A Simple $(1-\varepsilon)$-Approximation Semi-Streaming Algorithm for Maximum (Weighted) Matching

Received Apr 23, 2024
Revised Nov 29, 2024
Accepted Feb 07, 2025
Published Aug 01, 2025

Key words and phrases
Graph streaming algorithms, Semi-streaming algorithms, Multiplicative weight update method

Sepehr Assadi [a]

a School of Computer Science, University of Waterloo.

ABSTRACT. We present a simple semi-streaming algorithm for $(1-\varepsilon)$-approximation of bipartite matching in $O(\log(n)/\varepsilon)$ passes. This matches the performance of state-of-the-art "ε-efficient" algorithms—the ones with much better dependence on ε albeit with some mild dependence on n—while being considerably simpler.

The algorithm relies on a direct application of the multiplicative weight update method with a self-contained primal-dual analysis that can be of independent interest. To showcase this, we use the same ideas, alongside standard tools from matching theory, to present an equally simple semi-streaming algorithm for $(1-\varepsilon)$-approximation of weighted matchings in general (not necessarily bipartite) graphs, again in $O(\log(n)/\varepsilon)$ passes.

1. Introduction

We consider the maximum matching problem in the semi-streaming model of [27]: given any n-vertex graph $G=(V,E)$ whose edges are presented in a stream, the goal is to make a minimal number of passes over this stream and use a limited space of $\widetilde{O}(n) := O(n \cdot \text{polylog}\,(n))$ bits to output a $(1-\varepsilon)$-approximate maximum matching of G for some given $\varepsilon > 0$.

The maximum matching problem is arguably the most studied problem in the graph streaming literature at this point (see, e.g. [12] for a quick summary). Most relevant to our work, the first $(1-\varepsilon)$-approximation algorithm for maximum cardinality matching was designed by [47] which requires $(1/\varepsilon)^{O(1/\varepsilon)}$ passes. This algorithm has since been improved numerous times [26, 3, 2, 25, 35, 1, 15, 53, 29, 10, 28, 9, 32] culminating in the state-of-the-art, consisting of two incomparable families of algorithms:

Supported in part by a Sloan Research Fellowship, an NSERC Discovery Grant, and a Faculty of Math Research Chair grant from University of Waterloo. This article was invited from *Symposium on Simplicity in Algorithms, SOSA 2024* [6].

- **"Constant pass" algorithms.** We have the algorithm of [10] (and its precursor [3]) with $O(1/\varepsilon^2)$ passes for bipartite graphs and that of [28] with $\text{poly}(1/\varepsilon)$ passes for general graphs. Similarly, for weighted graphs, we have the algorithm of [3] with $O(\log(1/\varepsilon)/\varepsilon^2)$ passes for bipartite graphs and [32] with $\text{poly}(1/\varepsilon)$ passes for general graphs[1].
- **"ε-efficient" algorithms.**[2] We have the algorithm of [1] (and its precursor [2]) for weighted general graphs with $O(\log(n)/\varepsilon)$ passes, and a simpler and space-optimal algorithm of [9] with $O(\log n \cdot \log(1/\varepsilon)/\varepsilon)$ passes that is specific to bipartite cardinality matching.

See Table 1 for a detailed summary. Nevertheless, despite significant progress in bringing down the pass-complexity of more general cases, for the most basic version of the problem, namely, *maximum (cardinality) bipartite matching (MBM)*, the best bounds have been stuck at $O(1/\varepsilon^2)$ and $O(\log(n)/\varepsilon)$ passes for over a decade now (since [3] and [2], respectively). On the other hand, even the recent advances on multi-pass graph streaming lower bounds in [11, 20, 7, 12, 40] only rule out $o(\log(1/\varepsilon))$-pass algorithms for MBM [12] (under a certain combinatorial hypothesis), leaving an exponential gap open for progress on both ends.

In our opinion, one contributing factor to the lack of algorithmic progress is the fact that the $O(\log(n)/\varepsilon)$-pass algorithms of [2, 1] are quite complicated (even for MBM). While some simplifications have been made in [9], even this new algorithm is far from being simple. This is in contrast with constant-pass algorithms that, at least for MBM, admit quite simple algorithms in [10] (even already from [3]). The goal of this paper is to remedy this state of affairs.

Our Contributions

We present a novel way of approximating matchings that is easily implementable via semi-streaming algorithms (among others). The high level idea—with some ambiguity left on purpose—is:

1. Sample $\widetilde{O}(n/\varepsilon)$ edges uniformly and compute a maximum matching M of the sample.
2. If M is large enough, return M; otherwise, *(a)* find edges that "could have potentially led to a larger matching", *(b)* increase their "importance", and repeat the sampling.

This general idea of "sample-and-solve" is a staple in the graph streaming literature dating back, at the very least, to the *filtering* [44] and *sample-and-prune* [43] techniques (both used for implementing greedy algorithms). It relies on a fundamental power of semi-streaming algorithms: once we *sparsify* the input to fit into the memory, we can process it however we

1 It is worth mentioning that the dependence on ε in [28, 32] is quite high – it appears to be $O((1/\varepsilon)^{19})$ passes in [28] (see Lemma 5.6 of arXiv version 5) and can only be higher in [32].

2 We use this term to refer to algorithms that have much better dependence on parameter ε compared to the above line of work at the cost of having some mild dependence on n, which is almost always an $O(\log n)$ factor.

Citation	Space	Passes	Bip./Gen.	Size/Weight	Det./Rand.
"constant-pass" algorithms					
[47]	$\widetilde{O}_\varepsilon(n)$	$(1/\varepsilon)^{O(1/\varepsilon)}$	Gen.	Size	Rand.
[26]	$O(n \log n)$	$O(1/\varepsilon^8)$	Bip.	Size	Det.
[3]	$\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$	$O((1/\varepsilon^2) \cdot \log(1/\varepsilon))$	Bip.	Weight	Det.
[25]	$O(n \log n)$	$O(1/\varepsilon^5)$	Bip.	Size	Det.
[35]	$O(n \log n)$	$O(1/\varepsilon^2)$	Bip. (vertex arrival)	Size	Det.
[53]	$\widetilde{O}_\varepsilon(n)$	$(1/\varepsilon)^{O(1/\varepsilon)}$	Gen.	Size	Det.
[29]	$\widetilde{O}_\varepsilon(n)$	$(1/\varepsilon)^{O(1/\varepsilon^2)}$	Gen.	Weight	Det.
[10]	$O(n \log n)$	$O(1/\varepsilon^2)$	Bip.	Size	Det.
[28]	$\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$	$O((1/\varepsilon)^{19})$	Gen.	Size	Det.
[32]	$\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$	$(1/\varepsilon)^{O(1)}$	Gen.	Weight	Det.
"ε-efficient" algorithms					
[3]	$\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$	$O(\log n \cdot \mathrm{poly}(1/\varepsilon))$	Gen.	Weight	Det.
[2]	$\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$	$O(\log(n)/\varepsilon)$	Gen.	Size (value not edges)	Rand.
[1]	$\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$	$O(\log(n)/\varepsilon)$	Gen.	Weight	Rand.
[9]	$O(n \log n)$	$O(\log(n)/\varepsilon \cdot \log(1/\varepsilon))$	Bip.	Size	Det.
beyond semi-streaming algorithms					
[1]	$\widetilde{O}(n^{1+1/p} \cdot \mathrm{poly}(1/\varepsilon))$	$O(p/\varepsilon)$	Gen.	Weight	Rand.
[15]	$O(n^{1.5} \log(n)/\varepsilon)$	$O(1/\varepsilon)$	Bip.	Size	Rand.
[8]	$o_\varepsilon(n^2)$	1	Gen.	Size	Rand.
multi-pass lower bounds					
[20]	$n^{1+\Omega(1)}$	$\Omega(\log(1/\varepsilon)/\log\log n)$	Bip.	Size	Rand.
[7]	$n^{1+\Omega(1)}$	> 2	Bip.	Size	Rand.
[12]	$n^{1+\Omega(1)}$	$\Omega(\log(1/\varepsilon))$	Bip.	Size	Rand.

Table 1. Summary of the prior work on $(1-\varepsilon)$-approximate streaming matchings: *Bip./Gen.* refers to bipartite versus general graphs, *Size/Weight* refers to cardinality versus weighted matchings, and *Det./Rand.* refers to deterministic versus randomized algorithms. The space is stated in number of bits and thus $O(n \log n)$ is space-optimal.
Due to the elegant "weighted-to-unweighted" reduction of [17], for *bipartite* graphs, *all* results for unweighted matchings can be generalized to weighted matchings by increasing the space with a factor of $(1/\varepsilon)^{O(1/\varepsilon)}$, while keeping the number of passes *exactly* the same (not only asymptotically).
All lower bounds stated in the table hold under a combinatorial assumption regarding existence of moderately dense *Ruzsa-Szemeredi* graphs (see [7, 12] for more details; see also [40] for a two-pass unconditional lower bound.).
See also [30, 35, 36] for much better approximation lower bounds for single-pass semi-streaming algorithms, which currently rule out $1/(1+\ln 2) \approx 0.59$ approximation in [36]. Finally, see also [18, 41, 42] for multi-pass lower bounds that hold for different restricted families of algorithms.

want. Specifically in this context, once the algorithm only has $\widetilde{O}(n/\varepsilon)$ edges to work with in the sample, it can find its maximum matching or perform any "heavy" computation easily.

The approach proposed above, based on adjusting importance of edges, is clearly reminiscent of the *Multiplicative Weight Update (MWU)* method (see [5]) and its application in the Plotkin-Shmoys-Tardos framework for approximating packing/covering LPs [51]. There is *just* one issue here: we would like this algorithm to converge in $\approx 1/\varepsilon$ passes, while these MWU-based approaches only guarantee $\approx 1/\varepsilon^2$ iterations for convergence to a $(1-\varepsilon)$-approximate solution (see, e.g., [38]). Addressing this issue is the key difference in our work compared to prior work.

Prior approaches. The algorithms in [2, 1] also start with the same overall approach and address the above-mentioned issue through several steps: (i) using a non-standard LP relaxation of the problem, (ii) relying on the dual variables of this LP to guide step *(a)* of the approach, (iii) adding a penalty-term to the LP to maintain an $O(\log(n)/\varepsilon^2)$-iterations convergence guarantee as in the Plotkin-Shmoys-Tardos framework (to reduce the *width* of the resulting problem; see [51, 5]), (iv) "folding" $O(1/\varepsilon)$ iterations of this framework in $O(1)$ passes, and (v) using a notion of a "deferred (cut) sparsification" (instead of sampling) that allows for implementing this last step. We refer the reader to [1, Section 1 and 2 of arXiv version 3] for more details on this algorithm; here, we only note that the end result is a highly sophisticated algorithm that barely resembles the above strategy but can now run in $O(\log(n)/\varepsilon)$ passes.

The recent algorithm of [9] deviates from the above approach. Instead, it relies on more sophisticated optimizations tools in [52, 34, 21] based on the classical work in [49, 50] that give $\approx 1/\varepsilon$-iteration solvers directly. This approach, to a certain degree, does not take advantage of the aforementioned power of semi-streaming algorithms—meaning arbitrary computation power on sparse-enough inputs—and is highly tailored to bipartite cardinality matching[3].

Our approach. Unlike prior work, we are going to revert to the original approaches of [44, 43] and implement the above algorithmic approach *quite literally*, without relying on Plotkin-Shmoys-Tardos or similar generic frameworks.

Concretely, our algorithm for MBM is this: in step (a), find a *minimum (bipartite) vertex cover* of the *sampled* graph (relying on Konig's theorem; see Fact 2.1); we then consider any edge of the *original* graph not covered by this vertex cover as an edge that "could have potentially led to a larger matching". For step *(b)*, we double the importance of these edges (making them twice as likely to be sampled next)[4]. A simple analysis, relying on the duality of matchings and vertex covers, bounds the number of iterations by $O(\log(n)/\varepsilon)$, leading to the following result.

3 The work of [9] also have an algorithm for weighted bipartite matching but the pass-complexity depends linearly on the maximum weight of an edge (which can be polynomial in n) and hence is typically not efficient.

4 As a side note, this is a much more aggressive update rule compared to a typical MWU application, say in Plotkin-Shmoys-Tardos framework, which would have updated the weights by only a $(1+\varepsilon)$ factor; see also Section 5.5.

RESULT 1.1. There is a semi-streaming algorithm that given any n-vertex bipartite graph G and a parameter $\varepsilon \in (0,1)$, uses $O(n\log(n)/\varepsilon)$ bits of space and $O(\log(n)/\varepsilon)$ passes and with exponentially high probability[5] outputs a $(1-\varepsilon)$-approximate maximum matching of G.

We believe Result 1.1 is our main contribution as it already contains our key new ideas. Still, this result can be significantly generalized to cover maximum weight matchings in general graphs, with minimal additional effort and by relying on standard tools from matching theory.

RESULT 1.2. There is a semi-streaming algorithm that given any n-vertex general graph G with integer edge weights $w : E \to \mathbb{N}$ and a parameter $\varepsilon \in (0,1)$, uses $O(n\log^2(n)/\varepsilon)$ bits of space and $O(\log(n)/\varepsilon)$ passes and with exponentially high probability outputs a $(1-\varepsilon)$-approximate maximum weight matching of G.

Result 1.2 is now providing a considerably simpler algorithm compared to the main results of [1] (also with a better space-dependence by $\mathrm{poly}(\log n, 1/\varepsilon)$ factors). We hope this can pave the way for both future theoretical improvements and more practical algorithms for this fundamental problem[6].

Finally, we note that in the interest of keeping the main ideas in this paper as transparent as possible, we have opted to focus only on the most important aspects of our algorithms in our main arguments. Then, in Section 5, we point out several standard and not-so-standard extensions of our algorithms such as improved runtime, $O(1/\varepsilon)$-pass algorithms in $n^{1+\Omega(1)}$ space, derandomization, and others.

2. Preliminaries

Notation. For any graph $G = (V,E)$, we use n to denote the number of vertices and m as the number of edges. We further use $\mu(G)$ to denote the maximum matching size in G and $\mu(G,w)$ to denote the maximum matching weight in G under edge weights $w : E \to \mathbb{N}$.

In the following, we review some basic facts from matching theory in bipartite and general (weighted) graphs, separately. We note that our Result 1.1 and the arguments in Section 3 only rely on the basics for bipartite graphs, and thus a reader solely interested in that part of the paper, can safely skip Section 2.2 below.

2.1 Basics of Matching Theory in Bipartite Graphs

Let $G = (L,R,E)$ be a bipartite graph. Recall the following definitions:

5 Here, and throughout, 'with exponentially high probability' means with probability at least $1 - \exp(-\Theta(n))$.

6 It is worth mentioning that the generic approaches of [44, 43] that are most similar to our algorithms have indeed led to highly practical algorithms; see the empirical evaluations in the aforementioned papers.

— A matching M is a set of vertex-disjoint edges in E and a fractional matching $x \in [0,1]^E$ is an assignment to the edges so that for every vertex $v \in L \cup R$, we have $\sum_{e \ni v} x_e \leqslant 1$. We denote the size of a fractional matching x by $|x| := \sum_{e \in E} x_e$.

— Similarly, a vertex cover U is a set of vertices incident on every edge and a fractional vertex cover $y \in [0,1]^V$ is an assignment to the vertices so that for every edge $e = (u,v) \in E$, $y_u + y_v \geqslant 1$. We denote the size of a fractional vertex cover y by $|y| := \sum_{v \in L \cup R} y_v$.

(We only use fractional matchings and vertex covers in the analysis).

Konig's theorem [39] establishes duality of maximum (fractional) matchings and minimum (fractional) vertex covers in bipartite graphs.

FACT 2.1 (Konig's theorem). *In bipartite graphs, the sizes of maximum matchings, fractional matchings, minimum vertex covers, and fractional vertex covers are all the same.*

See the excellent book of Lovász and Plummer [45] on matching theory for more details.

2.2 Basics of Matching Theory in General (Weighted) Graphs

Let $G = (V,E)$ be a (general) graph with *integer* edge weights $w : E \rightarrow \mathbb{N}$. The duality between matchings and vertex covers no longer holds in general graphs, nor the equivalence of fractional matchings and integral ones (the way defined previously). Thus, one needs a more general definition. In the following, we use $odd(V)$ to denote the collection of all sets of vertices in V with *odd* cardinality. For a set $S \subseteq V$, we use $E[S]$ to denote the edges with *both* endpoints in S.

— As before, a matching M is a set of vertex-disjoint edges in E and $w(M)$ is its weight. We define a (general) fractional matching $x \in [0,1]^E$ as an assignment to the edges satisfying:

$$\text{for all } v \in V\colon \sum_{e \ni v} x_e \leqslant 1 \quad \text{and for all } S \in odd(V)\colon \sum_{e \in E[S]} x_e \leqslant \frac{|S|-1}{2}.$$

We define the weight of a fractional matching as $\sum_{e \in E} w(e) \cdot x_e$.

— The dual to maximum fractional matchings is the following *"odd-set cover"* problem. We define a fractional odd-set cover as a pair of assignments $y \in \mathbb{R}^V$ and $z \in \mathbb{R}^{odd(V)}$ to vertices and odd-sets in G satisfying the following:

$$\text{for all } e = (u,v) \in E\colon \quad y_u + y_v + \sum_{\substack{S \in odd(V) \\ e \in E[S]}} z_S \geqslant w(e). \tag{1}$$

The value of a fractional odd-set cover is then

$$|(y,z)| := \sum_{v \in V} y_v + \sum_{S \in odd(V)} \frac{|S|-1}{2} \cdot z_S.$$

We have the following result—similar in spirit to Fact 2.1 for bipartite graphs—on the duality of maximum fractional matchings and odd-set covers.

FACT 2.2 (Edmond's matching polytope theorem [24]). *In any graph, the weights of maximum weight matchings and fractional matchings, and the sizes of minimum odd-set covers are the same.*

Finally, we shall also use the following result on the structure of optimal fractional odd-set covers, which is crucial for the probabilistic analysis of our algorithm. Recall that a family of sets $\mathcal{F}$ is *laminar* iff for all $A, B \in \mathcal{F}$, either $A \cap B = \emptyset$ or $A \subseteq B$ or $B \subseteq A$.

FACT 2.3 (Cunningham-Marsh theorem [22]). *In any graph G with integer edge-weights, there is an optimal fractional odd-set cover $y \in \mathbb{R}^V$ and $z \in \mathbb{R}^{odd(V)}$ such that (i) both y and z only take integer values, and (ii) $z_S > 0$ only for a family of sets in $odd(V)$ that form a laminar family.*

Again, see [45] for more details and proofs of these facts.

3. Maximum Cardinality Bipartite Matching

We prove Result 1.1 in this section. We start with presenting our new algorithm in a generic and model-independent way, and the show how it can be implemented in the semi-streaming model.

ALGORITHM 3.1. A sample-and-solve approximation algorithm for maximum bipartite matching.

— **Input**: A bipartite graph $G = (L, R, E)$ and parameter $\varepsilon \in (0, 1)$;
— **Output**: A $(1 - \varepsilon)$-approximate maximum matching in G.

1. Start with **importance**[7] $q_e^{(1)} = 1$ for every edge $e \in E$ and define $Q^{(1)} := \sum_{e \in E} q_e^{(1)}$.
2. For $r = 1$ to $R := \frac{4}{\varepsilon} \cdot \log m$ iterations:
   a. Sample each edge $e \in E$ independently at random with probability:
   $$p_e^{(r)} := \frac{2n}{\varepsilon} \cdot \frac{q_e^{(r)}}{Q^{(r)}}. \tag{2}$$
   b. Compute a maximum matching $M^{(r)}$ and a minimum vertex cover $U^{(r)}$ of the sample.
   c. For any edge $e \in E$ not covered by $U^{(r)}$, update:
   $$q_e^{(r+1)} = q_e^{(r)} \cdot 2. \tag{3}$$
   Then, let $Q^{(r+1)} = \sum_{e \in E} q_e^{(r+1)}$.

7 While it is more common to refer to this concept as 'weight' in the context of MWU, given that we will eventually work with weighted matchings, we use the term 'importance' to avoid ambiguity.

3. Return the largest of matchings $M^{(r)}$ for $r \in [R]$.

THEOREM 3.2. *For any bipartite graph $G = (L, R, E)$ and parameter $\varepsilon \in (0, 1)$, Algorithm 3.1 outputs a matching of size at least $(1 - \varepsilon) \cdot \mu(G)$ in G with exponentially high probability.*

The proof follows the recipe of the MWU analysis (e.g., in [5]), using $Q^{(r)}$ as a potential function. The first lemma upper bounds $Q^{(R+1)}$ at the end of the algorithm.

LEMMA 3.3. *With exponentially high probability, $Q^{(R+1)} \leqslant (1 + \varepsilon/2)^R \cdot m$.*

PROOF. Fix any iteration $r \in [R]$. Let $F^{(r)} \subseteq E$ denote the set of edges not covered by $U^{(r)}$. We claim that with exponentially high probability, we have,

$$\sum_{e \in F^{(r)}} q_e^{(r)} \leqslant \frac{\varepsilon}{2} \cdot Q^{(r)}. \tag{4}$$

In words, Eq (4) states that the importance of edges not covered by $U^{(r)}$ in the graph, relative to the importances of iteration r, is “small” (despite the fact that $U^{(r)}$ was computed on a sample and not the entire input). Before proving this claim, let us see how it concludes the proof.

By a union bound over $O(\log(n)/\varepsilon)$ iterations of the algorithm, we have that for every $r \in [R]$,

$$\begin{aligned}
Q^{(r+1)} &= \sum_{e \in F^{(r)}} q_e^{(r+1)} + \sum_{e \in E \setminus F^{(r)}} q_e^{(r+1)} && \text{(by partitioning the edges in and out of } F^{(r)}\text{)} \\
&= \sum_{e \in F^{(r)}} 2 \cdot q_e^{(r)} + \sum_{e \in E \setminus F^{(r)}} q_e^{(r)} && \text{(by the update rule of the algorithm)} \\
&= \left(\sum_{e \in F^{(r)}} q_e^{(r)} \right) + Q^{(r)} && \text{(by the definition of } Q^{(r)}\text{)} \\
&\leqslant (1 + \frac{\varepsilon}{2}) \cdot Q^{(r)},
\end{aligned}$$

where the inequality is by Eq (4). The lemma then follows from this and since $Q^{(1)} = m$.

Proof of Eq (4). Let $U \subseteq V$ be any subset of vertices in the graph and $F(U)$ be the set of edges not covered by U. Suppose

$$\sum_{e \in F(U)} q_e^{(r)} > \frac{\varepsilon}{2} \cdot Q^{(r)}.$$

We show that in this case, with an exponentially high probability, U cannot be a vertex cover of the sampled edges either. Indeed, we have,

$$\begin{aligned}
\Pr(U \text{ is a vertex cover of sampled edges}) &= \Pr(\text{no edge from } F(U) \text{ is sampled}) \\
&= \prod_{e \in F(U)} (1 - p_e^{(r)}) \\
&\qquad \text{(by the independence and the sampling probability of edges)}
\end{aligned}$$

$$\leqslant \exp(-\sum_{e\in F(U)} p_e^{(r)}) \qquad \text{(as } (1-x) \leqslant e^{-x} \text{ for all } x \in [0,1])$$
$$= \exp(-\frac{2n}{\varepsilon} \cdot \sum_{e\in F(U)} \frac{q_e^{(r)}}{Q^{(r)}}) \qquad \text{(by the choice of } p_e^{(r)} \text{ in Eq (2))}$$
$$\leqslant \exp(-n)$$

by our assumption about U and importance of edges in $F(U)$ earlier.

A union bound over the 2^n choices of U ensures that with exponentially high probability, any choice of $U^{(r)}$ that is returned as a vertex cover of sampled edges should satisfy Eq (4). ■

On the other hand, we are going to show that if none of the matchings $M^{(r)}$ is sufficiently large, then the importance of at least one edge should have dramatically increased to the point that it will contradict the bounds in Lemma 3.3. The proof of this lemma is based on a simple primal-dual analysis using Konig's theorem in Fact 2.1.

LEMMA 3.4. *Suppose in every iteration $r \in [R]$, we have $|M^{(r)}| < (1-\varepsilon)\cdot\mu(G)$. Then, there exists at least one edge $e \in E$ such that $q_e^{(R+1)} \geqslant 2^{\varepsilon\cdot R}$.*

PROOF. By Konig's theorem (Fact 2.1) and the assumption in the lemma statement, we also have that $|U^{(r)}| < (1-\varepsilon)\cdot\mu(G)$ for every $r \in [R]$. Let M^* be a maximum matching of G. We thus have that in each iteration $r \in [R]$, at least $\varepsilon\cdot\mu(G)$ edges of M^* are not covered. By an averaging argument, this implies that there exists at least one edge e in M^* that is not covered in at least $\varepsilon\cdot R$ iterations. Hence, by the update rule of the algorithm, the importance of this edge increases to at least $2^{\varepsilon\cdot R}$, concluding the proof. ■

We are now ready to conclude the proof of Theorem 3.2.

PROOF OF THEOREM 3.2. Assume the event of Lemma 3.3 happens, thus

$$Q^{(R+1)} \leqslant (1+\varepsilon/2)^R \cdot m < 2^{3\varepsilon R/4+\log m}. \qquad \text{(as } (1+x) < 2^{3x/2} \text{ for } x>0)$$

Suppose towards a contradiction that none of the matchings $M^{(r)}$ for $r \in [R]$ computed by the algorithm are of size at least $(1-\varepsilon)\cdot\mu(G)$. Then, by Lemma 3.4, there is an edge $e \in E$ such that

$$q_e^{(R+1)} \geqslant 2^{\varepsilon R}.$$

Putting these two equations together, as $q_e^{(R+1)} \leqslant Q^{(R+1)}$ (by positivity of importances), we obtain

$$\varepsilon R < 3\varepsilon R/4 + \log m,$$

which only holds for $R < 4\log m/\varepsilon$, contradicting the choice of R in the algorithm. Thus, at least one of the matchings returned by the algorithm is of size $(1-\varepsilon)\cdot\mu(G)$, concluding the proof. ■

Semi-Streaming Implementation

We now present a semi-streaming implementation of Algorithm 3.1 in the following lemma.

LEMMA 3.5. *Algorithm 3.1 can be implemented in the semi-streaming model with $O(n\log(n)/\varepsilon)$ bits of memory and $O(\log(n)/\varepsilon)$ passes.*

PROOF. We implement each iteration of the algorithm in $O(1)$ streaming passes. The main part of the implementation is to maintain the importance of the edges *implicitly*. We do this as follows:

— For every vertex $v \in V$ and iteration $r \in [R]$, we maintain a bit $b(v,r)$ denoting if v belongs to the vertex cover $U^{(r)}$ of iteration r (this needs $R = O(\log(n)/\varepsilon)$ bits per vertex in total);

— Whenever an edge $e = (u,v) \in E$ arrives in the stream in the r-th pass, we can compute the number of times e has remained uncovered by $U^{(r')}$ for $r' < r$, denoted by $c(e,r)$. This is done by checking $b(u,r')$ and $b(v,r')$ stored so far; in particular,

$$c(e,r) = |\{r' < r \mid b(u,r') = b(v,r') = 0\}|.$$

The importance $q_e^{(r)}$ of the edge e in this pass r is then $2^{c(e,r)}$.

— Given we can calculate the importance of each edge upon its arrival, we can first make a single pass and compute the normalization factor $Q^{(r)} = \sum_{e\in E} q_e^{(r)}$. Then, we make another pass and for each arriving edge $e \in E$, we compute $q_e^{(r)}$ as above and sample the edges with the probability prescribed by Eq (2).

The rest of the algorithm can be implemented directly. In particular, the total number of edges sampled in each iteration is $O(n/\varepsilon)$ with exponentially high probability by Chernoff bound. Thus, in the semi-streaming algorithm, we can store these edges and then compute $M^{(r)}$ and $U^{(r)}$ at the end of the pass. This concludes the proof of the lemma. ∎

Result 1.1 now follows immediately from Theorem 3.2 and Lemma 3.5.

4. Maximum Weight General Matching

We now switch to proving Result 1.2 which is a vast generalization of Result 1.1. Interestingly however, despite its generality, the proof is more or less a direct "pattern matching" of the previous ideas to general weighted graphs using the existing rich theory of matchings reviewed in Section 2.2. As before, we start by presenting a model-independent algorithm first followed by its semi-streaming implementation. Also, for simplicity of exposition, we are going to present our algorithm with space- and pass-complexity depending on the parameter $W := \sum_{e\in E} w(e)$, and then show how to fix this using standard ideas and conclude the proof of Result 1.2. Our algorithm is as follows.

ALGORITHM 4.1. A sample-and-solve approximation algorithm for weighted general matching.

- **Input**: A (general) graph $G = (V, E)$ with weights $w : E \to \mathbb{N}$ and parameter $\varepsilon \in (0, 1)$;
- **Output**: A $(1 - \varepsilon)$-approximate maximum weight matching in G.

```
1. Start with importance $q_e^{(1)} = 1$ for every edge $e \in E$ and define $Q^{(1)} := \sum_{e \in E} w(e) \cdot q_e^{(1)}$.
2. Let $W := \sum_{e \in E} w(e)$. For $r = 1$ to $R := \frac{4}{\varepsilon} \cdot \log W$ iterations:
   a. Sample each edge $e \in E$ independently at random with probability:
         $$p_e^{(r)} := \frac{8n \cdot \ln(nW)}{\varepsilon} \cdot \frac{q_e^{(r)} \cdot w(e)}{Q^{(r)}}. \quad (5)$$
   b. Compute a maximum weight matching $M^{(r)}$ and a minimum odd-set cover solution
      $(y^{(r)}, z^{(r)})$ of the sample (using the original weights $w(\cdot)$ on sampled edges).
   c. For any edge $e \in E$ not covered by $(y^{(r)}, z^{(r)})$[8], update:
         $$q_e^{(r+1)} = q_e^{(r)} \cdot 2. \quad (6)$$
      Then, let $Q^{(r+1)} = \sum_{e \in E} w(e) \cdot q_e^{(r+1)}$.
3. Return the maximum weight matching among $M^{(r)}$'s for $r \in [R]$.
```

THEOREM 4.2. *For any graph $G = (V, E)$ with weights $w : E \to \mathbb{N}$ and parameter $\varepsilon \in (0, 1)$, Algorithm 4.1 outputs a matching of weight at least $(1 - \varepsilon) \cdot \mu(G, w)$ with exponentially high probability.*

We follow the same exact strategy as before. The first step is to bound the total sum of importances across the iterations. The following lemma is an analogue of Lemma 3.3. The key difference is a new union bound argument at the very end for all potential odd-set covers which needs to be more careful compared to the trivial 2^n-bound for vertex covers.

LEMMA 4.3. *With an exponentially high probability, $Q^{(R+1)} \leqslant (1 + \varepsilon/2)^R \cdot W$.*

PROOF. Fix any iteration $r \in [R]$. Let $F^{(r)} \subseteq E$ denote the set of edges *not* covered by $(y^{(r)}, z^{(r)})$ as defined in Line (c) of Algorithm 4.1. We claim that with exponentially high probability, we have,

$$\sum_{e \in F^{(r)}} q_e^{(r)} \cdot w(e) \leqslant \frac{\varepsilon}{2} \cdot Q^{(r)}. \tag{7}$$

In words, Eq (7) states that the total "importance × weight" of the edges not covered by the odd-set cover solution on the entire graph, relative to the importances of iteration r, is "small". Before proving this claim, let us see how it concludes the proof.

8 By this, we mean (y, z) does *not* satisfy Eq (1) for the given edge $e \in E$.

By a union bound over all iterations of the algorithm, we have that for every $r \in [R]$,

$$\begin{aligned}
Q^{(r+1)} &= \sum_{e \in F^{(r)}} q_e^{(r+1)} \cdot w(e) + \sum_{e \in E \setminus F^{(r)}} q_e^{(r+1)} \cdot w(e) && \text{(by partitioning the edges in and out of } F^{(r)}\text{)} \\
&= \sum_{e \in F^{(r)}} 2 \cdot q_e^{(r)} \cdot w(e) + \sum_{e \in E \setminus F^{(r)}} q_e^{(r)} \cdot w(e) && \text{(by the update rule of the algorithm)} \\
&= \left(\sum_{e \in F^{(r)}} q_e^{(r)} \cdot w(e) \right) + Q^{(r)} && \text{(by the definition of } Q^{(r)}\text{)} \\
&\leqslant (1 + \frac{\varepsilon}{2}) \cdot Q^{(r)},
\end{aligned}$$

where the inequality is by Eq (7). The lemma then follows from this and the choice of $q_e^{(1)} = 1$ for all edge $e \in E$ which implies $Q^{(1)} = \sum_{e \in E} 1 \cdot w(e) = W$.

Proof of Eq (7). Let $y \in \mathbb{R}^V$ and $z \in \mathbb{R}^{odd(V)}$ be any "potential" odd-set cover of G. We define $F(y,z) \subseteq E$ as the set of edges *not* covered by this potential odd-cut cover. Suppose

$$\sum_{e \in F(y,z)} q_e^{(r)} \cdot w(e) > \frac{\varepsilon}{2} \cdot Q^{(r)};$$

we show that in this case, with an exponentially high probability, (y,z) cannot be a feasible odd-set cover of the sampled edges either. Indeed, we have,

$$\begin{aligned}
\Pr((y,z) \text{ is feasible on sampled edges}) &= \Pr(\text{no edge from } F(y,z) \text{ is sampled}) \\
&= \prod_{e \in F(y,z)} (1 - p_e^{(r)}) \\
&\qquad \text{(by the independence and the sampling probability of edges)} \\
&\leqslant \exp(-\sum_{e \in F(y,z)} p_e^{(r)}) \\
&\qquad \text{(as } (1-x) \leqslant e^{-x} \text{ for all } x \in [0,1]\text{)} \\
&= \exp(-\frac{8n \cdot \ln{(nW)}}{\varepsilon} \cdot \sum_{e \in F(y,z)} \frac{q^{(r)} \cdot w(e)}{Q^{(r)}}) \\
&\qquad \text{(by the choice of } p_e^{(r)} \text{ in Eq (5))} \\
&\leqslant \exp(-4n \cdot \ln{(nW)}),
\end{aligned}$$

by our assumption about (y,z) and importance of edges in $F(y,z)$ earlier.

The last step of the proof is to union bound over all potential odd-set covers (y,z) using the calculated probabilities above. This step needs to be more careful compared to Lemma 3.3 because (y,z) can be fractional and even for integral values, z can have an exponential support, leading to a doubly exponential number of choices for it; this is too much for the above probabilities to handle. Both of these are handled using the Cunningham-Marsh theorem (Fact 2.3):

— Firstly, we can assume without loss of generality that (y, z) only take integer values in $[0 : W]$ in the optimal solutions computed in Line (b) of Algorithm 4.1. This, for instance, implies that the total number of choices for y that we need to consider is $(W+1)^n$.

— Secondly, and more importantly, we can use the laminarity of $\mathcal{F}(z) := \{S \in odd(V) \mid z_S > 0\}$ in the optimal solution. A standard observation about laminar families (over n elements/vertices) is that they can have size at most $2n - 1$[9]. We can use this to provide a (crude) upper bound the number of choices for z:

  — Define $T(n)$ as the number of laminar families on $[n]$[10]. We claim $T(n) \leqslant (4n) \cdot T(n-1)$. We can pick a laminar family on $n-1$ elements in $T(n-1)$ ways, and then decide to put the last element in *(a)* one of its (at most) $(2n-3)$ sets and "propagate" it to each of its supersets also, *(b)* in a singleton set and decide whether or not to propagate it, or *(c)* not place it anywhere. This leads to $< 4n$ options times $T(n-1)$, establishing the claim.

  — Given that $T(1) = 2$ (singleton or empty-set), we get $T(n) \leqslant (4n)^n$. To pick z, we can first pick $\mathcal{F}(z)$ using at most $(4n)^n$ ways, and then assign values from $[W]$ to each of its at most $(2n-1)$ sets. Hence, there are at most $W^{2n-1} \cdot (4n)^n$ choices for z.

All in all, we obtain that the total number of possible optimal solutions (y, z) that we need to take a union bound over can be (very) crudely upper bounded by the following (assuming $n > 4$ without loss of generality):

$$(W+1)^n \cdot W^{2n-1} \cdot (4n)^n < (2n)^{2n} \cdot W^{3n} < (nW)^{3n} < \exp(3n \cdot \ln(nW)).$$

Finally, we can apply a union bound over these many choices and get that with an exponentially high probability no such solution (y, z) can be a feasible solution on the sample. This concludes the proof. ■

We then prove an analogue of Lemma 3.4, using the duality of odd-set covers and (general) matchings, in place of vertex cover/matching duality in bipartite graphs.

LEMMA 4.4. *Suppose in every iteration $r \in [R]$, we have $w(M^{(r)}) < (1-\varepsilon) \cdot \mu(G, w)$. Then, there exists at least one edge $e \in E$ such that $q_e^{(R+1)} \cdot w(e) \geqslant 2^{\varepsilon \cdot R}$.*

PROOF. By Edmond's matching polytope theorem (Fact 2.2) and the assumption in the lemma statement, we also have that $|(y^{(r)}, z^{(r)})| < (1-\varepsilon) \cdot \mu(G)$ for every $r \in [R]$. Let M^* be a maximum matching of G. We thus have that in each iteration $r \in [R]$, at least $\varepsilon \cdot \mu(G)$ edges of M^* are not covered (as defined in Line (c) of Algorithm 4.1); otherwise, another application of Fact 2.2 to

9 Without loss of generality, assume $\mathcal{F}$ is a *maximal* laminar family on $[n]$. Proof by induction (base case is trivial): maximality ensures that there are two non-empty sets $A, B \in \mathcal{F}$ with $A \cup B = [n]$ and $A \cap B = \emptyset$. By induction, there are at most $2|A| - 1$ subsets of A in $\mathcal{F}$, at most $2|B| - 1$ subsets of B in $\mathcal{F}$, and the set $A \cup B$ can also be in $\mathcal{F}$, implying that $|\mathcal{F}| \leqslant (2|A| - 1) + (2|B| - 1) + 1 = 2n - 1$.

10 There are more accurate ways of bounding $T(n)$ (see, e.g. [19]), but given that these more accurate bounds do not help with our subsequent calculations, we just establish a crude upper bound with a self-contained proof here.

the covered edges of M^* implies that $|(y^{(r)}, z^{(r)})| \geqslant (1-\varepsilon) \cdot \mu(G)$ also, a contradiction. By an averaging argument, this implies that there exists at least one edge e in M^* that is not covered in at least $\varepsilon \cdot R$ iterations. Hence, by the update rule of the algorithm, the importance of this edge increases to at least $2^{\varepsilon \cdot R}$, concluding the proof. ■

We are now ready to conclude the proof of Theorem 4.2 exactly as in that of Theorem 3.2, using the above established lemmas instead.

PROOF OF THEOREM 4.2. Assume the event of Lemma 4.3 happens, thus

$$Q^{(R+1)} \leqslant (1+\varepsilon/2)^R \cdot W < 2^{3\varepsilon R/4 + \log W}. \qquad \text{(as } (1+x) < 2^{3x/2} \text{ for } x > 0)$$

Suppose towards a contradiction that none of the matchings $M^{(r)}$ for $r \in [R]$ computed by the algorithm are of weight at least $(1-\varepsilon) \cdot \mu(G, w)$. By Lemma 4.4, there is an edge $e \in E$ such that

$$q_e^{(R+1)} \geqslant 2^{\varepsilon R}.$$

Putting these two equations together, as $q_e^{(R+1)} \leqslant Q^{(R+1)}$ (by positivity of importances), we obtain

$$\varepsilon R < 3\varepsilon R/4 + \log W,$$

which only holds for $R < 4 \log W/\varepsilon$, contradicting the choice of R in the algorithm. Thus, at least one of the found matchings $M^{(r)}$'s is of weight $(1-\varepsilon) \cdot \mu(G, w)$, concluding the proof. ■

Semi-Streaming Implementation

Finally, we are going to show a semi-streaming implementation of Algorithm 4.1 in the following lemma. The proof is similar to that of Lemma 3.5 although again with crucial changes to account for the difference of vertex covers in Algorithm 3.1 with odd-set covers in Algorithm 4.1 (we need some minor modifications after this also in order to be able to prove Result 1.2 which will be done next).

LEMMA 4.5. *Algorithm 4.1 is implementable in the semi-streaming model with* $O(n \log(nW) \cdot \log(n)/\varepsilon)$ *bits of memory and* $O(\log(W)/\varepsilon)$ *passes.*

PROOF. We implement each iteration of the algorithm via $O(1)$ passes over the stream. The main part of the semi-streaming implementation is to maintain the importance of the edges implicitly. To do this, we do as follows:

— For every iteration/pass $r \in [R]$:
  - — Store the vector $y^{(r)}$ explicitly using $O(n \log W)$ bits using the integrality of the optimal solution (by Fact 2.3).
  - — Store the vector $z^{(r)}$ explicitly using $O(n \log n + n \log W)$ bits using the laminarity of support of z in the optimal solution (by Fact 2.3): this can be done because earlier *(a)* we bounded the number of laminar families by $(4n)^n$ and hence they require $O(n \log n)$

bits of representation, and *(b)* we bounded the number of sets in any laminar family by $2n - 1$, so we only need to store $O(n \log W)$ bits to store their values.

— Whenever an edge $e = (u, v) \in E$ arrives in the stream in the r-th pass, we can compute the number of times e has remained uncovered by $U^{(r')}$ for $r' < r$, denoted by $c(e, r)$, by checking $(y^{(r')}, z^{(r')})$ stored so far and counting the number of times they violate Eq (1) for this particular edge e. The importance $q_e^{(r)}$ of the edge e is then $2^{c(e,r)}$.

— Once we calculate the importance of an edge upon its arrival, we can sample the edge with probability prescribed by Eq (5), by making an additional pass to compute the normalization factor $Q^{(r)}$ first.

The rest of the algorithm can be implemented directly. In particular, the total number of edges sampled in each iteration is $O(n \log(nW)/\varepsilon)$ with exponentially high probability by Chernoff bound. Thus, in the semi-streaming algorithm, we can store these edges and then compute $M^{(r)}$ and $(y^{(r)}, z^{(r)})$ at the end of the pass. This concludes the proof of the lemma. ■

PROOF OF RESULT 1.2. The only remaining part to prove Result 1.2 is to remove the dependence on the parameter W, which can be done using an entirely standard idea.

We can use a single pass to find the maximum weight edge w^* and subsequently, ignore all edges with weight less than $(\varepsilon/n) \cdot w^*$ because the total contribution of those edges to the maximum weight matching is always less than ε-fraction of its weight. Thus, we can assume that the weights are in $[1 : n/\varepsilon]$ from now on. Moreover, we can assume without loss of generality that $\varepsilon > 1/n^3$ as otherwise, we can simply store the entire graph in $O(n^2 \log(n/\varepsilon)) = O(n \log(n)/\varepsilon)$ bits (consistent with the bounds of Result 1.2) and trivially solve the problem in one pass (as a side note, we are only interested in much larger values of ε anyway).

This step implies that without loss of generality, when proving Result 1.2, we can assume all edges have integer weights bounded by n^4 by re-scaling ε to some $\Theta(\varepsilon)$. Combining this with Theorem 4.2 and Lemma 4.5 now gives an $O(n \log^2(n)/\varepsilon)$ memory algorithm in $O(\log(n)/\varepsilon)$ passes. ■

5. Further Extensions and Discussions

As stated earlier, in the interest of having a clear and concise exposition of our results, we opted to focus solely on the most important aspects of our algorithms in Result 1.1 and Result 1.2. We now present some natural extensions of our results and further discuss the connection between our work and the Plotkin-Shmoys-Tardos Framework [51] for solving covering/packing LPs via MWU.

5.1 Fewer Passes in More Space

Our algorithms, similar to that of [1], can be made more pass efficient at the cost of increasing the space, allowing us to prove the following result:

COROLLARY 5.1. *For every $p \geqslant 1$ and $\varepsilon \in (0,1)$, there is a randomized streaming algorithm for $(1-\varepsilon)$-approximation of maximum weight matching in $O(n^{1+1/p}\log^2(n)/\varepsilon)$ space and $O(p/\varepsilon)$ passes.*

This in particular means that if instead of semi-streaming space, we allow the streaming algorithm to use $n^{1+\delta}$ space for any constant $\delta > 0$, then, the number of passes is only $O(1/\varepsilon)$. This also implies that even for semi-streaming algorithms, the pass-complexity can be brought down to $O(\log(n)/(\log\log(n)\cdot\varepsilon))$ passes by taking $p = (\log n/\log\log n)$ and having $n^{1+1/p} = \widetilde{O}(n)$.

A common technique for proving Corollary 5.1, from an already-existing semi-streaming algorithm, is "delayed sampling" used, e.g., in [1, 33]: we can "fold" multiple passes of the semi-streaming algorithm into a single-pass of the larger-space algorithm by oversampling the input first, and then do rejection sampling (see [33, step 2 of Section 1.3] for more details). While this approach would work for us also, it would require (slightly) more space (by some $n^{1/p}/\varepsilon$ factors), and more importantly an indirect analysis.

Instead, one can directly adjust our importance-sampling based approach in Algorithm 4.1. It simply involves sampling the edges by a factor of $O(n^{1/p})$ more and then increasing the importance of violated edges in the algorithm, quite aggressively, by a factor of $O(n^{1/p})$ instead of only 2. We now formalize this proof.

PROOF OF COROLLARY 5.1. Let η be a parameter (that will be later chosen to be $n^{1/p}$). We prove this corollary by modifying Algorithm 4.1 as follows:

1. Increase the sampling rate in Eq (5) by a factor of η.
2. Increase the importance of any violated edge in Eq (6) by a factor of $(1+\eta)$ instead.

The implications of these changes are:

1. The space of the algorithm is increased by an $O(\eta)$ factor because we store a larger sample.
2. The upper bound of $(1+\varepsilon/2)^R \cdot W$ on the potential function in Lemma 4.3 still holds. This is because Eq (7) now holds with $\varepsilon/2$ replaced by $\varepsilon/2\eta$ which "cancels out" the effect of increasing importances by a $(1+\eta)$ factor instead.
3. On the other hand the lower bound of $2^{\varepsilon R}$ on the potential function in Lemma 4.4 simply becomes $(1+\eta)^{\varepsilon\cdot R}$ given this new update rule on the importances.

Thus, by combining the steps above, we get that

$$\varepsilon R \log\eta \leqslant 3\varepsilon/4 \cdot R + \log W$$

which gives us the bound of $R = O(\log W/(\log\eta\cdot\varepsilon))$.

Now, recall that in the proof of Result 1.2, we argued that we can take W to be at most n^4. This implies that by setting $\eta = n^{1/p}$, we get

$$R = O(\log W/(\log \eta \cdot \varepsilon)) = O(\log n/(1/p \cdot \log n \cdot \varepsilon)) = O(p/\varepsilon),$$

This concludes the proof of Corollary 5.1. ■

Before moving on, we remark that in addition to [1] and Corollary 5.1, [15] also provides an $O(n^{1.5}/\varepsilon)$-space, $O(1/\varepsilon)$-pass algorithm for maximum (cardinality) bipartite matching. The algorithm behind our Result 1.1 (and its extension in Corollary 5.1) turns out to be quite similar in hindsight to the algorithm of [15] that also relies on a sample-and-solve approach using vertex covers to guide their sampling; however, unlike our approach, [15] sticks to uniform sampling and does not adjust any importances, which leads to the larger space-complexity of $O(n^{1.5})$ instead of (essentially) $n^{1+o(1)}$-space in our work for $O(1/\varepsilon)$-pass algorithms.

5.2 Derandomization via Cut Sparsifiers

The failure probability of our algorithms in Result 1.1 and Result 1.2 are exponentially small, which is better than the typical 'with high probability bounds' (namely, $1 - 1/\text{poly}(n)$ bounds) in the same context. But, in fact, we can fully derandomize the algorithms also at the cost of increasing the space by $\text{poly}(\log n, 1/\varepsilon)$ factors and keeping the number of passes asymptotically the same. We state and prove this part only for unweighted graphs, but with more technical work, one can also extend this to weighted graphs (we omit the latter result as it requires too much of a detour).

COROLLARY 5.2. *For every $p \geqslant 1$ and $\varepsilon \in (0, 1)$, there is a deterministic streaming algorithm for $(1-\varepsilon)$-approximation of maximum cardinality matching in $\widetilde{O}(n^{1+1/p}/\varepsilon^2)$ space and $O(p/\varepsilon)$ passes.*

Corollary 5.2 is proven via replacing the sampling step of the algorithm with *cut sparsifiers* of [16]: these are (re-weighted) subgraphs of the input that preserve weights of cuts to within a $(1 \pm \varepsilon)$-approximation while having only $O(n \log(n)/\varepsilon^2)$ edges. We note that in general cut sparsifiers are not good at preserving large matchings[11] but appear to be good at preserving "near feasible" vertex covers and odd-set covers we need for our primal-dual analysis. Finally, it is known how to compute an ε-cut sparsifier in the streaming model using a single pass and $\widetilde{O}(n/\varepsilon^2)$ space deterministically using any deterministic static (non-streaming) algorithm for this problem, say [13]; see, e.g., [48] for this elegant and quite simple reduction (based on the *merge-and-reduce* technique dating back to the work of [46] on quantile estimation).

11 There are examples wherein a cut sparsifier of graph with a perfect matching may only have a maximum matching of size $(n^{1/2+o(1)})$ edges (e.g., a union of a perfect matching plus $(n^{1/2+o(1)})$ vertices connected to all other vertices).

Before getting to the formal proof, which focuses on Result 1.2 (and Corollary 5.1), let us see an intuition for this result by focusing on derandomizing Result 1.1 for MBM instead. Suppose in Algorithm 3.1, instead of sampling edges proportional to importances, we pick a $\Theta(\varepsilon)$-cut sparsifier H of G with edges weighted by the importances. Then, we simply pick a vertex cover of H (ignoring the weights now). We claim that Eq (4) still holds. Let $U \subseteq V$ be a "potential" vertex cover so that the total importance of edges it does *not* cover is $> (\varepsilon/2) \cdot Q$, where Q is the importance of all edges in G in this iteration. One can show that H needs to contain at least one edge entirely in $V \setminus U$ to be able to be a, say, $(\varepsilon/100)$-cut sparsifier of G. This means that U could have not been chosen as a vertex cover of H. Thus, vertex covers of H satisfy Eq (4) and the rest of the analysis is the same (recall that H is a subgraph of G so a "good" vertex cover always exist). We now formalize the proof.

PROOF OF COROLLARY 5.2. We focus on proving the result for semi-streaming algorithms; extending this approach to larger-space algorithms with fewer passes is exactly as in Corollary 5.1 and thus we do not repeat the argument.

The algorithm is the following. Instead of sampling edges in Algorithm 4.1, we compute an $(\varepsilon/100)$-cut sparsifier H of G whose edges are weighted by the importances in this iteration. Then, we compute an odd-set cover (y, z) of H, ignoring all edge weights in this step, and continue exactly as before. Given that a cut sparsifier can be computed in $\widetilde{O}(n/\varepsilon^2)$ space in the semi-streaming model (see [48]), we will obtain the desired deterministic algorithm.

Recall that the sampling step was only used in the analysis in Lemma 4.3 and in particular to establish Eq (7) with an exponentially high probability. We instead show that this new approach deterministically satisfies Eq (7). The rest of the proof of Corollary 5.2 then follows verbatim as in Result 1.2. Thus, we only need to show the following:

— Let H be an $(\varepsilon/100)$-cut sparsifier of $G = (V, E)$ under the edge weights q_e. Then, any odd-set cover (y, z) of H satisfies Eq (7) deterministically, i.e.,

$$\sum_{e \in F(y,z)} q_e \leqslant (\varepsilon/2) \cdot Q, \tag{8}$$

where $F(y, z)$ is the set of violated edges by (y, z) in the unweighted graph G, and q_e and Q are the importance of edge $e \in E$, and total importance of all edges, respectively.

We now prove this statement. In the following, for a graph $G = (V, E)$ and two disjoint sets of vertices $A, B \subseteq V$, we define $cut_G(A)$ and $cut_G(A, B)$ as the weight of the cuts $(A, V \setminus A)$ and (A, B), respectively (we apply this to G with weight function being edge importances, and to H with the re-weighted weights of the sparsifier).

Let (y, z) be an optimal odd-set cover of H. By Cunningham-Marsh theorem (Fact 2.3), y and z are both integral and $\mathcal{F}(z) := \{S \in odd(V) \mid z_S > 0\}$ forms a laminar family. Moreover, given the optimality of (y, z) and since H is unweighted (when calculating the odd-set cover), we have that $y, z \in \{0, 1\}^n$ which implies that $\mathcal{F}(z) = S_1, S_2, \ldots, S_s$ is actually a collection of disjoint sets, and is disjoint from the support of y, denoted by T. Notice that the set of violated edges by (y, z) in G are the ones that are not inside $S_1, \ldots, S_s$, nor incident on T.

Suppose towards a contradiction that Eq (8) does not hold. Let (A, B) be a maximum cut of the graph $G[V \setminus T]$ among all cuts where each $S_i \in \mathcal{F}(z)$ is entirely on one side of the cut. Since a maximum cut always has weight at least half of the weight of edges in the graph, we have,

$$cut_G(A, B) > (\varepsilon/4) \cdot Q.$$

On the other hand, in any graph, we also have

$$cut_G(A) + cut_G(B) = cut_G(A \cup B) + cut_G(A, B).$$

Given that $cut_G(A), cut_G(B), cut_G(A \cup B) \leqslant Q$ trivially, and since H is an $(\varepsilon/100)$-cut sparsifier,

$$\begin{aligned} cut_H(A, B) &= cut_H(A) + cut_H(B) - cut_H(A \cup B) \\ &\geqslant cut_G(A) + cut_G(B) - cut_G(A \cup B) - 3 \cdot (\varepsilon/100) \cdot Q \\ &= cut_G(A, B) - 3 \cdot (\varepsilon/100) \cdot Q \\ &\geqslant (\varepsilon/4) \cdot Q - 3 \cdot (\varepsilon/100) \cdot Q > 0, \end{aligned}$$

which implies that there is at least one edge between A and B in H. But recall that none of the edges between A and B were covered by (y, z), contradicting the fact that (y, z) was a feasible odd-set cover of H. This proves Eq (8). ■

We shall remark that we were inspired by the use of cut sparsifiers in [1] for this part of the argument. Although, to our knowledge, the use of sparsifiers in [1] is for a different purpose of their "delayed sparsification" and folding $O(1/\varepsilon)$ iterations of their optimization method in $O(1)$ passes; we are instead using them for derandomization purposes (the algorithms of [1] are randomized despite using sparsifiers even in insertion-only streams).

5.3 Running Times of Our Algorithms

Given that the main resource of interest in the streaming model is the space, we did not put any emphasis on the runtime of our algorithms in the preceding discussions. It is clear that our algorithms run in polynomial time since finding maximum matchings and minimum vertex covers in bipartite graphs or minimum odd-set covers in general graphs can all be done in polynomial time. However, our algorithms can be made more time efficient, captured by the following corollary.

COROLLARY 5.3. *Both algorithms in Result 1.1 and Result 1.2 can be implemented in $\widetilde{O}(m/\varepsilon^2 + n/\varepsilon^3)$ time and the same asymptotic space and pass complexity.*

We note that in the semi-streaming model, our algorithms can only handle values of ε such that $1/\varepsilon = \text{poly}\log(n)$, as otherwise the space of the algorithm will be more than the semi-streaming restriction of $\widetilde{O}(n)$ bits. In this regime, our algorithms run in near-linear time.

We first recall the seminal algorithm of [23]—generalizing classical results in [31] for bipartite graphs—which we shall use as a blackbox in our algorithms.

PROPOSITION 5.4 ([23]). *There exists an algorithm that given any graph $G = (V, E)$ and any parameter $\varepsilon > 0$, outputs a $(1 - \varepsilon)$-approximate maximum weight matching and a $(1 + \varepsilon)$-approximate minimum odd-set cover of G in $O(m/\varepsilon)$ time. Moreover, the support of the odd-set cover returned by the algorithm forms a laminar family.*

We can now present a proof of Corollary 5.3.

PROOF OF COROLLARY 5.3. The only modification we need for this corollary is that in each iteration of Algorithm 3.1 (for Result 1.1) or Algorithm 4.1 (for Result 1.2), instead of finding an exact maximum matching or minimum vertex cover/odd-set cover, we run the algorithm in Proposition 5.4 to find a $(1 - \varepsilon)$-approximation to these problems on the sampled graph.

As for the analysis, Lemma 3.3 and Lemma 4.3 hold as before as they only relied on the feasibility of the dual solution, not their optimality. On the other hand, Lemma 3.4 and Lemma 4.4 relied on the optimality of the dual solutions in each step. However, both these results still hold with a $(1 - \varepsilon)$-approximation loss, namely, the conclusion of the lemmas hold as long as we assume

$$|M^{(r)}| < (1-\varepsilon)^2 \cdot \mu(G) \qquad \text{or} \qquad w(M^{(r)}) < (1-\varepsilon)^2 \cdot \mu(G, w);$$

this is simply because these conditions, plus the guarantee of Proposition 5.4, imply the same prior bounds on the value of optimum solution in the sampled graph as needed in Lemma 3.4 and Lemma 4.4. The rest of the analysis follows verbatim as before.

All in all, this ensures that the total runtime of the algorithms is only $\widetilde{O}(m/\varepsilon^2 + n/\varepsilon^3)$ time: there are $O(\log(n)/\varepsilon)$ passes and each pass takes $\widetilde{O}(m/\varepsilon)$ time for reading the stream and sampling the edges plus $\widetilde{O}(n/\varepsilon^2)$ time for running the algorithm of [23] on the sampled subgraph of size $\widetilde{O}(n/\varepsilon)$ edges. This concludes the proof. ■

An alternative way of looking at Corollary 5.3, from a purely time-complexity point of view, is the following: to obtain a $(1 - \varepsilon)$-approximation to maximum (weight) matching on arbitrary graphs, we only need to have an oracle for finding a $(1 - \Theta(\varepsilon))$-approximation to the same problem on *sparse* graphs with $\widetilde{O}(n/\varepsilon)$ edges; running this oracle for $O(\log(n)/\varepsilon)$ times on different sampled subgraphs of the input (plus some low overhead computation for updating the weights and performing the sampling), gives us a $(1 - \varepsilon)$-approximation on any (not necessarily sparse) graph as well.

5.4 Extension to Other Related Models

Our algorithmic approach in this paper is quite flexible and easily extends to many other models. In particular, given its sample-and-solve nature, the algorithm can be implemented via a linear sketch (see [4]), which also implies the following two results:

COROLLARY 5.5 (Extension to Dynamic streams). *There is a randomized semi-streaming algorithm for $(1-\varepsilon)$-approximation of weighted (general) matching in dynamic streams—with edge insertions and deletions—that for every $p \geqslant 1$, uses $\widetilde{O}(n^{1+1/p}/\varepsilon)$ space and $O(p/\varepsilon)$ passes.*

COROLLARY 5.6 (Extension to Massively Parallel Computation (MPC)). *There is a randomized MPC algorithm for $(1-\varepsilon)$-approximation of weighted (general) matchings that for every $p \geqslant 1$, uses machines of memory $\widetilde{O}(n^{1+1/p}/\varepsilon)$ and $\widetilde{O}(n^{1+1/p}/\varepsilon)$ global memory beside the input size, and $O(p/\varepsilon)$ rounds.*

As this is not the focus of the paper, we omit the definition and details of the models and instead refer the interested to [4, 48] and [37, 14] for each model, respectively. We only note that the prior results in [1] also achieved similar corollaries but this is not the case for the approach of [9].

5.5 Explicit Connections to MWU and Plotkin-Shmoys-Tardos Framework

It turns out that our algorithms can be cast in the Plotkin-Shmoys-Tardos (PST) Framework [51] for solving covering/packing LPs via MWU—despite the fact that the number of iterations is $O(\log(n)/\varepsilon^2)$ in this framework—by making the following observation (this discussion assumes a basic familiarity with this framework; see [5] for a quick introduction).

Firstly, suppose we use the PST framework for solving the (fractional) vertex/odd-set cover LP, which translates to maintaining "MWU weights" over the edges. The goal, perhaps counter intuitively, is to *fail* in finding a small vertex/odd-set cover, which implies we have found a "witness" to the existence of a large matching in the graph. The *oracle* used for the PST framework can be implemented by our sampling approach. The key observation is that this oracle is *extremely efficient* compared to a typical oracle, in that, it achieves a very accurate solution with a very small *width*. This leads to something interesting: the weights of the variables, under the "cautious" update rules of MWU, grows so slowly that the *same* oracle solution remains *approximately* valid for the next $O(1/\varepsilon)$ iterations!

The implication of the above for semi-streaming algorithms is that, effectively, one only needs to rerun the oracle, using another pass over the stream, for every $O(1/\varepsilon)$ iterations of the framework. This allows running the $O(\log(n)/\varepsilon^2)$ iterations of this framework in $O(\log(n)/\varepsilon)$ passes.

We shall however caution the reader that while the above intuition is morally true, implementing the algorithm and following the standard analysis this way is quite "messy" and does not seem to yield to a necessarily simple algorithm nor analysis. Thus, we find the direct proof presented in the paper much more illuminating and opted to provide that instead[12].

6. Concluding Remarks and Open Questions

In this paper, we presented a rather complete simplification of the prior $O(\log(n)/\varepsilon)$-pass algorithms of [1] in our Result 1.1 and Result 1.2.

The key open question at this point—which was also the key motivation behind this work itself—is to obtain the best of both worlds among the two types of pass-complexity of semi-streaming algorithms obtained for bipartite (cardinality) matching: the $O(\log(n)/\varepsilon)$ passes of [1, 9] and Result 1.1, and the $O(1/\varepsilon^2)$ passes of [3, 10].

Open question 1: *Can we design a semi-streaming algorithm for maximum bipartite cardinality matching with $O(1/\varepsilon)$ passes?*

We would like to make a (rather bold) conjecture that the *"right"* pass-complexity of this problem might be even (much) lower than $O(1/\varepsilon)$ passes. But, at this point, we seem to be far from achieving such results or ruling out their possibilities[13].

A slightly less exciting question than the above is to *significantly* reduce the pass-complexity of the "constant-pass" algorithms for maximum matching in general graphs in [28, 32] to match the results for MBM in [10] (see Footnote 1). In particular,

Open question 2: *Can we obtain a semi-streaming algorithm for maximum matching in general graphs (weighted or unweighted) with $O(1/\varepsilon^2)$ passes?*

We hope that by simplifying the state-of-the-art, our results in this paper can pave the way for addressing these questions. Note that as stated earlier, we can indeed provide a positive answer to both questions for the (strictly) *more relaxed* case when the space of the algorithms is $n^{1+\delta}$ for constant $\delta > 0$ (this was also already known by the prior work of [1]).

Acknowledgement

Many thanks to Soheil Behnezhad, Shang-En Huang, Peter Kiss, Rasmus Kyng, and Thatchaphol Saranurak for helpful discussions and to the organizers of the DIMACS Workshop on "Modern Techniques in Graph Algorithms" (June 2023)—Prantar Ghosh, Zihan Tan, and Nicole Wein—where these conversations happened.

12 We should also add that this connection was only made *in hindsight* after having the new algorithm and analysis.

13 [9] provides a semi-streaming $n^{3/4+o(1)}$-pass algorithm for solving MBM exactly, which can be seen as $(1-\varepsilon)$-approximation for $\varepsilon = 1/n$. Thus, at least for very small values of ε, we already know $\ll 1/\varepsilon$-pass algorithms.

I am also grateful to Aaron Bernstein, Aditi Dudeja, Arun Jambulapati, Michael Kapralov, Sanjeev Khanna, Kent Quanrud, and Janani Sundaresan for various helpful discussions about this problem over the years.

Additionally, I would like to thank Soheil Behnezhad for pointing out the similarity of Result 1.1 with the $O(n^{1.5}/\varepsilon)$-space $O(1/\varepsilon)$-pass streaming algorithm of [15] for MBM (as discussed in Section 5.1), and to Vikrant Ashvinkumar and Lap Chi Lau for many helpful comments on the presentation of this paper.

Finally, I am quite grateful to the anonymous reviewers of TheoretiCS for numerous detailed and insightful comments, and for suggesting simpler proof of Lemma 3.4 (and by extension Lemma 4.4) that are presented in the current version of the paper.

References

[1] **Kook Jin Ahn and Sudipto Guha**. Access to data and number of iterations: dual primal algorithms for maximum matching under resource constraints. *ACM Trans. Parallel Comput.* 4(4):17:1–17:40, 2018. DOI (1–5, 16, 17, 19, 21, 22)

[2] **Kook Jin Ahn and Sudipto Guha**. Laminar families and metric embeddings: non-bipartite maximum matching problem in the semi-streaming model. *CoRR*, abs/1104.4058, 2011. (1–4)

[3] **Kook Jin Ahn and Sudipto Guha**. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.* 222:59–79, 2013. DOI (1–3, 22)

[4] **Kook Jin Ahn, Sudipto Guha, and Andrew McGregor**. Analyzing graph structure via linear measurements. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, SODA 2012, pages 459–467, 2012. DOI (21)

[5] **Sanjeev Arora, Elad Hazan, and Satyen Kale**. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.* 8(1):121–164, 2012. DOI (4, 8, 21)

[6] **Sepehr Assadi**. A simple $(1 - \varepsilon)$-approximation semi-streaming algorithm for maximum (weighted) matching. *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 337–354. SIAM, 2024. DOI (1)

[7] **Sepehr Assadi**. A two-pass (conditional) lower bound for semi-streaming maximum matching. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 708–742. SIAM, 2022. DOI (2, 3)

[8] **Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li**. On regularity lemma and barriers in streaming and dynamic matching. *Proceedings of the ACM Symposium on Theory of Computing, STOC 2023*, pages 131–144. ACM, 2023. DOI (3)

[9] **Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian**. Semi-streaming bipartite matching in fewer passes and optimal space. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 627–669. SIAM, 2022. DOI (1–4, 21, 22)

[10] **Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan**. An auction algorithm for bipartite matching in streaming and massively parallel computation models. *4th Symposium on Simplicity in Algorithms, SOSA 2021*, pages 165–171. SIAM, 2021. DOI (1–3, 22)

[11] **Sepehr Assadi and Ran Raz**. Near-quadratic lower bounds for two-pass graph streaming algorithms. *IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 342–353. IEEE, 2020. DOI (2)

[12] **Sepehr Assadi and Janani Sundaresan**. Hidden permutations to the rescue: multi-pass streaming lower bounds for approximate matchings. *IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 909–932. IEEE, 2023. DOI (1–3)

[13] **Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava**. Twice-ramanujan sparsifiers. *SIAM J. Comput.* 41(6):1704–1721, 2012. DOI (17)

[14] **Paul Beame, Paraschos Koutris, and Dan Suciu**. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017. DOI (21)

[15] **Soheil Behnezhad, Mahsa Derakhshan, Hossein Esfandiari, Elif Tan, and Hadi Yami**. Brief announcement: graph matching in massive datasets. *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017*, pages 133–136. ACM, 2017. DOI (1, 3, 17, 23)

[16] **András A. Benczúr and David R. Karger**. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. *Proceedings of the ACM Symposium on the Theory of Computing, STOC 1996*, pages 47–55. ACM, 1996. DOI (17)

[17] **Aaron Bernstein, Aditi Dudeja, and Zachary Langley**. A framework for dynamic matching in weighted graphs. *Proceedings of the ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 668–681. ACM, 2021. DOI (3)

[18] **Lidiya Khalidah binti Khalil and Christian Konrad**. Constructing large matchings via query access to a maximal matching oracle. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020*, volume 182 of *LIPIcs*, 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. DOI (3)

[19] **Binh-Minh Bui-Xuan, Michel Habib, and Michaël Rao**. Tree-representation of set families and applications to combinatorial decompositions. *Eur. J. Comb.* 33(5):688–711, 2012. DOI (13)

[20] **Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu**. Almost optimal superconstant-pass streaming lower bounds for reachability. *SIAM Journal on Computing*, 2024. DOI (2, 3)

[21] **Michael B. Cohen, Aaron Sidford, and Kevin Tian**. Relative lipschitzness in extragradient methods and a direct recipe for acceleration. *Innovations in Theoretical Computer Science Conference, ITCS 2021*, volume 185 of *LIPIcs*, 62:1–62:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. DOI (4)

[22] **William H Cunningham and AB Marsh**. A primal algorithm for optimum matching. *Polyhedral Combinatorics: Dedicated to the memory of DR Fulkerson*:50–72, 1978. (7)

[23] **Ran Duan and Seth Pettie**. Linear-time approximation for maximum weight matching. *J. ACM*, 61(1):1:1–1:23, 2014. DOI (20)

[24] **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. (7)

[25] **Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav**. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012. DOI (1, 3)

[26] **Sebastian Eggert, Lasse Kliemann, and Anand Srivastav**. Bipartite graph matchings in the semi-streaming model. *Algorithms - ESA 2009, 17th Annual European Symposium, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 492–503. Springer, 2009. DOI (1, 3)

[27] **Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang**. On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348(2-3):207–216, 2005. (1)

[28] **Manuela Fischer, Slobodan Mitrovic, and Jara Uitto**. Deterministic $(1+\epsilon)$-approximate maximum matching with poly$(1/\epsilon)$ passes in the semi-streaming model and beyond. *54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022*, pages 248–260. ACM, 2022. DOI (1–3, 22)

[29] **Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson**. Weighted matchings via unweighted augmentations. *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2019*, pages 491–500, 2019. DOI (1, 3)

[30] **Ashish Goel, Michael Kapralov, and Sanjeev Khanna**. On the communication and streaming complexity of maximum bipartite matching. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 468–485. SIAM, 2012. DOI (3)

[31] **John E. Hopcroft and Richard M. Karp**. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* 2(4):225–231, 1973. (20)

[32] **Shang-En Huang and Hsin-Hao Su**. $(1-\varepsilon)$-approximate maximum weighted matching in distributed, parallel, and semi-streaming settings. *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2023*, pages 44–54. ACM, 2023. DOI (1–3, 22)

[33] **Piotr Indyk, Sepideh Mahabadi, Ronitt Rubinfeld, Jonathan R. Ullman, Ali Vakilian, and Anak Yodpinyanee**. Fractional set cover in the streaming model. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, volume 81 of *LIPIcs*, 12:1–12:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. DOI (16)

[34] **Arun Jambulapati, Aaron Sidford, and Kevin Tian**. A direct tilde{o}(1/epsilon) iteration parallel algorithm for optimal transport. *Annual Conference on Neural Information Processing Systems, NeurIPS 2019*, pages 11355–11366, 2019. (4)

[35] **Michael Kapralov**. Better bounds for matchings in the streaming model. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1679–1697, 2013. DOI (1, 3)

[36] **Michael Kapralov**. Space lower bounds for approximating maximum matching in the edge arrival model. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1874–1893. SIAM, 2021. DOI (3)

[37] **Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii**. A model of computation for mapreduce. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 938–948, 2010. DOI (21)

[38] **Philip N. Klein and Neal E. Young**. On the number of iterations for dantzig-wolfe optimization and packing-covering approximation algorithms. *SIAM J. Comput.* 44(4):1154–1172, 2015. DOI (4)

[39] **Dénes Kőnig**. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931. (6)

[40] **Christian Konrad and Kheeran K Naidu**. An unconditional lower bound for two-pass streaming algorithms for maximum matching approximation. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 2881–2899. SIAM, 2024. DOI (2, 3)

[41] **Christian Konrad and Kheeran K. Naidu**. On two-pass streaming algorithms for maximum bipartite matching. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021*, volume 207 of *LIPIcs*, 19:1–19:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. DOI (3)

[42] **Christian Konrad**, **Kheeran K. Naidu**, and **Arun Steward**. Maximum matching via maximal matching queries. *International Symposium on Theoretical Aspects of Computer Science, STACS 2023*, volume 254 of *LIPIcs*, 41:1–41:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI (3)

[43] **Ravi Kumar**, **Benjamin Moseley**, **Sergei Vassilvitskii**, and **Andrea Vattani**. Fast greedy algorithms in mapreduce and streaming. *ACM Trans. Parallel Comput.* 2(3):14:1–14:22, 2015. DOI (2, 4, 5)

[44] **Silvio Lattanzi**, **Benjamin Moseley**, **Siddharth Suri**, and **Sergei Vassilvitskii**. Filtering: a method for solving graph problems in MapReduce. *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2011*, pages 85–94, 2011. DOI (2, 4, 5)

[45] **László Lovász and Michael D Plummer**. Matching theory, volume 367. American Mathematical Soc., 2009. (6, 7)

[46] **Gurmeet Singh Manku**, **Sridhar Rajagopalan**, and **Bruce G. Lindsay**. Approximate medians and other quantiles in one pass and with limited memory. *Proceedings ACM SIGMOD International Conference on Management of Data, SIGMOD 1998*, pages 426–435. ACM Press, 1998. DOI (17)

[47] **Andrew McGregor**. Finding graph matchings in data streams. *8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Proceedings*, pages 170–181, 2005. DOI (1, 3)

[48] **Andrew McGregor**. Graph stream algorithms: a survey. *SIGMOD Rec.* 43(1):9–20, 2014. DOI (17, 18, 21)

[49] **Arkadi Nemirovski**. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.* 15(1):229–251, 2004. DOI (4)

[50] **Yurii E. Nesterov**. Dual extrapolation and its applications to solving variational inequalities and related problems. *Math. Program.* 109(2-3):319–344, 2007. DOI (4)

[51] **Serge A. Plotkin**, **David B. Shmoys**, and **Éva Tardos**. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.* 20(2):257–301, 1995. DOI (4, 15, 21)

[52] **Jonah Sherman**. Area-convexity, l_∞ regularization, and undirected multicommodity flow. *Proceedings of the ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 452–460. ACM, 2017. DOI (4)

[53] **Sumedh Tirodkar**. Deterministic algorithms for maximum matching on general graphs in the semi-streaming model. *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018*, 39:1–39:16, 2018. DOI (1, 3)