Rounding Dynamic Matchings Against an Adaptive Adversary

David Wajc

Carnegie Mellon University

Abstract

We present a new dynamic matching sparsification scheme. From this scheme we derive a framework for dynamically rounding fractional matchings against *adaptive adversaries*. Plugging in known dynamic fractional matching algorithms into our framework, we obtain numerous randomized dynamic matching algorithms against adaptive adversaries (the first such algorithms). In particular, our two main results are randomized algorithms which work against an adaptive adversary, achieving

1. $(2+\epsilon)$ approximation ratio in polylog worst-case update time.

2. $(2+\epsilon)$ approximation ratio in constant amortized update time.

Similar results were only previously known using randomization against *oblivious* adversaries. In comparison, no algorithms that work against adaptive adversaries were previously known to even guarantee *sub-polynomial* approximation ratio in worst-case *sub-polynomial* update time, or any *sub-linear* approximation ratio in constant amortized update time.

A third result we obtain from our framework are randomized algorithms which work against an adaptive adversary in bipartite graphs, achieving

3. better-than-two approximation ratio in *arbitrarily-small* polynomial update time.

The only such result with arbitrarily-small $\text{poly}(n)$ update time assumes an oblivious adversary. In comparison, previous algorithms which work against adaptive adversaries and guarantee any $(2-\epsilon)$-approximation ratio all require $\Omega(\sqrt[4]{m})$ update time.

Keywords: Dynamic Algorithms, Dynamic Matching, Matching Sparsifier, Adaptive Adversary

1 Introduction

The field of dynamic graph algorithms studies the maintenance of solutions to graph-theoretic problems subject to graph updates, such as edge additions and removals. For any such dynamic problem, a trivial approach is to recompute a solution from scratch following each update, using a static algorithm. Fortunately, significant improvements over this naïve polynomial-time approach are often possible, and many fundamental problems admit *polylogarithmic* update time algorithms. Notable examples include minimum spanning tree and connectivity [46, 47, 51, 69] and spanners [9, 14, 34]. Many such efficient dynamic algorithms rely on randomization and the assumption of a weak, *oblivious* adversary, i.e., an adversary which cannot decide its updates adaptively based on the algorithm's output. As recently pointed out by Nanongkai and Saranurak [58],

> It is a fundamental question whether the true source of power of randomized dynamic algorithms is the randomness itself or in fact the oblivious adversary assumption.

In this work, we address this question for the heavily-studied dynamic matching problem. For this problem, the assumption of an oblivious adversary is known to allow for constant-approximate worst-case polylogarithmic update time algorithms [3, 8, 14, 24, 61, 66].[1] In contrast, all deterministic algorithms with worst-case time guarantees have *polynomial* update time [15, 20, 41, 60, 63]. The main advantage of deterministic algorithms over their randomized counterparts is their robustness to *adaptive* adversaries; i.e., their guarantees even hold for update sequences chosen adaptively. Before outlining our results, we motivate the study of algorithms which are robust to adaptive adversaries by addressing some implications of the oblivious adversary assumption.

Static implications. As Mądry [57] observed, randomized dynamic algorithms' assumption of an oblivious adversary renders them unsuitable for use as a black box for many static applications. For example, [33, 36] show how to approximate multicommodity flows by repeatedly routing flow along approximate shortest paths, where edges' lengths are determined by their current congestion. These shortest path computations can be sped up by a dynamic shortest path algorithm, provided it works against an *adaptive* adversary (since edge lengths are determined by prior queries' outputs). This application has motivated much work on faster *deterministic* dynamic shortest path algorithms [11, 12, 13, 43, 44], as well as a growing interest in faster randomized dynamic algorithms which work against adaptive adversaries [25, 26, 42].

Dynamic implications. The oblivious adversary assumption can also make a dynamic algorithm $\mathcal{A}$ unsuitable for use by other *dynamic* algorithms, even ones which themselves assume an oblivious adversary! For example, for dynamic algorithms that use several copies of $\mathcal{A}$ whose inputs depend on each other's output, the different copies act as adaptive adversaries for one another, as the behavior of copy i can affect that of copy j, which in turn can affect that of copy i. (See [58].)

Faster algorithms that are robust to adaptive adversaries thus have the potential to speed up both static and dynamic algorithms. This motivated Nanogkai et al. [59], who studied dynamic MST, to ask whether there exist algorithms against adaptive adversaries for other well-studied dynamic graph problems, with similar guarantees to those known against oblivious adversaries.

In this paper we answer this latter question affirmatively for the dynamic matching problem, for which we give the first randomized algorithms that are robust to adaptive adversaries (and outperform known deterministic algorithms). As an application of these new algorithms, we present improved dynamic maximum weight matching algorithms.

[1] A dynamic algorithm has *worst-case* update time $f(n)$ if it requires $f(n)$ time for each update. It is said to have *amortized* update time $f(n)$ if it requires $O(t \cdot f(n))$ time for any sequence of t updates. If we assume an oblivious adversary, these time bounds need only hold for sequences chosen before the algorithm's run.

1.1 Our Contributions

Our main contribution is a framework for dynamically rounding fractional matchings against adaptive adversaries. That is, we develop a method which given a dynamically-changing fractional matching (i.e., a point $\vec{x}$ in the fractional matching polytope, $\mathcal{P} \triangleq \{\vec{x} \in \mathbb{R}^m_{\geq 0} \mid \sum_{e \ni v} x_e \leq 1 \ \forall v \in V\}$), outputs a matching M of size roughly equal to the value of the fractional matching, $\sum_e x_e$. This framework allows us to obtain dynamic matching algorithms robust to adaptive adversaries, including adversaries that see the algorithms' **entire state** after each update.

Key to our framework is a novel matching sparsification scheme, i.e., a method for computing a sparse subgraph which approximately preserves the maximum matching size. We elaborate on our sparsification scheme and dynamic rounding framework and their analyses in Section 1.2 and later sections. For now, we discuss some of the dynamic matching algorithms we obtain from applying our framework to various known dynamic fractional matching algorithms.

Our first result (applying our framework to [22]) is a $(2+\epsilon)$-approximate matching algorithm with worst-case polylogarithmic update time against an adaptive adversary.

Theorem 1.1. *For every $\epsilon \in (0, 1/2)$, there exists a (Las Vegas) randomized $(2+\epsilon)$-approximate algorithm with update time* $\text{poly}(\log n, 1/\epsilon)$ *w.h.p. against an* adaptive *adversary.*

All algorithms prior to this work either assume an oblivious adversary or have *polynomial* worst-case update time, for any approximation ratio.

Our second result (applying our framework to [23]) yields (amortized) *constant-time* algorithms with the same approximation ratio as Theorem 1.1, also against an adaptive adversary.

Theorem 1.2. *For every $\epsilon \in (0, 1/2)$, there exists a randomized $(2+\epsilon)$-approximate dynamic matching algorithm with* $\text{poly}(1/\epsilon)$ *amortized update time whose approximation and update time guarantees hold in expectation against an* adaptive *adversary.*

No constant-time algorithms against adaptive adversaries were known before this work, for *any* approximation ratio. A corollary of Theorem 1.2, obtained by amplification, is the first algorithm against adaptive adversaries with logarithmic amortized update time and $O(1)$-approximation w.h.p.

Finally, our framework also lends itself to better-than-two approximation. In particular, plugging in the fractional matching algorithm of [21] into our framework yields $(2-\delta)$-approximate algorithms with *arbitrarily-small* polynomial update time against adaptive adversaries in bipartite graphs.

Theorem 1.3. *For any constant $k \geq 10$, there exists a β_k-approximate dynamic bipartite matching algorithm with expected update time $O(n^{1/k})$ against an* adaptive *adversary, for $\beta_k \in (1, 2)$.*

Similar results were recently achieved for general graphs by [10], though only assuming an oblivious adversary. All other $(2-\delta)$-approximate algorithms are deterministic (and so do not need this assumption), but have $\Omega(\sqrt[4]{m})$ update time.

As a warm-up to our randomized rounding framework, we get a family of deterministic algorithms with arbitrarily-small polynomial update time, with the following time-approximation tradeoff.

Theorem 1.4. *For any $K > 1$, there exists a deterministic $O(K)$-approximate matching algorithm with worst-case $\tilde{O}(n^{1/K})$ update time.*

These algorithms include the first deterministic constant-approximate algorithms with any $o(\sqrt[4]{m})$ worst-case update time. They also include the first deterministic $o(\log n)$-approximate algorithm with worst-case polylog update time. No deterministic algorithms with worst-case polylog update time were known for any *sublinear* $n^{1-\epsilon}$ approximation ratio.

Application to Weighted Matching. Our dynamic matching algorithms have applications to dynamic maximum *weight* matching (MWM), by standard reductions (see [3, 68]). Since our matching algorithms work against adaptive adversaries, we can apply these reductions to our algorithms as a black box, and need not worry about the inner workings of these reductions. As an added bonus, the obtained MWM algorithms work against adaptive adversaries, since their constituent subroutines do. So, plugging any of our algorithms into the above reductions yields MWM algorithms against adaptive adversaries whose approximation ratio is roughly twice that of our dynamic matching algorithms, with a logarithmic slowdown. Many of the bounds obtained this way were not even known against *oblivious* adversaries.

1.2 Techniques

In this section we outline our sparsification scheme and framework for dynamic matching against adaptive adversaries. Specifically, we show how to use edge colorings—partitions of the edges into (few) matchings—to quickly round fractional matchings dynamically against adaptive adversaries.[2] Before detailing these, we explain why the work of Gupta and Peng [41] motivates the study of dynamic matching sparsification.

In [41], Gupta and Peng present a $(1+\epsilon)$-approximate $O(\sqrt{m}/\epsilon^2)$ time algorithm, using a sparsifier and what they call the "stability" of the matching problem, which lends itself to lazy re-computation, as follows. Suppose we compute a matching M of size at least C times $\mu(G)$, the maximum matching size in G. Then, regardless of the updates in the following period of $\epsilon \cdot \mu(G)$ steps, the edges of M not deleted during the period remain a $C+O(\epsilon)$ matching in the dynamic graph, since both the size of M and $\mu(G)$ can at most change by $\epsilon \cdot \mu(G)$ during such a period. So, for example, using a static $O(m/\epsilon)$-time $(1+\epsilon)$-approximate matching algorithm [56] every $\epsilon \cdot \mu(G)$ updates yields a $(1+O(\epsilon))$-approximate dynamic matching algorithm with amortized update time $O_\epsilon(m/\mu(G))$. To obtain better update times from this observation, Gupta and Peng apply this idea to a sparsifier of size $S = O(\min\{m, \mu(G)^2\})$ which contains a maximum matching of G and which they show how to maintain in $O(\sqrt{m})$ update time, using the algorithm of [60]. From this they obtain a $(1+O(\epsilon))$-approximate matching algorithm with update time $O(\sqrt{m})+(S/\epsilon)/(\epsilon \cdot \mu(G)) = O(\sqrt{m}/\epsilon^2)$. We note that this lazy re-computation approach would even allow for *polylogarithmic-time* dynamic matching algorithms with approximation ratio $C+O(\epsilon)$, provided we could compute C-approximate matching sparsifiers of (optimal) size $S = \tilde{O}_\epsilon(\mu(G))$,[3] in time $\tilde{O}_\epsilon(\mu(G))$.

In this work we show how to use edge colorings to sample such size-optimal matching sparsifiers in optimal time. For simplicity, we describe our approach in terms of the subroutines needed to prove Theorem 1.1, deferring discussions of extensions to future sections.

Suppose we run the dynamic fractional matching algorithm of [22], maintaining a constant-approximate fractional matching $\vec{x}$ in deterministic worst-case polylog time. Also, for some $\epsilon > 0$, we dynamically partition G's edges into $O(\log n)$ subgraphs G_i, for $i = 1, 2, \ldots, O(\log_{1+\epsilon}(n))$, where G_i is the subgraph induced by edges of x-value $x_e \in ((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}]$. By the fractional matching constraint ($\sum_{e \ni v} x_e \leq 1 \ \forall v \in V$) and since $x_e \geq (1+\epsilon)^{-i}$ for all edges $e \in E(G_i)$, the maximum degree of any G_i is at most $\Delta(G_i) \leq (1+\epsilon)^i$. We can therefore edge-color each G_i with $2(1+\epsilon)^i (\geq 2\Delta(G_i))$ colors in deterministic worst-case $O(\log n)$ time per update in G_i, using [18]; i.e., logarithmic time per each of the $\operatorname{poly}\log n$ many changes which algorithm $\mathcal{A}$ makes to $\vec{x}$ per update. Thus, edge coloring steps take worst-case $\operatorname{poly}\log n$ time per update. A simple averaging argument

[2] An orthogonal approach was recently taken by Cohen et al. [27], who used matchings to round fractional edge colorings, in an *online* setting.

[3] We note that any sparsifier containing a constant-approximate matching must have size $\Omega(\mu(G))$.

shows that the largest color in these different G_i is an $O(\log n)$-approximate matching, which can be maintained efficiently. Extending this idea further yields Theorem 1.4 (see Appendix A). So, picking a singe color yields a fairly good approximation/time tradeoff. As we show, randomly combining a few colors yields space- and time-optimal constant-approximate matching sparsifiers.

To introduce our random sparsification scheme, we start by considering the sample of a single color M among the $2(1+\epsilon)^i$ colors of the coloring of subgraph G_i. For each edge $e \in G_i$, since $x_e \in ((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}]$, when sampling a random color M among these $2(1+\epsilon)^i$ colors, we sample the unique color containing e with probability proportional to x_e. Specifically, we have

$$\Pr[e \in M] = \frac{1}{2(1+\epsilon)^i} \in \left[\frac{x_e}{2(1+\epsilon)}, \frac{x_e}{2}\right).$$

Our approach will be to sample $\min\left\{2(1+\epsilon)^i, \frac{2\log n}{\epsilon^2}\right\}$ colors without replacement in G_i. By linearity, this yields a subgraph H of G which contains each edge e with probability roughly

$$p_e \triangleq \min\left\{1,\, x_e \cdot \frac{\log n}{\epsilon^2}\right\}. \tag{1}$$

As shown by Arar et al. [3], sampling a subgraph H with each edge $e \in E[G]$ belonging to H *independently* with probability p_e as above, with $\vec{x}$ taken to be the $(2+\epsilon)$-approximate fractional matching output by [22], yields a $(2+\epsilon)$-approximate matching sparsifier.[4] We prove that sampling H in this *dependent* manner yields as good a matching sparsifier as does independent sampling.

To bound the approximation ratio of our dependent-sampling-based sparsifiers, we appeal to the theory of *negative association* (see Section 2.1). In particular, we rely on sampling without replacement being a negatively-associated joint distribution. This implies sharp concentration of (weighted) degrees of vertices in H, which forms the core of our analysis of the approximation ratio of this sparsification scheme. In particular, we show that our matching sparsification yields sparsifiers with approximation ratio essentially equaling that of any "input" fractional matching in bipartite graphs, as well as a $(2+\epsilon)$-approximate sparsifiers in general graphs, using the fractional matchings of [22, 23].

Finally, to derive fast dynamic algorithms from this sparsification scheme, we note that our matching sparsifier H is the union of only $\operatorname{poly}\log n$ many matchings, and thus has size $\tilde{O}(\mu(G))$. Moreover, sampling this sparsifier requires only $\operatorname{poly}\log n$ random choices, followed by writing H. Therefore, sampling H can be done in $\tilde{O}(\mu(G))$ time (given the edge colorings, which we maintain dynamically). The space- and time-optimality of our sparsification scheme implies that we can maintain a matching with approximation ratio essentially equal to that of the obtained sparsifier, in worst-cast $\operatorname{poly}\log n$ update time. In particular, we can re-sample such a sparsifier, and compute a $(1+\epsilon)$-approximate matching in it, in $\tilde{O}_\epsilon(\mu(G))$ time, after every period of $\epsilon\cdot\mu(G)$ steps. This results in an $\tilde{O}_\epsilon(\mu(G))/(\epsilon\cdot\mu(G)) = \tilde{O}_\epsilon(1)$ amortized time per update (which can be easily de-amortized). Crucially for our use, during such periods, $\mu(G)$ and $\mu(H)$ do not change by much, as argued before. In particular, during such short periods of few updates, an adaptive adversary—even one which sees **the entire state** of the algorithm after each update—cannot increase the approximation ratio by more than a $1+O(\epsilon)$ factor compared to the approximation quality of the sparsifier. The above discussion yields our result of Theorem 1.1—a $(2+\epsilon)$-approximate dynamic matching algorithm with worst-case polylogarithmic update time against adaptive adversaries. Generalizing this further, we design a framework for dynamically rounding fractional matchings against adaptive adversaries, underlying our results of Theorems 1.1, 1.2 and 1.3.

[4] We note that a simple argument implying H contains a $(2+\epsilon)$-*fractional* matching with respect to G only implies a $(3+\epsilon)$-approximation. This is due to the $\frac{3}{2}$ integrality gap of the fractional matching polytope, and in particular the fact that fractional matchings may be $\frac{3}{2}$ times larger than the largest matching in a graph (see, e.g., a triangle).

1.3 Related Work

Here we discuss the dynamic matching literature in more depth, contrasting it with the results obtained from our dynamic rounding framework.

The first non-trivial result for dynamic matching was given by Ivkovic and Lloyd [49], who showed how to maintain a maximal matching (which is a 2-approximate matching) in $O((m+n)^{1/\sqrt{2}})$ amortized update time. Much later, Sankowski [64] presented an $O(n^{1.495})$ update time algorithm for maintaining the *value* (size) of a maximum matching. Both algorithms above are only sublinear in $n+m$ for sufficiently sparse graphs, and are both far from the gold standard for data structures – polylog update time. On the other hand, several works show that for exact maximum matching, polylog update time is *impossible*, assuming several widely-held conjectures, including the strong exponential time hypothesis and the 3-SUM conjecture [1, 2, 29, 45, 53]. A natural question is then whether polylog update time suffices to maintain an *approximate* maximum matching.

Polylog-time algorithms. In seminal work, Onak and Rubinfeld [61] presented the first polylog-time algorithm for constant-approximate matching. Baswana et al. [7] improved this with an $O(\log n)$-time maximal (and thus 2-approximate) matching algorithm. Some years later a deterministic $(2+\epsilon)$-approximate matching algorithm with amortized $\text{poly}(\log n, 1/\epsilon)$ update time algorithm was presented by Bhattacharya et al. [21]. Solomon [66] then gave a randomized maximal matching algorithm with *constant* amortized time. Recently, several randomized $(2+\epsilon)$-approximate/maximal matching algorithms with worst-case polylog time were developed, with either the approximation ratio or the update time holding w.h.p. [3, 14, 24]. All prior randomized algorithms assume an oblivious adversary. Obtaining the same guarantees against an adaptive adversary remained open. Another line of work studied the dynamic maintenance of large *fractional* matchings in polylog update time, thus maintaining a good approximation of the maximum matching's *value* (though not a large matching) [17, 19, 22, 23, 40]. The best current bounds for this problem are deterministic $(2+\epsilon)$-approximate fractional matching algorithms with $\text{poly}(\log n, 1/\epsilon)$ worst-case and $\text{poly}(1/\epsilon)$ amortized update times [22, 23]. Our randomized algorithms of Theorems 1.1 and 1.2 match these bounds, for *integral* matching, against adaptive adversaries.

Polytime algorithms. Many sub-linear time dynamic matching algorithms have been developed over the years. These include $(1+\epsilon)$-approximate algorithms with $O(\sqrt{m}/\epsilon^2)$ worst-case update time algorithms [41, 63] (the former building on a maximal $O(\sqrt{m})$-time algorithm of [60]), and $(2+\epsilon)$-approximate algorithms with worst-case $O(\min\{\sqrt[3]{m}, \sqrt{n}\}/\text{poly}(\epsilon))$ update time [20]. The fastest polytime algorithm to date with worst-case update time is a $\left(\frac{3}{2}+\epsilon\right)$-approximate $O(\sqrt[4]{m}/\text{poly}(\epsilon))$-time algorithm in bipartite graphs [15] (similar amortized bounds are known in general graphs [16]). In contrast, we obtain algorithms with *arbitrarily-small* polynomial update time, yielding a constant approximation deterministically (Theorem 1.4), and even better-than-2 approximation in bipartite graphs against adaptive adversaries (Theorem 1.3). This latter bound matches bounds previously only known for dynamic *fractional* matching [21], and nearly matches a recent $O(\Delta^{\epsilon})$-time algorithm for general graphs, which assumes an oblivious adversary [10].

Matching sparsifiers. Sparsification is a commonly-used algorithmic technique. In the area of dynamic graph algorithms it goes back more than twenty years [32]. For the matching problem in various computational models, multiple sparsifiers were developed [5, 6, 15, 16, 20, 39, 41, 54, 63, 67]. Unfortunately for dynamic settings, all these sparsifiers are either polynomially larger than $\mu(G)$, the maximum matching size in G, or were not known to be maintainable in $n^{o(1)}$ time against adaptive adversaries. In this paper we show how to efficiently maintain a generalization of matching kernels of [20] of size $\tilde{O}(\mu(G))$, efficiently, against adaptive adversaries.

2 Preliminaries

A *matching* in a graph $G = (V, E)$ is a subset of vertex-disjoint edges $M \subseteq E$. The cardinality of a maximum matching in G is denoted by $\mu(G)$. An *edge coloring* of a graph $G = (V, E)$ is a partition of the edge set of G into matchings. That is, it is an assignment of colors to the edges of G such that each vertex is incident on at most one edge of each color. A *fractional matching* is a non-negative vector $\vec{x} \in \mathbb{R}^m_{\geq 0}$ satisfying the *fractional matching constraint*, $\sum_{e \ni v} x_e \leq 1 \ \ \forall v \in V$.

In a fully-dynamic setting the input is a dynamic graph G, initially empty, on a set of n fixed vertices V, subject to edge *updates* (additions and removals). An α-approximate matching algorithm $\mathcal{A}$ maintains a matching M of size at least $|M| \geq \frac{1}{\alpha} \cdot \mu(G)$. If the algorithm $\mathcal{A}$ is deterministic, $|M| \geq \frac{1}{\alpha} \cdot \mu(G)$ holds for any sequence of updates. If $\mathcal{A}$ is randomized, this bound on M's size can hold in expectation or w.h.p., though here one must be more careful about the sequence of updates. The strongest guarantees for randomized algorithms are those which hold for sequences generated by an adaptive adversary.

Dynamic Edge Coloring. An important ingredient in our matching algorithms are algorithms for the "complementary" problem of edge coloring, i.e., the problem of covering the graph's edge-set with matchings (colors). Vizing's theorem [70] asserts that $\Delta + 1$ colors suffice to edge color any graph of maximum degree Δ. (Clearly, at least Δ colors are needed.) In dynamic graphs, a deterministic $(2\Delta - 1)$-edge-coloring algorithm with $O(\log n)$ worst-case update time is known [18]. Also, a 3Δ-edge-coloring can be trivially maintained in $O(1)$ expected update time against an adaptive adversary, by picking random colors for each new edge (u, v) until an available color is picked.[5]

2.1 Negative Association

For our randomized sparsification algorithms, we sample colors without replacement. To bound (weighted) sums of edges sampled this way, we rely on the following notion of negative dependence, introduced by Joag-Dev and Proschan [50] and Khursheed and Lai Saxena [52].

Definition 2.1 (Negative Association)**.** *We say a joint distribution $X_1, X_2, \ldots, X_n$ is* negatively associated (NA)*, or alternatively that the random variables $X_1, X_2, \ldots, X_n$ are NA, if for any non-decreasing functions g and h and disjoint subsets $I, J \subseteq [n]$ we have*

$$\mathrm{Cov}(g(X_i : i \in I), h(X_j : j \in J)) \leq 0. \tag{2}$$

One trivial example of NA variables are independent variables, for which Inequality (2) is satisfied with *equality* for *any* functions f and g. A more interesting example of NA distributions are *permutation distributions*, namely a joint distribution where $(X_1, X_2, \ldots, X_n)$ takes on all permutations of some vector $\vec{x} \in \mathbb{R}^n$ with equal probability [50]. More elaborate NA distributions can be constructed from simple NA distributions as above by several NA-preserving operations, including scaling of variables by positive constants, and taking independent union [31, 50, 52]. (That is, if the joint distributions $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_m$ are both NA and are independent of each other, then the joint distribution $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m$ is also NA.)

An immediate consequence of the definition of NA is that NA variables are negatively correlated. A stronger consequence is that NA variables $X_1, \ldots, X_n$ satisfy $\mathbb{E}[\exp(\lambda \sum_i X_i)] \leq \prod_i \mathbb{E}[\exp(\lambda X_i)]$ (see [31]), implying applicability of Chernoff-Hoeffding bounds to sums of NA variables.

[5] Dynamic algorithms using fewer colors are known, though they are slower [30]. Moreover, as the number of colors $\gamma\Delta$ used only affects our update times by a factor of γ (and does not affect our approximation ratio), the above simple 2Δ- and 3Δ-edge-coloring algorithms will suffice for our needs.

Lemma 2.2 (Chernoff bounds for NA variables [31]). *Let X be the sum of NA random variables $X_1, X_2, \ldots, X_m$ with $X_i \in [0,1]$ for each $i \in [m]$. Then for all $\delta \in (0,1)$, and $\kappa \geq \mathbb{E}[X]$,*

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

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

Another tail bound obtained from $\mathbb{E}[\exp(\lambda\sum_i X_i)] \leq \prod_i \mathbb{E}[\exp(\lambda X_i)]$ for NA variables is Bernstein's Inequality, which yields stronger bounds for sums of NA variables with bounded variance. See, e.g., [28].

Lemma 2.3 (Bernstein's Inequality for NA Variables). *Let X be the sum of NA random variables $X_1, X_2, \ldots, X_k$ with $X_i \in [0, M]$ for each $i \in [k]$. Then, if $\sigma^2 = \sum_{i=1}^{k} \mathrm{Var}(X_i)$, then for all $a > 0$,*

$$\Pr[X > \mathbb{E}[X] + a] \leq \exp\left(\frac{-a^2}{2(\sigma^2 + aM/3)}\right).$$

3 Edge-Color and Sparsify

In this section we present our edge-coloring-based matching sparsification scheme, together with some useful properties of this sparsifier necessary to bound its quality. We then show how to implement this scheme in a dynamic setting against an adaptive adversary with $(1-\epsilon)$ loss in the approximation ratio. We start by defining our sparsification scheme in a static setting.

3.1 The sparsification scheme

Our edge-coloring-based sparsification scheme receives a fractional matching $\vec{x}$ as an input, as well as parameters $\epsilon \in (0,1), d \geq 1$ and integer $\gamma \geq 1$. It assumes access to a $\gamma\Delta$-edge-coloring algorithm for graphs of maximum degree Δ. For some logarithmic number of indices $i = 1, 2, \ldots, 3\log_{1+\epsilon}(n/\epsilon) = O(\log(n/\epsilon)/\epsilon)$, our algorithm considers subgraphs G_i induced by edges with x-value in the range $((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}]$, and $\gamma\Delta(G_i) \leq \gamma(1+\epsilon)^i$-edge-colors each such subgraph G_i. It then samples at most γd colors without replacement in each such G_i. The output matching sparsifier H is the union of all these sampled colors. The algorithm's pseudocode is given in Algorithm 1, below.

Algorithm 1 Edge-Color and Sparsify

1: **for all** $i \in \{1, 2, \ldots, \lceil 2\log_{1+\epsilon}(n/\epsilon)\rceil\}$ **do**
2: let $E_i \triangleq \{e \mid x_e \in ((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}]\}$.
3: compute a $\gamma\lceil(1+\epsilon)^i\rceil$-edge-coloring χ_i of $G_i \triangleq G[E_i]$. ▷ Note: $\Delta(G_i) < (1+\epsilon)^i$
4: Let S_i be a sample of $\min\{\gamma\lceil d(1+\epsilon)\rceil, \gamma\lceil(1+\epsilon)^i\rceil\}$ colors without replacement in χ_i.
5: **return** $H \triangleq (V, \bigcup_i \bigcup_{M\in S_i} M)$.

We note the obvious sparsity of this matching sparsifier H, implied by its being the union of $O\left(\frac{\log(n/\epsilon)}{\epsilon}\cdot\gamma\cdot d\right)$ matchings in G, all of which have size at most $\mu(G)$, by definition.

Observation 3.1. *The number of edges in H output by Algorithm 1 is at most*

$$|E(H)| = O\left(\frac{\log(n/\epsilon)}{\epsilon}\cdot\gamma\cdot d\cdot\mu(G)\right).$$

3.2 Basic properties of Algorithm 1

As we shall see in Appendix B.2, the subgraph H output by Algorithm 1 is a good matching sparsifier, in the sense that $\mu(H) \geq \frac{1}{c} \cdot \mu(G)$ for some small c, provided the fractional matching is a good approximate fractional matching. We will refer to this c as the *approximation ratio* of H. Our analysis of the approximation of H will rely crucially on the following lemmas of this section.

Throughout our analysis we will focus on the run of Algorithm 1 on some fractional matching $\vec{x}$ with some parameters d, γ and ϵ, and denote by H the output of this algorithm. For each edge $e \in E$, we let $X_e \triangleq \mathbb{1}[e \in H]$ be an indicator random variable for the event that e belongs to this random subgraph H. We first prove that the probability of this event occurring nearly matches p_e given by Equation (1) with $\frac{\log n}{\epsilon^2}$ replaced by d. Indeed, the choice of numbers of colors sampled in each G_i was precisely made with this goal in mind. The proof of the corresponding lemma below, which follows by simple calculation, is deferred to Appendix B.

Lemma 3.2. *If $d \geq \frac{1}{\epsilon}$ and $\gamma \geq 1$, then for every edge $e \in E$, we have*

$$\min\{1, x_e \cdot d\}/(1+\epsilon)^2 \leq \Pr[e \in H] \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon). \tag{3}$$

Moreover, if $x_e > \frac{1}{d}$, then $\Pr[e \in H] = 1$.

Crucially for our analysis, where we concern ourselves with bounding weighted vertex degrees, the variables X_e for edges of any given vertex are NA, as we establish in the following key lemma.

Lemma 3.3 (Negative Association of edges)**.** *For any vertex v, the variables $\{X_e \mid e \ni v\}$ are NA.*

To prove the above lemma, we will need to consider the following NA distributions.

Proposition 3.4. *Let $e_1, e_2, \ldots, e_n$ be some ordering of elements in a universe of size n. For each $i \in [k]$, let X_i be an indicator for elements e_i being sample in a sample of $k \leq n$ random elements without replacement from $e_1, e_2, \ldots, e_n$. Then $X_1, X_2, \ldots, X_n$ are NA.*

Proof. Randomly sampling k elements from $e_1, e_2, \ldots, e_n$ without replacement is equivalent to letting the vector $(X_1, X_2, \ldots, X_n)$ take on all permutations of a $0-1$ vector with k ones, with equal probability. The proposition therefore follows from NA of permutation distributions [50]. □

Proof of Lemma 3.3. For all G_i, add a dummy edge to v for each color not used by (non-dummy) edges of v in G_i. Randomly sampling $k = \min\{\lceil \gamma d \rceil, \lceil \gamma \cdot (1+\epsilon)^i \rceil\}$ colors in the coloring without replacement induces a random sample without replacement of the (dummy and non-dummy) edges of v in G_i. By Proposition 3.4, the variables $\{X_e \mid e \ni v, \text{non-dummy } e \in G_i\}$ are NA (since subsets of NA variables are themselves NA). The sampling of colors in the different G_i is independent, so by closure property of NA under independent union the variables $\{X_e \mid e \ni v\}$ are indeed NA. □

The negative correlation implied by negative association of the variables $\{X_e \mid e \ni v\}$ also implies that conditioning on a given edge $e' \ni v$ being sampled into H only decreases the probability of any other edge $e \ni v$ being sampled into H. So, from lemma 3.2 and 3.3 we obtain the following.

Corollary 3.5. *For any vertex v and edges $e, e' \ni v$, we have*

$$\Pr[e \in H \mid e' \in H] \leq \Pr[e \in H] \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon).$$

Finally, we will need to argue that the negative association of edges incident on any vertex v holds even after conditioning on some edge $e' \ni v$ appearing in H.

Lemma 3.6. *For any vertex v and edge $e' \ni v$, the variables $\{Y_e \triangleq [X_e \mid X_{e'} = 1] \mid e \ni v\}$ are NA.*

This lemma's proof is essentially the same as that of Lemma 3.3, while noting that if e' is in H, then the unique matching containing e' in the edge coloring of $G_i \ni e'$ must be sampled. This implies that the remaining colors sampled from the coloring of G_i also constitute a random sample without replacement, albeit a smaller sample from a smaller population (both smaller by one than their unconditional counterparts).

3.3 The Dynamic Rounding Framework

Here we present our framework for dynamically rounding fractional matchings.

Key to this framework is Observation 3.1, which implies that we can sample H using Algorithm 1 and compute a $(1+\epsilon)$-approximate matching in H in $O_\epsilon(\mu(G))$ time. This allows us to (nearly) attain the approximation ratio of this subgraph H dynamically, against an adaptive adversary.

Theorem 3.7. *Let $\gamma \geq 1$, $d \geq 1$ and $\epsilon > 0$. Let $\mathcal{A}_f$ be a constant-approximate dynamic fractional matching algorithm with update time $T_f(n,m)$. Let $\alpha = \alpha(d, \epsilon, \gamma, \mathcal{A}_f)$ be the approximation ratio of the subgraph H output by Algorithm 1 with parameters d, ϵ and γ when run on the fractional matching of $\mathcal{A}_f$. Let $\mathcal{A}_c$ be a dynamic $\gamma\Delta$-edge-coloring algorithm with update time $T_c(n,m)$. If the guarantees of $\mathcal{A}_f$ and $\mathcal{A}_c$ hold against an adaptive adversary, then there exists an $\alpha(1+O(\epsilon))$-approximate dynamic matching algorithm $\mathcal{A}$ against an* **adaptive** *adversary, with update time*

$$O\left(T_f(n,m) \cdot T_c(n,m) + \log(n/\epsilon) \cdot \gamma \cdot d/\epsilon^3\right).$$

Moreover, if $\mathcal{A}_f$ and $\mathcal{A}_c$ have worst-case update times, so does $\mathcal{A}$, and if the approximation ratio given by H is w.h.p., then so is the approximation ratio of $\mathcal{A}$.

Our approach for Theorem 3.7 is to maintain a fractional matching $\vec{x}$ using $\mathcal{A}_f$, and $\gamma \cdot (1+\epsilon)^i (\geq \gamma \cdot \Delta(G_i))$ edge colorings of the subgraphs $G_i \triangleq G[\{e \mid x_e \in ((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}]\}]$ using $\mathcal{A}_c$. This requires $T_c(n,m)$ time for each of the $T_f(n,m)$ changes to $\vec{x}$ by $\mathcal{A}_f$ in each update, or a total of $O(T_f(n,m) \cdot T_c(n,m))$ time per update. In addition, we use Algorithm 1 to sample a good matching sparsifier and $(1+\epsilon)$-approximate matching in H using a static $O(|E(H)|/\epsilon)$-time algorithm. By Observation 3.1, both these steps take $O\big(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\big)$ time. If $\mu(G) = O(1/\epsilon)$, (which we can verify using our constant-approximate fractional matching) we simply spend this $O\big(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\big) = O\big(\log(n/\epsilon) \cdot \gamma \cdot d/\epsilon^3\big)$ time for this computation each update and just use this $(1+\epsilon)$-approximate matching in H as our matching. Otherwise, we "spread" the above computation over periods of $O(\lceil \epsilon \cdot \mu(G) \rceil)$ updates, incurring a further $O\left(\log(n/\epsilon) \cdot \gamma \cdot d/\epsilon^3\right)$ time per update. As each update can only affect the approximation ratio by a $(1+O(\epsilon))$ multiplicative factor, the matching computed at the beginning of the previous period yields an $\alpha(1+O(\epsilon))$ approximation. The formal description and analysis of this framework can be found in Appendix B.1.

Remark. We note that a $\log(n/\epsilon)/\epsilon$ factor in the above running time is due to the size of H sampled at the beginning of a period being $|E(H)| = O_\epsilon(d \cdot \gamma \cdot \log(n/\epsilon) \cdot \mu(G)/\epsilon)$ and the number of subgraphs G_i being $O(\log(n/\epsilon)/\epsilon)$. For some of the fractional matchings we apply our framework to, the sparsifier H has a smaller size of $|E(H)| = O(\gamma \cdot d \cdot \mu(G))$, and we only need to sample colors from $O(\gamma \cdot d \cdot \mu(G))$ edge colorings to sample this subgraph. For these fractional matchings the update time of the above algorithm therefore becomes $T_f(n,m) \cdot T_c(n,m) + O(\gamma \cdot d/\epsilon^2)$.

Theorem 3.7 allows us to obtain essentially the same approximation ratio as that of H computed by Algorithm 1 in a static setting, but dynamically, and against an adaptive adversary. The crux of our analysis will therefore be to bound the approximation ratio of H, which we now turn to.

4 Overview of analysis of Algorithm 1

In order to analyze the approximation ratio of the subgraph H output by Algorithm 1 (i.e., the ratio $\mu(G)/\mu(H)$), we take two approaches, yielding different (incomparable) guarantees. One natural approach, which we take in Section 4.2, shows that Algorithm 1 run on an α-approximate fractional matching outputs a subgraph H which itself contains a fractional matching which is α-approximate in G. For bipartite graphs this implies H contains an α-approximate *integral* matching. For general graphs, however, this only implies the existence of a $\frac{3\alpha}{2}$-approximate integral matching in H, due to the integrality gap of the fractional matching polytope in general graphs. Our second approach does not suffer this deterioration in the approximation ratio compared to the fractional matching, for a particular (studied) class of fractional matchings. We start with this second approach.

4.1 Sparsifiers from Approximately-Maximal Fractional Matchings

Here we show how to avoid the multiplicative factor of $\frac{3}{2}$ implied by the integrality gap when sparsifying using (particularly well-structured) fractional matchings $\vec{x}$. To prove this improved approximation ratio we generalize the notion of *kernels*, introduced in [20] and later used by [3, 21]. In particular, we extend this definition to allow for *distributions* over subgraphs, as follows.

Definition 4.1. *(Kernels) A* (c, d, ϵ)-kernel *of a graph G is a (random) subgraph $\mathcal{H}$ of G satisfying:*

1. *For each vertex $v \in V$, the degree of v in $\mathcal{H}$ is at most $d_{\mathcal{H}}(v) \leq d$ always.*
2. *For each edge $e \in E$ with $\Pr[e \notin \mathcal{H}] > \epsilon$, it holds that $\mathbb{E}[\max_{v \in e} d_{\mathcal{H}}(v) \mid e \notin \mathcal{H}] \geq d/c$.*

If $\mathcal{H}$ is a deterministic distribution, we say $\mathcal{H}$ is a deterministic kernel.

Such a graph is clearly sparse, containing at most $O(nd)$ edges. (Crucially for our needs, the kernels we compute even have size $|E(H)| = \tilde{O}(\mu(G))$.) As shown in [3], deterministic $(c, d, 0)$-kernels have approximation ratio $2c(1+1/d)$. (Coincidentally, this proof also relies on edge colorings.) Generalizing this proof, we show that a randomized (c, d, ϵ)-kernel has approximation ratio $2c(1 + 1/d)$ in expectation. The key difference is that now rather than comparing $\mu(G)$ to the value of some fractional matching in $H \sim \mathcal{H}$, we compare $\mu(G)$ to some fractional matching's *expected* value. As this generalization is rather minor, we defer its proof to Appendix B.2.1.

Lemma 4.2. *Let $\mathcal{H}$ be a (c, d, ϵ)-kernel of G for some $c \geq \frac{1}{1-\epsilon}$. Then $\mathbb{E}[\mu(\mathcal{H})] \geq \frac{1}{2c(1+1/d)} \cdot \mu(G)$.*

As we show, the subgraph H output by Algorithm 1, when run on well-structured fractional matchings, contains such a kernel. Specifically, we show that H contains a kernel, provided the input fractional matching is *approximately maximal*, as in the following definition of Arar et al. [3].

Definition 4.3 (Approximately-Maximal Fractional Matching [3])**.** *A fractional matching $\vec{x}$ is* (c, d)-approximately-maximal *if every edge $e \in E$ either has fractional value $x_e > 1/d$ or it has one endpoint v with $\sum_{e \ni v} x_e \geq 1/c$ with all edges e' incident on this v having value $x_{e'} \leq 1/d$.*

Some syntactic similarity between definitions 4.1 and 4.3 should be apparent. For a start, both generalize maximal (integral or fractional) matchings, which is just the special case of $c = d = 1$. Both require an upper bound on the (weighted) degree of on any vertex, and stipulate that some edges have an endpoint with high (weighted) degree. Indeed, this similarity does not stop there, and as shown in [3], sampling each edge e of a (c, d)-approximately-maximal fractional matching independently with probability p_e as in (1) yields a deterministic $(c(1 + O(\epsilon), d(1 + O(\epsilon), 0)$-kernel w.h.p. As we show, sampling each edge e with probability roughly p_e, such that the edges are NA, as in Algorithm 1, yields the same kind of kernel, w.h.p.

Lemma 4.4. *Let $c \geq 1$, $\epsilon > 0$ and $d \geq \frac{9c(1+\epsilon)^2 \cdot \log n}{\epsilon^2}$. If $\vec{x}$ is a (c,d)-approximately-maximal fractional matching, then the subgraph H output by Algorithm 1 when run on $\vec{x}$ with ϵ and d as above is a deterministic $(c(1+O(\epsilon)), d(1+O(\epsilon)), 0)$-kernel, w.h.p.*

The proof broadly relies on Chernoff bounds for degrees of vertices, which are sums of NA variables, by Lemma 3.3, as follows. The first kernel constraint ($\Delta(H) \leq d(1+O(\epsilon))$) is rather direct. For the second constraint, we rely on the approximate-maximality of $\vec{x}$, noting that any edge not sampled in H must have $x_e \leq 1/d$ by Lemma 3.2, and so one of this edge's endpoints, $v \in e$, must satisfy $\sum_{e' \ni v} x_{e'} \geq 1/c$ and $x_{e'} \leq 1/d$ for all $e' \ni v$. This implies that $\min\{1, x_{e'} \cdot d\} = x_{e'} \cdot d$. Therefore, by Lemma 3.2, the expected degree of v is at least $\sum_{e' \ni v} x_{e'} \cdot d/(1+\epsilon)^2$, and this is sharply concentrated, by Lemma 2.2. (See Appendix B.2.2 for a full proof.)

In light of Lemma 4.4, we now turn to discussing further implications of Theorem 3.7.

As shown in [3], the output fractional matching of [22] is $(1+\epsilon, d)$-approximately-fractional, for some $d = \text{poly}(\log n, 1/\epsilon)$ large enough to satisfy the conditions of Lemma 4.4. Therefore, plugging in this $\text{poly}(\log n, 1/\epsilon)$ worst-case update time deterministic algorithm into Theorem 3.7 in conjunction with the deterministic $O(\log n)$-time 2Δ-edge-coloring algorithm of [20], we only obtain a Monte Carlo algorithm with guarantees similar to that of Theorem 1.1. However, since we can efficiently verify the high-probability events implying that H is a kernel, we can re-sample H if ever it is not a kernel. From this we recover the guarantees of Theorem 1.1 — a Las Vegas randomized dynamic $(2+\epsilon)$-approximate matching algorithm with $\text{poly}\log n$ update time w.h.p, which works against adaptive adversaries.

To obtain the constant-time algorithm of Theorem 1.2, we rely on the constant-time fractional matching algorithm of Bhattacharya and Kulkarni [23], which we show outputs a $(1+\epsilon, d)$-approximately-maximal matching for any $d > 1+\epsilon$ (see Appendix B.2.3). We show that these fractional matchings also only define $O(\mu(G))$ subgraphs G_i, as they only assign one of $O(\mu(G))$ x-values to all edges. This implies in particular that Algorithm 1 can sample H from such $\vec{x}$ using only $O(\gamma \cdot d \cdot \mu(G))$ random choices, yielding a subgraph of size $O(\gamma \cdot d \cdot \mu(G))$. Using a simple constant-expected-time 3Δ-edge-coloring algorithm, this improves the update time to $\text{poly}(1/\epsilon) + O(\gamma \cdot d/\epsilon)$. Taking $d = O(\log n)$, this yields the first $(2+\epsilon)$-approximate algorithms with amortized logarithmic update time against adaptive adaptive adversaries. Pleasingly, we can improve this bound further, and obtain a constant-time such algorithm. To this end, we show that sampling H using a (c,d)-approximately-maximal fractional matching with $d \geq \text{poly}(1/\epsilon)$ and removing high-degree vertices yields a (randomized) $(c(1+O(\epsilon)), d(1+O(\epsilon)), \epsilon)$-kernel. (See Appendix B.2.3.) From the above we thus obtain the first constant-time $(2+\epsilon)$-approximate algorithm, as stated in Theorem 1.2.

4.2 Fractional Matching Sparsifiers

The approach we apply in this section to analyze Algorithm 1 consists of showing that the subgraph H obtained by running Algorithm 1 on any fractional matching $\vec{x}$ with appropriate choices of d and ϵ supports a fractional matching $\vec{y}$ with $\mathbb{E}[\sum_e y_e] \geq \sum_e x_e(1-O(\epsilon))$. That is, we prove H is a near-lossless *fractional* matching sparsifier.

Lemma 4.5. *(Algorithm 1 Yields Fractional Matching Sparsifiers) Let $\epsilon \in (0, 1/2)$ and $d \geq \frac{3\log(2/\epsilon)}{8\epsilon}$. If H is a subgraph of G output by Algorithm 1 when run on a fractional matching $\vec{x}$ with parameters ϵ and d as above, then H supports a fractional matching $\vec{y}$ of expected value at least*

$$\mathbb{E}\left[\sum_e y_e\right] \geq \sum_e x_e(1-4\epsilon).$$

The full proof of this lemma is deferred to Appendix B.2.4. We briefly sketch it here. We first define a fractional matching in H with each edge e assigned value $z_e = (1-\epsilon)x_e/\min\{1, x_e \cdot d\}$. By Lemma 3.2 this immediately implies that $\mathbb{E}[\sum_e z_e] \geq (1-O(\epsilon))\sum_e x_e$. Next, we define a fractional matching $\vec{y}$, letting $y_e = z_e$, or $y_e = 0$ if $x_e \leq 1/d$ and some endpoint $v \in e$ has it fractional matching constraint violated by $\vec{z}$, i.e., $\sum_{e' \ni v} z_{e'} > 1$. The only delicate point of the analysis is showing that for edge e, conditioning on e being sampled into H, with probability $1-\epsilon$ both endpoints $v \in e$ have their fractional matching constraint satisfied by $\vec{z}$. For this, we note that the fractional degree of v conditioned on $e \in H$, is the sum of variables $z_{e'} \cdot [X_{e'} \mid X_e]$, which by closure of NA under scaling by positive constants and Lemma 3.6 these variable are NA. Therefore, applying Bernstein's Inequality (Lemma 2.3) to these low-variance NA variables, we find that with probability $1-\epsilon$, if $e = (u, v)$ is sampled, then both endpoints of e have $\sum_{e' \ni v} z_{e'} \leq 1$, in which case $y_e = z_e$, and so we have $\mathbb{E}[y_e] \geq (1-\epsilon) \cdot \mathbb{E}[z_e] \geq (1-O(\epsilon)) \cdot x_e$, as claimed.

For bipartite graphs, whose fractional matching polytopes are integral, Lemma 4.5 implies that running Algorithm 1 on an α-approximate fractional matching $\vec{x}$ yields an $\alpha(1+O(\epsilon))$-approximate subgraph H. For example, plugging the better-than-two approximate fractional matching algorithm of Bhattacharya et al. [21] into our dynamic matching framework, we obtain the first $(2-\delta)$-approximate algorithms with arbitrarily-small polynomial update time against adaptive adversaries in bipartite graphs, as stated in Theorem 1.3.

5 Conclusions and Open Questions

This paper provides the first randomized dynamic matching algorithms which work against adaptive adversaries and outperform deterministic algorithms for this problem. It obtains a number of such algorithmic results, based on a new framework for rounding fractional matchings dynamically against an adaptive adversary. Our results suggest a few follow up questions, of which we state a few below.

Maximum Weight Matching (MWM). The current best approximation for dynamic MWM with polylog worst-case update time against adaptive adversaries is $(4+\epsilon)$, obtained by applying the reduction of [68] to our algorithm of Theorem 1.1. This is far from the more familiar ratios of 2 or $(2+\epsilon)$ known to be achievable efficiently for this problem in other models of computation, such as streaming [38, 62] and the CONGEST model of distributed computation [37, 55]. Attaining such bounds in a dynamic setting in polylog update time (even amortized and against an oblivious adversary) remains a tantalizing open problem.

Better Approximation. To date, no efficient (i.e., polylog update time) dynamic matching algorithm with approximation better than two is known. As pointed out by Assadi et al. [4], efficiently improving on this ratio of two for maximum matching has been a longstanding open problem in many models, and has recently been proven impossible to do in an online setting [35]. Is the dynamic setting "easier" than the online setting, or is an approximation ratio of 2 the best approximation achievable in polylog update time?

Acknowledgements. This work has benefited from discussions with many people. In particular, the author would like to thank Anupam Gupta, Bernhard Haeupler, Seffi Naor and Cliff Stein for helpful discussions. The author would also like to thank Naama Ben-David, Bernhard Haeupler and Seffi Naor for comments on an earlier draft of this paper which helped improve its presentation. This work was supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.

Appendix

A Warm Up: Deterministic Algorithms

Here we discuss our deterministic matching algorithms obtained by generalizing the discussion in Section 1.1. First, we note that the $(2\Delta-1)$-edge-coloring algorithm of [18] works for multigraphs.

Lemma A.1 ([18]). *For any dynamic* multigraph G *with maximum degree* Δ*, there exists a deterministic* $(2\Delta-1)$*-edge-coloring algorithm with worst-case update time* $O(\log \Delta)$*.*

Broadly, the algorithm of [18] relies on binary search, relying on the following simple observation. For $(2\Delta-1)$ colors, if we add an edge (u,v), then the total number of colors used by u and v for all their (at most $\Delta-1$) edges other than (u,v), even counting repetitions, is at most $2\Delta-2$. That is, fewer than the number of colors in the entire palette, $[2\Delta-1]$. Consequently, either the range $\{1,2,\ldots,\Delta\}$ or $\{\Delta+1,\Delta+2,\ldots,2\Delta-1\}$ has a smaller number of colors used by u and v (again, counting repetitions). This argument continues to hold recursively in this range in which u and v have used fewer colors than available. With the appropriate data structures, this observation is easily implemented to support $O(\log \Delta)$ worst-case update time for both edge insertions and deletions (see [18] for details). As the underlying binary-search argument above did not rely on simplicity of the graph, this algorithm also works for multigraphs.

We now show how to use this simple edge-coloring algorithm in conjunction with dynamic fractional matching algorithms to obtain a family of deterministic algorithms allowing to trade off approximation ratio for worst-case update time.

Theorem 1.4. *For any* $K>1$*, there exists a deterministic* $O(K)$*-approximate matching algorithm with worst-case* $\tilde{O}(n^{1/K})$ *update time.*

Proof. We maintain in the background a 2.5-approximate fractional matching $\vec{x}$ using a deterministic algorithm with worst-case polylogarithmic update time, such as that of [22] run with $\epsilon=0.5$. Letting $\mathcal{R}:=n^{1/K}$, we define $O(K)$ multigraphs whose union contains all edges in G. Specifically, for each $i=1,2,\ldots,2\log_{\mathcal{R}}(2n)$ we let G_i be a multigraph whose edges are the edges of G of x-value $x_e\in[\mathcal{R}^{-i},\mathcal{R}^{-i+1}]$, with each such edge e having $\lceil x_e/\mathcal{R}^{-i}\rceil$ parallel copies in G_i. So, for example, an edge with x-value of $\mathcal{R}^{-i}$ will have a single parallel copy in G_i, and an edge wit x-value of $\mathcal{R}^{-i+1}$ will have $\lceil\mathcal{R}\rceil\leq n^{1/K}+1$ parallel copies in G_i. By the fractional matching constraint $(\sum_{e\ni v}x_e\leq 1\ \ \forall v\in V)$, the maximum degree in each graph G_i is at most $\Delta(G_i)\leq\mathcal{R}^i$. Therefore, using the edge coloring algorithm of [18] we can maintain a $2\Delta(G_i)-1\leq 2\cdot\mathcal{R}^i$ edge coloring in each G_i deterministically in worst-case $O(\log n)$ time per edge update in G_i. Since for any edge e a change to x_e causes at most $\lceil\mathcal{R}\rceil$ parallel copies of e to be added to or removed from multigraphs G_i, we find that each x-value changes performed by the fractional matching algorithm require $O(\mathcal{R}\cdot\log n)$ worst-case time. As the fractional algorithm has polylogarithmic update time (and therefore at most that many x-value changes per update), the overall update time of these subroutines is therefore at most $\tilde{O}(\mathcal{R})=\tilde{O}(n^{1/K})$. Our algorithm simply maintains as its matching the largest color class in any of these multigraphs. It remains to bound the approximation ratio of this approach.

First, we note that all edges not in any G_i, i.e., of x-value at most $\mathcal{R}^{-\log_{\mathcal{R}}(2n)}=1/(4n^2)$, contribute at most $\sum_{e:x_e\leq\epsilon^2/n^2}x_e\leq 1/4$ to $\sum_e x_e$. So, as $\vec{x}$ is a 2.5-approximate fractional matching, we have that

$$\sum_{e\in\bigcup G_i}x_e\geq\frac{1}{2.5}\cdot\mu(G)-\frac{1}{4}\geq\frac{1}{O(1)}\cdot\mu(G),$$

where as before, $\mu(G) \geq 1$ is the maximum matching size in G. (Note that if $\mu(G) = 0$ any algorithm is trivially 1-approximate.) Therefore, as $\mathcal{R} = n^{1/K}$ at least one of these $2\log_{\mathcal{R}}(2n) = O(K)$ multigraphs G_i must have total x-value at least

$$\sum_{e \in G_i} x_e \geq \frac{1}{O(K)} \cdot \frac{1}{O(1)} \cdot \mu(G) = \frac{1}{O(K)} \cdot \mu(G).$$

But, as this multigraph G_i has at least $|E(G_i)| = \sum_{e \in G_i} \lceil x_e / \mathcal{R}^{-i+1} \rceil \geq \sum_{e \in G_i} x_e \cdot \mathcal{R}^{i-1}$ edges, one of the $2\Delta(G_i) - 1 \leq 2\mathcal{R}^{i+1}$ colors (matchings) in G_i must have size at least

$$\frac{|E(G_i)|}{2\Delta(G_i) - 1} \geq \frac{\sum_{e \in G_i} x_e \cdot \mathcal{R}^{i-1}}{2\mathcal{R}^i} \geq \frac{\sum_{e \in G_i} x_e}{4} \geq \frac{1}{4} \cdot \frac{1}{O(K)} \cdot \mu(G) = \frac{1}{O(K)} \cdot \mu(G).$$

As this algorithm's matching is the largest color class in all the edge colorings of all the different G_i, it is $O(K)$ approximate, as claimed. □

Corollary A.2. *There exists a deterministic $O\big(\frac{\log n}{\log \log n}\big)$-approximate matching algorithm with worst-case* $\operatorname{poly}\log n$ *update time.*

Remark 1: We note that the algorithm of Theorem 1.4 requires $O(m \cdot n^{1/K})$ space to store the multigraphs G_i and their relevant data structures maintained by the algorithm, since each edge e in a graph G_i may have x-value precisely $\mathcal{R}^{-i+1}$, which means we represent this edge using $O(\mathcal{R}) = O(n^{1/K})$ parallel edges in G_i. It would be interesting to see if its approximation to worst-case update time tradeoff can be matched by a deterministic algorithm requiring $\tilde{O}(m)$ space.

Remark 2: We note that the matching maintained by our deterministic algorithms can change completely between updates. For applications where this is undesirable, combining this algorithm with a recent framework of Solomon and Solomon [65] yields a dynamic matching M' of roughly the same size while only changing $O(1/\epsilon)$ edges of M' per update.

B Deferred proofs of Section 3

Here we provide complete proofs of lemmas deferred from the main paper body, restated below for convenience.

First, we show that Algorithm 1 samples each edge into H with the probability given by (1) with $\frac{\log n}{\epsilon^2}$ replaced by d, up to multiplicative $(1+\epsilon)$ terms.

Lemma 3.2. *If $d \geq \frac{1}{\epsilon}$ and $\gamma \geq 1$, then for every edge $e \in E$, we have*

$$\min\{1, x_e \cdot d\}/(1+\epsilon)^2 \leq \Pr[e \in H] \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon). \tag{3}$$

Moreover, if $x_e > \frac{1}{d}$, then $\Pr[e \in H] = 1$.

Proof. Let i be the integer for which $x_e \in ((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}]$. That is, the i for which $e \in E(G_i)$.

If $(1+\epsilon)^{i-1} < d$, implying that $(1+\epsilon)^i < d(1+\epsilon)$, then Algorithm 1 samples all of the $\gamma\lceil(1+\epsilon)^i\rceil = \min\{\gamma\lceil d(1+\epsilon)\rceil, \gamma\lceil(1+\epsilon)^i\rceil\}$ colors in the edge coloring of G_i. Consequently, the edge e is sampled with probability one. On the other hand, $(1+\epsilon)^{i-1} < d$ also implies that $(1+\epsilon)^{-i+1} > \frac{1}{d}$ and therefore that $x_e > (1+\epsilon)^{-i} \geq \frac{1}{d(1+\epsilon)}$. Thus, the edge e is sampled with probability at most

$$\Pr[e \in H] = 1 \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon),$$

and trivially sampled with probability at least

$$\Pr[e \in H] = 1 \geq \min\{1, x_e \cdot d\}/(1+\epsilon)^2.$$

Moreover, if $x_e > \frac{1}{d}$, then $(1+\epsilon)^{-i+1} \geq x_e > \frac{1}{d}$, or put otherwise $(1+\epsilon)^{i-1} < d$, and so we find that every edge e with $x_e > \frac{1}{d}$ is sampled with probability $\Pr[e \in H] = 1 (= \min\{1, x_e \cdot d\})$. It remains t consider edges e with $x_e \leq \frac{1}{d}$, for which $\min\{1, x_e \cdot d\} = x_e \cdot d$, and which in particular belong to subgraphs G_i with i satisfying $(1+\epsilon)^{i-1} > d$.

Now, if i satisfies $(1+\epsilon)^{i-1} \geq d$, then we sample some $\gamma\lceil d \rceil = \min\{\gamma\lceil d \rceil, \lceil \gamma \cdot (1+\epsilon)^i \rceil\}$ colors in the edge coloring of G_i. As such, the probability of e appearing in H is precisely the probability that the color M containing e is one of the $\gamma\lceil d \rceil$ sampled colors in G_i, which by linearity of expectation happens with probability precisely

$$\Pr[e \in H] = \frac{\gamma\lceil d \rceil}{\gamma\lceil (1+\epsilon)^i \rceil} = \frac{\lceil d \rceil}{\lceil (1+\epsilon)^i \rceil}.$$

Now, since $d \geq \frac{\epsilon}{\epsilon}$ implies that $d+1 \leq d(1+\epsilon)$, the probability of e (which has $x_e \geq (1+\epsilon)^{-i}$) appearing in H is at most

$$\Pr[e \in H] = \frac{\lceil d \rceil}{\lceil (1+\epsilon)^i \rceil} \leq \frac{d+1}{(1+\epsilon)^i} \leq \frac{d(1+\epsilon)}{(1+\epsilon)^i} = d(1+\epsilon)^{-i+1} \leq x_e \cdot d \cdot (1+\epsilon).$$

On the other hand, since $(1+\epsilon)^{i-1} > d$, and $d \geq \frac{1}{\epsilon}$, we have that $(1+\epsilon)^i > d \geq \frac{1}{\epsilon}$, which implies that $(1+\epsilon)^i + 1 \leq (1+\epsilon)^{i+1}$. Consequently, the probability of e (which has $x_e \leq (1+\epsilon)^{-i+1}$) appearing in H is at least

$$\Pr[e \in H] = \frac{\lceil d \rceil}{\lceil (1+\epsilon)^i \rceil} \geq \frac{d}{(1+\epsilon)^i + 1} \geq \frac{d}{(1+\epsilon)^{i+1}} = d(1+\epsilon)^{-i-1} \geq x_e \cdot d/(1+\epsilon)^2.$$

This completes the proof for edge e in G_i for i satisfying $(1+\epsilon)^{i-1} \geq d$, as such edges e satisfy $(1+\epsilon)^{-i+1} \leq \frac{1}{d}$ and consequently $\min\{1, x_e \cdot d\} = x_e \cdot d$. □

B.1 The Dynamic Matching Framework

Here we give all details of our framework for obtaining dynamic matching algorithms against adaptive adversaries whose approximation ratio matches that of the subgraph H output by Algorithm 1.

Theorem 3.7. *Let $\gamma \geq 1$, $d \geq 1$ and $\epsilon > 0$. Let $\mathcal{A}_f$ be a constant-approximate dynamic fractional matching algorithm with update time $T_f(n,m)$. Let $\alpha = \alpha(d, \epsilon, \gamma, \mathcal{A}_f)$ be the approximation ratio of the subgraph H output by Algorithm 1 with parameters d, ϵ and γ when run on the fractional matching of $\mathcal{A}_f$. Let $\mathcal{A}_c$ be a dynamic $\gamma\Delta$-edge-coloring algorithm with update time $T_c(n,m)$. If the guarantees of $\mathcal{A}_f$ and $\mathcal{A}_c$ hold against an adaptive adversary, then there exists an $\alpha(1+O(\epsilon))$-approximate dynamic matching algorithm $\mathcal{A}$ against an* **adaptive** *adversary, with update time*

$$O\left(T_f(n,m) \cdot T_c(n,m) + \log(n/\epsilon) \cdot \gamma \cdot d/\epsilon^3\right).$$

Moreover, if $\mathcal{A}_f$ and $\mathcal{A}_c$ have worst-case update times, so does $\mathcal{A}$, and if the approximation ratio given by H is w.h.p., then so is the approximation ratio of $\mathcal{A}$.

Before proving this lemma, we first prove a simple intermediate lemma.

Lemma B.1. *Let $\vec{x}$ be a fractional matching in some graph G. Let $\mathcal{H}$ be the distribution over subgraph H of G obtained by running Algorithm 1 on $\vec{x}$ with parameters d, ϵ and γ. Then, if the edge colorings of Algorithm 1 based on $\vec{x}$ and the above parameters are given, we can sample a graph H sampled from $\mathcal{H}$, and compute a $(1+\epsilon)$-approximate matching in H, in time*

$$O\left(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\right).$$

Proof. Running Algorithm 1 on x with parameters d, ϵ, γ yields a subgraph H of G. By Observation 3.1, for any edge colorings as computed by Algorithm 1, this subgraph H has size at most

$$|E(H)| = O\left(\frac{\log(n/\epsilon)}{\epsilon} \cdot \gamma \cdot d \cdot \mu(G^{(t)})\right). \tag{4}$$

As determining which d colors to sample in each of the $O(log(n/\epsilon))$ subgraphs take $O(\log(n/\epsilon) \cdot d)$ time, sampling this graph H takes at most $O\big(\frac{\log(n/\epsilon)}{\epsilon} \cdot \gamma \cdot d \cdot \mu(G)\big)$ time (including the time to make these random choices). Furthermore, computing a $(1+\epsilon)$-approximate matching in this H, using an $O(m/\epsilon)$-time static algorithm [48, 56], takes time

$$O(|E(H)|/\epsilon) = O\left(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\right) \qquad \square$$

Our algorithm of Theorem 3.7 will periodically appeal to Lemma B.1, whose work it will "spread" across epochs of length $O(\lceil \epsilon \cdot \mu(G) \rceil)$, as follow.

Proof of Theorem 3.7. Algorithm $\mathcal{A}$ runs Algorithm $\mathcal{A}_f$ to maintain a fractional matching $\vec{x}$. In addition, it maintains $\lceil \gamma(1+\epsilon)^i \rceil$-edge-colorings in each subgraph $G_i := G[\{e \mid x_e \in (1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}\}]$, for all $i = 1, 2, \ldots, 2\log_{1+\epsilon}(n/\epsilon) = O\big(\frac{\log(n/\epsilon)}{\epsilon}\big)$. Maintaining this fractional matching and the different subgraphs' edge colorings appropriately require at most $O(T_f(n,m) \cdot T_c(n,m))$ time per update: $T_c(n,m)$ time for each of the at most $O(T_f(n,m))$ edge value changes $\mathcal{A}_f$ makes to the fractional matching $\vec{x}$ per update, as well as $T_f(n,m)$ time to update $\vec{x}$ and for maintaining $\sum_e x_e$. By Lemma B.1, the above edge colorings allow us to sample a subgraph H from the distribution $\mathcal{H}$ of subgraphs obtained when running Algorithm 1 on $G^{(t)}$, in time $O\left(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\right)$. We perform such computations periodically. In particular, we divide time into *epochs*, where during each epoch we spread the work of computing such a matching, as follows.

The first epoch begins after the first edge insertion. We denote by $G^{(t)}$ the graph G at the beginning of epoch t. Likewise, we denote by $\vec{x}^{(t)}$ the fractional matching at the beginning of epoch t. As $\vec{x}^{(t)}$ is a β-approximate fractional matching for some constant $\beta = O(1)$ by our hypothesis, we have that $|x^{(t)}|_1 \geq \frac{1}{\beta} \cdot \mu(G^{(t)})$. On the other hand, by the integrality gap of the fractional matching polytope, we also have that $|x^{(t)}|_1 \leq \frac{3}{2} \cdot \mu(G^{(t)})$. We will let epoch t have length $\lceil \epsilon \cdot |x^{(t)}|_1 \rceil$, which by the above is $O(\epsilon \cdot \mu(G^{(t)}))$.

If the fractional matching at the beginning of epoch t has value at most $|x^{(t)}|_1 \leq \frac{1}{\epsilon}$, the epoch has length one. We compute our matching with which to answer queries in this epoch by sampling $H^{(t)}$ and computing a $(1+\epsilon)$-approximate matching in $H^{(t)}$. By Lemma B.1 takes $O\left(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G^{(t)})\right) = O\left(\frac{\log(n/\epsilon)}{\epsilon^3} \cdot \gamma \cdot d\right)$ time, which falls within our claimed time bounds. Moreover, our matching at this point is an $\alpha(1+\epsilon)$-approximate matching in $G^{(t)}$, as desired.

For an epoch with $|x^{(t)}| > \frac{1}{\epsilon}$, which we will term *long*, we will compute $H^{(t)}$ and a $(1+\epsilon)$-approximate matching in H, but spread this work over the length of the epoch. We let the non-deleted edges of the previous matching (which are computed by the beginning of this long period)

be the matching used for queries during this epoch. One observation we will rely on for the analysis of long epochs is that if epoch t or $t+1$ is long, then since epochs t has length $O(\epsilon \cdot \mu(G))$, the maximum matching's cardinalities in $G^{(t)}$ and $G^{(t+1)}$ are similar. In particular, since we have that $\mu(G^{(t+1)}) - \mu(G^{(t)}) \leq O(\epsilon \cdot \mu(G^{(t)}))$, we have

$$\mu(G^{(t)}) \cdot (1 - O(\epsilon)) \leq \mu(G^{(t+1)}) \leq \mu(G^{(t)}) \cdot (1 + O(\epsilon)). \tag{5}$$

We now describe the steps taken during a long epoch.

During a long epoch, we sample a subgraph $H^{(t)}$ in $G^{(t)}$ and spread the time of computation of $H^{(t)}$ and this $(1+\epsilon)$-approximate matching in $H^{(t)}$ over the epoch. Ignoring some additional information we need to maintain to do this, this increases the update time by

$$\frac{O\left(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G^{(t)})\right)}{\lceil \epsilon |x^{(t)}|_1 \rceil} \leq \frac{O\left(\frac{\log(n/\epsilon)}{\epsilon^2} \cdot \gamma \cdot d \cdot \mu(G^{(t)})\right)}{\epsilon \cdot \frac{1}{\beta} \cdot \mu(G^{(t)})} = O\left(\frac{\log(n/\epsilon)}{\epsilon^3} \cdot \gamma \cdot d\right),$$

where the first inequality relied on $|x^{(t)}|_1 \geq \frac{1}{\beta} \cdot \mu(G^{(t)})$ and the last inequality relied on $\beta = O(1)$. Now, in order to perform these operations efficiently during the epoch, we need to maintain the edge colorings *at the beginning of the epoch*. This, however, is easily done by maintaining a mapping (using arrays and lists) from colors in each subgraph to a list of edges added/removed from this color during the epoch. This allows us to maintain $\vec{x}$ and the colorings induced by it, as well as maintain the colorings at the beginning of the epoch, at a constant overhead in the time to update $\vec{x}$ and the colorings, as well as the time to sample $H^{(t)}$. Finally, if space is a concern,[6] the list of updates from epoch t can be removed during the subsequent epoch in the same asymptotic time per step as required by the following epoch, as subsequent epochs have the same asymptotic length. We conclude that our algorithm runs within the claimed time bounds. It remains to analyze its approximation ratio for long epochs.

Recall that for any long epoch t, we use the non-deleted edges of some $(1+\epsilon)$-approximate matching $M^{(t-1)}$ in $H^{(t-1)}$ as our matching during epoch t. (Note that we have finished computing $M^{(t-1)}$ by the beginning of epoch t.) We now bound this algorithm's approximation ratio. By assumption we have that $\mu(H^{(t-1)}) \geq \frac{1}{\alpha} \cdot \mu(G^{(t-1)})$ at the beginning of the phase. Denote by $M \subseteq M^{(t-1)}$ the matching used to answer queries during some time point in epoch t. As M contains all edges of $M^{(t-1)}$ (which is a $(1+\epsilon)$-approximate matching in $H^{(t-1)}$), not including the edges of $M^{(t-1)}$ removed during epochs $t-1$ and t (of which there are at most $\lceil \epsilon \cdot |x^{(t-1)}|_1 \rceil + \lceil \epsilon \cdot |x^{(t)}|_1 \rceil$), we find that the size of M during any point in epoch t is at least

$$\begin{aligned}
|M| &\geq |M^{(t-1)}| - \lceil \epsilon \cdot |x^{(t-1)}|_1 \rceil - \lceil \epsilon \cdot |x^{(t)}|_1 \rceil \\
&\geq \frac{1}{1+\epsilon} \cdot \mu(H^{(t-1)}) - \lceil \epsilon \cdot |x^{(t-1)}|_1 \rceil - \lceil \epsilon \cdot |x^{(t)}|_1 \rceil \\
&\geq \frac{1}{\alpha(1+\epsilon)} \cdot \mu(G^{(t-1)}) - \left\lceil \epsilon \cdot \frac{3}{2} \cdot \mu(G^{(t-1)}) \right\rceil - \left\lceil \epsilon \cdot \frac{3}{2} \cdot \mu(G^{(t)}) \right\rceil \\
&\leq \frac{1}{\alpha(1+O(\epsilon))} \cdot \mu(G^{(t)})
\end{aligned}$$

where the second inequality follows from $|x^{(t)}|_1 \leq \frac{3}{2} \cdot \mu(G^{(t)})$ for all t, and the ultimate inequality follows from subsequent epochs' maximum matchings' cardinalities being similar, as stated in Equation (5). Consequently, our algorithm is $\alpha(1+O(\epsilon))$ approximate, as claimed. □

[6] And why wouldn't it be?

B.2 Analysis of Algorithm 1

In this section we analyze the approximation ratio of the subgraph H sampled by Algorithm 1. Combined with Theorem 3.7, this will imply our randomized algorithms against adaptive adversaries.

In Appendix B.2.1 we prove that kernels contain large integral matchings. In Sections B.2.2 and B.2.3 we show that our sparsification applied to approximately-maximal fractional matchings with appropriate choices of d yield such a kernel. Specifically, we show that it yields either a deterministic kernel w.h.p., or a randomized kernel. Combined with Theorem 3.7 (and some other ideas), we use these sections to prove Theorems 1.1 and 1.2, respectively. Finally, in Appendix B.2.4, we prove that the subgraph output by our sparsification method is a good fractional matching algorithm, from which we derive Theorem 1.3.

B.2.1 The Kernel Lemma

Recall that one notion of matching sparsifiers we rely on is (randomized) (c, d) kernels, restated below for ease of reference.

Definition 4.1. *(Kernels) A* (c, d, ϵ)-kernel *of a graph G is a (random) subgraph $\mathcal{H}$ of G satisfying:*

1. *For each vertex $v \in V$, the degree of v in $\mathcal{H}$ is at most $d_{\mathcal{H}}(v) \leq d$ always.*
2. *For each edge $e \in E$ with $\Pr[e \notin \mathcal{H}] > \epsilon$, it holds that $\mathbb{E}[\max_{v \in e} d_{\mathcal{H}}(v) \mid e \notin \mathcal{H}] \geq d/c$.*

If $\mathcal{H}$ is a deterministic distribution, we say $\mathcal{H}$ is a deterministic kernel.

Here we bound the expected maximum matching size of such a kernel in terms of $\mu(G)$.

Lemma 4.2. *Let $\mathcal{H}$ be a (c, d, ϵ)-kernel of G for some $c \geq \frac{1}{1-\epsilon}$. Then $\mathbb{E}[\mu(\mathcal{H})] \geq \frac{1}{2c(1+1/d)} \cdot \mu(G)$.*

Proof. Let M^* be some maximum matching in G (i.e., $|M^*| = \mu(G)$). For any realization H of $\mathcal{H}$, consider the following fractional matching:

$$f^H_{u,v} \triangleq \begin{cases} \frac{1}{d} & (u,v) \in H \setminus M^* \\ \max\{1 - \frac{d_h(u)+d_H(v)-2}{d}, 0\} & (u,v) \in H \cap M^*. \end{cases}$$

This is a feasible fractional matching due to the degree bound of H and the fractional values assigned to edges of a vertex v incident on an edge $e \in H \cap M^*$ being at most $\frac{d_H(v)-1}{d} + \frac{d-d_H(v)+1}{d} = 1$. We start by showing this fractional matching has high expected value, $\mathbb{E}_{H\sim\mathcal{H}}[\sum_e f^H_e]$.

To lower bound the above expected value, we consider the following variables, $y^H_v \triangleq \sum_{e \ni v} f^H_e$. By the handshake lemma, $\sum_{u,v} f^H_{u,v} = \frac{1}{2}\sum_v y^H_v$. Now, consider some edge $e = (u, v) \in M^*$. For any realization H of $\mathcal{H}$ with $e \in M^* \cap H$, we have $y^H_u + y^H_v \geq 1 (\geq \frac{1}{c})$ by construction. Therefore if $\Pr[e \notin \mathcal{H}] \leq \epsilon$, we have $\mathbb{E}[y^H_u + y^H_v] \geq 1 - \epsilon \geq \frac{1}{c}$ (by our choice of $c \geq \frac{1}{1-\epsilon}$). On the other hand, if $e \in M^* \setminus H$, then we have $y^H_u + y^H_v \geq \max_{v \in e} y^H_v \geq \max_{v \in e} d_H(v)/d$. But by the second property of (c, d, ϵ)-kernels we have that if $\Pr[e \notin \mathcal{H}] > \epsilon$, then $\mathbb{E}_{H\sim\mathcal{H}}[\max_{v\in e} d_H(v) \mid e \notin H] \geq d/c$. Consequently, we find that for each edge $e = (u, v) \in M^*$ with $\Pr[e \notin \mathcal{H}] > \epsilon$ we have

$$\begin{aligned}
\mathbb{E}_{H\sim\mathcal{H}}\left[y^H_u + y^H_v\right] &= \mathbb{E}_{H\sim\mathcal{H}}\left[y^H_u + y^H_v \mid e \in H\right] \cdot \Pr[e \in H] + \mathbb{E}_{H\sim\mathcal{H}}\left[y^H_u + y^H_v \mid e \notin H\right] \cdot \Pr[e \notin H] \\
&\geq \frac{1}{c} \cdot \Pr[e \in H] + \mathbb{E}_{H\sim\mathcal{H}}\left[\max_{v\in e} d_H(v)/d \mid e \notin H\right] \cdot \Pr[e \notin H] \\
&\geq \frac{1}{c} \cdot \Pr[e \in H] + \frac{d}{c} \cdot \frac{1}{d} \cdot \Pr[e \notin H] \\
&= \frac{1}{c}.
\end{aligned}$$

Now, as each vertex v neighbors at most one edge of the (optimal) matching M^*, we obtain

$$\mathbb{E}_{H\sim\mathcal{H}}\left[\sum_e f_e^H\right] = \frac{1}{2}\cdot\mathbb{E}_{H\sim\mathcal{H}}\left[\sum_v y_v^H\right] \geq \frac{1}{2c}\cdot|M^*| = \frac{1}{2c}\cdot\mu(G). \tag{6}$$

Therefore, $\mathcal{H}$ contains a large ($2c$-approximate) fractional matching in expectation.

Next, to show that $\mathcal{H}$ contains a large *integral* matching in expectation, we rely on Vizing's Theorem [70], which asserts that every multigraph of maximum degree Δ and maximum edge multiplicity μ has a proper $\Delta+\mu$ edge-coloring; i.e., a partition of the edge multiset into $\Delta+\mu$ edge-disjoint matchings. To use this theorem, we again consider a realization H of $\mathcal{H}$, and now construct a multigraph on the same vertex set V with each edge e replaced by $f_e^H\cdot d$ parallel copies (note that $f_e^H\cdot d$ is integral). By construction, the number of edges in this multigraph is $\sum_e f_e^H\cdot d$. By feasibility of f^H, we have that this multigraph has maximum degree at most $\max_v\sum_{e\ni v} f_v^H\cdot d\leq d$. By Vizing's Theorem, the simple subgraph obtained by ignoring parallel edges corresponding to edges in $H\cap M^*$ can be edge colored using $d+1$ colors. But for each edge $e=(u,v)\in H\cap M^*$, such a coloring uses at most $d_H(u)-1+d_H(v)-1$ distinct colors on edges other than (u,v) which are incident on u or v. To extend this $d+1$ edge coloring to a proper coloring of the multigraph, we color the $\max\{d-(d_H(u)-1+d_H(v)-1),0\}$ multiple edges (u,v) in this multigraph using some $\max\{d-(d_H(u)-1+d_H(v)-1),0\}$ colors of the palette of size $d+1$ which were not used on the other edges incident on u and v. We conclude that this multigraph, which is contained in H and has $\sum_e f_e\cdot d$ edges, is $d+1$ edge colorable and therefore one of these $d+1$ colors (matchings) in this edge coloring is an integral matching in H of size at least

$$\mu(H)\geq\frac{1}{d+1}\cdot\sum_e f_e^H\cdot d = \frac{1}{1+1/d}\cdot\sum_e f_e^H. \tag{7}$$

Taking expectation over $H\sim\mathcal{H}$ and combining (7) with (6), we obtain the desired result, namely

$$\mathbb{E}[\mu(\mathcal{H})]\geq\frac{1}{1+1/d}\cdot\mathbb{E}_{H\sim\mathcal{H}}\left[\sum_e f_e^H\right]\geq\frac{1}{2c(1+1/d)}\cdot\mu(G). \qquad \square$$

B.2.2 Kernels - w.h.p.

In this section we show that running Algorithm 1 on an approximately-maximal fractional matching $\vec{x}$ with sufficiently large $d=\Omega(\log n)$ yields a deterministic kernel, w.h.p. We then use this property and Theorem 3.7 to prove our main result, Theorem 1.1.

Lemma 4.4. *Let $c\geq 1$, $\epsilon>0$ and $d\geq\frac{9c(1+\epsilon)^2\cdot\log n}{\epsilon^2}$. If $\vec{x}$ is a (c,d)-approximately-maximal fractional matching, then the subgraph H output by Algorithm 1 when run on $\vec{x}$ with ϵ and d as above is a deterministic $(c(1+O(\epsilon)),d(1+O(\epsilon)),0)$-kernel, w.h.p.*

Proof. Consider some vertex v. By Lemma 3.2, we have that each edge $e\ni v$ is sampled with probability at most $\Pr[X_e=1]\leq\min\{1,x_e\cdot d\}\cdot(1+\epsilon)$. Combined with the fractional matching constraint $\sum_{e\ni v}x_e\leq 1$, this implies that the expected degree of v in H is at most

$$\mathbb{E}[d_H(v)]=\sum_{e\ni v}\mathbb{E}[X_e]\leq\sum_e x_e\cdot d\cdot(1+\epsilon)\leq d(1+\epsilon).$$

By Lemma 3.3, we have that the indicators $\{X_e\mid e\ni v\}$ are NA. Therefore, appealing to the upper tail bound of Lemma 2.2 for NA variables with $\delta=\epsilon>0$, we have that

$$\Pr[d_H(v)\geq d(1+3\epsilon)]\leq\Pr[d_H(v)\geq d(1+\epsilon)^2]\leq\exp\left(\frac{-\epsilon^2 d(1+\epsilon)}{3}\right)\leq\frac{1}{n^3},$$

where the last inequality relied on $d \geq \frac{9 \log n}{\epsilon^2}$.

Now, to prove the second property of kernels, consider some edge $e \in E$ such that $\Pr[e \notin H] > 0$. By Lemma 3.2, we have that $x_e \leq 1/d$. Therefore, as $\vec{x}$ is (c, d)-approximately-maximal, this implies that there exists some $v \in e$ with $\sum_{e' \ni v} x_{e'} \geq \frac{1}{c}$ and $x_{e'} \leq \frac{1}{d}$ for all $e' \ni v$. Therefore, by Lemma 3.2 each edge $e' \ni v$ is sampled with probability at least $x_e \cdot d/(1+\epsilon)$, and so by linearity of expectation, the expected degree of v in H is at least

$$\mathbb{E}[d_H(v)] = \sum_{e \ni v} \mathbb{E}[X_e] \geq \sum_e x_e \cdot d/(1+\epsilon) \geq d/(c(1+\epsilon)^2).$$

Recalling that the indicators $\{X_e \mid e \ni v\}$ are NA, we appeal to the lower tail bound of Lemma 2.2 with $\delta = \epsilon > 0$, from which we obtain that

$$\Pr[d_H(v) \leq d(1-\epsilon)/(c(1+\epsilon)^2)] \leq \exp\left(\frac{-\epsilon^2 d/(c(1+\epsilon))}{2}\right) \leq \frac{1}{n^3},$$

where the last inequality follows from $d \geq \frac{9c(1+\epsilon)^2 \log n}{\epsilon^2}$.

Taking union bound over the $O(n^2)$ bad events which would make H not be kernel as desired, we find that H is a $(c(1+O(\epsilon)), d(1+O(\epsilon), 0)$-kernel with high probability, as claimed. □

As show in [3], the output fractional matching of [22] is precisely such a fractional matching (also satisfying $d = \text{poly}(\log n, 1/\epsilon) \geq \frac{9c(1+\epsilon)^2 \log n}{\epsilon^2}$). Therefore, by Lemma 4.4, plugging in this $\text{poly}(\log n, 1/\epsilon)$ worst-case update time deterministic algorithm into Theorem 3.7, we obtain a Monte Carlo algorithm which is $(2+\epsilon)$ approximate w.h.p. and has worst-case update time $\text{poly}(\log n, 1/\epsilon)$. We now show how we can even obtain a Las Vegas algorithm from this approach which is always $(2+\epsilon)$ approximate and has polylog update time in expectation and w.h.p.

To get Las Vegas algorithms from our framework, we note that the bad events which can make H not be a kernel, analyzed in the proof of Lemma 4.4, can be tested in the same asymptotic time required to compute H. Inspecting the proof of Theorem 3.7, we see that spreading this work over epochs, too, this extra testing of whether H is a kernel only increases our update time by a multiplicative constant. We now address how to verify that H is a kernel.

First, we verify that the degrees of non-isolated vertices in H are at most $d(1+3\epsilon)$, in $O(|E(H)|)$ time. Next, for vertices v with $\sum_{e \ni v} x_e \geq 1/c$ and $x_e \leq 1/d$ for all $e \ni v$, we test if their degree in H is at least roughly d/c. Since there are at most $O(c \cdot \mu(G))$ such vertices, since $\sum_e x_e \leq \frac{3}{2}\mu(G)$, this takes $O(\mu(G))$ time for constant c. (This also requires maintaining a list of such vertices, easily done in constant worst-case time per update of some x_e.) By the proof of Lemma 4.4, w.h.p., none of these events which imply H is not be a kernel will occur. We can therefore obtain Las Vegas guarantees for our dynamic algorithms, by simply re-sampling H if ever it is not a kernel. This yields our main result, restated below.

Theorem 1.1. *For every $\epsilon \in (0, 1/2)$, there exists a (Las Vegas) randomized $(2+\epsilon)$-approximate algorithm with update time* $\text{poly}(\log n, 1/\epsilon)$ *w.h.p. against an* adaptive *adversary.*

B.2.3 Constant-Time Integral matching sparsifiers

In this section we show how to apply our framework to the recent fractional matching algorithm of [23] to obtain a constant-approximate algorithm with constant (amortized) update time. To this end, we start by showing that running Algorithm 1 with $d = \Omega(\log(1/\epsilon)/\epsilon^2)$ on a (c, d)-approximately-maximal fractional matching, and removing all edges of high-degree vertices in the output graph, yields a randomized kernel.

Lemma B.2. *Let $\epsilon \in (0, 1/2)$, $c \geq \frac{1}{1-\epsilon}$ and $d \geq \frac{3 \cdot \log(2/\epsilon)}{\epsilon^2}$. Let $\mathcal{H}$ be the distribution of subgraphs output by Algorithm 1 when run on a (c, d)-approximately maximal fractional matching $\vec{x}$ with ϵ and d as above. For any realization H of $\mathcal{H}$, we let H' be a graph obtained by removing all edges of vertices v of degree $d_H(v) > d(1+3\epsilon)$. Then the distribution $\mathcal{H}'$ over H' is a $(c(1+O(\epsilon)), d(1+3\epsilon), \epsilon)$-kernel.*

Proof. The fact that $\mathcal{H}'$ satisfies the first property of such a kernel is immediate, as we remove all edges of vertices of degree above $d(1+3\epsilon)$ in H to obtain H'.

First, we characterize edges e with $\Pr[e \notin \mathcal{H}'] > \epsilon$. By Lemma 3.2 we have that the expected degree of any vertex v in $\mathcal{H}$ is at most

$$\mathbb{E}[d_{\mathcal{H}}(v)] = \sum_{e \ni v} \mathbb{E}[X_e] \leq \sum_e x_e \cdot d(1+\epsilon) \leq d(1+\epsilon).$$

We apply the upper tail bound of Lemma 2.2 with $\delta = \epsilon$ to $d_{\mathcal{H}}(v) = \sum_{e \ni v} X_e$, which is the sum of NA variables, by Lemma 3.3. Relying on $d \geq \frac{3\log(2/\epsilon)}{\epsilon^2}$ and $\epsilon \leq \frac{1}{2} \leq 1$, this bound yields

$$\Pr[d_{\mathcal{H}}(v) > d(1+3\epsilon)] \leq \Pr\left[d_{\mathcal{H}}(v) \geq d(1+\epsilon)^2\right] \leq \exp\left(\frac{-\epsilon^2 \cdot d(1+\epsilon)}{2}\right) \leq \epsilon/2.$$

Therefore, by union bound, since $e \in H \setminus H'$ only if one (or both) of its two endpoints have degree above $d(1+3\epsilon)$ in H, we find that

$$\Pr[(u,v) \in H \setminus H'] \leq \Pr[d_{\mathcal{H}}(u) > d(1+3\epsilon)] + \Pr[d_{\mathcal{H}}(v) > d(1+3\epsilon)] \leq \epsilon. \tag{8}$$

From (8) we find that if $\Pr[e \notin \mathcal{H}'] > \epsilon$, then $\Pr[e \notin \mathcal{H}] > 0$. From this we deduce that $x_e < 1/d$, by Lemma 3.2. By the (c, d)-approximate-maximality of $\vec{x}$, this implies that e is incident on some vertex v such that for each edge $e' \ni v$ we have $x_{e'} \leq 1/d$ and $\sum_{e' \ni v} x_{e'} \geq 1/c$. Consider this vertex v. In order to bound the expected degree of v in H', we first now consider the event that some edge (u, v) of v sampled in H does not appear in H'.

Consider an edge (u, v). By Lemma 3.2 and Corollary 3.5, together with $\sum_{e \ni v} x_e \leq 1$, we find that conditioned on the edge (u, v) appearing in H, the vertex u has expected degree in H at most

$$\mathbb{E}[d_H(u) \mid (u,v) \in H] \leq \sum_{e \ni u} \mathbb{E}[X_e \mid (u,v) \in H] \leq 1 + \sum_{e \ni u, e \neq (u,v)} x_e \cdot d \cdot (1+\epsilon) \leq 1 + d(1+\epsilon).$$

We now recall that the variables $\{Y_e \triangleq [X_e \mid X_{(u,v)} = 1] \mid e \ni v\}$ are NA. Therefore, appealing to the upper tail bound of Lemma 2.2 for these NA variables Y_e with $\delta = \epsilon > 0$, we have that

$$\begin{aligned}\Pr[d_H(v) > d(1+3\epsilon) \mid X_{(u,v)} = 1] &\leq \Pr[d_H(v) \geq (1 + d(1+\epsilon)) \cdot (1+\epsilon) \mid X_{(u,v)} = 1] \\ &\leq \exp\left(\frac{-\epsilon^2(1+d(1+\epsilon))}{3}\right) \leq \epsilon/2,\end{aligned}$$

where the last inequality relied on $d \geq \frac{3\log(2/\epsilon)}{\epsilon^2}$. Denoting by B_u the bad event that u has more than $d(1+3\epsilon)$ edges in H, we have that $\Pr[B_u \mid X_{(u,v)} = 1] \leq \epsilon/2$.

Since each edge (u, v) in H is also in H' only if both B_u and B_v do not happen, we have that the degree of v in H' is at least $d_{H'}(v) \geq \sum_{(u,v)} X_{(u,v)} \cdot (1 - \mathbb{1}[B_u] - \mathbb{1}[B_v])$. But by Lemma 3.2, all edges $e = (u, v)$ containing v (which have $x_e \leq \frac{1}{d}$) are sampled with probability $\Pr[X_e = 1] \geq \min\{1, x_e \cdot d\}/(1+\epsilon)^2 = x_e \cdot d/(1+\epsilon)^2$, and so the expected degree of v in H' is at least

$$\begin{aligned}\mathbb{E}[d_{H'}(v)] &\geq \sum_{(u,v)} \mathbb{E}[X_{(u,v)}] \cdot (1 - \Pr[B_u \mid X_{(u,v)} = 1] - \Pr[B_v \mid X_{(u,v)} = 1]) \\ &\geq \sum_e \left(x_e \cdot d/(1+\epsilon)^2\right) \cdot (1-\epsilon) \geq \frac{1}{c} \cdot (d/(1+\epsilon)^2) \cdot (1-\epsilon).\end{aligned}$$

Thus, for every e with $\Pr[e \notin \mathcal{H}'] > \epsilon$ we have $\mathbb{E}[\max_{v \in e} d_{H'}] \geq d(1+3\epsilon)/c(1+O(\epsilon))$, as required. □

In order to obtain a constant-time algorithm using Lemma B.2, we need in particular some approximately-maximal fractional matching algorithm with constant update time. As it so happens, the algorithm of Bhattacharya and Kulkarni [23] is precisely such an algorithm. As the structure of the fractional matching output by this algorithm will prove useful in several ways for our analysis, we take a moment to outline this fractional matching's structure.

We say a dynamic fractional matching algorithm maintains a (β, c)-*hierarchical partition* if it assigns each vertex v a *level* ℓ_v, and each edge e an $x-$value $x_e = \beta^{-\ell_e}$, where $\ell_e = \max_{v \in e}\{\ell_v\} \pm O(1)$, for some constant β. The second property this fractional matching must guarantee is that each vertex v with $\ell_v > 0$ has $\sum_{e \ni v} x_e \geq 1/c$. Most prior dynamic fractional matching algorithms [17, 22, 23, 40], including that of [23], follow this approach, originally introduced by [19].

We first use the above structure of the fractional matching of [23] to show that it is approximately-maximal.

Lemma B.3. *There exists a deterministic $(1+\epsilon, d)$-approximately-maximal fractional matching algorithm, for any $d > 1+\epsilon$, with amortized update time $O(1/\epsilon^2)$.*

Proof. The algorithm we consider is precisely that of [23]. As the update time of this algorithm was proven in [23], it remains only to prove that it outputs an approximately-maximal fractional matchings as stated.

The algorithm of Bhattacharya and Kulkarni [23] maintains a $((1+\epsilon),(1+\epsilon))$-hierarchical partition with $x_e = (1+\epsilon)^{-\max_{v \in e}\{\ell_v\}-1}$. (This -1 term in the exponent is due to scaling down of the fractional matching to ensure it does not violate the fractional matching constraints.) For such a partition, we have that for any value $d \geq 1+\epsilon$, we have that any edge e with $x_e \leq \frac{1}{d}$ must have an endpoint $v \in e$ of level $\ell_v \geq \log_{1+\epsilon}(d) - 1$. But then all other incident edges $e' \ni v$ have x-value at most $x_{e'} \leq (1+\epsilon)^{-\ell_v - 1} \leq \frac{1}{d}$. Moreover, since the level of v is at least $\ell_v \geq \log_{1+\epsilon}(d) - 1 > 0$ (by our choice of $d > 1+\epsilon$), we also have that $\sum_{e' \ni v} x_{e'} \geq \frac{1}{c}$. In other words, the fractional matching $\vec{x}$ output by the algorithm of [23] is $(1+\epsilon, d)$-approximately-maximal. □

Lemmas B.2 and B.3 together with Theorem 3.7 imply a $(2+\epsilon)$-approximate dynamic algorithm with logarithmic update time against adaptive adversaries. We now explain how to obtain such an approximation in *constant* update time.

We note that any (β, c)-hierarchical partition must has at most $O(c \cdot \mu(G))$ vertices v of level $\ell_v > 0$. To see this, recall that all such vertices have $\sum_{e \ni v} x_e \geq 1/c$. Therefore,

$$\sum_{e \in E} x_e \geq \frac{1}{2} \sum_{v:\, \ell_v > 0} \sum_{e \ni v} x_e \geq \frac{1}{2c} \cdot |\{v \mid \ell_v > 0\}.$$

But since the integrality gap of the fractional matching polytope is at most $\frac{3}{2}$, we also have that

$$\frac{3}{2} \cdot \mu(G) \geq \sum_{e \in E} x_e \geq \frac{1}{2c} \cdot |\{v \mid \ell_v > 0\}|.$$

That is, for constant c as we consider, the number of vertices of level $\ell_v > 0$ is at most $O(\mu(G))$. This implies in particular that there are only $O(\mu(G))$ distinct levels assigned to vertices. But an edge's value is determined (possibly up to a constant additive term in the exponent) by the level of its highest-level endpoint. Therefore, there are only $O(\mu(G))$ many values $\max_{v \in e}\{\ell_v\}$ can take. Consequently, there are thus only $O(\mu(G))$ values any x_e can take. Hence, when running Algorithm 1 on $\vec{x}$ we only sample edges from $O(\mu(G))$ edge colorings of subgraphs G_i (which are induced by edges of similar x_e value). Thus, if we sample $d = \text{poly}(1/\epsilon)$ colors per (non-empty)

subgraph G_i, the choice of colors to sample can be done in $O(\mu(G)/\text{poly}(\epsilon))$ time, yielding a graph of size $|E(H)| = O(\mu(G)/\text{poly}(\epsilon))$. Extending the argument of Theorem 3.7 appropriately, using a 3Δ-edge-coloring algorithm with constant expected update time and the fractional matching algorithm of [23], together with Lemma 4.5 we obtain a $(2+\epsilon)$-approximate dynamic algorithm with *constant* update time.

Theorem 1.2. *For every $\epsilon \in (0, 1/2)$, there exists a randomized $(2+\epsilon)$-approximate dynamic matching algorithm with* $\text{poly}(1/\epsilon)$ *amortized update time whose approximation and update time guarantees hold in expectation against an* adaptive *adversary.*

B.2.4 Fractional matching sparsifiers

In this section we show that the subgraph H is a good *fractional* matching sparsifier, in the sense that it contains a high-valued fractional matching in terms of the input fractional matching $\vec{x}$. As the fractional matching polytope is integral (i.e., has integrality gap of one) in bipartite graphs, this will allow us to obtain *integral* matching sparsifiers for bipartite graphs.

Lemma 4.5. *(Algorithm 1 Yields Fractional Matching Sparsifiers) Let $\epsilon \in (0, 1/2)$ and $d \geq \frac{3\log(2/\epsilon)}{8\epsilon}$. If H is a subgraph of G output by Algorithm 1 when run on a fractional matching $\vec{x}$ with parameters ϵ and d as above, then H supports a fractional matching $\vec{y}$ of expected value at least*

$$\mathbb{E}\left[\sum_e y_e\right] \geq \sum_e x_e(1-4\epsilon).$$

Proof. Consider first the following intermediate assignment z of values to edges in H:

$$z_e = \begin{cases} (1-2\epsilon)\cdot x_e & x_e > \frac{1}{d} \\ (1-2\epsilon)\cdot \frac{1}{d} & x_e \leq \frac{1}{d}. \end{cases}$$

This can be written more succinctly as $z_e = x_e(1-2\epsilon)/\min\{1, x_e \cdot d\}$. Therefore, by our choice of $\vec{z}$ and by Lemma 3.2, we find that each edge e has expected z-value at least

$$\mathbb{E}[z_e] = \mathbb{E}[z_e \mid e \in H] \cdot \Pr[e \in H] \geq x_e(1-2\epsilon)/(1+\epsilon) \geq x_e(1-3\epsilon).$$

Our goal will be to define some fractional matching such that $\mathbb{E}[y_e] \geq \mathbb{E}[z_e]\cdot(1-O(\epsilon)) \geq x_e(1-O(\epsilon))$, which would imply our lemma by linearity of expectation.

Consider the following trivially-feasible fractional matching $\vec{y}$, given below

$$y_e = \begin{cases} 0 & x_e < 1/d \text{ and } \max_{v\in e}(\sum_{e'\ni v} z_{e'}) > 1 \\ z_e & \text{else.} \end{cases}$$

For edges e with $x_e \geq \frac{1}{d}$, we always have $y_e = z_e$, and so trivially $\mathbb{E}[y_e] = \mathbb{E}[z_e]$. Now, fix some edge $e' = (u, v)$ with $x_{e'} \leq \frac{1}{d}$ (which consequently has $z_e \leq \frac{1}{d} \leq \epsilon$.) By Corollary 3.5 and Lemma 3.2 we have that $\Pr[X_e \mid X_{e'}] \leq \Pr[X_e] \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon)$. Consequently, we have that

$$\begin{aligned} \mathbb{E}\left[\sum_{e\ni v} z_e \cdot X_e \,\middle|\, X_{e'}\right] &\leq z_{e'} + \sum_e x_e \cdot (1+\epsilon) \\ &\leq \max\left\{(1-2\epsilon)\cdot\sum_{e\ni v} x_e\cdot(1+\epsilon), \epsilon + \sum_{e\ni v: e\neq e'} x_e(1+\epsilon)\right\} \\ &\leq 1-\epsilon. \end{aligned}$$

We now upper bound the probability that this expression deviates so far above its expectation that $\vec{z}$ violates the fractional matching constraint of v.

By Lemma 3.6, we know that the variables $\{X_e \mid X_{e'}\}$ are NA. By closure of NA variables under scaling by positive constants, we have that the random variables $[z_e \cdot X_e \mid X_{e'}]$ are similarly NA. In order to effectively apply Bernstein's Inequality (Lemma 2.3) to these NA variables, we analyze their individual variances. By Lemma 3.2, we know that any edge e with $x_e > 1/d$ has $\Pr[X_e] = 1$, and so $\Pr[X_e \mid X_{e'}] = 1$. Consequently, the variance of $[z_e \cdot X_e \mid X_{e'}]$ is zero. On the other hand, for an edge e with $x_e \leq 1/d$ we have that $z_e \cdot [X_e \mid X_{e'}]$ is a binomial scaled by $z_e = \frac{1-2\epsilon}{d}$ with success probability at most $\Pr[X_e \mid X_{e'}] \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon) = x_e \cdot (1+\epsilon)$. Therefore, the variance of this variable is at most

$$\begin{aligned}\mathrm{Var}(z_e \cdot [X_e \mid X_{e'}]) &\leq \left(\frac{1-2\epsilon}{d}\right)^2 \cdot x_e \cdot d \cdot (1+\epsilon) \\ &\leq \frac{x_e}{d}.\end{aligned}$$

Therefore, all edges $e \ni v$ have that the conditional variance of their z-value is at most $\mathrm{Var}(z_e \cdot [X_e \mid X_{e'}]) \leq \frac{x_e}{d}$. Summing over all edges $e \ni v$ other than e', we have that

$$\mathrm{Var}\left(\sum_{e \ni v:\ e \neq e' \ x_e \leq \frac{1}{d}} [z_e \cdot X_e \mid X_{e'}]\right) \leq \sum_{e \ni v} \frac{x_e}{d} \leq \frac{1}{d}.$$

Now, recall from above that $\mathbb{E}\left[\sum_{e \ni v} z_e \cdot X_e \,\middle|\, X_{e'}\right]$. Therefore

$$\Pr\left[\sum_{e \ni v} z_e \geq 1 \,\middle|\, X_{e'}\right] \leq \Pr\left[\sum_{e \ni v} z_e \cdot [X_e \mid X_{e'}] \geq \sum_{e \ni v} z_e \cdot [X_e \mid X_{e'}] + \epsilon\right]$$

Denote by $F = \{e \ni v \mid e \neq e',\ x_e \leq \frac{1}{d}\}$ the set of edges which contribute to $\sum_{e \ni v,\, e \neq e'} z_e$ and for which $\mathrm{Var}(z_e \cdot [X_e \mid X_{e'}) \neq 0$. Applying Bernstein's Inequality (Lemma 2.3) to the NA variables with non-zero variance $\{z_e \cdot [X_e \mid X_{e'}] \mid e \in F\}$, each of which has absolute value at most $\frac{1-2\epsilon}{d} \leq \frac{1}{d}$ by definition, we find that

$$\begin{aligned}\Pr\left[\sum_{e \ni v} z_e \geq 1 \,\middle|\, X_{e'}\right] &\leq \Pr\left[\sum_{e \in F} z_e \cdot [X_e \mid X_{e'}] \geq \sum_{e \in F} z_e \cdot [X_e \mid X_{e'}] + \epsilon\right] \\ &\leq \exp\left(-\frac{\epsilon^2}{2 \cdot (1/d + \epsilon/3d)}\right) \\ &\leq \exp\left(-\frac{\epsilon^2}{8\epsilon/3d}\right) \\ &\leq \epsilon/2,\end{aligned}$$

where the last inequality follows from our choice of $d = \frac{3\log(2/\epsilon)}{8\epsilon}$.

Therefore, applying union bound to both endpoints of e', we find that conditioned on $e' = (u, v)$ being sampled, the probability that $y_e \neq z_e$ (and in particular $y_e = z_e$), which happens due to $\vec{z}$ not satisfying the fractional matching constraint of u or v, is

$$\Pr[y_e \neq z_e \mid X_e] \geq 1 - \epsilon,$$

from which we conclude that

$$\mathbb{E}[y_e] = \mathbb{E}[y_e = z_e \mid X_e] \cdot \mathbb{E}[z_e] \geq \mathbb{E}[z_e] \cdot (1-\epsilon) \geq x_e(1-3\epsilon)(1-\epsilon) \geq x_e(1-4\epsilon).$$

We conclude that the random subgraph H contains a fractional matching of expected value at least $1-4\epsilon$ times the fractional matching f in G. □

Remark. We note that we proved a stronger guarantee, namely that each edge e is assigned in expectation a y-value of at least $\mathbb{E}[y_e] \geq x_e(1-4\epsilon)$. It is also immediate that each edge is assigned a y-value of at most $\mathbb{E}[y_e] \leq \mathbb{E}[z_e] \leq x_e(1+\epsilon)$. This implies that Lemma 4.5 extends to rounding fractional *weighted* matchings.

It is well known that the integrality gap of the fractional matching polytope is one in bipartite graphs and $\frac{3}{2}$ in general graphs. Therefore, if H admits a fractional matching of value at least $\alpha \cdot \mu(G)$, then H contains an integral matching of value at least $\frac{1}{\alpha} \cdot \mu(G)$ or $\frac{2}{3\alpha} \cdot \mu(G)$ if G is bipartite or general, respectively. Consequently, Lemma 4.5 implies the following.

Lemma B.4. *For any $\epsilon \in (0, 1/2)$, Algorithm 1 run with an α-approximate dynamic fractional matching is $\frac{\alpha}{1-4\epsilon}$- and $\frac{3\alpha}{2(1-4\epsilon)}$-approximate on bipartite and general graphs, respectively.*

References

[1] Abboud, A. and Dahlgaard, S. 2016. Popular conjectures as a barrier for dynamic planar graph algorithms. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*. 477–486.

[2] Abboud, A. and Williams, V. V. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of the 55th Symposium on Foundations of Computer Science (FOCS)*. 434–443.

[3] Arar, M., Chechik, S., Cohen, S., Stein, C., and Wajc, D. 2018. Dynamic matching: Reducing integral algorithms to approximately-maximal fractional algorithms. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 7:1–7:16.

[4] Assadi, S., Bateni, M., and Mirrokni, V. 2019. Distributed weighted matching via randomized composable coresets. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 333–343.

[5] Assadi, S. and Bernstein, A. 2019. Towards a unified theory of sparsification for matching problems. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA)*.

[6] Assadi, S., Khanna, S., and Li, Y. 2016. The stochastic matching problem with (very) few queries. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. 43–60.

[7] Baswana, S., Gupta, M., and Sen, S. 2011. Fully dynamic maximal matching in $O(\log n)$ update time. In *Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS)*. 383–392.

[8] Baswana, S., Gupta, M., and Sen, S. 2015. Fully dynamic maximal matching in $O(\log n)$ update time. *SIAM Journal on Computing (SICOMP) 44,* 1, 88–113.

[9] Baswana, S., Khurana, S., and Sarkar, S. 2012. Fully dynamic randomized algorithms for graph spanners. *ACM Transactions on Algorithms (TALG) 8,* 4, 35.

[10] Behnezhad, S., Łącki, J., and Mirrokni, V. 2020. Fully dynamic matching: Beating 2-approximation in Δ^ϵ update time. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. To Appear.

[11] Bernstein, A. 2017. Deterministic partially dynamic single source shortest paths in weighted graphs. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*.

[12] Bernstein, A. and Chechik, S. 2016. Deterministic decremental single source shortest paths: beyond the $O(mn)$ bound. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*. 389–397.

[13] Bernstein, A. and Chechik, S. 2017. Deterministic partially dynamic single source shortest paths for sparse graphs. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 453–469.

[14] Bernstein, A., Forster, S., and Henzinger, M. 2019. A deamortization approach for dynamic spanner and dynamic maximal matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1899–1918.

[15] Bernstein, A. and Stein, C. 2015. Fully dynamic matching in bipartite graphs. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*. 167–179.

[16] Bernstein, A. and Stein, C. 2016. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 692–711.

[17] Bhattacharya, S., Chakrabarty, D., and Henzinger, M. 2017a. Deterministic fully dynamic approximate vertex cover and fractional matching in $O(1)$ amortized update time. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 86–98.

[18] Bhattacharya, S., Chakrabarty, D., Henzinger, M., and Nanongkai, D. 2018a. Dynamic algorithms for graph coloring. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1–20.

[19] Bhattacharya, S., Henzinger, M., and Italiano, G. 2018b. Dynamic algorithms via the primal-dual method. *Information and Computation 261*, 219–239.

[20] Bhattacharya, S., Henzinger, M., and Italiano, G. F. 2018c. Deterministic fully dynamic data structures for vertex cover and matching. *SIAM Journal on Computing (SICOMP) 47,* 3, 859–887.

[21] Bhattacharya, S., Henzinger, M., and Nanongkai, D. 2016. New deterministic approximation algorithms for fully dynamic matching. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*. 398–411.

[22] Bhattacharya, S., Henzinger, M., and Nanongkai, D. 2017b. Fully dynamic approximate maximum matching and minimum vertex cover in $O(\log^3 n)$ worst case update time. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 470–489.

[23] Bhattacharya, S. and Kulkarni, J. 2019. Deterministically maintaining a $(2+\epsilon)$-approximate minimum vertex cover in $O(1/\epsilon^2)$ amortized update time. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1872–1885.

[24] Charikar, M. and Solomon, S. 2018. Fully dynamic almost-maximal matching: Breaking the polynomial barrier for worst-case time bounds. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*.

[25] Chuzhoy, J. and Khanna, S. 2019. A new algorithm for decremental single-source shortest paths with applications to vertex-capacitated flow and cut problems. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. 389–400.

[26] Chuzhoy, J. and Saranurak, T. 2019. On dynamic shortest paths with adaptive adversary. Unpublished manuscript.

[27] Cohen, I. R., Peng, B., and Wajc, D. 2019. Tight bounds for online edge coloring. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*. To Appear.

[28] Cohen, I. R. and Wajc, D. 2018. Randomized online matching in regular graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 960–979.

[29] Dahlgaard, S. 2016. On the hardness of partially dynamic graph problems and connections to diameter. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*. 48:1–48:14.

[30] Duan, R., He, H., and Zhang, T. 2019. Dynamic edge coloring with improved approximation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1937–1945.

[31] Dubhashi, D. and Ranjan, D. 1996. Balls and bins: A study in negative dependence. *BRICS Report Series 3,* 25.

[32] Eppstein, D., Galil, Z., Italiano, G. F., and Nissenzweig, A. 1997. Sparsification – a technique for speeding up dynamic graph algorithms. *Journal of the ACM (JACM) 44,* 5, 669–696.

[33] Fleischer, L. K. 2000. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics 13,* 4, 505–520.

[34] Forster, S. and Goranci, G. 2019. Dynamic low-stretch trees via dynamic low-diameter decompositions. 377–388.

[35] Gamlath, B., Kapralov, M., Maggiori, A., Svensson, O., and Wajc, D. 2019. Online matching with general arrivals. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*. To Appear.

[36] Garg, N. and Koenemann, J. 2007. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing (SICOMP) 37,* 2, 630–652.

[37] Ghaffari, M., Gouleakis, T., Konrad, C., Mitrović, S., and Rubinfeld, R. 2018. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 129–138.

[38] Ghaffari, M. and Wajc, D. 2019. Simplified and space-optimal semi-streaming $(2+\epsilon)$-approximate matching. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA)*.

[39] Goel, A., Kapralov, M., and Khanna, S. 2012. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 468–485.

[40] Gupta, A., Krishnaswamy, R., Kumar, A., and Panigrahi, D. 2017. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*. 537–550.

[41] Gupta, M. and Peng, R. 2013. Fully dynamic $(1+\epsilon)$-approximate matchings. In *Proceedings of the 54th Symposium on Foundations of Computer Science (FOCS)*. 548–557.

[42] Gutenberg, M. P. and Wulff-Nilsen, C. 2020a. Decremental sssp in weighted digraphs: Faster and against an adaptive adversary. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. To Appear.

[43] Gutenberg, M. P. and Wulff-Nilsen, C. 2020b. Deterministic algorithms for decremental approximate shortest paths: Faster and simpler. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. To Appear.

[44] Henzinger, M., Krinninger, S., and Nanongkai, D. 2016. Dynamic approximate all-pairs shortest paths: Breaking the $O(mn)$ barrier and derandomization. *SIAM Journal on Computing (SICOMP) 45,* 3, 947–1006.

[45] Henzinger, M., Krinninger, S., Nanongkai, D., and Saranurak, T. 2015. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 21–30.

[46] Henzinger, M. R. and King, V. 1999. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM (JACM) 46,* 4, 502–516.

[47] Holm, J., De Lichtenberg, K., and Thorup, M. 2001. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM) 48,* 4, 723–760.

[48] Hopcroft, J. E. and Karp, R. M. 1973. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing 2,* 4, 225–231.

[49] Ivkovic, Z. and Lloyd, E. L. 1993. Fully dynamic maintenance of vertex cover. In *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science*. 99–111.

[50] Joag-Dev, K. and Proschan, F. 1983. Negative association of random variables with applications. *The Annals of Statistics*, 286–295.

[51] Kapron, B. M., King, V., and Mountjoy, B. 2013. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1131–1142.

[52] Khursheed, A. and Lai Saxena, K. 1981. Positive dependence in multivariate distributions. *Communications in Statistics - Theory and Methods 10,* 12, 1183–1196.

[53] Kopelowitz, T., Pettie, S., and Porat, E. 2016. Higher lower bounds from the 3sum conjecture. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1272–1287.

[54] Lee, E. and Singla, S. 2017. Maximum matching in the online batch-arrival model. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 355–367.

[55] Lotker, Z., Patt-Shamir, B., and Pettie, S. 2015. Improved distributed approximate matching. *Journal of the ACM (JACM) 62,* 5, 38.

[56] Micali, S. and Vazirani, V. V. 1980. An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs. In *Proceedings of the 21st Symposium on Foundations of Computer Science (FOCS)*. 17–27.

[57] Mądry, A. 2010. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*. 121–130.

[58] Nanongkai, D. and Saranurak, T. 2017. Dynamic spanning forest with worst-case update time: adaptive, las vegas, and $O(n^{1/2-\epsilon})$-time. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*. 1122–1129.

[59] Nanongkai, D., Saranurak, T., and Wulff-Nilsen, C. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *Proceedings of the 58.*

[60] Neiman, O. and Solomon, S. 2016. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Transactions on Algorithms (TALG) 12,* 1, 7.

[61] Onak, K. and Rubinfeld, R. 2010. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*. 457–464.

[62] Paz, A. and Schwartzman, G. 2018. A $(2+\epsilon)$-approximation for maximum weight matching in the semi-streaming model. *ACM Transactions on Algorithms (TALG) 15,* 2, 18.

[63] Peleg, D. and Solomon, S. 2016. Dynamic $(1+\epsilon)$-approximate matchings: a density-sensitive approach. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 712–729.

[64] Sankowski, P. 2007. Faster dynamic matchings and vertex connectivity. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 118–126.

[65] Solomon, N. and Solomon, S. 2018. Reoptimization via gradual transformations. *arXiv preprint arXiv:1803.05825*.

[66] Solomon, S. 2016. Fully dynamic maximal matching in constant update time. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*. 325–334.

[67] Solomon, S. 2018. Local algorithms for bounded degree sparsifiers in sparse graphs. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS)*. 52:1–52:19.

[68] Stubbs, D. and Vassilevska Williams, V. 2017. Metatheorems for dynamic weighted matching. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*.

[69] Thorup, M. 2000. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*. 343–350.

[70] Vizing, V. G. 1964. On an estimate of the chromatic class of a p-graph. *Diskret analiz 3*, 25–30.