Decremental Matching in General Weighted Graphs

Aditi Dudeja ✉
University of Salzburg, Austria

Abstract

In this paper, we consider the problem of maintaining a $(1-\varepsilon)$-approximate *maximum weight matching* in a dynamic graph G, while the adversary makes changes to the edges of the graph. In the fully dynamic setting, where both edge insertions and deletions are allowed, Gupta and Peng [20] gave an algorithm for this problem with an update time of $\tilde{O}_\varepsilon(\sqrt{m})$. We study a natural relaxation of this problem, namely the decremental model, where the adversary is only allowed to delete edges. For the unweighted version of this problem in general (possibly, non-bipartite) graphs, [3] gave a decremental algorithm with update time $O_\varepsilon(\text{poly}(\log n))$. However, beating $\tilde{O}_\varepsilon(\sqrt{m})$ update time remained an open problem for the *weighted* version in *general graphs*. In this paper, we bridge the gap between unweighted and weighted general graphs for the decremental setting. We give a $O_\varepsilon(\text{poly}(\log n))$ update time algorithm that maintains a $(1-\varepsilon)$ approximate maximum weight matching under adversarial deletions. Like the decremental algorithm of [3], our algorithm is randomized, but works against an adaptive adversary. It also matches the time bound for the unweighted version upto dependencies on ε and a $\log R$ factor, where R is the ratio between the maximum and minimum edge weight in G.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases Weighted Matching, Dynamic Algorithms, Adaptive Adversary

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.59

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2312.08996`

Funding *Aditi Dudeja*: This work is supported by Austrian Science Fund (FWF): P 32863-N. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 947702).

Acknowledgements Thanks to Aaron Bernstein, Sebastian Forster, Yasamin Nazari, and anonymous ICALP reviewers for valuable feedback.

1 Introduction

In a dynamic graph G with vertex set V, an adversary makes updates to the edge set E of the graph, and the algorithm is required to adjust to these changes and maintain a key property of the graph. An algorithm is called *decremental* if it can only cope with edge deletions, *incremental* if it can only cope with edge insertions, and *fully dynamic* if it can cope with both edge insertions and deletions. Typically, the algorithm has to optimize the update time, which is the time taken to adapt to a single edge update. For incremental (respectively, decremental) algorithms, one seeks to optimize the amortized update time, which is the average time taken over m edge insertions (respectively, deletions).

In this paper, we consider the problem of maintaining a $(1-\varepsilon)$-approximate matching in a dynamic graph. For the general fully dynamic setting, where we have an unweighted graph undergoing updates, the best known algorithms for this problem due to Bhattacharya, Kiss, Saranurak [11] and Assadi, Behnezhad, Khanna, Li [2] have update times of $O(m^{0.5-\Omega_\varepsilon(1)})$ and $O(n/\log^* n)$, respectively. Improving these bounds to $O_\varepsilon(\text{poly}(\log n))$ is a fundamental


Table 1 A Comparison of Weighted and Unweighted Dynamic Matching in General Graphs.

Setting	Approximation	Time (Unweighted)	Time (Weighted)
Fully Dynamic	$(1-\varepsilon)$	$O(m^{0.5-\Omega_\varepsilon(1)})$ [11]	$\tilde{O}_\varepsilon(\sqrt{m})$ [20]
Fully Dynamic	$(2-\sqrt{2})$	$\tilde{O}_\varepsilon(1)$ [5]	$\tilde{O}_\varepsilon(\sqrt{m})$ [20]
Fully Dynamic	0.609	$\tilde{O}(\min\{\Delta^{1/3}, m^{1/6}\})$ [7]	$\tilde{O}_\varepsilon(\sqrt{m})$ [20]
Incremental	$(1-\varepsilon)$	$O_\varepsilon(1)$ [18]	$\tilde{O}_\varepsilon(\sqrt{m})$ [20]
Decremental	$(1-\varepsilon)$	$\tilde{O}_\varepsilon(1)$ [3]	$\tilde{O}_\varepsilon(1)$ (new)

open problem, and conditional lower bounds of $\Omega(\sqrt{m})$ (see [21] and [23]) for the exact case, seem to suggest that this problem is quite challenging. This has motivated the study of more relaxed versions of this problem. For example, one line of research has focussed on obtaining considerably faster update times, but at the cost of worse approximation ratios [6, 12]. Another research direction has been to study this problem in partially dynamic settings. In the *incremental setting*, there has been steady progress over the years. Several results [19, 18, 11, 14] culminated in a $O_\varepsilon(1)$ update time algorithm for maintaining $(1-\varepsilon)$-approximate matching in incremental graphs. On the *decremental* side, [9] and [22] gave $O_\varepsilon(\text{poly}(\log n))$ update time algorithms for maintaining $(1-\varepsilon)$-approximate matching in bipartite graphs. Subsequently, [3] extended these results to the case of general graphs by giving a $O_\varepsilon(\text{poly}(\log n))$ update time algorithm for the same approximation ratio.

For the case of weighted graphs, the picture is less clear. A general reduction of Stubbs and Williams [25] shows that any α-approximation unweighted matching algorithm can be converted to a $(0.5-\varepsilon)\cdot\alpha$-approximation weighted matching algorithm with nearly the same update time. Subsequently, [8] showed that one can avoid this factor 2 loss in the approximation ratio for the specific case of **bipartite** graphs. However, for the case of general graphs (possibly, non-bipartite), there is a significant gap between the best known weighted and unweighted dynamic algorithms, as illustrated in Table 1. The main contribution of our paper is to close this gap between weighted and unweighted matching for general graphs in the decremental setting:

▶ **Result 1** (Formalized in Theorem 4). *Given a decremental weighted graph G, there is a randomized algorithm that with high probability maintains a $(1-\varepsilon)$-approximation to the maximum weight matching in G in total time $\tilde{O}_\varepsilon(m)$.*

Concurrent Work. Independent of this work, Chen, Sidford, and Tu also [15] gave a $O_\varepsilon(\text{poly}(\log n))$ update time algorithm for maintaining $(1-\varepsilon)$-maximum weight matching in decremental graphs. Their techniques are very different from ours, and extend the entropy regularization approach of [22], who originally gave this framework for bipartite unweighted graphs. Both approaches are of independent interest, since they use vastly different technical frameworks: [15] use insights from concave optimization, whereas our approach is combinatorial. Our approach also has a better dependence on $\log n$[1], and additionally uses simple algorithmic paradigms such as independent sampling or the static approximate matching algorithms of [16]. In contrast, their approach has better dependence on ε (in particular, their update time has a dependence of $(1/\varepsilon)^{O(1/\varepsilon)}$), but uses more involved algorithmic subroutines such as computing k-partial Gomory-Hu trees.

[1] while they don't state it explicitly, we determined that their algorithm has an update time of $O_\varepsilon(\log^{11} n)$.

2 High-Level Overview

Our high-level approach is to follow the congestion balancing framework of [9], who maintain a fractional matching that is robust to deletions. In order to do this, they hope to compute a fractional matching which "spreads out" its flow. Therefore, they impose a capacity function on the edges. These capacities are initially small, and are increased gradually. In particular, they repeatedly invoke a subroutine M-OR-E*() that does one of the following in $O_\varepsilon(m \cdot \log n)$ time (here $\mu(G)$ refers to the size of the maximum cardinality matching):

1. Either output a fractional matching $\vec{x}$, with $x(e) \leqslant \kappa(e)$ for all $e \in E$, and $\sum_{e \in E} x(e) \geqslant (1-\varepsilon) \cdot \mu(G)$, or,
2. Output a set of bottleneck edges E^*, along which we can increase capacities:
   a. We have, $\kappa(E^*) = O(\mu(G) \cdot \log n)$.
   b. Every large matching M has at least $\varepsilon \cdot \mu(G)$ edges in E^*.

Property 2 ensures that we increase capacities carefully: Property 2a ensures the capacity increase is small, and Property 2b ensures that we only increase capacity along important edges. The subroutine M-OR-E*() can be used as a black-box to get a decremental matching algorithm: if M-OR-E*() outputs a fractional matching, then we can round it to an integral matching (see [26, 10, 13]). Otherwise, the capacity obeying maximum fractional matching is too small, and the algorithm outputs a set of bottleneck edges E^*. Thus, we increase capacity along it. Property 2 ensures the matching obtained is "robust" to deletions since it ensures the capacities are small on average.

The authors of [9] also use an outer subroutine which repeatedly invokes M-OR-E*() to get a decremental matching algorithm. Similar to the case of [3], this subroutine carries over with some impediments for the case of **general weighted graphs** as well. But, M-OR-E*() is far more challenging to implement, and this will be the focus of this extended abstract.

2.1 M-or-E*() for General Weighted Graphs

We first explicate what the impediments for implementing M-OR-E*() in general graphs is. For bipartite graphs, M-OR-E*() is much easier to implement, since approximate fractional matchings obeying capacity constraints can be computed using approximate max flows. Similarly, the set of bottleneck edges E^* correspond to the minimum cut. However, in *general graphs*, we don't have such a natural correspondence to max-flow/min-cut due to the integrality gap. More specifically, we mean that not every fractional matching has a large integral matching in its support. On the other hand, while fractional matchings which satisfy odd set constraints do have large integral matchings in their support, this characterization doesn't seem computationally useful at first. [3], in the context of unweighted graphs defined a fractional matching that is both computationally easy to compute, and also avoids the integrality gap. In particular, their candidate fractional matching either puts flow 1 on an edge or a flow of at most ε. By a folklore lemma, one can show that such fractional matchings satisfy small odd set constraints, and therefore have an integrality gap of $1-\varepsilon$. This is sufficient for us, since we focus on only approximation algorithms.

In this work on weighted decremental graphs, we also focus on finding the fractional matchings mentioned above, since even in weighted graphs they are known to have a small integrality gap. Towards this, we show the following structural lemmas. We note that [3] also prove unweighted versions of these structural lemmas, but as we will state later, their proofs are tailored towards unweighted graphs. Consequently, to prove these lemmas for weighted graphs, we come up with significantly different proof strategies. In what follows, we will use $\mathsf{mwm}(G)$ to denote the maximum weight matching of G.

1. First, given a weighted graph G with capacities κ on the edges of the graph, we want to check if the maximum weight fractional matching obeying odd set constraints (denoted $\mathsf{mwm}(G,\kappa)$) is at least $(1-\varepsilon)\cdot\mathsf{mwm}(G)$. As mentioned before, we can't use approximate min-cost flow to find such a fractional matching. First, such an algorithm is computationally expensive. Secondly, the fractional matching returned by it may not obey odd set constraints. Therefore, akin to [3] we prove a sampling theorem: sample every edge e of G independently, with a probability $p(e) \propto \kappa(e)$ to construct an uncapacitated graph G_s. Then, we show, with high probability, $\mathsf{mwm}(G_s) \geqslant \mathsf{mwm}(G,\kappa) - \varepsilon\cdot n$. Thus, we can now use G_s as a substitute for (G,κ): more concretely, we can compute $(1-\varepsilon)\cdot\mathsf{mwm}(G_s)$ using efficient algorithms such as the one by [16], and thus, get an estimate on $\mathsf{mwm}(G,\kappa)$. The authors in [3] proved the unweighted version of this lemma. However, their proof strategy was to use the Tutte-Berge theorem. The Tutte-Berge theorem is the non-bipartite counterpart of the Hall's theorem. In Hall's theorem, deficiency is defined as $\max_{T\subseteq L}\{|T| - |N(T)|\}$, where L is the left vertex set and is equal to $n-\mu(G)$. In the Tutte-Berge theorem, deficiency is analogously defined. The authors in [3] showed that in G_s, with high probability, the deficiency is at most $n-(\mu(G,\kappa)+\varepsilon\cdot n)$, and consequently, $\mu(G_s) \geqslant \mu(G,\kappa) - \varepsilon\cdot n$. However, since Tutte-Berge theorem is specific to unweighted graphs, our proof deviates from this.
2. Suppose the sampling procedure tells us that $\mathsf{mwm}(G,\kappa) \geqslant (1-\varepsilon)\cdot\mathsf{mwm}(G)$. In this case, we would like to compute a fractional matching $\vec{x}$ such that $\sum_{e\in E} w(e)\cdot x(e) \geqslant \mathsf{mwm}(G,\kappa)$. Akin to [3], we consider any M such that $w(M) \geqslant (1-\varepsilon)\cdot\mathsf{mwm}(G_s)$, and we split up M into two parts $M_H \subseteq M$, which are edges with high capacity, and $M_L \subseteq M$ which are edges with low capacity, and $V_H = V(M_H)$ and $V_L = V(M_L)$. Let E_L be the low capacity edges of G. Then, we can show that with high probability, $|M_H| + \mathsf{mwm}(G[V_L]\cap E_L,\kappa) \geqslant \mathsf{mwm}(G_s) - \varepsilon\cdot n$. Since congestion balancing allows us to round capacities of M_H to 1, we only need to compute $(1-\varepsilon)$-approximation to $\mathsf{mwm}(G[V_L]\cap E_L,\kappa)$, and we would have the required fractional matching. There are a few impediments to showing this theorem, and to computing $\mathsf{mwm}(G[V_L]\cap E_L,\kappa)$.
   a. First, the unweighted analog of this theorem by [3], showed the opposite of the structural theorem in 1: it shows that deficiency of G_s can not reduce by too much either. To do this, they consider the bipartite double cover of G_s and (G,κ) and use Hall's theorem to derive this conclusion. As mentioned before, for weighted graphs, we don't have this tool at our disposal, and therefore, have to make other arguments. We look at the dual linear programs of G_s and (G,κ), and use facts about these to draw our conclusions. We believe our arguments can be seen as a generalization of the approach of [3]. Looking at the proof in this light also simplifies the proof considerably.
   b. The second challenge is the computational aspect. We would like to compute a $(1-\varepsilon)$-approximate maximum weight fractional matching in $G[V_L]\cap E_L$ that obeys κ. The authors in [3] observe that since $G[V_L]\cap E_L$ only consists of low capacity edges, we can transform this into a bipartite graph, and then use an approximate flow algorithm. However, for us that would mean computing an approximate min-cost flow, which is computationally expensive. Another way to do this is to use LP-solvers (see [1]). However, these incur additional $\log n$ factors, which we believe in the context of dynamic graph algorithms can be huge. Therefore, we give a simple scaling based algorithm that repeatedly uses maximal flow as a subroutine to obtain a $(1-\varepsilon)$ approximation to $\mathsf{mwm}(G[V_L]\cap E_L,\kappa)$, while avoiding these $\log n$ factors.

There are other peripheral challenges as well, such as computing the set E^*. [9] showed that these edges E^* corresponded to the min-cut, and [3] showed that one can use the duals of G_s to determine these edges. We extend the proof of [3] to weighted graphs. Secondly, the

congestion balancing framework, as shown in [3] and [9] only applies to unweighted graphs, we are able to show that it extends to weighted graphs as well. Finally, the known rounding schemes of [26, 10, 13] as stated only apply to unweighted graphs. We extend [26] to handle the weighted fractional matching we construct.

▶ Remark 2. The structural theorems stated in Item 1 and Item 2a incur an additive error of $\varepsilon \cdot n$. In order to account for this, we use the vertex sparsification technique of [4], which allows us to construct $O(R \cdot \log n)$ multigraphs H_i such that $|V(H_i)| \leqslant O(^{R \cdot \mathsf{mwm}(G)}/_{\varepsilon})$ and each matching of G is preserved in one of the H_i. Here, R is the ratio of the max-weight edge and min-weight edge in G. As is evident, this reduction will cause our algorithms to have a large dependence on R. However, we can reduce this dependence by using a reduction of [20] (see Lemma 3), which has been used similarly in other prior works (see [8, 15]). We refer the reader to Appendix C of [8] for a proof.

▶ **Lemma 3** ([20]). *Let G be a weighted graph with arbitrary weights with R being the ratio between maximum and minimum edge weights. Let $\varepsilon \in (0, 0.5)$ and let $\gamma_\varepsilon = (^1/_\varepsilon)^{O(1/\varepsilon)}$. If there is an algorithm $\mathcal{A}$ that maintains an α-approximate maximum weight matching in a graph whose weights are $\{1, 2, 3, \cdots, W\}$ with update time $T(n, m, \alpha, W)$, then there is an algorithm $\mathcal{A}'$ that maintains a $(1-\varepsilon)\cdot\alpha$-approximate maximum weight matching in G in update time $O(T(n, m, \alpha, \gamma_\varepsilon) \cdot \log R)$.*

In the rest of the paper we will assume that $\mathsf{mwm}(G) = \Omega(\log n)$. This is because the graph is undergoing deletions, and soon as $\mathsf{mwm}(G)$ drops below this value, we can switch to a different decremental algorithm which we describe in the full version of our paper. Additionally, from Remark 2 we can also assume that $\mathsf{mwm}(G) = \Omega(^{\varepsilon \cdot n}/_W)$. Finally, from Lemma 3 we can assume that our graphs are integer-weighted with weights in $\{1, 2, \cdots, W\}$. All our structural theorems will be proven using these assumptions in mind. We will justify these assumption in the full version of this paper.

3 Preliminaries

Throughout the paper, we will use G to refer to the current version of the graph, and let V and E be the vertex set and edge set of G, respectively. The graphs we deal with are weighted, and we use R to denote the ratio between the max-weight and min-weight edge. Additionally use $\mathsf{mwm}(G)$ to denote the weight of the maximum weight matching of G. During the course of the algorithm, we will maintain a fractional matching, which corresponds to a non-negative vector $\vec{x} \in [0,1]^m$ satisfying the following constraints : $\sum_{v \ni e} x(e) \leqslant 1$. For a set $S \subseteq E$, we let $x(S) = \sum_{e \in S} x(e)$. Given a capacity function κ on the edges of the graph, we say that $\vec{x}$ obeys κ if $x(e) \leqslant \kappa(e)$ for all $e \in E$. For a vector $\vec{x}$, we use $\mathrm{supp}(\vec{x})$ to denote the set of edges that are in the support of $\vec{x}$. For a fractional matching $\vec{x}$, we say that it satisfies odd set constraints if for all odd-sized sets $B \subseteq V$, we have, $\sum_{e \in G[B]} x(e) \leqslant \frac{|B|-1}{2}$. We use $\mathsf{mwm}(G, \kappa)$ to denote the size of the maximum weight fractional matching obeying the odd set constraints and the capacity function κ. Additionally, we will use $\gamma_\varepsilon = (^1/_\varepsilon)^{O(1/\varepsilon)}$, $\alpha_\varepsilon = 2^{^{W^2}/_{\varepsilon^3}} \cdot \log n$ and $\rho_\varepsilon = 2^{^{W^2}/_{\varepsilon^2}} \cdot \log n$. We also use V_{odd} to denote the set of all odd-sized subsets $B \subseteq V$. Our main result is a decremental algorithm for maintaining $(1-\varepsilon)$-approximate maximum weight matching in general graphs. In particular, we formally state our main theorem.

▶ **Theorem 4.** *Let G be a weighted graph with weight function w on the edges of the graph and let $\varepsilon \in (0, 1/2)$. There is a decremental algorithm with total update time $O_\varepsilon(m \cdot \log^7 n \cdot \log R)$ (amortized $\tilde{O}_\varepsilon(1)$) that maintains an integral matching M with $w(M) \geqslant (1-\varepsilon)\cdot \mathsf{mwm}(G)$, 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 $2^{(^1/_\varepsilon)^{O(1/\varepsilon)}}$.*

Roadmap for Extended Abstract. As mentioned in the overview, our main contribution is to give an algorithm M-OR-E*() for *weighted general graphs* (we call this subroutine WEIGHTEDM-OR-E*()). Thus, rest of this paper is dedicated to setting this up. We postpone the proof of Theorem 4 to the full version of this paper.

4 Properties of WeightedM-or-E*()

As mentioned in the overview, we will focus our attention on implementing M-OR-E*() in general **weighted** graphs. We will call this subroutine WEIGHTEDM-OR-E*() We additionally mentioned that we will be working with multigraphs, so this will motivate some definitions. After giving these definitions, we state the properties of WEIGHTEDM-OR-E*() that we need. Finally, we show that we are indeed able to implement WEIGHTEDM-OR-E*(). Recall, that is sufficient to design WEIGHTEDM-OR-E*() for the case of graph that have integer weights in $\{1, 2, \cdots, W\}$ by Section 2.1.

▶ **Definition 5.** *Let G be a multigraph. For a pair of vertices $u, v \in V$, we define $D_i(u, v)$ to be the edges e between u and v that have weight i. Additionally, we also have $D(u, v) = \cup_{i=1}^{W} D_i(u, v)$. Since we assume integral weights, these sets are well-defined. If e is an edge between $u, v \in V$, then $D_i(e) \coloneqq D_i(u, v)$ and $D(e) \coloneqq D(u, v)$.*

▶ **Definition 6.** *Let G be a weighted multigraph, with n vertices and m edges. Let κ be the capacity function on the edges of the graph, and let $\vec{x}$ be a fractional matching. We define $\vec{x}^C$ to be a vector, with support size $\min\left\{W \cdot \binom{n}{2}, m\right\}$, where for vertices $u, v \in V$, $x_i^C(u, v) = \sum_{e \in D_i(u,v)} x(e)$. That is, $\vec{x}^C$ is obtained by "collapsing" all edges of the same weight between a pair of vertices together. We now describe the opposite operation: the "distribution" operation. Similarly, suppose $\vec{y}$ is a vector with $|supp(\vec{y})| \leqslant \binom{n}{2} \cdot W$, where, $y_i(u, v)$ is the entry corresponding to the edge of weight i between u and v. Then, we define $\vec{y}^D$ to be an m length vector such that for every $e \in E$ between u, v with $w(e) = i$, $y^D(e) \coloneqq \frac{y_i(u,v) \cdot \kappa(e)}{\kappa(D_i(u,v))}$. Intuitively, $\vec{y}^D$ is a multigraph obtained by distributing the $y_i(u, v)$ among all edges of the same weight.*

We will next, state two folklore observations (proved in the full version of the paper), which motivates the main lemma which we want to prove.

▶ **Observation 7.** *Let $\vec{f}$ be a fractional matching on a multigraph, that puts flow 1 or, at most ε between every pair of vertices $u, v \in V$. Then, $\vec{f}$ satisfies odd set constraints of sets of size at most $1/\varepsilon$.*

▶ **Observation 8.** *Suppose $\vec{x}$ is a fractional matching that satisfies 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 odd set constraints for all odd sets, and therefore, $\mathsf{mwm}(supp(\vec{x})) \geqslant (1 - \varepsilon) \cdot \sum_{e \in E} w(e) \cdot x(e)$.*

We now state the main lemma, which we focus on proving in the extended abstract.

▶ **Lemma 9.** *Let G be a multigraph with edge weights in $\{1, 2, \cdots, W\}$. Then, there is an algorithm WEIGHTEDM-OR-E*() that takes as input $G, \kappa, \varepsilon \in (0, 0.5)$, and a parameter $\mu \geqslant \mathsf{mwm}(G) \cdot (1 - \varepsilon)$, and in time $O(m \cdot W \cdot \log n/\varepsilon)$ returns one of the following.*

1. *A fractional matching $\vec{x}$ such that $\sum_{e \in E} w(e) \cdot x(e) \geqslant (1 - 2\varepsilon) \cdot \mathsf{mwm}(G)$, with the following properties.*
   - **a.** *Let $e \in D_i(u, v)$ such that $e \in supp(\vec{x})$, with $\kappa(D_i(u, v)) \geqslant 1/\alpha_\varepsilon^2$, then $x(D_i(u, v)) = 1$, and we have, $x(e) = \frac{\kappa(e)}{\kappa(D_i(u,v))}$. Moreover, since $\vec{x}$ is a fractional matching, we have $x(D_j(u, v)) = 0$ for all $j \neq i$.*

   b. *Consider any $e \in D_i(u,v)$ such that $e \in supp(\vec{x})$, with $\kappa(D_i(u,v)) \leqslant 1/\alpha_\varepsilon^2$, then $x(D_i(u,v)) \leqslant \kappa(D_i(u,v)) \cdot \alpha_\varepsilon$, and $x(e) \leqslant \kappa(e) \cdot \alpha_\varepsilon$. Moreover, for any $e \in D_j(u,v)$ with $\kappa(D_j(u,v)) > 1/\alpha_\varepsilon^2$, we have $x(e) = 0$.*
2. *A set of edges E^* such that $\sum_{e \in E^*} w(e) \cdot \kappa(e) = O(\mathsf{mwm}(G) \cdot \log n)$, and for any integral matching M with $w(M) \geqslant (1-2\varepsilon) \cdot \mathsf{mwm}(G)$, we have $w(M \cap E^*) \geqslant \varepsilon \cdot \mathsf{mwm}(G)$. Additionally, for all $e \in E^*$, we have $\kappa(e) < 1$.*

We give some intuition about why $\vec{x}$ avoids the integrality gap. Consider the situation in which the algorithm WeightedM-or-E*() returns a fractional matching $\vec{x}$. For any pair of vertices u, v consider $\sum_{i=1}^{W} x_i^C(u,v)$, then this is either 1 or at most ε by Lemma 9(1). Thus, it satisfies odd set constraints for all odd sets of size at most $1/\varepsilon$ by Observation 7. By Observation 8 , we can then argue that $\vec{x}$ contains an integral matching of weight at least $(1+\varepsilon)^{-1} \cdot \sum_{u \neq v} \sum_{i=1}^{W} x_i^C(u,v) \cdot i$ in its support.

We will use WeightedM-or-E*() as a subroutine in our decremental matching algorithm, and we will get a matching $\vec{x}$ with the following properties. We mention these properties since they will help us visualize the fractional matching better. These properties are evident once we state the algorithm WeightedM-or-E*().

▶ **Property 10.** *The set E^* returned WeightedM-or-E*() has the following properties.*
1. *Each time WeightedM-or-E*() returns E^*, we will increase capacity along E^* by multiplying existing capacities by the same factor.*
2. *Consider $u, v \in V$, and let e, e' be two edges between u and v such that $w(e) = w(e')$, then $\kappa(e) = \kappa(e')$ at all times during the run of the algorithm.*

The next property follows directly as a consequence of Lemma 9(1).

▶ **Property 11.** *Let $\vec{x}$ be a matching output by Lemma 9, then $x(e) \leqslant \kappa(e) \cdot \alpha_\varepsilon^2$ for all $e \in E$. This is evident from Lemma 9(1).*

▶ **Definition 12.** *Let G be a multigraph, and let $\vec{x}$ be a fractional matching of G. Then, we split $supp(\vec{x})$ into two parts: $\vec{x}^i$ and $\vec{x}^f$, where, $\vec{x} = \vec{x}^i + \vec{x}^f$, and $supp(\vec{x}^i) = \{e \mid x(D_j(e)) > 1/\alpha_\varepsilon^2 \text{ and } w(e) = j\}$ and $supp(\vec{x}^f) = \{e \mid x(D_j(e)) \leqslant 1/\alpha_\varepsilon^2 \text{ and } w(e) = j\}$. When $\vec{x}$ is the output of WeightedM-or-E*(), then these correspond to the integral and fractional parts of $\vec{x}$.*

▶ **Property 13.** *Let G be any multigraph, and let $\vec{x}$ be a fractional matching of G that is returned by WeightedM-or-E*(). Then, for any pair of vertices, u, v, we have the following properties, which are a consequence of Lemma 9(1a) and Lemma 9(1b):*
1. *Either $D(u,v) \cap supp(\vec{x}^i) \neq \emptyset$ or $D(u,v) \cap supp(\vec{x}^f) \neq \emptyset$, but not both.*
2. *For any u, v, j such that $D_j(u,v) \cap supp(\vec{x}) \neq \emptyset$, we have $D_j(u,v) \subseteq supp(\vec{x})$.*

Thus, the supports of $\vec{x}^i$ and $\vec{x}^f$ are vertex disjoint.

▶ **Property 14.** *Let $\vec{x}$ be a fractional matching returned by WeightedM-or-E*(), consider $\vec{z} = \vec{x}^i$. Then, $\vec{z}^C$ is an integral matching. This is implied by Lemma 9(1).*

5 Building Blocks for WeightedM-or-E*()

In this section, we focus on stating the building blocks we need for Lemma 9. The algorithm WeightedM-or-E*() will proceed in phases. In Phase 1, we would like to determine if a given graph G, with weight function w and capacity function κ on the edges, has the

property that $\mathsf{mwm}(G,\kappa) \geqslant (1-\varepsilon)\cdot\mathsf{mwm}(G)$. We give the structural theorem related to this in Section 5.1. If it is indeed the case that $\mathsf{mwm}(G,\kappa) \geqslant (1-\varepsilon)\cdot\mathsf{mwm}(G)$, then the algorithm proceeds to Phase 2, and in this phase we want to find such a fractional matching. We describe the main structural theorem of this phase in Section 5.2. In Phase 2, there is an additional computational question, that of finding a $(1-\varepsilon)$-approximate maximum weight fractional matching efficiently. We give an algorithm for this in Section 5.4. Finally, if in Phase 1 we know that $\mathsf{mwm}(G,\kappa) < (1-\varepsilon)\cdot\mathsf{mwm}(G)$, then we increase capacities along E^*. In Section 5.3, we give a characterization of these edges. We start with stating some probabilistic tools that we will need, for example the Chernoff Bound:

▶ **Lemma 15** ([17]). *Let $X_1, X_2, \cdots, X_k$ be k negatively associated random variables with $0 \leqslant |X_i| \leqslant M$, and let $X = \sum_{i=1}^{k} X_i$. Suppose $\mu = \mathbb{E}[X]$, and $\mu_{min} \leqslant \mu \leqslant \mu_{max}$. Then, we have, for $\delta > 0$,*

$$\Pr\left(X \geqslant (1+\delta)\cdot\mu_{max}\right) \leqslant \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\frac{\mu_{max}}{M}}$$

$$\Pr\left(X < (1-\delta)\cdot\mu_{min}\right) \leqslant \exp\left(-\frac{\delta^2\cdot\mu_{min}}{(2+\delta)\cdot M}\right)$$

Additionally, we will also need Bernstein's inequality.

▶ **Lemma 16.** *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 $\sigma^2 = \sum_{i=1}^{k} \mathrm{Var}[X_i]$ and all $a > 0$,*

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

5.1 Phase 1 of WeightedM-or-E*()

Recall that we used $\mathsf{mwm}(G,\kappa)$ to denote the weight of the maximum weight fractional matching of G obeying κ as well as the odd set constraints. As in the congestion balancing step, we want to estimate $\mathsf{mwm}(G,\kappa)$. In the bipartite case, because of the correspondence between flows and fractional matchings, one could do this relatively easily. But in non-bipartite graphs, we don't have this correspondence.

In this section, we show the following structural lemma: if in a weighted graph G, we sample every edge with probability $p(e) = \min\{1, \kappa(e)\cdot\rho_\varepsilon\}$ to create a graph G_s, then $\mathsf{mwm}(G_s) \geqslant \mathsf{mwm}(G,\kappa) - \varepsilon\cdot\mathsf{mwm}(G)$. Thus, now we can estimate $\mathsf{mwm}(G_s)$ (and consequently, $\mathsf{mwm}(G,\kappa)$) by using existing results (such [16]). We remark that [3] proved the unweighted version of this lemma, but their proof strategy relied on showing that with high probability, G_s does not contain a Tutte set causing a high deficiency (see Lemma 28 in [3]).

▶ **Lemma 17.** *Let G be an integer weighted multigraph with weights in $\{1, 2, \cdots, W\}$, and with $\mathsf{mwm}(G) \geqslant \max\{{}^{\varepsilon\cdot n}/_{16\cdot W}, {}^{\log n}/_{\varepsilon^4}\}$, where $\varepsilon \in (0, {}^1/_2)$. Let κ be a capacity function on the edges of the graph, and let G_s be obtained by sampling every edge e independently with probability $p(e) = \min\{1, \kappa(e)\cdot\rho_\varepsilon\}$. Then, with high probability, $\mathsf{mwm}(G_s) \geqslant \mathsf{mwm}(G,\kappa) - \varepsilon\cdot\mathsf{mwm}(G)$.*

Proof. Let $\vec{x}$ denote the fractional matching that realizes $\mathsf{mwm}(G,\kappa)$. In order to prove the statement, we will construct a vector $\vec{z}$ in the support of G_s. This vector $\vec{z}$ will have the following properties: it will satisfy the fractional matching constraints and small blossom constraints with high probability. It will also have the property that $\sum_{e\in E)} w(e)\cdot z(e) \geqslant (1-\varepsilon)\cdot\sum_{e\in E} w(e)\cdot x(e) \geqslant \mathsf{mwm}(G,\kappa) - \varepsilon\cdot\mathsf{mwm}(G)$ with high probability as well.

Since with high probability G_s, contains a fractional matching $\vec{z}$ satisfying the above properties, there is an integral matching M in support of G_s with $w(M) \geqslant (1-\varepsilon)\cdot\sum_{e\in E} w(e)\cdot z(e) \geqslant \mathsf{mwm}(G,\kappa) - \varepsilon\cdot\mathsf{mwm}(G)$ (see Observation 8).

Let X_e denote the indicator random variable of the event $e \in G_s$. We define $\vec{z}$ as follows:

$$z(e) = X_e \cdot \frac{x(e)}{\min\{1, \kappa(e)\cdot\rho_\varepsilon\}} \cdot (1-\varepsilon)$$

Note that $\{z(e)\}_{e\in E}$ are independent random variables, and, $\mathbb{E}\left[\sum_{e\in E} z(e)\cdot w(e)\right] = (1-\varepsilon)\cdot\sum_{e\in E} w(e)\cdot x(e)$. Further, consider any e with $\kappa(e) \geqslant \frac{1}{\rho_\varepsilon}$, then $w(e)\cdot z(e) = w(e)\cdot x(e)$ always. Thus, we focus on e with $\kappa(e) < 1/\rho_\varepsilon$. We first have the following observation.

▶ **Observation 18.** *Note that,* $\sum_{e\in E:\kappa(e)<1/\rho_\varepsilon} w(e)\cdot x(e) \geqslant \varepsilon\cdot\mathit{mwm}(G)$. *Otherwise,* $\sum_{e\in E} z(e)\cdot w(e) \geqslant (1-\varepsilon)\cdot\sum_{e\in E} x(e)\cdot w(e) - \varepsilon\cdot\mathit{mwm}(G)$ *with probability* 1.

Using Chernoff bound with $M = \frac{W}{\rho_\varepsilon}$ and $\mu_{\min} = \varepsilon\cdot\mathsf{mwm}(G)$ (see Lemma 15), we have,

$$\Pr\left(\sum_{e\in E:\kappa(e)<1/\rho_\varepsilon} z(e)\cdot w(e) \leqslant \sum_{e\in E:\kappa(e)<1/\rho_\varepsilon} w(e)\cdot x(e) - 2\cdot\varepsilon^2\cdot\mathsf{mwm}(G)\right) = \exp\left(\frac{-\varepsilon^4\cdot\mathsf{mwm}(G)\cdot\rho_\varepsilon}{2\cdot W}\right)$$

By assumption, $\mathsf{mwm}(G) \geqslant 100\cdot\log n/\varepsilon^4$, we have that the above claim holds with probability at least $1 - O(1/n^{1/\varepsilon})$. We will now show that it obeys fractional matching constraints with high probability as well. Consider an edge e with $\kappa(e) > 1/\rho_\varepsilon$, we know that $e \in G_s$. Consequently, we have $\mathrm{Var}[z(e)] = 0$ for such an edge, since $z(e) = x(e)$ always. For any edge with $\kappa(e) < 1/\rho_e$, we have, $\mathrm{Var}[z(e)] \leqslant (1-2\varepsilon)\cdot\frac{x(e)}{\rho_\varepsilon}$. Since $z(e)$ are independent random variables, we have $\mathrm{Var}\left[\sum_{v\ni e} z(e)\right] = \sum_{v\ni e}\mathrm{Var}[z(e)] \leqslant (1-2\varepsilon)\cdot\rho_\varepsilon^{-1}$. Moreover, we know that $\mathbb{E}\left[\sum_{e\ni v} z(e)\right] \leqslant (1-\varepsilon)$. Thus, we want to compute the probability of the event that $\sum_{e\ni v} z(e) \geqslant 1$. Note that, in order to do this, it is sufficient to consider the edges e for which $\kappa(e) < \rho_\varepsilon^{-1}$, since for edges other than this, $z(e) = x(e)$. Thus, by Lemma 16, with $a = \varepsilon$, $M = \rho_\varepsilon^{-1}$, $\Pr\left(\sum_{e\ni v} z(e) > 1\right) = O\left(\frac{1}{n^{1/\varepsilon}}\right)$. ◀

Taking a union bound over all vertices in the graph, we have our claim. Next, we want to compute the probability that $\vec{z}$ also satisfies small odd set constraints. To see this, consider any odd set B such that $|B| \leqslant 1/\varepsilon$. We know that by definition of $\vec{x}$, we have, $\mathbb{E}\left[\sum_{e\in G[B]} z(e)\right] \leqslant (1-\varepsilon)\cdot\frac{|B|-1}{2}$. We can bound variance as well, $\mathrm{Var}\left[\sum_{e\in G[B]} z(e)\right] \leqslant (1-\varepsilon)\cdot\frac{|B|-1}{2}\cdot\frac{1}{\rho_\varepsilon}$. Consider the following subclaim.

▶ **Observation 19.** *Let* B *be any odd set with* $3 \leqslant |B| \leqslant 1/\varepsilon$. *Then,* $\frac{|B|-1}{2} \geqslant (1-\varepsilon)\cdot\frac{|B|-1}{2} + \varepsilon$.

Let $\mathcal{E}_B$ be the event that $\sum_{e\in G[B]} z(e) \geqslant \frac{|B|-1}{2}$, then from the above observation, and Lemma 16, $\Pr(\mathcal{E}_B) = O\left(\frac{1}{n^{1/\varepsilon^2}}\right)$. The second equality follows from the fact that we are considering small blossoms, that is, $|B| \leqslant 1/\varepsilon$. Taking a union bound over all small blossoms, we have our claim.

5.2 Phase 2 of WeightedM-or-E*()

The algorithm WeightedM-or-E*() proceeds to Phase 2 only if in Phase 1, if $\mathsf{mwm}(G,\kappa) \geqslant (1-\varepsilon)\cdot\mathsf{mwm}(G)$ (Lemma 17). In Phase 2, we now want to construct a good fractional matching: it should be close to $\mathsf{mwm}(G,\kappa)$ and also satisfy odd set constraints. The fractional

matching will be inspired by Observation 7 and is constructed as follows: Suppose M is a matching in G_s with weight at least $(1-\varepsilon)\cdot \mathsf{mwm}(G_s)$. Then, we will split up M into high-capacity part, M_H, and low capacity part M_L. We can introduce some slack in the congestion balancing framework, and round up the capacities of M_H. We will then compute a $(1-\varepsilon)$-approximate maximum weight fractional matching $\vec{f}$ on the low-capacity edges of $G[V(M_L)]$. Since these capacities are small, we can accomplish by computing a fractional matching in the bipartite double cover of $G[V(M_L)]$. For now, in this section, we will show that: $w(M_H) + \sum_{e\in E} f(e)\cdot w(e) \geqslant \mathsf{mwm}(G_s) - \varepsilon\cdot \mathsf{mwm}(G)$.

▶ **Definition 20.** *Let G be a multigraph, we define $E_L = \{e \in E \mid e \in D_i(u,v), \kappa(D_i(u,v)) \leqslant 1/\alpha_\varepsilon^2\}$. Intuitively, these correspond to the low capacity edges. Similarly, we define $\kappa^+(e) = \kappa(e)\cdot\alpha_\varepsilon$.*

[3] proved an unweighted version of this lemma, but their proof strategy was based on considering the bipartite double cover of G_s an (G,κ), and analysing the size of the Hall set in G_s. We take a different approach based on the dual programs of $\mathsf{mwm}(G_s)$ and $\mathsf{mwm}(G,\kappa)$. We believe our proof is arguably simpler.

▶ **Lemma 21.** *Let G be a weighted multigraph with maximum edge weight W. Suppose $\mathsf{mwm}(G) \geqslant \max\{\varepsilon\cdot n/16\cdot W, \log n/\varepsilon^4\}$ Then, with high probability, for all $X \subseteq V$, we have*

$$\mathsf{mwm}(G_s[X]) \leqslant \mathsf{mwm}(G[X]\cap E_L, \kappa^+) + \varepsilon\cdot\mathsf{mwm}(G)$$

Proof. Consider a fixed X, and from now on, we use $H \coloneqq G[X]\cap E_L$ and $H_s \coloneqq G_s[X]\cap E_L$. To prove this, we will make use of a primal-dual argument. We state the linear program for $\mathsf{mwm}(H,\kappa^+)$, and its dual.

$$\begin{aligned}
&\text{maximize} && \sum_{e\in E(H)} w(e)\cdot x(e) && \\
&\text{subject to} && && \\
&\sum_{e\ni v, e\in E(H)} x(e) && \leqslant 1 && \forall v \in X \\
&\sum_{e\in G[B]\cap E(H)} x(e) && \leqslant \tfrac{|B|-1}{2} && \forall B \in X_{\text{odd}} \\
&x(e) && \leqslant \kappa^+(e) && \forall e \in E(H)
\end{aligned}$$

The corresponding dual program is,

$$\begin{aligned}
&\text{minimize} && f(y,z,r) = \sum_{u\in V} y(u) + \sum_{B\subseteq X_{\text{odd}}} r(B)\cdot\left(\frac{|B|-1}{2}\right) + \sum_{e\in E(H)} z(e)\cdot\kappa^+(e) && \\
&\text{subject to} && && \\
& && y(u)+y(v)+z(e)+\sum_{B\in X_{\text{odd}}:(u,v)\in G[B]} r(B) \geqslant w(e) && \\
& && && \forall e\in E(H) \text{ between } u,v, \forall u,v\in X
\end{aligned}$$

By strong duality, we know that $\mathsf{mwm}(H,\kappa^+) = f(y,z,r)$ for optimal $\vec{y},\vec{z},\vec{r}$. Similarly, for the uncapacitated graph H_s we have the same primal and dual programs, except we don't have the third constraint in the primal program, and in the dual program we omit the z variables.

$$\begin{aligned}
&\text{maximize} && \sum_{e\in E(H_s)} w(e)\cdot x'(e) && \\
&\text{subject to} && && \\
&\sum_{e\ni v, e\in H_s} x'(e) && \leqslant 1 && \forall v\in X \\
&\sum_{e\in G[B]\cap E(H_s)} x'(e) && \leqslant \tfrac{|B|-1}{2} && \forall B\in X_{\text{odd}}
\end{aligned}$$

The corresponding dual program is as follows:

$$\begin{aligned}
&\text{minimize} && \sum_{u\in V} y'(u) + \sum_{B\subseteq X_{\text{odd}}} r'(B)\cdot\left(\frac{|B|-1}{2}\right) && \\
&\text{subject to} && && \\
&y'(u)+y'(v)+ && \sum_{B\in X_{\text{odd}}:(u,v)\in G[B]} r'(B) \geqslant w(e) && \forall e\in E(H_s) \text{ between } u,v, \forall u,v\in X
\end{aligned}$$

Let $g(y',r')$ denote the optimal for dual program corresponding to H_s. Note that this is a random variable, since H_s is a random graph. We will show that with high probability,

$$g(y',r') \leqslant f(y,r,z) + \varepsilon\cdot \mathsf{mwm}(G)$$

By duality, we have, $\mathsf{mwm}(H_s)\leqslant g(y',r')$, and $f(y,z,r)=\mathsf{mwm}(H,\kappa^+)$. This will show our claim for a fixed H, and then we take a union bound over all H.

We will use $\left\{\{y(u)\}_{u\in X}, \{r(B)\}_{B\in X_{\text{odd}}}\right\}$ to get a solution for the dual program for H_s. We will refer to this set of dual variables as an attempted cover. Let $E'=\{e \mid z(e)>0\}$. Observe that the edges left uncovered by attempted cover of H_s are precisely a subset of E'. We now modify the cover as follows.

$$\Delta y(u) = \sum_{e\ni v, e\in H_s} z(e), \text{ and, } y'(u) = y(u)+\Delta y(u)$$

Note that $\left\{\{y'(u)\}_{u\in X}, \{r(B)\}_{B\in X_{\text{odd}}}\right\}$ is a valid cover of H_s. To see this, consider edge $e\in H_s$ and let u,v be the endpoints of H_s. Suppose $e\notin E'$, then, $w(e)\leqslant y'(u)+y'(v)+\sum_{e\in G[B]} r(B)$. Similarly, for an edge $e\in E'$, we have, $w(e)\leqslant y'(u)+y'(v)+\sum_{e\in G[B]} r(B)$. Moreover, $g(y',r')=f(y,r,z)+\sum_{u\in X}\Delta y(u)$. Thus, it is sufficient to bound $\sum_{u\in X}\Delta y(u) = 2\cdot\sum_{e\in E'} z(e)\cdot X_e$, where X_e is the indicator variable that takes value 1 if $e\in H_s$, and 0 otherwise. Note that $\mathbb{E}\left[\sum_{e\in E'} z(e)\cdot X_e\right] \leqslant \mathsf{mwm}(H,\kappa^+)\cdot 2^{-W^2/\varepsilon^3}$. Using Chernoff and choosing $\delta = \frac{\varepsilon\cdot\mathsf{mwm}(G)\cdot 2^{W^2/\varepsilon^3}}{\mathsf{mwm}(H,\kappa^+)}$ and noting that $M=W$, we have,

$$\Pr\left(\sum_{e\in E'} z(e)\cdot X_e \geqslant 2\cdot\varepsilon\cdot\mathsf{mwm}(G)\right) \leqslant \exp\left(-\varepsilon\cdot\mathsf{mwm}(G)\cdot\frac{W}{\varepsilon^2}\right)$$

By assumption, $n\leqslant\frac{\mathsf{mwm}(G)\cdot W}{\varepsilon}$, thus taking a union bound over all subsets, we have our theorem. ◀

5.3 Phase 3: Finding Set E*

The algorithm WeightedM-or-E*() proceeds to Phase 3, if $\mathsf{mwm}(G, \kappa)$ is not large enough (see Lemma 17). At this point we have to increase capacity along some edges in the graph. Recall Lemma 9(2), the properties required of such edges. Phase 3 finds such edges E^*, and makes sure it has that this set has the property that $\sum_{e \in E^*} w(e) \cdot \kappa(e) = O(\mathsf{mwm}(G) \log n)$, and moreover, every good matching in G has significant weight going through E^*.

Similar to the case of [3], we will use the dual variables of G_s to describe E^*. We will first start with describing the dual program corresponding to the general matching LP:

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} y(u) + \sum_{B \subseteq V_{\text{odd}}} r(B) \cdot \left(\frac{|B| - 1}{2} \right) \\ \text{subject to} \quad & y(u) + y(v) + \sum_{B : (u,v) \in G[B]} r(B) \geqslant w(u,v) \quad \forall (u,v) \in E \end{aligned}$$

Additionally, we define $yr(e)$ to be the dual constraint corresponding to the edge e, and$f(y, r)$ to be the value of the objective function. We now describe some properties of Static-Weighted-Match(H, ε). This is the algorithm whose properties we use to describe E^*. We state these properties without proof for now and postpone the proof to the full version.

▶ **Lemma 22** ([16]). *There is an $O(m/\varepsilon \cdot \log 1/\varepsilon)$ time algorithm Static-Weighted-Match() that takes as input, a weighted graph G, and a parameter $\varepsilon > 0$, and outputs an integral matching M, and dual vectors $\vec{y}$ and $\vec{r}$ with the following properties.*

1. *It returns an integral matching M such that $w(M) \geqslant (1 - \varepsilon) \cdot \mathsf{mwm}(G)$*
2. *A set Ω of laminar odd-sized sets such that $\{B \in V_{odd} \mid r(B) > 0\} \subseteq \Omega$.*
3. *For all odd-sized sets B such that $|B| \geqslant 1/\varepsilon + 1$, $r(B) = 0$.*
4. *For all $v \in V$, $y(v)$ is an integral multiple of ε, and for all $B \in V_{odd}$, $r(B)$ is an integral multiple of ε.*
5. *For each edge $e \in E$, we have $yr(e) \geqslant (1 - \varepsilon) \cdot w(e)$, that is e is* approximately covered *by $\vec{y}$ and $\vec{r}$.*
6. *The value of the dual objective, $f(y, r)$ is at most $(1 + \varepsilon) \cdot \mathsf{mwm}(G)$.*

We now state our first claim, which in Section 6 will help us show that when $\mathsf{mwm}(G, \kappa) < (1 - 2\varepsilon) \cdot \mathsf{mwm}(G)$, then every large matching in G has a lot of weight in E^*.

▷ Claim 23. Suppose $H \subseteq G$, and $\vec{y}, \vec{r}$ are the dual vectors returned by Static-Weighted-Match(H, ε). Let $E_H = \{e \in E \mid yr(e) \geqslant (1 - \varepsilon) \cdot w(e)\}$. Let M be any matching of G, then $w(M \cap E \setminus E_H) \geqslant w(M) - (1 + \varepsilon)^2 \cdot \mathsf{mwm}(H)$.

Proof. Observe that if we scale up the dual variables $\vec{y}$ and $\vec{r}$ by $(1 + \varepsilon)$, then $\vec{y}$ and $\vec{r}$ is a feasible solution for the dual matching problem for the graph E_H. Thus, by weak duality and Lemma 22(6), we have $\mathsf{mwm}(E_H) \leqslant (1 + \varepsilon) \cdot f(y, r) \leqslant (1 + \varepsilon)^2 \cdot \mathsf{mwm}(H)$. Thus, we have: $w(M) \leqslant (1 + \varepsilon)^2 \cdot \mathsf{mwm}(H) + w(M \cap E \setminus E_H)$. ◁

We now describe the set E^*, and show that $w(\kappa(E^*)) = O(\mathsf{mwm}(G) \cdot \log n)$ with high probability.

▶ **Lemma 24.** *Let G be a multigraph such that $\mathsf{mwm}(G) \geqslant \varepsilon n / W$, and let κ be the capacity function on the edges of the graph. Suppose G_s is the graph obtained by sampling edge e with probability $p(e) = \min\{1, \rho_\varepsilon \cdot \kappa(e)\}$. Let $\vec{y}, \vec{r}$ be the duals returned by Static-Weighted-Match(G_s, ε). Let $E^* = \{e \mid yr(e) < (1 - \varepsilon) \cdot w(e)\}$, then with high probability, $w(\kappa(E^*)) = O(\mathsf{mwm}(G) \cdot \log n)$.*

Proof. We consider the set $\mathcal{D}$ of duals $\vec{y}, \vec{r}$ that satisfy the following properties.

1. For all $v \in V$, $y(v)$ is a multiple of ε, and for all $B \in V_{\text{odd}}$, $r(B)$ is a multiple of ε.
2. Let $\Omega = \{B \mid r(B) > 0\}$, then Ω is laminar.
3. If $r(B) > 0$ for some B, then $|B| \leqslant 1/\varepsilon$.

Observe that $\mathcal{D}$ contains all possible duals that could be returned by Static-Weighted-Match(). Now, we bound $|\mathcal{D}|$.

$$|\mathcal{D}| \leqslant \sum_{i=0}^{2n} \binom{n^{1/\varepsilon}}{i} \cdot \left(\frac{W}{\varepsilon}\right)^{n} \cdot \left(\frac{W}{\varepsilon}\right)^{2n} \leqslant 2^{10/\varepsilon^3 \cdot n \cdot \log W} \leqslant 2^{10/\varepsilon^3 \cdot \frac{\mathsf{mwm}(G)}{W} \cdot \log W}$$

This follows from the following argument. Since Ω is laminar (due to 2), there are at most $2n$ sets contained in Ω, and from 3, we can conclude that these laminar sets are chosen from among $n^{1/\varepsilon}$ sets. Similarly, 1 suggests that each $y(v)$ and chosen $r(B)$ can be assigned W/ε values, thus for a given choice of Ω, there are at most $\left(\frac{W}{\varepsilon}\right)^{n} \cdot \left(\frac{W}{\varepsilon}\right)^{2n}$ choices for $\vec{y}$ and $\vec{r}$. We can derive the last set of equations by the premise of our lemma: that $\mathsf{mwm}(G) \geqslant \varepsilon n/W$. Additionally, observe that $\mathcal{D}$ includes the set of duals that can be returned by G, and therefore $|\mathcal{D}|$ is an upper bound on the set of possible duals returned by Static-Weighted-Match().

Now, consider any $\vec{y}, \vec{r} \in \mathcal{D}$ and let $E_{y,r} = \{e \mid yr(e) < (1-\varepsilon) \cdot w(e)\}$. Observe that for all $e \in E_{y,r}$, we have $\kappa(e) < 1/\rho_\varepsilon$, since otherwise $\kappa(e) = 1$, and e would be included in G_s with probability 1 and be approximately covered by $\vec{y}, \vec{r}$ (by Lemma 22(5)). Thus, for all $e \in E_{y,r}$, we have $\kappa(e) < 1/\rho_\varepsilon$. Now, $\vec{y}, \vec{r}$ are such that $w(\kappa(E_{y,r})) > \mathsf{mwm}(G) \cdot \log n$, then none of the edges in $E_{y,r}$ were sampled. Note that in this case, $\kappa(E_{y,r}) \geqslant \mathsf{mwm}(G) \cdot \log n \cdot W^{-1}$. Let $\mathcal{E}_{y,r}$ denote the event that $\vec{y}, \vec{r}$ is returned by Static-Weighted-Match(G_s, ε) and $\mathcal{E}_2$ be the event that $E_{y,r}$ is not sampled into G_s. Then, observe that in this case,

$$\Pr(\mathcal{E}_{y,r}) \leqslant \Pr(\mathcal{E}_2) \leqslant \prod_{e \in E_{y,r}} (1 - p(e)) \leqslant \exp\left(-\sum_{e \in E_{y,r}} \kappa(e) \cdot \rho_\varepsilon\right) \leqslant \exp\left(-\frac{\mathsf{mwm}(G)}{W} \cdot \rho_\varepsilon\right)$$

Now, we want to upper bound the probability that Static-Weighted-Match(G_s, ε) outputs $\vec{y}, \vec{r}$ such that $w(\kappa(E_{y,r})) \geqslant \mathsf{mwm}(G) \cdot \log n$. To do this, we take a union bound over all $|\mathcal{D}|$ many possible duals. From the previous discussion we know that $|\mathcal{D}| \leqslant \exp\left(\frac{10}{\varepsilon^2} \cdot \frac{\mathsf{mwm}(G)}{W} \cdot \log W\right)$. This concludes the proof. ◀

5.4 Weighted Fractional Matching in Bipartite Graphs

The final ingredient we need for WeightedM-or-E*() is the following lemma, which computes a fractional matching on low capacity edges in bipartite graphs. We will show in Section 6 how to use this as a subroutine to do the same in general graphs.

▶ **Lemma 25.** *Consider a weighted bipartite multigraph G with the following property: for any $u, v \in V$ with $e \neq e'$ between u, v, then $w(e) \neq w(e')$. Suppose κ is the capacity function on the edges of the graph, and suppose the edge weights are in $\{1, 2, \cdots, W\}$. There is an algorithm that in $O(m \cdot W \cdot \log n \cdot 1/\varepsilon)$ time finds a fractional matching $\vec{x}$ obeying the capacity function κ such that $\sum_{e \in E} w(e) \cdot x(e) \geqslant (1-\varepsilon) \cdot \mathsf{mwm}(G, \kappa)$.*

Note that there are algorithms in the literature, which can compute weighted fractional matchings in bipartite graphs: such the one by [1]. However, like many LP solvers, one incurs additional $\log n$ factors in the time bound. In contrast, our algorithm incurs a W factor, but since we use Lemma 3, in the final algorithm, we will only incur a $\log W$ factor for the entire decremental algorithm. In contrast, using LP solvers, we will incur additional

$\log n$ factors for implementing WeightedM-or-E*() alone. In order to come up with the algorithm for Lemma 25, we first recall the maximum weight capacitated fractional matching linear program for **bipartite graphs**, and its corresponding dual.

$$\begin{aligned} &\text{maximize} \sum_{e\in E} w(e)\cdot x(e) \\ &\text{subject to} \\ &\quad x(e) \leqslant \kappa(e) && \forall e \in E \\ &\quad \sum_{e\ni v} x(e) \leqslant 1 && \forall v \in V \end{aligned}$$

$$\begin{aligned} &\text{minimize} \sum_{v\in L} y(v) + \sum_{u\in R} y(u) + \sum_{e\in E} z(e)\cdot\kappa(e) \\ &\text{subject to} \\ &\quad y(u)+y(v)+z(e) \geqslant w(e) && \forall e \in E \end{aligned}$$

Here, $yz(e) = y(u) + y(v) + z(e)$, where u, v are the two endpoints of e.

Our algorithm will be a scaling algorithm, which essentially reduces the problem of finding $(1-\varepsilon)$-approximate **maximum weight fractional matching** to the problem of finding a maximal fractional matching, which can be easily accomplished using known subroutines (see [24, 9, 3]). Before stating our algorithm, we give some definitions.

▶ **Definition 26** (Residual Graph)**.** *Given a bipartite graph $G = (L, R, E)$, with a capacity function κ on the edges of the graph. Let $\vec{x}$ be a fractional matching obeying κ. We define G_x to be the residual graph with respect to $\vec{x}$. In particular, this is a directed graph, where corresponding to each undirected edge $e \in G$, there are two directed edges, e_f (forward edges, directed from L to R), and e_b (backward edge, directed from R to L). Moreover, e_f has a residual capacity of $\kappa(e) - x(e)$, and e_b has a capacity of $x(e)$. Thus, if an edge e is saturated, that is $x(e) = \kappa(e)$, then e_f is not in G_x. Similarly, if $x(e) = 0$, then e_b is not in G_x.*

▶ **Definition 27** (Free Vertices)**.** *Given a graph G, and a fractional matching $\vec{x}$, we say that a vertex v is free with respect to $\vec{x}$ if $\sum_{e\ni v} x(e) < 1$.*

▶ **Definition 28.** *An augmenting path in G_x, will refer to a path starting and ending in a free vertex, and all the intermediate vertices on this path will be saturated.*

Before stating our algorithm, we will state the invariants that the algorithm will maintain. Subsequently, we will show that maintaining these invariants throughout implies that the algorithm will output a matching satisfying the approximation guarantees of Lemma 25 and in the desired runtime.

▶ Remark 29. We round down ε so that $1/\varepsilon$ is an integer.

▶ **Property 30.** *Our algorithm will at all times maintain, a fractional matching $\vec{x}$, and the dual variables $y(u)$ for all $u \in V$, and $z(e)$, for all $e \in E$ with the following properties.*

1. ***Granularity:*** *At all times, $y(u)$ for all $u \in V$, and $z(e)$ for all $e \in E$, are integral multiples of ε.*
2. ***Domination:*** *For all directed edges e_f and e_b in G_x, we have $yz(e) \geqslant w(e) - \varepsilon$.*
3. ***Tightness:*** *For all backward edges $e_b \in G_x$, we additionally have the property that $yz(e) \leqslant w(e) + \varepsilon$.*
4. ***Free Duals:*** *The $y(u)$ for free vertices $u \in L$ are equal, and are at most $y(v)$ for $v \in L \setminus Z$. The free vertex duals in R are always 0. The algorithm terminates when the free vertex duals in L are all 0.*
5. ***Complementary Slackness:*** *If $z(e) > 0$ for some $e \in E$, then $x(e) = \kappa(e)$.*

▶ **Definition 31** (Eligible Edges)**.** *Let $\vec{x}$ be the current fractional matching maintained by the graph, and let G_x denote the residual graph with respect to $\vec{x}$. A forward edge e_f is eligible if $yz(e) = w(e) - \varepsilon$, and backward edge e_b is said to be eligible if $yz(e) = w(e) + \varepsilon$. We use G_x^t to denote the eligible subgraph of G_x.*

▶ **Lemma 32.** *The fractional matching $\vec{x}$ returned by the algorithm mentioned in Property 30 has $\sum_{e\in E} w(e)\cdot x(e) \geqslant (1-\varepsilon)\cdot \mathit{mwm}(G)$.*

Proof. In order to see this, consider the dual variables returned by the algorithm. Note that if we increase each of the duals by a factor of $(1+\varepsilon)$, then the dual solutions $\vec{y}, \vec{z}$ maintained by the algorithm are feasible dual solutions to the dual program (this is by Property 30(2)). Thus, by weak duality, we have, $\sum_{u\in V} y(u) + \sum_{e\in E} z(e)\cdot\kappa(e) \geqslant (1-\varepsilon)\cdot \mathsf{mwm}(G)$. Due to complementary slackness, (Property 30(5)), we have, $\sum_{u\in V} y(u) + \sum_{e\in E} x(e)\cdot z(e) \geqslant (1-\varepsilon)\cdot \mathsf{mwm}(G)$. Due to the fact that free vertex duals, at the end of the algorithm are zero (Property 30(4)), we have, $\sum_{u\in V}\sum_{e\ni u} x(e)\cdot y(u) + \sum_{e\in E} z(e)\cdot x(e) \geqslant (1-\varepsilon)\cdot \mathsf{mwm}(G)$. Simplifying the above expression, and using Property 30(3), that is, tightness, we have, $\sum_{e\in E}(1+4\varepsilon)\cdot w(e)\cdot x(e) \geqslant (1-\varepsilon)\cdot \mathsf{mwm}(G)$.This shows the claim. ◀

The precise algorithm is stated in Algorithm 2. We now show that Algorithm 2 proves Property 30. Later, we will show runtime guarantees of Algorithm 2. Towards this, we start with the following observation.

▶ **Observation 33.** *Let $e\in E(G)$, then e_f and e_b cannot simultaneously appear in G_x^t.*

▶ **Definition 34.** *Let H and G be directed, capacitated graphs with capacity functions c_h and c_g respectively. Then, we say $E(H)\subseteq_c E(G)$ if $e\in H$ implies $e\in G$, and $c_h(e)\leqslant c_g(e)$.*

▷ Claim 35. After Line 13, there are no augmenting paths in the graph G_x^t.

Proof. Let $\vec{y}$ denote the fractional matching after the augmentation step. Then, in order to show the claim, it is sufficient to show $E(G_y^t)\subseteq_c E(G_x^t)$. Consider any forward edge $e_f\in E(G_x^t)$, observe that $e_b\notin E(G_x^t)$ due to Observation 33. Since we don't change the duals, $e_b\notin E(G_y^t)$ as well. Finally, e_f can either be augmented along or not, in either case, we have, $\kappa(e)-x(e)\geqslant\kappa(e)-y(e)$. Such an argument applies for any edge e_b as well. From this, we can conclude that any augmenting path that is present in G_y^t is in G_x^t as well, and this contradicts the fact that we found a maximal set of augmenting paths $\mathcal{P}$. The subsequent claims follow from induction, and we prove them in the full version. ◁

▷ Claim 36 (Granularity). Throughout the algorithm, Property 30(1) holds.

▷ Claim 37 (Domination). Throughout the algorithm, Property 30(2) holds.

▷ Claim 38 (Tightness). Throughout the algorithm, Property 30(3) property holds.

▷ Claim 39 (Free Duals). Throughout the algorithm, Property 30(4) holds.

▷ Claim 40 (Complementary Slackness). Throughout the algorithm, Property 30(5) holds.

5.5 Runtime

Before proving the runtime of Algorithm 2, we start with the following structural lemma.

▶ **Observation 41.** *In G_x^t, after dual adjustment step, there are only forward edges between $L\cap Z$ and $R\setminus Z$, and backward edges between $R\cap Z$ and $L\setminus Z$ i.e. all directed edges between Z and $V\setminus Z$ go from Z to $V\setminus Z$.*

Proof. Suppose there is a backward edge e_b between $L\cap Z$ and $R\setminus Z$ after the dual adjustment step. This implies, that after the dual adjustment step, $yz(e_b)=w(e)+\varepsilon$. Thus, prior to the dual adjustment step, we had that $yz(e_b)=w(e)+2\varepsilon$. However, this would contradict the

tightness property (see Property 30(3)). Similarly, suppose there is a forward edge e_f between $R \cap Z$ and $L \setminus Z$ after the dual adjustment step. This would imply that $yz(e_f) = w(e) - \varepsilon$ after the dual adjustment step. Thus, before the dual adjustment step, $yz(e_f) = w(e) - 2\varepsilon$, which would contradict the domination property (see Property 30(2)). ◀

▶ **Lemma 42.** *The graph G_x^t that is fed into* Augmentation *procedure is acyclic.*

Proof. We will show this by induction. Observe that at the beginning of the algorithm, there are no back edges, so the graph is acyclic. We now show that if no cycles are present in the graph, then dual adjustment and augmentation procedures cannot create cycles.

1. First consider 8, observe that if $e_b \in G_x^t$ is subjected to this, then $x(e) = \kappa(e)$ due to complementary slackness. Thus e_f does not exist, and cannot be added to G_x^t due to this modification. Moreover, Line 8 does not affect any other e. Thus, if the graph G_x^t was acyclic before, then it remains acyclic after Line 8.
2. Let $\vec{y}$ be the fractional matching after augmentation step. As we saw in Claim 35, $E(G_y^t) \subseteq_c E(G_x^t)$. Thus, if G_x^t is acyclic, then G_y^t is acyclic as well.
3. Now consider Line 17. By induction hypothesis, there are no cycles in G_x^t before Line 17 was executed. Suppose a cycle C exists in G_x^t after this step, then C contains either e_f or e_b, where e is an edge subjected to Line 17. However, for e_b, $\kappa(e) = x(e)$, and therefore, e_f cannot exist in G_x^t. Moreover, $e_b \notin G_x^t$ since before the dual modification $yz(e) = w(e) - \varepsilon$, and after dual adjustment, $yz(e) = w(e)$.
4. By induction hypothesis, there are no cycles in G_x^t before Line 18 and Line 19. These steps preserve eligibility and ineligibility status of any edge between $L \cap Z$ and $R \cap Z$ and those between $L \setminus Z$ and $R \setminus Z$. Thus, if a new cycle is created after Line 18 and Line 19, then it must contain a directed edge from Z to $V \setminus Z$, and from $V \setminus Z$ to Z. However, Observation 41 suggests that after this step, the latter are not in G_x^t. ◀

▶ **Observation 43.** *Property 30(4) holds. By Line 18, at the end of each iteration of the while loop, the free duals in $L \cap Z$ drops by ε. Therefore, total number of iterations is $O(\frac{W}{\varepsilon})$.*

▶ **Lemma 44** ([24])**.** *For G_x^t acyclic, maximal augmenting paths can be found in time $O(m \log n)$.*

▶ **Lemma 45.** *The total runtime of the algorithm is $O(m/\varepsilon \cdot W \cdot \log n)$.*

Proof. In a particular iteration of a while loop, we see how each of the steps contribute to the runtime. Consider step Line 8, this takes time $O(m)$ and similarly, the subsequent step of updating G_x^t takes time $O(m)$ as well. Since G_x^t is acyclic (by Lemma 42), we can use Lemma 44 to find the maximal set of augmenting paths in time $O(m \cdot \log n)$. Finally, the dual adjustment steps can also be accomplished in time $O(m)$. From Observation 43 implies the total runtime is $O(m/\varepsilon \cdot W \cdot \log n)$. ◀

6 Algorithm WeightedM-or-E*()

In this section, we now put everything together and describe the algorithm WeightedM-or-E*(). Then, we describe how to use Weighted-Frac-Match() to get fractional matchings obeying capacity, and odd set constraints in general graphs.

▶ **Definition 46.** *Given a matching M, define $E_L^M(G, \kappa) = E_L(G, \kappa) \cap M$, and V_L^M to be the set of vertices that are the endpoints of $E_L^M(G, \kappa)$.*

▶ **Definition 47.** *Given a multigraph G with weight function w and capacity function κ on the edges of G, we define the bipartite double cover of G, denoted $\text{BC}(G)$, with capacity function κ_{BC} and weight function w_{BC} as follows:*

1. *For every vertex $v \in V(G)$, make two copies v and v' in $V(\text{BC}(G))$.*
2. *If e is an edge between u, v, then we add an edge e' between u and v' and an edge e'' between u' and v. Moreover, $w_{BC}(e') = w_{BC}(e'') = w(e)$ and $\kappa_{BC}(e') = \kappa_{BC}(e'') = \kappa(e)$.*

▶ **Observation 48.** *For weighted graph G, with capacity κ, $\mathsf{mwm}(\text{BC}(G), \kappa_{BC}) \geqslant 2 \cdot \mathsf{mwm}(G, \kappa)$.*

▶ **Lemma 49.** *Given a weighted multigraph G (possibly non-bipartite) with capacity function κ, $\varepsilon \in (0, 1)$, with edge weights from $\{1, 2, \cdots, W\}$, and with the property that for all $i \in [W]$, and for all vertices $u, v \in V$ we have, $\kappa(D_i((u, v))) \leqslant 1/\alpha_\varepsilon$, then there is an algorithm Weighted-Frac-Match-General() that takes as input $G, \kappa, w, \varepsilon$ and outputs a fractional matching $\vec{x}$ such that $\sum_{e \in E} w(e) \cdot x(e) \geqslant (1 - \varepsilon) \cdot \mathsf{mwm}(G, \kappa)$, and further $\vec{x}$ obeys the capacities and the odd set constraints.*

Proof. In the previous section, we saw the algorithm Weighted-Frac-Match() that solved this problem for bipartite graphs. For the case of general graphs, we proceed as follows. We collapse edges of $\text{BC}(G)$ as in Definition 6, and run Weighted-Frac-Match$(\text{BC}(G), \kappa_{\text{BC}}, w_{\text{BC}}, \varepsilon)$, and obtain a fractional matching $\vec{y}$. We let $\vec{z} = \vec{y}^C$. To translate this into a valid fractional matching $\vec{x}$ in G, do the following: for each $e \in G$, consider $e', e'' \in \text{BC}(G)$, and let $x(e) = \frac{z(e') + z(e'')}{2}$. Note that $\vec{z}$ is a fractional matching, since $\vec{x}$ is and moreover since $\vec{x}$ satisfies capacity constraints since $\vec{z}$ does. Applying Observation 48, we know that $\sum_{e \in E} w(e) \cdot x(e) \geqslant (1 - \varepsilon) \cdot \mathsf{mwm}(G, \kappa)$. Moreover, $\frac{\vec{x}}{1+\varepsilon}$ satisfies all odd set constraints since it satisfies the premise of Observation 8. ◀

We show how Algorithm 1 satisfies the conditions of Lemma 9.

Proof. (Lemma 9) We first show the runtime of the algorithm. First note that G_s can be computed in $O(m)$ time since this only involves sampling edges independently. Moreover, by Lemma 22, we know that the runtime of Static-Weighted-Match(G, ε) is $O(m/\varepsilon \cdot \log 1/\varepsilon)$. Additionally, from Lemma 24, we can conclude that we can obtain set E^* using the output of Static-Weighted-Match(G_s, ε) and the time taken to do this is $O(m)$. Finally, from Lemma 49, we can conclude that we can run Line 7 in time $O(m/\varepsilon \cdot \log 1/\varepsilon \cdot W \cdot \log n)$.

Now, we turn to show Lemma 9(1). In this case, first note that V_L^M and $V(M_I)$ are disjoint. This follows from Algorithm 1 and Definition 46. Thus, since $\vec{y}, \vec{x}$ are fractional matchings, we can conclude that $\vec{z}$ is also a fractional matching. Note that we land in Lemma 9(1) if $w(M) \geqslant (1-\varepsilon) \cdot \mathsf{mwm}(G)$. From Lemma 21 we can conclude that for $H = E_L(G, \kappa) \cap G[V_L^M]$ and $H_s = E_L(G, \kappa) \cap G_s[V_L^M]$, we have that $\mathsf{mwm}(H_s) \leqslant \mathsf{mwm}(H, \kappa^+) + \varepsilon \cdot \mathsf{mwm}(G)$. Thus, we have, $\sum_{e \in E} y(e) \cdot w(e) + \sum_{e \in E} x(e) \cdot w(e) \geqslant (1 - \varepsilon) \cdot \mathsf{mwm}(G) - 2\varepsilon \cdot \mathsf{mwm}(G)$.

We now proceed to proof Lemma 9(1a) and (1b). Consider any edge $e \in \text{supp}(\vec{z})$ with $w(e) = i$ and $\kappa(D_i(e)) \leqslant 1/\alpha_\varepsilon^2$. Thus, we know that $e \in \text{supp}(\vec{x})$, and so by Lemma 49, we have that $z(e) = x(e) \leqslant \kappa^+(e) \leqslant \kappa(e) \cdot \alpha_\varepsilon$ and $z(D_i(e)) = x(D_i(e)) \leqslant \kappa^+(D_i(e)) \leqslant \kappa(D_i(e)) \cdot \alpha_\varepsilon$. Similarly, for any $e \in \text{supp}(\vec{z})$ with $w(e) = i$, and $\kappa(D_i(e)) > 1/\alpha_\varepsilon^2$. We know that $e \in \mathsf{supp}(\vec{y})$. By definition of $\vec{y}$, we have $z(e) = y(e) = \kappa(e)/\kappa(D_i(e))$, and $z(D_i(e)) = 1$. Finally, we prove Lemma 9(2). First consider Lemma 24, this lemma implies that $\sum_{e \in E^*} \kappa(e) \cdot w(e) = O(\mathsf{mwm}(G) \cdot \log n)$. Finally, by Claim 23, we have, $\sum_{e \in M \cap E^*} w(e) \geqslant w(M) - (1+\varepsilon)^2 \cdot \mathsf{mwm}(G_s)$, thus, for any M with $w(M) \geqslant (1-\varepsilon) \cdot \mathsf{mwm}(G)$, we have $\sum_{e \in M \cap E^*} w(e) \geqslant \varepsilon \cdot \mathsf{mwm}(G)$. This is implied by the fact that the algorithm returns E^* when $\mathsf{mwm}(G_s) \leqslant (1 - 5\varepsilon) \cdot \mathsf{mwm}(G)$. Finally, for all edges in E^*, $\kappa(e) < 1$, otherwise they'd be sampled into G_s. ◀

References

1. Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 526–538, 2011.
2. Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 131–144. ACM, 2023. doi:10.1145/3564246.3585110.
3. Sepehr Assadi, Aaron Bernstein, and Aditi Dudeja. Decremental matching in general graphs. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 11:1–11:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.11.
4. 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.
5. Amir Azarmehr, Soheil Behnezhad, and Mohammad Roghani. Fully dynamic matching: (2-$\sqrt{2}$)-approximation in polylog update time. *CoRR*, abs/2307.08772, 2023. doi:10.48550/arXiv.2307.08772.
6. Soheil Behnezhad. Dynamic algorithms for maximum matching size. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 129–162. SIAM, 2023. doi:10.1137/1.9781611977554.CH6.
7. Soheil Behnezhad and Sanjeev Khanna. New trade-offs for fully dynamic matching via hierarchical EDCS. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 3529–3566. SIAM, 2022. doi:10.1137/1.9781611977073.140.
8. Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 668–681. ACM, 2021. doi:10.1145/3406325.3451113.
9. 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.
10. Sayan Bhattacharya and Peter Kiss. Deterministic rounding of dynamic fractional matchings. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.27.
11. Sayan Bhattacharya, Peter Kiss, and Thatchaphol Saranurak. Dynamic $(1+\epsilon)$-approximate matching size in truly sublinear update time. *CoRR*, abs/2302.05030, 2023. doi:10.48550/arXiv.2302.05030.
12. Sayan Bhattacharya, Peter Kiss, Thatchaphol Saranurak, and David Wajc. Dynamic matching with better-than-2 approximation in polylogarithmic update time. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 100–128. SIAM, 2023. doi:10.1137/1.9781611977554.CH5.
13. Sayan Bhattacharya, Peter Kiss, Aaron Sidford, and David Wajc. Near-optimal dynamic rounding of fractional matchings in bipartite graphs. *CoRR*, abs/2306.11828, 2023. doi:10.48550/arXiv.2306.11828.
14. Joakim Blikstad and Peter Kiss. Incremental (1-ϵ)-approximate dynamic matching in o(poly(1/ϵ)) update time. In *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 22:1–22:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.22.

15 Jiale Chen, Aaron Sidford, and Ta-Wei Tu. Entropy regularization and faster decremental matching in general graphs. *arXiv preprint arXiv:2312.09077*, 2023.

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

17 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. URL: `http://www.cambridge.org/gb/knowledge/isbn/item2327542/`.

18 Fabrizio Grandoni, Stefano Leonardi, Piotr Sankowski, Chris Schwiegelshohn, and Shay Solomon. $(1 + \epsilon)$-approximate incremental matching in constant deterministic amortized time. In *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. `doi:10.1137/1.9781611975482.114`.

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

20 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.

21 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.

22 Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Regularized box-simplex games and dynamic decremental bipartite matching. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 77:1–77:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPICS.ICALP.2022.77`.

23 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.

24 Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983. `doi:10.1016/0022-0000(83)90006-5`.

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

26 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.

A Subroutines

Algorithm 1 WeightedM-or-E*$(G, \kappa, \varepsilon, \mu, w)$.

Include each edge $e \in E(G)$ independently with probability $p(e) = \kappa(e) \cdot \rho_\varepsilon$ in G_s.
Let $M, \vec{y}, \vec{r}$ be the output of Static-Weighted-Match(G_s, ε). ▷ Phase 1.
if $w(M) \leqslant (1 - 6\varepsilon) \cdot \mathsf{mwm}(G)$ **then**
    Return $E^* = \{e \mid yr(e) < (1 - \varepsilon) \cdot w(e)\}$. ▷ Phase 3
else ▷ Phase 2
    $M_I \leftarrow M \setminus E_L(G, \kappa)$, $\vec{y} \leftarrow M_I^D$ ▷ Convert M_I into a matching on multigraph
    $\vec{x} \leftarrow$ Weighted-Frac-Match-General$(G[V_L^M] \cap E_L(G, \kappa), \kappa^+, \varepsilon, w)$
end if
Return $\vec{z} \leftarrow \vec{y} + \vec{x}$.

Algorithm 2 WEIGHTED-FRAC-MATCHING(G, κ, ε).

Ensure: A fractional matching $\vec{x}$ with $\sum_{e \in E} w(e) \cdot x(e) \geqslant (1-\varepsilon) \cdot \mathsf{mwm}(G, \kappa)$

procedure INITIALIZATION:
  $x(e) \leftarrow 0$ for all $e \in E$
  $y(u) \leftarrow W - \varepsilon$ for all $u \in L$, $y(u) \leftarrow 0$ for all $u \in R$ ▷ Initializing vertex duals.
  $z(e) \leftarrow 0$ for all $e \in E$ ▷ Initializing edge duals.
end procedure
while y-values of free left vertices are greater than 0 **do**
  for $e_b \in G_x^t$ with $z(e) > 0$ **do**
    $z(e) \leftarrow z(e) - \min\{z(e), yz(e) - w(e) + \varepsilon\}$.
  end for
  Update G_x^t
  procedure AUGMENTATION:
    Find the maximal set $\mathcal{P}$ of augmenting paths in G_x^t.
    Augment along $\mathcal{P}$. Update $\vec{x}$ and G_x^t.
  end procedure
  procedure DUAL ADJUSTMENT:
    Let Z be the set of vertices reachable from free left vertices in G_x^t.
    For all ineligible e_b from $R \setminus Z$ to $L \cap Z$, with $yz(e) = w(e) - \varepsilon$, $z(e) \leftarrow z(e) + \varepsilon$.
    For all $u \in L \cap Z$, $y(u) \leftarrow y(u) - \varepsilon$.
    For all $u \in R \cap Z$, $y(u) \leftarrow y(u) + \varepsilon$.
  end procedure
end while