Stochastic Weighted Matching: $(1 - \varepsilon)$ Approximation

Soheil Behnezhad
University of Maryland
Email: soheil@cs.umd.edu

Mahsa Derakhshan
University of Maryland
Email: mahsa@cs.umd.edu

***Abstract*—Let $G = (V, E)$ be a given edge-weighted graph and let its *realization* $\mathcal{G}$ be a random subgraph of G that includes each edge $e \in E$ independently with probability p. We study a *stochastic matching* problem where the goal is to non-adaptively pick a sparse subgraph Q of G (without knowing the realization $\mathcal{G}$), such that the maximum weight matching among the realized edges of Q (i.e. graph $Q \cap \mathcal{G}$) in expectation approximates the maximum weight matching of the whole realization $\mathcal{G}$.**

In this paper, we prove that for any $\varepsilon \in (0, 1)$, every graph G has a subgraph Q that has maximum degree only $O_{\varepsilon,p}(1)$ and guarantees a $(1 - \varepsilon)$-approximation. That is, the maximum degree of Q depends only on ε and p (both of which are known to be necessary) and not for example on the number of nodes in G, the edge-weights, etc.

The stochastic matching problem has been studied extensively on both weighted and unweighted graphs. Previously, only existence of (close to) half-approximate subgraphs was known for weighted graphs [Yamaguchi and Maehara, SODA'18; Behnezhad *et al.*, SODA'19]. Our result substantially improves over these works, matches the state-of-the-art for unweighted graphs [Behnezhad *et al.*, STOC'20], and settles the approximation factor.

Full version. Due to the page limit, this version of the paper does not include all the proofs. The full version of the paper is available at [4].

1. Introduction

We study the *stochastic weighted matching* problem defined as follows. An arbitrary n-vertex graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{\geq 0}$ is given. A random subgraph $\mathcal{G}$ of G, called the *realization*, is then drawn by retaining each edge $e \in E$ independently with some fixed probability $p \in (0, 1]$. The goal is to choose a subgraph Q of G without knowing the realization $\mathcal{G}$ such that:

1) The maximum weight matching (MWM) among the *realized* edges of Q (i.e. graph $Q \cap \mathcal{G}$) approximates in expectation the MWM of the whole realization $\mathcal{G}$. Formally, we want the "approximation factor" $\mathbb{E}[\mu(Q \cap \mathcal{G})]/\mathbb{E}[\mu(\mathcal{G})]$ to be large where $\mu(H)$ denotes the weight of the MWM of H.
2) The subgraph Q has maximum degree $O(1)$. The constant here can (and in fact must) depend on p, but cannot depend on the structure of G such as the number of nodes or edge-weights.

Observe that by setting $Q = G$ we get an optimal solution, but the second constraint would be violated as the maximum degree in G could be very large. On the other hand, if we choose Q to be a single maximum weight matching of G, the maximum degree in Q would desirably be only one, but it is not possible to guarantee anything better than a p-approximation for this algorithm[1]. The stochastic matching problem therefore essentially asks whether it is possible to interpolate between these two extremes and pick a subgraph that is both sparse and provides a good approximation.

Applications. As its most straightforward application, the stochastic matching problem can be used as a *matching sparsifier* that approximately preserves the maximum (weight) matching under random edge failures [1]. It also has various applications in e.g. kidney exchange (see [10] for an extensive discussion) and online labor markets [8, 7]. For these applications, one is only given the base graph G but is tasked to find a matching in the realized subgraph $\mathcal{G}$. To do so, an algorithm can *query* each edge of G to see whether it is realized. Each of these queries typically maps to a time-consuming operation such as interviewing a candidate and thus few queries must be conducted. To do so, one can (non-adaptively) query only the $O(n)$ edges of Q and still expect to find an approximate MWM in the whole realization $\mathcal{G}$ which note may have $\Omega(n^2)$ edges.

Known bounds. As surveyed in Table I, both the weighted and unweighted variants of this problem have been studied extensively [9, 2, 3, 15, 8, 7, 1, 5, 13, 6] since the pioneering work of Blum *et al.* [9] who introduced the problem. For the unweighted case, early works achieved close to half approximation [9, 2, 3]. The second wave of results came close to 0.66-approximation [7, 1]. Eventually, it was shown in [6] that the approximation factor can be made $(1 - \varepsilon)$ for any constant $\varepsilon > 0$ [6]. All these results rely heavily on the underlying graph being unweighted.

For the weighted case, in contrast, all known results

[1]To see this, let G be a clique with unit weights. It is easy to prove that for fixed $p > 0$, a realization of G has a near-perfect matching with high probability, whereas only p fraction of the edges in the matching that forms Q are realized.

	Reference	Approx	Degree of Q	Notes
Unweighted	Blum, Dickerson, Haghtalab, Procaccia, Sandholm, & Sharma [9, 10]	$0.5 - \varepsilon$	$O_{\varepsilon,p}(1)$	
	Assadi, Khanna, & Li [2]	$0.5 - \varepsilon$	$O_{\varepsilon,p}(1)$	
	Assadi, Khanna, & Li [3]	0.5001	$O_p(1)$	
	Behnezhad, Farhadi, Hajiaghayi, & Reyhani [7]	0.6568	$O_p(1)$	
	Assadi & Bernstein [1]	$2/3 - \varepsilon$	$O_{\varepsilon,p}(1)$	
	Behnezhad, Derakhshan, & Hajiaghayi [6]	$1 - \varepsilon$	$O_{\varepsilon,p}(1)$	
Weighted	Yamaguchi & Maehara [15]	$0.5 - \varepsilon$	$O_{\varepsilon,p}(W \log n)$	
	Yamaguchi & Maehara [15]	$0.5 - \varepsilon$	$O_{\varepsilon,p}(W)$	Bipartite
	Behnezhad & Reyhani [8]	$0.5 - \varepsilon$	$O_{\varepsilon,p}(1)$	
	Behnezhad, Farhadi, Hajiaghayi, & Reyhani [7]	0.501	$O_p(1)$	
	This work	$1 - \varepsilon$	$O_{\varepsilon,p}(1)$	

Table I: Survey of known results for weighted and unweighted graphs in chronological order. For simplicity we have hidden the actual dependence on ε and p inside the O-notation. In the bounds above, W denotes the maximum edge-weight after scaling all the weights to integers.

remain close to half-approximation. The first result of this kind was proved by [15] who showed that by allowing Q's maximum degree to depend on the maximum weight W, one can obtain a 0.5-approximation. It was later proved in [8] through a different analysis of the same construction that dependence on W is not necessary to achieve a 0.5-approximation. Subsequently, the approximation factor was slightly improved to 0.501 using a different construction [7].

Our contribution. Our main result in this paper is as follows:

Theorem 1. *For any weighted graph G, any $p \in (0, 1]$, and any $\varepsilon > 0$, there is a subgraph Q of G with maximum degree $O_{\varepsilon,p}(1)$ that achieves a $(1-\varepsilon)$-approximation for the stochastic weighted matching problem.*

Not only Theorem 1 is the first result showing that a significantly better than 0.5-approximation is achievable for weighted graphs, but it essentially settles the approximation ratio. The remark below shows also that the dependence of the maximum degree of Q on both ε and p is necessary:

Remark 1.1. *For any $\varepsilon > 0$, any $(1-\varepsilon)$-approximate subgraph Q must have maximum degree $\Omega(\frac{\log 1/\varepsilon}{p})$ even when G is a unit-weight clique [2]. This also implies that the approximation ratio cannot be made $(1 - o(1))$ unless Q has $\omega(1)$ degree.*

For simplicity, we do not calculate the precise dependence of the maximum degree of Q on ε and p in this version of the paper. However, we remark that the $O_{\varepsilon,p}(1)$ term in Theorem 1 hides an exponential dependence on ε and p. We leave it as an open problem to determine whether a $\text{poly}(\frac{1}{\varepsilon p})$ degree subgraph can also achieve a $(1-\varepsilon)$-approximation.

2. Technical Overview & the Challenge with Weighted Graphs

In the literature of the stochastic matching problem, the subgraph Q typically has a very simple construction and much of the effort is on analyzing its approximation factor. A good starting point is the following Sampling algorithm proposed in [7]:[2] For some parameter $R = O_{\varepsilon,p}(1)$, draw R independent realizations $\mathcal{G}_1, \ldots, \mathcal{G}_R$ of G and let $Q \leftarrow \mathsf{MM}(\mathcal{G}_1) \cup \ldots \cup \mathsf{MM}(\mathcal{G}_R)$ where here $\mathsf{MM}(\cdot)$ returns an arbitrary maximum weight matching. It is clear that the maximum degree of Q is $R = O_{\varepsilon,p}(1)$ since each matching $\mathsf{MM}(\mathcal{G}_i)$ has maximum degree one; but what approximation does this construction of Q guarantee? Clearly $\mathbb{E}[\mu(\mathcal{G}_i)] = \mathbb{E}[\mu(\mathcal{G})]$ since each $\mathcal{G}_i$ is drawn from the same distribution as $\mathcal{G}$. However, observe that only p fraction of the edges of each matching $\mathsf{MM}(\mathcal{G}_i)$ in expectation appear in the actual realization $\mathcal{G}$. Hence, the challenge in the analysis is to show that the realized edges of these matchings can augment each other to construct a matching whose weight approximates $\textsc{opt} := \mathbb{E}[\mu(\mathcal{G})]$.

Since the weighted stochastic matching problem is a generalization of the unweighted version, all the challenges that occur for the unweighted variant carry over to the weighted case. Of key importance, is the so called "Ruzsa-Szemerédi barrier" which was first observed by [2] toward achieving a $(1-\varepsilon)$-approximation and was later broken in [6] for unweighted graphs using a notion of "vertex independent matchings" which we generalize to weighted graphs. Since the main contribution of this paper is solving the weighted version of the problem, we do not elaborate more on this barrier in this section and refer interested readers to Sections 1 and 2 of [6]. Instead, we discuss two challenges specific to weighted graphs and how we overcome them.

[2]As we will soon discuss, we do not analyze just the Sampling algorithm in this work, and combine it with a Greedy algorithm stated formally as Algorithm 1.

Challenge 1: Low-probability/high-weight edges. The analysis of the $\mathsf{Sampling}$ algorithm for unweighted graphs typically (see [7, 6]) relies on a partitioning of the edge-set E into "crucial" and "non-crucial" edges: Define $q_e := \Pr[e \in \mathsf{MM}(\mathcal{G})]$ and let $\tau = \tau(\varepsilon, p) \ll p$ be a sufficiently small threshold; an edge e is called "crucial" if $q_e \geq \tau$ and "non-crucial" if $q_e < \tau$. If we draw say $R = \frac{\log 1/\varepsilon}{\tau} = O_{\varepsilon,p}(1)$ realizations in the $\mathsf{Sampling}$ algorithm, simple calculations show that each crucial edge appears in at least one of $\mathsf{MM}(\mathcal{G}_1), \ldots, \mathsf{MM}(\mathcal{G}_R)$ and thus to Q, with probability at least $1-\varepsilon$. As a result, the subgraph Q only loses on non-crucial edges. To argue that this loss on the non-crucial edges does not hurt the expected size of the maximum realized matching in $\mathcal{Q}$ compared to that of $\mathcal{G}$, a key observation made in in the prior work, is that roughly speaking excluding any small subset of non-crucial edges from the graph does not hurt the matching size, since for unweighted graphs the contribution of each edge to the expected matching is precisely the probability that it is part of it, which is small for non-crucial edges.

For weighted graphs there is a third class of edges: Edges e with a small probability q_e of appearing in $\mathsf{MM}(\mathcal{G})$ but a relatively large weight w_e. On one hand, there could be a super-constant number of these edges connected to each vertex, so we cannot consider them crucial and add all of them to Q. On the other hand, even "ignoring" few edges of this type can significantly hurt the weight of the matching, so they cannot be regarded as non-crucial. This is precisely the reason that the analysis of [7] only guarantees a 0.501-approximation for weighted graphs but achieves up to 0.65-approximation for unweighted graphs. (See [7, Section 6] and in particular Figure 4 of [7].)

We handle low-probability/high-weight edges in a novel way. Particularly, we complement the $\mathsf{Sampling}$ algorithm (stated as Algorithm 2) with a Greedy algorithm (stated as Algorithm 1) which picks *some* of the low-probability high-weight edges and adds them to Q. Then in our analysis, any low-probability/high-weight edge that is picked by the Greedy algorithm is treated as if they are crucial, while the rest are regarded as non-crucial.

Challenge 2: Lack of the "sparsification lemma" for weighted graphs. Let us for now suppose that graph G is unweighted. It is often useful to assume $\mathbb{E}[\mu(\mathcal{G})] = \Omega(n)$ as for instance even by losing an additive εn factor in the size of the matching (say because a certain event fails around each vertex with probability ε), we can still guarantee a multiplicative $(1-O(\varepsilon))$-approximation. A "sparsification lemma" of Assadi *et al.* [2] which was also used in a crucial way in [6] guarantees that this assumption comes without loss of generality for unweighted graphs. This is achieved by modifying the graph and ensuring that each vertex is matched with a large probability. Specifically, if the expected size of the matching is $\text{OPT} \ll n$, the algorithm randomly contracts vertices into OPT/ε buckets and the resulting graph will now have $O(\text{OPT}/\varepsilon)$ vertices and still a matching of size at least $(1-\varepsilon)\text{OPT}$.

For weighted graphs, in contrast, the probability with which a vertex is matched is not a useful indicator of the weight that it contributes to the matching. For this reason, no equivalent of the sparsification lemma exists for weighted graphs. For another evidence that the sparsification lemma is not useful for weighted graphs, observe that by adding zero-weight edges we can assume w.l.o.g. that G is a clique. Therefore, each vertex v already has a probability $1 - o(1)$ of being matched (but perhaps via a zero-weight edge) and thus the reduction of [2] discussed above does not help.

Due to lack of the sparsification lemma, it is not sufficient to simply bound the probability of a "bad event" around each vertex by say ε when the graph is weighted. Rather, it is important to analyze the actual expected loss to the weight conditioned on that this bad event occurs. For this reason, our analysis turns out to be much more involved than the unweighted case. This appears both in generalizing the vertex-independent lemma (Section 5) to the weighted case, and in various other places in the analysis.

3. Basic Definitions and The Algorithm

3.1. General Notation

For any matching M, we use $w(M) := \sum_{e \in M} w_e$ to denote the weight of M and use $v \in M$ for any vertex v to indicate that there is an edge incident to v that belongs to M. We use $\mu(H)$ to denote the weight of the maximum weight matching in graph H. For any two vertices u and v, we use $d_G(u, v)$ to denote the size of the shortest path between u and v in graph G (note that this is not their weighted distance). For any event A, we use $\mathbf{1}(A)$ as the indicator of the event, i.e. $\mathbf{1}(A) = 1$ if A occurs and $\mathbf{1}(A) = 0$ otherwise.

3.2. Basic Stochastic Matching Notation/Definitions

We use OPT to denote $\mathbb{E}[\mu(\mathcal{G})]$. Note that OPT is just a number, the expected weight of the maximum weight matching in the realization $\mathcal{G}$. With this notation, to prove Theorem 1, we should prove that $\mathbb{E}[\mu(\mathcal{Q})] \geq (1-\varepsilon)\text{OPT}$, where $\mathcal{Q} := Q \cap \mathcal{G}$ is the realized subgraph of Q.

For any graph H, we use $\mathsf{MM}(H)$ to denote a maximum weight matching of H. In case H has multiple maximum weight matchings, $\mathsf{MM}(H)$ returns an arbitrary one. It would be useful to think of $\mathsf{MM}(\cdot)$ as a *deterministic* maximum weight matching algorithm that always returns the same matching for any specific input graph. Having this, for each edge e define

$$q_e := \Pr_{\mathcal{G}}[e \in \mathsf{MM}(\mathcal{G})] \qquad \text{and} \qquad \chi_e = w_e \cdot q_e. \quad (1)$$

Observe that χ_e is the expected weight that e contributes to matching $\mathsf{MM}(\mathcal{G})$. These definitions also naturally extend to subsets of edges $F \subseteq E$ for which we denote

$$q(F) := \sum_{e \in F} q_e, \qquad \text{and} \qquad \chi(F) := \sum_{e \in F} \chi_e.$$

The following equality is an easy consequence of definitions above.

Observation 3.1. $\chi(E) = \text{OPT}$.

Proof: By definition $\text{OPT} = \mathbb{E}[\mu(\mathcal{G})]$. The proof therefore follows since:

$$\begin{aligned} \mathbb{E}[\mu(\mathcal{G})] = \mathbb{E}[w(\mathsf{MM}(\mathcal{G}))] &= \mathbb{E}\left[\sum_{e \in \mathsf{MM}(\mathcal{G})} w_e\right] \\ &= \mathbb{E}\left[\sum_{e \in E} \mathbf{1}(e \in \mathsf{MM}(\mathcal{G})) \cdot w_e\right] \\ &= \sum_{e \in E} \Pr[e \in \mathsf{MM}(\mathcal{G})] \cdot w_e \\ &= \sum_{e \in E} q_e \cdot w_e = \sum_{e \in E} \chi_e = \chi(E), \end{aligned}$$

where the fourth equality follows simply from linearity of expectation. ■

3.3. The Algorithm

In what follows we describe two different algorithms that each picks a subgraph of graph G. The final subgraph Q is the union of the two subgraphs picked by these algorithms.

To state the first algorithm, let us first define function $\lambda : \mathbb{R} \times [0, 1] \to \mathbb{R}$ as:

$$\lambda(\Delta, \varepsilon) := \varepsilon^{-24}(\log \Delta)(\log \log \Delta)^C, \qquad (2)$$

where $C \geq 1$ is a large enough absolute constants that we fix later. This perhaps strange-looking function is defined in this way so that it satisfies the various equations that we will need throughout the analysis. Having it, the first algorithm we use is as follows:

Algorithm 1. GreedyAlgorithm$(G = (V, E), p, \varepsilon)$

```
1  $P \leftarrow \emptyset$.
2  while true do
3      $\Delta \leftarrow \max\{1, \text{max degree in subgraph } P\}$.
         // So in the first iteration, $\Delta = 1$.
4      $I_q \leftarrow \{(u, v) \in E \setminus P \mid q_e \geq p^2\varepsilon^{10} \cdot \Delta^{-\lambda(\Delta,\varepsilon)}\}$.
5      $I_d \leftarrow \{(u, v) \in E \setminus P \mid d_P(u, v) < \lambda(\Delta, \varepsilon)\}$.
6      $I \leftarrow I_d \cup I_q$.
7      if $\chi(I) \geq \varepsilon\text{OPT}$ then
8          $P \leftarrow P \cup I$.
9      else
10         return $P$.
```

From now on, when we use Δ we refer to the final value assigned to it during Algorithm 1, which is equivalent to the maximum degree of P (unless P remains empty, which in that case $\Delta = 1$).

Remark 3.2. *Algorithm 1 uses the value of q_e which is not given a priori. Naively, it can be computed for all edges by enumerating all possible realizations of G in exponential time. However, it is not hard to see that we only need to check $q_e > \tau$ where τ is a constant dependent on ε and p, hence a simple Monte Carlo algorithm can also find all the edges in I_q in polynomial time with high probability.*

The second algorithm which was proposed first in [7] is very simple and natural: Draw multiple random realizations and pick a maximum weight matching of each; formally:

Algorithm 2. SamplingAlgorithm$(G = (V, E), p, \varepsilon)$

```
1  $R \leftarrow \lceil p^{-2}\varepsilon^{-10}\Delta^{\lambda(\Delta,\varepsilon)} \rceil$.
2  for $i$ in $1 \ldots R$ do
3      Draw a realization $\mathcal{G}_i$ by retaining each
         edge $e \in E$ independently with
         probability $p$.
4  return $S := \mathsf{MM}(\mathcal{G}_1) \cup \ldots \cup \mathsf{MM}(\mathcal{G}_R)$.
```

As mentioned earlier, the final subgraph Q is the union of the outputs of Algorithms 1 and 2. That is, $Q = S \cup P$. The following lemma states that the algorithms terminate and that the maximum degree of Q is $O_{\varepsilon,p}(1)$.

Lemma 3.3. *Algorithms 1 and 2 terminate and the subgraph Q has maximum degree $O_{\varepsilon,p}(1)$.*

Proof: Algorithm 1 has an unconditional while loop, but we argue that it will terminate within at most $1/\varepsilon$ iterations. To see this, consider the progress of $\chi(P)$ after each iteration. Since none of the edges in I are in P due to its definition, in every iteration that the condition $\chi(I) \geq \varepsilon\text{OPT}$ of Line 7 holds, the value of

$\chi(P)$ increases by at least εOPT. On the other hand, since $P \subseteq E$ and $\chi(E) = $ OPT (Observation 3.1), we have $\chi(P) \leq$ OPT. Hence, after at most $1/\varepsilon$ iterations, the condition of Line 7 cannot continue to hold and the algorithm returns P. Algorithm 2 also clearly terminates as it simply runs a for loop finitely many times.

To bound the maximum degree of Q by $O_{\varepsilon,p}(1)$ we show that it suffices to bound the maximum degree Δ of P by $O_{\varepsilon,p}(1)$. To see this, first observe that if $\Delta = O_{\varepsilon,p}(1)$ then also $\lambda(\Delta, \varepsilon) = O_{\varepsilon,p}(1)$ by definition of λ. On the other hand, since S is simply the union of $R = O(p^{-2}\varepsilon^{-10}\Delta^{\lambda(\Delta,\varepsilon)})$ matchings, its maximum degree can also be bounded by $O(p^{-2}\varepsilon^{-10}\Delta^{\lambda(\Delta,\varepsilon)}) = O_{\varepsilon,p}(1)$. It thus only remains to prove $\Delta = O_{\varepsilon,p}(1)$.

To bound Δ, let Δ_i be the maximum degree of P by the end of iteration i of the while loop in Algorithm 1. We prove via induction that for any $i \leq 1/\varepsilon$ we have $\Delta_i = O_{\varepsilon,p}(1)$. This is sufficient for our purpose since we already showed above that the algorithm terminates within $1/\varepsilon$ iterations.

For the base case with $i = 0$ (i.e. before the start of the while loop) P is empty, hence indeed $\Delta_0 = O_{\varepsilon,p}(1)$. Now consider any iteration i. Take any vertex v and let $e = (u, v)$ be an edge that belongs to I at iteration i. By definition of I in Line 7, $e \in I_d \cup I_q$ so it remains to bound the maximum degree of I_d and I_q. If $e \in I_d$, there should be a path between u and v consisting of only the edges already in P that has length less than $\ell := \lambda(\Delta_{i-1}, \varepsilon)$. Since the maximum degree in P at this point is Δ_{i-1}, there are at most Δ_{i-1}^{ℓ} such paths ending at v. This is a simple upper bound on the number of edges in I_d connected to v at iteration i. On the other hand, if $e \in I_q$, then by definition $q_e \geq p^2\varepsilon^{10} \cdot \Delta_{i-1}^{-\ell}$. Combined with $\sum_{e \ni v} q_e \leq 1$, this means there are at most $p^{-2}\varepsilon^{-10} \cdot \Delta_{i-1}^{\ell}$ edges in I_q connected to v. Thus the degree of any vertex v increases by at most $\Delta_{i-1}^{\ell} + p^{-2}\varepsilon^{-10}\Delta_{i-1}^{\ell}$ and as a result:

$$\Delta_i \leq \Delta_{i-1} + \Delta_{i-1}^{\ell} + p^{-2}\varepsilon^{-10}\Delta_{i-1}^{\ell}.$$

By the induction hypothesis, $\Delta_{i-1} = O_{\varepsilon,p}(1)$ which also consequently implies $\ell = O_{\varepsilon,p}(1)$ since ℓ is a function of only Δ_{i-1} and ε. Therefore, $\Delta_i \leq O_{\varepsilon,p}(1)^{O_{\varepsilon,p}(1)} = O_{\varepsilon,p}(1)$. Observe that since $i \leq 1/\varepsilon$, this use of the asymptotic notation over the steps of the inductive argument does not lead to any undesirable blow-up and the final maximum degree is indeed $O_{\varepsilon,p}(1)$ as desired. ■

4. The Analysis

In this section, we analyze the approximation factor of the construction of Q described in the previous section.

Analysis via fractional matchings. Recall that our goal is to show graph $\mathcal{Q} := Q \cap \mathcal{G}$ has a matching of weight $(1 - O(\varepsilon))$OPT in expectation. Since Q is constructed independently from the realization $\mathcal{G}$, one can think of $\mathcal{Q}$ as a subgraph of Q that includes each edge of Q independently with probability p. To show this subgraph $\mathcal{Q}$ has a matching of weight close to OPT, we follow the by now standard recipe [7, 6] of constructing a fractional matching $\mathbf{x}$ on $\mathcal{Q}$, such that:

$$x_v := \sum_{e \ni v} x_e \leq 1 \qquad \forall v \in V \tag{3}$$

$$x_e \geq 0 \qquad \forall e \in \mathcal{Q} \tag{4}$$

$$x(U) := \sum_{e=(u,v):u,v \in U} x_e \leq \frac{|U|-1}{2} \qquad \forall U \subseteq V \text{ such that } |U| \text{ is odd and } \leq 1/\varepsilon. \tag{5}$$

Here (3) and (4) are simply fractional matching constraints. The last set of constraints (5), known as "blossom" [11] constraints, are needed to ensure that our fractional matching $\mathbf{x}$ can be turned into an integral matching of weight at least $(1 - \varepsilon)$ times that of $\mathbf{x}$. (See [14, Section 25.2] for more context on the matching polytope and blossom constraints. See also [7, Section 2.2] for a simple proof of this folklore lemma that blossom inequalities over subsets of size up to $1/\varepsilon$ are sufficient for a $(1-\varepsilon)$-approximation.) In addition to the constraints above, we want fractional matching $\mathbf{x}$ to have weight close to OPT so that we can argue $\mathcal{Q}$ has an integral matching of size $(1 - O(\varepsilon))$OPT. Formally, our goal is to construct $\mathbf{x}$ such that in addition to constraints (3–5), it satisfies:

$$\mathbb{E}\left[\sum_{e \in \mathcal{Q}} x_e w_e\right] \geq (1 - O(\varepsilon))\text{OPT}. \tag{6}$$

If $\mathbf{x}$ satisfies all these constraints, then we have $\mathbb{E}[\mu(\mathcal{Q})] \geq (1 - O(\varepsilon))$OPT, proving Theorem 1.

Observation 4.1. *To prove Theorem 1, it suffices to give a construction $\mathbf{x} : \mathcal{Q} \to [0, 1]$ satisfying constraints (3-5) and (6).*

The natural idea of using fractional matchings to analyze a solution for the stochastic matching problem was first used in [7] and later in [6]. Among the two, only [7] deals with weighted graphs, but there the constructed fractional matching is only shown to have an expected weight of at least 0.501OPT, guaranteeing only a 0.501-approximation. Here, not only our subgraph Q is constructed differently, but the way we construct the fractional matching is also fundamentally different and allows us to satisfy (6) and guarantee a $(1 - \varepsilon)$-approximation.

4.1. Toward Constructing x: A Partitioning of E

To construct fractional matching $\mathbf{x}$, we first partition the edge set E into $P \cup I' \cup N$, where P is simply the output of Algorithm 1, I' is the set of edges in set I defined in the last iteration of Algorithm 1 (for which the condition $\chi(I) \geq \varepsilon\text{OPT}$ of Line 7 fails), and N is the rest of edges, i.e. $N = E - P - I'$. On all edges $e \in I'$ we simply set $x_e = 0$, i.e., we do not use them in the fractional matching $\mathbf{x}$. For other edges $e \notin I'$, we use different constructions for $\mathbf{x}$ depending on whether $e \in P$ or $e \in N$. We describe the construction of $\mathbf{x}$ on P in Section 4-B and the construction on N in Section 4-C. Before that, let us state a number of simple observations regarding this partitioning.

Observation 4.2. $\chi(P) + \chi(N) \geq (1-\varepsilon)\text{OPT}$.

Proof: Recall that $\chi(E) = \text{OPT}$ by Observation 3.1. Combined with $E = P \cup I' \cup N$, this implies $\chi(P) + \chi(N) + \chi(I') \geq \text{OPT}$. To complete the proof, we argue that $\chi(I') \leq \varepsilon\text{OPT}$. To see this, recall that I' is defined as the set I in the last iteration of Algorithm 1. In the last iteration, the condition $\chi(I) \geq \varepsilon\text{OPT}$ of Line 7 in Algorithm 1 must fail (otherwise there would be another iteration), and thus $\chi(I') < \varepsilon\text{OPT}$. ■

Observation 4.3. *For any edge* $e = (u, v) \in N$, $q_e < p^2\varepsilon^{10}\Delta^{-\lambda(\Delta,\varepsilon)}$ *and* $d_P(u, v) \geq \lambda(\Delta, \varepsilon)$.

Proof: In the last iteration of Algorithm 1, all edges $e = (u, v)$ with $q_e \geq p^2\varepsilon^{10}\Delta^{-\lambda(\Delta,\varepsilon)}$ or $d_P(u, v) < \lambda(\Delta, \varepsilon)$ are either already in P or are added to $I = I'$; thus $e \notin N$ since $N = E - P - I'$. ■

4.2. Construction of the Fractional Matching x on P

To describe the construction, let us first state a "vertex-independent matching lemma" which we will prove in Section 5.

Lemma 4.4. *Let* $G' = (V', E', w')$ *be an edge-weighted base graph with maximum degree* Δ'. *Let* $\mathcal{G}'$ *be a random subgraph of* G' *that includes each edge* $e \in E'$ *independently with some probability* $p \in (0, 1]$. *Let* $\mathcal{A}(H)$ *be any (possibly randomized) algorithm that given any subgraph* H *of* G', *returns a (not necessarily maximum weight) matching of* H. *For any* $\varepsilon > 0$ *there is a randomized algorithm* $\mathcal{B}$ *to construct a matching* $Z = \mathcal{B}(\mathcal{G}')$ *of* $\mathcal{G}'$ *such that*

1) *For any vertex* v, $\Pr_{\mathcal{G}'\sim G',\mathcal{B}}[v \in Z] \leq \Pr_{\mathcal{G}'\sim G',\mathcal{A}}[v \in \mathcal{A}(\mathcal{G}')] + \varepsilon^3$.
2) $\mathbb{E}[w(Z)] \geq (1-\varepsilon)\mathbb{E}[w(\mathcal{A}(\mathcal{G}'))]$.
3) *For any vertex-subset* $\{v_1, v_2, \ldots\} \subseteq V'$ *such that for all* i, j, $d_{G'}(v_i, v_j) \geq \lambda$ *where* $\lambda = O(\varepsilon^{-24}\log\Delta' \operatorname{poly}(\log\log\Delta'))$, *events* $\{v_1 \in Z\}, \{v_2 \in Z\}, \{v_3 \in Z\}, \ldots$ *are all independent with respect to both the randomizations used in algorithm* $\mathcal{B}$ *and in drawing* $\mathcal{G}'$.

We use this lemma in the following way: The graph $G' = (V', E', w')$ of the lemma, is simply the subgraph P picked by Algorithm 1 and thus Δ' is simply the maximum degree of P which recall we denote by Δ. We let the random subgraph $\mathcal{G}'$ be the subset of edges in P that are realized, which we denote by $\mathcal{P}$. As discussed before, since P is chosen independently from how the edges are realized, conditioned on P each edge is still realized independently from the others, so the assumption that $\mathcal{P}$ is a random subgraph of P with edges realized independently is valid. Finally, we define the algorithm $\mathcal{A}(H)$ of the lemma for any subgraph $H \subseteq P$ as follows:

Algorithm 3. $\mathcal{A}(H)$

```
1 $H' \leftarrow H$.
2 Add any edge $e \in E \setminus P$ independently with
    probability $p$ to $H'$.
3 return $\mathsf{MM}(H') \cap H$.
```

Observe that with definition above, $\mathcal{A}(\mathcal{P})$ can be interpreted in the following useful way: The input subgraph $H = \mathcal{P}$ already includes each edge of P independently with probability p. Since initially $H' \leftarrow H$, and every edge $e \in E \setminus P$ is then added to H' independently with probability p, by the end of Line 2, H' will have the same distribution as the realization $\mathcal{G}$ of G. This means:

Observation 4.5. *The output of* $\mathcal{A}(\mathcal{P})$ *has the same distribution as* $\mathsf{MM}(\mathcal{G}) \cap P$.

Finally, once we obtain a matching using the algorithm above, we remove each edge from the matching independently with probability ε. Doing so, we only lose ε fraction of the weight of the matching in expectation, but we ensure that each vertex is matched with probability at most $1 - \varepsilon$ which will be useful later.

Let us for each vertex v define $q_v^P := \sum_{e:v\in e, e\in P} q_e$ to be the probability that v is matched in $\mathsf{MM}(\mathcal{G})$ via an edge in P. Using Lemma 4.4, the discussion above leads to the following statement (see the full version for the proof [4]).

Claim 4.6. *There is an algorithm* $\mathcal{B}$ *to construct a matching* Z *on the realized edges* $\mathcal{P}$ *of* P *s.t.:*

1) *For any vertex* v, $\Pr_{\mathcal{P},\mathcal{B}}[v \in Z] \leq \min\{q_v^P + \varepsilon^3, 1-\varepsilon\}$.
2) $\mathbb{E}[w(Z)] \geq (1-2\varepsilon)\chi(P)$.
3) *For any vertex-subset* $\{v_1, v_2, \ldots\} \subseteq V$ *such that for all* i, j, $d_P(v_i, v_j) \geq \lambda(\Delta, \varepsilon)$, *events* $\{v_1 \in Z\}, \{v_2 \in Z\}, \{v_3 \in Z\}, \ldots$ *are all independent with respect to both the randomizations used in algorithm* $\mathcal{B}$ *and the randomization in drawing* $\mathcal{P}$.
4) *Matching* Z *is independent of the realization of edges in* $E \setminus P$.

Once we construct matching Z on the realized edges of P using the algorithm above, for any edge $e \in P$ we set $x_e = 1$ if $e \in Z$ and $x_e = 0$ otherwise. Therefore, $\mathbf{x}$ is in fact integral on all edges of P. The properties of Z highlighted in Claim 4.6 will be later used in augmenting $\mathbf{x}$ via the realized edges among the edges in N.

4.3. Construction of the Fractional Matching x on N

We first formally describe construction of $\mathbf{x}$ on the edges in N, then discuss the main intuitions behind the construction.

1) The Construction: We give a step by step construction for $\mathbf{x}$. We first define an "assignment" $\mathbf{f} : E \to [0,1]$, then based on $\mathbf{f}$ define an assignment $\mathbf{g} : E \to [0,1]$, then based on $\mathbf{g}$ define an assignment $\mathbf{h} : E \to [0,1]$, and finally construct $\mathbf{x}$ from $\mathbf{h}$. For any assignment $\mathbf{a} \in \{\mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{x}\}$ we may use the following notation: For an edge e, a_e denotes the value of $\mathbf{a}$ on edge e. For a vertex v, $a_v := \sum_{e \ni v} a_e$ denotes the sum of assignments adjacent to v. The weight $w(\mathbf{a})$ denotes $\sum_{e \in E} a_e w_e$.

As outlined above, we first define $\mathbf{f} : E \to [0,1]$ on each edge e as follows:

$$f_e := \begin{cases} \frac{1}{R}\sum_{i=1}^{R} \mathbf{1}(e \in \mathsf{MM}(\mathcal{G}_i)) & \text{if } e \in N, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where recall that $\mathcal{G}_i$ is the ith drawn realization in Algorithm 2 and R is the total number of realizations drawn in Algorithm 2. In words, for any $e \in N$, the value of f_e denotes the fraction of matchings $\mathsf{MM}(\mathcal{G}_1), \ldots, \mathsf{MM}(\mathcal{G}_R)$ that include e.

Based on $\mathbf{f}$, we define $\mathbf{g}$ on each $e = (u,v)$ as:

$$g_e := \begin{cases} f_e & \text{if } f_e \le p^2\varepsilon^7\Delta^{-\lambda(\Delta,\varepsilon)},\ f_u \le 1 - q_u^P + \varepsilon^3, \\ & \text{and } f_v \le 1 - q_v^P + \varepsilon^3, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Next, based on $\mathbf{g}$, we define $\mathbf{h}$ on each edge $e = (u,v)$ as:

$$h_e := \begin{cases} \frac{g_e}{p\Pr[v \notin Z]\Pr[u \notin Z]} & \text{if } u \notin Z,\ v \notin Z, \\ & \text{and } e \text{ is realized} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Here, as defined in the previous section, the value of q_v^P for a vertex v denotes the probability that v is matched in $\mathsf{MM}(\mathcal{G})$ via an edge in P.

We are finally ready to define the construction of $\mathbf{x}$ on N. On each edge $e = (u,v) \in N$, we set:

$$x_e \leftarrow \begin{cases} \frac{h_e}{1+3\varepsilon} & \text{if } h_v \le 1+3\varepsilon \text{ and } h_u \le 1+3\varepsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

2) Intuitions and Proof Outline: Here we discuss the main intuitions behind the construction above for $\mathbf{x}$ on N in a slightly informal way. The rigorous proof that the final fractional matching $\mathbf{x}$ satisfies properties (3-6) is given in the forthcoming sections.

As mentioned above, for every edge $e \in N$, f_e simply denotes the fraction of matchings $\mathsf{MM}(\mathcal{G}_1), \ldots, \mathsf{MM}(\mathcal{G}_R)$ that include e. Therefore $\mathbf{f}$ is a linear combination of these integral matchings, and thus is a valid fractional matching. Another key observation here is that since each $\mathcal{G}_i$ has the same distribution as $\mathcal{G}$, the probability of each edge e appearing in each matching $\mathsf{MM}(\mathcal{G}_i)$ is exactly equal to the probability q_e that it appears in $\mathsf{MM}(\mathcal{G})$. This can be used to prove $\mathbb{E}[f_e] = q_e$ which also implies $\mathbb{E}[w(\mathbf{f})] = \chi(N)$. Thus, fractional matching $\mathbf{f}$ has precisely the weight $\chi(N)$ we need $\mathbf{x}$ to have on N. In addition (unlike q_e) the value of f_e is only non-zero on edges $e \in N$ that also belong to the output S of Algorithm 2. This is desirable since recall that if an edge $e \in N$ does not belong to S, then $e \notin Q$ and as a result $e \notin \mathcal{Q}$. Thus, we should ensure $x_e = 0$ since we want $\mathbf{x}$ to be a fractional matching of subgraph $\mathcal{Q}$.

In the next step of the construction, we define $\mathbf{g}$ based on $\mathbf{f}$. The key idea behind this definition is to get rid of possible "deviations" in $\mathbf{f}$ and ensure that $\mathbf{g}$ satisfies certain deterministic inequalities for g_e on all edges e, and g_v for all vertices v. It turns out that by carefully bounding the probability of these deviations, we can still argue that $\mathbf{g}$ has weight close to $\chi(N)$ just like $\mathbf{f}$.

Despite the desirable properties mentioned above, $\mathbf{g}$ is still far from the values we would like to assign to edges N in $\mathbf{x}$, for the following two reasons. First, we want $\mathbf{x}$ to be non-zero only on $\mathcal{Q}$, i.e. the realized edges in Q. However, in defining $\mathbf{g}$ we never look at edge realizations. Hence, it could be that $g_e > 0$ for an edge e that is not realized. The second problem is that we need to augment the matching Z already constructed in Section 4-B. More specifically, recall from Section 4-B that we have already assigned $x_e = 1$ to any edge $e \in Z$. Therefore, if we want $\mathbf{x}$ to be a valid fractional matching, all edges e that are incident to a matched vertex of Z should have $x_e = 0$. In defining $\mathbf{h}$, we address both issues at the same time. That is, for any edge e, if e is not realized or at least one of its endpoints is matched in Z, we set $h_e = 0$. Though note that we still want $\mathbb{E}[w(\mathbf{h})]$ to be close to $\mathbb{E}[w(\mathbf{g})]$ and $\chi(N)$. To compensate for the loss to the weight due to edges e for which $g_e > 0$ but $h_e = 0$, on each edge e that is eligible to be assigned $h_e > 0$, we multiply g_e by an appropriate amount that cancels out the probability of assigning $h_e = 0$. Doing so, we can ensure that $\mathbb{E}[w(\mathbf{h})]$ remains sufficiently close to $w(\mathbf{g})$ and thus $\chi(N)$.

Finally, recall from above that $\mathbf{f}$ is a valid fractional matching and thus so is $\mathbf{g}$ since $g_e \leq f_e$ on all edges. A next challenge is to make sure that once we obtain $\mathbf{h}$ by multiplying $\mathbf{g}$ on some edges, we still have a valid fractional matching. That, e.g. $h_v \leq 1$ for all vertices v. Toward achieving this, we first show that for each vertex v, the probability that $h_v > 1 + 3\varepsilon$ is very small. But these deviations do occur. Thus, in our final construction of $\mathbf{x}$, on any edge $e = (u, v)$ for which at least one of h_u and h_v exceeds $1 + 3\varepsilon$, we set $x_e = 0$ and set $x_e = h_e/(1+3\varepsilon)$ on the rest of the edges. This way, we guarantee that for any vertex v, $x_v \leq 1$. Moreover, due to the low probability of violations in $\mathbf{h}$, there is a small probability for any edge e to have $x_e = 0$ but $h_e > 0$. Therefore, $\mathbf{x}$ as defined, will have weight close to $\chi(N)$ in expectation on the edges in $\mathcal{Q} \cap N$. Combined with the construction of $\mathbf{x}$ on the edges in P which guarantees a weight of $\approx \chi(P)$ there, we obtain that overall $\mathbf{x}$ will have weight close to $\chi(P) + \chi(N)$ which is $\approx$ OPT as guaranteed by Observation 4.2. Therefore, $\mathbf{x}$ can be shown to satisfy all the needed properties required by Observation 4.1 thereby proving Theorem 1.

Due to space limits, we exclude the formal proof that the construction above for $\mathbf{x}$ satisfies the needed properties (3)-(6); the proofs can be found in the full version of the paper [4]. The only part that remains to be discussed is the vertex-independent matching lemma, which we describe next.

5. The Weighted Vertex-Independent Matching Lemma

In this section, we turn to prove Lemma 4.4 which was used in Section 4. We restate the lemma below and for simplicity of notation, drop the primes in symbols such as $G', \mathcal{G}', \Delta'$ as stated in Section 4 and use $G, \mathcal{G}, \Delta$ instead.

Lemma 4.4. (restated)**.** *Let $G = (V, E, w)$ be an edge-weighted base graph with maximum degree Δ. Let $\mathcal{G}$ be a random subgraph of G that includes each edge $e \in E$ independently with some fixed probability $p \in (0, 1]$. Let $\mathcal{A}(H)$ be any (possibly randomized) algorithm that given any subgraph H of G, returns a (not necessarily maximum weight) matching of H. For any $\varepsilon > 0$ there is a randomized algorithm $\mathcal{B}$ to construct a matching $Z = \mathcal{B}(\mathcal{G})$ of $\mathcal{G}$ such that*

1) *For any vertex v, $\Pr_{\mathcal{G}\sim G,\mathcal{B}}[v \in Z] \leq \Pr_{\mathcal{G}\sim G,\mathcal{A}}[v \in \mathcal{A}(\mathcal{G})] + \varepsilon^3$.*
2) *$\mathbb{E}[w(Z)] \geq (1-\varepsilon)\mathbb{E}[w(\mathcal{A}(\mathcal{G}))]$.*
3) *For any vertex-subset $\{v_1, v_2, \ldots\} \subseteq V$ such that for all i, j, $d_G(v_i, v_j) \geq \lambda$ where $\lambda = O(\varepsilon^{-24} \log \Delta \cdot \text{poly}(\log\log\Delta))$, events $\{v_1 \in Z\}, \{v_2 \in Z\}, \{v_3 \in Z\}, \ldots$ are all independent with respect to both the randomizations used in algorithm $\mathcal{B}$ and in drawing $\mathcal{G}$.*

Outline of the proof. To prove this lemma, we need to design an algorithm $\mathcal{B}(\mathcal{G})$ that satisfies all three properties. If we only had the first two properties to satisfy, we could simply use algorithm $\mathcal{A}$. The problem however, becomes challenging when we need to, in addition, satisfy the third property regarding the independence between the events $\{v_1 \in Z\}, \{v_2 \in Z\}, \{v_3 \in Z\}, \ldots$ for vertices $v_1, v_2, \ldots$, that are pair-wise far enough from each other. To ensure that our algorithm meets this condition, as it was done previously in the work of [6] for the unweighted variant of the lemma, we show that it can be implemented efficiently in the LOCAL model of computation (whose formal description follows).

The LOCAL model is a standard distributed computing model which consists of a network (graph) of processors with each processor having its own tape of random bits. Computation proceeds in synchronous rounds and in each round, processors can send unlimited size messages to each of their neighbors. Thus, to transmit a message from a node u to node v, we require at least $d(u, v)$ rounds. For the same reason, if an algorithm terminates within r-rounds of LOCAL, the output of any two nodes that have distance at least $2r$ from each other would be independent, which is essentially how we guarantee our independence property.

For simplicity, we explain our algorithm in a sequential setting in Algorithm 4, and later describe how it can be simulated in the LOCAL model. We define a recursive algorithm $\mathcal{B}_r(\mathcal{G})$ that given a parameter r, as the depth of recursion, and a subgraph of G, denoted by $\mathcal{G}$ outputs a matching of this graph. We give an informal overview of the algorithm in Section 5-A, and formally state it Section 5-B.

Comparison to [6]. For the proof, we follow the general recipe of [6] for the unweighted variant. However, in this work we face several new challenges which make design and the analysis of the algorithm more complicated. Most importantly, the previous work relies on two fundamental observations which do not hold in this work. First, in unweighted graphs, if there exist two constant numbers δ and σ such that for a $(1-\delta)$ fraction of the vertices $v \in V$ the following equation holds

$$\Pr_{\mathcal{G}\sim G,\mathcal{B}}[v \in \mathcal{B}(\mathcal{G})] \geq (1-\sigma) \Pr_{\mathcal{G}\sim G,\mathcal{A}}[v \in \mathcal{A}(\mathcal{G})],$$

then we have $\mathbb{E}[|\mathcal{B}(\mathcal{G})|] \geq (1-\sigma)\mathbb{E}[|\mathcal{A}(\mathcal{G}))|] - \delta n$. Evidently, this only holds for the size of the matching but not for its weight. Second, as a result of the sparsification lemma in the previous work (which we discuss in Section 2), they could assume $|\mathcal{A}(\mathcal{G})| = \Omega(n)$. Subsequently, to prove that $\mathcal{B}(\mathcal{G})$ provides a $(1-\varepsilon)$-approximation, they only needed to show that σ and δ are small enough constants. As we discussed in Section 2, the sparsification lemma does not hold for weighted graphs. Thus, we need to take a different approach in our analysis.

5.1. *Overview of the Algorithm*

We define a recursive algorithm $\mathcal{B}_r(\mathcal{G})$ that given a parameter r, as the depth of recursion, and a subgraph of G, denoted by $\mathcal{G}$ outputs a matching of this graph. The desired algorithm $\mathcal{B}(\mathcal{G})$ of Lemma 4.4 will be $\mathcal{B}_t(\mathcal{G})$ for some $t = O(\varepsilon^{-20})$ to be made precise later. This recursive algorithm is formalized as Algorithm 4 an overview of which we describe in this section.

For $r = 0$, algorithm $\mathcal{B}_0(\mathcal{G})$ simply returns an empty matching. For any $r > 0$, the idea is to use the matching constructed in $\mathcal{B}_{r-1}(\mathcal{G})$ and transform it to another one that is sufficiently heavier in expectation. However, this transformation needs to be in a way that the probability of a vertex being matched in $\mathcal{B}_r(\mathcal{G})$ does not exceed its budget $\Pr_{\mathcal{G}\sim G,\mathcal{A}}[v \in \mathcal{A}(\mathcal{G})]$ imposed by Lemma 4.4 (more precisely, we actually allow a slight slack of an additive ε^3 violation of this budget).

To satisfy vertex budgets, a crucial observation is that for some particular realizations we may allow some vertices to be matched with probability higher than their budget, only if we compensate for it by these vertices having lower probabilities of joining the matching in other realizations. In order to do this, in addition to the actual realization $\mathcal{G}$ of which we need to return a matching, we also draw multiple other (but still constantly many) realizations, and construct a matching on all of these instances. We will then only ensure that on average over these drawn instances, the weight of the matching increases by a constant fraction and that also on average, each vertex is matched with probability no more than that imposed in Lemma 4.4. This way, we have the freedom of matching a vertex with a higher probability than its budget in a specific instance, so long as we compensate for it in other instances. Similarly, we might construct a relatively low-weight matching for an instance if the others are relatively heavy.

To be more precise, in algorithm $\mathcal{B}_r(\mathcal{G})$, we have $\alpha = \varepsilon^{-12} + 1$ random realizations of G all drawn from the same distribution and denoted by $\mathcal{G}_1, \dots, \mathcal{G}_\alpha$, where $\mathcal{G}_1 = \mathcal{G}$ is our special realization. Our goal is to construct matchings $M'_1, \dots, M'_\alpha$ for them simultaneously. Now, since all instances $\mathcal{G}_1, \dots, \mathcal{G}_\alpha$ are drawn from the same distribution, so long as the fraction of instances in which a vertex v is part of the matching is in expectation smaller than its budget, and that the average matchings over these instances are sufficiently heavy in expectation, we can prove Lemma 4.4.

To make things more formal, let us first define "profiles".

Definition 5.1 (profiles)**.** $((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_k, M_k))$ *is a profile of size* k*, iff for any* $i \in [k]$*,* $\mathcal{G}_i$ *is a subgraph of* G *and* M_i *is a matching on* $\mathcal{G}_i$*.*

To construct matchings $M'_1, \dots, M'_\alpha$ for subgraphs $\mathcal{G}_1, \dots, \mathcal{G}_\alpha$ in algorithm $\mathcal{B}_r(\mathcal{G})$, we start by running $\mathcal{B}_{r-1}(\mathcal{G}_i)$ for any $i \in [\alpha]$, and obtain matchings $M_1, \dots, M_\alpha$ as a result. In the other words, we start from profile $((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_\alpha, M_\alpha))$ and transform it to $((\mathcal{G}_1, M'_1), \dots, (\mathcal{G}_\alpha, M'_\alpha))$ such that $\mathbb{E}[w(M'_i)]$ is sufficiently greater than $\mathbb{E}[w(M_i)]$ for a random $i \in [\alpha]$, while the constraints in the second and third properties of Lemma 4.4 are not violated.

To increase the matching weight as discussed above, we use an idea similar to finding augmenting paths in the classic weighted matching algorithms. However, ours rather than being a path over a simple graph, it may not be composed of the edges of multiple realizations among the instances $\mathcal{G}_1, \dots, \mathcal{G}_\alpha$. We call this structure a *multi-walk* and formally define it in Definition 5.2. Similar to how augmenting paths are used, we will use this structure to flip the membership of some edges in their corresponding matchings with the goal of increasing the expected size of the matchings. However, note that if we naively choose the multi-walks with the sole purpose of increasing the average size of the matchings, we might violate the second property of lemma, as it might lead to some vertices being matched with an undesirably large probability. Further, these multi-walks should not be "too long" or otherwise we cannot achieve a good independence.

To overcome the first issue, after probability of a vertex v being matched in our algorithm reaches a threshold, we mark it as *saturated*. When a vertex is saturated, our algorithm ensures that while augmenting the matchings (using multi-walks), it does not increase the number of matchings in which this vertex is matched. Having these constrains narrows down our choices of augmenting structures (multi-walks) significantly. However, we give a constructive proof, and show that this narrow set still in expectation includes a subset that can be used to increase the average size of our matchings sufficiently.

5.2. *The Formal Algorithm $\mathcal{B}(\mathcal{G})$*

We start with some definitions.

Definition 5.2 (multi-walks)**.** *We call a tuple* $W = ((s_1, e_1), \dots, (s_l, e_l))$ *a multi-walk of length* l *of profile* $P = ((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_k, M_k))$*, iff it satisfies:*

- *For any* $i \in [l]$*, we have* $s_i \in [k]$*, and* e_i *is an edge in subgraph* $\mathcal{G}_{s_i}$*.*
- $(e_1, \dots, e_k)$ *is a walk in graph* G*.*
- W *contains distinct elements, e.g., for any* i *and* j*, we have* $(s_i, e_i) \neq (s_j, e_j)$*.*

Given a profile $P = ((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_j, M_k))$ *and a multi-walk* $W = ((s_1, e_1), \dots, (s_l, e_l))$*, we say* $P \oplus$

Algorithm 4. The recursive algorithm $\mathcal{B}_r(\mathcal{G})$.

```
1  If $r = 0$, return an empty matching.
2  Set $\alpha \leftarrow \varepsilon^{-12} + 1$, $l \leftarrow 3\varepsilon^{-3}$.
3  For any $i \in [\alpha]$, construct $\mathcal{G}_i$ as follows. We set $\mathcal{G}_1 := \mathcal{G}$, and for any $1 < r$ subgraph $\mathcal{G}_i$ includes any
     edge $e \in G$ independently with probability $p$.
4  Define profile $P := ((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_\alpha, M_\alpha))$ where $M_i := \mathcal{B}_{r-1}(\mathcal{G}_i)$.
5  Call a vertex $v$ saturated iff $\Pr_{\mathcal{G}' \sim G, \mathcal{B}}[v \in Z_{r-1}] \leq \Pr_{\mathcal{G} \sim G, \mathcal{A}}[v \in \mathcal{A}(\mathcal{G}')] + \varepsilon^3 - 1/\alpha$, and unsaturated
     otherwise.
6  Let $\mathcal{W}_a$ be the set of alternating multi-walks of $P$ that are applicable with respect to the set of saturated
     vertices.
7  Construct the weighted hyper-graph $H = (V, E_H)$ as follows. For any multi-walk $W$ in set $\mathcal{W}_a$ with
     length at most $l$, $H$ contains a hyper-edge between vertices in $W$ with weight $g(W, P)$.
8  $M_H \leftarrow$ ApproxMatching($H$). // See Proposition 5.5 for the ApproxMatching() algorithm.
9  Iterate over all hyper-edges in $M_H$, apply their corresponding multi-walks on $P$, and let
     $P' := ((\mathcal{G}_1, M'_1), \dots, (\mathcal{G}_\alpha, M'_\alpha))$ be the final profile.
10 Return matching $M'_1$.
```

$W = ((\mathcal{G}_1, M'_1), \dots, (\mathcal{G}_j, M'_k))$ *is the result of applying* W *on* P *iff for any* $i \in [k]$, M'_i *is constructed as follows:*

$$\begin{aligned} M'_i = & M_i \cup \{e_j \mid i = s_j \text{ and } e_j \notin M_i\} \\ & \setminus \{e_j \mid i = s_j \text{ and } e_j \in M_i\}. \end{aligned}$$

Definition 5.3 (alternating multi-walks)**.** *A multi-walk* $W = ((s_1, e_1), \dots, (s_k, e_k))$ *of profile* $P = ((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_\alpha, M_\alpha))$ *is an alternating multi-walk iff it satisfies the two following conditions. First, for any* $i \in [k-1]$ *we have* $\mathbf{1}(e_i \in M_{s_i}) + \mathbf{1}(e_{i+1} \in M_{s_{i+1}}) = 1$. *Second,* $P \oplus W$ *is a profile. We further define* $g(W, P)$*, the gain of applying alternating multi-walk* W *on* P*, as*

$$g(W, P) = \sum_{(i, e') \in W} (\mathbf{1}(e' \notin M_i) - \mathbf{1}(e' \in M_i)) w(e').$$

Given $W = ((s_1, e_1), \dots, (s_l, e_l))$, an alternating multi-walk of $P = ((\mathcal{G}_1, M_1), \dots, (\mathcal{G}_k, M_k))$, and any vertex $v \in V$ we define $d_{W,v}$ and $\bar{d}_{W,v}$ as follows:

$$\begin{aligned} d_{W,v} &= |\{i : v \in e_i, \text{ and } e_i \in M_{s_i}\}|, \\ \bar{d}_{W,v} &= |\{i : v \in e_i, \text{ and } e_i \notin M_{s_i}\}|. \end{aligned} \tag{11}$$

Definition 5.4 (applicable multi-walks)**.** *Given a multi-walk* $W = ((s_1, e_1), \dots, (s_l, e_l))$ *of profile* P*, and a subset of vertices* V_s*, we say* W *is* applicable *with respect to a set of vertices* V_s *iff it is alternating and for any* $v \in V_s$ *it satisfies* $d_{W,v} \geq \bar{d}_{W,v}$.

To prove Lemma 4.4, we design an algorithm $\mathcal{B}$ that given a random realization of G outputs a matching Z and show that it satisfies the desired properties of the lemma. Algorithm 4 is the formal algorithm $\mathcal{B}_r(\mathcal{G})$ that given an integer number r and a realization $\mathcal{G}$ of G outputs a matching of $\mathcal{G}$. We set $\mathcal{B}(\mathcal{G}) = \mathcal{B}_t(\mathcal{G})$ for $t = c_t \varepsilon^{-20}$ where c_t is a sufficiently large constant.

In Algorithm 4 we have employed a subroutine for finding an approximate matching in a hypergraph. This is the only step of the algorithm that is non-trivial to implement in the LOCAL model efficiently; for that we use the following algorithm of Harris as a black-box:

Proposition 5.5 ([12, Theorem 1.2])**.** *Given a hypergraph of rank* r *and a constant* $\delta \in (0, 1/2)$*, there is an* $\tilde{O}(\log \Delta + r)$*-round algorithm in the* LOCAL *model to get an* $O(r)$*-approximation to maximum weight matching with probability at least* $1 - 1/\delta$*. Here the* $\tilde{O}$ *notation hides* $\operatorname{poly}\log\log\Delta$ *and* $\operatorname{poly}\log r$ *factors.*

Due to space limits, we do not provide the full proof that the algorithm as stated satisfies all the properties of Lemma 4.4 and leave it for the full version of the paper [4]. In what follows, we only prove that the first property of Lemma 4.4 is satisfied.

Let us start with a simple observation.

Observation 5.6. *For any* r*, matchings* $M'_1, \dots, M'_\alpha$ *in Algorithm* $\mathcal{B}_r(\mathcal{G})$ *are random variables that are drawn from the same distribution.*

Proof: This is due to the fact that matchings $M_1, \dots, M_\alpha$ are independent random variables from the same distribution, and that to obtain $M'_1, \dots, M'_\alpha$, based on these matchings, the algorithm is essentially oblivious to the index of these instances and thus are all "treated" in the same way. ■

We now prove the following claim that the first property of Lemma 4.4 is satisfied by our algorithm.

Claim 5.7. *For any vertex* $v \in V$*, we have* $\Pr_{\mathcal{G} \sim G, \mathcal{B}}[v \in Z] \leq \Pr_{\mathcal{G} \sim G, \mathcal{A}}[v \in \mathcal{A}(\mathcal{G})] + \varepsilon^3$.

Proof: We will prove a stronger claim which is for any $v \in V$, and any $r \leq t$, we have $q_{r,v} \leq q^{\mathcal{A}}_v + \varepsilon^3$, where

$$xq_{r,v} := \Pr_{\mathcal{G}\sim G,\mathcal{B}}[v \in \mathcal{B}_r(\mathcal{G})], \text{ and } q_v^{\mathcal{A}} := \Pr_{\mathcal{G}\sim G,\mathcal{B}}[v \in \mathcal{A}(\mathcal{G})].$$

We use proof by induction. The claim obviously holds for $r = 0$. For any $r > 0$, we assume that $q_{r-1,v} \leq q_v^{\mathcal{A}} + \varepsilon^3$ holds and obtain $q_{r,v} \leq q_v^{\mathcal{A}}$. Draw a random realization of G and denote it by $\mathcal{G}$ (i.e. $\mathcal{G} \sim G$). Consider matchings $M_i, \ldots, M_\alpha$, and $M'_i, \ldots, M'_\alpha$ from algorithm $\mathcal{B}_r(\mathcal{G})$, and let us define

$$\rho_{r,v} := |\{i : v \in M_i\}|/\alpha, \text{ and } \rho'_{r,v} = |\{i : v \in M'_i\}|/\alpha.$$

We claim that $q_{r-1,v} = \rho_{r,v}$ and $q_{r,v} = \rho'_{r,v}$ hold. The former is due to the fact that any $i \in [\alpha]$, M_i is the result of running algorithm $\mathcal{B}_{r-1}$ on a random realization of G which by definition is equal to $q_{r-1,v}$. For the latter, note that we have $M'_i = \mathcal{B}_r(\mathcal{G})$ and by Observation 5.6, we know that matchings $M'_i, \ldots, M'_\alpha$ are drawn from the same distribution. As a result, we get

$$|\{i : v \in M'_i\}| = \alpha \Pr[v \in \mathcal{B}_r(\mathcal{G})],$$

which implies $q_{r,v} = \rho'_{r,v}$.

We prove our induction step for the cases of $q_{r-1,v} \leq q_v^{\mathcal{A}} + \varepsilon^3 - 1/\alpha$, and $q_{r-1,v} > q_v^{\mathcal{A}} + \varepsilon^3 - 1/\alpha$ separately. We first show that if $q_{r-1,v} \leq q_v^{\mathcal{A}} + \varepsilon^3 - 1/\alpha$, (i.e., v is not saturated), then $\rho_{r,v} \geq \rho'_{r,v} - 1/\alpha$ holds, which can be interpreted as

$$q_v^{\mathcal{A}} + \varepsilon^3 - 1/\alpha \geq q_{r-1,v} \geq q_{r,v} - 1/\alpha,$$

and as a result $q_v^{\mathcal{A}} + \varepsilon^3 \geq q_{r,v}$. Let W_H denote the set of multi-walks corresponding to edges in M_H constructed in $\mathcal{B}_r(\mathcal{G})$. Since M_H is a matching, for any vertex v, there exists at most one multi-walk $W \in W_H$ that contains vertex v. In addition, since W is alternating, we have $|d_{W,v} - \bar{d}_{W,v}| \leq 1$, where $d_{W,v}$ and $\bar{d}_{W,v}$ are defined as

$$d_{W,v} = |\{i : v \in e_i, \text{ and } e_i \in M_{s_i}\}| \text{ , and}$$
$$\bar{d}_{W,v} = |\{i : v \in e_i, \text{ and } e_i \notin M_{s_i}\}|.$$

Since after applying a multi-walk W on a profile, membership of the edges in W flips in their corresponding matchings, we get $|\{i : v \in M_i\}| \geq |\{i : v \in M'_i\}| - 1$ which means $\rho_{r,v} \geq \rho'_{r,v} - 1/\alpha$. We now consider the case of $q_{r-1,v} \geq q_v^{\mathcal{A}} + \varepsilon^3 - 1/\alpha$, (i.e., v is saturated) and show that in this case, $\rho_{r,v} \geq \rho'_{r,v}$ holds. Due to W being applicable with respect to the set of saturated vertices, by Definition 5.4, it satisfies $d_{W,v} \geq \bar{d}_{W,v}$. This directly yields $\rho_{r,v} \geq \rho'_{r,v}$, and as a result $q_{r-1,v} \geq q_{r,v}$. Based on the induction hypothesis, we have $q_{r-1,v} \leq q_v^{\mathcal{A}} + \varepsilon^3$ which implies $q_{r,v} \leq q_v^{\mathcal{A}} + \varepsilon^3$ and completes the proof. ■

REFERENCES

[1] Sepehr Assadi and Aaron Bernstein. Towards a Unified Theory of Sparsification for Matching Problems. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 11:1–11:20, 2019.

[2] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The Stochastic Matching Problem with (Very) Few Queries. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 43–60, 2016.

[3] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The Stochastic Matching Problem: Beating Half with a Non-Adaptive Algorithm. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 99–116, 2017.

[4] Soheil Behnezhad and Mahsa Derakhshan. Stochastic Weighted Matching: $(1 - \varepsilon)$ Approximation. *CoRR*, abs/2004.08703, 2020.

[5] Soheil Behnezhad, Mahsa Derakhshan, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic Matching on Uniformly Sparse Graphs. In *Algorithmic Game Theory - 12th International Symposium, SAGT 2019, Athens, Greece, September 30 - October 3, 2019, Proceedings*, pages 357–373, 2019.

[6] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic Matching with Few Queries: $(1 - \varepsilon)$ Approximation. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1111–1124, 2020.

[7] Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic Matching with Few Queries: New Algorithms and Tools. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2855–2874, 2019.

[8] Soheil Behnezhad and Nima Reyhani. Almost Optimal Stochastic Weighted Matching with Few Queries. In *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, pages 235–249, 2018.

[9] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is Almost Bliss: Near-Optimal Stochastic Matching With Few Queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland,*

OR, USA, June 15-19, 2015, pages 325–342, 2015.

[10] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance Is Almost Bliss: Near-Optimal Stochastic Matching with Few Queries. *Operations Research*, 68(1):16–34, 2020.

[11] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

[12] David G. Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 700–724. IEEE Computer Society, 2019.

[13] Takanori Maehara and Yutaro Yamaguchi. Stochastic Monotone Submodular Maximization with Queries. *CoRR*, abs/1907.04083, 2019.

[14] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer Science & Business Media, 2003.

[15] Yutaro Yamaguchi and Takanori Maehara. Stochastic Packing Integer Programs with Few Queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 293–310, 2018.