

Dynamic Algorithms for Maximum Matching Size

Soheil Behnezhad*

Abstract

We study fully dynamic algorithms for maximum matching. This is a well-studied problem, known to admit several update-time/approximation trade-offs. For instance, it is known how to maintain a $1/2$ -approximate matching in $(\text{poly log } n)$ update time or a $2/3$ -approximate matching in $O(\sqrt{n})$ update time, where n is the number of vertices. It has been a long-standing open problem to determine whether either of these bounds can be improved.

In this paper, we show that when the goal is to maintain just the size of the matching (and not its edge-set), then these bounds can indeed be improved. First, we give an algorithm that takes $(\text{poly log } n)$ update-time and maintains a $.501$ -approximation ($.585$ -approximation if the graph is bipartite). Second, we give an algorithm that maintains a $(2/3 + \Omega(1))$ -approximation in $O(\sqrt{n})$ time for bipartite graphs.

Our results build on new connections to sublinear time algorithms. In particular, a key tool for both is an algorithm of the author for estimating the size of maximal matchings in $\tilde{O}(n)$ time [Behnezhad; FOCS 2021]. Our second result also builds on the *edge-degree constrained subgraph (EDCS)* of Bernstein and Stein [ICALP'15, SODA'16]. In particular, while it has been known that EDCS may not include a better than $2/3$ -approximation, we give a new characterization of such tight instances which allows us to break it. We believe this characterization might be of independent interest.

*Khoury College of Computer Sciences, Northeastern University. Webpage: behnezhad.com.

Contents

1	Introduction	1
1.1	Perspective: On Maintaining Size vs. Edges	2
2	Technical Overview	3
3	Preliminaries	4
4	Warm-Up: Beating Half for Bipartite Graphs	5
4.1	The Semi-Dynamic Algorithm (Proof of Lemma 4.2)	5
5	Beating Two-Thirds for Bipartite Graphs	9
5.1	Background on Edge-Degree Constrained Subgraphs (EDCS)	10
5.2	A Structural Result on Tight Instances of EDCS	10
5.3	The Semi-Dynamic Algorithm (Proof of Lemma 5.1) via Lemma 5.4	12
5.4	Proof of Lemma 5.4; the Characterization for Tight EDCS Instances	14
6	Beating Half for General Graphs	21
6.1	The Semi-Dynamic Algorithm (Proof of Lemma 6.1)	21
A	Needed Sublinear Algorithms From [11]	30
B	Proof of Lemma 4.1	31
C	Basic Facts	32

1 Introduction

We study approximate maximum matchings in fully dynamic graphs. We are given an n -vertex graph $G = (V, E)$ that is subject to both edge insertions and deletions. The goal is to maintain (the size of) an approximate maximum matching of G while spending a small time per update. For exact maximum matchings, there are conditional lower bounds that rule out any $O(n^{1-\epsilon})$ update-time algorithm [1, 32, 27] (see also [43, 46] for upper bounds). As such, much of the work in the literature of dynamic matching has been on approximate solutions; see [39, 8, 9, 37, 30, 23, 16, 17, 22, 21, 44, 25, 2, 18, 13, 14, 47, 19, 20, 42, 35, 29, 12] and their references.

1/2-Approximation: When the goal is to maintain a 1/2-approximation, then extremely fast (poly log n) update time algorithms have been known since the work of Baswana, Gupta, and Sen [8] in 2011 (see also [44, 18, 13]). These algorithms maintain a greedy maximal matching, for which the 1/2-approximation guarantee is tight. In sharp contrast, all known $(\frac{1}{2} + \Omega(1))$ -approximate algorithms require a polynomial update-time of $n^{\Omega(1)}$ [21, 14, 47, 12]. The following question, in particular, has been a major open problem of the area for more than a decade:

Open Problem 1. *Is it possible to maintain a $(\frac{1}{2} + \Omega(1))$ -approximation in poly log n update-time?*

To our knowledge, **Open Problem 1** was first asked by Onak and Rubinfeld [39] in 2010. Several subsequent papers have also imposed it as an important open problem. See e.g. [21, Section 4], [17, Section 7], [25, Section 1], [14, Section 1], [47, Section 5], and [12, Section 5].

The fastest known better-than-1/2 approximation requires $O(n^\epsilon)$ update time and obtains a $(\frac{1}{2} + f(\epsilon))$ -approximation, where $\epsilon > 0$ can be any constant. This was proved for general graphs by Behnezhad, Lacki, and Mirrokni [14]. Prior to that, an algorithm with the same trade-off was proposed by Bhattacharya, Henzinger, and Nanongkai [21] for maintaining the size (but not the edges) of the matching in bipartite graphs (see also [47, 20, 12]).

Our first main result is a positive resolution of **Open Problem 1** for maintaining the size (but not the edges) of the matching.

Theorem 1. *There is a fully dynamic algorithm that maintains a .501-approximation of the size of maximum matching in (poly log n) worst-case update time. The algorithm is randomized but works against adaptive adversaries.*

As a warm-up to our techniques, we prove an equivalent of **Theorem 1** for bipartite graphs in **Section 4** (see **Theorem 3**). The approximation ratio turns out to be much better for bipartite graphs: we get a $(2 - \sqrt{2}) \approx .585$ -approximation against oblivious adversaries, and a .542-approximation against adaptive adversaries.

2/3-Approximation: Another notable approximation/update-time trade-off was obtained by Bernstein and Stein [16, 17] in 2015, who showed that a $(2/3 - \epsilon)$ -approximation can be maintained in $O(\sqrt{n} \cdot \text{poly}(1/\epsilon))$ update time for bipartite graphs (see also [17, 29, 35, 12] for generalizations of this result to non-bipartite graphs). They achieved this by introducing an elegant matching sparsifier that they called an *edge-degree constrained subgraph* (EDCS). It is known that the guarantee of 2/3-approximation is tight for EDCS. That is, there are inputs on which the algorithm of Bernstein and Stein [17] does not obtain a better-than-2/3 approximation. Once we go slightly above 2/3-approximation, then the fastest known algorithms require a much larger nearly-linear-

in- n update-time [30]. In particular, a recent algorithm of Assadi, Behnezhad, Khanna, and Li [7] can be used to maintain a $(1 - o(1))$ -approximate matching in $n/(\log^* n)^{\Theta(1)}$ update-time.

It is worth noting that the $2/3$ -approximation has been a barrier in several settings. Most notably, is the *two-party one-way communication* model. For that model, Assadi and Bernstein [4] showed that an (almost) $2/3$ -approximation can be achieved with $O(n)$ communication. On the flip side, Goel, Kapralov, and Khanna [28] proved that a $(2/3 + \Omega(1))$ -approximation requires a much larger $n^{1+\Omega(1/\log \log n)} \gg n \text{ poly log } n$ communication. See also the paper of Assadi and Behnezhad [3] on the importance of the $2/3$ -approximation barrier, and discussions about it in the random-order streaming model. This motivates the next open question, see e.g. [12, Section 5].

Open Problem 2. *Is it possible to maintain a $(\frac{2}{3} + \Omega(1))$ -approximation in $O(\sqrt{n})$ update-time?*

Our second main result is a positive resolution of **Open Problem 2** for bipartite graphs, also for the matching size.

Theorem 2. *For an absolute constant $\delta_0 > 10^{-6}$, there is an algorithm that maintains a $(\frac{2}{3} + \delta_0)$ -approximation of the size of maximum matching in $O(\min\{m^{1/4}, \sqrt{\Delta}\} + \text{poly log } n) = O(\sqrt{n})$ worst-case update time. Here Δ is a fixed upper bound on the maximum degree and m is the number of edges. The algorithm is randomized but works against adaptive adversaries.*

To achieve **Theorem 2**, we prove a new characterization of tight instances of EDCS that might be of independent interest given the versatility of EDCS. We briefly overview this in **Section 2**. See **Section 5.2** for the formal statement of the characterization and a comparison with prior work.

Concurrent Work: In an independent and concurrent work, and similar to our **Theorem 1**, Bhattacharya, Kiss, Saranurak, and Wajc [24] show that a better-than- $1/2$ approximation of maximum matching size can be maintained in $(\text{poly log } n)$ update time, also answering **Open Problem 1** in the affirmative. The high-level approach of both works is the same. In particular, the key new ingredient in both is to use the sublinear time algorithm of Behnezhad [11] for augmenting a $1/2$ -approximate matching. However, the details are different (in particular under adaptive adversaries, [24] obtains a better quantitative improvement over $1/2$). We note that **Theorem 2** for beating $2/3$ -approximations is unique to our paper.

1.1 Perspective: On Maintaining Size vs. Edges

While majority of the algorithms in the literature maintain the edge-set of the matching, dynamic algorithms for maintaining the matching size have also been studied in several works. For instance, progress on dynamic algorithms for exact maximum matchings has been merely made on maintaining its size [43, 46]. These algorithms use algebraic techniques. Maintaining the size has also been studied for approximate matchings. In particular, Bhattacharya, Henzinger, and Nanongkai [21] studied dynamic algorithms for $(\frac{1}{2} + \Omega(1))$ -approximating the maximum matching size in bipartite graphs. The algorithm of [21] maintains a fractional matching, but not an integral one. The bound on the size of the maximum integral matching, nonetheless, follows from the fact that fractional matchings have integrality gap 1 for bipartite graphs. We note that after the work of [21], algorithms have been developed for “losslessly” rounding fractional matchings into integral ones dynamically [47, 20]. As such, fractional matchings can no longer be a source of separation for maintaining the size vs. the edge-set.

The fact that our algorithms in **Theorems 1** and **2** estimate the size of the maximum matching

as opposed to its edge-set is due to a very different reason than those mentioned above. One of the main building blocks of both algorithms is a recent *sublinear time* algorithm of the author [11] that 1/2-approximates the maximum matching size in $\tilde{O}(n)$ time.¹ Had the algorithm in [11] worked for *finding* a 1/2-approximate matching instead of just its size, then our algorithms could have also maintained the edges of the matching. But this is known to be impossible: $\Omega(n^2)$ queries are information theoretically needed to find any $O(1)$ -approximate maximum matching in the adjacency matrix model, even when the graph is promised to have a perfect matching [11].²

In light of the new connection to sublinear time algorithms discovered in this work and the existing separation for approximating the size vs. finding the edge-set of the matching for sublinear time algorithms, it remains a tantalizing future direction to explore whether such separations exist in the dynamic model. Conversely, resolving either of **Open Problems 1** and **2** positively for maintaining the edge-set of the matching would be an excellent result.

2 Technical Overview

Here we give an overview of our algorithms for **Theorems 1** and **2**. To convey the main intuitions in this section, let us make the simplifying assumption that the maximum matching size $\mu(G)$ of the graph G remains $\Omega(n)$ at all times. We note that this assumption can easily be lifted with just a $(\text{poly log } n)$ overhead in the update-time using a randomized “vertex sparsification” idea of the literature [5] (which can be made work against adaptive adversaries too [35]).

A Reduction to Sublinear Algorithms: Suppose that we have an algorithm \mathcal{A} that in T time α -approximates the maximum matching size of a static n -vertex graph. This can be turned into a dynamic algorithm with the “lazy” approach: run \mathcal{A} , keep the solution unchanged for $\Theta(\varepsilon n)$ updates, then repeat. Since the maximum matching size changes by at most one in each update and $\mu(G) = \Omega(n)$, then we have an $(\alpha - \varepsilon)$ -approximation at all times. The (amortized) update-time of the algorithm, on the other hand, is only $O(T/\varepsilon n)$. How is this useful? There is a rich body of work on sublinear time algorithms for maximum matching. Building on a long and beautiful line of work [41, 48, 38, 40, 26, 34, 11], the author showed that a $(1/2 - \varepsilon)$ -approximation can be obtained only in $\tilde{O}(n/\varepsilon^2)$ time [11], which is much smaller than the number of edges in the graph that can be as large as $\Omega(n^2)$. Plugged into the framework above, this gives an (almost) 1/2-approximation in $(\text{poly log } n)$ update-time. Up until very recently, no $o(n^2)$ time algorithm was known for beating 1/2-approximation. This long-standing barrier was very recently broken by Behnezhad, Roghani, Rubinstein, and Saberi [15], who gave an $O(n^{1+\varepsilon})$ time algorithm that obtains a $(\frac{1}{2} + \Omega_\varepsilon(1))$ -approximation. Thus, the same approximation can be maintained in $\tilde{O}(n^\varepsilon)$ update-time using the framework above. Unfortunately, however, this is slower than our desired $(\text{poly log } n)$ time, and somewhat curiously, matches the guarantee of the previous algorithms for maintaining the *edges* of the matching [21, 14].

Beyond Sublinear Algorithms: For traditional sublinear time algorithms, the only input provided is the graph itself. That is, the algorithm is only given query access to the graph (either to its adjacency list or its adjacency matrix). This, however, does not need to be the case for our target application in the framework above. Let us explain this with an example. As discussed, there are algorithms that take $\text{poly log } n$ worst-case update-time, and maintain the edges of a maximal

¹Here and throughout the paper, $\tilde{O}(f) = f \cdot \text{poly log } n$.

²The lower bound is simple. Suppose that the graph is a perfect matching with its vertex IDs being randomly permuted. Finding this matching requires $\Omega(n^2)$ queries to the adjacency matrix.

matching of a fully dynamic graph [9, 44, 18, 13]. One thing we can do, is to run such dynamic algorithms in the background. This way, when the time comes to call the static sublinear time algorithm to estimate the maximum matching size, in addition to providing query access to the graph, we can also feed this maximal matching into the sublinear time algorithm for free. This is, in fact, exactly what we do to prove [Theorem 1](#). In essence, we show that provided adjacency matrix access and given the edge-set of a maximal matching, there is a sublinear time algorithm that $(\frac{1}{2} + \Omega(1))$ -approximates the maximum matching size in $\tilde{O}(n)$ time. Plugged into the framework above, this leads to a $(\frac{1}{2} + \Omega(1))$ -approximate fully dynamic algorithm with poly log n update-time. For bipartite graphs, this sublinear time algorithm turns out to be very simple and clean, and achieves a much better approximation than half. We prove this as a warm-up in [Section 4](#).

Beating 2/3: For our [Theorem 2](#) which beats 2/3-approximation, we follow the same framework discussed above. However, instead of feeding a maximal matching into the sublinear algorithm, we maintain an edge-degree constrained subgraph (EDCS) and feed that into the sublinear algorithm. The EDCS, introduced by Bernstein and Stein [16], is a sparse subgraph that, for the right parameters, includes a 2/3-approximate maximum matching of its base graph. It is known that this bound is tight — that there are graphs for which an EDCS does not include a better than 2/3-approximation. To beat 2/3-approximation, one natural idea is to find the 2/3-approximate matching inside the EDCS, and then try to augment it. In general, a 2/3-approximate matching may not leave any augmenting path of length shorter than 5. It seems challenging to find (or even estimate the number of) length-5 augmenting paths efficiently in our model. To get around this, we prove a new characterization of the tight instances of EDCS. We show that when an EDCS H does not include a strictly larger than 2/3-approximate matching of its base graph G , then it must include a 2/3-approximate matching M that is far from being maximal. In fact, we show that there is a 1/3-approximate matching in G whose edges can be directly added to M . We believe this property might be of independent interest given the versatility of EDCS. See [Section 5.2](#) for more about this and a comparison to a previous characterization by Assadi and the author [3].

3 Preliminaries

Given a graph $G = (V, E)$ and a subset $U \subseteq V$, we use $G[U]$ to denote the induced subgraph of G on U . Given two disjoint subsets A, B of V , we use $G[A, B]$ to denote the bipartite subgraph of G including any of its edges between A and B . Given a vertex v , we use $N_G(v)$ to denote the set of neighbors of v in G . We may drop the subscript when the graph G is clear from the context.

A matching M for G is a subset of its edges such that no two of them share an endpoint. A maximum matching in G is a matching of the largest possible size. We use $\mu(G)$ to denote the size of the maximum matching in G . We say a number $\tilde{\mu}(G)$ is an α -approximation of $\mu(G)$ for $\alpha \in (0, 1]$ if $\alpha\mu(G) \leq \tilde{\mu}(G) \leq \mu(G)$.

Given a matching M , we use $V(M)$ to denote the set of vertices in M . For simplicity, given a vertex v we may write $v \in M$ instead of $v \in V(M)$. Given two matchings M, M^* , we use $M \oplus M^*$ to denote the symmetric difference of the two matchings, i.e., the edges that belong to exactly one of M and M^* . Given a permutation π over the edge-set E of a graph G , we use $\text{GMM}(G, \pi)$ to denote the matching obtained by greedily processing the edges of G in the order of E and adding each edge possible to the matching.

Throughout the paper *w.h.p.* abbreviates *with high probability* by which we mean probability at least $1 - 1/n^c$ for any desirably large constant $c \geq 1$.

4 Warm-Up: Beating Half for Bipartite Graphs

In this section, we prove the following theorem for bipartite graphs.³

Theorem 3. *For bipartite graphs, there is a randomized fully dynamic algorithm that maintains a $(1 - \varepsilon)(2 - \sqrt{2}) \approx .585$ -approximation of the size of maximum matching in $(\text{poly log } n)$ worst-case update-time against oblivious adversaries. If the adversary is adaptive, then there is an algorithm that maintains a .542 approximation in $(\text{poly log } n)$ worst-case update-time.*

First, we start with the algorithm for oblivious adversaries. Then show how to lift the oblivious adversary assumption.

Let us first state a general lemma that we use in all of our results. The lemma guarantees that given a “semi-dynamic” algorithm \mathcal{A} (which only upon being queried produces an estimate of the matching size), one can turn it into a dynamic algorithm \mathcal{B} for maintaining the maximum matching size at all times.

Lemma 4.1 (From Semi-Dynamic Algorithms to Fully-Dynamic Algorithms). *Let G be an n -vertex fully dynamic graph and let $\varepsilon > 0$ be a parameter. Suppose that there is a (randomized) data structure \mathcal{A} (this is the semi-dynamic algorithm) that takes $U(n)$ worst-case time per update to G and, upon being queried, \mathcal{A} produces in $Q(n, \varepsilon)$ time a number $\tilde{\mu}$, such that $\alpha\mu(G) - \varepsilon n \leq \mathbf{E}[\tilde{\mu}] \leq \mu(G)$. Then there is a randomized data structure \mathcal{B} (this is the fully-dynamic algorithm) that maintains a number $\tilde{\mu}'$ such that at any point during the updates, w.h.p., $(\alpha - \varepsilon)\mu(G) \leq \tilde{\mu}' \leq \mu(G)$. Algorithm \mathcal{B} takes $O\left(\left(U(n) + \frac{Q(n, \varepsilon^2)}{n}\right) \text{poly}(\log n, 1/\varepsilon)\right)$ worst-case update-time. Moreover, if \mathcal{A} works against adaptive adversaries, so does \mathcal{B} .*

We emphasize that we do not claim any novelty for [Lemma 4.1](#); its proof, which we present in [Appendix B](#), stitches together known ideas from the literature. Our main novelty for all of our results, is to provide the semi-dynamic algorithm that we plug into [Lemma 4.1](#). As a warm-up in this section, we describe this semi-dynamic algorithm for beating half-approximation in bipartite graphs. In particular, we prove the following lemma.

Lemma 4.2. *For any $\varepsilon > 0$ and any n -vertex fully dynamic graph G , there is a (randomized) data structure \mathcal{A} that takes $(\text{poly log } n)$ worst-case update-time against an oblivious adversary, and upon being queried takes $\tilde{O}(n/\varepsilon^3)$ time to produce a number $\tilde{\mu}$ such that $(2 - \sqrt{2}) \cdot \mu(G) - \varepsilon n \leq \mathbf{E}[\tilde{\mu}] \leq \mu(G)$.*

[Lemma 4.1](#) and [Lemma 4.2](#) together prove [Theorem 3](#) against oblivious adversaries.

Proof of [Theorem 1](#) against oblivious adversaries. Follows by plugging the semi-dynamic algorithm given by [Lemma 4.2](#) into [Lemma 4.1](#). □

4.1 The Semi-Dynamic Algorithm (Proof of [Lemma 4.2](#))

We start by describing the data structures that our semi-dynamic algorithm maintains.

³We note that in the first version of the paper, we obtained a .534-approximation for bipartite graphs. We thank an anonymous SODA'23 reviewer who suggested a tweak in the analysis, leading to the improved bound of [Theorem 3](#).

Data Structures: We maintain the adjacency matrix of the graph G .⁴ Additionally, we maintain a maximal matching M of G which can be done in (poly log n) worst-case update-time against oblivious adversaries [18, 13] (this is the only part of the algorithm that requires the oblivious adversary assumption, we show how this can be lifted at the end of this section). Therefore, overall, the dynamic algorithm requires (poly log n) worst-case update-time.

It remains to show how to produce the number $\tilde{\mu}$ in $\tilde{O}(n/\varepsilon^3)$ time using these data structures, which is what we focus on in the rest of this section.

The Query Algorithm: Our starting point is the following [Algorithm 1](#), which is inspired by a random-order streaming algorithm of Konrad, Magniez, and Mathieu [36].

Algorithm 1:

1. Let M be the maximal matching of G that we maintain.
2. Let $M' \subseteq M$ include each edge of M independently with probability $p = \sqrt{2} - 1$.
3. Let $V' := V(M')$ be the set of vertices matched by M' , and let $U := V \setminus V(M)$ be the vertices left unmatched by maximal matching M .
4. Let $H := G[V', U]$ be the induced bipartite subgraph of G between V' and U . (We will not construct this graph H explicitly.)
5. Let $g := \mathbf{E}_\pi[|\text{GMM}(H, \pi)| \mid M']$.
6. Return $\tilde{\mu}' := |M| + \max\{0, g - |M'|\}$.

Implementing [Algorithm 1](#), as stated, requires $\Omega(n^2)$ time which is much higher than our desired time in [Lemma 4.2](#). However, we show that one can estimate its output up to an additive εn error (which note can be tolerated by [Lemma 4.2](#)) much faster in $\tilde{O}(n/\varepsilon^3)$ time ([Lemma 4.3](#)). This quick implementation of [Algorithm 1](#) is what we use when the semi-dynamic algorithm is queried.

Lemma 4.3. *For any $\varepsilon > 0$, there is an algorithm that w.h.p. takes $\tilde{O}(n/\varepsilon^3)$ time and returns a number $\tilde{\mu}''$ such that $\tilde{\mu}' - \varepsilon n \leq \tilde{\mu}'' \leq \tilde{\mu}'$. Here $\tilde{\mu}'$ is the output of [Algorithm 1](#).*

To prove [Lemma 4.3](#), we use the following sublinear-time greedy matching estimator of the author [11]. See [Appendix A](#) for more details about [Proposition 4.4](#).

Proposition 4.4 ([11]). *Let $G = (V, E)$ be an n -vertex graph to which we have adjacency matrix query access. For any $\varepsilon > 0$, there is a randomized algorithm that w.h.p. takes $\tilde{O}(n/\varepsilon^3)$ time and returns a number \tilde{g} where $\mathbf{E}_\pi[|\text{GMM}(G, \pi)|] - \varepsilon n \leq \tilde{g} \leq \mathbf{E}_\pi[|\text{GMM}(G, \pi)|]$.*

Proof of Lemma 4.3. The construction of M', V', U in [Algorithm 1](#) takes $O(n)$ time since the edges of M are given. But we cannot afford to explicitly construct H as it may have $\Omega(n^2)$ edges. We get around this by using [Proposition 4.4](#) to estimate g without constructing H . To do this, it suffices to show that we can provide adjacency matrix access to H . Observe that an edge $e = (u, v)$ belongs to H iff one of its endpoints is in V' , the other is in U , and additionally $(u, v) \in E$. This can be checked for any (u, v) in $O(1)$ time since we explicitly store V', U and have adjacency matrix access to G . Thus, we can apply [Proposition 4.4](#) on graph H in $\tilde{O}(n/\varepsilon^3)$ time and obtain $g - \varepsilon n \leq \tilde{g} \leq g$. Using \tilde{g} instead of g in the output of [Algorithm 1](#) proves the lemma. \square

⁴Storing the adjacency matrix naively requires $O(n^2)$ space. While space-complexity is often not a concern for dynamic algorithms, we note that the space can also be reduced to $\tilde{O}(m)$ by simply storing the non-zero entries of the adjacency matrix in a binary search tree. This only blows up the time-complexity by a $O(\log n)$ factor.

Next, we turn to prove the approximation guarantee of [Algorithm 1](#). To do this, we start with the following useful proposition, proved by Konrad, Magniez, and Mathieu [\[36\]](#) in the context of streaming algorithms. It shows that a randomized greedy maximal matching obtains a much better than $1/2$ -approximation if it is run on a vertex-sampled subgraph of a bipartite graph.

Proposition 4.5 ([\[36, Theorem 3\]](#)). *Let $0 < p \leq 1$, let $G = (A, B, E)$ be a bipartite graph, let $A' \subseteq A$ include each vertex of A independently with probability p , and let H be the induced subgraph of G on vertex-set $A' \cup B$. Then for any permutation π over the edge-set of H ,*

$$\mathbf{E}_{A'}[|\text{GMM}(H, \pi)|] \geq \frac{p}{1+p} \cdot \mu(G).$$

We are now ready to analyze the approximation ratio. The following lemma gives a lower bound on the expected value of the estimate $\tilde{\mu}'$.

Lemma 4.6. *For [Algorithm 1](#), it holds that $\mathbf{E}[\tilde{\mu}'] \geq (2 - \sqrt{2})\mu(G)$.*

Proof. Let L, R be the two vertex parts in G . Define the following bipartite induced subgraphs of G :

$$\begin{aligned} F_L &:= G[V(M) \cap L, U \cap R], & F_R &:= G[V(M) \cap R, U \cap L], \\ H_L &:= G[V(M') \cap L, U \cap R], & H_R &:= G[V(M') \cap R, U \cap L]. \end{aligned}$$

Let us fix an arbitrary maximum matching M^* of G . Since M^* is a maximum matching of G , we have exactly $|M^*| - |M| = \mu(G) - |M|$ augmenting paths for M in $M^* \oplus M$. Let L_1 be the number of length one augmenting paths in $M^* \oplus M$ for M , noting that $L_1 = 0$ here since M is maximal.⁵ Note that every augmenting path of length at least three in $M^* \oplus M$ has one of its endpoint edges in F_L and the other in F_R . Moreover, these endpoint edges form a matching as they all belong to M^* . Hence,

$$\mu(F_L) \geq \mu(G) - |M| - L_1, \quad \mu(F_R) \geq \mu(G) - |M| - L_1. \quad (1)$$

Moreover, note that since M' is a random subsample of M , then each vertex in $V(M) \cap L$ belongs to $V(M') \cap L$ independently from the vertices in L (but not R) with probability p . As a result, H_L is an induced subgraph of F_L which includes all the vertices in one part, and p fraction of the vertices in the other part independently. Applying [Proposition 4.5](#), we thus get that

$$\mathbf{E}_{M'}[|\text{GMM}(H_L, \pi)|] \geq \frac{p}{1+p} \cdot \mu(F_L) \stackrel{(1)}{\geq} \frac{p}{1+p} \cdot (\mu(G) - |M| - L_1).$$

With essentially the same proof, we get the same lower bound for $\mathbf{E}_{M'}[|\text{GMM}(H_R, \pi)|]$. Since G is bipartite, H_L and H_R are vertex disjoint. Combined with $H = H_L \cup H_R$ this implies that

$$\mathbf{E}_{M'}[|\text{GMM}(H, \pi)|] = \mathbf{E}_{M'}[|\text{GMM}(H_L, \pi)|] + \mathbf{E}_{M'}[|\text{GMM}(H_R, \pi)|] \geq \frac{2p}{1+p} (\mu(G) - |M| - L_1).$$

This in turn implies that

$$\mathbf{E}_{M'}[g] = \mathbf{E}_{M', \pi}[|\text{GMM}(H, \pi)|] \geq \frac{2p}{1+p} (\mu(G) - |M| - L_1). \quad (2)$$

⁵The reason that we define L_1 is that later when we switch to adaptive adversaries, M will not be maximal.

Taking expectation over M' in the output $\tilde{\mu}'$ of [Algorithm 1](#), we get

$$\begin{aligned}
\mathbf{E}_{M'}[\tilde{\mu}'] &= \mathbf{E}_{M'}[|M| + \max\{0, g - |M'|\}] = |M| + \max\{0, \mathbf{E}_{M'}[g] - p|M|\} \\
&\geq (1-p)|M| + \frac{2p}{1+p}(\mu(G) - |M| - L_1) && \text{(By (2).)} \\
&= \left(1-p - \frac{2p}{1+p}\right)|M| + \frac{2p}{1+p}\mu(G) - \frac{2p}{1+p}L_1. && (3)
\end{aligned}$$

Since $p = \sqrt{2} - 1$, we get $1 - p - \frac{2p}{1+p} = 0$ and $\frac{2p}{1+p} = 2 - \sqrt{2}$. Combined with $L_1 = 0$, this implies

$$\mathbf{E}_{M'}[\tilde{\mu}'] \geq (2 - \sqrt{2})\mu(G). \quad \square$$

Next, we show that $\tilde{\mu}'$ does not over-estimate $\mu(G)$.

Lemma 4.7. *For [Algorithm 1](#), it holds with probability 1 that $\tilde{\mu}' \leq \mu(G)$.*

Proof. Take an arbitrary matching S in graph H of [Algorithm 1](#). Let M'' be the subset of edges in M' whose both endpoints are matched by S . By definition of H , every edge in S has one endpoint in $V' = V(M')$ and one endpoint in $U = V \setminus V(M)$. This means that $M \oplus S$ includes at least $|M''|$ length three augmenting paths for M , and so

$$\mu(G) \geq |M| + |M''|. \quad (4)$$

The fact that any edge in S has exactly one vertex in $V(M')$ implies that the number of vertices in $V(M')$ unmatched by S is at most $|V(M')| - |S| = 2|M'| - |S|$. As such, there are at least $|M'| - (2|M'| - |S|) = |S| - |M'|$ edges in M' whose both endpoints are matched by S . Hence, $|M''| \geq |S| - |M'|$. Plugging this into (4), we get that

$$\mu(G) \geq |M| + |S| - |M'|. \quad (5)$$

Now instead of an arbitrary matching, take S to be a maximum matching of H . Note, in particular, that $|S| \geq \mathbf{E}_\pi[|\text{GMM}(H, \pi)| \mid M'] = g$. Plugging this into (5), and noting also that $\mu(G) \geq |M|$ since M is a matching in G , we get that

$$\mu(G) \geq \max\{|M|, |M| + g - |M'|\} = |M| + \max\{0, g - |M'|\} = \tilde{\mu}'. \quad \square$$

We are now ready to complete the proof of [Lemma 4.2](#), which as discussed also proves [Theorem 1](#) for bipartite graphs.

Proof of [Lemma 4.2](#). The data structures that we store, as discussed, take only (poly log n) worst-case time to maintain. When the algorithm is queried, we return the output $\tilde{\mu}''$ of [Lemma 4.3](#). It takes $\tilde{O}(n/\varepsilon^3)$ time to produce this by [Lemma 4.3](#), which is the desired query time of [Lemma 4.2](#). Moreover, for the approximation ratio, we have

$$(2 - \sqrt{2})\mu(G) - \varepsilon n \stackrel{\text{Lemma 4.6}}{\leq} \mathbf{E}[\tilde{\mu}'] - \varepsilon n \stackrel{\text{Lemma 4.3}}{\leq} \mathbf{E}[\tilde{\mu}''] \stackrel{\text{Lemma 4.3}}{\leq} \mathbf{E}[\tilde{\mu}'] \stackrel{\text{Lemma 4.7}}{\leq} \mu(G).$$

This completes the proof of [Lemma 4.2](#). □

Adaptive Adversaries: The only part of the algorithm discussed above that requires the oblivious adversary assumption is the maintenance of maximal matching M . While it is not known whether a maximal matching can be maintained in $(\text{poly log } n)$ time against adaptive adversaries, there are algorithms for maintaining a (non-maximal) $(1/2 - \varepsilon)$ -approximate matching against adaptive adversaries in $\text{poly log } n$ time for any fixed $\varepsilon > 0$ [47]. Let M be one such matching maintained. Observe that in our analysis, we assumed that $L_1 = 0$ which no longer holds for non-maximal M . But note that L_1 being large is not actually that bad as we can simply run the sublinear algorithm of [11] on $G[V \setminus V(M)]$ to get a half-approximation of L_1 , obtaining a matching of size at least $|M| + |L_1|/2$. Returning the larger of this estimate and the output of [Algorithm 1](#), from (3), we get an estimator of size at least

$$(1 - \varepsilon) \max \left\{ \frac{1}{2} \mu(G) + \frac{1}{2} |L_1|, \left(1 - p - \frac{2p}{1+p} \right) |M| + \frac{2p}{1+p} \mu(G) - \frac{2p}{1+p} L_1 \right\}.$$

Setting $p = .3$, this is minimized for $L_1 = .084\mu(G)$, and the end result is a .542-approximation.

Remark 4.8. *We note that the algorithm we employed in this section uses the graph's bipartiteness in a crucial way. In particular, our definitions of graphs F_L and F_R , and the fact that they are vertex disjoint, crucially depends on the graph being bipartite. We later show in [Section 6](#) how one can also beat $1/2$ -approximation for general graphs, albeit with a smaller improvement. Before that, we continue to focus on bipartite graphs, and prove [Theorem 2](#) for them.*

5 Beating Two-Thirds for Bipartite Graphs

In this section, we prove [Theorem 2](#). Our plan, similar to [Section 4](#), is to turn a semi-dynamic algorithm into a fully-dynamic one. However, instead of [Lemma 4.1](#), which blows up the update-time by a $\text{poly log } n$ factor, we use a more refined variant of it that doesn't lose the $\text{poly log } n$ factor. We can do this in this section because our semi-dynamic algorithm will actually guarantee a multiplicative approximation instead of a multiplicative-additive one.

Let us now state the guarantee of our semi-dynamic algorithm in this section.

Lemma 5.1. *For any n -vertex fully dynamic bipartite graph G of maximum degree at most Δ , there is a (randomized) data structure \mathcal{A} that takes $O(\sqrt{\Delta})$ worst-case update-time, and upon being queried takes $O(n\sqrt{\Delta}) + \tilde{O}(n)$ time to produce a number $\tilde{\mu}$ such that for some absolute constant $\delta_0 > 1.8 \times 10^{-6}$, it holds w.h.p. that $(\frac{2}{3} + \delta_0) \cdot \mu(G) \leq \tilde{\mu} \leq \mu(G)$.*

[Lemma 5.1](#) suffices to prove [Theorem 2](#):

Proof of [Theorem 2](#). Let ε be small enough that $\delta_0 - \varepsilon > 10^{-6}$, where $\delta_0 > 1.8 \times 10^{-6}$ is as in [Lemma 5.1](#). We run the semi-dynamic data structure \mathcal{A} of [Lemma 5.1](#) in the background, which takes $O(\sqrt{\Delta})$ worst-case update-time against adaptive adversaries. Suppose that we call the oracle of algorithm \mathcal{A} once to produce the estimate $\tilde{\mu}$. Observe that within the next $\varepsilon\tilde{\mu}$ updates, the maximum matching size of G changes by at most $\varepsilon\tilde{\mu}$, even against adaptive adversaries. Hence, we can use this oracle in a lazy way: we call it once, return $(1 - \varepsilon)\tilde{\mu}$ as the output, do not change the output for $\varepsilon\tilde{\mu}$ updates, then repeat. This way, the amortized time spent on the oracle is indeed $(O(n\sqrt{\Delta}) + \tilde{O}(n)) / \varepsilon n = O(\sqrt{\Delta}) + \text{poly log } n$. This can be turned into a worst-case bound using a well-known ‘spreading’ technique (see [30]). Letting T denote the time spent by the oracle, the

idea is to spread its computation over $\Theta(\varepsilon\tilde{\mu})$ updates, each performing $\Theta(T/\varepsilon\tilde{\mu})$ operations of the algorithm. When the computation of the oracle finishes, we change our output and immediately start spreading its next execution. Finally, we note that by losing only a $(1 - \varepsilon)$ factor in the approximation, one can assume that $\Delta \leq O(\sqrt{m}/\varepsilon)$ using a marking algorithm of [45] (see [29, Section 5] or [12, Section 4.7] for how this can be used for dynamic algorithms). Thus, overall, the update-time that we get is $O(\min\{\sqrt{\Delta}, m^{1/4}\} + \text{poly log } n) = O(\sqrt{n})$ in the worst-case.

The approximation guarantee follows immediately from the high probability and multiplicative guarantee of the semi-dynamic algorithm in [Lemma 5.1](#). \square

In order to prove [Lemma 5.1](#), we build on the edge-degree constrained subgraph (EDCS) of Bernstein and Stein [16]. An EDCS can be used to maintain the edges of a $(2/3 - \varepsilon)$ -approximate maximum matching in $O(\sqrt{n} \text{poly}(1/\varepsilon))$ time [17], and this bound is known to be tight for it. We show how to go beyond 2/3-approximation by proving a certain characterization of the tight instances of EDCS, where it only obtains a 2/3-approximation. We first give some background on EDCS in [Section 5.1](#), then state our characterization for its tight instances in [Section 5.2](#), and then use this characterization to prove [Lemma 5.1](#).

5.1 Background on Edge-Degree Constrained Subgraphs (EDCS)

The EDCS is a matching sparsifier introduced by Bernstein and Stein [16], defined as follows:

Definition 5.2 ([16]). *For any $\beta > \beta_- \geq 1$, a subgraph H of G is a (β, β_-) -EDCS of G if*

- for all edges $(u, v) \in H$, $\deg_H(u) + \deg_H(v) \leq \beta$, and
- for all edges $(u, v) \in G \setminus H$, $\deg_H(u) + \deg_H(v) \geq \beta_-$.

It is known that for any integers $\beta > \beta_- \geq 1$, any graph G has a (β, β_-) -EDCS. The main property of EDCS, first proved in [16, 17] and further refined in [4, 10], is that if $\beta \geq 1/\varepsilon$ and $\beta_- \geq (1 - \varepsilon)\beta$, then $\mu(G) \geq (\frac{2}{3} - O(\varepsilon))\mu(G)$. Moreover, this guarantee is tight. That is, the 2/3 factor cannot be replaced by a larger constant even if $\beta_- = \beta - 1 = \Theta(n)$ [16]. We refer interested readers to the paper of Assadi and Bernstein [4] for an excellent overview of EDCS and its applications across various models.

Several algorithms are known for maintaining an EDCS in dynamic graphs [16, 17, 12, 29, 35]. Here we state a simple and clean algorithm of Grandoni, Schwiegelshohn, Solomon, and Uzdur [29] which is deterministic, works for general graphs, and its update-time bound holds in the worst-case.

Proposition 5.3 ([29]). *Let G be a fully dynamic graph and let Δ be a fixed upper bound on its maximum degree. For some $\beta = \Theta(\sqrt{\Delta} \text{poly}(1/\varepsilon))$, one can maintain the edges of a $(\beta, (1 - \varepsilon)\beta)$ -EDCS H of a graph G , as well as the edges of a $(1 - \varepsilon)$ -approximate maximum matching M_H of H deterministically in worst-case update-time $O(\sqrt{\Delta} \text{poly}(1/\varepsilon))$.*

Combined with the guarantee above on the approximation ratio of EDCS, we get a fully dynamic algorithm that maintains a $(\frac{2}{3} - \varepsilon)$ -approximate matching of G in $O(\sqrt{\Delta} \text{poly}(1/\varepsilon))$ update-time.

5.2 A Structural Result on Tight Instances of EDCS

Recall that our goal in [Lemma 5.1](#) is to go beyond 2/3-approximation. Towards this, we prove a structural result on the tight instances of EDCS, and then use it to break 2/3-approximation.

Lemma 5.4 is this characterization. In essence, it shows that when an EDCS H with the right parameters does not include a strictly larger than $2/3$ -approximation, then there is an almost $2/3$ -approximate matching M in H that is far from being maximal (not maximum) for G . More precisely, **Lemma 5.4** guarantees that there must be a matching of size nearly $\mu(G)/3$ in G among vertices left unmatched by M .

Lemma 5.4 (On Tight Instances of EDCS in Bipartite Graphs). *Let $\varepsilon \in (0, \frac{1}{120})$, let H be a $(\beta, (1-\varepsilon)\beta)$ -EDCS of a bipartite graph G for any $\beta > (1-\varepsilon)\beta \geq 1$. Let V_{mid} and V_{low} respectively include vertices v such that $\deg_H(v) \in [.4\beta, .6\beta]$ and $\deg_H(v) \in [0, .2\beta]$. Let $H' := H[V_{low}, V_{mid}]$ be the subgraph of H on edges with one endpoint in V_{mid} and one in V_{low} . Let $\delta \in (2\varepsilon, \frac{1}{60})$ be any parameter. If $\mu(H) \leq (\frac{2}{3} + \delta)\mu(G)$ then the following hold:*

- (P1) $\mu(H') \geq (\frac{2}{3} - 120\sqrt{\delta})\mu(G)$.
- (P2) For any matching M in H' , $\mu(G[V_{mid} \setminus V(M)]) \geq (\frac{1}{3} - 800 \cdot \delta)\mu(G)$.
- (P3) $|V_{mid}| < 8\mu(G)$.

The proof of **Lemma 5.4** is quite involved, so we defer it to **Section 5.4**. It is worth noting that we did not attempt to optimize the constants in **Lemma 5.4**.

Comparison to a Characterization of Assadi and Behnezhad [3]: Prior to this work, another characterization of tight instances of EDCS was given in [3] in the context of random order streaming algorithms. The characterization of [3] implies existence of a nearly $\frac{2}{3}$ -approximate matching in subgraph $H[V_{mid}]$ when $\mu(H)$ is not strictly larger than $\frac{2}{3}\mu(G)$. Roughly speaking, [3] showed that this matching in $H[V_{mid}]$ can be found early on in the stream, and showed how the rest of the stream could be used to discover many length-five augmenting paths for it and beat $2/3$ -approximation for random-order streams. We do not know how to find or estimate the number of length-five augmenting paths efficiently in the dynamic setting. Fortunately, the guarantee of **Lemma 5.4** helps us avoid them all together, and only focus on length-one augmenting paths instead. The following figure illustrates how the two guarantees differ. It is a tight instance of EDCS, where the vertices in V_{mid} all have degree $\beta/2$ in H . The dashed edges are missed from the EDCS while all other edges are present. It can be confirmed that these missed edges alone imply that $\mu(H) \leq \frac{2}{3}\mu(G)$. The blue matching M_L on the left is the matching in $H[V_{mid}]$ used by [3]. The green matching M_R on the right is the matching of H' that **Lemma 5.4** guarantees to exist. While both are of size $\frac{2}{3}\mu(G)$, the key difference is that each edge of M_R has exactly one endpoint in V_{mid} , whereas both endpoints of all edges of M_L are in V_{mid} . Consequently, while M_L is nearly maximal for G and only leaves length-five augmenting paths, M_R is far from being maximal for G and all the dashed edges can be directly added to it.

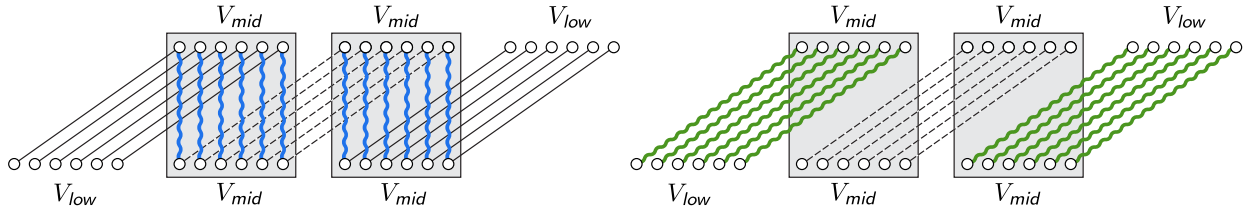


Figure 1: Comparison of the matching implied in the work of Assadi and Behnezhad [3] (the blue matching on the left), and the matching guaranteed by **Lemma 5.4** (the green matching on the right).

5.3 The Semi-Dynamic Algorithm (Proof of Lemma 5.1) via Lemma 5.4

In this section, we show how the characterization of Lemma 5.4 can be used to prove our semi-dynamic algorithm in Lemma 5.1 for beating 2/3-approximation in bipartite graphs.

Data Structures: As in Section 4, we start by describing the data structures maintained by the semi-dynamic algorithm. We maintain the adjacency matrix of the graph G . Additionally, we run Proposition 5.3 to maintain the edges of a $(\beta, (1 - \varepsilon)\beta)$ -EDCS H of G for $\beta = \Theta(\sqrt{\Delta} \text{poly}(1/\varepsilon))$, as well as a $(1 - \varepsilon)$ -approximate maximum matching M_H of H in $O(\sqrt{\Delta} \text{poly}(1/\varepsilon))$ worst-case update-time, where we will set ε to be a sufficiently small absolute constant.

It remains to show how the query algorithm works. That is, how we produce the $(\frac{2}{3} + \delta_0)$ -approximate estimate $\tilde{\mu}$ for $\mu(G)$.

The Query Algorithm: When queried, we run (an estimate) of the following algorithm:

Algorithm 2:

1. Let H be the $(\beta, (1 - \varepsilon)\beta)$ -EDCS that we maintain in our data structure.
2. Let $V_{\text{mid}} := \{v \mid \deg_H(v) \in [.4\beta, .6\beta]\}$ and let $V_{\text{low}} := \{v \mid \deg_H(v) \in [0, .2\beta]\}$.
3. Let $H' := H[V_{\text{low}}, V_{\text{mid}}]$.
4. Find a $(1 - \varepsilon)$ -approximate maximum matching $M_{H'}$ in H' .
5. Let $F := G[V_{\text{mid}} \setminus V(M_{H'})]$. (We will not construct F explicitly.)
6. Let $g := |\text{GMM}(F, \pi)|$ for an arbitrary permutation π .
7. Return $\tilde{\mu}' := \max\{|M_H|, |M_{H'}| + g\}$, where M_H is the $(1 - \varepsilon)$ -approximate matching of H that we maintain in our data structures.

Let us show that the output of Algorithm 2 can be estimated efficiently, in the desired time of Lemma 5.1.

Lemma 5.5. *Let $\tilde{\mu}'$ be as in Algorithm 2. There is an algorithm that takes $O(n\sqrt{\Delta} \text{poly}(1/\varepsilon)) + \tilde{O}(n \text{poly}(1/\varepsilon))$ time and returns a number $\tilde{\mu}''$ such that w.h.p. (i) $\tilde{\mu}'' \leq \tilde{\mu}'$, (ii) $\tilde{\mu}'' \geq |M_H|$, and (iii) $\tilde{\mu}'' \geq |M_{H'}| + g - \varepsilon|V_{\text{mid}}|$.*

Proof. First, note that H , V_{mid} , V_{low} , and H' can all be explicitly constructed in time linear in the size of H , which is $O(n\beta)$ (since any (β, \cdot) -EDCS has maximum degree at most β). Moreover, a $(1 - \varepsilon)$ -approximate matching of an m -edge graph can be found in $O(m/\varepsilon)$ time using the algorithm of Hopcroft and Karp [33]. Therefore, since H' is a subgraph of H and thus also has at most $O(n\beta)$ edges, it takes $O(n\beta/\varepsilon)$ time to construct matching $M_{H'}$ of H' . Finally, instead of constructing F explicitly and computing g for it, we use Proposition 4.4. Note that since we have adjacency matrix access to graph G , we can provide adjacency matrix to its subgraph F as well. Moreover, since F has only $|V_{\text{mid}}|$ vertices by its definition, Proposition 4.4 takes $O(|V_{\text{mid}}|/\varepsilon^3) = \tilde{O}(n/\varepsilon^3)$ time to produce \tilde{g} such that w.h.p. $\mathbf{E}_\pi |\text{GMM}(F, \pi)| - \varepsilon|V_{\text{mid}}| \leq \tilde{g} \leq \mathbf{E}_\pi |\text{GMM}(F, \pi)|$. Using this instead of g in Algorithm 2, we obtain the estimate $\tilde{\mu}''$ that satisfies guarantees (i), (ii), (iii) of the lemma. The final running time of the algorithm is $O(n\beta/\varepsilon) + \tilde{O}(n/\varepsilon^3)$ which is the desired bound of the lemma given that $\beta = \Theta(\sqrt{\Delta} \text{poly}(1/\varepsilon))$. \square

Next, we focus on the approximation ratio of the output of Algorithm 2. The following lemma essentially lower bounds the output of Lemma 5.5 by $(\frac{2}{3} + \delta_0)\mu(G)$.

Lemma 5.6. Let $\delta_0 = 1.8 \times 10^{-6}$. For *Algorithm 2*, at least one of the inequalities $|M_H| \geq (\frac{2}{3} + \delta_0)\mu(G)$ and $|M_{H'}| + g - \varepsilon|V_{\text{mid}}| \geq (\frac{2}{3} + \delta_0)\mu(G)$ must hold.

Proof. Let $\delta = 1.9 \times 10^{-6}$ and suppose that we set $\varepsilon < \delta/100$. If $|M_H| \geq (1 - \varepsilon)(\frac{2}{3} + \delta)\mu(G)$, then

$$|M_H| \geq \left(\frac{2}{3} + \delta - \varepsilon\right)\mu(G) \geq \left(\frac{2}{3} + 0.99\delta\right)\mu(G) \geq \left(\frac{2}{3} + \delta_0\right)\mu(G),$$

which is exactly the first inequality. So let us assume that $|M_H| < (1 - \varepsilon)(\frac{2}{3} + \delta)\mu(G)$. Given that $|M_H| \geq (1 - \varepsilon)\mu(H)$ for being a $(1 - \varepsilon)$ -approximate matching, we get that $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$. Plugging this into the characterization of *Lemma 5.4* for the tight instances of EDCS (noting in particular that δ and ε satisfy the range constraints), we get:

(P1) $\mu(H') \geq (\frac{2}{3} - 120\sqrt{\delta})\mu(G)$. This, in particular, implies that $|M_{H'}| \geq (1 - \varepsilon)(\frac{2}{3} - 120\sqrt{\delta})\mu(G)$.

(P2) For any matching M in H' , $\mu(G[V_{\text{mid}} \setminus V(M)]) \geq (\frac{1}{3} - 800 \cdot \delta)\mu(G)$. Using $M_{H'}$ for M in this statement, this means that $\mu(F) \geq (\frac{1}{3} - 800\delta)\mu(G)$. Since g is at least half the size of $\mu(F)$ for being the size of a maximal matching, we thus get that $g \geq \frac{1}{2}(\frac{1}{3} - 800\delta)\mu(G)$.

(P3) $|V_{\text{mid}}| \leq 8\mu(G)$.

From this, we can infer the second inequality as follows:

$$\begin{aligned} |M_{H'}| + g - \varepsilon|V_{\text{mid}}| &\geq (1 - \varepsilon)\left(\frac{2}{3} - 120\sqrt{\delta}\right)\mu(G) + \frac{1}{2}\left(\frac{1}{3} - 800\delta\right)\mu(G) - 8\varepsilon\mu(G) \\ &\geq (1 - \varepsilon)\left(\frac{2}{3} - 120\sqrt{\delta} + \frac{1}{6} - 400\delta - 8\varepsilon\right)\mu(G) \\ &\geq (1 - \varepsilon)\left(\frac{2}{3} + 1.9 \times 10^{-6} - 8\varepsilon\right)\mu(G) \quad (\text{Since } \delta = 1.9 \times 10^{-6}.) \\ &\geq \left(\frac{2}{3} + \delta_0\right)\mu(G). \quad (\text{Since } \delta_0 = 1.8 \times 10^{-6} \text{ and } \varepsilon < \delta/100.) \end{aligned}$$

This completes the proof. \square

Next, we show that the output $\tilde{\mu}'$ of *Algorithm 2* does not overestimate the matching size.

Observation 5.7. For *Algorithm 2*, it holds with probability 1 that $\tilde{\mu}' \leq \mu(G)$.

Proof. *Algorithm 2* sets $\tilde{\mu}' = \max\{|M_H|, |M_{H'}| + g\}$. Clearly $|M_H| \leq \mu(G)$ since M_H is a matching of $H \subseteq G$. On the other hand, take the matching $M_F = \text{GMM}(F, \pi)$ and note that g is defined to be $|M_F|$ in *Algorithm 2*. Since $F = G[V_{\text{mid}} \setminus V(M_{H'})]$, the edges of M_F are vertex disjoint from $M_{H'}$. Hence, $M_F \cup M_{H'}$ is a matching of G , and so $|M_F \cup M_{H'}| = |M_{H'}| + g \leq \mu(G)$. \square

We are now ready to finish the proof of *Lemma 5.1*.

Proof of Lemma 5.1. The data structures that we store, as discussed, take $O(\sqrt{\Delta} \text{poly}(1/\varepsilon)) = O(\sqrt{\Delta})$ worst-case update time. When the oracle is called, we call the algorithm of *Lemma 5.5* and return its estimate $\tilde{\mu}''$. Its running time is $O(n\sqrt{n}) + \tilde{O}(n)$ since we set ε to be an absolute constant. This is the desired running time in *Lemma 5.1*. For the approximation, first note by *Lemma 5.5* and *Observation 5.7*, that we have $\tilde{\mu}'' \leq \mu(G)$ w.h.p. Moreover, by *Lemma 5.6* and *Lemma 5.5*, we have $\tilde{\mu}'' \geq (\frac{2}{3} + \delta_0)\mu(G)$ w.h.p. This completes the proof. \square

5.4 Proof of Lemma 5.4; the Characterization for Tight EDCS Instances

In this section, we prove Lemma 5.4. Let L and R with $|L| = |R| = n$ be the two vertex parts for graph G . We use the following standard extension of the Hall's theorem.

Proposition 5.8 (Extended Hall's Theorem [31]). *Let $G = (L, R, E)$ be a bipartite graph and $|L| = |R| = n$. Then,*

$$\max(|A| - |N(A)|) = n - \mu(G),$$

where A ranges over L or R , separately. We refer to such set A as a witness set.

Let A be the Hall's witness of H as defined in Proposition 5.8. Suppose w.l.o.g. that $A \subseteq L$. Define $\bar{A} := L \setminus A$, $B := N_H(A)$, $\bar{B} := R \setminus B$. Fix a maximum matching M^* of graph G . Let \bar{M}^* be the edges in M^* that have one endpoint in A and one endpoint in \bar{B} . Note that no edge of \bar{M}^* can belong to H . We also define $S := V(\bar{M}^*)$, $W := (A \cup \bar{B}) \setminus S$, and $T = \bar{A} \cup B$. See Figure 2.

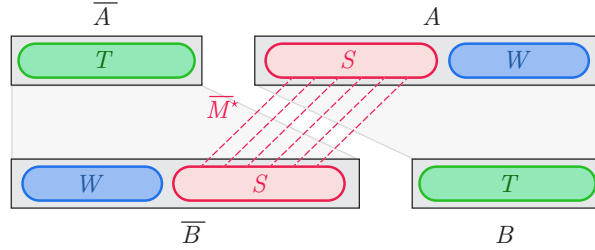


Figure 2: An illustration of the Hall's witness A for H , along with the sets $B, \bar{A}, \bar{B}, S, W$.

We show in this section that when H does not include a larger than $2/3$ -approximation, then (almost) all vertices in S and T must have degree very close to $\beta/2$ and so belong to U (as defined in Lemma 5.4). Additionally, we show that very few vertices of W belong to U .

To proceed, we need some notation that we summarize below.

- H_S, H_W : We partition the edges of H into two subgraphs H_S and H_W . The edges between S and T belong to H_S and the edges between W and T belong to H_W .
- $d(v), d_S(v), d_W(v)$: The degree of a vertex v in graphs H, H_S , and H_W respectively.
- m, m_S, m_W : The number of edges in H, H_S , and H_W respectively. (Other than the discussion of this section, we use m to denote the number of edges of G instead.)
- $\bar{d}_S(T) := m_S/|T|$, $\bar{d}_W(T) := m_W/|T|$, $\bar{d}(T) := m/|T|$: These are the average degrees of vertices in T in graphs H_S, H_W , and H respectively.
- $\bar{d}(S) := m_S/|S|$, $\bar{d}(W) := m_W/|W|$: The average degrees in S and W respectively.
- $\hat{T} := \{v \in T : d(v) > (1 + \alpha)d_S(v)\}$ where $\alpha := 3\sqrt{\delta}$.

The characterization: Our proofs proceed by assuming $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, and then proving some structural properties of the subgraph H . In particular, we show that if $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then all the following must hold:

- $|S \setminus V_{\text{mid}}| \leq 199 \cdot \delta\mu(G)$ (stated as Claim 5.17). That is, almost all vertices in S belong to V_{mid} .

- $|T \setminus V_{\text{mid}}| \leq 600 \cdot \delta \mu(G)$ (stated as [Claim 5.18](#)). That is, almost all vertices in T belong to V_{mid} .
- $|W \setminus V_{\text{low}}| \leq 33 \cdot \sqrt{\delta} \mu(G)$ (stated as [Claim 5.19](#)). That is, almost all vertices in W belong to V_{low} .

We note that the upper bound on $|S \setminus V_{\text{mid}}|$ can also be inferred from the characterization of Assadi and Behnezhad [[3](#)]. The other two bounds require new ideas.

We first show how these properties imply our desired [Lemma 5.4](#) for bipartite graphs.

Proof of [Lemma 5.4](#) for bipartite graphs. We prove (P1), (P2), and (P3) of [Lemma 5.4](#) one by one.

(P1): $\mu(H') \geq (\frac{2}{3} - 120\sqrt{\delta})\mu(G)$. Take the maximum matching M^* of G we fixed at the beginning of this section. Recall that \overline{M}^* is the edges of M^* that go from A to \overline{B} , and also recall that we defined $S = V(\overline{M}^*)$. Now take the matching $\widetilde{M}^* := M^* \setminus \overline{M}^*$. Since M^* is a matching and $V(\overline{M}^*) = S$, we get that no vertex in S can be matched by \widetilde{M}^* . Hence, any edge in \widetilde{M}^* must have one endpoint in T and one endpoint in W . Call an edge $e \in \widetilde{M}^*$ *bad* if it has an endpoint in $W \setminus V_{\text{low}}$ or an endpoint in $T \setminus V_{\text{mid}}$, and *good* otherwise. We have

$$(\# \text{ of bad edges}) \leq |W \setminus V_{\text{low}}| + |T \setminus V_{\text{mid}}| \stackrel{\text{Claims 5.18, 5.19}}{\leq} 33\sqrt{\delta}\mu(G) + 600\delta\mu(G) < 119\sqrt{\delta}\mu(G),$$

where the last inequality holds since $\delta < 1/60$. From this, we get that

$$\begin{aligned} (\# \text{ of good edges}) &\geq |\widetilde{M}^*| - 119\sqrt{\delta}\mu(G) \\ &= (\mu(G) - |S|/2) - 119\sqrt{\delta}\mu(G) \\ &\quad (\text{Since } |M^*| = \mu(G), |S| = 2|\overline{M}^*|, \text{ and } |\widetilde{M}^*| = |M^*| - |\overline{M}^*|.) \\ &\geq \mu(G) - \frac{1}{2}(\frac{2}{3} + 3\delta)\mu(G) - 119\sqrt{\delta}\mu(G) \quad (|S| \leq (\frac{2}{3} + 3\delta)\mu(G) \text{ by } \text{Claim 5.15.}) \\ &> \left(\frac{2}{3} - 120\sqrt{\delta}\right)\mu(G). \quad (\text{Holds since } \delta < 1/60.) \end{aligned}$$

Now observe that since a good edge (u, v) has one endpoint in V_{low} and one endpoint in V_{mid} , we have $d(u) + d(v) \leq .2\beta + .6\beta = .8\beta$. Since H is a $(\beta, (1-\varepsilon)\beta)$ -EDCS of G and $.8\beta < (1-\varepsilon)\beta$, excluding (u, v) from H would violate the second property of EDSCS. Hence, all good edges in \widetilde{M}^* must belong to H . Additionally, all good edges must also belong to subgraph H' of H , since they have one endpoint in V_{mid} and one in V_{low} . Thus, we get that $\mu(H') \geq (\# \text{ of good edges}) \geq (\frac{2}{3} - 120\sqrt{\delta})\mu(G)$.

(P2): For any matching M in H' , $\mu(G[V_{\text{mid}} \setminus V(M)]) \geq (\frac{1}{3} - 800\delta)\mu(G)$.

We say an edge $e \in \overline{M}^*$ is *wasted* if at least one of its endpoints is matched by M . Take a wasted edge $(u, v) \in \overline{M}^*$ and suppose that its endpoint v is matched to some vertex w in M . First, note that $v \in S$ since $S = V(\overline{M}^*)$ and note that $w \in T$ since $(u, w) \in M \subseteq H' \subseteq H$ and all edges of S go to T in H . Second, note that since M is a matching in H' , one vertex of (w, v) must belong to V_{mid} and one to V_{low} . From this, we get that for any wasted edge in \overline{M}^* , there is at least one dedicated vertex in $S \cup T$ that belongs to V_{low} . Hence,

$$\begin{aligned} (\# \text{ of wasted edges}) &\leq |S \cap V_{\text{low}}| + |T \cap V_{\text{low}}| \\ &\leq |S \setminus V_{\text{mid}}| + |T \setminus V_{\text{mid}}|. \quad (\text{Since } V_{\text{low}} \cap V_{\text{mid}} = \emptyset \text{ by their definition.}) \\ &\leq 199\delta\mu(G) + 600\delta\mu(G) \quad (\text{By } \text{Claims 5.17 and 5.18.}) \end{aligned}$$

$$= 799\delta\mu(G).$$

Since any edge of \overline{M}^* that is not wasted belongs to $G[V_{\text{mid}} \setminus V(M)]$, we get that

$$\begin{aligned} \mu(G[V_{\text{mid}} \setminus V(M)]) &\geq |\overline{M}^*| - 799\delta\mu(G) = \frac{1}{2}|S| - 799\delta\mu(G) \\ &\geq \frac{1}{2} \left(\frac{2}{3} - 2\delta \right) \mu(G) - 799\delta\mu(G) && \text{(By Claim 5.15.)} \\ &= \left(\frac{1}{3} - 800\delta \right) \mu(G). \end{aligned}$$

(P3): $|V_{\text{mid}}| < 2\mu(G)$. We have

$$\begin{aligned} |V_{\text{mid}}| &= |V_{\text{mid}} \cap S| + |V_{\text{mid}} \cap T| + |V_{\text{mid}} \cap W| && \text{(Since } S, T, W \text{ partition } V.) \\ &\leq |S| + |T| + |V_{\text{mid}} \cap W| \\ &\leq |S| + |T| + |W \setminus V_{\text{low}}| && \text{(Since } V_{\text{low}} \cap V_{\text{mid}} = \emptyset.) \\ &\leq |S| + |T| + 33\sqrt{\delta}\mu(G) && \text{(By Claim 5.19.)} \\ &< 8\mu(G), \end{aligned}$$

where the last inequality follows from $|S| = 2|\overline{M}^*| \leq 2\mu(G)$, $|T| = \mu(H) \leq \mu(G)$, and $33\sqrt{\delta} < 4.3$ (since $\delta < 1/60$). \square

Thus, it just remains to prove the three upper bounds above on $|S \setminus V_{\text{mid}}|$, $|T \setminus V_{\text{mid}}|$, and $|W \setminus V_{\text{low}}|$. We continue with some basic guarantees of the Hall's witness in [Section 5.4.1](#), prove a parametrized guarantee on $\bar{d}_S(T)$ in [Section 5.4.2](#), prove some useful auxiliary claims in [Section 5.4.3](#), and then the upper bounds on $|S \setminus V_{\text{mid}}|$, $|T \setminus V_{\text{mid}}|$, and $|W \setminus V_{\text{low}}|$ in [Section 5.4.4](#).

5.4.1 Basic Guarantees of Hall's Witness

[Claims 5.9](#) to [5.11](#) below are all by now standard in analyzing EDCS. We provide the full proofs nonetheless to keep our discussion of this section self-contained.

Claim 5.9. $\mu(H) = |T|$.

Proof. We have $|T| = |\overline{A}| + |B| = n - (|A| - |B|) = n - (n - \mu(H)) = \mu(H)$, where the third equation follows from [Proposition 5.8](#) and the fact that A is a witness set of H . \square

Claim 5.10. $|S| \geq 2(\mu(G) - \mu(H))$.

Proof. Let $H^* := M^* \cup H$. Note that $\mu(H^*) \geq |M^*| = \mu(G)$. From [Proposition 5.8](#), this means that $|A| - |N_{H^*}(A)| \leq n - \mu(H^*) \leq n - \mu(G)$. On the other hand, since A is a witness set for H , we know that $|A| - |B| = n - \mu(H)$. Putting the two together, we get that $|N_{H^*}(A)| - |B| \geq \mu(G) - \mu(H)$. From the construction of H^* , this implies that at least $\mu(G) - \mu(H)$ edges of A in M^* should go to \overline{B} , implying that $|\overline{M}^*| \geq \mu(G) - \mu(H)$. The claim follows since $|S| = 2|\overline{M}^*|$ by definition. \square

Claim 5.11. $\mu(H) \geq \frac{2\bar{d}(S)}{2\bar{d}(S) + \bar{d}_S(T)} \cdot \mu(G)$.

Proof. Observe from definition that $\bar{d}_S(T) \cdot |T| = m_S = \bar{d}(S) \cdot |S|$. Therefore $|T| = \frac{|S|\bar{d}(S)}{\bar{d}_S(T)}$. Plugging $|S| \geq 2(\mu(G) - \mu(H))$ of [Claim 5.10](#), and $\mu(H) = |T|$ of [Claim 5.9](#), we get that

$$\mu(H) \geq \frac{2(\mu(G) - \mu(H))\bar{d}(S)}{\bar{d}_S(T)} \Leftrightarrow \mu(H) \geq \frac{2\bar{d}(S)}{2\bar{d}(S) + \bar{d}_S(T)} \cdot \mu(G). \quad \square$$

5.4.2 A Parametrized Lower Bound on the degrees of T to S

The following [Lemma 5.12](#) is our most technical lemma of this section. It gives a useful lower bound on the average degree $\bar{d}_S(T)$ based on a parameter $\gamma \geq 0$. We will later show that if $\mu(H)$ is not much larger than $\frac{2}{3}\mu(G)$, then γ should be very close to zero, implying several useful properties on the structure of such tight instances.

Lemma 5.12. *It holds that $\bar{d}_S(T) \leq (1 - \gamma)\beta - \bar{d}(S)$ where*

$$\gamma := \frac{\sum_{v \in S} (d(v) - \bar{d}(S))^2 + \frac{1}{2} \sum_{v \in T} (d_S(v) - \bar{d}_S(T))^2 + \frac{1}{4} |\hat{T}| \delta \bar{d}_S(T)^2}{m_S \beta} \geq 0.$$

Proof. Since H is a $(\beta, (1 - \varepsilon)\beta)$ -EDCS, we get from the first condition of EDCS that

$$\sum_{(u,v) \in H_S} d(u) + d(v) \leq \sum_{(u,v) \in H_S} \beta = m_S \beta. \quad (6)$$

Let us now focus on the LHS of (6). Each vertex $v \in S \cup T$ participates in the sum $d_S(v)$ times for each of its edges in H_S , and each time adds a value of $d(v)$ to the sum. Hence,

$$\begin{aligned} \sum_{(u,v) \in H_S} d(u) + d(v) &= \sum_{v \in S \cup T} d_S(v) d(v) \\ &= \sum_{v \in S} d(v)^2 + \sum_{v \in T} d_S(v) d(v) \quad (\text{Since } d_S(v) = d(v) \text{ for all } v \in S.) \\ &= |S| \bar{d}(S)^2 + \sum_{v \in S} (d(v) - \bar{d}(S))^2 + \sum_{v \in T} d_S(v) d(v) \\ &\quad (\text{This follows from applying Fact C.1 on the first quadratic sum.}) \\ &\geq m_S \bar{d}(S) + \sum_{v \in S} (d(v) - \bar{d}(S))^2 + \sum_{v \in T} d_S(v) d(v). \quad (\text{Since } |S| \bar{d}(S) = m_S.) \end{aligned}$$

Plugging this lower bound on the LHS of (6) back to (6) and moving the terms, we get that

$$\sum_{v \in T} d_S(v) d(v) \leq m_S (\beta - \bar{d}(S)) - \sum_{v \in S} (d(v) - \bar{d}(S))^2. \quad (7)$$

Next, we focus on the LHS of (7). For any vertex $v \in T$ define $x_v := d_S(v) - \bar{d}_S(T)$. We have

$$\begin{aligned} \sum_{v \in T} d_S(v) d(v) &\geq \sum_{v \in T \setminus \hat{T}} d_S(v)^2 + (1 + \alpha) \sum_{v \in \hat{T}} d_S(v)^2 \quad (\text{By definition of } \hat{T}.) \\ &= \sum_{v \in T} d_S(v)^2 + \alpha \sum_{v \in \hat{T}} d_S(v)^2 \\ &= \sum_{v \in T} (\bar{d}_S(T) + x_v)^2 + \alpha \sum_{v \in \hat{T}} (\bar{d}_S(T) + x_v)^2 \quad (\text{By definition of } x_v.) \\ &= |T| \bar{d}_S(T)^2 + \sum_{v \in T} x_v^2 + 2 \bar{d}_S(T) \sum_{v \in T} x_v + \alpha \sum_{v \in \hat{T}} (\bar{d}_S(T) + x_v)^2 \\ &= |T| \bar{d}_S(T)^2 + \sum_{v \in T} x_v^2 + \alpha \sum_{v \in \hat{T}} (\bar{d}_S(T) + x_v)^2 \\ &\quad (\text{Since } \sum_{v \in T} x_v = \sum_{v \in T} (d_S(v) - \bar{d}_S(T)) = \sum_{v \in T} d_S(v) - \sum_{v \in T} \bar{d}_S(T) = m_S - m_S = 0.) \end{aligned}$$

$$\begin{aligned}
&\geq |T|\bar{d}_S(T)^2 + (1 - \alpha) \sum_{v \in T} x_v^2 + \sum_{v \in \hat{T}} (\alpha x_v^2 + \alpha(\bar{d}_S(T) + x_v)^2) \\
&\geq |T|\bar{d}_S(T)^2 + (1 - \alpha) \sum_{v \in T} x_v^2 + \sum_{v \in \hat{T}} \alpha \bar{d}_S(T)^2 / 4 \\
&\text{(If } |x_v| \geq \bar{d}_S(T)/2 \text{ then } \alpha x_v^2 \geq \alpha \bar{d}_S(T)^2 / 4 \text{ otherwise } \alpha(\bar{d}_S(T) + x_v)^2 \geq \alpha \bar{d}_S(T)^2 / 4.) \\
&\geq m_S \cdot \bar{d}_S(T) + \frac{1}{2} \sum_{v \in T} (d_S(v) - \bar{d}_S(T))^2 + |\hat{T}| \cdot \alpha \bar{d}_S(T)^2 / 4. \\
&\quad \text{(By definition of } x_v, \text{ and since } \alpha = 3\sqrt{\delta} < 3\sqrt{1/60} < 1/2.)
\end{aligned}$$

Plugging this lower bound on the LHS of (7) back to (7), we get that

$$m_S \cdot \bar{d}_S(T) + (1 - \alpha) \sum_{v \in T} (d_S(v) - \bar{d}_S(T))^2 + |\hat{T}| \cdot \alpha \bar{d}_S(T)^2 / 4 \leq m_S(\beta - \bar{d}(S)) - \sum_{v \in S} (d(v) - \bar{d}(S))^2.$$

Moving the terms, we get that

$$\begin{aligned}
\bar{d}_S(T) &\leq (\beta - \bar{d}(S)) - \frac{\sum_{v \in S} (d(v) - \bar{d}(S))^2 + \frac{1}{2} \sum_{v \in T} (d_S(v) - \bar{d}_S(T))^2 + |\hat{T}| \alpha \bar{d}_S(T)^2 / 4}{m_S} \\
&= (\beta - \bar{d}(S)) - \gamma\beta && \text{(By definition of } \gamma \text{ in the claim statement.)} \\
&= (1 - \gamma)\beta - \bar{d}(S).
\end{aligned}$$

This is the desired upper bound on $\bar{d}_S(T)$. Note also that the non-negativity of γ follows from the fact that all the terms in it are non-negative. \square

5.4.3 Some Auxiliary Claims

Before proving our main characterization, we prove a few useful claims in this section. Namely, that under $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, we have

- $(1 - \varepsilon)\beta/2 \leq \bar{d}(S) \leq (1 + 3\delta)\beta/2$ (stated as [Claim 5.13](#)).
That is, the average degree of S in H should be close to $\beta/2$.
- $(1 - 6\delta)\beta/2 \leq \bar{d}_S(T) \leq (1 + \varepsilon)\beta/2$ (stated as [Claim 5.14](#)).
That is, the average degree of T to S in H should be close to $\beta/2$.
- $(\frac{2}{3} - 2\delta)\mu(G) \leq |S| \leq (\frac{2}{3} + 3\delta)\mu(G)$ (stated as [Claim 5.15](#)).
That is, $|S|$ should be close to $\frac{2}{3}\mu(G)$.
- $\gamma \leq 2\delta$ (stated as [Claim 5.16](#)).
That is, the parameter γ of [Lemma 5.12](#) should be small.

We now state and prove these claims one by one.

Claim 5.13. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $(1 - \varepsilon)\beta/2 \leq \bar{d}(S) \leq (1 + 3\delta)\beta/2$.*

Proof. We first prove the lower bound which in fact holds regardless of the assumption of the claim. Since H has no edges between A and \bar{B} by definition, then any edge in \bar{M}^* must be missing from H . Hence, by the second condition of EDCS, for any $(u, v) \in \bar{M}^*$, $d(u) + d(v) \geq (1 - \varepsilon)\beta$.

Thus $\sum_{(u,v) \in \overline{M}^*} d(u) + d(v) \geq |\overline{M}^*|(1 - \varepsilon)\beta$. The LHS equals $\sum_{v \in S} d(v) = m_S$ since $S = V(\overline{M}^*)$. Dividing through by $|S| = 2|\overline{M}^*|$ we obtain $\bar{d}(S) \geq (1 - \varepsilon)\beta/2$.

For the upper bound, suppose for contradiction that $\bar{d}(S) > (1 + 3\delta)\beta/2$. We have

$$\begin{aligned} \mu(H) &\geq \frac{2\bar{d}(S)}{2\bar{d}(S) + \bar{d}_S(T)}\mu(G) && \text{(By Claim 5.11.)} \\ &\geq \frac{2\bar{d}(S)}{2\bar{d}(S) + \beta - \bar{d}(S)}\mu(G) && \text{(Since } \bar{d}_S(T) \leq (1 - \gamma)\beta - \bar{d}(S) \leq \beta - \bar{d}(S) \text{ by Lemma 5.12.)} \\ &= \frac{2\bar{d}(S)}{\beta + \bar{d}(S)}\mu(G) > \frac{(1 + 3\delta)\beta}{\beta + (1 + 3\delta)\beta/2}\mu(G) = \frac{2 + 6\delta}{3 + 3\delta}\mu(G) \geq \left(\frac{2}{3} + \delta\right)\mu(G), \end{aligned}$$

where the last inequality holds for all $0 \leq \delta \leq 1/3$. This contradicts the assumption of the claim, and so the claimed upper bound on $\bar{d}(S)$ must hold. \square

Claim 5.14. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $(1 - 6\delta)\beta/2 \leq \bar{d}_S(T) \leq (1 + \varepsilon)\beta/2$.*

Proof. The upper bound follows from $\bar{d}_S(T) \leq \beta - \bar{d}(S)$ of Lemma 5.12 and $\bar{d}(S) \geq (1 - \varepsilon)\beta/2$ of Claim 5.13. For the lower bound, suppose for contradiction that $\bar{d}_S(T) < (1 - 6\delta)\beta/2$. We have

$$\begin{aligned} \mu(H) &\stackrel{\text{Claim 5.11}}{\geq} \frac{2\bar{d}(S)}{2\bar{d}(S) + \bar{d}_S(T)}\mu(G) \stackrel{\text{Claim 5.13}}{\geq} \frac{(1 - \varepsilon)\beta}{(1 - \varepsilon)\beta + \bar{d}_S(T)}\mu(G) \geq \frac{(1 - \varepsilon)\beta}{\beta + (1 - 6\delta)\beta/2}\mu(G) \\ &= \frac{2 - 2\varepsilon}{3 - 6\delta}\mu(G) \geq \left(\frac{2}{3} + \delta\right)\mu(G), \end{aligned}$$

where the last inequality holds for all $1/2 > \delta \geq 2\varepsilon \geq 0$. This contradicts the assumption of the claim and proves the lower bound on $\bar{d}_S(T)$. \square

Claim 5.15. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $(\frac{2}{3} - 2\delta)\mu(G) \leq |S| \leq (\frac{2}{3} + 3\delta)\mu(G)$.*

Proof. For the lower bound, observe that

$$|S| \stackrel{\text{Claim 5.10}}{\geq} 2(\mu(G) - \mu(H)) > 2\left(\mu(G) - \left(\frac{2}{3} + \delta\right)\mu(G)\right) \geq \left(\frac{2}{3} - 2\delta\right)\mu(G).$$

For the upper bound, we have

$$\mu(H) \stackrel{\text{Claim 5.9}}{=} |T| = \frac{m_S}{\bar{d}_S(T)} = \frac{|S|\bar{d}(S)}{\bar{d}_S(T)} \stackrel{\text{Claims 5.13 and 5.14}}{\geq} \frac{|S|(1 - \varepsilon)\beta/2}{(1 + \varepsilon)\beta/2} \geq (1 - 2\varepsilon)|S|.$$

Hence, the assumption $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$ implies that

$$|S| \leq \frac{1}{1 - 2\varepsilon} \left(\frac{2}{3} + \delta\right)\mu(G) \stackrel{(\varepsilon < 1/120)}{<} (1 + 4\varepsilon) \left(\frac{2}{3} + \delta\right)\mu(G) \stackrel{(\delta > 2\varepsilon, 0 < \delta < 1/60)}{<} \left(\frac{2}{3} + 3\delta\right)\mu(G).$$

\square

Claim 5.16. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $\gamma \leq 2\delta$.*

Proof. Suppose for contradiction that $\gamma > 2\delta$. Then

$$\begin{aligned} \bar{d}_S(T) &\leq (1 - \gamma)\beta - \bar{d}(S) && \text{(By Lemma 5.12)} \\ &< (1 - 2\delta)\beta - \bar{d}(S) \\ &\leq (1 - 2\delta)\beta - (1 + 3\delta)\beta/2 && \text{(By Claim 5.13)} \\ &\leq (1 - 6\delta)\beta/2. \end{aligned}$$

This contradicts Claim 5.14 that $\bar{d}_S(T) \geq (1 - 6\delta)\beta/2$ and proves the claim. \square

5.4.4 The Main Characterization

Having proved the auxiliary claims above and the parametrized guarantee of [Lemma 5.12](#), we ready to prove the main characterizations of this section on $|S \setminus V_{\text{mid}}|$, $|T \setminus V_{\text{mid}}|$, and $|W \setminus V_{\text{low}}|$.

Claim 5.17. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $|S \setminus V_{\text{mid}}| \leq 199 \cdot \delta\mu(G)$.*

Proof. Suppose for contradiction that $|S \setminus V_{\text{mid}}| > 199 \cdot \delta\mu(G)$. If $v \notin V_{\text{mid}}$, then $d(v) \notin [.4\beta, .6\beta]$ by definition of V_{mid} . Since $(1 - \varepsilon)\beta/2 \leq \bar{d}(S) \leq (1 + 3\delta)\beta/2$ by [Claim 5.13](#) and $\delta < 1/60$ and $\varepsilon < 1/120$, we get $.49\beta < \bar{d}_S(T) < .53\beta$. Thus for all $v \in S \setminus V_{\text{mid}}$, we have $(d(v) - \bar{d}(S))^2 \geq (.07\beta)^2$. Hence, by definition of γ in [Lemma 5.12](#) and the non-negativity of the terms in its numerator, we get that

$$\begin{aligned} \gamma &\geq \frac{\sum_{v \in S} (d(v) - \bar{d}(S))^2}{m_S \beta} \geq \frac{|S \setminus V_{\text{mid}}| \cdot (.07\beta)^2}{m_S \beta} = \frac{|S \setminus V_{\text{mid}}| \cdot (.07\beta)^2}{|S| \bar{d}(S) \beta} \stackrel{\text{Claim 5.13}}{\geq} \frac{|S \setminus V_{\text{mid}}| \cdot (.07\beta)^2}{|S| (1 + 3\delta) \beta^2 / 2} \\ &\geq 2(1 - 4\delta)(.07)^2 \frac{|S \setminus V_{\text{mid}}|}{|S|} \geq \frac{1}{110} \cdot \frac{|S \setminus V_{\text{mid}}|}{|S|}. \quad (\text{Since } \delta < 1/60.) \\ &\geq \frac{1}{110} \cdot \frac{199 \cdot \delta\mu(G)}{(\frac{2}{3} + 3\delta)\mu(G)} \quad (\text{By Claim 5.15 and the assumption that } |S \setminus V_{\text{mid}}| > 199 \cdot \delta\mu(G).) \\ &> 2\delta. \quad (\text{Since } \delta < 1/60.) \end{aligned}$$

But this contradicts [Claim 5.16](#) that $\gamma \leq 2\delta$, completing the proof. \square

Claim 5.18. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $|T \setminus V_{\text{mid}}| \leq 600 \cdot \delta\mu(G)$.*

Proof. Suppose for contradiction that $|T \setminus V_{\text{mid}}| > 600 \cdot \delta\mu(G)$. If $v \notin V_{\text{mid}}$, then by definition of V_{mid} , we have $d(v) \notin [.4\beta, .6\beta]$. Since by [Claim 5.14](#) $(1 - 6\delta)\beta/2 \leq \bar{d}_S(T) \leq (1 + \varepsilon)\beta/2$ and $\delta < 1/60$ and $\varepsilon < 1/120$, we get $.45\beta \leq \bar{d}_S(T) < .51\beta$. Thus for all $v \in T \setminus V_{\text{mid}}$, we have $(d_S(v) - \bar{d}_S(T))^2 \geq (.05\beta)^2$. Hence, by definition of γ in [Lemma 5.12](#) and the non-negativity of the terms in its numerator,

$$\begin{aligned} \gamma &\geq \frac{\frac{1}{2} \sum_{v \in T} (d_S(v) - \bar{d}_S(T))^2}{m_S \beta} \geq \frac{|T \setminus V_{\text{mid}}| \cdot (.05\beta)^2}{2m_S \beta} = \frac{|T \setminus V_{\text{mid}}| \cdot (.05\beta)^2}{2|S| \bar{d}(S) \beta} \stackrel{\text{Claim 5.13}}{\geq} \frac{|T \setminus V_{\text{mid}}| \cdot (.05\beta)^2}{|S| (1 + 3\delta) \beta^2} \\ &\geq (1 - 4\delta)(.05)^2 \frac{|T \setminus V_{\text{mid}}|}{|S|} \geq \frac{1}{414} \cdot \frac{|T \setminus V_{\text{mid}}|}{|S|} \quad (\text{Since } \delta < 1/60.) \\ &\geq \frac{1}{414} \cdot \frac{600\delta\mu(G)}{(\frac{2}{3} + 3\delta)\mu(G)} \quad (\text{By Claim 5.15 and the assumption that } |T \setminus V_{\text{mid}}| > 600\delta\mu(G).) \\ &> 2\delta. \quad (\text{Since } \delta < 1/60.) \end{aligned}$$

But this contradicts [Claim 5.16](#) that $\gamma \leq 2\delta$, completing the proof. \square

Claim 5.19. *If $\mu(H) < (\frac{2}{3} + \delta)\mu(G)$, then $|W \setminus V_{\text{low}}| \leq 33\sqrt{\delta}\mu(G)$.*

Proof. First, we claim that $|\hat{T}| < 22\delta|S|/\alpha$. Suppose for contradiction that $|\hat{T}| \geq 22\delta|S|/\alpha$. By definition of γ in [Lemma 5.12](#) and the non-negativity of the terms in its numerator, we have

$$\begin{aligned} \gamma &\geq \frac{\frac{1}{4} |\hat{T}| \alpha \bar{d}_S(T)^2}{m_S \beta} = \frac{|\hat{T}| \alpha \bar{d}_S(T)^2}{4|S| \bar{d}(S) \beta} \stackrel{\text{Claims 5.13 and 5.14}}{\geq} \frac{|\hat{T}| \alpha ((1 - 6\delta)\beta/2)^2}{4|S| ((1 + 3\delta)\beta/2) \beta} = \frac{1}{8} \cdot \frac{\alpha(1 - 6\delta)^2}{(1 + 3\delta)} \cdot \frac{|\hat{T}|}{|S|} \\ &\geq \frac{1}{8} \alpha (1 - 15\delta) \cdot \frac{|\hat{T}|}{|S|} \stackrel{(0 < \delta < 1/60)}{\geq} \frac{\alpha}{11} \cdot \frac{\hat{T}}{|S|} \stackrel{(|\hat{T}| \geq 22\delta|S|/\alpha)}{\geq} 2\delta. \end{aligned}$$

But this contradicts [Claim 5.16](#) that $\gamma \leq 2\delta$, therefore we must have $|\hat{T}| < 22\delta|S|/\alpha$.

Now take a vertex $v \in W \setminus V_{\text{low}}$. We have $d(v) \geq .2\beta$ by definition of V_{low} . This means that $m_W \geq .2\beta|W \setminus V_{\text{low}}|$. On the other hand, since any vertex has degree at most β in a $(\beta, (1 - \varepsilon)\beta)$ -EDCS, and for any vertex $v \in T \setminus \hat{T}$ we have $d(v) \leq (1 + \alpha)d_S(v)$ by definition of \hat{T} , we get

$$m_W = \sum_{v \in T} d_W(v) = \sum_{v \in T} (d(v) - d_S(v)) \leq \sum_{v \in T \setminus \hat{T}} \alpha d_S(v) + \sum_{v \in \hat{T}} \beta \leq \alpha m_S + \beta |\hat{T}|.$$

Additionally, since $m_S = |S|\bar{d}(S)$ we get by [Claim 5.13](#) that $m_S \leq |S|(1 + 3\delta)\beta/2$. Combined with our earlier lower bound on m_W , this implies that

$$.2\beta|W \setminus V_{\text{low}}| \leq \alpha|S|(1 + 3\delta)\beta/2 + \beta|\hat{T}| \quad \Leftrightarrow \quad |W \setminus V_{\text{low}}| \leq 5(\alpha(1 + 3\delta)|S|/2 + |\hat{T}|).$$

Using our earlier bound of $|\hat{T}| < 22\delta|S|/\alpha$, we get that

$$|W \setminus V_{\text{low}}| \leq 5(\alpha(1 + 3\delta)|S|/2 + 22\delta|S|/\alpha).$$

Since $\alpha = 3\sqrt{\delta}$ by definition and $0 < \delta < 1/60$, this implies $|W \setminus V_{\text{low}}| < 46\sqrt{\delta}|S|$. Given the upper bound of $|S| \leq (\frac{2}{3} + 3\delta)\mu(G)$ in [Claim 5.15](#) and since $\delta < 1/60$, we get $|W \setminus V_{\text{low}}| \leq 46(\frac{2}{3} + 3\delta)\mu(G) \leq 33\sqrt{\delta}\mu(G)$. \square

As discussed earlier, [Claims 5.17 to 5.19](#) together imply [Lemma 5.4](#) for bipartite graphs.

6 Beating Half for General Graphs

Our discussion of [Section 4](#) crucially relied on the graph G being bipartite. In this section, we prove [Theorem 1](#) for general graphs. Our main result of this section is the following semi-dynamic algorithm, akin to [Lemma 4.2](#), but now for general graphs.

Lemma 6.1. *For any $\varepsilon > 0$ and any n -vertex fully dynamic graph G , there is a (randomized) data structure \mathcal{A} that takes (poly log n) worst-case update-time, and upon being queried takes $\tilde{O}(n/\varepsilon^5)$ time to produce a number $\tilde{\mu}$ such that $.5018 \cdot \mu(G) - \varepsilon n \leq \mathbf{E}[\tilde{\mu}] \leq \mu(G)$.*

The proof of [Theorem 1](#) for general graphs follows from [Lemma 6.1](#):

Proof of [Theorem 1](#) for general graphs. Follows by plugging the data structure \mathcal{A} of [Lemma 6.1](#) as the data structure \mathcal{A} in [Lemma 4.1](#) and choosing sufficiently small ε such that $.501 \geq .5018 - \varepsilon$. \square

6.1 The Semi-Dynamic Algorithm (Proof of [Lemma 6.1](#))

The data structures that we maintain are exactly the same as those in [Section 4](#). Namely, we maintain the adjacency matrix of the graph G and a $(1/2 - \varepsilon)$ -approximate matching M of G in (poly log n) worst-case update-time [[18](#), [13](#)] against adaptive adversaries. It remains to show how to produce the number $\tilde{\mu}$ in $\tilde{O}(n/\varepsilon^5)$ time using these data structures, which is what we focus on in the rest of this section.

The first idea is to define a (random) bipartite subgraph $G_B = (L, R, E_B)$ of G . We construct G_B in a way that all edges of M belong to E_B . Specifically, for each edge in M we put one of

its endpoints in L and the other in R arbitrarily. The rest of the vertices (i.e., $V \setminus V(M)$) are independently and uniformly added either to L or R . An edge belongs to G_B iff it belongs to G and it has one endpoint in L and one in R . We note that a similar randomization was used in [14].

Observation 6.2. M is a maximal matching of G_B .

Proof. Holds since $M \subseteq E_B$, M is a maximal matching of G , and G_B is a subgraph of G . \square

Note that if we had $\mu(G_B) = \mu(G)$, we could simply run the algorithm of Section 4 on graph G_B . However, $\mu(G_B)$ can be smaller than $\mu(G)$, and so additional ideas are needed.

The following Algorithm 3 is analogous to Algorithm 1 of Section 4.

Algorithm 3:

1. Let M and $G_B = (L, R, E_B)$ be as above.
2. Let $M' \subseteq M$ include each edge of M independently with probability $p = .03$.
3. Let $V' := V(M')$ and $U := V \setminus V(M)$.
4. Let $V'_R := V' \cap R$, $V'_L := V' \cap L$, $U_R := U \cap R$, $U_L := U \cap L$.
5. Let $H_R := G_B[V'_R, U_L]$ be the induced bipartite subgraph of G_B between V'_R and U_L .
6. Let $H_L := G_B[V'_L, U_R]$ be the induced bipartite subgraph of G_B between V'_L and U_R .
7. For $e = (u, v)$, $v \in R$, $u \in L$, let $q_e := \Pr_\pi[v \in \text{GMM}(H_R, \pi), u \in \text{GMM}(H_L, \pi) \mid M']$ for a random permutation π .
8. Let $\ell := |\text{GMM}(G[V \setminus V(M)], \pi)|$ for any arbitrary permutation π .
9. Return $\tilde{\mu}' := |M| + \max\{\ell, \sum_{e \in M'} q_e\}$.

To implement Algorithm 3, we use the following Proposition 6.3, which builds on the techniques developed in [11]. See Appendix A for the proof.

Proposition 6.3 ([11]). *Let $G = (V, E)$ be an n -vertex graph to which we have adjacency matrix query access and let $K \subseteq V$ be an arbitrary subset. For any $\varepsilon > 0$ and $v \in K$ chosen u.a.r., there is an algorithm that succeeds with probability $1 - \varepsilon n/|K|$ and in $\tilde{O}(n^2/(\varepsilon|K|))$ expected time returns whether v is matched by $\text{GMM}(G, \pi)$, where π is a u.a.r. permutation of E drawn by the algorithm. The probabilistic statements depend both on the randomization of π and the randomization of $v \sim K$.*

Lemma 6.4. *For any $\varepsilon > 0$, there is an algorithm that w.h.p. takes $\tilde{O}(n/\varepsilon^5)$ time and returns a number $\tilde{\mu}''$ such that $\tilde{\mu}' - \varepsilon n \leq \tilde{\mu}'' \leq \tilde{\mu}'$. Here $\tilde{\mu}'$ is the output of Algorithm 3.*

Proof. Since M is given, we can construct L , R , M' , V'_R , V'_L , U_R , and U_L in $O(n)$ time and also store for each vertex to which one of these sets it belongs. However, we will not compute H_R or H_L explicitly as they may have $\Omega(n^2)$ edges. Note, however, that any adjacency matrix query to H_R or H_L can be answered in $O(1)$ time since we have adjacency matrix access to G and explicitly have these graphs' vertex sets stored.

Let us condition on the outcome of M' for the rest of the proof. Observe that $\sum_{e \in M'} q_e \leq \sum_{e \in M'} 1 = |M'|$. So if $|M'| \leq \varepsilon n$ then returning $\tilde{\mu}'' = |M|$ proves the lemma. Thus, let us assume $|M'| > \varepsilon n$. We do not know how to compute q_e for every edge in M' . Instead, we show how to estimate the value of the sum $\sum_{e \in M'} q_e$.

Let $k = 48 \log n / \varepsilon^2$. For any $i \in [k]$, we pick an edge $e_i = (v_i, u_i)$ from M' each uniformly at random (with replacement), assuming w.l.o.g. that $u_i \in L, v_i \in R$. For any $i \in [k]$, we run [Proposition 6.3](#) once on graph H_R for vertex v_i and once on graph H_L for vertex u_i for error parameter $\varepsilon' = \varepsilon^2/2$. We then let X_i be the indicator of the event that both v_i and u_i are returned to be matched by [Proposition 6.3](#). Since H_R and H_L are vertex disjoint by construction, the dependence of u_i and v_i (in that they are both endpoints of the same edge in M') does not affect the guarantees of [Proposition 6.3](#). In particular, u_i (resp. v_i) is still a vertex chosen u.a.r. from V'_L (resp. V'_R).

Since $|V'_R| = |V'_L| = |M'| \geq \varepsilon n$, the set K in our call to [Proposition 6.3](#) has size εn at least. Hence, [Proposition 6.3](#) takes $\tilde{O}(n^2/(\varepsilon' \varepsilon n)) = \tilde{O}(n/\varepsilon^3)$ expected time for each $i \in [k]$, and has success probability $1 - \varepsilon' n/(\varepsilon n) = 1 - \varepsilon/2$. Since we call it k times, the total time-complexity is $\tilde{O}(nk/\varepsilon^2) = \tilde{O}(n/\varepsilon^5)$ in expectation. We will show later how to turn this into high probability.

Since, as discussed, our call to [Proposition 6.3](#) has failure probability $\leq \varepsilon/2$, we get that

$$\mathbf{E}[X_i] = \frac{1}{|M'|} \sum_{(u,v) \in M'} (q_e \pm \varepsilon/2) = \left(\frac{1}{|M'|} \sum_{(u,v) \in M'} q_e \right) \pm \varepsilon/2. \quad (8)$$

Define $X := \sum_i X_i$ and $Q := \frac{X|M'|}{k}$. Since the X_i 's are independent (as each call to [Proposition 6.3](#) generates a fresh random permutation), we get from the Chernoff bound that with probability $1 - 2n^{-4}$, $X = \mathbf{E}[X] \pm \sqrt{12 \mathbf{E}[X] \log n}$. As such, we get that w.h.p.

$$\begin{aligned} Q &= \frac{(\mathbf{E}[X] \pm \sqrt{12 \mathbf{E}[X] \log n})|M'|}{k} = \frac{(k \mathbf{E}[X_i] \pm \sqrt{12k \mathbf{E}[X_i] \log n})|M'|}{k} \\ &= \mathbf{E}[X_i]|M'| \pm .5\varepsilon|M'| \quad (\text{Since } \mathbf{E}[X_i] \leq 1 \text{ and } k = 48 \log n / \varepsilon^2.) \\ &= \sum_{(u,v) \in M'} q_e \pm \varepsilon|M'| \quad (\text{By (8).}) \\ &= \sum_{(u,v) \in M'} q_e \pm .5\varepsilon n. \quad (\text{Since } |M'| \leq n/2.) \end{aligned}$$

Note also that ℓ is simple to approximate within a $(1 + \varepsilon)$ factor using the algorithm of [\[11\]](#) as black-box. Therefore, returning $\tilde{\mu}'' = |M| + \max\{\ell, Q\} - .5\varepsilon n$ guarantees $\tilde{\mu}' - \varepsilon n \leq \tilde{\mu}'' \leq \tilde{\mu}'$ w.h.p.

Since the expected running time is $\tilde{O}(n/\varepsilon^5)$, by Markov's inequality the algorithm terminates in $2 \times \tilde{O}(n/\varepsilon^5)$ time with probability at least $1/2$. Thus, we can run $O(\log n)$ independent instances of the algorithm, and return the output of the one that first terminates. This way, our algorithm w.h.p. terminates in $\tilde{O}(n/\varepsilon^5)$ time. Since the guarantee on $\tilde{\mu}''$ holds with probability $1 - 1/\text{poly}(n)$, it should hold for all $O(\log n)$ instances (and so the one that first terminates) still w.h.p. \square

Next, we turn to analyze the approximation ratio of [Algorithm 3](#).

[Proposition 4.5](#), which was used in the proof of [Lemma 4.6](#), only gives a lower bound on the size of the matching. For our discussion of this section, however, we need a more fine-tuned bound guaranteed by the following proposition of [\[14\]](#).

Proposition 6.5 ([\[14, Lemma 5.2\]](#)). *Let $0 < p \leq 1$, let $G = (A, B, E)$ be a bipartite graph, let $A' \subseteq A$ include each vertex of A independently with probability p , and let H be the induced subgraph of G on vertex-set $A' \cup B$. Fix an arbitrary permutation π over the edge-set of H and fix an*

arbitrary matching M of G . Let X be the number of edges in M whose endpoint in A is matched in $\text{GMM}(H, \pi)$; then

$$\mathbf{E}_{A'}[X] \geq p(|M| - 2p|A|).$$

Lemma 6.6. For [Algorithm 3](#), it holds that $\mathbf{E}[\tilde{\mu}'] \geq .5018\mu(G)$.

Proof. Fix an arbitrary maximum matching M^* of G . Observe that there are exactly $\mu(G) - |M^*|$ augmenting paths in $M \oplus M^*$ for M . Denoting the number of length one augmenting paths in $M \oplus M^*$ by L_1 , there are exactly $\mu(G) - |M| - L_1$ augmenting paths of length at least three. Each of these augmenting paths has exactly two (endpoint) edges that have one vertex matched in M and one endpoint unmatched in M . Putting together these edges, we obtain a matching Z with $2(\mu(G) - |M| - L_1)$ edges, all of which belong to M^* . The number of length two components in $M \oplus Z$ is at most $|M|$. The rest of the components are length three augmenting paths for M . Thus, there are at least $|Z| - |M| = 2\mu(G) - 3|M| - 2L_1$ length three augmenting paths for M in $M \oplus Z$. Let $P = (a, u, v, b)$ be one of these length-three augmenting paths with $(u, v) \in M$ and $(a, u), (v, b) \in M^*$. Suppose w.l.o.g. that we assign $u \in L$ and $v \in R$ in [Algorithm 3](#). Then P remains an augmenting path in G_B if $a \in R$ and $b \in L$, which happens with probability $1/4$. Under this event, we say P *survives* to G_B . Let \mathcal{P} be the set of all the length three augmenting paths in $M^* \oplus M$ that survive to G_B , and note that

$$\mathbf{E}_{L,R} |\mathcal{P}| \geq \frac{1}{4}(2\mu(G) - 3|M| - 2L_1). \quad (9)$$

Define

$$\begin{aligned} M_L^* &:= \{(a, u) \in M^* \mid u \in L, a \in R, (a, u, \cdot, \cdot) \in \mathcal{P}\}, \\ M_R^* &:= \{(v, u) \in M^* \mid v \in R, b \in L, (\cdot, \cdot, v, b) \in \mathcal{P}\}, \\ M_{\mathcal{P}} &:= \{(u, v) \in M \mid (\cdot, u, v, \cdot) \in \mathcal{P}\}. \end{aligned}$$

Furthermore, define

$$F_L := G_B[V(M) \cap L, U_R], \quad F_R := G_B[V(M) \cap R, U_L].$$

Let π be any arbitrary permutation of the edges in E . Let X_L (resp. X_R) denote the number of vertices in $V(M_L^*) \cap L$ that are matched by $\text{GMM}(H_L, \pi)$ (resp. $\text{GMM}(H_R, \pi)$). Noting that H_L is an induced subgraph of F_L including each of its vertices in its $V(M) \cap L$ part independently from each other with probability p , we can apply [Proposition 6.5](#) (on graph F_L fixing matching M_L^*) to obtain that

$$\mathbf{E}_{M'}[X_L] \geq p(|M_L^*| - 2p|V(M) \cap L|) = p(|\mathcal{P}| - 2p|M|). \quad (10)$$

With essentially the same proof, we also get that

$$\mathbf{E}_{M'}[X_R] \geq p(|M_R^*| - 2p|V(M) \cap R|) = p(|\mathcal{P}| - 2p|M|). \quad (11)$$

Now let $M'_{\mathcal{P}}$ be the edges in $M_{\mathcal{P}}$ that also belong to subsample M' of M in [Algorithm 3](#). Let Y be the number of edges $(u, v) \in M'_{\mathcal{P}}$ where u is matched by $\text{GMM}(H_L, \pi)$ and v is matched by $\text{GMM}(H_R, \pi)$. We have

$$Y \geq |M'_{\mathcal{P}}| - (|M'_{\mathcal{P}}| - X_L) - (|M'_{\mathcal{P}}| - X_R) = X_R + X_L - |M'_{\mathcal{P}}|.$$

Taking expectation over M' , plugging (10) and (11), and noting that $\mathbf{E}|M'_{\mathcal{P}}| = p|M_{\mathcal{P}}| = p|\mathcal{P}|$, we get

$$\mathbf{E}_{M'}[Y] \geq 2p(|\mathcal{P}| - 2p|M|) - p|\mathcal{P}| = p|\mathcal{P}| - 4p^2|M|.$$

Further taking expectation over the randomization of L, R , we get that

$$\begin{aligned} \mathbf{E}_{M',L,R}[Y] &\geq p \mathbf{E}_{L,R}|\mathcal{P}| - 4p^2|M| \stackrel{(9)}{\geq} p \cdot \frac{1}{4}(2\mu(G) - 3|M| - 2L_1) - 4p^2|M| \\ &= \frac{p}{2}\mu(G) - \left(\frac{3p}{4} + 4p^2\right)|M| - \frac{p}{2}L_1. \end{aligned}$$

From this, we get that

$$\mathbf{E} \left[\sum_{e \in M'} q_e \right] \geq \mathbf{E} \left[\sum_{e \in M'_{\mathcal{P}}} q_e \right] = \mathbf{E}[Y] \geq \frac{p}{2}\mu(G) - \left(\frac{3p}{4} + 4p^2\right)|M| - \frac{p}{2}L_1. \quad (12)$$

Therefore, we have

$$\begin{aligned} \mathbf{E}[\tilde{\mu}'] &= |M| + \max \left\{ \ell, \mathbf{E} \left[\sum_{e \in M'} q_e \right] \right\} \\ &\geq |M| + \max \left\{ \frac{L_1}{2}, \mathbf{E} \left[\sum_{e \in M'} q_e \right] \right\} \\ &\text{(Since } \ell \text{ is the size of a maximal matching in } G[V \setminus V(M)] \text{ and } \mu(G[V \setminus V(M)]) \geq L_1.) \\ &\geq |M| + \max \left\{ \frac{L_1}{2}, \frac{p}{2}\mu(G) - \left(\frac{3p}{4} + 4p^2\right)|M| - \frac{p}{2}L_1 \right\} \quad \text{(By (12).)} \\ &= \max \left\{ |M| + \frac{L_1}{2}, \frac{p}{2}\mu(G) + \left(1 - \frac{3p}{4} - 4p^2\right)|M| - \frac{p}{2}L_1 \right\} \\ &\geq (1 - \varepsilon) \max \left\{ \frac{\mu(G)}{2} + \frac{L_1}{2}, \left(\frac{1}{2} + \frac{1}{8}p - 2p^2\right)\mu(G) - \frac{p}{2}L_1 \right\} \\ &\quad \text{(Since } |M| \geq (1 - \varepsilon)\mu(G)/2 \text{ and } \left(1 - \frac{3p}{4} - 4p^2\right) = 0.9739 > 0 \text{ as } p = 0.03.) \\ &\geq (1 - \varepsilon) \max \left\{ \frac{\mu(G)}{2} + \frac{L_1}{2}, .5019\mu(G) - .015L_1 \right\} \quad \text{(Since } p = .03.) \\ &\geq (1 - \varepsilon).5018\mu(G). \quad \text{(This holds for all values of } L_1.) \end{aligned}$$

This completes the proof. \square

Lemma 6.7. For Algorithm 3, it holds with probability 1 that $\tilde{\mu}' \leq \mu(G)$.

Proof. Condition on the outcome of M' in Algorithm 3. Then run the process of constructing matchings $\text{GMM}(H_R, \pi)$ and $\text{GMM}(H_L, \pi)$ for a random π . Define

$$\mathcal{Y} = \{(a, u, v, b) \mid (a, u) \in \text{GMM}(H_L, \pi), (u, v) \in M, (v, b) \in \text{GMM}(H_R, \pi)\}.$$

Note that \mathcal{Y} is a collection of length-three augmenting paths for M . Hence, we can apply all of them at the same time on M . This implies that $\mu(G) \geq |M| + |\mathcal{Y}|$. Moreover, we have

$$\mathbf{E}_{\pi}[|\mathcal{Y}| \mid M'] = \sum_{e \in M'} \Pr[v \in \text{GMM}(H_R, \pi), u \in \text{GMM}(H_L, \pi) \mid M'] = \sum_{e \in M'} q_e.$$

Given this expected value, there must be a choice of π with $|\mathcal{Y}| \geq \sum_{e \in M'} q_e$. This suffices to show

$$\mu(G) \geq |M| + \sum_{e \in M'} q_e = \tilde{\mu}'. \quad \square$$

We are now ready to complete the proof of [Lemma 6.1](#).

Proof of Lemma 6.1. The data structures that we store, as discussed, take only (poly log n) worst-case time to maintain against an adaptive adversary. When the algorithm is queried, we return the output $\tilde{\mu}''$ of [Lemma 6.4](#). It takes $\tilde{O}(n/\varepsilon^5)$ time to produce this by [Lemma 6.4](#), which is the desired query time of [Lemma 6.1](#). Moreover, for the approximation ratio, we have

$$.5018\mu(G) - \varepsilon n \stackrel{\text{Lemma 6.6}}{\leq} \mathbf{E}[\tilde{\mu}'] - \varepsilon n \stackrel{\text{Lemma 6.4}}{\leq} \mathbf{E}[\tilde{\mu}''] \stackrel{\text{Lemma 6.4}}{\leq} \mathbf{E}[\tilde{\mu}'] \stackrel{\text{Lemma 6.7}}{\leq} \mu(G).$$

This completes the proof of [Lemma 6.1](#). □

References

- [1] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.
- [2] Moab Arar, Shiri Chechik, Sarel Cohen, Cliff Stein, and David Wajc. 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*, pages 7:1–7:16, 2018.
- [3] Sepehr Assadi and Soheil Behnezhad. 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)*, volume 198 of *LIPICs*, pages 19:1–19:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [4] Sepehr Assadi and Aaron Bernstein. Towards a Unified Theory of Sparsification for Matching Problems. In *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASICs*, pages 11:1–11:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- [5] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 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*, pages 99–116. ACM, 2017.
- [6] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The Stochastic Matching Problem with (Very) Few Queries. *ACM Trans. Economics and Comput.*, 7(3):16:1–16:19, 2019.
- [7] Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. *CoRR*, abs/2207.09354, 2022.
- [8] Surender Baswana, Manoj Gupta, and Sandeep Sen. 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*, pages 383–392. IEEE Computer Society, 2011.

- [9] Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully Dynamic Maximal Matching in $O(\log n)$ Update Time (Corrected Version). *SIAM J. Comput.*, 47(3):617–650, 2018.
- [10] Soheil Behnezhad. Improved Analysis of EDCS via Gallai-Edmonds Decomposition. *CoRR*, abs/2110.05746, 2021.
- [11] Soheil Behnezhad. Time-Optimal Sublinear Algorithms for Matching and Vertex Cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021.
- [12] Soheil Behnezhad and Sanjeev Khanna. 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*, pages 3529–3566. SIAM, 2022.
- [13] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Cliff Stein, and Madhu Sudan. 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*, pages 382–405. IEEE Computer Society, 2019.
- [14] Soheil Behnezhad, Jakub Lacki, and Vahab S. Mirrokni. 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*, pages 2492–2508. SIAM, 2020.
- [15] Soheil Behnezhad, Mohammad Roghani, Aviad Rubinfeld, and Amin Saberi. Beating Greedy Matching in Sublinear Time. 2023.
- [16] Aaron Bernstein and Cliff Stein. Fully Dynamic Matching in Bipartite Graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2015.
- [17] Aaron Bernstein and Cliff Stein. 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*, pages 692–711. SIAM, 2016.
- [18] Aaron Bernstein, Sebastian Forster, and Monika Henzinger. 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*, pages 1899–1918, 2019.
- [19] Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A Framework for Dynamic Matching in Weighted Graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021, to appear*, 2021.
- [20] Sayan Bhattacharya and Peter Kiss. 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)*, pages 27:1–27:14, 2021.

- [21] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. 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*, pages 398–411. ACM, 2016.
- [22] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. 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*, pages 470–489. SIAM, 2017.
- [23] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic Fully Dynamic Data Structures for Vertex Cover and Matching. *SIAM J. Comput.*, 47(3):859–887, 2018.
- [24] Sayan Bhattacharya, Peter Kiss, Thatchaphol Saranurak, and David Wajc. Dynamic Matching with Better-than-2 Approximation in Polylogarithmic Update Time. 2023.
- [25] Moses Charikar and Shay Solomon. 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*, pages 33:1–33:14, 2018.
- [26] Yu Chen, Sampath Kannan, and Sanjeev Khanna. Sublinear Algorithms and Lower Bounds for Metric TSP Cost Estimation. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 30:1–30:19, 2020.
- [27] Søren Dahlgaard. On the Hardness of Partially Dynamic Graph Problems and Connections to Diameter. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 48:1–48:14, 2016.
- [28] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012.
- [29] Fabrizio Grandoni, Chris Schwiegelshohn, Shay Solomon, and Amitai Uzrad. 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*, pages 12–23. SIAM, 2022.
- [30] Manoj Gupta and Richard Peng. Fully Dynamic $(1 + \epsilon)$ -Approximate Matchings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 548–557. IEEE Computer Society, 2013.
- [31] Philip Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 1(1): 26–30, 1935.
- [32] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30, 2015.

- [33] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.
- [34] Michael Kapralov. 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*, pages 1874–1893. SIAM, 2021.
- [35] Peter Kiss. 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*, volume 215 of *LIPICs*, pages 94:1–94:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [36] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012.
- [37] Ofer Neiman and Shay Solomon. 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*, pages 745–754, 2013.
- [38] Huy N. Nguyen and Krzysztof Onak. Constant-Time Approximation Algorithms via Local Improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336, 2008.
- [39] Krzysztof Onak and Ronitt Rubinfeld. 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*, pages 457–464. ACM, 2010.
- [40] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A Near-Optimal Sublinear-Time Algorithm for Approximating the Minimum Vertex Cover Size. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1123–1131, 2012.
- [41] Michal Parnas and Dana Ron. Approximating the Minimum Vertex Cover in Sublinear Time and a Connection to Distributed Algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.
- [42] Mohammad Roghani, Amin Saberi, and David Wajc. 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*, volume 215 of *LIPICs*, pages 111:1–111:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [43] Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 118–126. SIAM, 2007.
- [44] Shay Solomon. 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*, pages 325–334. IEEE Computer Society, 2016.

- [45] Shay Solomon. Local algorithms for bounded degree sparsifiers in sparse graphs. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 52:1–52:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- [46] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic Matrix Inverse: Improved Algorithms and Matching Conditional Lower Bounds. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 456–480, 2019.
- [47] David Wajc. Rounding Dynamic Matchings Against an Adaptive Adversary. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 194–207. ACM, 2020.
- [48] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 225–234. ACM, 2009.

A Needed Sublinear Algorithms From [11]

In our proofs, we used [Propositions 6.3](#) and [4.4](#) which are implied by a result of the author in [11]. In this section, we prove why these propositions follow from [11].

In its Section 4, [11] gives an algorithm that works in the adjacency list model. The algorithm is then adapted to the adjacency matrix model using a reduction that is provided in [11, Section 5]. The reduction works as follows. Let $G = (V, E)$ be the graph to which we have adjacency matrix access. A graph $H = (V_H, E_H)$ is then defined based on G in such a way that any adjacency list query to H can be answered with a single adjacency matrix query to G . The graph H has two vertex disjoint copies $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of G with a number of edges added between G_1 and G_2 . Additionally each vertex in V_2 has $10n/\varepsilon$ leaves adjacent to it where $n = |V|$.

Partition V_2 into V_2' and $V_2'' := V_2 \setminus V_2'$ such that V_2' includes a vertex $v \in V_2$ iff the lowest rank edge of v according to π is to a vertex in $V_1 \cup V_2$ (i.e., not to a leaf of v). Let $A := \text{GMM}(H[V_1], \pi) = \text{GMM}(G, \pi)$, $B := \text{GMM}(H, \pi) \cap ((V_1 \times V_1) \cup (V_1 \times V_2'))$, and $C = B \cap (V_1 \times V_1)$.

Observation A.1. *Any vertex in V belongs to V_2' with probability at most ε .*

Proof. For a random permutation π over E_H , the lowest rank edge of each vertex $v \in V_2$ goes to $V_1 \cup V_2$ with probability at most $\frac{2n}{10n/\varepsilon + 2n} < \varepsilon$. \square

Define the *match-status* of a vertex v in a matching M to be the indicator $\mathbf{1}(v \in V(M))$.

Claim A.2. *For a random π , there are, in expectation, at most $3\varepsilon n$ vertices in V_1 whose match-status for A is different from C .*

Proof. Define $S = A \oplus B$. We claim that any component of S must be a path with at least one endpoint in V_2' . To see this, take an arbitrary component C of S . The lowest rank edge $e \in C$ must have an endpoint in V_2' or else we should have $e \in A, B$ as e belongs to both $H[V_1]$ and H and has no lower rank edge in either A or B , which contradicts $e \in S$. This edge e should be an endpoint

of the path C , since any vertex in V_2' has degree at most 1 in S (as it cannot be matched in A). From this, we get that the total number of connected components in S is at most $|V_2'|$. As a result, there are at most $2|V_2'|$ vertices with a different match-status in A and B .

Since C is a sub-matching of B , excluding its edges that go from V_1 to V_2' , any vertex v with a different match-status in A and C must either have a different match-status in A and B , or it should be matched in both A and B , but its match in B is through a $V_1 \times V_2'$ edge. So, in total, at most $2|V_2'| + |V_2'|$ vertices may have a different match-status in A and C . This completes the proof since $\mathbf{E}|V_2'| \leq \varepsilon n$ by [Observation A.1](#). \square

Proof of [Proposition 4.4](#). It is shown in the proof of [[11](#), Lemma 5.4] that one can w.h.p. find an estimate of $\mathbf{E}|C|$ (denoted $M_1(\pi)$ in [[11](#)]) with an additive error of $O(\varepsilon n)$ in $\tilde{O}(n/\varepsilon^3)$ time. [Claim A.2](#) above shows that $\mathbf{E}|C| = \mathbf{E}|A| \pm O(\varepsilon n) = \mathbf{E}|\text{GMM}(G, \pi)| \pm O(\varepsilon n)$. Subtracting a sufficiently large additive factor of $O(\varepsilon n)$ from the output gives [Proposition 4.4](#). \square

Proof of [Proposition 6.3](#). It is shown in [[11](#), Eq (9)] that for any arbitrary vertex $u \in V_H$, one can determine which edge of u (if any) belongs to $\text{GMM}(H, \pi)$ in time $T(u, \pi)$ where $\sum_{u \in V_H} \mathbf{E}_\pi[T(u, \pi)] = \tilde{O}(n^2/\varepsilon)$. Thus, for $v \sim K$ chosen u.a.r., $\mathbf{E}_{\pi, v}[T(v, \pi)] = \frac{1}{|K|} \sum_{v \in K} \mathbf{E}_\pi[T(v, \pi)] = O(n^2/(\varepsilon|K|))$, which is the claimed time. Since the edge of v in $\text{GMM}(H, \pi)$ is also given (if any) by this process, we can determine if it belongs to C . By [Claim A.2](#), at most $3\varepsilon n$ vertices in V_1 (and so in K) have a different match-status in C and $A = \text{GMM}(G, \pi)$. Thus, probability (taken over $v \sim K$) of choosing a vertex that is not among these εn out of K is at least $(|K| - 3\varepsilon n)/|K| = 1 - O(\varepsilon n/|K|)$. \square

B Proof of [Lemma 4.1](#)

In this section, we argue why [Lemma 4.1](#) holds. Let us restate the lemma first.

[Lemma 4.1 \(restated\)](#). *Let G be an n -vertex fully dynamic graph and let $\varepsilon > 0$ be a parameter. Suppose that there is a (randomized) data structure \mathcal{A} (this is the semi-dynamic algorithm) that takes $U(n)$ worst-case time per update to G and, upon being queried, \mathcal{A} produces in $Q(n, \varepsilon)$ time a number $\tilde{\mu}$, such that $\alpha\mu(G) - \varepsilon n \leq \mathbf{E}[\tilde{\mu}] \leq \mu(G)$. Then there is a randomized data structure \mathcal{B} (this is the fully-dynamic algorithm) that maintains a number $\tilde{\mu}'$ such that at any point during the updates, w.h.p., $(\alpha - \varepsilon)\mu(G) \leq \tilde{\mu}' \leq \mu(G)$. Algorithm \mathcal{B} takes $O\left(\left(U(n) + \frac{Q(n, \varepsilon^2)}{n}\right) \text{poly}(\log n, 1/\varepsilon)\right)$ worst-case update-time. Moreover, if \mathcal{A} works against adaptive adversaries, so does \mathcal{B} .*

Let us prove the lemma step by step. First, we prove the following lemma which guarantees all the desired properties of algorithm \mathcal{B} , except that instead of a multiplicative approximation, it achieves a multiplicative-additive $(\alpha, \varepsilon n)$ -approximation.

[Lemma B.1](#). *Let algorithm \mathcal{A} be as in [Lemma 4.1](#). There is an algorithm \mathcal{C} that maintains a number $\tilde{\mu}_c$ such that at any point during the updates, it holds w.h.p. that $\alpha\mu(G) - O(\varepsilon n) \leq \tilde{\mu}_c \leq \mu(G)$. Algorithm \mathcal{C} takes $O\left(\left(U(n) + \frac{Q(n, \varepsilon)}{n}\right) \text{poly}(\log n, 1/\varepsilon)\right)$ worst-case update-time. Moreover, if \mathcal{A} works against adaptive adversaries, then so does \mathcal{C} .*

Proof. We run algorithm \mathcal{A} in the background, paying a worst-case update-time of $U(n)$ because of it. We then take the lazy approach. We query \mathcal{A} to produce the estimate $\tilde{\mu}$, return $\tilde{\mu}' := \tilde{\mu} - 2\varepsilon n$ as our output, then we do not change the output for the next εn updates, and repeat the same process. Because every edge update can change the size of the maximum matching by at most one (even against adaptive adversaries), it will hold at all times that $\alpha\mu(G) - 3\varepsilon n \leq \mathbf{E}[\tilde{\mu}'] \leq \mu(G) - \varepsilon n$.

Moreover, because we query algorithm \mathcal{A} every εn updates, the overall amortized update time of the algorithm is $O(U(n) + \frac{Q(n,\varepsilon)}{\varepsilon n})$. It remains to (i) turn the amortized update-time bound to worst-case, and (ii) turn the expected approximation bound to a high probability bound.

Let us address (i) first. This can be done using a well-known ‘spreading’ idea (see [30] for more details). Instead of executing the oracle of algorithm \mathcal{A} over one update, we spread it over multiple updates. More precisely, we spread the $Q(n, \varepsilon)$ time needed for the oracle over $\varepsilon n/2$ updates, each performing $2\frac{Q(n,\varepsilon)}{\varepsilon n}$ operations of it. When the process finishes, we update our solution as before, and immediately start spreading the next call to the oracle.

We now address (ii). To turn the approximation guarantee into a high probability bound, we simply run $O(\log n)$ independent instances of the algorithm above, and return the average of these $O(\log n)$ outputs as our output. Note that if algorithm \mathcal{A} works against adaptive adversaries, then so should all of these $O(\log n)$ instances and thus our algorithm as well. \square

Let us now show how we can get rid of the additive εn error. First, note that if the maximum matching size is guaranteed to be $\Omega(n)$ at all times, then a multiplicative-additive $(\alpha, \varepsilon n)$ -approximation for it is indeed a multiplicative $(\alpha - O(\varepsilon))$ -approximation. Indeed up to a poly $\log n$ increase in the update-time, this assumption comes w.l.o.g. due to a ‘vertex sparsification’ idea of the literature [6, 35]. In particular, suppose $\mu(G) \ll n$ and suppose that we know $\mu(G)$. Then the idea is to randomly contract the vertices into $O(\mu(G)/\varepsilon)$ vertices, then remove self-loops and parallel edges. This way, it is not hard to see that the resulting graph still has a matching of size at least $(1 - \varepsilon)\mu(G)$ in expectation, but has much fewer vertices. The problem with this approach is that it only works against oblivious adversaries since an adaptive adversary may insert edges among the contracted nodes. However, [35] showed that taking poly $\log n$ of these vertex sparsified subgraphs are resilient against adaptive adversaries also. In particular, the following was proved in [35], which combined with Lemma B.1 implies Lemma 4.1.

Proposition B.2 ([35, Corollary 4.101]). *If there is a dynamic algorithm for maintaining an $(\alpha, \delta n)$ -approximate maximum matching for dynamic graphs in update time $O(T(n, \delta))$ then there is a randomized algorithm for maintaining an $(\alpha - \varepsilon)$ -approximate maximum matching with update time $O(T(n, \varepsilon) \log^4 n / \varepsilon^8)$ which works against adaptive adversaries given the underlying algorithm also does.*

C Basic Facts

Fact C.1. *Let x_1, \dots, x_n be an arbitrary set of reals. Then denoting $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$, it holds that $\sum_{i=1}^n x_i^2 = n\bar{x}^2 + \sum_{i=1}^n (x_i - \bar{x})^2 \geq n\bar{x}^2$.*

Proof. For any $i \in [n]$ denote $y_i = (x_i - \bar{x})$. We have

$$\sum_{i=1}^n x_i^2 = \sum_{i=1}^n (\bar{x} + y_i)^2 = \sum_{i=1}^n (\bar{x}^2 + y_i^2 + 2\bar{x}y_i) = n\bar{x}^2 + \sum_{i=1}^n y_i^2 + 2\bar{x} \sum_{i=1}^n y_i = n\bar{x}^2 + \sum_{i=1}^n y_i^2,$$

where the last equality follows from $\sum_{i=1}^n y_i = \sum_{i=1}^n (x_i - \bar{x}) = \sum_{i=1}^n x_i - \sum_{i=1}^n \bar{x} = 0$. \square