On b-Matching and Fully-Dynamic Maximum k-Edge Coloring

Antoine El-Hayek
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Kathrin Hanauer
Faculty of Computer Science, University of Vienna, Austria

Monika Henzinger
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

Given a graph G that undergoes a sequence of edge insertions and deletions, we study the Maximum k-Edge Coloring problem (MkEC): Having access to k different colors, color as many edges of G as possible such that no two adjacent edges share the same color. While this problem is different from simply maintaining a b-matching with $b = k$, the two problems are related. However, maximum b-matching can be solved efficiently in the static setting, whereas MkEC is NP-hard and even APX-hard for $k \geq 2$.

We present new results on both problems: For b-matching, we show a new integrality gap result and we adapt Wajc's matching sparsification scheme [50] for the case where b is a constant.

Using these as basis, we give three new algorithms for the dynamic MkEC problem: Our `MatchO` algorithm builds on the dynamic $(2+\epsilon)$-approximation algorithm of Bhattacharya, Gupta, and Mohan [9] for b-matching and achieves a $(2+\epsilon)\frac{k+1}{k}$-approximation in $O(poly(\log n, \epsilon^{-1}))$ update time against an *oblivious* adversary. Our `MatchA` algorithm builds on the dynamic $(7+\epsilon)$-approximation algorithm by Bhattacharya, Henzinger, and Italiano [10] for fractional b-matching and achieves a $(7+\epsilon)\frac{3k+3}{3k-1}$-approximation in $O(poly(\log n, \epsilon^{-1}))$ update time against an *adaptive* adversary. Moreover, our reductions use the dynamic b-matching algorithm as a black box, so any future improvement in the approximation ratio for dynamic b-matching will automatically translate into a better approximation ratio for our algorithms. Finally, we present a greedy algorithm with $O(\Delta + k)$ update time, which guarantees a 2.16 approximation factor.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases dynamic algorithm, graph algorithm, matching, b-matching, edge coloring

Digital Object Identifier 10.4230/LIPIcs.SAND.2025.4

Related Version *Full Version*: `https://arxiv.org/abs/2310.01149`

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (MoDynStruct, No. 101019564) and the Austrian Science Fund (FWF) grant DOI 10.55776/Z422, grant DOI 10.55776/I5982, and grant DOI 10.55776/P33775 with additional funding from the netidee SCIENCE Stiftung, 2020–2024. This work was further supported by the Federal Ministry of Education and Research (BMBF) project, 6G-RIC: 6G Research and Innovation Cluster, grant 16KISK020K.

1 Introduction

In large data centers, new technologies such as optical switches allow for quick adaptations of the network topology that are optimally tailored to current traffic demands. Indeed, network performance has been identified to be a major bottleneck for the scalability of computations in the cloud [35, 42]. Each optical switch can establish a set of direct connections between pairs of data center racks, such that each rack has a high-bandwidth connection to at most

one other rack via the switch. The regular network infrastructure remains in place and can be used for all other traffic. With their fast reconfiguration time and the possibility to use multiple appliances in parallel, optical switches have become a promising means to mitigate the performance bottleneck. A major algorithmic challenge is how to configure them optimally.

Consider an undirected graph G where each node represents a data center rack and the edges indicate pairs of racks with high communication demand. We refer to G as *demand graph*. Each set of connections realized by a single optical switch is a *matching* in G, i.e., a set of pairwise node-disjoint edges, and it is desirable that this matching is large, such that a substantial amount of the traffic is routed via the high-bandwidth direct connections. For $k \in \mathbb{N}$ optical switches, the problem hence amounts to finding a collection of k pairwise edge-disjoint matchings M_i, $1 \leq i \leq k$, with maximum total cardinality $\sum_{i=1}^{k} |M_i|$. We call this the *maximum k-disjoint matching problem*. Identifying each matching with a unique color, it can equivalently be rephrased as a *maximum k-edge coloring* problem (MkEC), where the goal is to maximize the number of colored edges. As communication demands naturally change frequently over time, we study the problem in the dynamic setting. In some situations, e.g. when remote resources such as GPUs are accessed via the network, the demand graph may be bipartite, which is why we also consider bipartite graphs.

There exists a related, but different problem, the *maximum* $\mathbf{b}$*-matching problem*: for a graph $G = (V, E)$ and $\mathbf{b} \in \mathbb{N}^V$, find a maximum-cardinality subset of edges $H \subseteq E$ such that each vertex $v \in V$ is incident to at most b_v edges in H. Note that one can set b_v to a constant k for all $v \in V$, but this does not yield the MkEC problem: there is no requirement that H can be edge-colored completely with k colors. Consider, e.g., a graph $G = (V, E)$ that is a length-3 cycle. A solution H to the $\mathbf{b}$-matching problem with $b_v = 2\ \forall v \in V$ contains all three edges of G, while a solution to the maximum 2-edge coloring problem can only color two edges of G. The third edge has to remain uncolored. This example shows that an optimal solution to the MkEC problem can be 1.5 times smaller than one to the $\mathbf{b}$-matching problem with $b_v = k$ for all $v \in V$. Furthermore, a solution to the latter does not always give a solution to the former. In general, deciding whether a graph with maximum degree Δ can be edge-colored with Δ colors or whether $\Delta + 1$ are required is a well-known NP-hard problem [32] (Vizing [49] showed that every simple graph needs either Δ or $\Delta(G) + 1$ colors to color all edges). On bipartite graphs, however, Δ colors always suffice.

Let $G = (V, E)$ be a graph with $n = |V|$ and $m = |E|$. For the fully dynamic $\mathbf{b}$-matching problem, Bhattacharya, Henzinger, and Italiano [11] gave a constant approximation algorithm with $O(\log^3 n)$ update time which works against an adaptive adversary. If the adversary is oblivious, there also is a $(2 + \epsilon)$-approximation with $O(1/\epsilon^4)$ update time by Bhattacharya, Gupta, and Mohan [9].

The only prior work for the fully dynamic MkEC problem is an experimental analysis of various dynamic algorithms by Hanauer, Henzinger, Ost and Schmid [30], who, among others, also describe a near-linear-time, fully-dynamic 3-approximation algorithm for the weighted case. Dropping the weights, we show how to significantly improve the update time to $O(\text{poly}(\log n, \epsilon^{-1}))$ while achieving an approximation ratio to almost $(2 + 2/k)$ against an *oblivious* adversary and $(7 + \epsilon)(1 + 4/(3k - 1))$ (for any $\epsilon > 0$) against an *adaptive* adversary. The problem is known to be NP-hard and even APX-hard for $k \geq 2$ [23].

Our contributions. We show that the integrality gap for the $\mathbf{b}$-matching problem is $\frac{3\beta}{3\beta-1}$ where $\beta = \min_{v \in V} b_v$. In the case where $\mathbf{b}$ is the constant vector with $b_v = k\ \forall v \in V$, we adapt the elegant rounding technique given by Wajc [50], who showed how to round a

Table 1 Previous and New Algorithms for Dynamic Maximum k-Edge-Coloring. The "Det.?" column states whether or not the algorithm is deterministic.

Algorithm	Update Time	Approximation Ratio	Det.?	Adversary	Theorem	Sect.
algorithms in [30]	$O(n)$ or more	3	yes	–		
	$O(1)$	unknown	no	–		
`Greedy`	$O(k+\Delta)$	$1+2\frac{\sqrt{3}}{3} \approx 2.155$	yes	–	14	7.1
`MatchO`	$O(\text{poly}(\log n, \epsilon^{-1}))$	$(2+\epsilon)(1+1/k)$	no	oblivious	16	7.2.1
⌞ `bipartite`	$O(\text{poly}(\log n, \epsilon^{-1}))$	$(2+\epsilon)$	no	oblivious	17	7.2.1
`MatchA`	$O(\text{poly}(\log n, \epsilon^{-1}))$	$(7+\epsilon)\frac{3k+3}{3k-1}$	no	adaptive	18	7.2.2
⌞ `bipartite`	$O(\text{poly}(\log n, \epsilon^{-1}))$	$(7+\epsilon)$	no	adaptive	19	7.2.2

fractional matching to an integral matching in a dynamic graph, to round a given fractional k-matching[1] to an integral k-matching whose size is linear (up to polylogarithmic factors) in the size of the optimal solution.

For the dynamic MkEC problem, we describe and analyze three dynamic approximation algorithms, with different trade-offs between their update time and approximation ratio. We also give two versions specific to bipartite graphs, where the approximation ratio is improved. See Table 1 for a summary. Our algorithms `MatchO` and `MatchA` represent a general black-box reduction from dynamic maximum k-edge coloring to dynamic **b**-matching. Thus, any improvement in the approximation ratio of maximum **b**-matching immediately leads to an improvement of the approximation ratio of our algorithms.

Most proofs are only available in the appendix due to space restrictions.

2 Related Work

We only give a short overview here, see Appendix B for an extended version.

Edge Coloring. Given a graph G of maximum degree Δ, its *chromatic index* $\chi'(G)$ is the smallest value q such that all edges of G can be colored with q colors. Generally, $\Delta \leq \chi'(G) \leq \Delta + 1$ [49], whereas $\chi'(G) = \Delta$ [36] for bipartite graphs. Deciding whether $\chi'(G) = \Delta$ or $\chi'(G) = \Delta + 1$ is NP-hard already for $\Delta = 3$ [32], even if G is regular [38].

For an n-vertex m-edge graph G, Gabow [27] gave an $O(m\Delta \log n)$-time coloring algorithm that uses at most $\Delta + 1$ colors. Misra and Gries [41] gave an algorithm that needs $O(mn)$ running time, and which was improved to $O(m\sqrt{n})$ running time by Sinnamon [46]. A recent series of works further reduced the running time to $\tilde{O}(n^2)$ (Assadi [2]), $\tilde{O}(mn^{1/3})$(Bhattacharya et al. [6]), and $\tilde{O}(mn^{1/4})$ (Bhattacharya et al. [8]). Duan et al. [20] further reduced the time to $O(m \cdot \text{poly}(\log n, \epsilon^{-1}))$ as long as $\Delta = \Omega(\log^2 n \cdot \epsilon^{-2})$, but use up to $(1+\epsilon)\Delta$ colors. For bipartite graphs, Cole et al. [19] gave an optimal algorithm with $O(m \log \Delta)$ running time. Cohen et al. [18] recently studied the problem in the online setting and proved various competitive ratio results.

For dynamic graphs, Bhattacharya et al. [7] show how to maintain a $(2\Delta-1)$-edge coloring in $O(\log n)$ worst-case update time, and that a $(2+\epsilon)\Delta$-edge coloring can be maintained with $O(1/\epsilon)$ expected update time. If $\Delta = \Omega(\log^2 n \cdot \epsilon^{-2})$, Duan et al. [20] maintain an edge-coloring using $(1+\epsilon)\Delta$ colors in amortized $O(\log^8 n \cdot \epsilon^{-4})$ update time.

[1] Whenever **b** is the all-k vector for some constant k, we will refer to the problem as k-matching instead of **b**-matching.

Maximum k-Edge Coloring. The problem was first studied by Favrholdt and Nielsen [22] in the online setting, who show that every algorithm that never chooses to not color ("reject") a colorable edge has a competitive ratio between 1/0.4641 and 2, and that any online algorithm is at most $\frac{4}{7}$-competitive. Feige et al. [23] showed that for every $k \geq 2$, there exists an $\epsilon_k > 0$ such that it is NP-hard to approximate the problem within a ratio better than $(1 + \epsilon_k)$. They also describe a static $(1 - (1 - 1/k)^k)^{-1}$-approximation algorithm for general k. The currently best approximation ratios are 1/0.842 for $k = 2$ and $\frac{15}{13}$ for $k = 3$ [17, 34].

The maximum k-edge coloring problem was first studied in the edge-weighted setting by Hanauer et al. [31]. Here, instead of finding a maximum-cardinality subset of the edges, the total weight of the colored edges is to be maximized. In a follow-up work, Hanauer et al. [30] design a collection of different dynamic and batch-dynamic algorithms for weighted k-edge coloring. Their focus is again more on the practical side. Ferdous et al. [24] recently studied the problem in the streaming setting.

Matching. The matching problem has been subject to extensive research both in the unweighted and weighted case [40, 28, 33, 26, 21]. Various dynamic algorithms with different trade-offs between update time and approximation ratio also exist for general [44, 3, 47, 43, 10, 12, 29] and bipartite graphs [14, 16]. Wajc [50] gives a metatheorem for rounding a dynamic fractional matching against an adaptive adversary and a $(2 + \epsilon)$-approximate algorithm with $O(1)$ update time or $O(\text{poly}(\log n, \epsilon^{-1}))$ worst-case update time.

b-Matching. Gabow [25] gives a $O(\sqrt{\|b\|_1}m)$-time algorithm to compute a $\mathbf{b}$-matching in the unweighted, static setting, and an $O(\|b\|_1 \cdot \min(m \log n, n^2))$-time algorithm for weighted graphs. Ahn and Guha's algorithm [1] computes a $(1 + \epsilon)$-approximation for $\mathbf{b}$-matching and runs in $O(m \,\text{poly}(\log n, \epsilon^{-1}))$ time. Bienkowski et al. [13] give an online $O(\log b)$-approximate solution, which is asympotically optimal in the online setting.

For dynamic graphs, Bhattacharya et al. [11] give a deterministic algorithm that maintains an $O(1)$-approximate fractional k-matching with $O(\log^3 n)$ amortized update time. This is improved by Bhattacharya et al. [9], who show how to maintain an integral $(2+\epsilon)$-approximate $\mathbf{b}$-matching in expected amortized $O(1/\epsilon^4)$ update time against an oblivious adversary.

3 Technical Overview

Our starting point is the following observation: The two problems $\mathbf{b}$-matching and k-edge coloring seem very similar if $\mathbf{b}$ is the vector having $b_v = k$ for every vertex $v \in V$. Indeed, the colored edges in a k-edge coloring form a k-matching. Vice versa, a maximum k-matching always contains a $\frac{k+1}{k}$-approximate k-edge coloring, as one can always color the edges with $k + 1$ colors and discard the least-used color. This close connection is one major ingredient for the two main algorithms we present, the dynamic `MatchO` and `MatchA` algorithms: Find a good k-matching in the graph first, then color it using as few colors as possible, and finally discard all edges of the least-used colors. The goal is to perform updates in $O(\text{poly}(\log n, \epsilon^{-1}))$ time.

Our `MatchO` algorithm, which works against oblivious adversaries, is a combination of known algorithms: We use the aforementioned $(2 + \epsilon)$-approximation by Bhattarcharya et al. [9] to find a k-matching. Duan et al.'s algorithm [20] colors the edges with $(1 + \epsilon)\Delta$ colors. Discarding the least-used colors yields a $(2 + \epsilon)\frac{k+1}{k}$-approximation for MkEC.

For our `MatchA` algorithm, which is designed to work against an adaptive adversary, we could have used the so-far best algorithm for $\mathbf{b}$-matching by Bhattacharya et al. [11], which however only guarantees an $O(1)$-approximation. The algorithm is based on a $(7 + \epsilon)$-approximate *fractional* $\mathbf{b}$-matching algorithm, whose solution is rounded. We present an

alternative rounding approach and thus obtain an improved integral k-matching algorithm that guarantees a $(7+\epsilon)\frac{3k}{3k-1}$-approximation, which is at most $(8.4+\epsilon)$ (for $k=2$). Following the same scheme as for `MatchO`, we thus obtain a $(7+\epsilon)\frac{3k+3}{3k-1}$-approximation for MkEC.

Our new rounding technique works as follows: Given a fractional k-matching, we partition the edges of our graph by powers of $(1+\epsilon)$ according to their fractional value and maintain a 3Δ-coloring of each subgraph (which is not related to the coloring we will output). We then construct a sparsified graph by choosing a subset of colors uniformly at random and keeping only edges of these colors. Intuitively, the sparsified graph consists of those edges with high fractional values. We show that if an optimal k-edge coloring of the input graph colors s edges, then the sparsified graph contains at most $O(s \cdot \text{poly}(\log n, \epsilon^{-1}))$ edges. Running Ahn and Guha's algorithm [1] on the sparsified graph takes $O(s \cdot \text{poly}(\log n, \epsilon^{-1}))$ time and computes an integral k-matching. We can thus afford to rerun Ahn and Guha's algorithm every $\Omega(\epsilon s)$ updates and still have polylogarithmic update time. Recomputing the rounding this often also ensures the approximation ratio is still good enough. We further prove that the integrality gap of the k-matching problem is $\frac{3k}{3k-1}$ and hence small. This ensures that the sparsified graph, which we show contains a large fractional k-matching, also contains a large integral k-matching, and thus that the graph output by Ahn and Guha's algorithm is a good rounding of the original fractional matching.

We also show that the integrality gap of **b**-matching is small. The argument starts by noting that the integrality gap of **b**-matching on *bipartite* graphs is 1, which is a known result. We then prove that every **b**-matching polytope is half-integral. To do so, we build a bipartite graph that encodes the original graph. Every extremal point of the polytope of the original graph can be encoded as half the sum of two extremal points of the bipartite polytope, and thus is half-integral. Once we have a half-integral solution, we proceed to show that we can round it to an integer solution without losing much of the fractional solution. Essentially, we look at the dual variables of the **b**-matching, that is, for each vertex, we look at its weighted degree. We then show that while rounding, out of every carefully chosen three vertices, only one can see its degree drop, under the crucial condition that its weighted degree was already equal to b_v. This allows us to prove that the integrality gap is $\frac{3\beta}{3\beta-1}$, with $\beta = \min_{v \in V} b_v$.

On bipartite graphs, many aspects of the problem become easier: the integrality gap of the b-matching algorithm, as well as the existence of efficient edge-coloring algorithms, like the one by Cole, Ost, and Schirra [19], which colors all of the edges with Δ colors in $O(m \operatorname{polylog} n)$ time. This improves the approximation ratio of `MatchO` and `MatchA` to $(2+\epsilon)$ and $7(1+\epsilon)$ respectively.

4 Preliminaries

Unless denoted otherwise, we consider an undirected, unweighted graph $G = (V, E)$ with $n := |V|$ and $m := |E|$. G is dynamic, that is it undergoes an a priori unknown series of *updates* in the form of *edge insertions* and *edge deletions*. The *update time* of an algorithm is the time it needs to process an update before accepting the next one. These updates are controlled by an adversary, that can be either *adaptive*, that is, can see the internal state of our data structure and choose the next update accordingly, or *oblivious*, that is, has to decide all of the updates before we start running our algorithm. G, n, and m always refer to the current graph and its number of vertices and edges, respectively, i.e., including all updates that occurred beforehand. Edges are treated as subsets of vertices of size 2. We use $\Delta(G)$ to denote the maximum vertex degree in G. If it is clear from the context, we may omit G and just write Δ.

▶ **Definition 1** (Edge Coloring). *Let $G = (V, E)$ be a graph, $k \in \mathbb{N}_+$, and $f : E \mapsto [k] \cup \{\perp\}$ an edge coloring of the edges of G. We say that:*

1. *f is a* proper *coloring of E if for any adjacent edges e, e', we have that either $f(e) \neq f(e')$, or $f(e) = f(e') = \perp$.*
2. *f is a* total *coloring of E if $f^{-1}(\perp) = \varnothing$. We say it is* partial *to emphasize it is not necessarily total, that is, $f^{-1}(\perp)$ may or may not be equal to $\varnothing$.*
3. *A color $c \in [k]$ is* free *on node v (according to f) if no edge incident to v has color c, that is, for every $e \ni v$, $f(e) \neq c$.*
4. *An edge e is* colored *if $f(e) \in [k]$ and* uncolored *otherwise.*

Given $G = (V, E)$ and $k \in \mathbb{N}$, the *maximum k-edge coloring problem* consists in finding a proper coloring $f : E \mapsto [k] \cup \{\perp\}$ of E such that the set of colored edges $|f^{-1}([k])|$ is maximized. In the rest of this paper, unless specified otherwise, every (k-)coloring is proper and we use f to denote an arbitrary (proper) (k-)coloring, and f^* to denote an optimal (k-)coloring.

A *matching* $M \subseteq E$ is a subset of edges such that for every distinct pair $e, e' \in M$, $e \cap e' = \varnothing$. Note that given a k-edge coloring f, $f^{-1}(c)$ is a matching for each $c \in [k]$. A set of edges $M \subseteq E$ is a *k-matching* if for all $v \in V$, $|\{e \in M : e \ni v\}| \leq k$. Given an n-dimensional vector $\mathbf{b}^V$, we say that a set of edges $M \subseteq E$ is a *$\mathbf{b}$-matching* if for every $v \in V$, $|\{e \in M : e \ni v\}| \leq b_v$. Thus, a k-matching is a $\mathbf{b}$-matching where $\mathbf{b} = k^V$, and a matching is a 1-matching ($k = 1$).

▶ **Theorem 2** (Coloring an Approximate k-Matching, extension of [23]). *Let G be a graph and H a subgraph such that H is a solution of an α-approximation algorithm for k-matching on G, for some $\alpha \geq 1$. Let f be a total coloring of H using $k + \ell$ colors, with $\ell \in \mathbb{N}$. Then, discarding the ℓ least used colors from f yields an $(\alpha \cdot \frac{k+\ell}{k})$-approximate k-edge coloring of G. In particular, if $k = \Delta(H)$ and $\ell = \epsilon\Delta(H)$, the approximation ratio is $\alpha \cdot (1 + \epsilon)$.*

▶ **Corollary 3.** *Given a graph G, let s^* be the size of an optimum k-matching in G and let p^* be the size of an optimum k-edge coloring. Then $p^* \leq s^* \leq \frac{k+1}{k} p^*$.*

5 The b-Matching Polytope

In this section, we will show the following theorem.

▶ **Theorem 4** (Integrality Gap Theorem). *The integrality gap of the $\mathbf{b}$-matching polytope is $\frac{3\beta}{3\beta-1}$, where $\beta = \min_{v \in V} b_v$. The integrality gap of the bipartite $\mathbf{b}$-matching polytope is 1.*

While the result on bipartite graphs was known [45], the general result does not exist in the literature to the best of our knowledge. We first show that the (fractional) $\mathbf{b}$-matching polytope is half-integral[2], then round an optimal solution for the fractional $\mathbf{b}$-matching problem to get an integral solution with a good approximation ratio.

In this section, a *trail* is a walk with no repeated edges (but possibly repeated nodes), a *circuit* is a closed trail, and an *Eulerian circuit* is a circuit that visits all edges.

The $\mathbf{b}$-matching polytope is defined as follows:

▶ **Definition 5** (fractional $\mathbf{b}$-matching polytope). *Let $G = (V, E)$ be an undirected graph. The fractional $\mathbf{b}$-matching polytope $\mathcal{P}(G)$ is:*

$$\mathcal{P}(G) = \{\mathbf{x} \in [0, 1]^E : \forall v \in V \sum_{e \ni v} x_e \leq b_v\}$$

[2] Even though the $\mathbf{b}$-matching polytope is well-researched, we could not find this result in the literature.

It is well-known that if G is bipartite, then $\mathcal{P}(G)$ is integral, i.e., every vertex of the polytope has integer entries:

▶ **Lemma 6** ([45]). *Let $G = (V, E)$ be a bipartite graph. Then $\mathcal{P}(G)$ is integral.*

We will now show that the fractional **b**-matching polytope over general graphs is half-integral, that is, the entries of all its vertices are in $\{0, \frac{1}{2}, 1\}$. To do so, given a graph G, we build a graph G' that is bipartite and show a relationship between vertices of $\mathcal{P}(G)$ and $\mathcal{P}(G')$.

▶ **Lemma 7.** *Let $G = (V, E)$ be a graph. Then $\mathcal{P}(G)$ is half-integral.*

Next, we introduce a technique for rounding an optimal solution in $\mathcal{P}(G)$ to an integer solution that has similar total cost. This requires the following helper lemma:

▶ **Lemma 8.** *Let G be a graph that contains no even circuit. Then G has no even cycles, and no two odd cycles in G share a node.*

We also recall the Euler partition of a graph:

▶ **Definition 9** (Euler Partition). *Let $G = (V, E)$ be a graph. A* Euler Partition *of G is a partition of its edges into trails and circuits such that every node of odd degree is the endpoint of exactly one trail, and every node of even degree is the endpoint of no trail.*

It is easy to see that such a partition exists for every graph, as one can compute one by removing maximal trails from G until no edge remains. We now have all the tools necessary to find the integrality gap of the **b**-matching polytope, to which we give the proof sketch here, and the full proof in the appendix.

▶ **Theorem 4** (Integrality Gap Theorem). *The integrality gap of the* **b***-matching polytope is $\frac{3\beta}{3\beta-1}$, where $\beta = \min_{v \in V} b_v$. The integrality gap of the bipartite* **b***-matching polytope is 1.*

Proof Sketch. Since the fractional polytope is half-integral, we find an optimal fractional solution $\mathbf{x}$ where all entries are in $\{0, \frac{1}{2}, 1\}$. We then consider a Euler partition of the union of edges whose value is $\frac{1}{2}$. First, consider all trails that start and end at different vertices, and alternatively round up and down the values of the edges. This does not affect the optimality of the solution: for each inner vertex of the trail, the same number of incident edges is rounded up and down. For each end vertex x, $\sum_{e \ni v} x_e \leq b_v - \frac{1}{2}$, so rounding up does not violate the condition at x.

Note that all trails must have even length, otherwise $\mathbf{x}$ would not be optimal. We again remove all integral edges from the subgraph, and end up with a Eulerian graph. We then apply the same strategy to every even circuit in the subgraph, until there are no more even circuits. By Lemma 8, the remaining graph is composed of disjoint odd cycles. We can thus again apply the strategy of alternatively rounding up and down the values of the edges, except that this time, there might be two consecutive edges we might need to round down, which may decrease the solution value. The worst-case scenario occurs for length-3 cycles, as the rounded solution is only 2/3 as good as the fractional solution. But this only happens when no node on the cycle can "afford" both incident edges to be rounded up, that is, all dual inequalities are tight to a 1/2 additive factor. This gives the result. ◀

6 The Sparsification Scheme

Given a dynamic fractional k-matching algorithm $\mathcal{A}_m$, this section provides a sparsification scheme for fractional k-matching that enables us to give an algorithm $\mathcal{A}$ that maintains an approximate integral k-matching:

▶ **Theorem 10** (Sparsify and Round). *Let G be a dynamic graph, $\epsilon > 0$, $\alpha = O(1)$, and $\mathcal{A}_m$ a dynamic algorithm that maintains an α-approximate fractional k-matching of G in $O(T_m)$ update time. Then there exists an algorithm $\mathcal{A}$ that maintains an $\alpha(1+\epsilon)\frac{3k}{3k-1}$-approximate integral k-matching in $O(T_m + \text{poly}(\log n, \epsilon^{-1}))$ amortized update time against an adaptive adversary. If G is bipartite, the approximation ratio reduces to $\alpha(1+\epsilon)$.*

This result will be particularly useful for the `MatchA` algorithm in Section 7.2.2. To prove the theorem, we consider a dynamic graph G and assume we have access to a dynamic algorithm $\mathcal{A}_m$ that has update time $O(T_m)$ and maintains a fractional k-matching $\mathbf{x}$ with approximation factor α. The goal is to maintain a sparse subgraph H of G that contains an integral k-matching whose size is within an $\alpha(1+\epsilon)\frac{3k}{3k-1}$ factor of the size of an optimal k-matching in G.

To this end, we separate our strategy into two schemes: the update scheme is repeated at every update, while the request scheme is only repeated when we need the sparse graph H. Our technique is an adaptation and generalization of an efficient sparsification by Wajc [50] for rounding fractional (1-)matchings to integral ones.

Update Scheme

Let $\mathbf{x}$ be the fractional k-matching maintained by $\mathcal{A}_m$ of value $c(\mathbf{x}) = \sum_{e\in E} x_e$. Let $\ell := 2\log_{(1+\epsilon)}(n\epsilon^{-1})$. We maintain ℓ subgraphs $G_1, \ldots, G_\ell$ of G, where $G_i = (V, E_i)$ and

$$E_i = \left\{e \in E \colon x_e \in \left((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}\right]\right\}$$

Let $E^+ = \bigcup_{i\in[\ell]} E_i$. Note that E^+ is a subset of E and does not contain edges e such that $x_e \le (1+\epsilon)^{-\ell} = \frac{\epsilon^2}{n^2}$. Hence, $\sum_{e\in E\setminus E^+} x_e \le m\frac{\epsilon^2}{n^2} \le \epsilon^2$ and we obtain

$$\sum_{e\in E^+} x_e \ge c(\mathbf{x}) - \epsilon^2 \ge c(\mathbf{x})(1-\epsilon). \tag{1}$$

In each G_i, we maintain a proper, total $(3\lceil k(1+\epsilon)^i\rceil)$-edge coloring. Since in G_i, every edge satisfies $x_e > (1+\epsilon)^{-i}$, and $\mathbf{x}$ is a k-matching, we know that the maximum degree of G_i can be no higher than $k(1+\epsilon)^i$. Hence, every edge modified in G_i can be colored in expected constant time[3]. As each update of G can modify only $O(T_m)$ edges, there are at most $O(T_m)$ modifications in total to all subgraphs G_i. The recoloring caused by each modification can be handled in expected constant time, which implies $O(T_m)$ expected time per update to G.

Request Scheme

We fix a parameter $d := \max\{\frac{1}{k\epsilon}, \frac{4\log(2/\epsilon)}{k\epsilon^2}\}$. Whenever we need access to the sparse graph H, we obtain a set of edges H_i from each G_i by choosing uniformly at random without replacement up to $\lceil kd(1+\epsilon)\rceil$ colors, and setting H_i to be the set of edges colored with those colors. Then, $H := (V, \bigcup_{i=1}^{\ell} H_i)$. Obtaining H takes $O(|H|)$ time.

We analyze the size of H with respect to $\left|f^{*-1}([k])\right|$, the size of an optimal k-edge coloring in G. Let $p^* := \left|f^{*-1}([k])\right|$ and s^* be the size of a maximum k-matching on G. Each collection of k colors in H_i creates a k-edge coloring in G, and has thus size at most p^*. We

[3] By maintaining a hash table at each vertex of the free colors, since we have more than $3\Delta(G_i)$ colors available. By picking a color at random, we have a probability higher than $\frac{1}{3}$ for this color to be free at both end nodes, and thus we only need three random picks in expectation to find a free color.

have $O(d(1+\epsilon))$ such collections in H_i, and hence $|H_i| = O(d(1+\epsilon)p^*) = O(d(1+\epsilon)s^*)$. Therefore,

$$|H| \leq \sum_{i=1}^{\ell} |H_i| \leq O\left(\ell d(1+\epsilon)s^*\right) = O\left(s^* \cdot \log n \operatorname{poly}(\epsilon^{-1})\right).$$

Together with Equation (1), Lemmas 12 and 13 below show that in expectation, H contains a fractional k-matching of total value at least $c(\mathbf{x})(1-\epsilon)(1-6\epsilon) \geq c(\mathbf{x})(1-7\epsilon)$. By our Integrality Gap Theorem (Theorem 4), H then contains an integral k-matching of cardinality greater than $c(\mathbf{x})(1-7\epsilon)\frac{3k-1}{3k}$. Since the fractional dynamic algorithm outputs an α-approximation $c(\mathbf{x})$ of the optimal fractional solution, we have that $s^* \leq \alpha c(\mathbf{x})$. Therefore H contains a k-matching of cardinality greater than $\frac{3k-1}{3k}(1-7\epsilon)s^*/\alpha$. We thus get the following theorem:

▶ **Theorem 11** (Sparsification). *Let G be a dynamic graph, $\epsilon > 0$, and $\mathcal{A}_m$ a dynamic algorithm that maintains an α-approximate fractional k-matching of G in $O(T_m)$ update time. Let s^* be the size of an optimal k-matching in G. Then the sparsification scheme maintains a sparsification H of G that runs in $O(T_m)$ update time and $O(s^* \cdot \operatorname{poly}(\log n, \epsilon^{-1}))$ request time. In expectation, H contains an integral k-matching of size at least $s^*/\left(\alpha\frac{3k}{3k-1}(1+\epsilon)\right)$ and satisfies $|H| = O(s^* \log n \operatorname{poly}(\epsilon^{-1}))$. If G is bipartite, the approximation ratio reduces to $\alpha(1+\epsilon)$.*

We now prove the Sparsify and Round Theorem:

Proof of Theorem 10. We build the algorithm $\mathcal{A}$ that maintains an approximate integral k-matching as follows: We use $\mathcal{A}_m$ to maintain an α-approximation of a fractional k-matching in G. At a given update, we compute a sparsification H of G using Theorem 11 of size $O(s^* \operatorname{poly}(\log n, \epsilon^{-1}))$ that contains an integral k-matching of size at least $\frac{1}{\alpha\frac{3k}{3k-1}(1+\epsilon)} \cdot s^*$. We then run the Ahn-Guha algorithm [1] on H in $O(s^* \operatorname{poly}(\log n, \epsilon^{-1}))$ time to get a $\alpha\frac{3k}{3k-1}(1+O(\epsilon))$-approximate k-matching s of G. Since every update only changes the size of an optimal solution by at most 1, the computed k-matching remains a good approximation of the optimal solution over the next ϵs updates, which yields an amortized update time of $O(\operatorname{poly}(\log n, \epsilon^{-1}))$. ◀

Next, we state Lemmas 12 and 13, whose proofs are given in the appendix.

▶ **Lemma 12.** *Let $i \in \mathbb{N}$, and $G_i = (V, E_i)$ be a graph. Let $\boldsymbol{x}$ be a fractional k-matching on G_i that satisfies $x_e \in \left((1+\epsilon)^{-i}, (1+\epsilon)^{-i+1}\right]$, and $d \geq \frac{1}{k\epsilon}$. If $d \geq (1+\epsilon)^{i-1}$, set $H_i := E_i$. Otherwise, let f_i be a total $(3\lceil k(1+\epsilon)^i \rceil)$-edge coloring of G_i. Sample $3\lceil kd \rceil$ colors uniformly at random (without replacement), and set H_i to be the set of all edges colored by one of the sampled colors. Then each edge e is sampled with probability $\mathbb{P}[e \in H_i]$ such that*

$$\min\{1, x_e \cdot d\} \cdot (1+\epsilon)^{-2} \leq \mathbb{P}[e \in H_i] \leq \min\{1, x_e \cdot d\} \cdot (1+\epsilon). \tag{2}$$

Then, if $x_e > \frac{1}{d}$, $\mathbb{P}[e \in H_i] = 1$. Moreover, any two adjacent edges are negatively associated, that is, for any edges e and e' that share a node, we have $\mathbb{P}[X_e | X'_e] \leq \mathbb{P}[X_e]$. Furthermore, the random variables $\{[X_e | X_{e'}] \mid e \ni v\}$ are negatively associated for any $v \in V, e' \ni v$.

▶ **Lemma 13** (Sparsification). *Let $\boldsymbol{x}$ be a fractional k-matching on a graph G, $\epsilon \in (0, \frac{1}{2})$, and $d \geq \max\left\{\frac{1}{k\epsilon}, \frac{4\log(2/\epsilon)}{k\epsilon^2}\right\}$. Let H be a subgraph of G, where each edge of G is sampled with probability $\mathbb{P}[e \in H]$, where $\mathbb{P}[e \in H] = 1$ if $x_e > \frac{1}{d}$ and*

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

Let $X_e := \mathbb{1}[e \in H]$. The edges need not be sampled independently, however two edges that are adjacent need to be negatively associated, that is, for any edges e and e' that share a node, we have $\mathbb{P}[X_e | X'_e] \leq \mathbb{P}[X_e]$. We also require that for any $v \in G, e' \ni v$, the random variables $\{[X_e | X_{e'}] | e \ni v\}$ are negatively associated. Then, H has a fractional k-matching $\mathbf{y}$ of expected value at least

$$\mathbb{E}\left[\sum_e y_e\right] \geq \sum_e x_e(1-6\epsilon).$$

7 Dynamic Algorithms for Maximum k-Edge Coloring

7.1 The `Greedy` Algorithm

Our first algorithm follows a simple greedy scheme: If an edge is added to the graph, we only check whether there is a common free color available at both its end nodes, and if it is the case, we color this edge with that color, otherwise we do nothing. If an edge colored c is removed from the graph, we try to color one edge adjacent to each of its end nodes with color c. This keeps the coloring maximal, and is thus a $(1+2\frac{\sqrt{3}}{3})$-approximation, as shown by Favrholdt and Nielsen [22]. If we maintain a table of free colors at every vertex, each insertion takes $O(\min\{k, \Delta\})$ time, as we have to go through all used colors at the endpoints to find a free one. Every edge deletion takes $O(\Delta)$ time, as we have to check whether the color c is free at the end of 2Δ edges in the worst case.

▶ **Theorem 14** (Greedy Algorithm). *The `Greedy` algorithm is deterministic and maintains a $(1+2\frac{\sqrt{3}}{3} \approx 2.155)$-approximation of a maximum k-edge coloring in $O(\Delta)$ update time.*

Note that in the case of bounded arboricity, one can maintain an orientation of the graph such that the out-degree of each node is at most a multiple of the arboricity, as shown by Chekuri et al. [15], in polylogarithmic time. In this case, each vertex is "responsible" for letting neighboring out-vertices know about its available colors. In that case, the update time drops to $O(c)$, where c is the arboricity of the graph, as the limiting factor now is not finding a suitable color for an edge, but rather updating the available colors on each vertex.

7.2 The Amortized Algorithms

Our next two algorithms rely on k-matchings. The idea is to maintain an approximate k-matching, and have it totally colored. However, as coloring can be expensive, we will not use the coloring algorithm at every update, but rather only once over multiple rounds, so we can amortize its cost. The following lemma formalizes the amortization technique:

▶ **Lemma 15** (Amortization). *Let $\epsilon > 0$, and assume we compute a coloring f that colors $p := \left|f^{-1}([k])\right|$ edges of a graph G. Then assume we have up to $\lfloor \epsilon p \rfloor$ edge insertions and deletions to G. Define a coloring g on G as follows: $g(e) := f(e)$ if e was already in G before the modifications, and $g(e) := \perp$ otherwise. Then if f is an α-approximation of maximum k-edge coloring before the updates, g is a $\alpha(1+3\epsilon)$-approximation after the updates if $\epsilon \leq \frac{1}{3}$.*

7.2.1 The MatchO Algorithm

If the updates to the graph are controlled by an oblivious adversary, we can use Bhattacharya, Gupta, and Mohan's algorithm [9] for dynamic **b**-matching. They maintain an integral $(2+\epsilon)$-approximation of **b**-matching against an oblivious adversary in $O(\epsilon^{-4})$ update time.

Our MatchO algorithm works as follows: We use Bhattacharya et al.'s algorithm to maintain a $(2+\epsilon)$-approximation for k-matching, which is represented by a graph H. Then, we compute a $(k+1)$-edge coloring using Gabow's coloring algorithm [27] on H, and discard the least used color to obtain a $(2+\epsilon)\frac{k+1}{k}$-approximation f of a k-edge coloring, guaranteed by Theorem 2. We refrain from recomputing the coloring for the next $\lfloor \epsilon \left|f^{-1}([k])\right| \rfloor$ updates, while continuing to update the k-matching after each update operation. By the Amortization Lemma (Lemma 15), this yields a $(1+3\epsilon)(2+\epsilon)\frac{k+1}{k}$-approximation, which is a $(2+5\epsilon)\frac{k+1}{k}$-approximation if $\epsilon \leq \frac{1}{3}$. This is particularly efficient if $\Delta(H) = O(\log^2 n \cdot \epsilon^{-2})$.

If on the other hand $\Delta(H) = \Omega(\log^2 n \cdot \epsilon^{-2})$, using the edge-coloring by Duan, He, and Zhang [20], we color all the edges of H with $(1+\epsilon)\Delta(H)$ colors. Discarding the $\epsilon\Delta(H)$ colors that color the fewest edges among them, this yields a $(2+O(\epsilon))$ approximation of the maximum k-edge coloring by Theorem 2.

Running time analysis. Let s be the size of H at the time of recoloring. Computing the recoloring (whether it is a $(k+1)$ or $(1+\epsilon)\Delta(H)$-edge coloring) and removing the least used color(s) gives a k-edge coloring f of size at least $\frac{k}{k+1}s$. Thus, $s \leq \frac{k+1}{k}\left|f^{-1}([k])\right| = O(\left|f^{-1}([k])\right|)$.

Let us now analyze the amortized update time. For each update, we spend $O(\epsilon^{-4})$ time to maintain the k-matching. If $\Delta(H) = O(\log^2 n \cdot \epsilon^{-2})$, coloring the k-matching with Gabow's algorithm takes $O(s \cdot \Delta(H) \log n) = O(s \cdot \text{poly}(\log n, \epsilon^{-1}))$ time. If on the other hand $\Delta(H) = \Omega(\log^2 n \cdot \epsilon^{-2})$, coloring the k-matching with Duan, He, and Zhang's algorithm takes $O(s\,\text{poly}(\log n, \epsilon^{-1}))$ time. Either way, this can be amortized over the next $\lfloor \epsilon \left|f^{-1}([k])\right| \rfloor$ updates, yielding an amortized update time of $O(\text{poly}(\log n, \epsilon^{-1}))$ for the complete algorithm.

We hence have the following result:

▶ **Theorem 16** (MatchO Algorithm). *Against an oblivious adversary, the* ***MatchO*** *algorithm runs in* $O(\text{poly}(\log n, \epsilon^{-1}))$ *amortized update time and maintains a* k*-edge coloring that is in expectation a* $(2+\epsilon)\frac{k+1}{k}$*-approximation.*

In the case of a bipartite graph, we can color the k-matching with k colors in $O(m \log k)$ time using the algorithm by Cole, Ost, and Schirra [19] instead of either Gabow's or Duan, He, and Zhang's algorithm. This yields a better approximation ratio:

▶ **Theorem 17** (Bipartite MatchO Algorithm). *Against an oblivious adversary, the* ***bipartite MatchO*** *algorithm runs in* $O(\text{poly}(\log n, \epsilon^{-1}))$ *amortized update time and maintains a* k*-edge coloring that is in expectation a* $(2+\epsilon)$*-approximation.*

7.2.2 The MatchA Algorithm

If the updates to the graph are controlled by an adaptive adversary, we can use Bhattacharya, Henzinger and Italiano's [11] algorithm for fractional **b**-matching. They maintain an $(7+\epsilon)$-approximate fractional **b**-matching of the dynamic graph in $O(\log(m+n)\epsilon^{-2})$ time against an adaptive adversary.

Our MatchA algorithm works as follows: We use Bhattacharya et al.'s algorithm as described in Section 6 to maintain a sparsification H of G. By the Sparsify and Round Theorem (Theorem 10), this takes $O(\log(m+n)\epsilon^{-2})$ update time and $O(|H|)$ time to output H when

requested. Let f^* be an optimal k-edge coloring of G of size $p^* := \left|f^{*-1}([k])\right|$. By Theorem 10, H in expectation contains an integral k-matching of size at least $p^* / \left(7\frac{3k}{3k-1}(1+\epsilon)\right)$ and $|H| = O(p^* \log n \cdot \text{poly}(\epsilon^{-1}))$.

Similarly to the case with an oblivious adversary, we will only compute a coloring in few, carefully selected rounds, and thus will not need to access H in every round. More specifically, we use the amortization technique from Lemma 15 to determine in which rounds to recolor H for the current graph. To recolor we run Ahn and Guha's static algorithm for **b**-matching [1] on H to compute a $(1+\epsilon)$-approximate (integral) k-matching H' of H. This ensures that the sparse graph has degree at most k. Finally, we either compute a $(k+1)$-edge coloring using Gabow's algorithm for edge-coloring [27] of H', if $\Delta(H') = O(\log^2 n \cdot \epsilon^{-2})$, or a $(1+\epsilon)\Delta(H)$-edge coloring using Duan, He, and Zhang's Algorithm otherwise, and discard the least used colors to obtain a $7(1+\epsilon)\frac{3k}{3k-1}\frac{k+1}{k}$-approximation f of the maximum cardinality k-edge coloring of G.

We refrain from recomputing the coloring for the next $\left\lfloor \epsilon \left|f^{-1}([k])\right| \right\rfloor$ updates, while continuing to maintain the k-matching. By Lemma 15, this yields an $7(1+3\epsilon)(1+\epsilon)\frac{3k+3}{3k-1}$-approximation f' of the current f^*, which is an $7(1+5\epsilon)\frac{3k+3}{3k-1}$-approximation if $\epsilon \leq \frac{1}{3}$.

Running time analysis. Let us analyze the amortized update time. For each update, we must first spend $O(\log n \cdot \text{poly}(\epsilon^{-1}))$ time to maintain the fractional k-matching and its sparsification. Requesting the sparse graph H takes $O(|H|)$ time. Finding a k-matching in it takes $O(|H| \cdot \text{poly}(\log n, \epsilon^{-1}))$ time [1]. Let s be the size of that k-matching after the current update. Coloring the k-matching with either Gabow's or Duan, He, and Zhang's algorithm takes $O(s\,\text{poly}(\log n, \epsilon^{-1}))$ time. Thus the total time for all updates in an interval starting at a recoloring and containing all updates up to the next recoloring is $O((s+|H|)\cdot\text{poly}(\log n, \epsilon^{-1}))$.

Let s^* be the size of the optimum k-matching after the current update. Recall that $s^* \leq \frac{k+1}{k}p^*$ by Corollary 3. Thus, we have that $s \leq s^* \leq \frac{k+1}{k}p^* = O(p^*)$. We also have $|H| = O(p^* \log n \cdot \text{poly}(\epsilon^{-1}))$. Hence, the total update time in an interval is $O(p^* \cdot \text{poly}(\log n, \epsilon^{-1}))$

Note also that $\left|f'^{-1}([k])\right| = O(\left|f^{-1}([k])\right|)$ as we recolor every $\left\lfloor \epsilon \left|f^{-1}([k])\right| \right\rfloor$ updates and each update changes the size of the k-edge coloring by at most one. Thus it follows that

$$p^* \leq 7(1+5\epsilon)\frac{3k+3}{3k-1}\left|f'^{-1}([k])\right| = O(\left|f'^{-1}([k])\right|) = O(\left|f^{-1}([k])\right|).$$

Hence we can amortize the total update time of an interval over the length of an interval, which consists of $\left\lfloor \epsilon \left|f^{-1}([k])\right| \right\rfloor$ updates, to achieve an amortized update time of $O(k \cdot \text{poly}(\log n, \epsilon^{-1}))$, resulting in the following theorem.

▶ **Theorem 18** (MatchA Algorithm). *Against an adaptive adversary, the* `MatchA` *algorithm runs in expected* $O(\text{poly}(\log n, \epsilon^{-1}))$ *amortized update time and maintains a* k*-edge coloring that is in expectation a* $7(1+\epsilon)\frac{3k+3}{3k-1}$*-approximation.*

Similarly to the previous section, in the case of a bipartite graph, we can color the k-matching with k colors in $O(m \log k)$ time using Cole, Ost, and Schirra's algorithm [19] instead of either Gabow's or that of Duan, He, and Zhang. This also drops the $\frac{k+1}{k}$ factor in the approximation computation above. Moreover, the integrality gap for the k-matching becomes 1, removing the $\frac{3k}{3k-1}$ factor. This yields a better approximation ratio:

▶ **Theorem 19** (Bipartite MatchA Algorithm). *Against an adaptive adversary, the* `bipartite MatchA` *algorithm runs in expected* $O(\text{poly}(\log n, \epsilon^{-1}))$ *amortized update time and maintains a* k*-edge coloring that is in expectation an* $7(1+\epsilon)$*-approximation.*

8 Conclusion

In this work, we have initiated the study of fully dynamic approximation algorithms for the NP-hard k-edge coloring problem by presenting and analyzing three dynamic algorithms. Moreover, we have demonstrated the close relationship between **b**-matching and k-edge coloring, making any advances in **b**-matching to automatically translate into better results for k-edge coloring. In the future, it would be thus interesting to investigate more into **b**-matching algorithms. In particular there is space for improvement in finding dynamic (fractional or not) **b**-matching algorithms against an adaptive adversary with approximation ratio better than 7 which still run in polylogarithmic time.

References

1 Kook Jin Ahn and Sudipto Guha. Near linear time approximation schemes for uncapacitated and capacitated b-matching problems in nonbipartite graphs. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 239–258. SIAM, 2014. doi:10.1137/1.9781611973402.18.

2 Sepehr Assadi. Faster vizing and near-vizing edge coloring algorithms. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4861–4898. SIAM, 2025. doi:10.1137/1.9781611978322.165.

3 Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal matching in O (log n) update time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 383–392. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.89.

4 Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021. doi:10.1109/FOCS52979.2021.00089.

5 Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 668–681. ACM, 2021. doi:10.1145/3406325.3451113.

6 Sayan Bhattacharya, Din Carmon, Martín Costa, Shay Solomon, and Tianyi Zhang. Faster $(\Delta+1)$-edge coloring: Breaking the $m\sqrt{n}$ time barrier. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 2186–2201. IEEE, 2024. doi:10.1109/FOCS61266.2024.00128.

7 Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1–20. SIAM, 2018. doi:10.1137/1.9781611975031.1.

8 Sayan Bhattacharya, Martín Costa, Shay Solomon, and Tianyi Zhang. Even faster $(\delta+1)$-edge coloring via shorter multi-step vizing chains. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4914–4947. SIAM, 2025. doi:10.1137/1.9781611978322.167.

9 Sayan Bhattacharya, Manoj Gupta, and Divyarthi Mohan. Improved algorithm for dynamic b-matching. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 15:1–15:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.15.

10 Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In Piotr Indyk, editor, *Proceedings of the*

Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, pages 785–804. SIAM, 2015. doi:10.1137/1.9781611973730.54.

11 Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Dynamic algorithms via the primal-dual method. *Inf. Comput.*, 261:219–239, 2018. doi:10.1016/j.ic.2018.02.005.

12 Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 398–411. ACM, 2016. doi:10.1145/2897518.2897568.

13 Marcin Bienkowski, David Fuchssteiner, and Stefan Schmid. Optimizing reconfigurable optical datacenters: The power of randomization. In Dorian Arnold, Rosa M. Badia, and Kathryn M. Mohror, editors, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2023, Denver, CO, USA, November 12-17, 2023*, pages 83:1–83:11. ACM, 2023. doi:10.1145/3581784.3607057.

14 Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 384–393. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.48.

15 Chandra Chekuri, Aleksander Bjørn Grodt Christiansen, Jacob Holm, Ivor van der Hoog, Kent Quanrud, Eva Rotenberg, and Chris Schwiegelshohn. Adaptive out-orientations with applications. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3062–3088. SIAM, 2024. doi:10.1137/1.9781611977912.110.

16 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 612–623. IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

17 Zhi-Zhong Chen, Sayuri Konno, and Yuki Matsushita. Approximating maximum edge 2-coloring in simple graphs. *Discret. Appl. Math.*, 158(17):1894–1901, 2010. doi:10.1016/J.DAM.2010.08.010.

18 Ilan Reuven Cohen, Binghui Peng, and David Wajc. Tight bounds for online edge coloring. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1–25. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00010.

19 Richard Cole, Kirstin Ost, and Stefan Schirra. Edge-coloring bipartite multigraphs in O(E log D) time. *Comb.*, 21(1):5–12, 2001. doi:10.1007/s004930170002.

20 Ran Duan, Haoqing He, and Tianyi Zhang. Dynamic edge coloring with improved approximation. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1937–1945. SIAM, 2019. doi:10.1137/1.9781611975482.117.

21 Ran Duan and Seth Pettie. Approximating maximum weight matching in near-linear time. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 673–682. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.70.

22 Lene M. Favrholdt and Morten N. Nielsen. On-line edge-coloring with a fixed number of colors. *Algorithmica*, 35(2):176–191, 2003. doi:10.1007/s00453-002-0992-3.

23 Uriel Feige, Eran Ofek, and Udi Wieder. Approximating maximum edge coloring in multigraphs. In Klaus Jansen, Stefano Leonardi, and Vijay V. Vazirani, editors, *Approximation Algorithms for Combinatorial Optimization, 5th International Workshop, APPROX 2002, Rome, Italy, September 17-21, 2002, Proceedings*, volume 2462 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 2002. doi:10.1007/3-540-45753-4_11.

24 S. M. Ferdous, Bhargav Samineni, Alex Pothen, Mahantesh Halappanavar, and Bala Krishnamoorthy. Semi-streaming algorithms for weighted k-disjoint matchings. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 53:1–53:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ESA.2024.53.

25 Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 448–456. ACM, 1983. doi:10.1145/800061.808776.

26 Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In David S. Johnson, editor, *SODA 1990, 22-24 January, San Francisco, CA, USA*, pages 434–443. SIAM, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320229.

27 Harold N Gabow, Takao Nishizeki, Oded Kariv, Daniel Leven, and Osamu Terada. Algorithms for edge-coloring. In *Technical report 41/85*. Tel Aviv University, 1985.

28 Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM*, 38(4):815–853, 1991. doi:10.1145/115234.115366.

29 Manoj Gupta and Richard Peng. Fully dynamic (1+ e)-approximate matchings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 548–557. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.65.

30 Kathrin Hanauer, Monika Henzinger, Lara Ost, and Stefan Schmid. Dynamic demand-aware link scheduling for reconfigurable datacenters. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications, New York City, NY, USA, May 17-20, 2023*, pages 1–10. IEEE, 2023. doi:10.1109/INFOCOM53939.2023.10229050.

31 Kathrin Hanauer, Monika Henzinger, Stefan Schmid, and Jonathan Trummer. Fast and heavy disjoint weighted matchings for demand-aware datacenter topologies. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, May 2-5, 2022*, pages 1649–1658. IEEE, 2022. doi:10.1109/INFOCOM48880.2022.9796921.

32 Ian Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981. doi:10.1137/0210055.

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

34 Marcin Jakub Kaminski and Lukasz Kowalik. Beyond the vizing's bound for at most seven colors. *SIAM J. Discret. Math.*, 28(3):1334–1362, 2014. doi:10.1137/120899765.

35 Mehrdad Khani, Manya Ghobadi, Mohammad Alizadeh, Ziyi Zhu, Madeleine Glick, Keren Bergman, Amin Vahdat, Benjamin Klenk, and Eiman Ebrahimi. SiP-ML: high-bandwidth optical network interconnects for machine learning training. In *Proceedings of the ACM SIGCOMM*, pages 657–675, 2021.

36 Dénes König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916.

37 Adrian Kosowski. Approximating the maximum 2- and 3-edge-colorable subgraph problems. *Discret. Appl. Math.*, 157(17):3593–3600, 2009. doi:10.1016/j.dam.2009.04.002.

38 Daniel Leven and Zvi Galil. NP completeness of finding the chromatic index of regular graphs. *J. Algorithms*, 4(1):35–44, 1983. doi:10.1016/0196-6774(83)90032-9.

39 L. Lovasz and M. D. Plummer. *Matching Theory*, volume 121 of *North-Holland Mathematics Studies*. Elsevier Science Ltd., 1986.

40 Silvio Micali and Vijay V. Vazirani. An o(sqrt(|v|) |e|) algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science,*

Syracuse, New York, USA, 13-15 October 1980, pages 17–27. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.12.

41 Jayadev Misra and David Gries. A constructive proof of vizing's theorem. *Inf. Process. Lett.*, 41(3):131–133, 1992. doi:10.1016/0020-0190(92)90041-S.

42 Jeffrey C Mogul and Lucian Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012. doi:10.1145/2378956.2378964.

43 Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 745–754. ACM, 2013. doi:10.1145/2488608.2488703.

44 Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 457–464. ACM, 2010. doi:10.1145/1806689.1806753.

45 Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

46 Corwin Sinnamon. A randomized algorithm for edge-colouring graphs in $o(m\sqrt{n})$ time. *CoRR*, abs/1907.03201, 2019. arXiv:1907.03201.

47 Shay Solomon. Fully dynamic maximal matching in constant update time. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.43.

48 Daniel Stubbs and Virginia Vassilevska Williams. Metatheorems for dynamic weighted matching. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.58.

49 Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Discret. Analiz.*, 3:25–30, 1964.

50 David Wajc. Rounding dynamic matchings against an adaptive adversary. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 194–207. ACM, 2020. doi:10.1145/3357713.3384258.

A Omitted Proofs

Proof of Theorem 2. Let s^* be the size of an optimum k-matching in G and let p^* be the size of an optimum k-edge coloring. Since any k-edge coloring is a k-matching, we have that $p^* \leq s^*$. Let $s = |H|$. The ℓ colors that color the smallest number of edges color at most $\frac{\ell}{k+\ell}s$ edges, otherwise the average of colored edges by color exceeds $\frac{s}{\ell+k}$. Let p be the number of edges colored by the remaining colors. Then $p \geq \frac{k}{k+\ell}s \geq \frac{k}{k+\ell}\frac{1}{\alpha}s^* \geq \frac{k}{k+\ell}\frac{1}{\alpha}p^*$. ◀

Proof of Theorem 4. Since the **b**-matching polytope is half-integral, we can find an optimal fractional b-matching $\mathbf{x}$ of $G = (V, E)$ such that $\mathbf{x}$ is half-integral. Let $H = (V, E_p)$ be the subgraph of G with $E_p = \{e \in E \colon x_e \notin \{0, 1\}\}$. If $E_p = \varnothing$, $\mathbf{x}$ is integral.

Otherwise, consider an Euler partition of H. If it contains a trail T which starts and ends at different nodes, write $T = \{e_1, \ldots, e_{|T|}\}$ where each e_i is adjacent to e_{i+1}. Let $\mathbf{x}^+$ be defined by

$$x_e^+ := \begin{cases} x_e & \text{if} \quad e \notin T, \\ x_e + \frac{1}{2}(-1)^{i+1} & \text{if} \quad e = e_i, \end{cases}$$

i.e., we alternatingly round up and down by $\frac{1}{2}$ along the trail. We clearly have that $\mathbf{x}^+ \in \mathcal{P}(G)$, since $\mathbf{x} \in [0, 1]^m$ and for each node $v \in V$ except the endpoints of T, an equal number of edges incident to v is rounded up and down. An end point v of the trail might have at most one more edge that sees its value increase than decrease by $\frac{1}{2}$. Since the node v is of odd degree in E_p, we have that $\sum_{e \ni v} x_e \leq b_v - \frac{1}{2}$, which implies $\sum_{e \ni v} x_e^+ \leq b_v$, i.e. the condition is still satisfied. We moreover have that $\sum_e x_e \leq \sum_e x_e^+$.

We can, hence, reduce the number of nodes of odd degree in E_p by creating solutions that are as good as $\mathbf{x}$ that have fewer and fewer odd degree nodes. We end up with H having only even degree nodes. By Euler's Theorem, there exists an Euler Circuit on every connected component of H.

For every node u, let $x(u) := \sum_{e \ni u} x_e$, and let C be a connected component of H. If there exists a node $u \in C$ such that $x(u) < \mathbf{b}_u$, then $x(u) \leq \mathbf{b}_u - 1$, and we can write the connected component $C = \{e_1, \ldots, e_{|C|}\}$ as an Eulerian circuit where each e_i is adjacent to $e_{(i+1) \bmod |C|}$, and where e_1 is adjacent to u. Then define:

$$x_e^+ := \begin{cases} x_e & \text{if} \quad e \notin C \\ x_e + \frac{1}{2}(-1)^{i+1} & \text{if} \quad e = e_i \end{cases}$$

We clearly have that $\mathbf{x}^+ \in \mathcal{P}(G)$, since $\mathbf{x} \in [0, 1]^m$, and for each node $v \in V$ except for u, we have that the number of edges adjacent to v losing $\frac{1}{2}$ is equal to the number of edges adjacent to v winning $\frac{1}{2}$ canceling out. Node u has at most two more edge that sees its value increase than decrease by $\frac{1}{2}$. Since $x(u) \leq b_u - 1$, we have $\sum_{e \ni u} x_e^+ \leq b_u$, the condition is still satisfied. We moreover have that $\sum_e x_e \leq \sum_e x_e^+$.

We can also show that even if no $u \in C$ satisfies $x(u) < b_u$, but C has an even number of nodes, the number of edges in the component is even, and thus the Eulerian circuit of C is of even length, and thus computing $\mathbf{x}^+$ as above will yield an integer solution on C. Note as well that if C is of size 1, then C contains no edges and x is already integral on C.

We end up with connected components $C_1, \ldots, C_\ell$ of odd size at least 3, where every node $u \in \bigcup_{i \in [\ell]} C_i$ satisfies $x(u) = b_u$. Each C_i can be seen as an Eulerian circuit. We will now build an integer solution $\mathbf{x}^-$ that approximates $\mathbf{x}$. For every $i \in [\ell]$, pick an arbitrary node $u_i \in C_i$, and write $C_i = \{e_1^i, \ldots, e_{|C_i|}^i\}$ where each e_j^i is adjacent to $e_{j+1 \bmod |C|}^i$, and

where e_1^i is adjacent to u^i. Define then:

$$x_e^- := \begin{cases} x_e & \text{if} \quad e \notin \bigcup_{i\in[\ell]} C_i \\ x_e + \frac{1}{2}(-1)^j & \text{if} \quad e = e_j^i \end{cases}$$

We clearly have that $\mathbf{x}^- \in \mathcal{P}(G)$, since $\mathbf{x} \in [0,1]^m$ and for each node $v \in V$, we have that the number of edges adjacent to v losing $\frac{1}{2}$ is larger than the number of edges adjacent to v winning $\frac{1}{2}$, and thus $\sum_{e\ni v} x_e^- \leq \sum_{e\ni v} x_e \leq b_v$. We moreover have that

$$\begin{aligned}\sum_e x_e^- &= \frac{1}{2}\sum_{v\in V} \mathbf{x}^-(v) = \frac{1}{2}\sum_{v\in V\setminus\bigcup_{i\in[\ell]} C_i} x(v) + \frac{1}{2}\sum_{i\in[\ell]}\Big(x(u_i) - 1 + \sum_{v\in C_i, v\neq u_i} x(v)\Big) \\ &\geq \frac{1}{2}\sum_{v\in V\setminus\bigcup_{i\in[\ell]} C_i} x(v) + \frac{1}{2}\sum_{i\in[\ell]}\Big(\mathbf{b}_{u_i} - 1 + \sum_{v\in C_i, v\neq u_i} b_v\Big) \\ &= \frac{1}{2}\sum_{v\in V\setminus\bigcup_{i\in[\ell]} C_i} x(v) + \frac{1}{2}\sum_{i\in[\ell]}\Big(-1 + \sum_{v\in C_i} b_v\Big).\end{aligned}$$

However, we have that $\sum_{v\in C_i} b_v \geq |C_i|\,\beta$ and that for each $i \in [\ell]$, $|C_i| \geq 3$ and thus $\ell \leq \frac{1}{3}\left|\bigcup_{i\in[\ell]} C_i\right|$. Therefore:

$$\frac{1}{2}\sum_{i\in[\ell]}\Big(-1+\sum_{v\in C_i} b_v\Big) \geq -l+\frac{1}{2}\sum_{i\in[l]}\sum_{v\in C_i} b_v \geq \big(1-\frac{1}{3\beta}\big)\frac{1}{2}\sum_{i\in[l]}\sum_{v\in C_i} b_v = \big(1-\frac{1}{3\beta}\big)\frac{1}{2}\sum_{i\in[l]}\sum_{v\in C_i} x(v)$$

Hence,

$$\sum_e x_e^- \geq \frac{1}{2}\sum_{v\in V\setminus\bigcup_{i\in[\ell]} C_i} x(v) + \big(1-\frac{1}{3\beta}\big)\frac{1}{2}\sum_{i\in[l]}\sum_{v\in C_i} x(v) \geq \big(1-\frac{1}{3\beta}\big)\frac{1}{2}\sum_{v\in V} x(v).$$

Since $\mathbf{x}^-$ is integral, the theorem follows. ◀

Proof of Lemma 7. Let us write $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. Define $G' = (V', E')$, where $V' = \{v'_1, \ldots v'_n, v''_1, \ldots, v''_n\}$, and $E' = \{e'_1, \ldots, e'_m, e''_1, \ldots, e''_m\}$, where for each $i \in [m]$, if $e_i = (v_j, v_\ell)$ for some $j, \ell \in [n]$, then e'_i and e''_i are defined as $e'_i = (v'_j, v''_\ell)$ and $e''_i = (v''_j, v'_\ell)$. Define b' such that $b'_{v'_i} = b'_{v''_i} = b_{v_i}$ for all i.

Let $\mathbf{x}$ be a vertex of $\mathcal{P}(G)$. Then $\mathbf{y}$, defined as $y_{e'_i} = y_{e''_i} = x_{e_i}$ for every $i \in [m]$, satisfies $\mathbf{y} \in \mathcal{P}(G')$. Indeed, for every $i \in [n]$ we have that $\sum_{v'_i\ni e} y_e = \sum_{v''_i\ni e} y_e = \sum_{v_i\ni e} x_e \leq b_{v_i}$.

Since $\mathbf{y} \in \mathcal{P}(G')$, and G' is bipartite, by Lemma 6, $\mathbf{y}$ is a convex combination of some $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(\ell)}$ such that for each $1 \leq j \leq \ell$ it holds that $\mathbf{y}^{(j)} \in \mathcal{P}(G')$ and all entries of $\mathbf{y}^{(j)}$ are in $\{0,1\}$. Let $\lambda_1, \ldots, \lambda_\ell \in [0,1]$ such that $y = \sum_{j\in[\ell]} \lambda_j \mathbf{y}^{(j)}$, and define for every $j \in [\ell]$, $\mathbf{x}^{(j)}$ such that $x_{e_i}^{(j)} = \frac{1}{2}(y_{e'_i}^{(j)} + y_{e''_i}^{(j)})$ for every $i \in [m]$, which is half-integer. Note that for every $i \in [m]$, $x_{e_i} = \frac{1}{2}(y_{e'_i} + y_{e''_i}) = \frac{1}{2}\sum_{j\in[\ell]} \lambda_j (y_{e'_i}^{(j)} + y_{e''_i}^{(j)}) = \sum_{j\in[\ell]} \lambda_j x_{e_i}^{(j)}$ and, hence, $\mathbf{x} = \sum_{j\in[\ell]} \lambda_j \mathbf{x}^{(j)}$. Thus, $\mathbf{x}$ is a convex combination of $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}$. We are left with showing that $\mathbf{x}^{(j)}$ belongs to $\mathcal{P}(G)$ for every $1 \leq j \leq \ell$. To do so note that for every $i \in [n], j \in [\ell]$,

$$\sum_{v_i\ni e} x_e^{(j)} = \frac{1}{2}\sum_{v_i\ni e} y_{e'}^{(j)} + y_{e''}^{(j)} = \frac{1}{2}\left(\sum_{v'_i\ni e'} y_{e'}^{(j)} + \sum_{v''_i\ni e''} y_{e''}^{(j)}\right) \leq b_{v_i},$$

which implies that $\mathbf{x}^{(j)} \in \mathcal{P}(G)$. Since $\mathbf{x}$ is a vertex of $\mathcal{P}(G)$ and a convex combination of $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}$, it must be equal to one of them, say $\mathbf{x} = \mathbf{x}^{(j)}$ for some $j \in [\ell]$, and thus is half-integral. ◀

Proof of Lemma 8. Since every even cycle is an even circuit, it is straightforward to see that G has no even cycle. Let us now assume that there exist two odd cycles C_1 and C_2 that share a node u in G. Suppose that C_1 and C_2 share no edge. Then, the circuit that starts at u and visits the first cycle, then the second, is an even circuit, a contradiction. Thus, C_1 and C_2 share at least one edge. Let $\{v, w\}$ be a shared edge such that the other neighbor of w in C_1 is different from its neighbor in C_2 (this edge must exists otherwise $C_1 = C_2$). Consider the path in C_2 from w to v that does not go through $\{v, w\}$. This path ends in C_1. Let v' denote the first node on that path that is different from w and is in C_1. We now have three edge-disjoint paths from w to v': two in C_1 and one in C_2. Two of them must have the same parity and thus together form an even circuit. ◀

Proof of Lemma 12. Wajc [50] has proven that choosing edges using that method (that is, choosing colors uniformly at random then picking all edges of those colors in a proper coloring) ensures that any two adjacent edges are negatively associated, and that for any $v \in V, e' \ni v$, the random variables $\{[X_e | X_{e'}] \mid e \ni v\}$ are negatively associated. Let e be an edge of E_i. We have three cases:

Case 1: If $d \geq (1+\epsilon)^{i-1}$, then all colors are sampled and $H_i = E_i$. Moreover, $x_e d \geq (1+\epsilon)^{-i}(1+\epsilon)^{i-1} = \frac{1}{1+\epsilon}$, and (2) holds trivially.

Case 2: If $x_e > \frac{1}{d}$, then $(1+\epsilon)^{-i+1} \geq x_e > \frac{1}{d}$ and in particular $d \geq (1+\epsilon)^{i-1}$, we thus refer to the previous case.

Case 3: If $x_e \leq \frac{1}{d}$ and $d \leq (1+\epsilon)^{i-1}$, then e is sampled with probability $\mathbb{P}[e \in H_i] = \frac{3\lceil kd \rceil}{3\lceil k(1+\epsilon)^i \rceil}$. Since $kd \geq \frac{1}{\epsilon}$, we have that $kd + 1 \leq kd(1+\epsilon)$, and thus:

$$\mathbb{P}[e \in H_i] = \frac{3\lceil kd \rceil}{3\lceil k(1+\epsilon)^i \rceil} \leq \frac{kd+1}{k(1+\epsilon)^i} \leq \frac{kd(1+\epsilon)}{k(1+\epsilon)^i} \leq x_e \cdot d \cdot (1+\epsilon) = \min\{1, x_e \cdot d\} \cdot (1+\epsilon)$$

On the other hand, since $(1+\epsilon)^i \geq (1+\epsilon)^{i-1} \geq d \geq \frac{1}{k\epsilon}$, we have that $k(1+\epsilon)^i + 1 \leq k(1+\epsilon)^{i+1}$. Therefore,

$$\mathbb{P}[e \in H_i] = \frac{3\lceil kd \rceil}{3\lceil k(1+\epsilon)^i \rceil} \geq \frac{kd}{k(1+\epsilon)^i + 1} \geq \frac{kd}{k(1+\epsilon)^{i+1}} \geq \frac{x_e \cdot d}{(1+\epsilon)^2} = \frac{\min\{1, x_e \cdot d\}}{(1+\epsilon)^2}.$$ ◀

For the proof of Lemma 13, we need the following inequality:

▶ **Theorem 20** (Bernstein's Inequality for Negatively Associated Variables)**.** *Let Y be the sum of negatively associated random variables $Y_1, \ldots, Y_\ell$, with $Y_i \in [0, U]$ for each $i \in [\ell]$. Then, for $\sigma^2 = \sum_{i=1}^{\ell} \mathrm{Var}(Y_i)$ and all $a > 0$,*

$$\mathbb{P}[Y > \mathbb{E}[Y] + a] \leq \exp\left(\frac{-a^2}{2(\sigma^2 + aU/3)}\right)$$

Proof of Lemma 13. Let $\mathbf{z} \in \mathbb{R}^E$ with $z_e := \frac{x_e(1-3\epsilon)}{\min\{1, x_e \cdot d\}} \cdot X_e$. By equation (3),

$$\mathbb{E}[z_e] = \mathbb{E}[z_e | X_e] \cdot \mathbb{P}[X_e] \geq x_e(1-3\epsilon) \cdot (1+\epsilon)^{-2} \geq x_e(1-5\epsilon) \tag{4}$$

Therefore, $\mathbf{z}$ is a good approximation for $\mathbf{x}$ in the sense that $\mathbb{E}\left[\sum_e z_e\right] \geq \sum_e x_e(1-5\epsilon)$. However, as we are scaling up x_e to get z_e for some edges e, $\mathbf{z}$ might not be a feasible fractional k-matching. We obtain a feasible fractional k-matching $\mathbf{y}$ from $\mathbf{z}$ as follows:

$$y_e := \begin{cases} 0 & \text{if } x_e < 1/d \text{ and } \max_{v \in e}\left\{\sum_{e' \ni v} z_{e'}\right\} > k \\ z_e & \text{otherwise} \end{cases}$$

▷ Claim 21. $\mathbf{y}$ is a feasible fractional k-matching.

Proof. Consider a node v. If $\sum_{e\ni v} z_e \le k$, then $\sum_{e\ni v} y_e \le \sum_{e\ni v} z_e \le k$. Otherwise, if $\sum_{e\ni v} z_e > k$, then $\sum_{e\ni v} y_e = \sum_{e\ni v, x_e\ge 1/d} z_e = \sum_{e\ni v, x_e\ge 1/d} x_e(1-3\epsilon) \le k$. ◁

To complete the proof, we will now show that for every edge e, we have $\mathbb{E}[y_e] \ge (1-\epsilon)\mathbb{E}[z_e]$. If $x_e \ge \frac{1}{d}$, this trivially follows since $y_e = z_e$ and thus $\mathbb{E}[y_e] = \mathbb{E}[z_e]$. We thus concentrate on the case $x_e < \frac{1}{d}$ and bound by $(1-\epsilon)$ the probability of the event $y_e \ne z_e$, that is the event $\max_{v\in e}(\sum_{e'\ni v} z_{e'}) > k$. In particular, we will consider the case $X_e = 1$.

Let v be an endpoint of e. We have that $x_e < 1/d \le k\epsilon$ (because we choose d such that $d \ge 1/k\epsilon$). Let $e' \ne e$ such that $v \in e'$. Since X_e and $X_{e'}$ are negatively correlated, we have that $\mathbb{P}[X_{e'}|X_e] \le \mathbb{P}[X_{e'}] \le \min\{1, x_{e'}\cdot d\}\cdot(1+\epsilon)$, by equation (3). Therefore:

$$\mathbb{E}[z_{e'}|X_e] = \frac{x_{e'}(1-3\epsilon)}{\min\{1,x_{e'}d\}}\cdot\mathbb{P}[X_{e'}|X_e] \le \frac{x_{e'}(1-3\epsilon)}{\min\{1,x_{e'}d\}}\cdot\mathbb{P}[X_{e'}] \le x_{e'}(1-3\epsilon)(1+\epsilon) \le x_{e'}(1-2\epsilon)$$

Hence:

$$\mathbb{E}\left[\sum_{e'\ni v} z_{e'}\Big|X_e\right] = \mathbb{E}[z_e|X_e] + \sum_{e'\ni v, e\ne e'} \mathbb{E}[z_{e'}|X_e] \le k\epsilon + \sum_{e'\ni v, e\ne e'} x_{e'}(1-2\epsilon) \le k(1-\epsilon)$$

We therefore expect $\mathbf{z}$ to not violate the constraint on v. To bound the probability that $\mathbf{z}$ does violate the constraint, we first compute the variance of $\left[\sum_{e'\ni v} z_{e'}|X_e\right]$, and in particular, for every $e' \ni v$, the variance of $[z_{e'}|X_e]$.

If e' is such that $x_{e'} \ge \frac{1}{d}$, then $\mathbb{P}[X_{e'}] = 1$, and in particular $\mathbb{P}[X_{e'}|X_e] = 1$. The variance of $z_e = \frac{x_e(1-3\epsilon)}{\min\{1,x_e d\}}\cdot X_e$ is therefore 0. On the other hand, if $x_{e'} < 1/d$, then $[z_{e'}|X_e]$ is a Bernoulli random variable scaled by $\frac{1-3\epsilon}{d}$, with success probability at most $\mathbb{P}[X_{e'}\big|X_e] \le min\{1, x_{e'}d\}\cdot(1+\epsilon) = x_{e'}d(1+\epsilon)$. Therefore, the variance of this random variable is at most

$$\mathrm{Var}([z_{e'}|X_e]) \le \left(\frac{1-3\epsilon}{d}\right)^2 \cdot x_{e'}d(1+\epsilon) \le \frac{x_{e'}}{d}.$$

Summing over all edges, we get:

$$\sum_{e'\ni v} \mathrm{Var}\left([z_{e'}|X_e]\right) \le \sum_{e'\ni v} \frac{x_{e'}}{d} \le \frac{k}{d}$$

Since the variables $\{[X_{e'}|X_e] \mid e' \ni v\}$ are negatively associated, so are the variables $\{[z_{e'} \mid X_e]|e' \ni v\}$, by closure of negative association under scaling by positive constants. Therefore, we can use Bernstein's inequality (cf. Theorem 20).

For $\left[\sum_{e'\ni v} z_{e'}\big|X_e\right]$ to go over k, it needs to exceed its expectation by at least $k\epsilon$. Since for all e' such that $x_{e'} > 1/d, [z_{e'}|X_e]$ is deterministic, this is equivalent to the sum only over edges e' such that $x_{e'} \le 1/d$ exceeding its expectation by at least $k\epsilon$. In that case, $z_{e'} \le \frac{(1-3\epsilon)}{d} \le \frac{1}{d}$. We thus have

$$\begin{aligned}\mathbb{P}\left[\sum_{e'\ni v} z_{e'} > k\Big|X_e\right] &\le \mathbb{P}\left[\sum_{e'\ni v, x_{e'}\le\frac{1}{d}} z_{e'} > \mathbb{E}\Big[\sum_{e'\ni v, x_{e'}\le\frac{1}{d}} z_{e'}\big|x_e\Big] + \epsilon k\Big|X_e\right] \\ &\le \exp\left(-\frac{\epsilon^2k^2}{2\cdot(k/d+\epsilon k/3d)}\right) \le \exp\left(-\frac{\epsilon^2 k}{2\cdot(1/d+\epsilon/3d)}\right) \le \exp\left(-\frac{\epsilon^2 k}{4d}\right),\end{aligned}$$

which is at most $\epsilon/2$ since $d \geq \frac{4\log(2/\epsilon)}{k\epsilon^2}$.

For y_e, we thus have that, conditioned on $e \in H$, the probability of the constraints on each of the nodes of e being violated is at most ϵ. By union bound, we thus have $\mathbb{P}[y_e = z_e | X_e] \geq (1-\epsilon)$. Combined with Equation (4), we have:

$$\mathbb{E}[y_e] = \frac{x_e(1-3\epsilon)}{\min\{1, x_e d\}} \cdot \mathbb{P}[y_e = z_e | X_e] \cdot \mathbb{P}[X_e] \geq (1-\epsilon) \cdot \mathbb{E}[z_e] \geq x_e(1-6\epsilon)$$

We thus conclude that H contains a fractional k-matching of expected value at least $1-6\epsilon$ times the value of the fractional k-matching $\mathbf{x}$ in G. ◀

Proof of Lemma 15. Let f^* be an optimal coloring before the updates, and g^* be an optimal coloring after the updates. Let $q := \left|g^{-1}([k])\right|$, $p^* := \left|f^{*-1}([k])\right|$, and $q^* := \left|g^{*-1}([k])\right|$. Since every deleted edge decreases the number of colored edges by at most 1, we have:

$$q \geq p - \lfloor \epsilon p \rfloor \geq p(1-\epsilon)$$

Similarly, since every added edge increases the size of the optimal coloring by at most 1, we have:

$$q^* \leq p^* + \lfloor \epsilon p \rfloor \leq p^*(1+\epsilon)$$

If f is an α-approximation, then $p^* \leq \alpha \cdot p$. Therefore, as long as $\epsilon \leq \frac{1}{3}$:

$$q^* \leq p^*(1+\epsilon) \leq \alpha \cdot p(1+\epsilon) \leq \alpha \cdot \frac{1+\epsilon}{1-\epsilon} \cdot q \leq \alpha \cdot q(1+3\epsilon)$$

Hence, whenever we compute a partial coloring of a dynamic graph of size p, we can charge the cost of that computation to the next $\lfloor \epsilon p \rfloor$ updates without recomputing anything, and while losing a $(1+3\epsilon)$ approximation factor at most. ◀

B Related Work

In this section, we give a more extensive overview over related work. As the maximum k-edge coloring problem is also closely related to various matching problems, we include relevant results for these problems.

Edge Coloring. Given a graph G, its *chromatic index* $\chi'(G)$ is the smallest value q such that all edges of G can be colored with q colors. It is straightforward that $\Delta(G) \leq \chi'(G)$, where $\Delta(G)$ denotes the maximum vertex degree in G. Vizing [49] showed that $\chi'(G) \leq \Delta(G)+1$. For bipartite graphs, $\chi'(G) = \Delta(G)$ [36]. In general, it is NP-hard to decide whether a given graph G has $\chi'(G) = \Delta(G)$ or $\chi'(G) = \Delta(G)+1$ already for $\Delta(G) = 3$ [32], and even if G is regular [38]. Note that if $\Delta(G) = 1$, then G's edges form a matching, whereas if $\Delta(G) = 2$, then G is a collection of cycles and paths and $\chi'(G) = 2$ iff all cycles have even length. Another lower bound on the chromatic index is given by the *odd density* $\rho(G) := \max_{S \subseteq V, |S| = 2i+1} \left\lceil \frac{E(S)}{i} \right\rceil$, where $E(S)$ denotes the set of edges in the subgraph induced by S: As all edges of the same color form a matching, at most $\left\lfloor \frac{|S|}{2} \right\rfloor = i$ edges of $E(S)$ can share the same color, so at least $\left\lceil \frac{E(S)}{i} \right\rceil$ colors are necessary to color $E(S)$. Edmonds [39] showed that the *fractional chromatic index* is equal to $\max\{\Delta(G), \rho(G)\}$.

Misra and Gries [41] designed an algorithm that uses at most $\Delta(G)+1$ colors. It processes the edges in arbitrary order and colors each in $O(n)$ time, thus resulting in an $O(mn)$ running

time overall. If new edges are added to the graph, they can be colored in $O(n)$ time. Their algorithm improved on an earlier approach by Gabow [27], which has a running time of $O(m\Delta(G)\log n)$. Simmanon [46] reduces the time for finding a $(\Delta(G)+1)$-edge coloring to $O(m\sqrt{n})$. By allowing more colors – up to $(1+\epsilon)\Delta$ colors – Duan, He and Zhang [20] further reduce the running time to $O(m \cdot \text{poly}(\log n, \epsilon^{-1}))$ as long as $\Delta(G) = \Omega(\log^2 n \cdot \epsilon^{-2})$. For bipartite graphs, Cole, Ost, and Schirra [19] gave an optimal algorithm (it uses Δ colors) with $O(m \log \Delta(G))$ running time.

Cohen, Peng, and Wajc [18] recently studied the edge coloring problem in the online setting and proved various competitive ratio results.

For the dynamic setting, Bhattacharya, Chakrabarty, Henzinger, and Naongkai [7] show how to maintain a $(2\Delta(G)-1)$-edge coloring in $O(\log n)$ worst-case update time. They also show that a $(2+\epsilon)\Delta(G)$-edge coloring can easily be maintained with $O(1/\epsilon)$ expected update time. If $\Delta(G) = \Omega(\log^2 n \cdot \epsilon^{-2})$, Duan, He and Zhang [20] maintain an edge-coloring using $(1+\epsilon)\Delta$ colors in amortized $O(\log^8 n \cdot \epsilon^{-4})$ update time.

Maximum k-Edge Coloring. For the maximum k-edge coloring problem, the number of available colors is limited to some $k \in \mathbb{N}$ and the task is to find a maximum-cardinality subset of the edges H such that for the subgraph restricted to H, $G\big|_H$, $\chi'(G\big|_H) \leq k$.

This problem was first studied by Favrholdt and Nielsen [22] in the online setting. They propose and analyze the competitive ratio of various online algorithms and show that every algorithm that never chooses to not color ("reject") a colorable edge has a competitive ratio between 0.4641 and $\frac{1}{2}$, and that any online algorithm is at most $\frac{4}{7}$-competitive.

Feige, Ofek, and Wieder [23] considered the k-edge coloring problem in the static setting and for multigraphs, motivated by a call-scheduling problem in satellite-based telecommunication networks. The authors show that for every $k \geq 2$, there exists an $\epsilon_k > 0$ such that it is NP-hard to approximate the problem within a ratio better than $(1+\epsilon_k)$. They also describe a static $(1-(1-1/k)^k)^{-1}$-approximation algorithm for general k as well as a $\frac{13}{10}$-approximation for $k=2$. The former algorithm applies a greedy strategy and works by repeatedly computing a maximum-cardinality matching M, then removing M from graph. As all edges in a matching can be colored with the same color, k repetitions yield a k-edge coloring. They also note that for a multigraph of multiplicity d, a $\frac{k+d}{k}$-approximate solution can be obtained by first computing a k-matching, then coloring the subgraph using $k+d$ colors (which is always possible, in analogy to Vizing's theorem), and then discarding the d colors that color the fewest edges. For simple graphs (i.e., $d=1$), this yields an approximation ratio of $\frac{k+1}{k}$. The authors also give an algorithm with an approximation ratio tending to $\frac{1}{\alpha}$ as $k \to \infty$, where α denotes the best approximation ratio for the chromatic index in multigraphs.

Several improved approximation results for the cases where $k=2$ and $k=3$ exist. The currently best ratios are $\frac{6}{5}$ for $k=2$ and $\frac{5}{4}$ for $k=3$ [37].

The maximum k-edge coloring problem was first studied in the edge-weighted setting by Hanauer, Henzinger, Schmid, und Trummer [31]. Here, instead of finding a maximum-cardinality subset of the edges, the total weight of the colored edges is to be maximized. The authors describe several approximation and heuristic approaches to tackle the problem in practice and provide an extensive experimental performance evaluation on real-world graphs. They also show that a double-greedy approach, where successively k weighted matchings are computed by a greedy algorithm, yields a $O(1)$-approximation. In a follow-up work, Hanauer, Henzinger, Ost, and Schmid [30] design a collection of different dynamic and batch-dynamic algorithms for the weighted k-edge coloring. Their focus is again more on the practical side. Ferdous et al. [24] recently studied the problem in the streaming setting.

Matching. The matching problem in the static setting has been subject to extensive research both in the unweighted and weighted case. The currently fastest deterministic algorithms for unweighted matching in general and bipartite graphs have a running time of $O(m\sqrt{n})$ [40, 28, 33]. For weighted matching on general graphs, the currently best running time is $O(n(m + n \log n))$ [26]. An excellent overview, which also encloses approximation ratios, is given by Duan and Pettie [21].

For the dynamic setting, Onak and Rubinfeld [44] present a randomized $O(1)$-approximation algorithm with $O(\log^2 n)$ update time. The algorithm by Baswana, Gupta, and Sen [3] improves the running time to $O(\log n)$ and the approximation ratio to 2. Later, Solomon [47] reduced the amortized expected update time to $O(1)$. Wajc [50] gives a metatheorem for rounding a dynamic fractional matching against an adaptive adversary and a $(2+\epsilon)$-approximate algorithm with constant update time or $O(\text{poly}(\log n, \epsilon^{-}1))$ worst-case update time.

Neiman and Solomon [43] show that a $\frac{3}{2}$-approximate matching can be maintained deterministically in $O(\sqrt{m})$ worst-case update time. Bhattacharya, Henzinger, and Italiano [10] give a deterministic $(3+\epsilon)$-approximation with $O(m^{1/3}\epsilon^{-2})$ amortized update time, as well as an $(4+\epsilon)$-approximation with $O(m^{1/3}\epsilon^{-2})$ worst-case update time. An improved algorithm is given by Bhattacharya, Henzinger, and Nanongkai [12], which has $O(\text{poly}(\log n, \frac{1}{\epsilon}))$ amortized update time and an approximation ratio of $(2+\epsilon)$. A $(1+\epsilon)$-approximation algorithm with $O(\sqrt{m}\epsilon^{-2})$ worst-case update time is given by Gupta and Peng [29] for $\epsilon < \frac{1}{2}$. The authors also give an algorithm for the weighted case with the same approximation ratio and a worst-case update time of $O(\sqrt{m}\epsilon^{-2-O(1/\epsilon)} \log W)$, where W is the largest weight of an edge in the graph.

For bipartite graphs, Bosek, Leniowski, Sankowski, and Zych [14] give a partially dynamic algorithm for either the insertions-only or deletions-only setting, which runs in $O(m\sqrt{n})$ total time, thus matching the time of the Hopcroft-Karp static algorithm [33]. Due to the direct reduction of matching to maximum flow, the static case can now be solved in $O(m^{1+o(1)})$ time thanks to a breakthrough result of Chen, Kyng, Liu and Peng [16].

Stubbs and Williams [48] show how to transform a dynamic α-approximation algorithm for the unweighted matching problem to a $(2+\epsilon)\alpha$-approximation algorithm for the weighted setting. The running time increases by a factor of $\epsilon^{-2} \log^2 W$, where W denotes the maximum weight of an edge. Bernstein, Dudeja, and Langley [5] improve on this result w.r.t. running time and show that a $(1+\epsilon)\alpha$-approximation algorithm can be obtained in the case of bipartite graphs.

Various results [4] also exist for the "value" version of the problem, where one is only interested in the maximum size or weight, but not in the set of edges.

b-Matching. Gabow [25] gives a $O(\sqrt{\|b\|_1}m)$-time algorithm to compute a $\mathbf{b}$-matching in the unweighted, static setting. If the $\mathbf{b}$-matching is weighted, the running time is $O(\|b\|_1 \cdot \min(m \log n, n^2))$. Ahn and Guha [1] give an algorithm that computes an $(1+\epsilon)$-approximation for $\mathbf{b}$-matching and runs in $O(m \,\text{poly}(\log n, \epsilon^{-1}))$ time.

For dynamic graphs, Bhattacharya, Henzinger, and Italiano [11] give a deterministic algorithm that maintains an $O(1)$-approximate fractional k-matching with $O(\log^3 n)$ amortized update time. This result is improved by Bhattacharya, Gupta, and Mohan [9], who show how to maintain an integral $(2+\epsilon)$-approximate $\mathbf{b}$-matching in expected amortized $O(1/\epsilon^4)$ update time, against an oblivious adversary.