Adaptive Sparsification for Matroid Intersection

Kent Quanrud
Dept. of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

We consider the matroid intersection problem in the independence oracle model. Given two matroids over n common elements such that the intersection has rank k, our main technique reduces approximate matroid intersection to logarithmically many primal-dual instances over subsets of size $\tilde{O}(k)$. This technique is inspired by recent work by [2] and requires additional insight into structuring and efficiently approximating the dual LP. This combination of ideas leads to faster approximate maximum cardinality and maximum weight matroid intersection algorithms in the independence oracle model. We obtain the first nearly linear time/query approximation schemes for the regime where $k \leq n^{2/3}$.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Matroid intersection, adaptive sparsification, multiplicative-weight udpates, primal-dual

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.118

Category Track A: Algorithms, Complexity and Games

Funding *Kent Quanrud*: Supported in part by NSF grant CCF-2129816.

Acknowledgements We thank the reviewers for their careful and helpful feedback. We thank Adrian Vladu for teaching us about [2].

1 Introduction

Matroid intersection is a classical problem in combinatorial optimization for which faster algorithms have been a recent topic of interest.

A *matroid*, $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, consists of a set of n elements $\mathcal{N}$ and a collection of subsets $\mathcal{I}$ of $\mathcal{N}$, known as the independent sets, that satisfy the following properties: (i) the empty set is independent, (ii) every subset of an independent set is independent (hereditary property), and (iii) if A and B are two independent sets with $|A| > |B|$, then there exists an element in $A \setminus B$ that can be added to B to still have an independent set (exchange property). These properties imply that every *maximal* set also has maximum cardinality. The maximum cardinality of any independent set is called the *rank*. Examples of matroids include the family of forests of a graph (the graphic matroid) and the family of independent sets of vectors in a vector space (the linear matroid).

Matroid intersection. The problem of matroid intersection considers two matroids, $\mathcal{M}_1 = (\mathcal{N}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{N}, \mathcal{I}_2)$, defined on a common ground set $\mathcal{N}$, and seeks the largest set that is independent in both matroids. Formally, the goal is to maximize the size of a set $S \subseteq \mathcal{N}$ such that $S \in \mathcal{I}_1 \cap \mathcal{I}_2$. Unlike matroids, a maximal cardinality independent set in the matroid intersection is not necessarily a maximum cardinality independent set. The maximum cardinality, denoted OPT, is also called the *rank* of the matroid intersection and denoted by k. Matroid intersection generalizes bipartite matching, and has other connections in combinatorial optimization. For example, by Edmonds' directionless tree packing theorem,

matroid intersection captures the maximum number of rooted arborescences that can be packed into a directed graph, and the directed rooted connectivity [11]. See [27, 15] for additional background and connections.

Algorithms addressing matroid intersection in general are commonly framed in the *independence oracle model*. Here the algorithm is allowed to query if a given set S is independent in a matroid. When stating running times in this model, we let Q denote the running time of a single independence oracle.

The first polynomial time algorithm for matroid intersection was given by [10] by reduction to matroid union [12]. This algorithm ran in $O\big(n^4Q\big)$ time. More direct augmenting path algorithms were developed by [1, 21]. The algorithm in [21] ran in $O\big(nk^2Q\big)$ time. Generalizing ideas from bipartite matching [17], [9] gave a faster matroid intersection algorithm running in $O\big(nk^{1.5}Q\big)$ time. Truncating the algorithm early implies a $(1-\epsilon)$-approximation in $O(nkQ/\epsilon)$ time for $\epsilon \in (0,1)$ [8].

Recently there has been a resurgence of interest in faster algorithms in the independence oracle model. [7, 23] gave $O(nk\log(k)Q)$ time algorithms, leveraging the observation that the auxiliary graph can be searched faster than it can be built out explicitly. [5] pushed this direction further and obtained $\tilde{O}\big(n^{9/5}Q\big)$ randomized and $\tilde{O}\big(n^{11/6}Q\big)$ deterministic time algorithms, the first $o(n^2)$ time algorithms for $k=\Omega(n)$.[1] Finally, [4] obtained a $\tilde{O}\Big(n\sqrt{k}Q/\epsilon\Big)$ time deterministic algorithm for $(1-\epsilon)$-matroid intersection. This faster approximation algorithm implied faster exact algorithms running in $\tilde{O}\big(nk^{3/4}Q\big)$ randomized time and $\tilde{O}\big(nk^{5/6}Q\big)$ deterministic time. [4]'s algorithms represent the state of the art.

Weighted matroid intersection. In the weighted matroid intersection problem, we are also given weights $c:\mathcal{N}\to\mathbb{R}_{>0}$. The goal is to compute $I\in\mathcal{I}_1\cap\mathcal{I}_2$ of maximum weight $c(I)$. Edmonds [10] gave the first polynomial time algorithm. Faster algorithms were developed in [21, 13, 6, 16, 22]. Frank's citeFrank1981a algorithm runs in $O(k(T+n\log n))$ time, where T is the running time of any exact matroid intersection algorithm. The algorithm in [22] runs in $O\big(n^2\log(n)Q+n^3\operatorname{polylog}(n)\big)$ time.

There is also recent interest in fast $(1-\epsilon)$-approximation algorithms [18, 8]. The $(1-\epsilon)$-approximation algorithm in [8] runs in $\tilde{O}\big(nkQ/\epsilon^2\big)$ time.

1.1 Results

Our primary focus is on faster approximation algorithms. As alluded to above, approximation algorithms can play a role in exact algorithms. (The fastest algorithms use augmenting paths to extend approximate solutions to exact ones.) They are also useful in their own right when one is willing to tolerate some error in exchange for scalability.

We introduce adaptive sparsification to matroid intersection in order to develop faster approximation algorithms. The new sparsification technique reduces approximate matroid intersection to $O(\log(n))$ instances of approximate matroid intersection and the dual problem over a subset of $O(k\log(n))$ elements. The technique holds for both weighted and unweighted matroid intersection. We leverages these ideas to obtain improved running times for approximating the unweighted and weighted settings.

Maximum cardinality matroid intersection. Henceforth, let $\mathcal{M}_1=(\mathcal{N},\mathcal{I}_1)$ and $\mathcal{M}_2=(\mathcal{N},\mathcal{I}_2)$ be two matroids over a common groundset of n elements, let k be the rank of their intersection, and let $\epsilon\in(0,1)$. In the following, the "dual" refers to the dual of the standard

[1] $\tilde{O}(\cdots)$ hides polylogarithmic factors.

packing LP for matroid intersection, and is introduced formally below in Section 1.2. We first show how to reduce approximate matroid intersection over n elements to approximating $O(\log n)$ primal-dual instances of matroid intersection over $\tilde{O}(k)$ elements.

▶ **Lemma 1.** *Suppose a $(1-\epsilon)$-maximum matroid intersection and a $(1+\epsilon)$-approximate dual solution in $\mathcal{I}_1 \cap \mathcal{I}_2$, over a subset of m elements, can be computed with high probability in $\mathcal{T}_\epsilon(m)$ time in the independence oracle model. Then a $(1-\epsilon)$-approximate matroid intersection can be computed with high probability in $O(n\log(n)Q/\epsilon + \mathcal{T}_\epsilon(k\log(n)/\epsilon)/\epsilon)$ randomized time in the independence oracle model, where Q represents an independence query.*

To apply Lemma 1 to matroid intersection, we need a fast algorithm to compute both a $(1-\epsilon)$-maximum matroid intersection and an $(1+\epsilon)$-dual solution. Recall that [4] computes a $(1-\epsilon)$-maximum matroid intersection in $\tilde{O}\left(n\sqrt{k}Q/\epsilon\right)$ time. The solution returned by [4] has stronger properties (based on the length of augmenting paths), and we leverage these properties to compute a $(1+\epsilon)$-approximate dual solution without increasing the running time.

▶ **Lemma 2.** *A $(1-\epsilon)$-maximum matroid intersection I, and an $(1+\epsilon)$-dual solution (S,T), can be computed in $\tilde{O}\left(n\sqrt{k}Q/\epsilon\right)$ deterministic time.*

Using this as a $(1\pm\epsilon)$-primal dual approximation algorithm in Lemma 1, with $\mathcal{T}_\epsilon(n) = \tilde{O}\left(n\sqrt{k}Q/\epsilon\right)$, we have the following improved running time for approximate matroid intersection.

▶ **Theorem 3.** *A $(1-\epsilon)$-maximum matroid intersection and an $(1+\epsilon)$-dual solution can be computed with high probability in $O\left(n\log(n)Q/\epsilon + k^{3/2}\log^{O(1)}(k)Q/\epsilon^3\right)$ randomized time in the independence oracle model.*

Compared to previous results, the improved running time in Theorem 3 removes the $\text{poly}(k)$-factor from the dominant term of n. Theorem 3 gives the first nearly linear time approximation scheme for the regime where $k \le n^{2/3}$.

Theorem 3 does not imply a faster algorithm for exact matroid intersection, but brings us significantly closer. This is because there are two bottlenecks in [4]. Theorem 3 addresses one of them. The remaining bottleneck is a subroutine augmenting an independent set one element at a time. The current best bound for this subroutine is $\tilde{O}\left(n\sqrt{k}Q\right)$ time per augmenting path [5].

Maximum weight matroid intersection. We now consider maximum weight matroid intersection. In addition to the inputs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\epsilon \in (0,1)$, let $c : \mathcal{N} \to \mathbb{R}_{\ge 0}$ be an input weight vector.

As in the unweighted case, we show how to reduce approximate maximum weight matroid intersection to approximating $O(\log(n))$ primal-dual instances over subsets of $\tilde{O}(k)$ elements. Here the "dual" refers to the dual LP, introduced later in Section 3. For the weighted setting, the sparsification technique requires particularly structured dual solutions which we call "compact" dual solutions. We elaborate more on compact dual solutions in Section 3 and for the time being state the lemma informally.

▶ **Lemma 4** (Informal). *Suppose that a $(1-\epsilon)$-maximum weight matroid intersection and a "compact" $(1+\epsilon)$-minimum dual solution over a subset of m elements can be computed in $\mathcal{T}_\epsilon(m)$ time with high probability. Then a $(1-\epsilon)$-maximum weight matroid intersection and a $(1+\epsilon)$-minimum dual solution can be computed with high probability in $O(n\log(n)Q/\epsilon + \mathcal{T}_\epsilon(k\log(n)/\epsilon)/\epsilon)$ randomized time in the independence oracle model.*

Using this technique requires a $(1 \pm \epsilon)$-primal-dual approximation algorithm where the dual solution has nice "compact" properties. A fast primal algorithm is given by [8] (accelerated by [4]) and we extend it to give an approximate dual solution that is also compact.

▶ **Lemma 5** (Informal). *A $(1-\epsilon)$-maximum weight matroid intersection and a "compact" $(1+\epsilon)$-minimum dual solution can be computed in $\tilde{O}\left(n\sqrt{k}Q/\epsilon^2\right)$ time.*

Putting these two results together gives the following improved running time for approximate maximum weight matroid intersection.

▶ **Theorem 6.** *A $(1-\epsilon)$-maximum weight matroid intersection and a $(1+\epsilon)$-minimum dual solution can be computed with high probability in $O\left(n\log(n)Q/\epsilon + k^{3/2}Q/\epsilon^4\right)$ randomized time in the independence oracle model.*

Compared to the previous state of the art, Theorem 6 removes the $\text{poly}(k)$-factor from the dominant term n. It also reduces the $\text{poly}(1/\epsilon)$-factor against n, from $1/\epsilon^2$ to $1/\epsilon$. Theorem 6 is the first approximation scheme for weighted matroid intersection running in nearly linear time for all $k \leq n^{2/3}$.

1.2 High-level overview of the algorithms and techniques

The first technique we introduce to matroid intersection is adaptive sparsification. In graph algorithms, sparsification is a powerful and (by now) standard technique where a dense input graph is reduced to a sparse one, while (approximately) preserving salient properties like the size of every cut [3] or the Laplacian [29, 28]. These algorithms are also fast. There is previous work sparsifying matroids individually [20, 25, 26], generalizing cut sparsification.

For matroid intersection, we are aware of two instances of sparsification. [7] sparsifies matroid intersection by first approximating a linear relaxation for matroid intersection. They then randomly round their solution x to a solution y with support of size $O\left(k\log(n)/\epsilon^2\right)$. Concentration bounds from [20] imply that the support of y contains a $(1-\epsilon)$-approximate matroid intersection with high probability. The only catch to this approach is that it takes $\tilde{O}\left(n^2Q/k\epsilon^2\right)$ time to compute the point x. The second instance is the very recent work of [19], which reduces the number of elements to $\tilde{O}\left(k/\epsilon^{O(1)}\right)$ while preserving the value of the matroid intersection up to a $(3/2+\epsilon)$-factor. However this is far from preserving the intersection up to a $(1+\epsilon)$-factor. (We note that the techniques of [19] have additional motivating factors including communication complexity and the streaming model.) It seems difficult to accurately preserve the matroid intersection via a "one-shot" static sparsifier without introducing another bottleneck in the running time.

This work pivots away from static sparsifiers to *adaptive* ones, where a large instance of matroid intersection is reduced to a limited number of sparse instances. The sparse instances are generated sequentially by random samples, where the distribution of each sample adapts to the outcomes of previous iterations. We are inspired by and build upon a recent and elegant work by Assadi [2], which used adaptive sampling to compute $(1-\epsilon)$-approximate maximum weight matchings in the semi-streaming model. We briefly sketch the ideas from [2]. While [2]'s techniques extend to general graphs, we restrict our discussion to bipartite matching as it is a special case of matroid intersection. The input is a bipartite graph $G = (V, E)$, and we are constrained to memory of size $\tilde{O}(|V|)$. In particular, for dense graphs, one cannot hold the entire graph in memory. The algorithm may read the edges E one by one in a streaming fashion; each iteration over E is called a "pass".

There are several known results in this model and the contribution of [2] was to give a simpler algorithm competitive with the state of the art. [2] reduces $(1-\epsilon)$-maximum weight matching to $O(\log(n)/\epsilon)$ successive instances of $(1-\epsilon)$-maximum weight matching and $(1+\epsilon)$-minimum vertex cover over subgraphs of $\tilde{O}(|V|/\epsilon)$ edges. Each $\tilde{O}(|V|/\epsilon)$-size instance is obtained by a nonuniform sample of the edges that can be implemented in a single pass over the edges.

The probabilities are based on multiplicative weights. Initially, all edges have the same weight and are sampled uniformly. Each iteration, the algorithm obtains a $(1+\epsilon)$-minimum vertex cover that covers all the sampled edges, but not necessarily all the input edges. The sampling probability of each uncovered edge is doubled. Intuitively the algorithm is trying to sample a small set of edges that forces the dual covering solutions of the sample to cover all the edges, at least on average. Edges that are frequently covered have exponentially smaller weight; edges that are not covered much have exponentially larger weight.

While [2] gives a direct analysis, one can interpret their algorithm a little more generally within a standard MWU framework applied to the dual vertex cover LP. We broaden the argument to covering problems in general. In the following, a $(1+\epsilon)$-approximation algorithm refers to a point $x \in \mathcal{P}$ such that $\langle b, x\rangle \le (1+\epsilon)\,\mathrm{OPT}$ and $(1+\epsilon)Ax \ge \mathbb{1}$.

▶ **Lemma 7.** *Consider a covering LP of the form*

$$\textit{mimimize } \langle b, x\rangle \textit{ over } x \in \mathcal{P} \textit{ s.t. } Ax \ge \mathbb{1},$$

where $\mathcal{P}$ is a convex set, and A has nonnegative coefficients and m constraints. Suppose one has access to an oracle that, given any nonnegative set of weights $w \in \mathbb{R}^m_{\ge 0}$, computes a point x such that:

(a) $\langle b, x\rangle \le (1+\epsilon)\,\mathrm{OPT}$

(b) $\sum_{i:(Ax)_i \ge 1} w_i \ge (1-\epsilon)\sum_i w_i$.

Suppose the oracle also returns a list of all constraints $i \in [m]$ covered by x (i.e., such that $(Ax)_i \ge 1$). Then one can compute a $(1+\epsilon)$-approximation solution as the average of $L = O(\log(m)/\epsilon)$ solutions returned by the oracle for an adaptively chosen sequence of L weight vectors.

The intuition for the oracle problem is as follows. We are given a covering LP, and the challenge is to satisfy *all* the covering constraints simultaneously. The oracle problem relaxes this uniform requirement by assigning nonnegative weights to each constraint, and asks for a solution that satisfies *most* of the constraints by weight. The small difference between satisfying all constraints, and satisfying *almost* all the constraints, is just large enough to permit random sampling and other techniques that trade a controlled amount of error for significantly faster running times. The surrounding framework adjusts the weights dynamically so that on average, the oracle solutions (approximately) cover all the constraints simultaneously.

Matroid intersection. To apply this framework to matroid intersection we must understand the dual covering problem, which requires the notions of a rank function and a span function. For a given matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, and set $S \subseteq \mathcal{N}$, the *rank of* S, denoted $\mathrm{rank}(S)$, is the maximum cardinality of any independent subset of S. For $S \subseteq \mathcal{N}$, the *span* of S, $\mathrm{span}(S)$, is the set of elements whose inclusion does not increase the rank, including the elements in S: $\mathrm{span}(S) = \{e \in \mathcal{N} : \mathrm{rank}(S+e) = \mathrm{rank}(S)\}$. A set S is *closed* if $S = \mathrm{span}(S)$. Let rank_1 and rank_2 denote the rank functions of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. Similarly, let span_1 and span_2 denote the span functions of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively.

The standard LP relaxation for matroid intersection asks for the maximum nonnegative modular function dominated by rank_1 and rank_2:

$$\begin{aligned} &\text{maximize } x(\mathcal{N}) \text{ over } x : \mathcal{N} \to \mathbb{R}_{\geq 0} \\ &\quad \text{s.t. } x(S) \leq \text{rank}_1(S) \text{ and } x(S) \leq \text{rank}_2(S) \text{ for all } S \subseteq \mathcal{N}. \end{aligned} \tag{1}$$

Here we denote the sum $x(S) \stackrel{\text{def}}{=} \sum_{e \in S} x_e$ for $S \subseteq \mathcal{N}$.

From a dual perspective, for all partitions $(S, \bar{S})$ of the ground set (where $\bar{S} = \mathcal{N} \setminus S$), $\text{rank}_1(S) + \text{rank}_2(\bar{S})$ is an upper bound on the size of any matroid intersection $I \in \mathcal{I}_1 \cap \mathcal{I}_2$. (We have $|I \cap S| \leq \text{rank}_1(S)$ and $|I \cap \bar{S}| \leq \text{rank}_2(\bar{S})$). It is also an upper bound on the LP relaxation as can be seen from duality as follows.

Consider the problem of minimizing $\text{rank}_1(S) + \text{rank}_2(\bar{S})$ over all $S \subseteq V$. The standard LP relaxation is the dual LP of the matroid intersection LP (1):

$$\begin{aligned} &\text{minimize } \sum_{S \subseteq \mathcal{N}} y_S \, \text{rank}_1(S) + z_S \, \text{rank}_2(S) \text{ over } y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0} \\ &\quad \text{s.t. } \sum_{S : e \in S} y(S) + z(S) \geq 1 \text{ for all } e \in \mathcal{N}. \end{aligned} \tag{2}$$

A classical theorem by [10] states that $\max_{I \in \mathcal{I}_1 \cap \mathcal{I}_2} |I| = \min_{S \subseteq V} \text{rank}_1(S) + \text{rank}_2(\bar{S})$, hence both LPs (1) and (2) have integral optimum solutions.

Lemma 7 applies a variation of the MWU framework to the dual covering LP (2). The MWU framework incrementally builds a fractional solution (y, z) over $L = O(\log(n)/\epsilon)$ iterations. Initially $(y, z) = (\mathbb{0}, \mathbb{0})$. Each iteration ℓ queries the oracle for a particular set of weights $w^{(\ell)} \in \mathbb{R}^{\mathcal{N}}_{\geq 0}$, returning $(\tilde{y}^{(\ell)}, \tilde{z}^{(\ell)})$ as described in Lemma 7. We increase y by $\tilde{y}^{(\ell)}/L$ and z by $\tilde{z}^{(\ell)}/L$. The key point is how the weights are chosen. For each element e, in the ℓth iteration, we have

$$w^{(\ell)}(e) = \exp(-(\#\text{ iterations } k < \ell \text{ where } (\tilde{y}^{(k)}, \tilde{z}^{(k)}) \text{ covers } e)).$$

The weight of an element e decays exponentially with the number of oracle solutions that cover e.

To implement the oracle for matroid intersection, given a set of weights $w : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$, we sample $O(k \log(n)/\epsilon)$ elements $\mathcal{N}' \subseteq \mathcal{N}$ in proportion to w. We then compute a $(1-\epsilon)$-maximum matroid intersection I and a dual $(1+\epsilon)$-minimum dual integral solution (S', T'), where $S', T', \subseteq \mathcal{N}'$, for the subproblem over $\mathcal{N}'$. Of course S' and T' do not cover any elements outside of $\mathcal{N}'$, and would fail to satisfy Item b of Lemma 7. We enlarge these sets by taking their spans, $S = \text{span}_1(S')$ and $T = \text{span}_2(T')$, which hopefully includes most of the elements from $\mathcal{N} \setminus \mathcal{N}'$ (by weight w). (S, T) (encoded in the LP by their indicator vectors) is the solution returned by our oracle.

Assuming each iteration implements the oracle of Lemma 7, Lemma 7 asserts that the average of the dual solutions gives a $(1+\epsilon)$-approximate dual solution to the matroid intersection problem. We really want a $(1 - O(\epsilon))$-approximate matroid intersection. Recall that each iteration also gives a matroid intersection I within a $(1 - O(\epsilon))$-factor of a dual solution over the subproblem. Therefore the maximum cardinality of I over all iterations is within a $(1 - O(\epsilon))$-factor of the average dual solution. The average dual solution is feasible (up to scaling by $(1 + O(\epsilon))$), certifying that I is a $(1 - O(\epsilon))$-matroid intersection.

This describes the approximation algorithm for maximum cardinality matroid intersection. Pseudocode is given in Figure 1. The analysis requires both an MWU analysis of Lemma 7, and a more matroid-specific analysis to implement the oracle. The former is similar to

1. Let $w(d) = 1$ for all $d \in \mathcal{N}$.
2. For $\ell = 1, \dots, L$, where $L = O(\log(n)/\epsilon)$:
   A. Let $\mathcal{N}' \subseteq \mathcal{N}$ sample $O(k \log(n)/\epsilon)$ elements with repetition in proportion to $w(e)$. // $\tilde{O}(n + k/\epsilon)$
   B. Compute a $(1-\epsilon)$-approximate matroid intersection $I^{(\ell)}$ and a $(1+\epsilon)$-approximate dual solution $(\tilde{S}^{(\ell)}, \tilde{T}^{(\ell)})$. // $\tilde{O}(k^{3/2}Q/\epsilon^2)$
   C. Let $S^{(\ell)} = \text{span}_1(\tilde{S}^{(\ell)})$ and $T^{(\ell)} = \text{span}_2(\tilde{T}^{(\ell)})$. // $\tilde{O}(nQ)$
   D. For all elements $d \in S^{(\ell)} \cup T^{(\ell)}$, set $w(d) = w(d)/e$. // $\tilde{O}(n)$
3. Let $I^{(\ell)}$ maximize $|I^{(\ell)}|$ over all iterations $\ell \in [L]$. Define (y, z) to be the fractional average of the dual solutions,

$$y = \frac{1}{L}\sum_{\ell=1}^{L} 1_{S^{(\ell)}} \text{ and } z = \frac{1}{L}\sum_{\ell=1}^{L} 1_{T^{(\ell)}},$$

   where 1_X denotes the indicator vector of $X \subseteq \mathcal{N}$.
   Return $(I^{(\ell)}, (1+O(\epsilon))y, (1+O(\epsilon))z)$.

Figure 1 A randomized, $(1 \pm \epsilon)$-approximation algorithm for maximum cardinality matroid intersection and the dual LP.

the analysis given in [2]. Implementing the oracle has two components. First we need to show how to extend $(1-\epsilon)$-approximate matroid intersection algorithms to also produce a $(1+\epsilon)$-approximate dual solutions, without increasing the running time. We also need to prove that a $(1+\epsilon)$-approximate dual solution on the sampled subset $\mathcal{N}'$ extends to a solution satisfying the oracle model of Lemma 7.

Consider this latter point regarding sampling and the oracle problem. For the special case of bipartite matching, where dual solutions are vertex covers, the argument in [2] takes a union bound over all $2^{|V|}$ possible subsets of vertices; the logarithm of this bound is then approximately the size of the sample that is needed. For matroid intersection the dual is more abstract. There are naively 2^n dual integral solutions, which is too large. To transfer [2]'s argument to matroid intersection, we need a bound of the order of $n^{O(k)}$ on the number of dual solutions. We obtain this bound by restricting our attention to closed sets, and identifying closed sets with maximal independent subsets.

Weighted matroid intersection. One can approach weighted matroid intersection similarly. At a high level, to implement the oracle of Lemma 7, we reduce the input size in each iteration by random sampling, and build on previous $(1+\epsilon)$-maximum weight matroid intersection algorithms to extract good dual solutions. The weights inject additional technical details to each component. Greater effort is needed to bound the number of dual solutions, and this motivates the notion of "compact" dual solutions (defined in Section 3). Computing a compact dual solution efficiently requires a closer examination of the "approximate weight splitting" certificate of [8]. To this end, we give a new primal-dual proof of correctness that also efficiently constructs a compact dual solution. After addressing these combinatorial components, we recover the high-level theme of using adaptive sampling to reduce a problem over n elements to smaller primal-dual instances of the problem over roughly k elements.

Pseudocode for the approximate maximum weight matroid intersection algorithm is given in Figure 2. Several of the steps require technical elaboration and we defer a detailed discussion to Section 3.

1. Let $w(e) = 1$ for all $e \in \mathcal{N}$.
2. For $\ell = 1, \ldots, L$, where $L = O(\log(n)/\epsilon)$:
   A. Let $\mathcal{N}' \subseteq \mathcal{N}$ sample $O(k \log(n)/\epsilon)$ elements with repetition in proportion. // $\tilde{O}(k/\epsilon)$
   B. Compute a $(1-\epsilon)$-approximate matroid intersection $I^{(\ell)}$ and a ϵ-approximate cost-splitting certificate $(c_{1,\ell}, c_{2,\ell})$. // $\tilde{O}\big(k^{3/2}/\epsilon^2\big)$
   C. Extract from $(I^{(\ell)}, c_{1,\ell}, c_{2,\ell})$ a compact, $(1+\epsilon)$-approximate dual solution $\tilde{y}^{(\ell)}, \tilde{z}^{(\ell)} : 2^{\mathcal{N}'} \to \mathbb{R}_{\geq 0}$ over $\mathcal{N}'$. // $\tilde{O}(kQ)$
   D. Define $y^{(\ell)}, z^{(\ell)} : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ by

$$y^{(\ell)}(\mathrm{span}_1(S')) = \lceil \tilde{y}^{(\ell)}(S') \rceil_{1+\epsilon} \qquad \text{for all } S' \in \mathrm{support}(\tilde{y}^{(\ell)}),$$
$$z^{(\ell)}(\mathrm{span}_2(T')) = \lceil \tilde{z}^{(\ell)}(T') \rceil_{1+\epsilon} \qquad \text{for all } T' \in \mathrm{support}(\tilde{z}^{(\ell)}),$$

   where $\lceil x \rceil_{1+\epsilon} \stackrel{\text{def}}{=} (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$ rounds x up to the nearest power of $1+\epsilon$. // $\tilde{O}(nQ)$
   E. For all elements $e \in \mathcal{N}$, if e is covered by $(y^{(\ell)}, z^{(\ell)})$, set $w(e) = e^{-1} w(e)$. // $O(n \log(k) Q)$
3. Let $I^{(\ell)}$ maximize $c(I^{(\ell)})$ over $\ell \in [L]$. Return $I^{(\ell)}$ and $\frac{1+O(\epsilon)}{L} \sum_{\ell=1}^{L} (y^{(\ell)}, z^{(\ell)})$.

Figure 2 A randomized, $(1 - O(\epsilon))$-approximation algorithm for weighted matroid intersection and the dual LP.

Conclusion. These algorithms are natural extensions of [2]'s algorithm for approximate bipartite matching. Conceptually, they highlight two new perspectives beyond improved running times for approximate matroid intersection. The first is exposing the versatility of the techniques in [2], beyond matchings in graphs. While we focus on matroid intersection abstractly in the independence oracle model, the techniques are simple and high-level enough to be applied to the diverse family of concrete instances of matroid intersection studied elsewhere. We also give an explicit interface to an oracle model for positive LPs that extends beyond matroid intersection. The second is to reiterate the importance of the dual of matroid intersection problems, at least from the perspective of fast approximation algorithms. Improvements for the dual approximation problems were critical to sparsifying and ultimately accelerating approximation algorithms for the primal problem.

The clear open problem is improving the running time for exact unweighted matroid intersection. We were surprised to realize that the improved approximation algorithm did not immediately imply a faster exact running time. We hope this work draws attention to the remaining bottleneck mentioned above.

Organization. We divide the rest of the article into three main parts:

Section 2: The full details of the matroid intersection algorithm and analysis, completing the description in Section 1.2, assuming and interfacing with the oracle framework of Lemma 7.

Section 3: The weighted matroid intersection algorithm and analysis, again interfacing with the oracle framework of Lemma 7.

Section 4: An MWU analysis of the general oracle framework described in Lemma 7.

2 Matroid intersection

The matroid intersection algorithm was described in Section 1.2, along with an overview of the techniques and the analysis. Pseudocode was presented in Figure 1. To briefly review, the overall algorithm follows the MWU framework described in Lemma 7, applied to the dual LP for matroid intersection, (2). The dual LP has a constraint for each element, so the MWU framework maintains a weight for each element reflecting how well the constraint is being met. The oracle problem in Lemma 7 is a relaxation of (2) where we are only required to satisfy most, rather than all, of the dual covering constraints by weight. To solve the oracle problem quickly, we first randomly sample $O(k\log(n)/\epsilon)$ elements in proportion to their weights. Then we compute a $(1-\epsilon)$-approximate matroid intersection, and a $(1+\epsilon)$-approximate dual solution over the sampled elements. We then extend the dual solution to an infeasible dual solution over all the elements, that satisfies the oracle.

The MWU framework has $O(\log(n)/\epsilon)$ iterations. Each iteration has two bottlenecks. The first bottleneck comes from sampling and updating the weights of each element, and takes $O(nQ)$ time. The second bottleneck is approximating the matroid intersection and its dual over the sampled set of elements. If we let $\mathcal{T}_\epsilon(m)$ denote the time of this step, then the algorithm takes $O(n\log(n)/\epsilon + \mathcal{T}_\epsilon(k\log(n)/\epsilon)\log(n)/\epsilon)$ randomized time overall.

To complete the proof of Theorem 3, there are two points to address:

Section 2.1: Given an $(1\pm\epsilon)$-primal dual oracle for matroid intersection running in $\mathcal{T}_\epsilon(m)$ time, we implement the oracle from Lemma 7 in $O(nQ + \mathcal{T}_\epsilon(k\log(n)/\epsilon))$ randomized time. This implies Lemma 1, which formalizes the reduction to approximate primal-dual matroid intersection.

Section 2.2: Recall that the framework requires solutions to both matroid intersection and its dual LP. We extend the $(1-\epsilon)$-maximum matroid intersection algorithm for [4] to give a $(1+\epsilon)$-dual solution without increasing the running time. This gives $\mathcal{T}_\epsilon(m) = \tilde{O}\big(m^{1.5}Q/\epsilon\big)$, hence Lemma 2, and completes the proof of Theorem 3 via Lemma 1.

2.1 Implementing the oracle

In this section, we assume access to a $(1\pm\epsilon)$-primal dual approximation algorithm for matroid intersection in $\mathcal{M}_1$ and $\mathcal{M}_2$, running in $\mathcal{T}_\epsilon(m)$ time for any subset of m elements. We show how to use this algorithm to implement an oracle satisfying Lemma 7 for the dual LP of matroid intersection.

Recall that the algorithm takes as input w, samples $\tilde{O}(k/\epsilon)$ elements $\mathcal{N}' \subseteq \mathcal{N}$ in proportion to w, computes $(1\pm\epsilon)$-primal and dual solutions over $\mathcal{N}'$, and expands out the dual solution by taking their spans in all of $\mathcal{N}$. The key point of the analysis is understanding the random sample $\mathcal{N}'$. We want to ensure that any approximate dual solution (S', T') over $\mathcal{N}'$, when expanded out to (S, T) where $S = \mathrm{span}_1(S')$ and $T = \mathrm{span}_2(T')$, covers most of $\mathcal{N}$ by weight. The approach is based on [2] and will use an upper bound the number of distinct pairs (S, T) with certain nice properties. The first step in this direction is to count the number of closed sets in a given matroid.

▶ **Lemma 8.** *Let $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ be a matroid with n elements and rank k. Then there are at most n^k closed sets in $\mathcal{M}$.*

Proof. For every closed set S, let I_S be a maximum independent subset of S. We have, $I_S \in \mathcal{I}$, $I_S \subseteq S$, and $\mathrm{span}(I_S) = \mathrm{span}(S)$. We claim the mapping from S to I_S is injective. Indeed, if S and T are closed and $I_S = I_T$, then $S = \mathrm{span}(S) = \mathrm{span}(I_S) = \mathrm{span}(I_T) = \mathrm{span}(T) = T$, so $S = T$. Thus the sets I_S are distinct. Meanwhile, every $I \in \mathcal{I}$ has cardinality at most k. So there are at most n^k closed sets. ◀

The following lemma shows that with high probability, any closed pair of sets (S, T) (with S closed in $\mathcal{M}_1$ and T closed in $\mathcal{M}_2$) either covers almost all of $\mathcal{N}$ by weight, or with high probability, does not cover at least one element in the random sample $\mathcal{N}'$.

▶ **Lemma 9.** *Let $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$ be a set of nonnegative weights. Let $\mathcal{N}'$ sample $O(k \log(n)/\epsilon)$ elements from $\mathcal{N}$ with repetition. Then with high probability we have the following: for all $A, B \subseteq \mathcal{N}$ such that A is closed in $\mathcal{M}_1$, B is closed in $\mathcal{M}_2$, $\operatorname{rank}_1(A) = \operatorname{rank}_2(B) \leq O(k)$, and $w(A \cup B) < (1-\epsilon)w(\mathcal{N})$, $\mathcal{N}'$ samples at least one element outside $A \cup B$.*

Proof. Since $\operatorname{rank}_1(A), \operatorname{rank}_2(B) \leq O(k)$, we can assume that $\mathcal{M}_1$ and $\mathcal{M}_2$ each have rank $O(k)$. Then, by Lemma 8, there are at most $n^{O(k)}$ choices of sets $A, B \subseteq \mathcal{N}$ such that A is closed in $\mathcal{M}_1$ and B is closed in $\mathcal{M}_2$.

Now fix such a pair A, B, and suppose $w(A \cup B) \leq (1-\epsilon)w(\mathcal{N})$. The probability that $\mathcal{N}' \subseteq A \cup B$ is

$$\left(\frac{w(A \cup B)}{w(\mathcal{N})}\right)^{|\mathcal{N}'|} \leq e^{-\epsilon|\mathcal{N}'|} \leq n^{-Ck}$$

for an arbitrarily large constant C. The claim now follows by taking the union bound over A, B. ◀

Now we put everything together. The following lemma gives a subroutine satisfying the requirements of the oracle in Lemma 7.

▶ **Lemma 10.** *Let $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$ be a set of nonnegative weights.*
In $O(n \log(n) Q + \mathcal{T}_\epsilon(k \log(n)/\epsilon))$ randomized time, one can compute an independent set I and sets S closed in $\mathcal{M}_1$ and T closed in $\mathcal{M}_2$ such that, with high probability:

- **i** $\operatorname{rank}(S) + \operatorname{rank}(T) \leq (1+\epsilon)|I|$.
- **ii** $w(S \cup T) \geq (1-\epsilon)w(\mathcal{N})$.

Proof. For ease of convention, we prove the claim with ϵ replaced by $O(\epsilon)$; the constant can then be removed by decreasing ϵ by a constant factor. (We adopt the same convention in subsequent proofs.)

Let $\mathcal{N}'$ sample $O(k \log(n)/\epsilon)$ elements with repetition in proportion to w. In $\mathcal{T}_\epsilon(k \log(n)/\epsilon)$ time, we compute a $(1-\epsilon)$-maximum matroid intersection I and a $(1+\epsilon)$-dual solution (A', B') in the restriction to $\mathcal{N}'$. We compute $A = \operatorname{span}_1(A')$ and $B = \operatorname{span}_2(B')$ in $O(nQ)$ time. We claim that I, A, and B satisfy the lemma.

First, we have $\operatorname{rank}_1(A) + \operatorname{rank}_2(B) = \operatorname{rank}_1(A') + \operatorname{rank}_2(B') \leq (1 + O(\epsilon))|I|$. Second, by Lemma 9, with high probability; since $\mathcal{N}' \subseteq A' \cup B' \subseteq A \cup B$, A is closed in $\mathcal{M}_1$, and B is closed in $\mathcal{M}_2$; we have $w(A \cup B) \geq (1-\epsilon)w(\mathcal{N})$, as desired. ◀

2.2 Fast primal-dual approximations for matroid intersection

It remains to give a fast primal-dual approximation for matroid intersection. The algorithm in [4] gives a $(1-\epsilon)$-maximum matroid intersection but not a $(1+\epsilon)$-dual solution. That said, one might expect a $(1+\epsilon)$-dual solution to be implicit in any given $(1-\epsilon)$-maximum matroid intersection algorithm in order to certify that the solution I is $(1-\epsilon)$-maximum. Such is the case here and we show how to extract a $(1+\epsilon)$-dual solution efficiently.

The $(1-\epsilon)$-maximum matroid intersection algorithm in [4], like other $(1-\epsilon)$-approximation algorithms, outputs an independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ for which the length of the minimum "augmenting path" (defined in a moment) is at least $2/\epsilon$. This length implies that I is a $(1-\epsilon)$-maximum matroid intersection (just as in bipartite matching). We use this bound on the minimum length of an augmenting bound to extract a $(1+\epsilon)$-dual solution.

To go into further detail we must first introduce augmenting paths in the context of matroid intersection. The notion generalizes augmenting paths in bipartite matching. In the context of matroid intersection, for a fixed independent set I, an augmenting path is a sequence $P = (e_0, e_1, \dots, e_h)$ of elements alternating between $\mathcal{N} \setminus I$ and I. Additionally, an augmenting path must start and end with elements in $\mathcal{N} \setminus I$, and its symmetric difference with I, $I \triangle P$, must be independent in both matroids.

Augmenting paths are paths in an auxiliary directed bipartite graph between $\mathcal{N} \setminus I$ and I called the *exchange graph*. We have a directed edge (e, d) from $\mathcal{N} \setminus I$ to I iff $I - d + e \in \mathcal{I}_2$. We have a directed edge (d, e) from I to $\mathcal{N} \setminus I$ iff $I - d + e \in \mathcal{I}_1$.

Let $F_1 = \mathcal{N} \setminus \text{span}_1(I)$ be the set of free/uncovered elements in $\mathcal{M}_1$, and $F_2 = \mathcal{N} \setminus \text{span}_2(I)$ be the free elements in $\mathcal{M}_2$. All augmenting paths are between F_1 and F_2, but unlike bipartite matching, not all paths from F_1 to F_2 in the exchange graph are augmenting paths. However, all shortest paths from F_1 to F_2 are always augmenting paths. Thus many matroid intersection algorithms augment along shortest (F_1, F_2)-paths in the exchange graph.

If the minimum length of any augmenting path for I is at least $2/\epsilon$, then the typical argument that I is $(1-\epsilon)$-maximum is as follows. Given an optimal solution I^*, by contracting $I \cap I^*$, we may assume I^* and I are disjoint. $I \triangle I^*$ decomposes into even-length cycles and augmenting paths in the exchange graph. I and I^* have the same number of elements in each even-length cycle. For an augmenting path P, if P has length at least $2/\epsilon$, then $|I^* \cap P| \leq (1+\epsilon)|I \cap P|$. It follows that $|I^*| \leq (1+\epsilon)|I|$.

This argument does not imply an algorithm for a $(1+\epsilon)$-dual solution. (We do not have access to I^*.) Still, we can use the length bound to obtain a $(1+\epsilon)$-dual solution, giving another proof that I is $(1-\epsilon)$-maximum. Formally we prove the following.

▶ **Lemma 11.** *Let $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ be an independent point in the intersection. Suppose the minimum length of any augmenting path is at least $2/\epsilon$. Then in $O(n \log(k) Q/\epsilon)$ time, one can compute sets $S, T \subseteq \mathcal{N}$ such that $S \cup T = \mathcal{N}$ and $\text{rank}_1(S) + \text{rank}_2(T) \leq (1+\epsilon)|I|$.*

Proof. The desired sets S and T will be induced by the distances layers from F_1 in the exchange graph. For each index $i \in \mathbb{Z}_{\geq 0}$, let L_i be the set of elements at distance i from F_1. For example, $L_0 = F_1$, $L_i \subseteq \mathcal{N} \setminus I$ for even i, and $L_i \subseteq I$ for odd i.

To extract a good dual solution from these layers, we first need to construct them quickly. [7, 23] provide the following.

▶ **Fact 12.** *For $h = O(1/\epsilon)$, the first h layers $L_0, L_1, \dots, L_{h-1}$ of the exchange graph can be computed in $O(n \log(k) Q/\epsilon)$ time.*

To refer to elements in I and $\mathcal{N}$ by distance layer, we introduce the following notation. For all i, let $\mathcal{N}_i = \mathcal{N} \cap L_i$, and for odd i, let $\mathcal{I}_i = \mathcal{I} \cap L_i$. These layers partition the ground set, with the even layers partitioning $\mathcal{N} \setminus I$ and the odd layers partitioning I.

Suppose we construct the layers $(L_0 = F_1), \dots, L_h$ up to distance h from F_1 for $h \geq 2/\epsilon$. Since $h \geq 2/\epsilon$, there is an odd index i with $1 \leq i \leq h$ such that $|I_i| \leq \epsilon |I|$. Let $S = \mathcal{N} \setminus \left(\bigcup_{j<i} \mathcal{N}_i \right)$ be all elements in layer i and beyond, and $T = \bigcup_{j \leq i} \mathcal{N}_j$ be all the elements up to layer i. (S, T) is a discrete and feasible solution to the dual because $S \cup T = \mathcal{N}$. To prove that it is a $(1+\epsilon)$-dual solution, it suffices to show that $\text{rank}_1(S) + \text{rank}_2(T) \leq (1+\epsilon)|I|$.

Let $I^+ = I \setminus \left(\bigcup_{j<i} I_j \right) = I \cap S$. We claim that $S \subseteq \text{span}_1(I^+)$. Clearly $S \cap I = I^+ \subseteq \text{span}_1(I^+)$. Now consider an element $e \in S \setminus I$. We have $e \in \text{span}_1(I)$ because $e \notin \mathcal{N}_0$. If $e \notin \text{span}_1(I^+)$, then $I - d + e \in \mathcal{I}_1$ for some $d \in I \setminus I^+$. Then (d, e) is an exchange in the exchange graph. We have $d \in I_j$ for some $j \leq i - 2$, hence $d \in \mathcal{N}_\ell$ for some $\ell < i$. But then $e \notin S$, a contradiction.

Now let $I^- = \bigcup_{j \le i} I_j = I \cap T$. We claim that $T \subseteq \text{span}_2(I^-)$. We have $T \cap I \subseteq \text{span}_2(I^-)$. Consider any element $e \in T \setminus I$. We have $e \in \text{span}_2(I)$ since $e \in L_j$ for some $j < i \le h$, and there are no (F_1, F_2)-paths of length $< h$. If $e \notin \text{span}_2(I^-)$, then $I - d + e \in \mathcal{I}_2$ for some $d \in I \setminus I^-$. That is, (e, d) is an edge in the exchange graph. $e \in T$ implies that the distance from F_1 to e is at most $i - 1$, and the distance from F_1 to d is at most i. But $d \notin I^-$ implies that the distance from F_1 to d is at least $i + 2$, a contradiction.

Thus $S \subseteq \text{span}_1(I^+)$ and $T \subseteq \text{span}_2(I^-)$ where I^+ and I^- are subsets of I overlapping on I^-. Since $I^+ \cap I^- = I_i$, we have

$$\begin{aligned}\text{rank}(S) + \text{rank}(T) &\le \text{rank}(\text{span}_1(I^+)) + \text{rank}(\text{span}_2(I^-)) \\ &= |I^+| + |I^-| = |I| + |I_i| \le (1+\epsilon)|I|,\end{aligned}$$

as desired.

To recap, given an independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ for which the length of the minimum augmenting path is at least $2/\epsilon$, we build out the first $O(1/\epsilon)$ layers of the exchange graph in $O(n \log(k) Q/\epsilon)$ time. We identify an index i such that $|I_i| \le \epsilon |I|$. We assemble the set $S = \mathcal{N} \setminus (L_0 \cup L_1 \cup \cdots \cup L_{i-1})$ of all elements in the ith layer and beyond, and the set $T = L_0 \cup L_1 \cdots \cup L_i$ of all elements up to the ith layer. This takes $O(n)$ time given the first $O(1/\epsilon)$ layers. (S, T) is the desired $(1+\epsilon)$-dual solution, completing the proof of Lemma 11. ◀

This concludes our discussion on maximum cardinality matroid intersection.

3 Weighted matroid intersection

We now consider the weighted matroid intersection problem. Similar to the unweighted setting, the high-level approach combines adaptive sparsification with a fast $(1 \pm \epsilon)$-primal-dual approximation algorithm to improve the running time dependence on the input n. These components are more technically challenging in the weighted setting.

Recall that $\mathcal{M}_1 = (\mathcal{N}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{N}, \mathcal{I}_2)$ are two matroids over the same ground set $\mathcal{N}$, $c : \mathcal{N} \to \mathbb{R}_{\ge 1}$ is a set of input costs over $\mathcal{N}$, and $\epsilon \in (0, 1)$ is an input parameter.[2] The goal is to compute $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ with cost $c(I)$ at least a $(1 - \epsilon)$-fraction of the maximum cost of any such I. We assume without loss of generality that $c(e) \in [1, \text{poly}(n)]$ for all e.[3]

We start by introducing the LP relaxation of weighted matroid intersection:

$$\begin{aligned}&\text{maximize } \langle c, x \rangle \text{ over } x \in \mathbb{R}^E_{\ge 0} \\ &\quad \text{s.t. } x(S) \le \text{rank}_1(S) \text{ and } x(S) \le \text{rank}_2(S) \text{ for all } S \subseteq \mathcal{N}.\end{aligned} \tag{3}$$

The dual LP is as follows.

$$\begin{aligned}&\text{minimize } \sum_S \text{rank}_1(S) y_S + \text{rank}_2(S) z_S \text{ over } y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\ge 0} \\ &\quad \text{s.t. } \sum_{S : e \in S} y_S + z_S \ge c_e \text{ for all } e \in \mathcal{N}.\end{aligned} \tag{4}$$

For $e \in \mathcal{N}$, we say that (y, z) covers e if it meets the covering constraint for e in the dual LP (4).

---

[2] c is normally called "weights" in the weighted matroid intersection problem, but we will refer to them as "costs" to help distinguish them from the auxiliary weights generated by the framework.

[3] We assume each singleton set $\{e\}$ is independent for all e by removing all violating e in a preprocessing step. Then $\text{OPT} \ge \max_e c(e)$, and we can drop any element e with $c(e) \le (\epsilon/k) \max_f c(f)$ without decreasing the optimum value by more than an ϵ-fraction. Rescaling, all weights lie in the range $[1, k/\epsilon]$.

Recall that the overall framework maintains auxiliary weights $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$. For a fixed set of weights $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$, we say that y, z covers a $(1-\epsilon)$-fraction of elements by w if the total weight of elements covered by e is at least $(1-\epsilon)$-fraction of the total weight of all elements.

To apply Lemma 7 to the LPs (3) and (4), we need to fulfill an oracle problem defined as follows. The oracle takes as input a set of auxiliary weights $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$. With high probability, the oracle must return a dual solution y, z that covers a $(1-\epsilon)$-fraction of elements by weight.

Our implementation of the oracle is broken down into two components: (a) a randomized sparsification step applying a $(1 \pm \epsilon)$-primal-dual approximation algorithm to a randomly sampled subset of $\tilde{O}(k \log(n)/\epsilon)$ elements; and (b) the $(1 \pm \epsilon)$-primal-dual approximation oracle. Formally stating the interface of these two parts requires the notion of "compact" dual solutions and compact primal-dual algorithms. We define these now and give further background in appropriate subsections later.

We say that (y, z) is *compact* if the supports of y and z are of the form

$$\text{support}(y) \subseteq \{\text{span}_1(e_1), \text{span}_1(e_1, e_2), \ldots, \text{span}_1(e_1, \ldots, e_k)\}$$
$$\text{support}(z) \subseteq \{\text{span}_2(f_1), \text{span}_2(f_1, f_2), \ldots, \text{span}_2(f_1, \ldots, f_k)\}$$

for two sequences of k elements $e_1, \ldots, e_k \in \mathcal{N}$ and $f_1, \ldots, f_k \in \mathcal{N}$. We define a *compact* $(1 \pm \epsilon)$*-primal-dual weighted matroid intersection* algorithm that takes as input $\mathcal{M}_1 = (\mathcal{N}, \mathcal{I}_1)$, $\mathcal{M}_2 = (\mathcal{N}, \mathcal{I}_2)$, and $c : \mathcal{N} \to \mathbb{R}_{\geq 0}$, and returns $(1-\epsilon)$-maximum weight independent set and a compact $(1+\epsilon)$-minimum solution to the dual LP (4).

Now we can state the guarantees of part (a).

▶ **Lemma 13.** *Let $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$, and suppose there is a compact $(1 \pm \epsilon)$-primal-dual weighted matroid intersection algorithm running in $\mathcal{T}_\epsilon(m)$ on subsets of $\mathcal{N}$ of size m. Then in $O(\mathcal{T}_\epsilon(k \log(n)/\epsilon))$ time, one can compute an independent set I and a compact dual solution (y, z) such that (y, z) covers a $(1-\epsilon)$-fraction of $\mathcal{N}$ by w.*

Now we formally state the guarantees for part (b); namely, a compact $(1 \pm \epsilon)$-primal-dual approximation algorithm for weighted matroid intersection. The following algorithm extends the $(1-\epsilon)$-approximation for weighted matroid intersection to also give a compact dual solution.

▶ **Lemma 14.** *There is a compact $(1 \pm \epsilon)$-primal dual algorithm running in $\mathcal{T}_\epsilon(m) = \tilde{O}\left(m\sqrt{k}Q/\epsilon^2\right)$ time.*

We prove Lemma 13 in Section 3.1 and Lemma 14 in Section 3.2. Combining Lemmas 13 and 14 gives a subroutine implementing the oracle of Lemma 7 running in $\tilde{O}\left(nQ/\epsilon + k^{1.5}/\epsilon^3\right)$ time. Because the dual solution (y, z) returned by the oracle is compact, we can identify all the elements covered by (y, z) in $O(n \log(k)Q)$ time.[4] Lemma 7 repeats these two steps for $O(\log(n)/\epsilon)$ iterations, giving a total running time of $O\left(n \log(n)Q/\epsilon + k^{1.5} \log^{O(1)}(n)Q/\epsilon^4\right)$.

[4] For example, suppose the support of y is generated by the elements $e_1, \ldots, e_k$. Given an element d, one can binary search for the first index i such that $d \in \text{span}_1(e_1, \ldots, e_i)$. Then d is covered by $\text{span}_1(e_1, \ldots, e_j)$ for all $j \geq i$. With simple preprocessing, one then extracts the total amount that y covers d in $O(1)$ time. Similarly the amount that z covers d can be computed in $O(\log(k)Q)$ time with simple preprocessing.

3.1 Sparse reduction to compact $(1 \pm \epsilon)$-primal-dual approximations: proof of Lemma 13

In this section we prove Lemma 13. Let $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$. We need to compute an independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and a compact $y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ such that:

(i) $(1 + O(\epsilon))c(I) \geq \sum_S \operatorname{rank}_1(S) y_S + \operatorname{rank}_2(S) z_S$.

(ii) (y, z) covers a $(1 - \epsilon)$-fraction of $\mathcal{N}$ by w.

The algorithm is as follows.

1. Let $\mathcal{N}' \subseteq \mathcal{N}$ sample $O(k \log(n)/\epsilon)$ elements, with repetition, in proportion to w.
2. Run the $(1 \pm \epsilon)$-primal dual approximation algorithm on $\mathcal{N}'$, producing $I \subseteq \mathcal{N}'$ with $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and a compact solution $y', z' : 2^{\mathcal{N}'} \to \mathbb{R}_{\geq 0}$. Without loss of generality, the nonzero coordinates of y' and z' are in the range $[1/\operatorname{poly}(n), \operatorname{poly}(n)]$.
3. Define $y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ by setting $y(\operatorname{span}_1(S)) = \lceil y'(S) \rceil_{1+\epsilon}$ for $S \in \operatorname{support}(y')$ and similarly set $z(\operatorname{span}_2(T)) = \lceil z'(T) \rceil_{1+\epsilon}$ for $T \in \operatorname{support}(z')$, where $\lceil x \rceil_{1+\epsilon} \stackrel{\text{def}}{=} (1+\epsilon)^{\lceil \log_{1+\epsilon} x \rceil}$ rounds x up to the nearest power of $1 + \epsilon$.
4. Return I and (y, z).

Consider the range of (y, z) output by the algorithm. We first observe that (y, z) is compact. Indeed, since (y', z') was compact, there is a sequence of elements $e_1, \ldots, e_k$ such that all sets in the support of y' have the form $S'_i = \operatorname{span}_1(e_1, \ldots, e_i) \cap \mathcal{N}'$ for some prefix $e_1, \ldots, e_i$. Then the sets in the support of y have the form $S_i = \operatorname{span}_1(S'_i) = \operatorname{span}_1(e_1, \ldots, e_i)$. Thus the support of y is generated by prefixes of $e_1, \ldots, e_k$. Symmetrically the same holds for z for a different sequence of k elements depending on z'. Thus (y, z) is compact.

We also observe that each nonzero value of y or z is one of the $O(\log(n)/\epsilon)$ powers of $(1 + \epsilon)$ in the range $[1/\operatorname{poly}(n), \operatorname{poly}(n)]$.

For y, there are at most n^k ways to choose the sequence $e_1, \ldots, e_k$ that determines its support, and $(C \log(n)/\epsilon)^k$ ways to assign their values for some constant $C > 0$. Likewise for z. Altogether, there are at most $n^{O(k)}$ choices of (y, z) in the range of the algorithm.

Call a solution (y, z) in the range *good* if it covers an $(1 - \epsilon)$-fraction of $\mathcal{N}$ by w, and *bad* otherwise. We want to argue that with high probability, the solution (y, z) returned by the algorithm is good. We know that the output (y, z) covers all the elements in $\mathcal{N}'$. Thus it suffices to show that for all bad (y, z) in the range, $\mathcal{N}'$ samples at least one element that is not covered by (y, z).

To this end, fix a bad (y, z) in the range. A random element sampled from $\mathcal{N}$ in proportion to w is covered by (y, z) with probability at most $1 - \epsilon$. The probability that all of $\mathcal{N}'$ is covered by (y, z) is bounded above by $(1 - \epsilon)^{|\mathcal{N}'|} = n^{-O(k)}$. Taking the union bound over all bad (y, z) in the range, we conclude with high probability, $\mathcal{N}'$ samples at least one uncovered element for every bad (y, z). In this event, since the output (y, z) covers $\mathcal{N}'$, (y, z) must be good.

This completes the proof of Lemma 13.

3.2 Compact $(1 \pm \epsilon)$-primal-dual weighted matroid intersection: proof of Lemma 14

In this section we prove Lemma 14. We are given two matroids $\mathcal{M}_1 = (\mathcal{N}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{N}, \mathcal{I}_2)$ over a common set $\mathcal{N}$ of size n, $c : \mathcal{N} \to \mathbb{R}_{\geq 1}$, and $\epsilon \in (0, 1)$. We let k be the rank of the intersection $\mathcal{I}_1 \cap \mathcal{I}_2$. We will compute $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and a *compact* dual solution $y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ such that $(1 + O(\epsilon))c(I) \geq \sum_S \operatorname{rank}_1(S) y(S) + \operatorname{rank}_2(S) z(S)$.

As mentioned previously, a fast approximation algorithm for weighted matroid intersection is given by [8]. The stated running time is $\tilde{O}\left(nkQ/\epsilon^2\right)$ and breaks down as follows. There are $\tilde{O}(1/\epsilon)$ outer iterations. Each iteration invokes an approximation algorithm for unweighted matroid intersection that returns an independent set I with minimum augmenting path length $O(1/\epsilon)$. [8] truncate Cunningham's algorithm to execute an inner iteration in $\tilde{O}(nkQ/\epsilon)$ time. The subroutine using Cunningham's algorithm can be replaced by the $\tilde{O}\left(n\sqrt{k}Q/\epsilon\right)$-time algorithm of [4].[5] This gives a $(1-\epsilon)$-approximation algorithm for weighted matroid intersection running in $\tilde{O}\left(n\sqrt{k}Q/\epsilon^2\right)$ time.

The real challenge is to also compute a compact $(1+\epsilon)$-approximate dual solution within the same running time. This requires additional background on the framework of [8].

Cost-splitting. The fast approximation algorithm in [8] is based on the cost-splitting approach to matroid intersection described in [14]. A *cost-splitting* of c is a decomposition $c_1 + c_2$ where $c_1, c_2 : \mathcal{N} \to \mathbb{R}_{\geq 0}$.[6] A cost-splitting can be used to certify a maximum cost independent set as follows.

▶ **Fact 15** ([14]). *Suppose $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and $c = c_1 + c_2$ is a cost-splitting such that I is a c_1-maximum independent set in $\mathcal{M}_1$ and c_2-maximum independent set in $\mathcal{M}_2$. Then I is a maximum cost independent set in the intersection of $\mathcal{M}_1$ and $\mathcal{M}_2$.*

The proof is immediate: given (I, c_1, c_2) as in fact 15, and letting I^* denote an optimum solution, we have $c_1(I) \geq c_1(I^*)$ and $c_2(I) \geq c_2(I^*)$. Since $c = c_1 + c_2$, $c(I) \geq c(I^*)$. Note that this proof does not involve the dual LP.

An approximate version of fact 15 is given in [8] to certify $(1-\epsilon)$-approximate solutions. This proof also does not construct a dual solution, let alone compact one.

Before describing how to extract the desired dual solution from [8], we give additional background on the LPs Equations (3) and (4). This background is not necessary for our analysis, but it gives some intuition for the eventual claim.

[10] proved that LP (3) is totally dual integral. Moreover, any optimal solution (y, z) can be uncrossed and merged so that the supports of y and z each form a chain. Expanding on the latter point, suppose the support of y is a chain of the form $S_1 \subsetneq S_2 \subsetneq \cdots \subsetneq S_\ell$. We can replace each S_i with $\mathrm{span}(S_i)$ without increasing the objective or decreasing the coverage on any element. Thus, replacing each S_i with $\mathrm{span}(S_i)$, we can assume each S_i is closed. We have $0 < \mathrm{rank}_1(S_1) < \mathrm{rank}_1(S_2) < \cdots < \mathrm{rank}_1(S_\ell) \leq \mathrm{rank}_1(\mathcal{N})$. Let $I_0 = \emptyset$, and for $i = 1, 2, \ldots, \ell$ in sequence, let I_i extend I_{i-1} to a maximum cardinality independent set of S_i. Then $S_i = \mathrm{span}(I_i)$ for each i. Let $e_1, \ldots, e_h$ enumerate the elements of I_1, then $I_2 \setminus I_1$, and so forth, so that each I_i is a prefix of the form $I_i = \{e_1, \ldots, e_{j_i}\}$ for some index $j_i \in [h]$. Since $I_\ell = \{e_1, \ldots, e_h\} \in \mathcal{I}_1$, $h \leq \mathrm{rank}_1(\mathcal{N})$.

This exercise shows that if the support of y is a chain, then it is induced by prefixes of a sequence of $h \leq \mathrm{rank}_1(\mathcal{N})$ elements $e_1, \ldots, e_h$. Symmetrically if the support of z is a chain, then it is also generated by a sequence of $\mathrm{rank}_2(\mathcal{N})$ elements. Consequently there are at most $n^{\mathrm{rank}_1(\mathcal{N}) + \mathrm{rank}_2(\mathcal{N})}$ ways for y and z to have chain supports.

[5] A minor technical point to address is that the unweighted matroid intersection is not over $\mathcal{M}_1$ and $\mathcal{M}_2$, but auxiliary "weight-induced matroids" $\mathcal{M}_{1,c_1}$ and $\mathcal{M}_{2,c_2}$ induced by weights c_1 and c_2. Fortunately it is easy to identify edges of the exchange graph of $\mathcal{M}_{1,c_1}$ and $\mathcal{M}_{2,c_2}$ via independence oracles to the input matroids $\mathcal{M}_1$ and $\mathcal{M}_2$. The algorithm in [4] can be adapted to $\mathcal{M}_{1,c_1}$ and $\mathcal{M}_{2,c_2}$ just as Cunningham's algorithm was adapted in [8].

[6] [14] calls this a "weight-splitting". We refer to this as a "cost-splitting" because we are referring to c as costs.

Thus a compact solution (y, z) is similar to a solution (y, z) where the support is a chain. However, compact solutions restrict the generating sequences of elements to have length at most k, and $k \leq \min\{\text{rank}_1(\mathcal{N}), \text{rank}_2(\mathcal{N})\}$.

As alluded to earlier, [8] computes a $(1-\epsilon)$-approximation I along with vectors c_1, c_2 that approximate the (exact) cost-splitting describe in [14]. We formalize the approximation conditions as follows: For fixed I and $\epsilon \in (0, 1)$, an *ϵ-approximate cost-splitting certificate* is a pair of weight vectors $c_1, c_2 : \mathcal{N} \to \mathbb{R}_{\geq 0}$ with the following properties:

(a) $(1+\epsilon)(c_1 + c_2) \geq c$ elementwise.
(b) $c_1(I) + c_2(I) \leq (1+\epsilon)c(I)$.
(c) I is a c_1-maximum independent set in $\mathcal{M}_1$ restricted to $\text{span}_1(I)$.
(d) I is a c_2-maximum independent set in $\mathcal{M}_2$ restricted to $\text{span}_2(I)$.
(e) $c_1(e) \leq \epsilon$ for all $e \in \mathcal{N} \setminus \text{span}_1(I)$.
(f) $c_2(e) \leq \epsilon$ for all $e \in \mathcal{N} \setminus \text{span}_2(I)$.

[8] produces an independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and an ϵ-approximate cost-splitting certificate c_1, c_2. We show how to extract a compact $(1+\epsilon)$-approximate dual solution from I, c_1, and c_2.

▶ **Lemma 16.** *Let $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and let $c_1, c_2 : \mathcal{N} \to \mathbb{R}_{\geq 0}$ be an ϵ-approximate cost-splitting certificate. Then in $\tilde{O}(nQ)$ time, one can compute $y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ such that*

(i) *(y, z) are feasible for the dual LP (4).*
(ii) *(y, z) are compact.*
(iii) $\sum_S \text{rank}_1(S) y_S + \text{rank}_2(S) y_S \leq (1 + O(\epsilon))c(I)$

The last point says that I and (y, z) mutually certify each other to be $(1 \pm O(\epsilon))$-approximations to their respective problem. Taking ϵ down to 0 gives an exact compact dual optimum solution certifying I to be exactly optimum.

Proof of Lemma 16. For $t \in \mathbb{R}_{\geq 0}$, let

$$Y_t = \text{span}_1(\{e \in I : c_1(e) \geq t\}) \text{ and } Z_t = \text{span}_2(\{e \in I : c_2(e) \geq t\}).$$

Let $y, z : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be defined by

$$y = (1 + O(\epsilon)) \int_0^\infty 1_{Y_t}\, dt \text{ and } z = (1 + O(\epsilon)) \int_0^\infty 1_{Z_t}\, dt,$$

where 1_S denotes the indicator vector for $S \subseteq \mathcal{N}$ in $\mathbb{R}^{2^{\mathcal{N}}}$. Observe that y is supported by a chain of elements listing I in decreasing order of c_1, and z is supported by a chain of elements listing I in decreasing order of c_2. It remains to show that y, z is feasible for the dual LP (4), and has objective value within a $(1 + O(\epsilon))$-factor of $c(I)$.

To show that (y, z) is feasible, fix $e \in \mathcal{N}$. If $c_1(e) \geq \epsilon$, then $e \in \text{span}_1(I)$. Since I is c_1-maximum in $\mathcal{M}_1$, $e \in \text{span}(Y_t)$ for all $t \leq c_1(e)$. Therefore,

$$\sum_{S : e \in S} y_S \geq (1 + O(\epsilon)) \int_0^{c_1(e)} 1\, dt = (1 + O(\epsilon))c_1(e).$$

Symmetrically, if $c_2(e) \geq \epsilon$, then $\sum_{S : e \in S} z_S \geq (1 + O(\epsilon))c_2(e)$.

Now we have three cases. If $c_1(e) \geq \epsilon$ and $c_2(e) \geq \epsilon$, then

$$\sum_{S : e \in S} y_S + z_S \geq (1 + O(\epsilon))c_1(e) + (1 + O(\epsilon))c_2(e) \geq c(e).$$

If $c_1(e) \le \epsilon$, then $c_1(e) \le \epsilon c(e)$, and

$$\sum_{S:e\in S} y_S + z_S \ge \sum_{S:e\in S} z_S \ge (1+O(\epsilon))c_2(e) \ge (1+O(\epsilon))((1-\epsilon)c(e) - c_1(e))$$
$$\ge (1+O(\epsilon))(1-2\epsilon)c(e) \ge c(e).$$

Symmetrically, if $c_2(e) \le \epsilon$, then $\sum_{S:e\in S} y_S + z_S \ge (1-2\epsilon)c(e)$. Thus (y,z) forms a feasible dual solution if $c_1(e) \le \epsilon$, $c_2(e) \le \epsilon$, or neither.

Now consider the objective value. The contribution from y is

$$\sum_S \operatorname{rank}_1(S) y_S = (1+O(\epsilon)) \int_0^\infty \operatorname{rank}(Y_t)\, dt$$
$$= (1+O(\epsilon)) \int_0^\infty |Y_t \cap I|\, dt = (1+O(\epsilon))c_1(I).$$

Symmetrically, z contributes $\sum_S \operatorname{rank}_2(S) z_S = (1+O(\epsilon))c_2(I)$. Together, we have

$$\sum_S \operatorname{rank}_1(S) y_S + \operatorname{rank}_2(S) z_S = (1+O(\epsilon))(c_1(I)+c_2(I)) \le (1+O(\epsilon))I,$$

as desired. ◀

Now we complete the proof of Lemma 13. Using the algorithm of [8] with the $(1-\epsilon)$-approximated unweighted matroid intersection algorithm of [4] as a subroutine, we compute an $(1-\epsilon)$-approximate maximum cost independent set I and a ϵ-approximate cost-splitting certificate c_1, c_2, in $\tilde{O}\left(n\sqrt{k}Q/\epsilon^2\right)$ time. By Lemma 16, we extract a feasible compact $(1+\epsilon)$-dual solution (y,z) in $\tilde{O}(nQ/\epsilon)$ time. We return I and (y,z). The overall running time is $\tilde{O}\left(nQ/\epsilon + k^{3/2}/\epsilon^3\right)$. This completes the proof of Lemma 13 and of the faster approximate matroid intersection algorithm.

4 MWU analysis

In this section we prove Lemma 7, which claims that given a covering LP, solving a certain relaxed coverage problem for $\tilde{O}(1/\epsilon)$ iterations leads to a $(1-\epsilon)$-approximation overall.

Like other proofs, we fix ϵ, and describe and analyze an algorithm that returns a point $x \in \mathcal{P}$ with objective value $\langle b, x\rangle \le (1+O(\epsilon))\,\mathrm{OPT}$ and coverage $Ax \ge (1-O(\epsilon))\mathbb{1}$. The claimed $(1\pm\epsilon)$ bounds then follow by decreasing ϵ by a constant factor.

There are multiple perspectives on the MWU framework presenting essentially the same proof in different styles. Our presentation follows the conventions of the analysis from [24].

The algorithm builds a solution x incrementally over $L = O(\log(m)/\epsilon)$ iterations. We let $x^{(\ell)}$ denote the value of x after ℓ iterations. It starts with $x^{(0)} = \mathbb{0}$. Each iteration ℓ invoke the oracle for a set of weights $w^{(\ell)}$ and produces a point $y^{(\ell)}$, and updates $x^{(\ell)} = x^{(\ell-1)} + y^{(\ell)}/L$. The output, $x^{(L)} = \sum_\ell y^{(\ell)}/L$, is the average of the points returned by the oracle. The exact steps in the ℓth iteration are as follows:

1. For each $i \in [m]$, compute $w_i^{(\ell)} = e^{-\operatorname{load}^{(\ell-1)}(i)}$, where
$$\operatorname{load}^{(\ell-1)}(i) = |\{k \in \{1,\dots,\ell-1\} : (Ay^{(k)})_i \ge 1\}|$$
is the number of previous iterations k where $y^{(k)}$ covers w_i.
2. Invoke the oracle with respect to $w^{(\ell)}$, returning $y^{(\ell)}$.
3. Set $x^{(\ell)} = x^{(\ell-1)} + y^{(\ell)}/L$.

Each call to the randomized oracle succeeds with high probability, and there are $L = O(\log(m)/\epsilon)$ iterations. By the union bound, all oracle calls succeed with high probability. For the remainder of the analysis we assume this is the case.

We want to show that $x^{(L)} = \sum_{\ell=1}^{L} y^{(\ell)}/L$ is a $(1 + O(\epsilon))$ approximation. It is easy to see that the objective value is good: we have

$$\langle b, x^{(L)} \rangle = \frac{1}{L} \sum_{\ell=1}^{L} \langle b, y^{(\ell)} \rangle \geq (1+\epsilon)\,\mathrm{OPT}$$

because the oracle guarantees $\langle b, y^{(\ell)} \rangle \geq (1+\epsilon)\,\mathrm{OPT}$ for all ℓ. It remains to show that $Ax^{(L)} \geq (1 - O(\epsilon))\mathbb{1}$. The key to the analysis is understanding why the weights are selected as they are.

For each iteration ℓ, let $M_\ell = \{i \in [m] : (Ay^{(\ell)})_i \geq 1\}$ be the set of coordinates covered in the ℓth iteration. The oracle guarantees that for each iteration $\ell \in [L]$,

$$\sum_{i \in M_\ell} w_i^{(\ell-1)} \geq (1-\epsilon)\langle w^{(\ell-1)}, \mathbb{1} \rangle.$$

We claim that

$$\sum_{i \in M_\ell} w_i^{(\ell)} \geq (1 - O(\epsilon))\langle w^{(\ell)}, \mathbb{1} \rangle \tag{5}$$

for all $\ell \in [L]$. To this end, we have

$$\begin{aligned}\sum_{i \notin M_\ell} w_i^{(\ell)} &\overset{(a)}{\leq} \sum_{i \notin M_\ell} w_i^{(\ell-1)} \leq \epsilon \langle w^{(\ell-1)}, \mathbb{1} \rangle \leq \frac{\epsilon}{(1-\epsilon)} \sum_{i \in M_\ell} w_i^{(\ell-1)} \\ &\overset{(b)}{=} \frac{\epsilon e}{(1-\epsilon)} \sum_{i \in M_\ell} w_i^{(\ell)} \leq \frac{\epsilon e}{(1-\epsilon)} \langle w^{(\ell)}, \mathbb{1} \rangle,\end{aligned}$$

as desired. Here, (a) is because $w^{(\ell)}$ is decreasing in ℓ. (b) is because for each $i \in M_\ell$, the weight decreases by e^{-1}.

To complete the analysis, define $f(\ell)$ by

$$f(\ell) = -\frac{1}{L} \log \left(\sum_{i=1}^{m} e^{-\mathrm{load}^{(\ell)}(i)} \right) = -\frac{1}{L} \log \left(\sum_{i=1}^{m} w_i^{(\ell)} \right).$$

$f(\ell)$ gives a smooth approximation of the minimum value of $\mathrm{load}^{(\ell)}$; we have

$$\min_i \frac{\mathrm{load}^{(\ell)}(i)}{L} - \epsilon \leq f(x) \leq \min_i \frac{\mathrm{load}^{(\ell)}(i)}{L}$$

because $L \geq \log(m)/\epsilon$. Since $(Ax^{(L)})_i \geq \mathrm{load}^{(L)}(i)$ for all i, it suffices to show that $f(L) \geq 1 - O(\epsilon)$. We have

$$f(L) = f(0) + \sum_{\ell=1}^{L} f(\ell) - f(\ell-1) = -\epsilon - \frac{1}{L} \sum_{\ell=1}^{L} \log \left(\frac{\sum_{i=1}^{m} w_i^{(\ell)}}{\sum_{i=1}^{m} w_i^{(\ell-1)}} \right)$$

For each iteration ℓ, we have

$$\begin{aligned}\sum_{i=1}^{m} w_i^{(\ell)} &\overset{(c)}{\leq} (1 + O(\epsilon)) \sum_{i \in M_\ell} w_i^{(\ell)} \overset{(d)}{=} (1 + O(\epsilon)) e^{-1} \sum_{i \in M_\ell} w_i^{(\ell-1)} \\ &\leq (1 + O(\epsilon)) e^{-1} \sum_{i} w_i^{(\ell-1)} \leq e^{-(1-O(\epsilon))} \sum_{i} w_i^{(\ell-1)}.\end{aligned}$$

Here (c) critically uses inequality (5). (d) is because all elements $i \in M_\ell$ are covered by $y^{(\ell)}$, increasing their loads by 1 and their weight by e^{-1}. Thus

$$\log\left(\frac{\sum_{i=1}^m w_i^{(\ell)}}{\sum_i w_i^{(\ell-1)}}\right) \leq -(1 - O(\epsilon)).$$

Plugging back in, we have $f(L) \geq -\epsilon - \frac{1}{L}\sum_{\ell=1}^{L}(-(1 - O(\epsilon))) = 1 - O(\epsilon)$, as desired.

References

1. Martin Aigner and Thomas A. Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158(1):231–245, 1971.
2. Sepehr Assadi. A simple (1 - ϵ)-approximation semi-streaming algorithm for maximum (weighted) matching. In Merav Parter and Seth Pettie, editors, *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 337–354. SIAM, 2024. doi:10.1137/1.9781611977936.31.
3. András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015.
4. Joakim Blikstad. Breaking o(nr) for matroid intersection. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.31.
5. Joakim Blikstad, Jan van den Brand, Sagnik Mukhopadhyay, and Danupon Nanongkai. Breaking the quadratic barrier for matroid intersection. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 421–432. ACM, 2021. doi:10.1145/3406325.3451092.
6. Carl Brezovec, Gérard Cornuéjols, and Fred Glover. Two algorithms for weighted matroid intersection. *Mathematical Programming*, 36(1):39–53, October 1986.
7. Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, Sahil Singla, and Sam Chiu-wai Wong. Faster matroid intersection. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1146–1168. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00072.
8. Chandra Chekuri and Kent Quanrud. A fast approximation for maximum weight matroid intersection. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 445–457. SIAM, 2016.
9. William H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM J. Comput.*, 15(4):948–957, 1986.
10. Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications (Proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969)*, pages 69–87. Gordon and Breach, New York, 1970.
11. Jack Edmonds. Some well-solved problems in combinatorial optimization. In B. Roy, editor, *Combinatorial Programming: Methods and Applications*, pages 285–301, Dordrecht, 1975. Springer Netherlands.
12. Jack Edmonds and Delbert Ray Fulkerson. Transversals and matroid partition. *Journal of Research National Bureau of Standards Section B*, 69:147–153, 1965.
13. András Frank. A weighted matroid intersection algorithm. *Journal of Algorithms*, 2(4):328–336, December 1981.
14. András Frank. A quick proof for the matroid intersection weight-splitting theorem. *EGRES Quick-Proof*, (2008-03), 2008.

15 András Frank. *Connections in combinatorial optimization*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2011.

16 Satoru Fujishige and Xiaodong Zhang. An efficient cost scaling algorithm for the independent assignment problem. *Journal of the Operations Research Society of Japan*, 38(1):124–136, 1995.

17 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.

18 Chien-Chung Huang, Naonori Kakimura, and Naoyuki Kamiyama. Exact and approximation algorithms for weighted matroid intersection. *Math. Program.*, 177(1-2):85–112, 2019. doi:10.1007/S10107-018-1260-X.

19 Chien-Chung Huang and François Sellier. Robust sparsification for matroid intersection with applications. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 2916–2940. SIAM, 2024. doi:10.1137/1.9781611977912.104.

20 David R. Karger. Random sampling and greedy sparsification for matroid optimization problems. *Math. Program.*, 82:41–81, 1998.

21 Eugene L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9(1):31–56, December 1975.

22 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1049–1065. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.68.

23 Huy L. Nguyen. A note on cunningham's algorithm for matroid intersection. *CoRR*, abs/1904.04129, 2019. arXiv:1904.04129.

24 Kent Quanrud. Nearly linear time approximations for mixed packing and covering problems without data structures or randomization. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA@SODA 2020, Salt Lake City, UT, USA, January 6-7, 2020*, pages 69–80. SIAM, 2020.

25 Kent Quanrud. Faster exact and approximation algorithms for packing and covering matroids via push-relabel. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 2305–2336. SIAM, 2024. doi:10.1137/1.9781611977912.82.

26 Kent Quanrud. Quotient sparsification for submodular functions. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 5209–5248. SIAM, 2024. doi:10.1137/1.9781611977912.187.

27 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

28 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

29 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 81–90. ACM, 2004.