma))^{1/3}) = \Theta((\log^* n)^{-\delta} \log((\log^* n)^{\delta}))^{1/3})\\ =&O((\log^* n)^{-\delta/4}).\end{aligned}$$

By Lemma 4.8, to show this, it suffices to show that F is an α-hitting set of G (Definition 4.2) at any point in the dense regime. To see this, observe that immediately after we call Algorithm 1, F must be an α-hitting set of G simply by the guarantee of Theorem 1. However, for the next $n^{\omega-1}\log^2 n$ updates until we re-run Algorithm 1, both the graph G and F change due to the updates to the graph. Observe that edge insertions cause no problem since any edge added will be added to F as well. But edge deletions may cause a problem. In particular, recall that we subsample $o(1)$ fraction of edges of the good pairs in Algorithm 1, and if they are all removed then we no longer have an α-hitting set. Indeed, given that the adaptive adversary is aware of this sampled subset, he can attempt to remove these edges one by one. The crucial observation, here, is that right after we call Algorithm 1, Claim 4.5 guarantees that there are w.h.p. at least $|F_3(X,Y)| \geqslant n^2/\log^6 n$ subsampled edges between any two large enough subsets $X \subseteq C_i, Y \subseteq C_j$ of any good pair (C_i, C_j). On the other hand, our guarantee of Theorem 1 that F is an α-hitting set only requires $|F_3(X,Y)| > 0$. As a result, even if the adversary attempts to remove edges of F_3 one by one within the next $n^{\omega-1}\log^2 n \ll n^2/\log^6 n$ updates, $F_3(X,Y)$ will remain non-empty and so F remains an α-hitting set.

Moreover, since F is an α-matching cover of G, we get from Definition 4.1, taking $M^\star$ to be an arbitrary maximum matching of G and taking sets A and B to each include one endpoint of each edge in $M^\star$ arbitrarily, we get that

$$\mu(F) \geqslant \mu(F[A,B]) \geqslant \mu(G[A,B]) - \alpha n = |M^\star| - \alpha n = \mu(G) - \alpha n.$$

Putting together the bounds above, we get that at any point during the updates in the dense regime, F includes a matching of size at least

$$\mu(F) \geqslant \mu(G) - \alpha n = \mu(G) - O(n/(\log^* n)^{\delta/4}) \geqslant (1-o(1))\mu(G),$$

where the last equality holds since $\mu(G) \geqslant \Omega(n/(\log^* n)^{\delta/8})$ as discussed above. Running the algorithm of Proposition 5.1 on top of this, we maintain a $(1-\varepsilon)(1-o(1))\mu(G) = (1-o(1))\mu(G)$ size matching overall. □

6 SINGLE-PASS STREAMING ALGORITHMS

We prove Result 1 and Result 3 in this section. Both algorithms rely on using matching covers iteratively in the same way and differ primarily on how they compute matching covers and some additional steps. Because of this, we first present and prove a generic result that uses matching covers in a blackbox way to obtain a streaming algorithm for finding matching covers and then extend it separately to obtain for Result 1 and Result 3. When presenting our results, we focus more on the correctness of our algorithms, but defer proofs of the space usage to the full version.

6.1 A Streaming Algorithm for Matching Covers

We present an algorithm that computes the matching cover of a graph presented in a stream by iteratively computing matching covers of smaller subsets of the stream without losing "much" on the quality of the final matching cover. For technical reasons that will become clear later, we need this algorithm to work for multi-graphs as well.

Proposition 6.1. *For any integer $k \geqslant 1$ and any $\alpha \in (0, 1/10)$, there exists a single-pass streaming algorithm that computes an α-matching cover of n-vertex multi-graphs with at most m edges in space*

$$O\Big(\frac{m}{k} \cdot \log\Big(\frac{n^2 \cdot k}{m}\Big) + \mathsf{MC}(n, \alpha/2k) \cdot \log\Big(\frac{n^2}{\mathsf{MC}(n,\alpha/2k)}\Big) \cdot \log k\Big);$$

here, we assume we are given a subroutine Matching-Cover *that given adjacency matrix access to any n-vertex graph with m/k edges,*

can compute an $(\alpha/2k)$-matching cover with $\mathsf{MC}(n, \alpha/2k) \leqslant m/2k$ edges in $O((m/k) \cdot \log(n^2 \cdot k/m))$ space. The streaming algorithm requires calling Matching-Cover *$O(k)$ times and is deterministic as long as the* Matching-Cover *subroutine is deterministic.*

The algorithm in Proposition 6.1 is based on a novel use and modification of the widely used "Merge and Reduce" technique in the streaming literature (used previously e.g., for quantile estimation [62, 69] or cut/spectral sparsifiers [71]). We give a high level overview of the algorithm here and present the formal description in Algorithm 3.

The algorithm maintains $t := O(\log k)$ different buffers $B_1, \ldots, B_t$ of edges throughout the stream (all these buffers store their edges using the succinct dynamic dictionary of Proposition 3.8 to save space). Buffer B_1 simply starts reading edges from the stream until it collects m/k edges; it will then use the (offline) subroutine Matching-Cover over these edges with parameter $\alpha' = \alpha/2k$ to obtain an α'-matching cover of the subgraph of input on edges in B_1. Edges of this matching cover are then inserted to buffer B_2 and we empty buffer B_1, which will continue reading edges from the stream again. In the mean time, whenever buffer B_2 gets "full", this time meaning that it receives twice as many edges as $\mathsf{MC}(n, \alpha')$, we compute another α'-matching cover using Matching-Cover, this time over the edges in B_2, pass them to buffer B_3, and empty B_2 which continues receiving edges from buffer B_1. This process is done the same way across all buffers until all edges of the stream have passed (we prove buffer B_t never gets full so not having a buffer B_{t+1} is not a problem). At the end, we argue that the edges that are remained across all buffers $B_1, \ldots, B_t$ at the end of the stream form an α-matching cover of the input.

Algorithm 3. An algorithm for Proposition 6.1.

Input: A multi-graph $G = (V, E)$ in the stream with n edges and at most m edges. We are also given integer $k \geqslant 1$ and approximation parameter $\alpha > 0$, and access to the (offline) subroutine Matching-Cover as specified in Proposition 6.1.

Output: An α-matching cover of G.

Parameters: We set $t := (\log k + 2)$ and $\alpha' := \alpha/2k$.

(i) Maintain the following **buffers** of edges $B_1, \ldots, B_t$ using succinct dynamic dictionary of Proposition 3.8 (we specify the details in Lemma 6.2):
  (a) Buffer B_1: add any arriving edge (u, v) arrives in the stream to B_1. Once size of B_1 reaches m/k, run Matching-Cover to find an α'-matching-cover of the subgraph (V, B_1) of G and add all those edges to B_2. Restart B_1 by deleting all its current edges.
  (b) Buffers B_i for $i > 1$: once size of B_i reaches $2 \cdot \mathsf{MC}(n, \alpha')$, run Matching-Cover to find an α'-matching-cover of the subgraph (V, B_i) of G and add all those edges to B_{i+1}[a]. Restart B_i by deleting all its current edges.

(ii) Return $(B_1 \cup \ldots \cup B_t)$ at the end of the stream.

[a] We will show in Claim 6.3 that this step never happens for buffer B_t, namely, it never gets "full", and thus the algorithm is well-defined even though there is no buffer B_{t+1}.

The analysis of the algorithm involves showing that: (i) fewer and fewer edges find their way to higher-indexed buffers, (ii) the repeated application of Matching-Cover does not blow up the approximation guarantee by too much, and (iii) all this can be implemented in a relatively small space. We now present the formal algorithm and its analysis.

We start by analyzing the space complexity of Algorithm 3.

Lemma 6.2. *Algorithm 3 can be implemented in space of*

$$O\left(\frac{m}{k} \cdot \log\left(\frac{n^2 \cdot k}{m}\right) + t \cdot \mathsf{MC}(n, \alpha') \cdot \log\left(\frac{n^2}{\mathsf{MC}(n, \alpha')}\right)\right).$$

We now prove the correctness of Algorithm 3. To do so, we need the following definitions:

- Let $H_1^1, \ldots, H_{k_1}^1$ denote the k_1 separate matching covers constructed by the algorithm over the edges of buffer B_1, one for each time that we restart B_1. Let $G^2 := H_1^1 \cup \ldots \cup H_{k_1}^1$ denote the graph that is sent to buffer B_2 throughout the algorithm (for notational convenience, we also define $G^1 = G$ as the input graph, namely, the graph that is sent to buffer B^1).
- For any $i \in [2 : t-1]$, similarly, let $H_1^i, \ldots, H_{k_i}^i$ denote the k_i separate matching covers constructed by the algorithm over the edges of buffer B_i. Let $G^{i+1} := H_1^i \cup \ldots \cup H_{k_i}^i$ denote the graph that is sent to buffer B_{i+1} throughout the algorithm.

We claim that the number of subgraphs at buffer B_i drops by a factor of 2^i compared to B_1, and defer the proof to the full version.

Claim 6.3. *For any $i \in [t-1]$, $k_i \leqslant k/2^{i-1}$ and $k_t = 0$ meaning that bucket B_t never generates a matching cover (namely, it never gets full).*

The following lemma captures the loss on the size of maximum matching that the algorithm maintains from one buffer to the next one. In other words, the cost we have to pay for introduction of each level of buffers.

Lemma 6.4. *For any $i \in [t-1]$ and any disjoint subsets of vertices $X, Y \subseteq V$,*

$$\mu\left((G^{i+1} \cup B_i^f \cup \ldots \cup B_1^f)[X, Y]\right) \geqslant \mu\left((G^i \cup B_{i-1}^f \cup \ldots \cup B_1^f)[X, Y]\right) - k_i \cdot \alpha' \cdot n.$$

where B_j^f for $j \in [t]$ is the final content of the buffer at the end of the stream.

Proof. Fix any $i \in [t-1]$ and a maximum matching M_i^* of $(G^i \cup B_{i-1}^f \cup \ldots \cup B_1^f)[X, Y]$. We construct a matching M_{i+1} in $(G^{i+1} \cup B_i^f \cup \ldots \cup B_1^f)[X, Y]$ such that $|M_{i+1}| \geqslant |M_i^*| - k_i \cdot \delta \cdot n$. This will then immediately implies the lemma. To continue we need some more definition.

For any H_j^i for $j \in [k_i]$, let B_j^i denote the content of buffer B_i when the algorithm creates H_j^i. This way, H_j^i is a matching-cover of (V, B_j^i). Moreover, $B_1^i, \ldots, B_{k_i}^i$ together with B_i^f partition all the edges that are ever sent to buffer B_i, namely, the graph G^i. These edges are also further disjoint from $B_{i-1}^f, \ldots, B_1^f$ since the latter set

of edges were never sent to buffer B_i. We can partition the edges of M_i^* between these sets and along the way define our matching M_{i+1} as well:

- For any $j \in [k_i]$, let $M_{i,j}^* := M_i^* \cap B_j^i$ and $M_{i,j}$ be the maximum matching in H_j^i between $X(M_{i,j}^*)$ and $Y(M_{i,j}^*)$.
- For any $i' \in [i]$, let $M_{i'}^{*,f} := M_i^* \cap B_i^f$ and $M_{i'}^f := M_{i'}^{*,f}$ which is between $X(M_{i'}^{*,f})$, $Y(M_{i'}^{*,f})$.
- Define $M_{i+1} := M_{i,1}^* \cup \cdots \cup M_{i,k_i}^* \cup M_i^f \cup \cdots \cup M_i^f$.

We note that M_{i+1} is a matching between X and Y because the sets of vertices $X(M_{i,j}^*)$ and $X(M_{i,j}^*)$ for $j \in [k_i]$, as well as $X(M_{i'}^{*,f})$ and $Y(M_{i'}^{*,f})$ for $i' \in [i]$ are all disjoint given they are defined with respect to a fixed matching M_i^* over disjoint sets of edges. Moreover, M_{i+1} belongs to $(G^{i+1} \cup B_i^f \cup \ldots \cup B_1^f)[X, Y]$ as H_j^i is part of G^{i+1} for $j \in [k_i]$. It thus only remains to bound the size of M_{i+1}.

For all $i' \in [i]$, $M_{i'}^f$ and $M_{i'}^{*,f}$ are the same so there is nothing to do here. For $j \in [k_i]$, we have,

$$\begin{aligned}
|M_{i,j}| =& \mu\left(H_j^i[X(M_{i,j}^*), Y(M_{i,j}^*)]\right) \\
\geqslant& \mu\left(B_j^i[X(M_{i,j}^*), Y(M_{i,j}^*)]\right) - \alpha' \cdot n \\
& \text{(as } H_j^i \text{ is a } \alpha'\text{-matching-cover of } B_j^i \text{ and by Definition 4.1)} \\
=& |M_{i,j}^*| - \alpha \cdot n. \\
& \text{(as } M_{i,j}^* \text{ is a perfect matching in } B_j^i \text{ between } X(M_{i,j}^*) \text{ and } Y(M_{i,j}^*))
\end{aligned}$$

Thus,

$$\begin{aligned}
|M_{i+1}| &= \sum_{j=1}^{k_i} |M_{i,j}| + \sum_{i'=1}^{i} |M_{i'}^f| \\
&\geqslant \sum_{j=1}^{k_i} (|M_{i,j}^*| - \alpha' \cdot n) + \sum_{i'=1}^{i} |M_{i'}^{*,f}| \\
&= |M_i^*| - k_i \cdot \alpha' \cdot n,
\end{aligned}$$

concluding the proof. □

We can now conclude the bound on the approximation ratio of the algorithm.

Lemma 6.5. *Algorithm 3 outputs an α-matching cover of any input multi-graph G.*

Proof. Recall that for every $i \in [t]$, B_i^f denotes the final content of the buffer B_i. Moreover by Claim 6.3, buffer B_t never gets full and thus $B_t^f = G^t$. Finally, the algorithm returns $H := (B_1^f, \ldots, B_t^f)$. Fix any disjoint sets of vertices $X, Y \subseteq V(G)$. We have,

$$\begin{aligned}
\mu(H[X, Y]) &= \mu\left((B_t^f \cup B_{t-1}^f \cup \ldots \cup B_1^f)[X, Y]\right) \\
& \qquad \text{(by the definition of } H) \\
&= \mu\left((G^t \cup B_{t-1}^f \cup \ldots \cup B_1^f)[X, Y]\right) \qquad (\text{as } B_t^f = G^t) \\
&\geqslant \mu\left((G^{t-1} \cup B_{t-2}^f \cup \ldots \cup B_1^f)[X, Y]\right) - k_{t-1} \cdot \alpha' \cdot n \\
& \qquad \text{(by Lemma 6.4 for } i = t-1) \\
&\geqslant \mu(G[X, Y]) - \sum_{i=1}^{t-1} k_i \cdot \alpha' \cdot n \\
& \text{(by repeatedly applying Lemma 6.4 for all } i < t-1 \text{ and since } G^1 = G) \\
&\geqslant \mu(G) - \sum_{i=1}^{t-1} (k/2^{i-1}) \cdot \alpha' \cdot n \\
& \qquad \text{(by Claim 6.3, } k_i \leqslant k/2^{i-1}) \\
&\geqslant \mu(G) - 2k \cdot \alpha' \cdot n \\
& \qquad \text{(by the sum of the geometric series)} \\
&= \mu(G) - \alpha \cdot n. \qquad \text{(by the choice of } \alpha' = \alpha/2k)
\end{aligned}$$

This implies that for every disjoint subsets of vertices $X, Y \subseteq V(G)$, we have $\mu(H[X, Y]) \geqslant \mu(G[X, Y]) - \alpha \cdot n$, thus making H an α-matching cover of G by Definition 4.1. □

Proof of Proposition 6.1. The bound on the space complexity of the algorithm follows from Lemma 6.2 by plugging the value of $\alpha' = \alpha/2k$ and $t = \log k + 1$. The correctness follows from Lemma 6.5. Finally, Algorithm 3 is deterministic modulo any potential randomness used by Matching-Cover. □

6.2 A Streaming Matching Algorithm via Regularity Lemma

We now use Proposition 6.1 together with our Theorem 1 to formalize Result 1 as follows.

Theorem 3 (Formalization of Result 1). *There is a randomized single-pass streaming algorithm that with high probability computes a $(1 - o(1))$-approximate matching of a graph presented in a stream with adversarial order of edge arrivals in $n^2/(\log^* n)^{\Omega(1)}$ space and polynomial time.*

Proof. Note that, to apply Proposition 6.1, we need a subroutine Matching-Cover for computing an $(\alpha/2k)$-matching cover (for parameters α and k to be determined soon) on any n-vertex graph with n^2/k edges. Theorem 1 provides such an algorithm with parameters

$$(\alpha/2k) = \frac{1}{(\log^* n)^{\delta_1}} \quad \text{and} \quad \mathsf{MC}(n, \alpha/2k) = \frac{n^2}{(\log^* n)^{\delta_2}},$$

for some absolute constants $\delta_1, \delta_2 \in (0, 1)$. Let $\alpha = 1/(\log^* n)^{3\delta_1/4}$ and $k = \frac{1}{2} \cdot (\log^* n)^{\delta_1/4}$, which satisfies the conditions above. Moreover, by Proposition 3.4, we can implement Algorithm 1 of Theorem 1 in polynomial time and space $O((n^2/k) \cdot \log k) = n^2/(\log^* n)^{\Omega(1)}$, given only adjacency matrix access to its input graph. This way, by Proposition 6.1, we obtain a single-pass streaming algorithm that with high probability computes an α-matching cover in space $n^2/(\log^* n)^{\Omega(1)}$.

The main algorithm in the theorem is as follows. We store the first $2n^2/k$ edges in the stream using succinct dynamic dictionary of Proposition 3.8 in $n^2/(\log^* n)^{\Omega(1)}$ space. In parallel, we also run the algorithm mentioned above to obtain an α-matching cover of G. The space complexity and polynomial runtime of the algorithm is thus already established.

We now prove the correctness. If $\mu(G) \leqslant n/k$, then by Fact 3.1, we have stored all edges of the graph and thus at the end can

simply return a maximum matching of the stored edges; to do so, we simply run Hopcroft-Karp algorithm [56] by providing it with the adjacency matrix of the stored edges using `member` query on the succinct dynamic dictionary (which only requires $O(n \log n)$ additional space beside the input). Thus, in this case, we obtain an exact maximum matching of the input graph.

If $\mu(G) > n/k$, then we can pick X and Y in the definition of matching cover output by the algorithm of Proposition 6.1 to be the endpoints of the maximum matching of G, and have,

$$\mu(H) \geqslant \mu(G) - \alpha \cdot n \geqslant (1 - \alpha \cdot k) \cdot \mu(G) = (1 - 1/(\log^* n)^{\delta_1/2})\mu(G),$$

which is $(1 - o(1)) \cdot \mu(G)$ as desired. This concludes the proof. □

6.3 A Streaming Matching Algorithm via RS Graph Upper Bounds

We formalize Result 3 as follows in this subsection ($\mathrm{RS}(n, \beta)$ below was defined in Definition 3.6).

Theorem 4 (Formalization of Result 3). *There exists an absolute constant $\eta > 0$ such that the following is true. There is a randomized single-pass streaming algorithm that for any $1 \leqslant k \leqslant n$ and $\varepsilon \in (0, 1/100)$, with high probability, computes a $(1-\varepsilon)$-approximate matching of a graph presented in a stream with adversarial order of edge arrivals in exponential time and space*

$$O\Big(\frac{n^2}{k} \cdot \log^2 k + \mathrm{RS}(n, \eta \cdot \varepsilon^2/k) \cdot \log\left(\frac{n^2}{\mathrm{RS}(n, \eta \cdot \varepsilon^2/k)}\right) \cdot \log^2 k \cdot \log(k/\varepsilon)\Big).$$

Moreover, the algorithm can return an additive $\varepsilon \cdot n$ approximation deterministically in exponential time and space

$$O\Big(\frac{n^2}{k} \cdot \log k + \mathrm{RS}(n, \varepsilon/16k) \cdot \log\left(\frac{n^2}{\mathrm{RS}(n, \varepsilon/16k)}\right) \cdot \log k \cdot \log(k/\varepsilon)\Big).$$

Roughly speaking, by ignoring lower order terms and in asymptotic notation, Theorem 4 gives a streaming algorithm for $(1-o(1))$-approximation of matchings in a single pass with adversarial order of edge arrivals using essentially $(n^2/k + \mathrm{RS}(n, o(1/k)))$ space for any integer $k \geqslant 1$.

Before proving Theorem 4, let us present a corollary of this theorem with concrete bounds on the space by using Fox's triangle removal lemma (Proposition 3.5) to bound the RS-graph density terms in Theorem 4 (this appears to be the only known method for bounding density of RS graphs with $o(n)$-size induced matchings; moreover, we are not aware of any reference that bounds the density of the type of RS graphs we need, thus we present a proof of that here also for completeness).

Corollary 6.6. *There is a deterministic single-pass streaming algorithm that computes a $(1 - o(1))$-approximate matching of a graph presented in a stream with adversarial order of edge arrivals in $n^2/2^{\Omega(\log^* n)}$ space and exponential time.*

We defer the proof of this corollary to the full version. To continue, we need to recall some additional tools from prior work. specific specific to our algorithms in this subsection.

6.3.1 Additional Tools from Prior Work.

Matching covers via RS graphs. Goel et al. [50] showed that matching covers and RS graphs are intimately connected: on bipartite graphs, the density of best construction for either can be bounded by the density of other one for closely related parameters. We need this result for general graphs as well which follows from the result of [50] using a simple argument[6].

Proposition 6.7 (an extension of [50, Theorem 9.2] to general graphs)**.** *For any $\alpha \in (0, 1)$ and $n \geqslant 1$, there exists an α-matching cover of any n-vertex graph with number of edges bounded by*

$$\mathrm{MC}(n, \alpha) \leqslant \mathrm{RS}(n, \alpha/8) \cdot O(\log(1/\alpha)).$$

Proof. The result of [50] is formally as follows (to match the definitions in our paper, our formulation is slightly different from the statements in [50] but they are equivalent):

> [50, Theorem 9.2]: For any bipartite graph $G' = (L', R', E')$ with n vertices on *each* side and $\alpha' \in (0, 1)$, there exists a subgraph H' with $\mathrm{RS}(2n, 3\alpha'/4) \cdot O(\log(1/\alpha'))$ edges such that for any disjoint subsets of vertices $X \subseteq L'$ and $Y \subseteq R'$,
>
> $$\mu(H'[X, Y]) \geqslant \mu(G'[X, Y]) - \alpha' \cdot (2n).$$

We now use this to prove the bound for general graphs as well. Let $G = (V, E)$ be any (not necessarily bipartite) graph. Consider the bipartite double cover of G obtained by copying vertices of G twice into sets V_1 and V_2 and connecting any vertex $u_1 \in V_1$ to $v_2 \in V_2$ iff (u, v) is an edge in G. Let G' denote this graph and so G' is a bipartite graph with n vertices on each side.

Compute an α'-matching cover H' of this bipartite graph using Theorem 9.2 of [50] for parameter $\alpha' = \alpha/2$ (for α given to us in the proposition statement). Thus, H' contains $\mathrm{RS}(2n, 3\alpha'/4) \cdot O(\log(1/\alpha'))$ edges. Create a subgraph H (not necessarily bipartite) on the same vertices as G by adding the edges (u, v) to H iff either (u_1, v_2) or (v_1, u_2) was an edge in H'. This way, the number of edges in H will be at most

$$\mathrm{RS}(2n, 3\alpha'/4) \cdot O(\log(1/\alpha')) \leqslant \mathrm{RS}(n, \alpha'/4) \cdot O(\log(1/\alpha')),$$

where the inequality is by Claim 3.7 that relates density of RS graphs with similar parameters.

We now argue that H is an α-matching cover of G. Fix any disjoint subsets of vertices X, Y in G. Consider $X_1 \subseteq V_1$ and $Y_2 \subseteq V_2$ corresponding to these two subsets over vertices of G' (and H'):

$$\begin{aligned}\mu(H'[X_1, Y_2]) &\geqslant \mu(G'[X_1, Y_2]) - \alpha' \cdot (2n) \\ &\text{(by Definition 4.1 as } H' \text{ is an } \alpha\text{-matching cover of } G') \\ &\geqslant \mu(G[X, Y]) - \alpha' \cdot (2n),\end{aligned}$$

as by the construction of G' any edge $(u, v) \in G[X, Y]$ also has a copy $(u_1, v_2) \in G'[X_1, Y_2]$ and thus $\mu(G'[X_1, Y_2]) \geqslant \mu(G[X, Y])$. Moreover, since X and Y are disjoint, the endpoints of the maximum matching in $H'[X_1, Y_2]$ are disjoint from each other; thus, they are mapped to *unique* edges in H also between X and Y, implying that

$$\mu(H[X, Y]) = \mu(H'[X_1, Y_2]) \geqslant \mu(G[X, Y]) - 2\alpha' \cdot n.$$

[6] We can in fact prove this result with better bounds nearly matching those of [50] using a white-box application of the techniques in [50]; however, since the actual constants do not matter for our application in this paper, we opted for the simpler and more direct proof that uses the result of [50] in a black-box way.

Noting that $\alpha' = \alpha/2$ in the above equations, concludes the proof. □

Vertex-sparsification for matchings. We also use the reductions of Assadi et al. [16] and Chitnis et al. [39] for reducing the number of vertices while preserving maximum matching size approximately. The original versions of the reductions in these work only achieved constant probability of success and boost this to a high probability bound by applying it $\Theta(\log n)$ times in parallel. In our setting, we cannot afford this direct success amplification. Thus, we instead use the following variant proven by Assadi et al. [15] that achieves a high success probability directly.

Proposition 6.8 ([15, Lemma 3.8]; see also [16, 39])**.** *For any graph $G = (V, E)$, integer $\text{opt} \geqslant 1$, and parameter $\theta \in (0, 1)$, uniformly at random pick a function $h : V \to [8 \cdot \text{opt}/\theta]$. Consider this multi-graph $H = (V_H, E_H)$ obtained from G and h:*

- *V_H is the range of the function of h, thus $|V_H| = 8 \cdot \text{opt}/\theta$.*
- *For any edge $(u, v) \in G$, there is an edge $(h(u), h(v)) \in E_H$.*

If $\mu(G) \leqslant \text{opt}$, then,

$$\Pr_h\left(\mu(H) < (1-\theta)\cdot\mu(G)\right) \leqslant \exp\left(-\frac{\mu(G)}{4}\right).$$

6.3.2 Proof of Theorem 4. We now use these prior tools combined with our Proposition 6.1 to prove Theorem 4. Recall that Proposition 6.1 returns an α-matching cover which can only guarantee an additive approximation not a multiplicative one. Thus, we first use the vertex-sparsification of Proposition 6.8 to reduce the number of vertices in G to $O(\mu(G))$—by guessing $\mu(G)$ in geometric values—so that an additive approximation also becomes a multiplicative one. We then use Proposition 6.7 to compute the matching covers in Algorithm 3 of Proposition 6.1.

Algorithm 4. The randomized algorithm in Theorem 4.

Input: A graph $G = (V, E)$ in the stream with n edges and at most $\binom{n}{2}$ edges. We are also given integer $k \geqslant 1$ and approximation parameter $\varepsilon \in (0, 1)$ as in Theorem 4.

Output: A $(1-\varepsilon)$-approximate maximum matching of G.

(i) For $i = 1$ to $t := \log k$ iterations in parallel:
  (a) Let $\text{opt}_i := n/2^{i+1}$ and pick a hash function $h_i : V \to [32 \cdot \text{opt}_i/\varepsilon]$.
  (b) Consider the multi-graph G_i obtained from G and h_i using Proposition 6.8; each edge of G arriving in the stream can be mapped to an edge of G_i using h_i.
  (c) Run Algorithm 3 on G_i with parameters k and $\alpha = \varepsilon^2/64$ and $m = \binom{n}{2}$ to obtain an α-matching cover H_i. We use the matching cover construction of Proposition 6.7 as the subroutine Matching-Cover (as specified in Claim 6.9 below).

(ii) Store the first n^2/k edges of the stream using succinct dynamic dictionary of Proposition 3.8 as the subgraph H_0.

(iii) Return a maximum matching in $H_0 \cup H_1 \cup \ldots \cup H_t$ (specified in Claim 6.9 below).

We bound the space and approximation of Algorithm 4 in the following two claims, respectively.

Claim 6.9. *Algorithm 4 (deterministically) requires space of*

$$O\left(\frac{n^2}{k}\cdot\log^2 k + \text{RS}\left(n, \frac{\varepsilon^2}{1024k}\right)\cdot\log\left(\frac{n^2}{\text{RS}(n, \varepsilon^2/1024k)}\right)\cdot\log^2 k\cdot\log\frac{k}{\varepsilon}\right).$$

Claim 6.10. *Algorithm 4 outputs a $(1-\varepsilon)$-approximate matching with high probability.*

Proof. Suppose first that $\mu(G) \leqslant n/2k$. By Fact 3.1, G in this case has at most $2n \cdot \mu(G) \leqslant n^2/k$ edges. Thus, in step (ii) of the algorithm, we are simply storing all edges and thus the algorithm returns an exact answer.

Now suppose $\mu(G) > n/2k$. This means that there is an index $i \in [t]$ such that $\frac{n}{2^{i+1}} \leqslant \mu(G) < \frac{n}{2^i}$. For this choice of i, we have $\text{opt}_i \leqslant \mu(G) < 2\cdot\text{opt}_i$ (and $\mu(G) > n^2/2k \geqslant n/2$). By Proposition 6.8 for $\theta = \varepsilon/2$ and $\text{opt} = 2 \cdot \text{opt}_i > \mu(G)$, and $h_i : V \to [32\text{opt}_i/\varepsilon]$, we have, $\Pr_{h_i}(\mu(G_i) < (1-\varepsilon/2)\cdot\mu(G)) \leqslant \exp\left(-\frac{\mu(G)}{4}\right) \ll 1/\text{poly}(n)$, where we used that fact $32\text{opt}_i/\varepsilon = 8 \cdot \text{opt}/\theta$. We condition on the complement of this event which happens with high probability. Based on this, we further have that $n_i := |V(G_i)| = \frac{32}{\varepsilon}\cdot\text{opt}_i \leqslant \frac{32}{\varepsilon}\cdot\mu(G)$. Since H_i is an α-matching cover of G_i, by letting X and Y in Definition 4.1 to be the endpoints of the maximum matching of G_i, we have, $\mu(H_i) \geqslant \mu(G_i) - \alpha\cdot n_i \geqslant (1-\varepsilon/2)\cdot\mu(G) - (\varepsilon^2/64)\cdot\frac{32}{\varepsilon}\cdot\mu(G) = (1-\varepsilon)\cdot\mu(G)$. Thus, returning the maximum matching of H_i as part of $H_0 \cup \ldots \cup H_t$ achieves a $(1-\varepsilon)$-approximation. □

Theorem 4 for randomized case now follows from Claims 6.9 and 6.10. For the deterministic part with additive approximation guarantee, we simply forgo guessing $\mu(G)$ as well as using vertex-sparsification of Proposition 6.8 at all; instead, we just run Algorithm 3 over the entire input and use Proposition 6.7, the same way as above exactly, as the subroutine Matching-Cover for computing an α-matching cover. Since we now only need an additive $\varepsilon \cdot n$ guarantee, we can take $\alpha = \varepsilon$ directly which implies the improved bounds on the space as well.

REFERENCES

[1] Kook Jin Ahn and Sudipto Guha. 2011. Linear Programming in the Semi-streaming Model with Application to the Maximum Matching Problem. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 6756)*, Luca Aceto, Monika Henzinger, and Jirí Sgall (Eds.). Springer, 526–538.

[2] Kook Jin Ahn and Sudipto Guha. 2018. Access to Data and Number of Iterations: Dual Primal Algorithms for Maximum Matching under Resource Constraints. *ACM Trans. Parallel Comput.* 4, 4 (2018), 17:1–17:40.

[3] Josh Alman and Virginia Vassilevska Williams. 2021. A Refined Laser Method and Faster Matrix Multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, Dániel Marx (Ed.). SIAM, 522–539.

[4] Noga Alon. 2002. Testing subgraphs in large graphs. *Random Struct. Algorithms* 21, 3-4 (2002), 359–370.

[5] Noga Alon, Richard A. Duke, Hanno Lefmann, Vojtech Rödl, and Raphael Yuster. 1992. The Algorithmic Aspects of the Regularity Lemma (Extended Abstract). In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*. IEEE Computer Society, 473–481.

[6] Noga Alon, Ankur Moitra, and Benny Sudakov. 2012. Nearly complete graphs decomposable into large induced matchings and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, Howard J. Karloff and Toniann Pitassi (Eds.). ACM, 1079–1090.

[7] Noga Alon and Asaf Shapira. 2006. A Characterization of Easily Testable Induced Subgraphs. *Combinatorics, Probability & Computing* 15, 6 (2006), 791–805.

[8] Moab Arar, Shiri Chechik, Sarel Cohen, Cliff Stein, and David Wajc. 2018. Dynamic Matching: Reducing Integral Algorithms to Approximately-Maximal Fractional Algorithms. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. 7:1–7:16.

[9] Sepehr Assadi. 2022. A Two-Pass (Conditional) Lower Bound for Semi-Streaming Maximum Matching. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*. SIAM, 708–742.

[10] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. 2019. 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*. 1616–1635.

[11] Sepehr Assadi and Soheil Behnezhad. 2021. Beating Two-Thirds For Random-Order Streaming Matching. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference) (LIPIcs, Vol. 198)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:13.

[12] Sepehr Assadi and Aaron Bernstein. 2019. 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*. 11:1–11:20.

[13] Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. 2022. Semi-Streaming Bipartite Matching in Fewer Passes and Optimal Space. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, Joseph (Seffi) Naor and Niv Buchbinder (Eds.). SIAM, 627–669.

[14] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2017. The Stochastic Matching Problem: Beating Half with a Non-Adaptive Algorithm. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*. ACM, 99–116.

[15] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2019. The Stochastic Matching Problem with (Very) Few Queries. *ACM Trans. Economics and Comput.* 7, 3 (2019), 16:1–16:19.

[16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. 2016. Maximum Matchings in Dynamic Graph Streams and the Simultaneous Communication Model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 1345–1364.

[17] Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. 2021. An Auction Algorithm for Bipartite Matching in Streaming and Massively Parallel Computation Models. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, Hung Viet Le and Valerie King (Eds.). SIAM, 165–171.

[18] Sepehr Assadi and Ran Raz. 2020. Near-Quadratic Lower Bounds for Two-Pass Graph Streaming Algorithms. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, Sandy Irani (Ed.). IEEE, 342–353.

[19] Surender Baswana, Manoj Gupta, and Sandeep Sen. 2011. Fully Dynamic Maximal Matching in O (log n) Update Time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE Computer Society, 383–392.

[20] Surender Baswana, Manoj Gupta, and Sandeep Sen. 2018. Fully Dynamic Maximal Matching in $O(\log n)$ Update Time (Corrected Version). *SIAM J. Comput.* 47, 3 (2018), 617–650.

[21] Soheil Behnezhad. 2022. To appear in SODA 2023. Dynamic Algorithms for Maximum Matching Size. *CoRR* abs/2207.07607 (2022. To appear in SODA 2023).

[22] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Cliff Stein, and Madhu Sudan. 2019. Fully Dynamic Maximal Independent Set with Polylogarithmic Update Time. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. IEEE Computer Society, 382–405.

[23] Soheil Behnezhad and Sanjeev Khanna. 2022. New Trade-Offs for Fully Dynamic Matching via Hierarchical EDCS. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*. SIAM, 3529–3566.

[24] Soheil Behnezhad, Jakub Lacki, and Vahab S. Mirrokni. 2020. Fully Dynamic Matching: Beating 2-Approximation in Δ^{ε} Update Time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. SIAM, 2492–2508.

[25] Aaron Bernstein. 2020. Improved Bounds for Matching in Random-Order Streams. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*. 12:1–12:13.

[26] Aaron Bernstein, Aditi Dudeja, and Zachary Langley. 2021. A Framework for Dynamic Matching in Weighted Graphs. In *Proccedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021, to appear*.

[27] Aaron Bernstein, Sebastian Forster, and Monika Henzinger. 2019. A Deamortization Approach for Dynamic Spanner and Dynamic Maximal Matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. 1899–1918.

[28] Aaron Bernstein and Cliff Stein. 2015. 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 (Lecture Notes in Computer Science, Vol. 9134)*, Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann (Eds.). Springer, 167–179.

[29] Aaron Bernstein and Cliff Stein. 2016. Faster Fully Dynamic Matchings with Small Approximation Ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. SIAM, 692–711.

[30] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. 2018. Deterministic Fully Dynamic Data Structures for Vertex Cover and Matching. *SIAM J. Comput.* 47, 3 (2018), 859–887.

[31] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. 2016. New Deterministic Approximation Algorithms for Fully Dynamic Matching. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. ACM, 398–411.

[32] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. 2017. Fully Dynamic Approximate Maximum Matching and Minimum Vertex Cover in $O(\log^3 n)$ Worst Case Update Time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. SIAM, 470–489.

[33] Sayan Bhattacharya and Peter Kiss. 2021. Deterministic Rounding of Dynamic Fractional Matchings. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*. 27:1–27:14.

[34] Sayan Bhattacharya, Peter Kiss, Thatchaphol Saranurak, and David Wajc. 2022. To appear in SODA 2023. Dynamic Matching with Better-than-2 Approximation in Polylogarithmic Update Time. *CoRR* abs/2207.07438 (2022. To appear in SODA 2023).

[35] Yitzhak Birk, Nathan Linial, and Roy Meshulam. 1993. On the uniform-traffic capacity of single-hop interconnections employing shared directional multichannels. *IEEE Transactions on Information Theory* 39, 1 (1993), 186–191.

[36] Andrej Brodnik and J. Ian Munro. 1999. Membership in Constant Time and Almost-Minimum Space. *SIAM J. Comput.* 28, 5 (1999), 1627–1640.

[37] Moses Charikar and Shay Solomon. 2018. Fully Dynamic Almost-Maximal Matching: Breaking the Polynomial Worst-Case Time Barrier. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. 33:1–33:14.

[38] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. 2021. Almost optimal super-constant-pass streaming lower bounds for reachability. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. ACM, 570–583.

[39] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. 2016. 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, January 10-12, 2016*. 1326–1344.

[40] David Conlon and Jacob Fox. 2013. Graph removal lemmas. *Surveys in combinatorics* 409 (2013), 1–49.

[41] Graham Cormode, Jacques Dark, and Christian Konrad. 2019. Independent Sets in Vertex-Arrival Streams. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. 45:1–45:14.

[42] Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. 2020. 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*. 1773–1785.

[43] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2005. On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348, 2-3 (2005), 207–216.

[44] Moran Feldman and Ariel Szarf. 2021. Maximum Matching sans Maximal Matching: A New Approach for Finding Maximum Matchings in the Data Stream Model. *CoRR* abs/2109.05946. To appear in APPROX 2022. (2021).

[45] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. 2002. Monotonicity testing over general poset domains. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*. 474–483.

[46] Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. 2022. Deterministic $(1+\varepsilon)$-approximate maximum matching with poly$(1/\varepsilon)$ passes in the semi-streaming model and beyond. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, Stefano Leonardi and Anupam Gupta (Eds.). ACM, 248–260.

[47] Jacob Fox. 2011. A new proof of the graph removal lemma. *Annals of Mathematics* 174, 1 (2011), 561–579.

[48] Jacob Fox, Hao Huang, and Benny Sudakov. 2017. On graphs decomposable into induced matchings of linear sizes. *Bulletin of the London Mathematical Society* 49, 1 (2017), 45–57.

[49] Jacob Fox, Hao Huang, and Benny Sudakov. 2017. On graphs decomposable into induced matchings of linear sizes. *Bulletin of the London Mathematical Society* 49, 1 (2017), 45–57.

[50] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. 2012. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*. SIAM, 468–485.

[51] WT Gowers. 2001. Some unsolved problems in additive/combinatorial number theory. *preprint* 4 (2001).

[52] Fabrizio Grandoni, Chris Schwiegelshohn, Shay Solomon, and Amitai Uzrad. 2022. Maintaining an EDCS in General Graphs: Simpler, Density-Sensitive and with Worst-Case Time Bounds. In *5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022, Virtual Conference, January 10-11, 2022*. SIAM, 12–23.

[53] Manoj Gupta and Richard Peng. 2013. Fully Dynamic $(1 + \varepsilon)$-Approximate Matchings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. IEEE Computer Society, 548–557.

[54] Philip Hall. 1987. On representatives of subsets. *Classic Papers in Combinatorics* (1987), 58–62.

[55] Johan Håstad and Avi Wigderson. 2003. Simple analysis of graph tests for linearity and PCP. *Random Struct. Algorithms* 22, 2 (2003), 139–160.

[56] John E. Hopcroft and Richard M. Karp. 1973. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.* 2, 4 (1973), 225–231.

[57] Zoran Ivkovic and Errol L. Lloyd. 1993. Fully Dynamic Maintenance of Vertex Cover. In *Graph-Theoretic Concepts in Computer Science, 19th International Workshop, WG '93, Utrecht, The Netherlands, June 16-18, 1993, Proceedings (Lecture Notes in Computer Science, Vol. 790)*. Springer, 99–111.

[58] Sagar Kale and Sumedh Tirodkar. 2017. Maximum Matching in Two, Three, and a Few More Passes Over Graph Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA (LIPIcs, Vol. 81)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:21.

[59] Michael Kapralov. 2013. 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*. 1679–1697.

[60] Michael Kapralov. 2021. Space Lower Bounds for Approximating Maximum Matching in the Edge Arrival Model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. SIAM, 1874–1893.

[61] Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. 2021. Towards tight bounds for spectral sparsification of hypergraphs. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, Samir Khuller and Virginia Vassilevska Williams (Eds.). ACM, 598–611.

[62] Zohar S. Karnin, Kevin J. Lang, and Edo Liberty. 2016. Optimal Quantile Approximation in Streams. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, Irit Dinur (Ed.). IEEE Computer Society, 71–78.

[63] Peter Kiss. 2022. Deterministic Dynamic Matching in Worst-Case Update Time. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (LIPIcs, Vol. 215)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 94:1–94:21.

[64] Dénes König. 1916. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Math. Ann.* 77, 4 (1916), 453–465.

[65] Christian Konrad. 2015. Maximum Matching in Turnstile Streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, September 14-16, 2015, Proceedings*. 840–852.

[66] Christian Konrad. 2018. A Simple Augmentation Method for Matchings with Applications to Streaming Algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*. 74:1–74:16.

[67] Christian Konrad, Frédéric Magniez, and Claire Mathieu. 2012. 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 (Lecture Notes in Computer Science, Vol. 7408)*. Springer, 231–242.

[68] Christian Konrad and Kheeran K. Naidu. 2021. On Two-Pass Streaming Algorithms for Maximum Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference) (LIPIcs, Vol. 207)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:18.

[69] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. 1999. Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh (Eds.). ACM Press, 251–262.

[70] Andrew McGregor. 2005. 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 (Lecture Notes in Computer Science, Vol. 3624)*, Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan (Eds.). Springer, 170–181.

[71] Andrew McGregor. 2014. Graph stream algorithms: a survey. *SIGMOD Rec.* 43, 1 (2014), 9–20.

[72] Ofer Neiman and Shay Solomon. 2013. Simple deterministic algorithms for fully dynamic maximal matching. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. 745–754.

[73] Krzysztof Onak and Ronitt Rubinfeld. 2010. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 457–464.

[74] Rasmus Pagh. 2001. Low Redundancy in Static Dictionaries with Constant Query Time. *SIAM J. Comput.* 31, 2 (2001), 353–363.

[75] Rajeev Raman and S. Srinivasa Rao. 2003. Succinct Dynamic Dictionaries and Trees. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings (Lecture Notes in Computer Science, Vol. 2719)*. Springer, 357–368.

[76] Mohammad Roghani, Amin Saberi, and David Wajc. 2022. Beating the Folklore Algorithm for Dynamic Matching. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (LIPIcs, Vol. 215)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 111:1–111:23.

[77] Imre Z Ruzsa and Endre Szemerédi. 1978. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai* 18 (1978), 939–945.

[78] Shay Solomon. 2016. Fully Dynamic Maximal Matching in Constant Update Time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. IEEE Computer Society, 325–334.

[79] Shay Solomon. 2016. Fully Dynamic Maximal Matching in Constant Update Time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. IEEE Computer Society, 325–334.

[80] Endre Szemerédi. 1975. *Regular partitions of graphs*. Technical Report. Stanford Univ Calif Dept of Computer Science.

[81] Terence Tao and Van H Vu. 2006. *Additive combinatorics*. Vol. 105. Cambridge University Press.

[82] David Wajc. 2020. *Matching Theory Under Uncertainty*. Ph. D. Dissertation. Carnegie Mellon University.

[83] David Wajc. 2020. Rounding Dynamic Matchings Against an Adaptive Adversary. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. ACM, 194–207.

Received 2022-11-07; accepted 2023-02-06