Decremental Matching in General Graphs

Sepehr Assadi*[1], Aaron Bernstein†[1], and Aditi Dudeja‡[1]

[1]Rutgers University

Abstract

We consider the problem of maintaining an approximate maximum integral matching in a dynamic graph G, while the adversary makes changes to the edges of the graph. The goal is to maintain a $(1+\varepsilon)$-approximate maximum matching for constant $\varepsilon > 0$, while minimizing the update time. In the fully dynamic setting, where both edge insertion and deletions are allowed, Gupta and Peng (see [GP13]) gave an algorithm for this problem with an update time of $O(\sqrt{m}/\varepsilon^2)$.

Motivated by the fact that the $O_\varepsilon(\sqrt{m})$ barrier is hard to overcome (see Henzinger, Krinninger, Nanongkai, and Saranurak [HKNS15]; Kopelowitz, Pettie, and Porat [KPP16]), we study this problem in the *decremental* model, where the adversary is only allowed to delete edges. Recently, Bernstein, Probst-Gutenberg, and Saranurak (see [BGS20]) gave an $O(\text{poly}(\log n/\varepsilon))$ update time decremental algorithm for this problem in *bipartite graphs*. However, beating $O(\sqrt{m})$ update time remained an open problem for *general graphs*.

In this paper, we bridge the gap between bipartite and general graphs, by giving an $O_\varepsilon(\text{poly}(\log n))$ update time algorithm that maintains a $(1+\varepsilon)$-approximate maximum integral matching under adversarial deletions. Our algorithm is randomized, but works against an adaptive adversary. Together with the work of Grandoni, Leonardi, Sankowski, Schwiegelshohn, and Solomon [GLS+19] who give an $O_\varepsilon(1)$ update time algorithm for general graphs in the *incremental* (insertion-only) model, our result essentially completes the picture for partially dynamic matching.

1 Introduction

In dynamic graph algorithms, the main goal is to maintain a key property of the graph while an adversary makes changes to the edges of the graph. An algorithm is called *incremental* if it handles only insertions, *decremental* if it handles only deletions and *fully dynamic* if it handles both insertions as well as deletions. The goal is to minimize the update time of the algorithm, which is the time taken by the algorithm to adapt to a single adversarial edge insertion or deletion and output accordingly. For incremental/decremental algorithms, one typically seeks to minimize the *total update time*, which is the aggregate sum of update times over the *entire* sequence of edge insertions/deletions.

We consider the problem of maintaining a $(1+\varepsilon)$-approximation to the maximum matching in a dynamic graph. In the fully dynamic setting, the best known update time for this problem is $O(\sqrt{m})$ (see [GP13]), and the conditional lower bounds proved in the works of [HKNS15] and [KPP16] suggest that $O(\sqrt{m})$ is a hard barrier to break through. For this reason, several relaxations of this problem have been studied. For example, one line of research has shown that we can get considerably faster update times if we settle for large approximation factors (see for example [BHI15, BHN16, BK21, BS16, Waj20, RSW22, BK22]). Another research direction has been to consider the more relaxed incremental or decremental models. In the incremental (insertion-only) setting, there have been a series of upper and lower bound results [BLSZ14, Dah16, Gup14], culminating in the result of [GLS+19], who gave an optimal $O_\varepsilon(m)$ *total* update time (amortized $O_\varepsilon(1)$) for $(1+\varepsilon)$-approximate maximum matching.

*sepehr.assadi@rutgers.edu. Supported by NSF CAREER Grant CCF-2047061, and a gift from Google Research.
†bernstei@gmail.com. Funded by NSF CAREER Grant 1942010.
‡aditi.dudeja@rutgers.edu

The decremental (deletion-only) setting requires an entirely different set of techniques. In fact, $O(\sqrt{m})$ update time ($O(m^{1.5})$ total time) remained the best known until recently, when [BGS20] gave an poly$(\log n/\varepsilon)$ amortized update time algorithm for bipartite graphs. However, achieving a similar result for non-bipartite graphs remained an open problem. Our main theorem essentially closes the gap between bipartite and general graphs.

Theorem 1. Let G be an unweighted graph and let $\varepsilon \in (0, 1/2)$. Then, there exists a decremental algorithm with total update time $\tilde{O}_\varepsilon(m)$ (amortized $\tilde{O}(1)$) that maintains an integral matching M of value at least $(1-\varepsilon) \cdot \mu(G)$, with high probability, where G refers to the current version of the graph. The algorithm is randomized but works against an adaptive adversary. The dependence on ε is $2^{O(1/\varepsilon^2)}$.

Our result largely completes the picture for partially dynamic matching by showing that in general graphs one can achieve poly$(\log n)$ update time in both incremental and decremental settings. But there are a few secondary considerations that remain. Firstly, our update time is $O(\text{poly}(\log n))$, rather than the $O(1)$ for the incremental setting (see [GLS+19]). Secondly, both the incremental of [GLS+19] and our decremental result for general graphs have an exponential dependence on $1/\varepsilon$, whereas incremental/decremental algorithms for bipartite graphs have a polynomial dependence on $1/\varepsilon$ (see [GLS+19, Gup14]).

In algorithms literature, it has been the case that efficient matching algorithms for bipartite graphs do not easily extend to general graphs. Existence of blossoms (among other things), poses a technical challenge to obtaining analogous results for the general case. Consider the polynomial time algorithms for maximum matching for bipartite graphs, the most efficient algorithm, using alternating BFS, was discovered by Hopcroft and Karp; Karzanov (see [HK73, Kar73]) in 1973. However, several new structural facts and algorithmic insights were used by Micali and Vazirani to get the same runtime for general graphs (see [MV80]). This is also a feature of recent work in different models, such as the streaming (see [AG11, EKMS11] and [FMU21]), fully dynamic (see [BS15, BS16] and [BHN16, BLM20]), and parallel models (see [FGT16, ST17] and [MV00, AV20b]). We refer the reader to Section 1.3 of [AV20a] for a detailed discussion of this phenomenon.

2 High-Level Overview

Our algorithm for Theorem 1 follows the high-level framework of *congestion balancing* introduced by Bernstein, Probst-Gutenberg, and Saranurak [BGS20]. They used this framework to solve approximate decremental matching in bipartite graphs, and also to solve more general flow problems. But the framework as they used it was entirely limited to cut/flow problems. As we discuss below, extending this framework to non-bipartite graphs introduces significant technical challenges. Moreover, our result shows how the key subroutine of congestion balancing is naturally amenable to a primal-dual analysis, which we hope can pave the way for this technique to be applied to other decremental problems.

2.1 Previous Techniques

A key observation of [BGS20] is that it is sufficient to develop an $\tilde{O}_\varepsilon(m)$ algorithm that does the following: it either maintains a fractional matching of size at least $(1-2\varepsilon) \cdot \mu(G)$ or certifies that $\mu(G)$ has dropped by a $(1-\varepsilon)$ factor because of adversarial deletions. Since a result of [Waj20] enables us to round any bipartite fractional matching to an integral matching of almost the same value, their observation gives an algorithm to maintain an integral matching of size $(1-3\varepsilon) \cdot \mu(G)$ in $\tilde{O}_\varepsilon(m)$ time under adversarial deletions. To motivate why [BGS20] consider computing a fractional matching, consider the following "lazy" algorithm that works with an integral matching: compute an $(1+\varepsilon)$ approximate integral matching M of G using a static $O(m/\varepsilon)$ algorithm, wait for $\varepsilon \cdot \mu(G)$ edges of M to be deleted and then recompute the matching. Since we assume an adaptive adversary, the update time could be as large as $\Omega(m^2/\varepsilon \cdot n)$, this is because the adversary could proceed by only deleting edges of M. As a result, the goal should be to maintain a *robust* matching that can survive many deletions. Thus, [BGS20] aim to maintain a "balanced"

fractional matching $\vec{x}$ that attempts to put a low value on every edge. In doing so, the adversary will have to delete a lot of edges of G to reduce the value of $\vec{x}$ by $\varepsilon \cdot \mu(G)$.

Balanced Fractional Matching in Bipartite Graphs In order to ensure that the fractional matching is spread out and robust, the authors of [BGS20] impose a capacity function κ on the edges of the graph (initially, all edges have low capacity) and compute a fractional matching obeying these capacities. The main ingredient of the algorithm is the subroutine M-OR-E*(G, ε, κ) which returns one of the following in $\tilde{O}_\varepsilon(m)$ time:

(a) A fractional matching $\vec{x}$ such that $\sum_{e \in E} x(e) \geqslant (1-\varepsilon) \cdot \mu(G)$ and $x(e) \leqslant \kappa(e)$ for all $e \in E$, or,

(b) A set of edges E^* such that have the following two properties.

  (a) The total capacity through E^* must be small: $\kappa(E^*) = O(\mu(G) \log n)$ and,

  (b) For all $|M| \geqslant (1-3\varepsilon) \cdot \mu(G)$, $|M \cap E^*| \geqslant \varepsilon \cdot \mu(G)$.

Property (a) ensures that the total capacity increase is small, while Property (b) ensures that we only increase capacity on important edges that are actually needed to form a large matching. The authors of [BGS20] show that M-OR-E*() can be used as a black-box to solve decremental matching: at each step, M-OR-E*() is used to find a large fractional matching $\vec{x}$ (this matching is then rounded using [Waj20] to get an integral matching), or to output the set E^* along which we increase capacities. They are able to show that because of Properties (a) and (b), the edge capacities remain small on average.

The *congestion balancing* framework of [BGS20] thus, consists of an outer algorithm that uses M-OR-E*() as a subroutine. The outer algorithm for bipartite graphs carries over to general graphs as well. But, M-OR-E*() is significantly more challenging to implement for the case of general graphs, so this subroutine will be our focus for the rest of the high level review. For the case of bipartite graphs the algorithm M-OR-E*() is easier to implement because maximum fractional matchings correspond to maximum flows in bipartite graphs. Hence, existing algorithms for approximate maximum flows can be used to find the approximate maximum fractional matching obeying capacity κ. Moreover, if such a fractional matching is not large, then in bipartite graphs, the set of bottleneck edges is exactly a minimum cut of the graph. For general graphs, due to the odd set constraints, max flow, which was the key analytic and algorithmic tool in [BGS20], no longer corresponds to a maximum fractional matching that avoids the integrality gap.

2.2 Our Contribution: Implementing M-or-E*() in General Graphs

At a high-level, there are several structural and computational challenges to implementing M-OR-E*() in the case of general graphs. We explain what the potential impediments are, and detail how our techniques circumvent these.

Fractional Matchings in General Graphs In general graphs, not all fractional matchings have a large integral matching in their support and therefore, cannot be rounded to give a large matching. While fractional matchings that obey odd set constraints do avoid the integrality gap, it seems hard to compute such a matching that also obeys capacity function κ. In order to get past this, we define a candidate fractional matching that is both easy to compute as well as contains a large integral matching in its support. More concretely, our fractional matching either puts flow one through an edge, or a flow of value at most ε. It can be proved that such a fractional matching obeys all small odd set constraints, and avoids the integrality gap. Our main contribution are two structural lemmas which show that we can find our candidate matching efficiently.

(a) First, given a graph G with capacity κ, we want to determine if the *value* of the maximum fractional matching obeying κ and odd set constraints (denoted $\mu(G, \kappa)$) is at least $(1-\varepsilon) \cdot \mu(G)$. In general graphs, we do this by giving a sampling theorem: let G_s be the graph created by sampling edge e with probability proportional to $\kappa(e)$, then $\mu(G_s) \geqslant \mu(G, \kappa) - \varepsilon \cdot n$ with high probability. Thus, $\mu(G_s)$

is a good proxy for $\mu(G, \kappa)$ and it can be estimated efficiently by running any integral matching algorithm on G_s.

(b) Suppose we have determined at some point that $\mu(G, \kappa)$ is large, we are still left with the task of finding a fractional matching. Our next contribution is a structural theorem that enables us to deploy existing flow algorithms to find such a matching. Let M be the approximate maximum matching of G_s. Let $M_L = \{e \in M \mid \kappa(e) \leqslant \beta\}$ and $M_H = \{e \in M \mid \kappa(e) > \beta\}$, where $\beta = O(\text{poly}(\log n))$, and L and H are for low and high respectively. Let $V_L = V(M_L)$ and $V_H = V(M_H)$. Intuitively, M breaks up our vertex set into two parts: vertices matched by low capacity edges (denoted V_L) and those that are matched by high capacity edges (denoted V_H). By adding some slack to our capacity constraints (we show that some slack can be incorporated in congestion balancing framework), we are able to treat the high-capacity edges as integral and compute a matching on V_H using a black box for integral matching in general graphs. Additionally, we show that the maximum fractional matching on low capacity edges of $G[V_L]$ has value at least as much as $|M_L|$ (the low capacity edges of M), up to an additive error of $\varepsilon \cdot n$. To compute this fractional matching, we show that because we are only considering edges of small capacity, small odd set constraints are automatically satisfied, so we can transform $G[V_L]$ into a *bipartite graph* and then use an existing flow algorithm.

The second obstacle is finding the set E^*. As mentioned before, for the case of bipartite graphs, the max flow-min cut theorem gives us an easy characterization of the bottleneck edges. However, for the case of general graphs, this characterization is not clear. To get around this, we consider the dual of the matching LP of G_s, and show that the bottleneck edges can be identified by considering the dual constraints associated with the edges. This generalizes the cut-or-matching approach of [BGS20]. Since G_s is integral, we can compute the approximate dual by using an existing primal-dual algorithm of [DP14] for integral matching in general graphs.

Additionally, there are some secondary technical challenges as well. As mentioned before, our structural theorems only guarantee preservation of matching sizes up to an additive error of $\varepsilon \cdot n$. When $\mu(G) = o(n)$, then the results, applied directly are insufficient for us. To get around this, we use a vertex sparsification technique to get $O_\varepsilon(\log n)$ multigraphs which preserve all matchings of G, but contain only $O(\mu(G)/\varepsilon)$ vertices. However, we now have to show that all of our ideas work for multigraphs as well. Finally, the rounding scheme of [Waj20] cannot be applied as a black-box to any fractional matching in a general graph. Thus, unlike in [BGS20], we cannot use [Waj20] as a black box and instead have to embed its techniques into the congestion balancing framework.

Assumption on Matching Size Let G be the input graph with vertex set V, note that if $\mu(G) \leqslant 100 \log |V|$ at any time, then we can maintain a $(1+\varepsilon)$ approximate matching using Definition 3.2 and Lemma 3.3 from [GP13] to solve the problem in $\tilde{O}(m/\varepsilon)$ time. Thus, we only run our algorithm while $\mu(G) \geqslant 100 \log |V|$. As mentioned before, our structural theorems, Lemma 29 and Lemma 40 are proved for multigraphs H with matching size at least $\mu(H) = \Omega(\varepsilon \cdot |V(H)|)$. This is because Lemma 52 allows us to reduce to the problem of decremental matching in multigraphs with large matching. Now, suppose H is the multigraph obtained by running the reduction of Lemma 52 on G, the input graph. Then, the structural theorems, Lemma 29 and Lemma 40 hold with probability at least $1 - \exp(-\mu(H))$. Since $\mu(H) = \Omega(\mu(G))$ and $\mu(G) \geqslant 100 \log |V|$, these structural theorems hold with high probability.

3 Preliminaries

We consider the problem of maintaining an approximate maximum integral matching in a graph G in the decremental setting. In this setting, we are given a graph (possibly non-bipartite) $G_0 = (V_0, E_0)$ with $|V_0| = n$ and $|E_0| = m$, and the adversary deletes edges from the graph one at a time. The goal is to maintain an approximate maximum matching of the graph G as edges are deleted, while minimizing the *total update time*, that is the aggregate sum of update times over the *entire* sequence of deletions.

Notation. Throughout the paper, we will use G to refer to the current version of the graph, and let V and E be the vertex and edge sets of G respectively. Additionally, let $\mu(G)$ denote the size of the maximum integral matching of G (the current graph). During the course of the algorithm, we will maintain a fractional matching which corresponds to a non-negative vector $\vec{x} \in \mathbb{R}_{\geqslant 0}$ satisfying fractional matching constraints: $\sum_{e \ni v} x(e) \leqslant 1$. For a set $S \subseteq E$, then we let $x(S) = \sum_{e \in S} x(e)$. Given a capacity function $\kappa(e)$, we say that x obeys κ if $x(e) \leqslant \kappa(e)$ for all $e \in E$. For a vector $\vec{x}$, we use $\text{supp}(\vec{x})$ to be set of edges that are in the support of $\vec{x}$. For a fractional matching $\vec{x}$, we say that $\vec{x}$ satisfies odd set constraints if for every odd-sized $B \subseteq V$, $\sum_{e \in G[B]} x(e) \leqslant \frac{|B|-1}{2}$.

Throughout this paper, for any $\varepsilon \in (0,1)$, we will let $\alpha_\varepsilon = \log n \cdot 2^{60/\varepsilon^2}$ and $\rho_\varepsilon = \log n \cdot 2^{40/\varepsilon^2}$. We now restate our main result.

Theorem 1. Let G be an unweighted graph and let $\varepsilon \in (0, 1/2)$. Then, there exists a decremental algorithm with total update time $\tilde{O}_\varepsilon(m)$ (amortized $\tilde{O}(1)$) that maintains an integral matching M of value at least $(1-\varepsilon) \cdot \mu(G)$, with high probability, where G refers to the current version of the graph. The algorithm is randomized but works against an adaptive adversary. The dependence on ε is $2^{O(1/\varepsilon^2)}$.

In order to solve this problem, we reduce to the case where it is sufficient to solve the same problem in multigraphs which have a large matching. More concretely, we prove the following theorem, and then show how that implies an algorithm for Theorem 1.

Theorem 2. We are given an unweighted multi-graph $G_0 = (V, E_0)$. Let $|V| = n$, and $\varepsilon \in (0, 1/2)$. Suppose $\mu(G_0) \geqslant \varepsilon \cdot n/100$ and $\mu \geqslant (1-\varepsilon) \cdot \mu(G)$. Suppose G is subject to adversarial deletions. There is an algorithm Dec-Matching(G, μ, ε), which processes deletions in $\tilde{O}_\varepsilon(m)$ total update time and has the following guarantees.

(a) When the algorithm terminates, $\mu(G) < (1-2\varepsilon) \cdot \mu$. Upon termination the algorithm outputs "no".

(b) Until the algorithm terminates, it maintains an integral matching with size at least $(1-20\varepsilon) \cdot \mu$.

The main contribution of our paper is to give an algorithm that proves Theorem 2. We first show how an algorithm for Theorem 2 gives us an algorithm for Theorem 1. For this, we need the following reduction, which has been used previously in other settings such as for stochastic matchings (see for example [AKL19] and [CCE+16]). Here, we use it in the decremental setting.

Lemma 3. [AKL19] Let $\delta \in (0,1)$ and let G be a graph with maximum matching size at least μ and at most $(1+\delta) \cdot \mu$. There is an algorithm Vertex-Red(G, μ, δ) that takes as input G, μ, and δ, and in $O(m \cdot \log n/\delta^4)$ time returns $\lambda = \frac{100 \cdot \log n}{\delta^4}$ multi-graphs $H_1, \cdots, H_\lambda$ that have the following properties, where the second property holds with probability at least $1 - \exp(-\mu \cdot \log n/\delta^4)$.

(a) For all $i \in [\lambda]$, $|V(H_i)| = \frac{4 \cdot (1+\delta) \cdot \mu}{\delta}$ and $E(H_i) \subseteq E(G)$.

(b) Suppose G is subject to deletions, which we simulate on each of the H_i's. More concretely, if an edge e is deleted from G, then its copy in each of the H_i's (if present) is also deleted. Suppose at the end of the deletion process, $\mu(G) \geqslant \tilde{\mu}$ for some $\tilde{\mu} \geqslant (1-2\delta) \cdot \mu$. Then, $\mu(H_i) \geqslant (1-\delta) \cdot \tilde{\mu}$ for some $i \in [\lambda]$.

Additionally, we will need the following algorithm to compute an integral matching in the static setting in general graphs (see [DP14]).

Lemma 4. [DP14] There is an $O(m/\varepsilon)$ time algorithm Static-Match() that takes as input a graph G, and returns an integral matching M of G with $|M| \geqslant (1-\varepsilon) \cdot \mu(G)$.

We now show how Theorem 2 in combination with Lemma 3 and Lemma 4 implies Theorem 1. We first give an informal description of the algorithm.

Description of Algorithm for Theorem 1 (Algorithm 1) The algorithm takes as input a graph G, and a parameter $\varepsilon > 0$. It first estimates the size of the maximum matching of G by running Static-Match(G, ε) on input G and ε. We denote this estimate by μ. The algorithm then initiates a procedure to create multiple parallel instances of the algorithm Dec-Matching(). As mentioned in Theorem 2, the algorithm Dec-Matching() can maintain a $(1+\varepsilon)$-approximate maximum matching in a multi-graph, provided the size of the matching in the multi-graph is large. So, we use the reduction mentioned in Lemma 3 which creates multigraphs $H_1, H_2, \cdots, H_\lambda$, which have have $O(\mu(G)/\varepsilon)$ vertices as opposed to n vertices, and we run parallel instances of Dec-Matching$(H_i, \mu \cdot (1-\varepsilon), \varepsilon)$ for all $i \in [\lambda]$. This procedure also instantiates a pointer cur, which points to the least indexed H_i, with a matching of size at least $(1-\varepsilon) \cdot \mu$. The algorithm will output the matching indexed by cur (denoted M_{CUR}). The algorithm then moves on to a procedure for handling deletions of edges. If an edge $e \in G$ is deleted, then it is deleted from each of the H_i's. If at this point, the size of the matching of any H_{CUR} by a large value, then we increment the value of cur. If the sizes of the maximum matchings of all H_i have dropped by a significant amount, then we can conclude that with high probability, $\mu(G)$ has also dropped by a significant amount (by Lemma 3(b)), and we restart the algorithm.

Proof of Theorem 1. We show that Algorithm 1 proves Theorem 1. Suppose we run Algorithm 1 on G, ε. We first argue that it returns an integral matching of G of size at least $(1-10\varepsilon) \cdot \mu(G)$, where $\mu \geqslant \mu(G) \cdot (1-\varepsilon)$. To see this, first observe that the multi-graphs $H_1, \cdots, H_\lambda$ obtained as an output of Vertex-Red$(G, \mu \cdot (1+\varepsilon), \varepsilon)$ have $\frac{4 \cdot (1+\varepsilon) \cdot \mu}{\varepsilon} \leqslant 8 \cdot \mu/\varepsilon$ vertices, and $\mu(H_i) \geqslant (1-\varepsilon) \cdot \mu$. Thus, H_i satisfy the requirements of Dec-Matching() mentioned in Theorem 2. Hence, each instance of Dec-Matching(H_i, μ, ε), until it returns no, maintains a matching of size at least $(1-9\varepsilon) \cdot \mu$, which is at least $(1-10\varepsilon) \cdot \mu(G)$.

We now want to bound the runtime of the algorithm. To do this, we first make the following claim.

Claim 5. Each time Algorithm 1 executes Line 32, $\mu(G)$ has dropped by a factor of $(1-2\varepsilon)$.

Proof. First observe that every time the algorithm returns to Line 1, it must be the case that all the Dec-Matching$(H_i, \mu \cdot (1-\varepsilon), \varepsilon)$ returned no, which implies by Theorem 2(a) that for all $i \in [\lambda]$, $\mu(H_i) < (1-2\varepsilon) \cdot (1-\varepsilon) \cdot \mu$. This in turn implies that $\mu(G) < (1-2\varepsilon) \cdot \mu$. Suppose for contradiction that this is not the case, then from Lemma 3(b) we can conclude that there is a matching of size at least $(1-3\varepsilon) \cdot \mu$ in some H_i. More specifically, if $\mu(G) \geqslant (1-2\varepsilon) \cdot \mu$, then it satisfies the hypothesis of Lemma 3(b), and we know that $\mu(H_i) \geqslant (1-3\varepsilon) \cdot \mu$ for some $i \in [\lambda]$. This implies that Dec-Matching$(H_i, (1-\varepsilon) \cdot \mu, \varepsilon)$ wouldn't have terminated for some $i \in [\lambda]$ (by Theorem 2(a)). Consequently, Line 32 wouldn't have been executed, which is a contradiction. Thus, each time we execute Line 32, $\mu(G)$ has dropped from at least μ to at most $(1-2\varepsilon) \cdot \mu$. This proves our claim. □

From Claim 5, we can conclude that we return to Line 3 at most $\log_{1+\varepsilon} n$ times. Thus, to upper bound the runtime of Algorithm 1, it is sufficient to upper bound the runtime of each of the procedures. The runtime of the first procedure, which instantiates λ instances of Dec-Matching$(H_i, \mu \cdot (1-\varepsilon), \varepsilon)$ is dominated by the runtime of Vertex-Red(G, μ, ε), which is run once in the procedure, and the runtime of Static-Match(H_i, ε) which is run λ times (once per multi-graph H_i). Thus, the total time of this procedure is $O(m/\varepsilon \cdot \log 1/\varepsilon \cdot \lambda)$, which is $O(m/\varepsilon^5 \cdot \log n \cdot \log 1/\varepsilon)$.

Next, we upper bound the runtime of the second procedure. If an edge e is deleted from G, then the time to delete it from each of the multigraphs $H_1, \cdots, H_\lambda$ is λ. Finally, we are also maintaining at most λ active instances of $\mathcal{A}_i$, and their total update time is $O_\varepsilon(m \cdot \text{poly}(\log n))$. Consequently, we have that the total update time of Algorithm 1 is $O_\varepsilon(m \cdot \text{poly}(\log n))$. □

4 Algorithm for Theorem 2

We use $\mu(G, \kappa)$ to denote the value of the maximum fractional matching of G obeying the capacity function κ and the odd set constraints. To give an algorithm for Theorem 2, we need two ingredients. We

Algorithm 1

Input: Graph G and a parameter $\varepsilon > 0$

Output: A matching M with $|M| \geqslant (1 - 8\varepsilon) \cdot \mu(G)$

1: $M \leftarrow$ Static-Match(G, ε).
2: $\mu \leftarrow |M|$
3: **procedure** Instantiating Dec-Matching()
4: $i \leftarrow 1$.
5: cur$\leftarrow \lambda + 1$.
6: Let $H_1, H_2, \cdots, H_\lambda$ be the multi-graphs returned by Vertex-Red(G, μ, ε).
7: Label $H_1, \cdots, H_\lambda$ as active.
8: **for** $i \leqslant \lambda$ **do**
9: **if** $|$Static-Match$(H_i, \varepsilon)| < (1 - \varepsilon) \cdot \mu$ **then**
10: Label H_i as inactive.
11: **end if**
12: **end for**
13: **for** H_i labelled active **do**
14: Initialize Dec-Matching$(H_i, \mu \cdot (1 - \varepsilon), \varepsilon)$ (denoted $\mathcal{A}_i$).
15: Let M_i be the matching maintained by $\mathcal{A}_i$.
16: **end for**
17: Let cur be the least index i such that H_i is active.
18: **if** cur $> \lambda$ **then**
19: terminate.
20: **end if**
21: **end procedure**
22: **procedure** Handling deletion of edge e
23: **for** H_i labelled active **do**
24: Feed the deletion of e to $\mathcal{A}_i$.
25: **if** $\mathcal{A}_i$ returns no **then**
26: Label H_i as inactive.
27: **end if**
28: **end for**
29: **if** H_{cur} labelled active **then** ▷ Label may changed in Line 24.
30: Continue to output M_{cur}.
31: **else**
32: **if** cur $> \lambda$ **then**
33: Return to Line 1. ▷ Start over with new estimate for μ.
34: **else**
35: cur $\leftarrow$ cur $+ 1$
36: Goto Line 29.
37: **end if**
38: **end if**
39: **end procedure**

first maintain a balanced fractional matching of the multigraph. To do this, we will show an algorithm M-OR-E*() that given as input a capacitated graph G either outputs a fractional matching nearly obeying these capacities if $\mu(G,\kappa) \geqslant (1-\varepsilon)\cdot\mu(G)$, or if $\mu(G,\kappa) < (1-\varepsilon)\cdot\mu(G)$, it outputs a set of edges E^* along which capacities must be increased. When we say that the capacities are nearly obeyed, we mean that the can be exceeded only by a small factor. Then, we round the fractional matching using a known algorithm. We start with some definitions, and then go on to state the guarantees of the two algorithms, and show how it gives us an algorithm for Theorem 2.

Definition 6. Given a multi-graph G, for a pair of vertices u and v, define $D(u,v)$ to be the set of edges between u and v. Similarly, if e is an edge between u, v, then $D(e) \coloneqq D(u,v)$.

Definition 7. Let G be a multigraph with n vertices and m edges. Let κ be a capacity function on the edges. Suppose $\vec{x}$ is a fractional matching of G ($\vec{x}$ is a vector of length m). Then, we define $\vec{x}^C$ to be a vector of support size at most $\min\left\{m, \binom{n}{2}\right\}$, where for a pair of vertices, u and v, $x^C(u,v) \coloneqq \sum_{e\in D(u,v)} x(e)$. Essentially, if $\vec{x}$ is a fractional matching on a multigraph, then $\vec{x}^C$ is a fractional matching obtained by "collapsing" all edges together. Similarly, if $\vec{y}$ is a vector of support at most $\binom{n}{2}$, then we define $\vec{y}^D$ to be an m length vector such that for every $e \in E$ between a pair of vertices u and v, $y^D(e) \coloneqq \frac{y(u,v)\cdot\kappa(e)}{\kappa(D(e))}$ (D is for distributed, and we distribute the flow among the edges in proportion to their capacity).

Remark 8. Note that in doing the transformations in Definition 7, the support size of the transformed vector is always at most m. Thus, it doesn't negatively affect our runtime.

We now state our core ingredient, which either finds a balanced fractional matching of the multigraph G, or gives a set of edges E^* along which we can increase capacity.

Lemma 9. Let G be a multi-graph with $\mu(G) \geqslant \varepsilon\cdot n/16$. Let κ be a capacity function on the edges of G. There is an algorithm M-OR-E*(), that takes as input $G, \kappa, \varepsilon \in (0, 1/2)$ and $\mu \geqslant (1-\varepsilon)\cdot\mu(G)$ and in time $O(m\cdot\log n/\varepsilon)$ returns one of the following.

(a) A fractional matching $\vec{x}$ of value at least $(1-10\varepsilon)\cdot\mu$ with the following properties.

  (i) For any $e \in \mathrm{supp}(\vec{x})$ with $\kappa(D(e)) > 1/\alpha_\varepsilon^2$, $x(e) = \frac{\kappa(e)}{\kappa(D(e))}$, and $x(D(e)) = 1$.

  (ii) For any $e \in \mathrm{supp}(\vec{x})$ with $\kappa(D(e)) \leqslant 1/\alpha_\varepsilon^2$, $x(e) \leqslant \kappa(e)\cdot\alpha_\varepsilon$ and $x(D(e)) \leqslant \kappa(D(e))\cdot\alpha_\varepsilon$.

(b) A set E^* of edges such that $\kappa(E^*) = O(\mu\log n)$ such that for any integral matching M with $|M| \geqslant (1-3\varepsilon)\cdot\mu$, we have $|M\cap E^*| \geqslant \varepsilon\mu$. Moreover, $\kappa(e) < 1$ for all $e \in E^*$. Additionally, for every pair of vertices $u, v \in V$, either $D(u,v)\cap E^* = \emptyset$ or $D(u,v) \subseteq E^*$.

We give some intuition for Lemma 9. Recall that we need a balanced fractional matching that contains a large integral matching in its support. However, as mentioned before, in the case of general graphs, finding such a balanced fractional matching is not straightforward. In order to get past this obstacle, we define a balanced fractional matching that is easy to find and also avoids the integrality gap. We will explain how to find it in the subsequent sections. For now, we explain at a high-level, why the fractional matching $\vec{x}$ found by Lemma 9 avoids the integrality gap. Consider $\vec{x}^C$. Observe from Lemma 9(a), that for any pair of vertices $u, v \in G$, either $x^C((u,v)) = 1$ or $x^C((u,v)) \leqslant \varepsilon$. Thus, $\vec{x}$ satisfies odd-set constraints for all odd sets of size at most $1/\varepsilon$. By a folklore lemma, we can then argue that $\vec{x}$ contains an integral matching of size at least $(1+\varepsilon)^{-1}\cdot\sum_{u\neq v} x^C((u,v))$.

We will use M-OR-E*() as a subroutine in the algorithm DEC-MATCHING(). The fractional matching output by M-OR-E*() will have certain properties, we state these properties now, since it will be helpful in visualizing the fractional matching. We give a proof for these later on.

Property 10. We will use M-OR-E*$(G,\mu,\kappa,\varepsilon)$ as a subroutine in DEC-MATCHING(G,μ,ε) to get a matching $\vec{x}$ with the following properties.

(a) Each time M-OR-E*() returns the set E^*, we increase capacity along E^* by multiplying $\kappa(e)$ for each $e \in E^*$ by the same factor.

(b) Consider $u, v \in V$ and let e and e' be edges in between u, v. Then, $\kappa(e') = \kappa(e)$ at all times during the run of the algorithm.

The next property follows immediately from Lemma 9(a).

Property 11. Let $\vec{x}$ be the matching output by M-OR-E*$(G, \mu, \kappa, \varepsilon)$, then $x(e) \leqslant \kappa(e) \cdot \alpha_\varepsilon^2$ for all $e \in E$.

Definition 12. Let G be a multi-graph, let κ be a capacity function on the edges of G and let $\vec{x}$ be a fractional matching obeying κ. Let $\varepsilon \in (0, 1)$. Then, we split $\vec{x}$ into two parts, $\vec{x}^f$ and $\vec{x}^i$, where $\vec{x} = \vec{x}^f + \vec{x}^i$, and $\text{supp}(\vec{x}^f) = \{e \in E \mid \kappa(D(e)) < 1/\alpha_\varepsilon^2\}$ and $\text{supp}(\vec{x}^i) = \{e \in E \mid \kappa(D(e)) \geqslant 1/\alpha_\varepsilon^2\}$ (here, $\vec{x}^f$ stands for fractional and $\vec{x}^i$ stands for integral. Although the edges in $\vec{x}^i$ are not technically integral, they are large enough for us to round them).

We briefly state the implications of this definition. We do not use these properties in our proof, but it will be useful to state it nevertheless.

Property 13. Let G be any multigraph, and let $\vec{x}$ be a fractional matching of G. Then, for any pair of vertices u, v, either $D(u, v) \subseteq \text{supp}(\vec{x}^i)$ and $D(u, v) \cap \text{supp}(\vec{x}^f) = \emptyset$ or, $D(u, v) \subseteq \text{supp}(\vec{x}^f)$ and $D(u, v) \cap \text{supp}(\vec{x}^i) = \emptyset$.

Observation 14. Suppose $\vec{x}$ is a fractional matching returned by M-OR-E*() and consider $\vec{z} = \vec{x}^i$. Then $\text{supp}(\vec{z}^C)$ is a matching. This is implied by Lemma 9(a)(i).

The second ingredient is a method to round fractional matchings. This theorem is implicit in [Waj20], however, we prove it in Appendix D for completeness.

Lemma 15. [Waj20] Suppose G is an unweighted simple graph, let $\varepsilon \in (0, 1/2)$, and let $\vec{x}$ be a fractional matching of G such that $x(e) \leqslant \varepsilon^6$. Then, there is a dynamic algorithm SPARSIFICATION$(\vec{x}, \varepsilon)$, that has the following properties.

(a) The algorithm maintains a subgraph $H \subseteq \text{supp}(\vec{x})$ such that $|E(H)| = O_\varepsilon(\mu(\text{supp}(\vec{x})) \cdot \text{poly}(\log n))$, and with high probability $\mu(H) \geqslant (1 - \varepsilon) \cdot \sum_{e \in E} x(e)$.

(b) The algorithm handles the following updates to $\vec{x}$: the adversary can either remove an edge from $\text{supp}(\vec{x})$ or for any edge e, the adversary can reduce $x(e)$ to some new value $x(e) \geqslant 0$.

(c) The algorithm handles the above-mentioned updates in $\tilde{O}_\varepsilon(1)$ worst-case time.

We now show how M-OR-E*() and SPARSIFICATION() give us an algorithm for Theorem 2. This algorithm, called DEC-MATCHING() is stated in Algorithm 2. Before proving Theorem 2, we give a brief description of Algorithm 2.

Description of Dec-Matching(). The algorithm takes as input a multigraph H, a parameter ε, an estimate μ on the size of the maximum matching of H. It then instantiates the capacities of the graph H. The algorithm then starts a new phase by running M-OR-E*() on this capacitated graph. The algorithm returns either a large fractional matching or if the matching is small, then it outputs a set of edges E^*. The algorithm DEC-MATCHING() in the latter case, increases capacities along E^*. Suppose at some point the algorithm finds a large fractional matching $\vec{x}$. Recall that $\vec{x}$ output by M-OR-E*() is a matching with the following property: consider $\vec{x}^C$, then either $x^C((u, v)) = 1$ or $x^C((u, v)) \leqslant 1/\alpha_\varepsilon$ (this is guaranteed by Lemma 9(a)). The algorithm extracts the latter part, and sparsifies it using SPARSIFICATION(). Since the fractional matching input to SPARSIFICATION() has flow at most $1/\alpha_\varepsilon$ between any pair of vertices, it satisfies the hypothesis of Lemma 15 and obtain a graph S containing $\tilde{O}(\mu(H))$ edges and a large integral matching. Thus, STAT-MATCH(S, ε) is run and this matching is combined with the "integral" part of $\vec{x}$ and is output.

Algorithm 2 DEC-MATCHING(H, μ, ε)

1: For all $e \in E(H)$, $\kappa(e) \leftarrow 1/\alpha_\varepsilon^{\lceil \log_{\alpha_\varepsilon} n \rceil}$
2: **procedure** STARTING A NEW PHASE
3: $\quad \mu' \leftarrow$ STATIC-MATCH(H, ε) ▷ $\mu' \geqslant (1-\varepsilon)\cdot\mu(H)$
4: $\quad$ **if** $\mu' \leqslant (1-3\varepsilon)\cdot\mu$ **then**
5: $\quad\quad$ Return NO, and terminate.
6: $\quad$ **else**
7: $\quad\quad$ **while** M-OR-E*$(H, \kappa, \varepsilon, \mu')$ returns E^* **do**
8: $\quad\quad\quad$ For all $e \in E^*$, $\kappa(e) \leftarrow \kappa(e)\cdot\alpha_\varepsilon$
9: $\quad\quad$ **end while**
10: $\quad$ **end if**
11: **end procedure**
12: Let $\vec{x}$ be the matching returned by M-OR-E*(). ▷ This is a matching in a multigraph.
13: $\vec{y} \leftarrow \vec{x}^f$, $\vec{z} \leftarrow \vec{x}^i$ ▷ We will update $\vec{y}, \vec{z}$ as edges are deleted.
14: $S \leftarrow$ SPARSIFICATION$(\vec{y}^C, \varepsilon)$ ▷ Recall that Sparsification is dynamic
15: $M \leftarrow$ STATIC-MATCH$(S, \varepsilon) \cup \text{supp}(\vec{z}^C)$. ▷ The matching M that is output.
16: COUNTERM $\leftarrow 0$ ▷ Counter for deletions to the integral matching.
17: COUNTERX $\leftarrow 0$ ▷ Counter for deletions to the fractional matching.
18: **procedure** FOR DELETION OF EDGE e FROM H
19: $\quad$ **if** $e \in \text{supp}(\vec{x})$ **then**
20: $\quad\quad$ Delete e from $\text{supp}(\vec{x})$
21: $\quad\quad$ Update $\vec{z}^C$ accordingly. ▷ See Line 13.
22: $\quad\quad$ Update $\vec{y}^C$ accordingly; SPARSIFICATION from Line 14 then updates S.
23: $\quad\quad$ COUNTERX $\leftarrow$ COUNTERX $+\vec{x}(e)$.
24: $\quad$ **end if**
25: $\quad$ **if** COUNTERX $> \varepsilon\cdot\mu$ **then**
26: $\quad\quad$ Goto Line 3 ▷ End current phase and start new one.
27: $\quad$ **end if**
28: $\quad$ **if** $e \in M$ **then**
29: $\quad\quad$ Delete e from M
30: $\quad\quad$ COUNTERM $\leftarrow$ COUNTERM $+1$
31: $\quad$ **end if**
32: $\quad$ **if** COUNTERM $> \varepsilon\cdot\mu$ **then** ▷ The matching M is out of date
33: $\quad\quad$ COUNTERM $\leftarrow 0$
34: $\quad\quad$ $M \leftarrow$ STATIC-MATCH$(S, \varepsilon) \cup \text{supp}(\vec{z}^C)$ ▷ Recompute outputted matching M
35: $\quad$ **end if**
36: **end procedure**

The algorithm then begins a procedure to handle deletions. If an edge e is deleted, then it is removed from $\text{supp}(\vec{x})$, and SPARSIFICATION() algorithm updates S in $\tilde{O}_\varepsilon(1)$ worst case time (Lemma 15(c)). The algorithm also maintains counters for deletions to $\vec{x}$ and M. If the total value deleted from $\vec{x}$ exceeds $\varepsilon\cdot\mu$, then it ends the current phase, and starts a new one. On the other hand, if the number of edges deleted from M exceeds $\varepsilon\cdot\mu$, then, the algorithm recomputes M by running STATIC-MATCH(S, ε) again (since we know that $\vec{x}$ still contains a large integral matching in its support).

We also state the following lemma which we prove later in Section 5.

Lemma 16. The subroutine M-OR-E*() and Line 3 are called at most $O(\alpha_\varepsilon^3 \log n)$ times in Algorithm 2 during the course of deletion of m edges.

We restate our theorem and give a proof.

Theorem 2. We are given an unweighted multi-graph $G_0 = (V, E_0)$. Let $|V| = n$, and $\varepsilon \in (0, 1/2)$. Suppose $\mu(G_0) \geqslant \varepsilon\cdot n/100$ and $\mu \geqslant (1-\varepsilon)\cdot\mu(G)$. Suppose G is subject to adversarial deletions. There is

an algorithm Dec-Matching(G, μ, ε), which processes deletions in $\tilde{O}_\varepsilon(m)$ total update time and has the following guarantees.

(a) When the algorithm terminates, $\mu(G) < (1-2\varepsilon)\cdot\mu$. Upon termination the algorithm outputs "no".

(b) Until the algorithm terminates, it maintains an integral matching with size at least $(1-20\varepsilon)\cdot\mu$.

Proof of Theorem 2. We first argue that Algorithm 2 satisfies the properties of Theorem 2. First observe that when the algorithm executes Line 5, $\mu' \leqslant (1-3\varepsilon)\cdot\mu$. Since $\mu' \geqslant (1-\varepsilon)\cdot\mu(H)$, this implies that $\mu(H) \leqslant (1-2\varepsilon)\cdot\mu$. Thus, when the algorithm terminates, $\mu(H) \leqslant (1-2\varepsilon)\cdot\mu$, which proves Theorem 2(a).

Next, we want to argue that while the algorithm does not terminate, it outputs an integral matching of size at least $\mu\cdot(1-20\varepsilon)$. This corresponds to Theorem 2(b). We first argue that the algorithm indeed outputs matching. Note that the output matching M is the union of $\text{supp}(\vec{z}^C)$ and the output of Sparsification$(\vec{y}^C, \varepsilon)$. Note that M is a union of two matchings, is implied by Observation 14 and Lemma 15. Moreover, these matchings are vertex disjoint and this follows from Property 13. Thus, M is a matching.

Next,we want to argue that $|M| \geqslant (1-20\varepsilon)\cdot\mu$. First, observe that the fractional matching output by M-or-E*() has value at least $(1-10\varepsilon)\cdot\mu$. Next, observe that because of Lemma 9(a)(ii), the matching output by M-or-E*() satisfies the conditions of Lemma 15: that is, $\vec{y}^C$ has the property that the flow through any edge in the support, $\vec{y}^C(e) \leqslant 1/\alpha_\varepsilon \leqslant \varepsilon^3$. Thus, $\mu(S) \geqslant (1-\varepsilon)\cdot\sum_{e\in\text{supp}(\vec{y}^C)} y^C(e)$. Therefore, the matching output by Sparsification$(\vec{y}^C, \varepsilon)$ is an integral matching of size at least $(1-2\varepsilon)\cdot\sum_{e\in\text{supp}(\vec{y}^C)}\vec{y}^C(e)$. Thus, we can conclude,

$$\begin{aligned}|M| &\geqslant \sum_{e\in\text{supp}(\vec{x}^i)} x(e) + (1-2\varepsilon)\cdot\sum_{e\in\text{supp}(\vec{x}^f)} x(e) \geqslant (1-2\varepsilon)\cdot\sum_{e\in\text{supp}(\vec{x})} x(e)\\ &\geqslant (1-2\varepsilon)\cdot(1-10\varepsilon)\cdot\mu \geqslant (1-12\varepsilon)\cdot\mu.\end{aligned}$$

Thus, the algorithm maintains an integral matching of size at least $(1-20\varepsilon)\cdot\mu$ at all times.

Running Time: We now bound the runtime of the algorithm. Let us first consider the time to initialize a phase. From Lemma 16 we know that M-or-E*() is called $O(\alpha_\varepsilon^3 \log n)$ times, so this is also an upper bound on the number of phases. Thus, Static-Match() in Lines 3 and 15 is also called at most $O(\alpha_\varepsilon^3 \log n)$ times, since these are only called once per phase. All of these subroutines have runtime $\tilde{O}_\varepsilon(m)$, so since each is executed $\tilde{O}_\varepsilon(1)$ times, the total runtime of all of them is $\tilde{O}_\varepsilon(m)$.

We also need to bound the runtime of Sparsification$(\vec{y}^C, \varepsilon)$ and the total contribution of Static-Match() in Line 34. Let us start by bounding the runtime of Sparsification(). The time to initialize this procedure in Line 14 is $\tilde{O}(m)$, and this is only done once per phase, so again the total time spent on initialization is $\tilde{O}(m)$. We also spend time updating the output sparsifier S in Line 22. The worst-case runtime time of Sparsification() is $O(\log n)$ per update to $\vec{x}$ (see Lemma 15(b) and (c)). But each update to $\vec{x}$ corresponds to an adversarial deletion of some edge e, so there at most m adversarial deletions, so the total overall time (among all phases) to update Sparsification is $\tilde{O}(m)$.

Finally, we want to count the contribution of Static-Match() in Line 34. Observe that by Lemma 15, the subgraph S has size $\tilde{O}(\mu)$. Thus, the runtime of Static-Match(S, ε) is $\tilde{O}(\mu/\varepsilon)$. Now, observe that Line 34 is only executed when CounterM has increased from 0 to $\varepsilon\cdot\mu$, and the counter increases by at most 1 per adversarial deletion, so there are at least $\varepsilon\cdot\mu$ adversarial deletions per execution of Line 34. Thus, Stat-Match(S, ε) is called at most $m/\varepsilon\cdot\mu$ times in Line 34, and the total contribution here is $O(m/\varepsilon^2)$.

□

5 Proof of Lemma 16

To prove Lemma 16, which gives an upper bound on the number of times the subroutine M-OR-E*() is called in Algorithm 2, we will follow the framework of [BGS20]. We will define a potential function, and show that it increases as κ is increased, and edges are deleted. While their framework applies to simple graphs, we verify that it is possible to extend it to multigraphs as well. Towards this, we give a few definitions.

Definition 17. Let G be a multigraph, and let κ be the capacity function on the edges of the graph (if the adversary deletes an edge e, then let $\kappa(e)$ be the capacity of the edge right before it is deleted). Let μ be the input to DEC-MATCHING(), when it is run on G, ε. Let $\mathcal{M}$ be the set of all integral matchings of G of size at least $(1-3\varepsilon)\cdot\mu$. Define the cost of an edge e to be $c(e) = \log(n\cdot\kappa(e))$. For any integral matching M, define $c(M) = \sum_{e\in M} c(e)$. Define $\Pi(G,\kappa) = \min_{M\in\mathcal{M}} c(M)$. If $\mathcal{M} = \emptyset$, then $\Pi(G,\kappa) = \infty$.

Observation 18. Initially, $\kappa(e) = 1/\alpha_\varepsilon^{\lceil \log_{\alpha_\varepsilon} n\rceil}$, so $\Pi(G,\kappa) = 0$. Note that capacities κ only change (increase) in Line 8. Consequently, the capacity function κ is monotonically increasing. Moreover, edges of G are only deleted. Thus, $\Pi(G,\kappa)$ is monotonically increasing as well.

Observation 19. When $\Pi(G,\kappa) = \infty$, then Line 5 of Algorithm 2 causes the algorithm to terminate.

Proof. Suppose Line 5 doesn't cause Algorithm 1 to terminate, then, $\mu' \geqslant (1-3\varepsilon)\cdot\mu$. Thus, $\mu(G) \geqslant \mu' \geqslant (1-3\varepsilon)\cdot\mu$, and $\mathcal{M} \neq \emptyset$. □

Lemma 20. In DEC-MATCHING(), we say that we begin a new phase when $\varepsilon\cdot\mu$ value of the fractional matching $\vec{x}$ has been deleted (that is, the value of COUNTERX has increased to $\varepsilon\cdot\mu$). Suppose we are in a phase when the algorithm does not terminate. Let κ be the final capacities right before we process deletions. Then, $\Pi(G,\kappa) = O(\mu\log n)$.

Proof. First observe that for any edge e, $\kappa(e) \leqslant 1$, so $c(e) = O(\log n)$. This is implied by the fact that E^* only consists of edges that have $\kappa(e) < 1$ and $\kappa(e)$ is a power of α_ε (see Lemma 9(b)). This is because, we start with $\kappa(e)$ to be a power of α_ε initially (see Line 1), and each time we increase $\kappa(e)$, we multiply it by α_ε. Moreover, for any matching M, we know that $|M| \leqslant \mu\cdot(1+\varepsilon)$, since, $\mu \geqslant \mu(G)\cdot(1-\varepsilon)$. Thus, $c(M) \leqslant (1+\varepsilon)\cdot\mu\cdot\log n$. This implies that $\Pi(G,\kappa) = O(\mu\log n)$. □

Definition 21. Let E_0 be the initial set of edges and let $\kappa(E_0) = \sum_{e\in E_0}\kappa(e)$.

Lemma 22. Suppose a call to M-OR-E*$(G,\mu,\kappa,\varepsilon)$ returns the set E^* instead of a matching. Let κ' denote the new edge capacities after increasing capacities along E^*. Then,

(a) $\kappa'(E_0) \leqslant \kappa(E_0) + \alpha_\varepsilon\cdot\mu\cdot\log n$, and

(b) $\Pi(G,\kappa') \geqslant \Pi(G,\kappa) + \mu/\varepsilon$.

Proof. We begin by recalling Lemma 9(b) that $\kappa(E^*) \leqslant \mu\log n$. Thus, $\kappa'(E^*) = O(\alpha_\varepsilon\cdot\mu\cdot\log n)$, since it is obtained by scaling up $\kappa(E^*)$ by a α_ε. Thus, we have,

$$\kappa'(E_0) = \kappa(E_0\setminus E^*) + \kappa'(E^*)$$

This implies that:

$$\kappa'(E_0) - \kappa(E_0\setminus E^*) = \kappa'(E^*) = \alpha_\varepsilon\cdot\mu\cdot\log n$$

The LHS is lower bounded by $\kappa'(E_0) - \kappa(E_0)$, since $\kappa(E_0) \geqslant \kappa(E_0\setminus E^*)$. This proves the first part of our claim. To prove the second part, first notice that E^* contains at least $\varepsilon\cdot\mu$ edges of every matching of size at least $(1-3\varepsilon)\cdot\mu$ (implied by Lemma 9(b)). Let $M, M' \in \mathcal{M}$ be the matchings that minimize $\Pi(G,\kappa)$ and $\Pi(G,\kappa')$ respectively, and let c and c' be the cost functions associated with κ and κ' respectively. Observe that $|M| \geqslant (1-3\varepsilon)\cdot\mu$ and $|M'| \geqslant (1-3\varepsilon)\cdot\mu$ (by definition of $\mathcal{M}$, see Definition 17). Then, we have the following.

$$\Pi(G,\kappa') = c'(M') = \sum_{e\in M'}\log(n\cdot\kappa'(e)) = \sum_{e\in M'\setminus E^*}\log(n\cdot\kappa(e)) + \sum_{e\in M'\cap E^*}\log(n\cdot\kappa(e)\cdot\alpha_\varepsilon)$$

$$= \sum_{e \in M'} \log(n \cdot \kappa(e)) + |M' \cap E^*| \cdot \log \alpha_\varepsilon$$
$$= c(M') + \mu/\varepsilon \geqslant \Pi(G, \kappa) + \mu/\varepsilon$$
(Since $|M' \cap E^*| \geqslant \varepsilon \cdot \mu$ and $\alpha_\varepsilon = 2^{32/\varepsilon^2}$)

This proves our claim. □

Observation 23. The total number of calls to M-OR-E*() (till DEC-MATCHING() terminates) that return E^* are upper bounded by $O(\varepsilon \cdot \log n)$. This is because the potential function $\Pi(G, \kappa)$ is upper bounded by $O(\mu \log n)$, and each call to M-OR-E*() that returns E^*, increments $\Pi(G, \kappa)$ by μ/ε (see Lemma 22(b)). Thus, we can deduce that $\kappa(E_0) \leqslant \alpha_\varepsilon \cdot \mu \cdot \varepsilon \cdot \log n$. This is because each call to M-OR-E*() that returns E^* increases $\kappa(E_0)$ by at most $\alpha_\varepsilon \cdot \mu \cdot \log n$ (see Lemma 22(a)), and there are at most $\varepsilon \cdot \log n$ such calls.

We now want to upper bound the number of calls made to M-OR-E*() that return a matching. This is upper bounded by the number of phases.

Lemma 24. The total number of phases is at most $O(\alpha_\varepsilon^3 \cdot \log n)$.

Proof. We define $\Phi_{\mathsf{del}} \coloneqq \sum_{e \in E_{\mathsf{del}}} \kappa(e)$ to be the capacity of deleted edges. Observe that $\Phi_{\mathsf{del}} \leqslant \kappa(E_0) \leqslant \alpha_\varepsilon \cdot \varepsilon \cdot \mu \cdot \log n$. This is implied by the definition of $\kappa(E_0)$ and Observation 23. Moreover, each time we start a new phase, at least $\varepsilon \cdot \mu$ value of the fractional matching has been deleted. That is $\sum_{e \in E} x(e)$ has dropped by $\varepsilon \cdot \mu$. From Property 11, one can conclude that for any edge e, $x(e) \leqslant \alpha_\varepsilon^2 \cdot \kappa(e)$. This implies that a phase contributes at least $\varepsilon \cdot \mu / \alpha_\varepsilon^2$ to Φ_{del}. Using the fact that Φ_{del} is upper bounded by $\alpha_\varepsilon \cdot \varepsilon \cdot \mu \cdot \log n$, we can conclude that the total number of phases is at most $\alpha_\varepsilon^3 \cdot \log n$. □

Proof of Lemma 16. From Lemma 24 and Observation 23, we conclude that the total number of calls to M-OR-E*() are upper bounded by $O(\alpha_\varepsilon^3 \cdot \log n)$. Finally, the number of calls to STATIC-MATCH() due to Line 3 are upper bounded by the number of phases, which are at most $O(\alpha_\varepsilon^3 \cdot \log n)$ by Lemma 24. □

6 Ingredients for Algorithm M-or-E*()

Recall that we use $\mu(G, \kappa)$ to denote the value of the maximum fractional matching of G obeying capacity function κ and the odd set constraints. As in the congestion balancing setup of [BGS20], we want to check if $\mu(G, \kappa) \geqslant (1 - \varepsilon) \cdot \mu(G)$. However, unlike in bipartite graphs, where we can use flows to find fractional matching, there is no simple way to check if $\mu(G, \kappa) \geqslant (1 - \varepsilon) \cdot \mu(G)$ in general graphs. Our first structural result circumvents this issue. Let G_s be the graph obtained by sampling every edge e with probability $p(e) = \min\{1, \kappa(e) \cdot \rho_\varepsilon\}$. We show that $\mu(G_s) \geqslant \mu(G, \kappa) - \varepsilon \cdot \mu(G)$. Thus, we can run STATIC-MATCH(G_s, ε) to estimate $\mu(G, \kappa)$.

At a high level, M-OR-E*() proceeds in three phases. In Phase 1, it creates G_s and computes $\mu(G_s)$. If this matching is large, it proceeds to Phase 2, where it finds a fractional matching $\vec{x}$ such that $\sum_{e \in E} x(e) \geqslant (1 - \varepsilon) \cdot \mu(G)$. On the other hand, if $\mu(G_s)$ is small, then it proceeds to Phase 3, where it finds the set of edges E^* along which it increases capacity. In the subsequent sections, we will state the main structural properties we use in each of the phases. Finally, in Section 5, we put together these ingredients to give M-OR-E*(), and prove Lemma 9.

6.1 Phase 1 of M-or-E*()

Before we formally state the main guarantees of Phase 1, we will state some standard results in matching theory, that we will use in our main result for Phase 1.

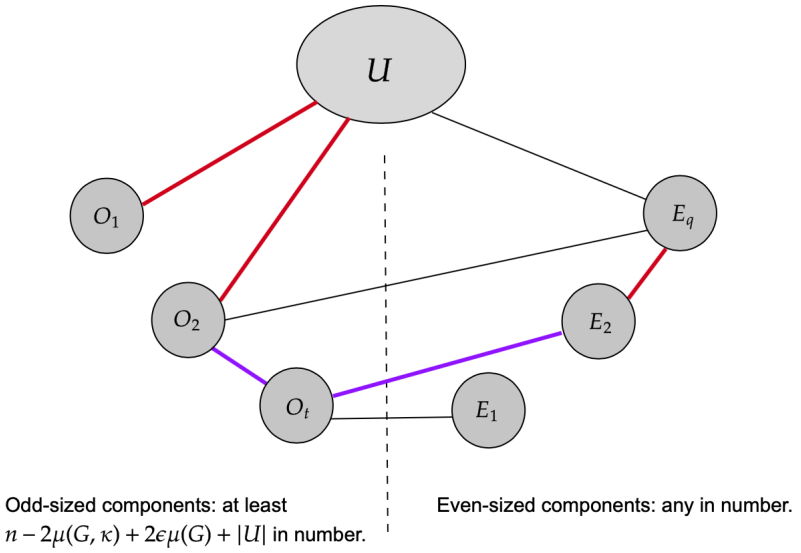

Figure 1: The figure shows the graph G, and a partition $\mathcal{P} = \{U, E_1, \cdots, E_q, O_1, \cdots, O_t\}$ satisfying (a) and (b). The thick edges (in red and purple), are the edges in $\text{supp}(\vec{x})$, where $\vec{x}$ is the fractional matching realizing $\mu(G,\kappa)$. The purple edges (edges between the odd components, or between odd and even components) correspond to $E^{\mathcal{P}}_{\text{miss}}$, and $\sum_{e \in E^{\mathcal{P}}_{\text{miss}}} x(e) \geqslant 2 \cdot \varepsilon \cdot \mu(G)$.

6.1.1 Some Standard Ingredients For Phase 1

The first ingredient we use is the Tutte-Berge formula.

Definition 25. Let G be any graph (possibly containing multiedges), and let $U \subseteq V$, then $\text{odd}_G(V \setminus U)$ refers to the number of odd components in $G[V \setminus U]$.

Lemma 26 (Tutte-Berge Formula)**.** [Sch03] The size of a maximum matching in a graph $G = (V, E)$ is equal to $\frac{1}{2} \min_{U \subseteq V} (|U| + |V| - \text{odd}_G(V \setminus U))$.

Additionally, we will use some properties of the matching polytope.

Lemma 27. [Sch03] Let G be any graph, and let $\vec{x}$ be a fractional matching that in addition to the fractional matching constraints, also satisfies the following for all odd-sized $U \subseteq V$: $\sum_{e \in G[U]} x(e) \leqslant \frac{|U|-1}{2}$. Then, there is an integral matching $M \subseteq \text{supp}(x)$ with $|M| = \sum_{e \in E} x(e)$.

Definition 28. Let G be any multigraph, and let $S, T \subseteq V$, then $\delta_G(S,T)$ is defined as the set of edges that have one endpoint in S and the other in T. Additionally, for $S \subseteq V$, we define $\delta_G(S)$ to be the set of edges that have one end point in S, and the other in $V \setminus S$.

6.1.2 Main Lemma for Phase 1

As mentioned earlier, in Phase 1 of M-OR-E*(), we first create a sampled graph G_s. In the following lemma, we show that $\mu(G_s)$ is a good estimate for $\mu(G,\kappa)$ with high probability.

Lemma 29. Let G be a multigraph with $\mu(G) \geqslant \varepsilon \cdot n/16$ where $\varepsilon \in (0, 1/2)$. Let κ be a capacity function on the edges of G, and let G_s be obtained by sampling every edge $e \in G$ with probability $p(e) = \min\{\kappa(e) \cdot \rho_\varepsilon, 1\}$. Let $\mu(G,\kappa)$ be the value of the maximum fractional matching of G obeying the capacities κ, and the odd set constraints. Then, with high probability, $\mu(G_s) \geqslant \mu(G,\kappa) - \varepsilon \cdot \mu(G)$.

Proof. We want to show that with probability at least $1 - 1/n^2$, $\mu(G_s) \geqslant \mu(G,\kappa) - \varepsilon \cdot \mu(G)$. In order to do this, by Lemma 27, it is sufficient to show that with probability at least $1 - 1/n^2$, $\frac{1}{2} \min_{U \subseteq V} (|U| + |V| - \text{odd}_{G_s}(V - U)) \geqslant \mu(G,\kappa) - \varepsilon \cdot \mu(G)$.

Towards this, we consider a fixed partition $\mathcal{P}$ of V into sets $U, O_1, \cdots, O_t, E_1, \cdots, E_q$ with the following properties (see Figure 1).

(a) We have, $q \geqslant 0$ and $t > n - 2 \cdot \mu(G, \kappa) + 2\varepsilon \cdot \mu(G) + |U|$.

(b) Sets O_i for $i \in [t]$ are odd-sized sets and sets E_l for $l \in [q]$ are even-sized sets.

Note that if $\mu(G_s) < \mu(G, \kappa) - \varepsilon \cdot \mu(G)$, then there is a partition $\mathcal{P} = \{U, O_1, \cdots, O_t, E_1, \cdots, E_q\}$ of G_s, satisfying (a) and (b) such that $V \setminus U$ is the union of components $O_1, \cdots, O_t, E_1, \cdots, E_q$. Note that if $V \setminus U$ is the union of disjoint components $O_1, \cdots, O_t, E_1, \cdots, E_q$ then $\delta_{G_s}(O_i, O_l) = \emptyset$ for all $i \neq l$ and $\delta_{G_s}(O_i, E_l) = \emptyset$ for all $i \neq l$ (this is evident from Lemma 26). Thus, to upper bound the probability that $\mu(G_s) < \mu(G, \kappa) - \varepsilon \cdot \mu(G)$, it is sufficient to upper bound the probability that for all partitions $\mathcal{P} = \{U, O_1, \cdots, O_t, E_1, \cdots, E_q\}$ satisfying (a) and (b), *none* of the edges $E^{\mathcal{P}}_{\mathsf{miss}} = \{e \mid e \in \delta_G(O_i, O_l) \text{ for } i \neq l\} \cup \{e \mid e \in \delta_G(O_i, E_l) \text{ for } i \in [t], l \in [q]\}$ are sampled in G_s. In order to bound this probability, we make the following claim.

Claim 30. For a partition $\mathcal{P}$ satisfying (a) and (b), $\kappa(E^{\mathcal{P}}_{\mathsf{miss}}) \geqslant 2 \cdot \varepsilon \cdot \mu(G)$.

Proof. Let $\vec{x}$ be a fractional matching obeying odd set constraints and capacity function κ such that $\sum_{e \in E} x(e) = \mu(G, \kappa)$. In order to prove this claim, we show that if $\kappa(E^{\mathcal{P}}_{\mathsf{miss}}) < 2 \cdot \varepsilon \cdot \mu(G)$, then, $x(E^{\mathcal{P}}_{\mathsf{miss}}) > \kappa(E^{\mathcal{P}}_{\mathsf{miss}})$, which will contradict the fact that $\vec{x}$ is a fractional matching obeying κ.

With this proof strategy in mind, for contradiction assume that $\kappa(E^{\mathcal{P}}_{\mathsf{miss}}) < 2 \cdot \varepsilon \cdot \mu(G) \leqslant n - 2 \cdot \mu(G, \kappa) + 2 \cdot \varepsilon \cdot \mu(G)$. The last inequality follows from the fact that $\mu(G, \kappa)$ corresponds to the value of the maximum fractional matching, so, $\mu(G, \kappa) \leqslant \frac{n}{2}$. Since $\vec{x}$ obeys odd set constraints, $\sum_{l \leqslant t} x(\delta_G(O_l)) \geqslant t$ (from Lemma 27). Note that $\sum_{l \leqslant t} x(\delta_G(O_l, U)) \leqslant |U|$, otherwise for some $v \in U$, $\sum_{e \ni v} x(e) > 1$, violating the fact that $\vec{x}$ is a fractional matching. Next, we observe that $\sum_{l \leqslant t} x(\delta_G(O_l)) = x(E^{\mathcal{P}}_{\mathsf{miss}}) + \sum_{l \leqslant t} x(\delta_G(O_l, U))$. This follows from the fact that all edges emanating out of O_i in G, are either incident on other O_j, or E_k or U. We have the following set of inequalities.

$$\begin{aligned} x(E^{\mathcal{P}}_{\mathsf{miss}}) &= \sum_{l \leqslant t} x(\delta_G(O_l)) - \sum_{l \leqslant t} x(\delta_G(O_l, U)) \\ &\geqslant t - |U| \\ &> n - 2 \cdot \mu(G, \kappa) + 2 \cdot \varepsilon \cdot \mu(G) + |U| - |U| \end{aligned}$$

Thus, that $x(E^{\mathcal{P}}_{\mathsf{miss}}) \geqslant n - 2\mu(G, \kappa) + 2\varepsilon\mu(G) > \kappa(E^{\mathcal{P}}_{\mathsf{miss}})$. This is a contradiction. This concludes our proof, and we know that $\kappa(E^{\mathcal{P}}_{\mathsf{miss}}) \leqslant 2 \cdot \varepsilon \cdot \mu(G)$. □

Additionally, we have the following claim, which allows us to only focus on $\mathcal{P}$ for which all $e \in E^{\mathcal{P}}_{\mathsf{miss}}$ have $\kappa(e) < 1/\rho_\varepsilon$.

Observation 31. Suppose $\kappa(e) \geqslant 1/\rho_\varepsilon$ for any $e \in E^{\mathcal{P}}_{\mathsf{miss}}$, then,

$$\Pr\left(\text{None of the edges in } E^{\mathcal{P}}_{\mathsf{miss}} \text{ are sampled in } G_s\right) = 0.$$

The above observation follows from the fact that an edge e is sampled with probability $\min\{1, \kappa(e) \cdot \rho_\varepsilon\}$. From the above claim, we can deduce that if for any partition $\mathcal{P}$, if $E^{\mathcal{P}}_{\mathsf{miss}}$ has an edge e with $\kappa(e) \geqslant 1/\rho_\varepsilon$, then the contribution of $E^{\mathcal{P}}_{\mathsf{miss}}$ to our probability bound will be 0. Hence, it is sufficient to focus on $E^{\mathcal{P}}_{\mathsf{miss}}$ where all edges e have $\kappa(e) < 1/\rho_\varepsilon$. We now bound the following probability.

$$\begin{aligned} \Pr\left(\text{None of the edges in } E^{\mathcal{P}}_{\mathsf{miss}} \text{ are sampled in } G_s\right) &\leqslant \prod_{e \in E^{\mathcal{P}}_{\mathsf{miss}}} (1 - p(e)) \\ &\leqslant \exp\left(-\sum_{e \in E^{\mathcal{P}}_{\mathsf{miss}}} p(e)\right) \end{aligned}$$

$$= \exp\left(-\sum_{e \in E^{\mathcal{P}}_{\text{miss}}} \kappa(e) \cdot \rho_\varepsilon\right)$$

(Since $p(e) = \kappa(e) \cdot \rho_\varepsilon$ (from Claim 31 and discussion above))

$$= \exp\left(-\varepsilon \cdot 2^{16/\varepsilon^2} \cdot \mu(G) \cdot \log n\right)$$

(From Claim 30 and the fact that $\rho_\varepsilon = 2^{16/\varepsilon^2} \cdot \log n$)

$$\leqslant \exp\left(-2^{15/\varepsilon^2} \cdot \mu(G) \cdot \log n\right).$$

Note that the total number of partitions of graph G are upper bounded by $2^{n \cdot \log n}$. Hence, this also upper bounds the total number of partitions of G satisfying (a) and (b). Since $n \leqslant 16 \cdot \mu(G)/\varepsilon$ and $\varepsilon < 1/2$, the bound on the number of partitions is at most $2^{16\mu/\varepsilon \cdot \log n}$ (by assumption), taking a union bound over all the partitions, we know that the with probability at least $1 - \exp\left(-\mu(G) \cdot \log n/\varepsilon\right)$, in G_s, we have no partition $\mathcal{P} = \{U, O_1, \cdots, O_t, E_1, \cdots, E_q\}$ satisfying (a) and (b). Thus, by Lemma 26, we have that with high probability, $\mu(G_s) \geqslant \mu(G, \kappa) - \varepsilon \cdot \mu(G)$. □

6.2 Phase 2 of M-or-E*()

The algorithm proceeds to Phase 2 only if the integral matching M_s found in G_s is close to $\mu(G, \kappa)$. Recall that our goal is to compute a fractional matching so that we can apply the congestion balancing framework. However, as mentioned before there is no straightforward way of computing a maximum fractional matching in a general graph obeying capacity κ. To overcome this, we give a candidate fractional matching which is easy to compute, and is sufficient for our purposes (that is, it avoids the integrality gap). At a high level, this is what Phase 2 does, and in this section we describe our candidate fractional matching and show that it is close in value to $\mu(G_s)$ with high probability, and therefore it is close to $\mu(G, \kappa)$ as well (by Lemma 29).

6.2.1 Preliminaries for Phase 2

Phase 2 starts by computing $M_s = \textsc{Static-Match}(G_s, \varepsilon)$ and then uses M_s to compute the desired fractional matching. We will split M_s into low capacity edges and high capacity edges, and as a result split V into vertices matched using high capacity edges, and low capacity edges. We begin by giving a formal definition of low capacity edges.

Definition 32. Let G be any multigraph, and let κ be a capacity function on the edges of G. Let $\varepsilon \in (0, 1/2)$. Define $E_L(G, \kappa) = \{e \in E \mid e \in D(u, v) \text{ and } \kappa(D(u, v)) \leqslant 1/\alpha_\varepsilon^2\}$. Intuitively, $E_L(G, \kappa)$ is the set of low total capacity edges.

As mentioned in the high-level overview, in order to prove our probabilistic claims, we will give some slack to the capacities. This motivates our next definition.

Definition 33. Let G be a multi-graph, and let κ be a capacity function on the edges of G. Let $\varepsilon \in (0, 1/2)$. We define the capacity function κ^+ as follows.

(a) For all $e \in E_L(G, \kappa)$, $\kappa^+(e) = \kappa(e) \cdot \alpha_\varepsilon$.

(b) For all $e \in E \setminus E_L(G, \kappa)$, $\kappa^+(e) = \kappa(e)$.

To make our analysis easier to follow, we need the following definition of a bipartite double cover of G.

Definition 34. Let G be a multi-graph and let κ be a capacity function on the edges of G. We define the bipartite double cover $\textsc{bc}(G)$ to be a bipartite graph with capacity function $\kappa_{\textsc{bc}}$ as follows.

(a) For every vertex $v \in V(G)$, make two copies v and v' in $V(\textsc{bc}(G))$.

(b) If e is an edge between $u, v \in V(G)$, then for each such e we add two edges e' and e'', one between u and v' and the other between v and u'. We let $\kappa_{\textsc{bc}}(e') = \kappa_{\textsc{bc}}(e'') = \kappa(e)$.

We have the following standard claim relating $\mu(G)$ and $\mu(\text{BC}(G))$.

Claim 35. For any multi-graph G, $\mu(\text{BC}(G)) \geqslant 2 \cdot \mu(G)$.

Next, we state the following lemma, which follows from standard techniques, and we give a formal proof of it in Appendix A. The lemma essentially states that a fractional matching which has low flow on all edges has a very small integrality gap.

Lemma 36. Let G be a multigraph, and let $\varepsilon \in (0,1)$. Let κ be a capacity function on the edges of G, with $\kappa(D(e)) \leqslant 1/\alpha_\varepsilon$ for all $e \in E(G)$. Then, $\mu(\text{BC}(G), \kappa_{\text{BC}}) \leqslant 2 \cdot (1+\varepsilon) \cdot \mu(G, \kappa)$, where $\mu(G, \kappa)$ is the maximum fractional matching of G obeying κ and the odd set constraints, and $\mu(\text{BC}(G), \kappa_{\text{BC}})$ is the maximum fractional matching of $\text{BC}(G)$ obeying κ_{BC}.

Additionally, we will need the following version of Hall's theorem, which follows as a corollary of Lemma 26.

Proposition 37 (Extended Hall's Theorem)**.** Let $G = (L \cup R, E)$ be a bipartite graph with $n = |L| = |R|$, then $\mu(G) = n - \max_{S \subseteq L}(|S| - |N_G(S)|)$, where $N_G(S)$ refers to the neighbourhood of S in G.

In order to prove Lemma 40, we will use the following version of Chernoff bound.

Lemma 38 (Chernoff Bound)**.** Let $X_1, \cdots, X_k$ be negtively correlated random variables, and let X denote their sum, and let $\mu = \mathbb{E}[X]$. Suppose $\mu_{\min} \leqslant \mu \leqslant \mu_{\max}$, then for all $\delta > 0$, $\Pr(X \geqslant (1+\delta)\mu_{\max}) \leqslant \left(\frac{e^\delta}{(1+\delta)^\delta}\right)^{\mu_{\max}}$.

Additionally, we state the following observation, we refer the readers to [BGS20] for a proof of this observation. The proof follows from a standard application of the max-flow min-cut theorem, and we refer the reader to [BGS20] for a proof sketch.

Observation 39. Let G be any bipartite multigraph, with vertex bipartitions S and T. Let κ be the capacity function on the edges of the graph. Then, for any $C \subseteq S$, and $D \subseteq T$, we have, $|S| - |C| + |D| + \kappa(C, T \setminus D) \geqslant \mu(G, \kappa)$. Moreover, there are sets $C \subseteq S$ and $D \subseteq T$ such that equality holds.

6.2.2 Main Result for Phase 2

We briefly give some intuition about the statement of our claim. Recall in the high level review, we mentioned that M, the integral matching of G_s can split into two parts M_H, which is the high capacity part, and M_L, the low capacity part, and we defined $V_H = V(M_H)$ and $V_L = V(M_L)$. We said that congestion balancing allows us to give slack to capacities, and therefore, we can round up the capacities of M_H to 1. However, we still want to compute a fractional matching $G[V_L]$. In order to do this, we observe that if the fractional matching in $G[V_L]$ is only on low capacity edges, then we can use flow algorithms to compute such a matching. Therefore, our main structural result for Phase 2 states that if $\vec{y}$ is a fractional matching on the low capacity edges of $G[V_L]$, then with high probability $\sum_{e \in E} y(e) \geqslant |M_L| - \varepsilon \cdot \mu(G)$. We now state this result formally.

Lemma 40. Let G be a multigraph and let $\varepsilon \in (0, 1/2)$ and suppose $\mu(G) \geqslant \varepsilon \cdot n/16$. Let κ be a capacity function on the edges, and let G_s be the graph obtained from G by sampling each edge e with probability $p(e) = \kappa(e) \cdot \rho_\varepsilon$. Let $E_L \coloneqq E_L(G, \kappa)$. Then, for all $W \subseteq V$, we have, with high probability,

$$\mu(\text{BC}(G_s[W] \cap E_L)) \leqslant \mu(\text{BC}(G[W] \cap E_L), \kappa^+_{\text{BC}}) + \varepsilon\mu(G).$$

Remark 41. Note that by definition of E_L, all edges $e \in G[W] \cap E_L$ have $\kappa(e) \leqslant 1/\alpha^2_\varepsilon$. Thus, $\kappa^+_{\text{BC}}(e) = \kappa_{\text{BC}}(e) \cdot \alpha_\varepsilon$ for all $e \in \text{BC}(G[W] \cap E_L)$.

Before we prove it, we have the following statement as a corollary of Lemma 40.

Corollary 42. Let G be a multi-graph, and let $\varepsilon \in (0,1)$. Let κ be a capacity function on the edges of G with $\kappa(e) \leqslant 1/\alpha_\varepsilon$ for all $e \in E(G)$. Then, with high probability, **for all** $W \subseteq V$, $\mu(G_s[W] \cap E_L) \leqslant \mu(G[W] \cap E_L, \kappa^+) + 2\varepsilon\mu(G)$.

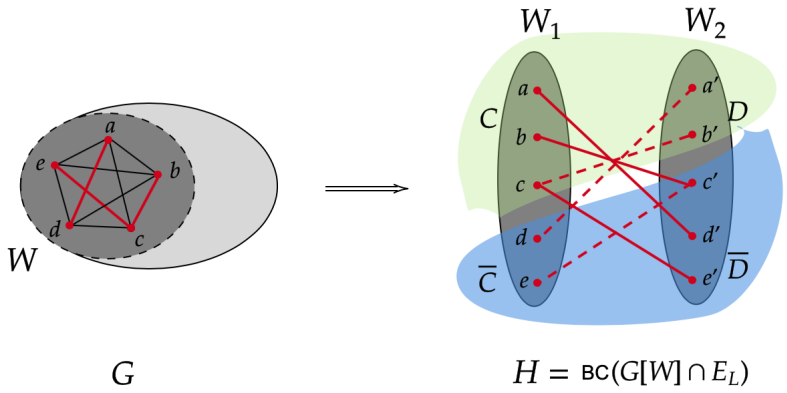

Figure 2: In the left panel, we consider the graph G, and $W = \{a, b, c, d, e\}$. The red edges correspond to the low capacity edges of $G[W]$, denoted $G[W] \cap E_L$. On the right panel, we have the bipartite graph $\text{BC}(G[W] \cap E_L)$, which has bipartitions W_1 and W_2. The solid red edges are the edges going between C and $\bar{D}$, and these edges have capacity $\kappa(C, \bar{D})$.

Proof. The inequality in the statement follows due to the following line of reasoning.

$$\begin{aligned}
2 \cdot \mu(G_s[W] \cap E_L) &\leqslant \mu(\text{BC}\,(G_s[W] \cap E_L)) \\
&\quad \text{(since } 2 \cdot \mu(H) \leqslant \mu(\text{BC}(H)), \text{ see Claim 35)} \\
&\leqslant \mu(\text{BC}(G[W] \cap E_L), \kappa^+_{\text{BC}}) + \varepsilon\mu(G) \\
&\quad \text{(by Lemma 40)} \\
&\leqslant 2 \cdot (1+\varepsilon) \cdot \mu(G[W] \cap E_L, \kappa^+) + \varepsilon\mu(G) \\
&\quad \text{(by Lemma 36)} \\
&\leqslant 2 \cdot \mu(G[W] \cap E_L, \kappa^+) + 3\varepsilon\mu(G).
\end{aligned}$$

The last inequality follows from the fact that any fractional matching in G obeying κ^+ and odd set constraints is upper bounded by $\mu(G)$ (by Lemma 27). Dividing by two on both sides, we have our claim. □

Proof of Lemma 40. Consider a fixed $W \subseteq V$. From now, we will use H to denote $\text{BC}(G[W] \cap E_L)$ and let H_s denote $\text{BC}(G_s[W] \cap E_L)$. For the bipartite graph H, we will use W_1 and W_2 to denote the two bipartitions of H corresponding to W (see Figure 2 for an illustration). We now want to prove that $\mu(H_s) \leqslant \mu(H, \kappa^+_{\text{BC}}) + \varepsilon\mu(G)$. To prove this, it is sufficient to show a set $C \subseteq W_1$ such that $|C| - |N_{H_s}(C)| \geqslant |W_1| - \mu(H, \kappa^+_{\text{BC}}) - \varepsilon\mu(G)$ with high probability. Then, by Proposition 37, we have the following inequality.

$$|W_1| - \mu(H, \kappa^+_{\text{BC}}) - \varepsilon\mu(G) \leqslant |C| - |N_{H_s}(C)| \leqslant |W_1| - \mu(H_s).$$

This would prove our claim. Towards this, we consider the set $C \subseteq W$ satisfying the following equation (applying Observation 39 to H with κ^+_{BC} on edges), and show that this is the required set.

$$|W_1| - |C| + |D| + \kappa^+_{\text{BC}}(C, W_2 \setminus D) = \mu(H, \kappa^+_{\text{BC}}). \tag{1}$$

We want to show that $|C| - |N_{H_s}(C)| \geqslant |W_1| - \mu(H, \kappa^+_{\text{BC}}) - \varepsilon\mu(G)$ with high probability. Let L be the set of vertices in $N_{H_s}(C) \cap W_2 \setminus D$. We know that $|N_{H_s}(C)| \leqslant |D| + |L|$.

Let $E_H(C, W_2 \setminus D)$ be the set of edges in H between C and $W_2 \setminus D$. Let X be the random variable that denotes the number of edges in $E_H(C, W_2 \setminus D)$ that are sampled in H_s. Note that X is not a sum of independent random variables. Recall that H is a subgraph of $\text{BC}(G)$, and suppose $e \in G[W] \cap E_L$ is included in G_s, then e' and e'' (recall e' and e'' are copies of e in $\text{BC}(G)$) are both included in H_s else both are excluded. Thus, the random variables associated with e' and e'' are correlated with each other.

We instead consider an arbitrary subset of $E^*_H(C, W_2 \setminus D)$ of $E_H(C, W_2 \setminus D)$ that satisfies the following properties.

(a) For $e \in G[W]$, if $\{e', e''\} \subset E_H(C, W_2 \setminus D)$, then exactly one of e' or e'' is included in $E^*_H(C, W_2 \setminus D)$.

(b) For $e \in G[W]$, if $\{e', e''\} \cap E_H(C, W_2 \setminus D) = \{e'\}$, then only e' is included in $E^*_H(C, W_2 \setminus D)$.

(c) For $e \in G[W]$, if $\{e', e''\} \cap E_H(C, W_2 \setminus D) = \{e''\}$, then only e'' is included in $E^*_H(C, W_2 \setminus D)$.

We consider the random variable Y that denotes the number of edges in $E^*_H(C, W_2 \setminus D)$ that are included in H_s. Observe that Y is a sum of independent random variables satisfying the condition of Lemma 38. Moreover, $X \leqslant 2Y$. Thus, it is sufficient to upper bound the value Y can take with high probability. Note that for any $e \in E_H(C, W_2 \setminus D)$, using the definition of H, $\kappa_{\text{BC}}(e) < 1/\alpha_\varepsilon^2$. Since $\rho_\varepsilon < \alpha_\varepsilon$, $p(e) = \rho_\varepsilon \cdot \kappa_{\text{BC}}(e) < 1$. So, we have,

$$\mathbb{E}[Y] \leqslant \sum_{e \in E^*_H(C, W_2 \setminus D)} p(e) \leqslant \rho_\varepsilon \cdot \kappa_{\text{BC}}(C, W_2 \setminus D) \leqslant \frac{\kappa^+_{\text{BC}}(C, W_2 \setminus D)}{2^{16/\varepsilon^2}}.$$

The last inequality follows from the fact that in H, $\kappa^+_{\text{BC}}(e) = \kappa_{\text{BC}}(e) \cdot \alpha_\varepsilon$ for all $e \in H$. By definition of H and Definition 33, for all edges $e \in H$, the corresponding original edge in G is in $E_L(G, \kappa)$. We want to bound the following probability.

$$\Pr\left(Y \geqslant \frac{\kappa^+_{\text{BC}}(C, W_2 \setminus D)}{2^{16/\varepsilon^2}} + 4 \cdot \varepsilon\mu(G)\right)$$

Applying Lemma 38 with $\delta = \frac{4 \cdot \varepsilon\mu(G) \cdot 2^{16/\varepsilon^2}}{\kappa^+_{\text{BC}}(C, W_2 \setminus D)}$, we have,

$$\begin{aligned}
\Pr\left(Y \geqslant \frac{\kappa^+_{\text{BC}}(C, W_2 \setminus D)}{2^{16/\varepsilon^2}} + 4 \cdot \varepsilon\mu(G)\right) &= \exp\left(\varepsilon\mu(G) - \varepsilon\mu(G) \log\left(1 + \frac{4 \cdot \varepsilon\mu(G) \cdot 2^{16/\varepsilon^2}}{\kappa^+_{\text{BC}}(C, W_2 \setminus D)}\right)\right) \\
&\text{(Since } \kappa^+_{\text{BC}}(C, W_2 \setminus D) \leqslant 4 \cdot \mu(G) \text{ as proved below.)} \\
&\leqslant \exp\left(\varepsilon\mu(G) - \varepsilon\mu(G) \log\left(1 + \varepsilon \cdot 2^{16/\varepsilon^2}\right)\right) \\
&\text{(From Equation (1), we have } \kappa^+_{\text{BC}}(C, W_2 \setminus D) \leqslant 4 \cdot \mu(G)) \\
&\leqslant \exp\left(\varepsilon\mu(G) - \varepsilon\mu(G) \log\left(1 + 2^{16/\varepsilon^2}\right)\right) \\
&\text{(Using the fact that } 2^{1/\varepsilon^2} \geqslant 1/\varepsilon) \\
&= \exp\left(\varepsilon\mu(G) - 16\mu(G)/\varepsilon\right). \\
&\text{(Using the fact that } 2^{16/\varepsilon^2} \leqslant 2^{16/\varepsilon^2} + 1)
\end{aligned}$$

To see why $\kappa^+_{\text{BC}}(C, W_2 \setminus D) \leqslant 4 \cdot \mu(G)$, consider Equation (1),

$$\begin{aligned}
\kappa^+_{\text{BC}}(C, W_2 \setminus D) &= \mu(H, \kappa^+_{\text{BC}}) - |W_1| + |C| - |D| \\
&\leqslant 2 \cdot (1 + \varepsilon) \cdot \mu(G[W] \cap E_L(G, \kappa), \kappa^+_{\text{BC}}) - |W_1| + |W_1| \\
&\text{(Using Lemma 36, the fact that } H = \textsc{bc}(G[W] \cap E_L(G, \kappa)), \text{ and } \kappa^+ \text{ satisfies the hypothesis.)} \\
&\leqslant 4 \cdot \mu(G).
\end{aligned}$$

Taking a union bound over all W, which are at most $2^{16 \cdot \mu(G)/\varepsilon}$ many, we have our bound (this is because by statement of the lemma, $\mu(G) \geqslant \varepsilon \cdot n/16$). □

6.3 Phase 3: Finding set E^*

The algorithm M-OR-E*() proceeds to Phase 3 if the matching found in G_s in Phase 1 is small, and hence $\mu(G, \kappa)$ is too small. In this case, we need to find a set E^* satisfying the properties of Lemma 9. In particular, we need to find a set E^* with $\kappa(E^*) = O(\mu(G) \log n)$ such that for every large matching M, there are a lot of edges going through E^*. In order to do this, we rely on the properties of the dual variables associated with the matching problem. The algorithm STATIC-MATCH(), luckily for us, solves both the primal as well as the dual solution. We first begin by stating the Dual Matching Program, and then we state the properties of dual variables guaranteed by STATIC-MATCH().

Dual Matching Program The dual of the maximum cardinality matching linear program is the following (see [Sch03]).

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} y(u) + \sum_{B \subseteq V_{\text{odd}}} \frac{|B|-1}{2} z(B) \\ \text{subject to} \quad & yz(e) \geqslant 1, \qquad e \in E(G), \\ & y(u) \geqslant 0 \qquad \forall u \in U, \\ & z(B) \geqslant 0 \qquad \forall B \subseteq V_{\text{odd}} \end{aligned}$$

where,

(a) We define $yz((u,v)) \coloneqq y(u) + y(v) + \sum_{\substack{B \in V_{\text{odd}}, \\ (u,v) \in G[B]}} z(B)$ and,

(b) We define $f(y,z) \coloneqq \sum_{v \in V} y(u) + \sum_{B \subseteq V_{\text{odd}}} \frac{|B|-1}{2} z(B)$.

We now describe some properties of the Static-Match(). We state them without proof for now, and postpone the full proof to the Appendix C.

Lemma 43. There is an $O(m/\varepsilon)$ time algorithm Static-Match() that takes as input a simple graph G with m edges and a parameter $\varepsilon > 0$, and returns a matching M, and dual vectors $\vec{y}$ and $\vec{z}$ that have the following properties.

(a) It returns an integral matching M such that $|M| \geqslant (1-\varepsilon) \cdot \mu(G)$.

(b) A set Ω of laminar odd-sized sets, such that $\{B \mid z(B) > 0\} \subseteq \Omega$.

(c) For all odd-sized B with $|B| \geqslant 1/\varepsilon + 1$, $z(B) = 0$.

(d) Each $y(v)$ is a multiple of ε and $z(B)$ is a multiple of ε.

(e) For every edge $e \in E$, $yz(e) \geqslant 1 - \varepsilon$. We say that such an edge e is *approximately covered* by $\vec{y}$ and $\vec{z}$.

(f) The value of the dual objective, $f(y,z)$ is at most $(1+\varepsilon) \cdot \mu(G)$.

6.3.1 Main Guarantees of Phase 3

We now state the following helper lemma which will be instrumental in proving one of the two main properties of E^*, namely, that every large matching has a lot of edges passing through E^*.

Lemma 44. Suppose G is a graph, and let $H \subset G$ be a subgraph of G. Let $\vec{y}, \vec{z}$ be the dual variables returned on execution of Static-Match() on H and ε. Let $E_H = \{e \in G \mid yz(e) \geqslant 1 - \varepsilon\}$. Let M be any matching of G, then $E \setminus E_H$ contains at least $|M| - (1+\varepsilon)^2 \cdot \mu(H)$ edges of M.

Proof. Suppose we scale up the dual variables $\vec{y}$ and $\vec{z}$ by a factor of $1+\varepsilon$. Then, $\vec{y}$ and $\vec{z}$ is a feasible solution for the dual matching program for the graph E_H. Thus, using weak duality, we have that $\mu(E_H) \leqslant (1+\varepsilon)^2 \cdot \mu(H)$ (this inequality follows from Lemma 43(f)). Suppose M is a matching of G, and suppose $E \setminus E_H$ contains fewer than $|M| - (1+\varepsilon)^2 \cdot \mu(H)$ edges of M, and therefore, fewer than $|M| - \mu(E_H)$ edges of M, then this implies that $|M| = |M \cap E_H| + |M \cap E \setminus E_H| < \mu(E_H) + |M| - \mu(E_H) = |M|$, which is a contradiction. Thus, $E \setminus E_H$ contains at least $|M| - \mu(E_H) \geqslant |M| - (1+\varepsilon)^2 \cdot \mu(H)$ edges of M. □

We now state set E^*, and show that it has the property that $\kappa(E^*) = O(\mu(G) \log n)$.

Lemma 45. Let G be a graph multi-graph such that $\mu(G) \geqslant \varepsilon \cdot n/16$, and let κ be a capacity function on the edges of the graph. Suppose G_s is the graph obtained by sampling every edge $e \in E$ with probability $p(e) = \kappa(e) \cdot \rho_\varepsilon$. Let $\vec{y}, \vec{z}$ be the duals returned by Static-Match(G_s, ε). Let $E^* = \{e \in E \mid yz(e) < 1 - \varepsilon\}$. Then, with high probability, $\kappa(E^*) = O(\mu(G) \log n)$.

Proof. We consider the set $\mathcal{D}$ of assignments $\vec{y}, \vec{z}$ to the vertices and odd components that satisfy the following properties.

(a) For all $v \in V$, $y(v)$ is a multiple of ε, and for all $B \subset V$, $z(B)$ is a multiple of ε.

(b) Let $\Omega = \{B \subset V \mid z(B) > 0\}$, then Ω is laminar.

(c) If $z(B) > 0$ for some $B \subseteq V$, then $|B| \leqslant 1/\varepsilon$.

Observe that,

$$\begin{aligned}
|\mathcal{D}| &\leqslant \sum_{i=0}^{2n} \binom{n^{1/\varepsilon}}{i} \cdot \left(\frac{1}{\varepsilon}\right)^n \cdot \left(\frac{1}{\varepsilon}\right)^{2n} \\
&\leqslant n \cdot \binom{n^{1/\varepsilon}}{2n} \cdot \left(\frac{1}{\varepsilon}\right)^n \cdot \left(\frac{1}{\varepsilon}\right)^{2n} \\
&\leqslant 2^{(4/\varepsilon)\cdot n \log n} \\
&\leqslant 2^{(16/\varepsilon^2)\cdot \mu(G) \log n} \\
&\text{(Since } \mu(G) \geqslant \varepsilon \cdot n/16\text{)}
\end{aligned}$$

This number follows from the following argument. Since Ω is laminar, it can contain at most $2n$ sets. Moreover, from (c), we deduce that these $2n$ sets are chosen from among $n^{1/\varepsilon}$ sets. Further, from (a), we deduce that every vertex can be assigned at most $1/\varepsilon$ values, and every $B \in \Omega$ can be assigned $1/\varepsilon$ values. Therefore, for a given choice of Ω, there are at most $(1/\varepsilon)^n \cdot (1/2\varepsilon)^{2n}$ choices for $\vec{y}$ and $\vec{z}$. Moreover, the above number also upper bounds the number of possible duals that can be a returned by the algorithm Static-Match(), since the duals in $\mathcal{D}$ satisfy a subset of the properties satisfied by the duals returned by Static-Match().

Now consider a fixed $\vec{y}, \vec{z}$ that satisfies the above-mentioned properties, and let E^* be the set of edges that are not approximately covered by $\vec{y}, \vec{z}$. Note that for any $e \in E^*$, $\kappa(e) \leqslant 1/\alpha_\varepsilon$. If not, then, $\kappa(e) = 1$, then it would be sampled in G_s with probability 1, and be approximately covered by $\vec{y}, \vec{z}$. Thus, for any $e \in E^*$, $p(e) = \kappa(e) \cdot \rho_\varepsilon$. Now, suppose $\kappa(E^*) > 17\mu(G) \log n$. Then, observe that if $\vec{y}, \vec{z}$ is output by Static-Match() (denote this event by $\mathcal{E}_{y,z}$) then none of the edges in E^* were sampled (denote this event by $\mathcal{E}_2$). Thus, we have,

$$\Pr(\mathcal{E}_{y,z}) \leqslant \Pr(\mathcal{E}_2) \leqslant \prod_{e \in E^*} (1 - p(e)) \leqslant \exp\left(-\sum_{e \in E^*} \kappa(e) \cdot \rho_\varepsilon\right) \leqslant \exp\left(-17 \cdot 2^{1/\varepsilon^2} \cdot \mu(G) \log n\right)$$

Now, observe that if we are able to upper bound $\Pr\left(\bigcup_{\vec{y},\vec{z} \in \mathcal{D}} \mathcal{E}_{y,z}\right)$, then our lemma follows. This upper bound follows by taking a union bound over the set $\mathcal{D}$.

$$\Pr\left(\bigcup_{\vec{y},\vec{z} \in \mathcal{D}} \mathcal{E}_{y,z}\right) \leqslant \exp\left(-17 \cdot 2^{1/\varepsilon^2} \cdot \mu \log n\right) \cdot 2^{(16/\varepsilon^2)\cdot \mu \log n} = O\left(\exp\left(-2^{1/\varepsilon^2 \cdot \mu \log n}\right)\right)$$

□

7 Algorithm M-or-E*()

In this section, we give the main subroutine M-or-E*() (see Lemma 9). We first define a few terms, and state a known result about finding an approximate fractional matching.

Definition 46. Recall Definition 32, and consider an integral matching M, define $E_L^M(G, \kappa) = E_L(G, \kappa) \cap M$, and let V_M^L be the endpoints of $E_L^M(G, \kappa)$.

Lemma 47. Given a multigraph G (possibly non-bipartite), with edge capacities κ and $\varepsilon \in (0,1)$, such that $\kappa(D(e)) \leqslant 1/\alpha_\varepsilon$ for all $e \in E$, then there is an algorithm Frac-Match() that takes as input G, κ and ε, and returns a fractional matching $\vec{x}$ such that $\sum_{e\in E} x(e) \geqslant (1-\varepsilon)\cdot\mu(G,\kappa)$, obeying the capacities κ and the odd set constraints. The runtime of this algorithm is $O(m\cdot\log n/\varepsilon)$.

Proof. For the case of bipartite graphs, there is an algorithm that takes as input a graph G, an edge capacity function κ and $\varepsilon \in (0,1)$, and returns in $O(m/\varepsilon)$ time, a $(1+\varepsilon)$-approximate fractional matching of G, obeying capacities κ (see [BGS20]). We use this algorithm for a non-bipartite graph as follows: we run the algorithm on $\text{BC}(G,\kappa)$, to obtain a matching $\vec{x}$. To get a fractional matching in G that obeys κ, we do the following: let $e \in E(G)$ and let $e', e'' \in E(\text{BC}(G))$ be copies of e. We let $z(e) = \frac{x(e')+x(e'')}{2}$. Then, the matching $\vec{z}$ obeys κ as well as the fractional matching constraints since $\vec{x}$ obeys them. □

For the purpose of the algorithm recall, definition of κ^+ in Definition 33 given κ and ε. We now formalize the algorithm M-or-E*(). Recall that it takes as input a multigraph G with $\mu(G) \geqslant \varepsilon\cdot n/16$.

Algorithm 3 M-or-E*$(G,\kappa,\varepsilon,\mu)$

1: Include each $e \in E(G)$ independently with probability $p(e) = \kappa(e)\cdot\rho_\varepsilon$ into graph G_s.
2: Let M and $\vec{y},\vec{z}$ be the output of Static-Match(G_s,ε). ▷ Phase 1
3: **if** $|M| < \mu - 6\varepsilon\mu$ **then** ▷ Phase 3
4: Return $E^* = \{e \in E(G) \mid \kappa(e) < 1 \text{ and } yz(e) < 1-\varepsilon\}$.
5: **else** ▷ Phase 2
6: $M_I \leftarrow M \setminus E_L(G,\kappa)$
7: $\vec{y} \leftarrow M_I^D$ ▷ Converting M_I into a matching on a multigraph.
8: $\vec{x} \leftarrow$ Frac-Match$(G[V_L^M] \cap E_L(G,\kappa), \kappa^+, \varepsilon)$
9: **end if**
10: Return $\vec{y}+\vec{x}$.

We now show Lemma 9 holds, we first restate it.

Lemma 9. Let G be a multi-graph with $\mu(G) \geqslant \varepsilon\cdot n/16$. Let κ be a capacity function on the edges of G. There is an algorithm M-or-E*(), that takes as input $G,\kappa,\varepsilon \in (0,1/2)$ and $\mu \geqslant (1-\varepsilon)\cdot\mu(G)$ and in time $O(m\cdot\log n/\varepsilon)$ returns one of the following.

(a) A fractional matching $\vec{x}$ of value at least $(1-10\varepsilon)\cdot\mu$ with the following properties.
  (i) For any $e \in \text{supp}(\vec{x})$ with $\kappa(D(e)) > 1/\alpha_\varepsilon^2$, $x(e) = \frac{\kappa(e)}{\kappa(D(e))}$, and $x(D(e)) = 1$.
  (ii) For any $e \in \text{supp}(\vec{x})$ with $\kappa(D(e)) \leqslant 1/\alpha_\varepsilon^2$, $x(e) \leqslant \kappa(e)\cdot\alpha_\varepsilon$ and $x(D(e)) \leqslant \kappa(D(e))\cdot\alpha_\varepsilon$.

(b) A set E^* of edges such that $\kappa(E^*) = O(\mu\log n)$ such that for any integral matching M with $|M| \geqslant (1-3\varepsilon)\cdot\mu$, we have $|M\cap E^*| \geqslant \varepsilon\mu$. Moreover, $\kappa(e) < 1$ for all $e \in E^*$. Additionally, for every pair of vertices $u,v \in V$, either $D(u,v)\cap E^* = \emptyset$ or $D(u,v) \subseteq E^*$.

Proof of Lemma 9. We first show the runtime of the algorithm. Graph G_s can be computed in time $O(m)$. Using Lemma 43, we conclude that we can compute E^* in order $O(m/\varepsilon)$ time by running Static-Match(G_s,ε) and Lemma 47 implies that Line 8 takes $O(m\cdot\log n/\varepsilon)$ time.

We show Lemma 9(a). First observe that V_L^M and $V(M_I)$ are disjoint, and since $\vec{y}$ and $\vec{x}$ are fractional matchings, $\vec{z}$ is also a fractional matching. Note that $|M| \geqslant \mu - 7\varepsilon\mu$. Moreover, $\sum_{e\in E} x(e) \geqslant (1-\varepsilon)\cdot\mu(G[V_L^M]\cap E_L,\kappa^+)$. This follows from applying Lemma 47 to $G[V_L^M]\cap E_L(G,\kappa)$ with capacity function κ^+. Recall Definition 33 and Definition 32 to see that $\kappa^+(D(e)) \leqslant 1/\alpha_\varepsilon$ for $e \in G[V_L^M]\cap E_L(G,\kappa)$, thus satisfying the requirements of Lemma 47. Next, applying Corollary 42, we have, $\mu(G_s[V_L^M]\cap E_L) \leqslant \mu(G[V_L^M]\cap E_L,\kappa^+) + 5\cdot\varepsilon\mu(G)$. Thus, we have $\sum_{e\in E} x(e) \geqslant (1-\varepsilon)\cdot\big(\mu(G_s[V_L^M]\cap E_L) - 5\varepsilon\mu\big) \geqslant (1-\varepsilon)\cdot(|M\setminus M_I| - 5\varepsilon\mu)$. This is because $M\setminus M_I$ is a matching of $G_s[V_L^M]\cap E_L$. So, $\sum_{e\in E} z(e) \geqslant |M_I| + (1-\varepsilon)\cdot(|M\setminus M_I| - 5\varepsilon\mu) \geqslant (1-\varepsilon)\cdot(\mu - 12\varepsilon\mu) \geqslant \mu - 13\varepsilon\mu$.

We now show Lemma 9(a)(i) and (ii). Consider any edge $e \in \text{supp}(\vec{z})$ with $\kappa(D(e)) \leqslant 1/\alpha_\varepsilon^2$, $e \in$

$G[V_L^M] \cap E_L(G,\kappa)$. Thus, $e \in \text{supp}(\vec{x})$, and therefore, from Lemma 47, $z(e) = x(e) \leqslant \kappa^+(e) \leqslant \kappa(e) \cdot \alpha_\varepsilon$ and $z(D(e)) = x(D(e)) \leqslant \kappa(D(e)) \cdot \alpha_\varepsilon$. Similarly, for any $e \in \text{supp}(\vec{z})$ with $\kappa(D(e)) > 1/\alpha_\varepsilon^2$, $e \in \text{supp}(\vec{y})$. By definition of $\vec{y}$, $z(e) = y(e) = \kappa(e)/\kappa(D(e))$ (recall Definition 7) and $z(D(e)) = 1$. This proves our claim.

Next, we show Lemma 9(b). First recall from the assumption of Lemma 9 that $\mu \geqslant (1-\varepsilon) \cdot \mu(G)$. From this fact and Lemma 45, we can conclude that $\kappa(E^*) = O(\mu \log n)$ and that for all $e \in E^*$, $\kappa(e) < 1$. Next, observe that $\mu(G_s) \leqslant (1+\varepsilon) \cdot |M| \leqslant (1-6\varepsilon) \cdot \mu$. Applying Lemma 44 with $H = G_s$, we have, that for any matching M' of G, $|E^* \cap M'| \geqslant |M'| - (1+\varepsilon)^2 \cdot (1-6\varepsilon) \cdot \mu$. If $|M'| \geqslant (1-3\varepsilon) \cdot \mu$, then we have $|E^* \cap M'| \geqslant \varepsilon \cdot \mu$. Finally, consider any pair of vertices u, v and let $e', e'' \in D(u,v)$. Then, either both e', e'' are both approximately covered by $\vec{y}, \vec{z}$ or neither of them are. This implies that either $D(u,v) \subseteq E^*$ or $D(u,v) \cap E^* = \emptyset$. □

A Missing Proofs

We start by giving a formal proof of the following lemma.

Lemma 36. Let G be a multigraph, and let $\varepsilon \in (0,1)$. Let κ be a capacity function on the edges of G, with $\kappa(D(e)) \leqslant 1/\alpha_\varepsilon$ for all $e \in E(G)$. Then, $\mu(\text{BC}(G), \kappa_{\text{BC}}) \leqslant 2 \cdot (1+\varepsilon) \cdot \mu(G,\kappa)$, where $\mu(G,\kappa)$ is the maximum fractional matching of G obeying κ and the odd set constraints, and $\mu(\text{BC}(G), \kappa_{\text{BC}})$ is the maximum fractional matching of $\text{BC}(G)$ obeying κ_{BC}.

To prove the above lemma, we state the following observation.

Observation 48. Suppose a fractional matching $\vec{x}$ of G satisfies the odd set constraints for all odd sets of size smaller than $3/\varepsilon + 1$, then, the fractional matching $\vec{z} = \frac{\vec{x}}{1+\varepsilon}$ satisfies all odd set constraints.

Proof. Suppose $\vec{x}$ is a fractional matching of G that satisfies odd set constraints for all odd sets of size smaller than $3/\varepsilon + 1$. Let $\vec{z}$ be a fractional matching obtained by scaling down $\vec{x}$ by $1+\varepsilon$. Now, consider any odd set B with $|B| \leqslant 3/\varepsilon$, then $\vec{x}$, and therefore $\vec{z}$ satisfies the odd set constraint corresponding to this. So, we consider B such that $|B| \geqslant 3/\varepsilon + 1$, then,

$$\frac{|B|}{|B|-1} = 1 + \frac{1}{|B|-1} \leqslant 1 + \varepsilon/3$$

The last inequality follows because of the fact that $|B| - 1 \geqslant 3/\varepsilon$. Since $\vec{x}$ satisfies the fractional matching constraints, this implies: $\sum_{e \in G[B]} x(e) \leqslant \frac{|B|}{2}$. Thus, we have, $\sum_{e \in G[B]} z(e) \leqslant \sum_{e \in G[B]} \frac{x(e)}{1+\varepsilon} \leqslant \frac{|B|}{2 \cdot (1+\varepsilon)}$. By the above argument, this is upper bounded by $\frac{|B|-1}{2}$. Thus, $\vec{z}$ satisfies all fractional matching constraints as well as the odd set constraints. □

We now state the following lemma.

Proof of Lemma 36. Consider the maximum fractional matching $\vec{x}$ of $\text{BC}(G,\kappa)$, that obeys the capacity constraints κ. Let $\vec{z}$ be a fractional matching of (G,κ) that is obtained from $\vec{x}$ as follows. Let $e \in E(G)$, and let e', e'' be copies of e in $\text{BC}(G,\kappa)$. Then, we let $z(e) = \frac{x(e') + x(e'')}{2}$. Note that $\vec{z}$ obeys fractional matching constraints, since $\vec{x}$ does. Additionally, observe that $z(e) \leqslant \kappa(e)$. Thus, for any odd set $B \subset V$ with $|B| \leqslant 3/\varepsilon$, we have,

$$\begin{aligned}
\sum_{e \in G[B]} z(e) &= \sum_{u,v \in B} z(D(u,v)) \\
&\leqslant \sum_{u,v \in B} \kappa(D(u,v)) \\
&\leqslant 1/\alpha_\varepsilon \\
&\quad \text{(Since } \kappa(D(e)) \leqslant 1/\alpha_\varepsilon\text{)} \\
&\leqslant \sum_{u,v \in B} \varepsilon/3
\end{aligned}$$

$$
\begin{aligned}
&\text{(Since } {}^{1}/_{\alpha_\varepsilon} \leqslant {}^{\varepsilon}/_{3} \text{ for } \varepsilon < {}^{1}/_{2}) \\
&\leqslant \frac{\varepsilon}{3} \cdot \frac{|B| \cdot (|B|-1)}{2} \\
&\leqslant \frac{|B|-1}{2} \\
&\text{(Since } |B| \leqslant {}^{3}/_{\varepsilon})
\end{aligned}
$$

Thus, $\vec{z}$ satisfies the small odd set constraints, and we know from Observation 48 that $\vec{y} = \frac{\vec{z}}{1+\varepsilon}$ satisfies all odd set constraints in addition to the fractional matching constraints. Thus, we have, $(1+\varepsilon) \cdot \sum_{e \in E(G)} y(e) = \sum_{e \in E(G)} z(e) = 0.5 \sum_{e \in \text{BC}(G)} x(e)$. Thus, if $\vec{x}$ is the optimal fractional matching of $\text{BC}(G)$, then we have,

$$
\begin{aligned}
\mu(G,\kappa) \cdot (1+\varepsilon) &\geqslant (1+\varepsilon) \cdot \sum_{e \in E(G)} y(e) \\
&= 0.5 \sum_{e \in \text{BC}(G)} x(e) \\
&\text{(From the discussion above)} \\
&= 0.5 \cdot \mu(\text{BC}(G), \kappa) \\
&\text{(Since } \vec{x} \text{ is the maximum fractional matching of } (G,\kappa))
\end{aligned}
$$

This proves our claim. □

B Reduction from Simple to Multigraphs

In this section, we will prove the following lemma.

Lemma 3. [AKL19] Let $\delta \in (0,1)$ and let G be a graph with maximum matching size at least μ and at most $(1+\delta) \cdot \mu$. There is an algorithm Vertex-Red(G, μ, δ) that takes as input G, μ, and δ, and in $O(m \cdot {}^{\log n}/_{\delta^4})$ time returns $\lambda = \frac{100 \cdot \log n}{\delta^4}$ multi-graphs $H_1, \cdots, H_\lambda$ that have the following properties, where the second property holds with probability at least $1 - \exp\left(-{}^{\mu \cdot \log n}/_{\delta^4}\right)$.

(a) For all $i \in [\lambda]$, $|V(H_i)| = \frac{4 \cdot (1+\delta) \cdot \mu}{\delta}$ and $E(H_i) \subseteq E(G)$.

(b) Suppose G is subject to deletions, which we simulate on each of the H_i's. More concretely, if an edge e is deleted from G, then its copy in each of the H_i's (if present) is also deleted. Suppose at the end of the deletion process, $\mu(G) \geqslant \tilde{\mu}$ for some $\tilde{\mu} \geqslant (1-2\delta) \cdot \mu$. Then, $\mu(H_i) \geqslant (1-\delta) \cdot \tilde{\mu}$ for some $i \in [\lambda]$.

In order to do that, we first start by proving the following simple observation.

Observation 49. Let G be any simple graph with maximum matching size $\mu(G)$, then G contains at most $2^{O(\mu(G) \cdot \log n)}$ matchings of any size.

Proof. Since the total number of edges is at most n^2, there are at most $\binom{n^2}{j}$ ways of choosing a matching of size j. Thus, we have, the total number of matchings possible is at most:

$$
\sum_{j=1}^{\mu(G)} \binom{n^2}{j} \leqslant \sum_{j=1}^{\mu(G)} 2^{2 \cdot j \log n} \leqslant \mu(G) \cdot 2^{2 \cdot \mu(G) \log n} \leqslant 2^{3 \cdot \mu(G) \log n}
$$

This proves our claim. □

Now, in order to prove Lemma 3, we first give a procedure that takes as input a simple graph, and outputs a multigraph with $O({}^{\mu(G)}/_{\varepsilon})$ vertices that preserves a fixed matching M of G approximately with probability $1 - \exp\left(-O\left(|M| \cdot \varepsilon^3\right)\right)$.

Algorithm 4 VERTEX-RED-BASIC(G, τ, ε)

Input: Graph G, a sparsification parameter τ, and a parameter $\varepsilon > 0$

Output: A multigraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = \tau$

1: Partition V into τ bins, $\mathcal{V} \coloneqq (B_1, B_2 \cdots, B_\tau)$, by assigning every vertex to one of the τ bins uniformly at random.
2: For a vertex u, let $B(u)$ denote the bin chosen for u. For any edge $(u, v) \in E$, if $B(u) \neq B(v)$, then add an edge $e_{u,v}$ between $B(u)$ and $B(v)$.
3: Return the multigraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

Lemma 50. Let G be a simple graph, let $\varepsilon \in (0, 1)$, and let $\tau \geqslant \frac{4 \cdot \mu(G)}{\varepsilon}$. Let $\mathcal{G} = \text{VERTEX-RED-BASIC}(G, \tau, \varepsilon)$, and let M be any fixed matching of G. Then with probability $1 - 2^{-\frac{|M| \cdot \varepsilon^3}{32}}$, there exists a matching $\mathcal{M}$ in $\mathcal{G}$ such that if $\mathcal{M} \subset M$, and $|\mathcal{M}| \geqslant (1 - \varepsilon) \cdot |M|$.

Proof. Let $\delta = \varepsilon/4$, and we define $t \coloneqq 2|M|$. Since $\varepsilon < 1$, this implies that $\delta < 1/2$. Recall that we have $\tau = \frac{4\mu(G)}{\varepsilon}$ bins, and we combine these bins arbitrarily so that we end up with $\frac{t}{\delta}$ groups. Call these groups $Z_1, \cdots, Z_{t/\delta}$. Note that every vertex v lands in bin Z_i with probability δ/t. Call a group Z_i bad if it doesn't contain even one of the $2|M|$ vertices of M. Associate a random variable X_i with Z_i that takes value 1 if Z_i is bad, and zero otherwise. Let $\mathsf{X} = \sum_{i=1}^{t/\delta} \mathsf{X}_i$ denote the number of bad groups. We have,

$$\Pr(\mathsf{X}_i = 1) = \left(1 - \frac{\delta}{t}\right)^t = e^{-\delta} \leqslant 1 - \delta + \delta^2/2$$

Thus, we have, $\mathbb{E}[\mathsf{X}] \leqslant t/\delta \left(1 - \delta + \delta^2/2\right)$. Note that X is a sum of negatively correlated variables. This is because if Z_i is empty, then Z_j has an increased likelihood of being not empty for $i \neq j$. So, using Lemma 38, we have,

$$\begin{aligned}
\Pr\left(\mathsf{X} \geqslant \left(1 + \delta^2\right) \cdot \frac{t}{\delta} \cdot \left(1 - \delta + \delta^2/2\right)\right) &\leqslant \exp\left(-\delta^4 \cdot \frac{t}{\delta} \cdot \left(1 - \delta + \frac{\delta^2}{2}\right)\right) \\
&\leqslant \exp\left(-\delta^3 \cdot t + \delta^4 \cdot t\right) \\
&\leqslant \exp\left(\frac{-\delta^3}{2} \cdot t\right) \\
&\quad (\text{Since } \delta < \frac{1}{2})
\end{aligned}$$

Thus, with probability at least $1 - 2^{-\frac{|M| \cdot \varepsilon^3}{128}}$, $\mathsf{X} \leqslant \left(\frac{t}{\delta} - t + 2\delta t\right)$. Thus, with probability at least $1 - 2^{-\frac{|M| \cdot \varepsilon^3}{128}}$, we have that $t - 2\delta \cdot t$ groups are good, which means that they contain at least one vertex of M. Now, we need to show that if at least $t - 2\delta \cdot t$ bins are good, then the edges of M form a matching $\mathcal{M}$ of $\mathcal{G}$ such that $|\mathcal{M}| \geqslant |M| \cdot (1 - \varepsilon)$. For every bin, we fix one vertex of M, and remove the rest. Since $t - 2\delta \cdot t$ of the bins are good, this implies that we lost at most $2\delta \cdot t$ vertices of the matching M. Consequently, we deleted at most $2\delta \cdot t$ edges of M. The remaining edges have their endpoints in different bins, and therefore, they form a matching in $\mathcal{G}$. The size of this matching is $|M| - 2\delta \cdot t$, which is equal to $|M| - \varepsilon|M|$. Thus, we have our claim. □

Lemma 51. Let VERTEX-RED() be an algorithm that does independent runs of VERTEX-RED-BASIC(). Let $H_1, H_2 \cdots, H_\lambda$ be multigraphs on independent runs of VERTEX-RED-BASIC() on input $G, \tau \geqslant \frac{4 \cdot \mu(G)}{\varepsilon}$ and $\varepsilon \in (0, 1/2)$, for $\lambda \geqslant \frac{1600 \cdot \log n}{\varepsilon^4(1-\varepsilon)}$. Then, with probability at least $1 - \exp(-O(\mu(G) \cdot \log n/\varepsilon))$, for every matching M with $|M| \geqslant (1 - \varepsilon) \cdot \mu(G)$, there is a matching M' in some H_i such that $M' \subset M$, and $|M'| \geqslant (1 - \varepsilon)|M|$.

Proof. Consider a fixed matching M, of size at least $(1 - \varepsilon) \cdot \mu(G)$, then with probability at least $1 - \exp\left(-\frac{\mu(G) \cdot (1-\varepsilon) \cdot \varepsilon^3}{128}\right)$, a fixed H_i contains a matching $M' \subset M$ with $|M'| \geqslant (1 - \varepsilon)|M|$. This is implied by Lemma 50. Since each of the H_j's are independent runs of VERTEX-RED(), with probability at least $1 - \exp\left(-\frac{12 \cdot \mu(G) \cdot \log n}{\varepsilon}\right)$, some H_i contains a matching $M' \subset M$ with $|M'| \geqslant (1 - \varepsilon)|M|$. Taking a union bound over all possible matchings of size at least $(1 - \varepsilon) \cdot \mu(G)$, from Observation 49, we have that with

probability at least $1 - \exp\left(-\frac{9 \cdot \mu(G) \cdot \log n}{\varepsilon}\right)$, for every matching M with $|M| \geqslant (1-\varepsilon) \cdot \mu(G)$, there is a matching M' in some H_i such that $M' \subset M$ and $|M'| \geqslant (1-\varepsilon) \cdot |M|$. □

We now show the following.

Lemma 52. Let G be a simple graph, and let $H_1, \cdots, H_\lambda$ be the output of independent runs of Static-Match() with G and ε as input, where $\lambda \geqslant \frac{1600 \cdot \log n}{\varepsilon^4 \cdot (1-\varepsilon)}$. Consider an adversary that deletes edges from G. We simulate this deletion process in H_i's, that is, if (u, v) is deleted from G, then $e_{u,v}$ (if present) is deleted from each of the H_i's. Let μ_i^t denote the size of the maximum matching of H_i after t deletions, and similarly, let μ^t denote the size of the maximum matching of G after t deletions. Suppose $\mu^t > (1-\varepsilon)\mu^0$, then, $\mu_i^t > (1-2\varepsilon)\mu^0$ for some $i \in [\lambda]$.

Proof. Let M be a matching in G satisfying the statement of the lemma. That is, suppose at time t, $|M| > (1-\varepsilon)\mu^0$. Since the adversary is only performing deletions, $|M| > (1-\varepsilon)\mu^0$ initially, when no deletions had been performed. So, by Lemma 51, before any deletions are performed, there is a matching M' in one of the H_i's such that $M' \subset M$, and $|M'| \geqslant (1-\varepsilon) \cdot |M| > (1 - 2\varepsilon + \varepsilon^2)\mu^0$. Since all of the edges of M survive at time t, this is true of M' as well. Thus, $\mu_i^t > (1-2\varepsilon)\mu^0$ for some $i \in [\lambda]$. This proves our claim. □

We now show that Lemma 3 follows from the lemmas proved in this section.

Proof of Lemma 3. Consider Lemma 3(a). This is implied by the properties of Vertex-Red(), Lemma 51, and Lemma 50. Property Lemma 3(b) is on the other hand implied by the contrapositive of Lemma 52. □

C Properties of Static-Match()

In this section we will show Lemma 43, which we first restate.

Lemma 43. There is an $O(m/\varepsilon)$ time algorithm Static-Match() that takes as input a simple graph G with m edges and a parameter $\varepsilon > 0$, and returns a matching M, and dual vectors $\vec{y}$ and $\vec{z}$ that have the following properties.

(a) It returns an integral matching M such that $|M| \geqslant (1-\varepsilon) \cdot \mu(G)$.

(b) A set Ω of laminar odd-sized sets, such that $\{B \mid z(B) > 0\} \subseteq \Omega$.

(c) For all odd-sized B with $|B| \geqslant 1/\varepsilon + 1$, $z(B) = 0$.

(d) Each $y(v)$ is a multiple of ε and $z(B)$ is a multiple of ε.

(e) For every edge $e \in E$, $yz(e) \geqslant 1 - \varepsilon$. We say that such an edge e is *approximately covered* by $\vec{y}$ and $\vec{z}$.

(f) The value of the dual objective, $f(y, z)$ is at most $(1+\varepsilon) \cdot \mu(G)$.

We first note that the properties (a)-(b) and (d)-(f) are evident in Property 3.1 and the proof of Lemma 2.3 of [DP14]. We now show that (c) also holds.

Proof of Lemma 43. Let Basic-Static-Match() be the algorithm satisfying of [DP14] satisfying (a)-(b) and (d)-(f). We now show how to obtain Static-Match() from Basic-Static-Match(). We run Basic-Static-Match() with input G and $\delta = \varepsilon/3$. Thus, we get a matching M of size at least $(1 - \varepsilon/3) \cdot \mu(G)$. Moreover, the duals $\vec{y}$ and $\vec{z}$ output by the algorithm have $yz(V) \leqslant (1 + \varepsilon/3) \cdot \mu(G)$. We consider the following procedure: for every $B \in \Omega$ with $z(B) > 0$ and $|B| \geqslant \frac{3}{\varepsilon} + 1$, we increase $y(v)$ by $z(B)/2$ for every $v \in B$, and we decrease $z(B)$ to 0. Thus, each $y(v)$ is still a multiple of ε and each $z(B)$ is still a multiple of ε. We conclude that (d) still holds. Moreover, this transformation keeps the value of the dual constraint for every edge the same. Thus, we still satisfy (e). We update Ω by removing B from it. The set Ω still remains laminar. Thus, (b) is still satisfied. We first observe that the time taken to do this is linear in the sum of the sizes of the odd sets B with $z(B) > 0$. Note that since $yz(V)$ sums to at most $(1 + \varepsilon/3)\mu(G)$,

and non-zero $z(B)$ have value at least $\varepsilon/3$, this implies that the sums of the sizes of the odd sets is at most $3/\varepsilon \cdot (1+\varepsilon/3) \cdot \mu(G)$. Thus, the time taken for the procedure is $O(m/\varepsilon)$. Next, we observe that the value of $yz(V)$ changes by at most $\sum_{B:|B|\geqslant 3/\varepsilon+1} \frac{z(B)}{2}$. This value is at most $\varepsilon/3 \cdot (1+\varepsilon/3) \cdot \mu(G)$, as shown by the following calculation.

$$\left(\frac{3}{\varepsilon}\right) \cdot \sum_{B:|B|\geqslant 3/\varepsilon+1} \frac{z(B)}{2} \leqslant \sum_{B:|B|\geqslant 3/\varepsilon+1} \left(\frac{|B|-1}{2}\right) \cdot z(B) \leqslant (1+\varepsilon/3) \cdot \mu(G)$$

Thus, the new $yz(V)$ has value at most $(1+\varepsilon/3)^2 \cdot \mu(G)$. This implies that (f) holds. Finally, since every blossom B of size at least $3/\varepsilon + 1$ has $z(B) = 0$, thus, (c) holds. □

D Rounding Fractional Matchings

In this section, we will prove Lemma 15. More concretely, we give state the procedure Sparsification() that takes as input a fractional matching $\vec{x}$ of a simple graph G, and outputs a graph H, of size $\tilde{O}(\mu(G))$. If $x(e) \leqslant \varepsilon^6$ for all $e \in E(G)$, then, H contains an integral matching of size at least $(1-10\varepsilon) \cdot \sum_{e\in E} x(e)$ in its support. The proof of this theorem is implicit in the work of [Waj20], but we describe it here for completeness. We first state the algorithm, and then describe some of its properties. The algorithm uses a dynamic edge coloring algorithm as a subroutine. The update time of the subroutine is $O(\log n)$ in the worst case. The sparsification algorithm takes as input a fractional matching $\vec{x}$ and a parameter $\varepsilon > 0$. Then, it classifies $E(G)$ into classes as follows: $E_i = \left\{e \mid x(e) \in [(1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1})\right\}$. Note that we only consider edges $e \in E(G)$ with $x(e) \geqslant (\varepsilon/n)^2$, since the total contribution of these edges to fractional matching is at most ε^2, and we can afford to ignore them if we want to compute a $(1+\varepsilon)$ approximation. We now state the algorithm.

Algorithm 5 Sparsification$(\vec{x}, \varepsilon)$

1: $d \leftarrow \frac{4\cdot\log(2/\varepsilon)}{\varepsilon^2}$
2: **for** $i \in \left\{1, 2, \cdots, 2\log_{1+\varepsilon}(n/\varepsilon)\right\}$ **do**
3: Compute a $3\lceil(1+\varepsilon)^i\rceil$-edge colouring Φ_i of E_i.
4: Let S_i be a sample of $3 \cdot \min\left\{\lceil d\rceil, \lceil(1+\varepsilon)^i\rceil\right\}$ colours without replacement in Φ_i.
5: Return $K = (V, \cup_i \cup_{M\in S_i} M)$
6: **end for**

We state the guarantees of the edge coloring subroutine.

Observation 53. The size of K output by Sparsification$(\vec{x}, \varepsilon)$ is,

$$|E(H)| = O\left(\frac{\log(n/\varepsilon)}{\varepsilon} \cdot d \cdot \mu(\text{supp}(H))\right)$$

Lemma 54. [BCHN18] There is a deterministic dynamic algorithm that maintains a $2\Delta - 1$ edge coloring of a graph G in $O(\log n)$ worst case update time, where Δ is the maximum degree of the graph G.

Observation 55. Let $\varepsilon \in (0, 1/2)$ and suppose the input to Sparsification$(\vec{x}, \varepsilon)$ is a matching $\vec{x}$ with $x(e) \leqslant \varepsilon^6$, then, $|S_i| = 3 \cdot \lceil d\rceil$.

To show that K contains a matching of size at least $(1-\varepsilon) \cdot \sum_{e\in E(G)} x(e)$ in its support, we will show a random fractional matching $\vec{y}$ in K that sends flow at most ε through each of its edges. Moreover, $\mathbb{E}\left[\sum_{e\in E} y(e)\right] \geqslant (1-6\varepsilon) \cdot \sum_{e\in E(G)} x(e)$. This will show the existence of a large matching that sends flow at most ε through each of its edges. To show this, we give some properties of the algorithm.

Lemma 56. Let $\varepsilon \in (0, 1/2)$ and let $\vec{x}$ be the input to Sparsification$(\vec{x}, \varepsilon)$ such that $x(e) \leqslant \varepsilon^6$ for all $e \in E(G)$. Then, for every edge e, $\Pr(e \in K) \in [x(e)\cdot d/(1+\varepsilon)^2, x(e) \cdot d \cdot (1+\varepsilon)]$.

Proof. Since $\vec{x}$ has the property that for every $e \in E$, $x(e) \leqslant \varepsilon^6$, this implies that we sample $3 \cdot \lceil d \rceil$ from Φ_i colors for each i (from Observation 55). Thus, if we consider an edge $e \in E_i$, then, we have,

$$\Pr(e \in K) = \frac{\lceil d \rceil}{\lceil (1+\varepsilon)^i \rceil} \leqslant \frac{d \cdot (1+\varepsilon)}{(1+\varepsilon)^i} \leqslant d \cdot (1+\varepsilon) \cdot x(e)$$

The last inequality follows from the fact that $x(e) \in \left[(1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}\right]$. Moreover, we have,

$$\frac{\lceil d \rceil}{\lceil (1+\varepsilon)^i \rceil} \geqslant \frac{d}{(1+\varepsilon)^i + 1} \geqslant \frac{d}{(1+\varepsilon)^{i+1}} \geqslant \frac{d \cdot x(e)}{(1+\varepsilon)^2}$$

The first inequality follows from the fact that $\lceil d \rceil \geqslant d$, and $\lceil (1+\varepsilon)^i \rceil \leqslant (1+\varepsilon)^i + 1$. The second inequality follows from the following reasoning.

$$(1+\varepsilon)^{-i} \leqslant x(e) \leqslant \varepsilon^6 \leqslant 1/d \leqslant \varepsilon$$

Thus, $1/\varepsilon \leqslant (1+\varepsilon)^i$, and $1 \leqslant \varepsilon \cdot (1+\varepsilon)^i$, so the second inequality follows. The last inequality follows from the fact that $x(e) \in \left[(1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}\right]$. □

For an edge $e \in E(G)$, we define X_e to be an indicator random variable that takes value 1 if $e \in K$, and 0 otherwise.

Observation 57. For a vertex v, the variables $\{X_e \mid e \text{ incident on } v\}$ are negatively associated.

Proof. If the edges e and e' incident on v belong in E_i and E_j where $i \neq j$, then X_e and $X_{e'}$ are independent. On the other hand, if they belong in the same E_i, then recall we did an edge colouring on E_i, so e' and e got different colours. So, if the colour corresponding to e is picked into K, then this reduces the probability of the colour corresponding to e' being picked into K. □

Lemma 58. From Observation 57 and Lemma 56, we can conclude that for edges e and e' incident on v,

$$\Pr(X_e = 1 \mid X_{e'} = 1) \leqslant \Pr(X_e = 1) \leqslant x(e) \cdot d \cdot (1+\varepsilon)$$

Lemma 59. For any vertex v and edge e' incident on v, the variables $\{[X_e \mid X_{e'}] \mid e \text{ incident on } v\}$ are negatively associated.

Lemma 60 (Concentration Bound)**.** Let X be the sum of negatively associated random variables $X_1, \cdots, X_k$ with $X_i \in [0, M]$ for each $i \in [k]$. Then, for $\sigma^2 = \sum_{i=1}^{k} \operatorname{Var}[X_i]$ and all $a > 0$,

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

Following theorem directly implies Lemma 15.

Theorem 61. Let K be the subgraph of G output by Sparsification(), when run on the matching $\vec{x}$ with parameters $\varepsilon \in (0, 1/2)$. If $x(e) \leqslant \varepsilon^6$ for all $e \in E$, then K supports a fractional matching $\vec{y}$ such that $y(e) \leqslant \varepsilon$, and $\sum_{e \in E} y(e) \geqslant (1 - 6\varepsilon) \cdot \sum_{e \in E} x(e)$.

Proof. To show the theorem, we will describe a process that simulates Sparsification$(\vec{x}, \varepsilon)$, and outputs a random matching $\vec{y}$ such that $y(e) \leqslant \varepsilon$ for every $e \in E$. Additionally, $\mathbb{E}\left[\sum_{e \in E} y(e)\right] \geq (1 - 6\varepsilon) \sum_{e \in E} x(e)$. As an intermediate step, let $z(e) = \frac{(1-4\varepsilon)}{d} \cdot X_e$. So, we have,

$$\mathbb{E}[z(e)] \geqslant \mathbb{E}[z(e) \mid X_e = 1] \cdot \Pr(X_e = 1) \geqslant \frac{1-4\varepsilon}{d} \cdot \frac{x(e) \cdot d}{(1+\varepsilon)^2} \geqslant (1 - 6\varepsilon) \cdot x(e)$$

The second to last inequality follows from Lemma 56. Next define $\vec{y}$ as follows:

$$y(e) = \begin{cases} 0 \text{ if for one of the endpoints } v \text{ of } e,\ \sum_{e' \in v} z(e') > 1 \\ z(e) \text{ otherwise.} \end{cases}$$

Essentially, our procedure is creating matching $\vec{z}$ as follows: it assigns value $\frac{1-4\varepsilon}{d}$ to $z(e)$ if e was included in K, and 0 otherwise. The value $y(e)$ is the same $z(e)$ in all cases except when e is picked into K, but at one of the endpoints v, $\sum_{e' \ni v} z(e') > 1$, that is, the fractional matching constraint is violated at v. We show that it is unlikely that an edge (when picked) has one of its endpoints violated. Let e' be an edge incident on v, we to bound the probability that $z(e') \neq y(e')$.

$$\begin{aligned}
\mathbb{E}\left[\sum_{e \in v} z(e) \mid X_{e'} = 1\right] &\leqslant \frac{1-4\varepsilon}{d} + \mathbb{E}\left[\sum_{e' \neq e, e \in v} z(e) \mid X_{e'}\right] \\
&\leqslant \varepsilon + \sum_{e \in v} x(e) \cdot (1+\varepsilon) \cdot d \cdot \left(\frac{1-4\varepsilon}{d}\right) \\
&\text{(Since } {}^{1}/_{d} \leqslant \varepsilon \text{ and from Lemma 58)} \\
&\leqslant (1-\varepsilon)
\end{aligned}$$

Note that at any end point of e', conditioned on e' being sampled, the expected sum of $z(e)$'s at that endpoint is upper bounded by $(1-\varepsilon)$. Now, $z(e)$ is assigned value 0 only if the sum of the $z(e)$'s deviates from the expected value by ε. To see this, we want to compute $\operatorname{Var}[[z(e) \mid X_{e'}]]$, where e and e' share an end point. Note that $[z(e) \mid X_{e'}]$ takes value $\frac{1-4\varepsilon}{d}$ with probability $\Pr(X_e \mid X_{e'})$, otherwise it takes value 0.

$$\begin{aligned}
\operatorname{Var}[[z(e) \mid X_{e'}]] &\leqslant \mathbb{E}\left[[z(e) \mid X_{e'}]^2\right] \\
&\leqslant \left(\frac{1-4\varepsilon}{d}\right)^2 \cdot \Pr(X_e \mid X_{e'}) \\
&\leqslant \left(\frac{1-4\varepsilon}{d}\right)^2 \cdot x(e) \cdot d \cdot (1+\varepsilon) \\
&\text{(From Lemma 58)} \\
&\leqslant \frac{x(e)}{d}
\end{aligned}$$

This implies that $\sum_{e \in v} \operatorname{Var}[[z(e) \mid X_{e'}]] \leqslant \frac{1}{d}$. So, we want to compute the probability that the sum of the random variables $\{[z(e) \mid X_{e'}]\}$ deviates from the expected value by ε. Applying Lemma 60, we have,

$$\begin{aligned}
\Pr\left(\sum_{e \in v} [z(e) \mid X_{e'}] \geqslant \mathbb{E}\left[\sum_{e \in v} [z(e) \mid X_{e'}]\right] + \varepsilon\right) &\leqslant \exp\left(\frac{-\varepsilon^2}{2\left(\frac{1}{d} + \frac{\varepsilon}{3 \cdot d}\right)}\right) \\
&\text{(Since } z(e) \in [0, {}^{1-4\varepsilon}/_{d}]) \\
&\leqslant \exp\left(-\varepsilon^2 \cdot 0.25 \cdot d\right) \\
&\text{(Since } \varepsilon \in (0,1)) \\
&\leqslant \frac{\varepsilon}{2} \\
&\text{(Since } d = \frac{4 \cdot \log({}^{2}/_{\varepsilon})}{\varepsilon^2}\text{)}
\end{aligned}$$

Taking union bound over both endpoints, we know that $\Pr(y(e) = z(e) \mid X_e = 1) \geqslant (1-\varepsilon)$. Thus, we have:

$$\begin{aligned}
\mathbb{E}[y(e)] &= \left(\frac{1-4\varepsilon}{d}\right) \cdot \Pr(y(e) = z(e)) \\
&= \left(\frac{1-4\varepsilon}{d}\right) \cdot \Pr(y(e) = z(e) \mid X_e = 1) \Pr(X_e = 1) \\
&\geqslant \left(\frac{1-4\varepsilon}{d}\right) \cdot (1-\varepsilon) \cdot \frac{x(e) \cdot d}{(1+\varepsilon)^2} \\
&\geqslant (1-7\varepsilon) \cdot x(e)
\end{aligned}$$

Thus, $\mathbb{E}\left[\sum_{e \in E} y(e)\right] \geqslant (1-7\varepsilon) \cdot \sum_{e \in E} x(e)$. Moreover, for all $e \in E$, $y(e) \leqslant \varepsilon^6$. This proves our claim. □

We now restate the lemma, and then show its proof.

Lemma 15. [Waj20] Suppose G is an unweighted simple graph, let $\varepsilon \in (0, 1/2)$, and let $\vec{x}$ be a fractional matching of G such that $x(e) \leqslant \varepsilon^6$. Then, there is a dynamic algorithm $\textsc{Sparsification}(\vec{x}, \varepsilon)$, that has the following properties.

(a) The algorithm maintains a subgraph $H \subseteq \text{supp}(\vec{x})$ such that $|E(H)| = O_\varepsilon(\mu(\text{supp}(\vec{x})) \cdot \text{poly}(\log n))$, and with high probability $\mu(H) \geqslant (1-\varepsilon) \cdot \sum_{e \in E} x(e)$.

(b) The algorithm handles the following updates to $\vec{x}$: the adversary can either remove an edge from $\text{supp}(\vec{x})$ or for any edge e, the adversary can reduce $x(e)$ to some new value $x(e) \geqslant 0$.

(c) The algorithm handles the above-mentioned updates in $\tilde{O}_\varepsilon(1)$ worst-case time.

Proof. Note that (a) is implied by Theorem 61, and the fact that any fractional matching $\vec{y}$ with $y(e) \leqslant \varepsilon$ for all $e \in E$, satisfies odd set constraints for all odd sets of size at most $1/\varepsilon$. To see (b), note that deleting e from $\text{supp}(\vec{x})$ corresponds to just deleting e from G_i, and since our edge coloring algorithm is able to handle edge insertions and deletions in $\tilde{O}_\varepsilon(1)$ time, such updates can be handled in $\tilde{O}_\varepsilon(1)$ update time (see Lemma 54). Finally, if an update reduces $x(e)$ for some $e \in E$, then this corresponds to deleting e from some G_i and adding it to G_j for some $j < i$. Thus, the edge colouring algorithms running on G_i and G_j have to handle an edge insertion and deletion respectively, and this can be done in $\tilde{O}_\varepsilon(1)$ time (see Lemma 54). □

References

[AG11] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'11, page 526–538, Berlin, Heidelberg, 2011. Springer-Verlag.

[AKL19] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–19, 2019.

[AV20a] Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the nc model. In *ITCS*, 2020.

[AV20b] Nima Anari and Vijay V. Vazirani. Planar graph perfect matching is in nc. *J. ACM*, 67(4), may 2020.

[BCHN18] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–20. SIAM, 2018.

[BGS20] Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental reachability, scc, and shortest paths via directed expanders and congestion balancing. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1123–1134. IEEE, 2020.

[BHI15] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, page 785–804, USA, 2015. Society for Industrial and Applied Mathematics.

[BHN16] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 398–411, New York, NY, USA, 2016. Association for Computing Machinery.

[BK21] Sayan Bhattacharya and Peter Kiss. Deterministic rounding of dynamic fractional matchings. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[BK22] Soheil Behnezhad and Sanjeev Khanna. *New Trade-Offs for Fully Dynamic Matching via Hierarchical EDCS*, pages 3529–3566. SIAM, 2022.

[BLM20] Soheil Behnezhad, Jakub Lacki, and Vahab Mirrokni. Fully dynamic matching: Beating 2-approximation in δ^ϵ update time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2492–2508. SIAM, 2020.

[BLSZ14] Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 384–393, 2014.

[BS15] Aaron Bernstein and Cliff Stein. Fully dynamic matching in bipartite graphs. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2015.

[BS16] Aaron Bernstein and Cliff Stein. Faster fully dynamic matchings with small approximation ratios. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 692–711. SIAM, 2016.

[CCE+16] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1326–1344. SIAM, 2016.

[Dah16] Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. *CoRR*, abs/1602.06705, 2016.

[DP14] Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

[EKMS11] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63:490–508, 2011.

[FGT16] Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-nc. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 754–763, New York, NY, USA, 2016. Association for Computing Machinery.

[FMU21] Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$-approximate maximum matching with poly$(1/\epsilon)$ passes in the semi-streaming model. *CoRR*, abs/2106.04179, 2021.

[GLS+19] Fabrizio Grandoni, Stefano Leonardi, Piotr Sankowski, Chris Schwiegelshohn, and Shay Solomon. $(1 + \epsilon)$-approximate incremental matching in constant deterministic amortized time. 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 1886–1898. SIAM, 2019.

[GP13] Manoj Gupta and Richard Peng. Fully dynamic (1+ e)-approximate matchings. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 548–557, 2013.

[Gup14] Manoj Gupta. Maintaining approximate maximum matching in an incremental bipartite graph in polylogarithmic update time. In *FSTTCS*, 2014.

[HK73] John E. Hopcroft and Richard M. Karp. An n5/2 algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2:225–231, 1973.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015.

[Kar73] A. Karzanov. On finding a maximum flow in a network with special structure and some applications. *Matematicheskie Voprosy Upravleniya Proizvodstvom (L.A. Lyusternik, ed.), Moscow State Univ. Press, Moscow, 1973, Issue 5, pp. 81–94, in Russian.*, 1973.

[KPP16] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1272–1287. SIAM, 2016.

[MV80] 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.

[MV00] Meena Mahajan and Kasturi R. Varadarajan. A new nc-algorithm for finding a perfect matching in bipartite planar and small genus graphs (extended abstract). In *STOC '00*, 2000.

[RSW22] Mohammad Roghani, Amin Saberi, and David Wajc. Beating the Folklore Algorithm for Dynamic Matching. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111:1–111:23, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Sch03] Alexander Schrijver. Combinatorial optimization. polyhedra and efficiency. 2003.

[ST17] Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-nc. *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 696–707, 2017.

[Waj20] David Wajc. Rounding dynamic matchings against an adaptive adversary. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 194–207, 2020.