Subquadratic Weighted Matroid Intersection Under Rank Oracles

Ta-Wei Tu
National Taiwan University, Taipei, Taiwan

Abstract

Given two matroids $\mathcal{M}_1 = (V, \mathcal{I}_1)$ and $\mathcal{M}_2 = (V, \mathcal{I}_2)$ over an n-element integer-weighted ground set V, the weighted matroid intersection problem aims to find a common independent set $S^* \in \mathcal{I}_1 \cap \mathcal{I}_2$ maximizing the weight of S^*. In this paper, we present a simple deterministic algorithm for weighted matroid intersection using $\widetilde{O}(nr^{3/4} \log W)$ rank queries, where r is the size of the largest intersection of $\mathcal{M}_1$ and $\mathcal{M}_2$ and W is the maximum weight. This improves upon the best previously known $\widetilde{O}(nr \log W)$ algorithm given by Lee, Sidford, and Wong [FOCS'15], and is the first subquadratic algorithm for polynomially-bounded weights under the standard independence or rank oracle models. The main contribution of this paper is an efficient algorithm that computes shortest-path trees in weighted exchange graphs.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Mathematics of computing → Combinatorial optimization; Mathematics of computing → Matroids and greedoids

Keywords and phrases Matroids, Weighted Matroid Intersection, Combinatorial Optimization

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.63

Acknowledgements I would like to thank Prof. Hsueh-I Lu for advising the project and helpful suggestions on the writing and notation. I also thank Min-Yen Tsai for proof-reading an initial draft of this paper and the anonymous reviewers of ISAAC 2022 for their useful comments.

1 Introduction

Matroid Intersection. A matroid is an abstract structure that models the notion of independence on a given ground set V. In particular, a subset $S \subseteq V$ is either *independent* or *dependent*, such that the family of independent sets is well-structured (see Section 2 for a complete definition). Matroids model many fundamental combinatorial objects, and examples of independent sets of a matroid include acyclic subgraphs of an undirected graph and linearly independent rows of a matrix. One of the most important optimization problems related to matroids is *matroid intersection*: Given two matroids, we would like to find a set with the largest cardinality that is independent in both matroids. Similarly, in the weighted case, each element in the ground set is associated with an integer weight, and the weighted matroid intersection problem is to find the maximum-weight common independent set. These problems have been extensively studied in the past since they capture many combinatorial optimization problems such as bipartite matching and colorful spanning trees.

Oracle Model. Since we are dealing with general matroids without additional constraints, we have to specify a way of reading the description of the two matroids. One way is to express them directly by reading the truth table of independence. However, that would require an exponentially-sized input. Instead, we are given oracle access to the matroids, which gives us information about a queried set $S \subseteq V$. Standard oracles include the *independence oracle*, which returns whether S is independent in $O(\mathcal{T}_{\mathsf{ind}})$ time, and the *rank oracle*, which returns the rank, i.e., the size of the largest independent subset, of S in $O(\mathcal{T}_{\mathsf{rank}})$ time. In this paper, we focus on the stronger rank oracle model.

Prior Work. Polynomial-time algorithms for both the weighted and unweighted matroid intersection problems have long been designed and improved. For the unweighted case, Edmonds [10, 11, 12], Lawler [17], and also Aigner and Dowling [1] gave algorithms that run in $O(nr^2 \cdot \mathcal{T}_{\mathsf{ind}})$ time. Here, n denotes the number of elements in V and r denotes the size of the largest intersection of the two matroids. Cunningham [7] obtained an $O(nr^{3/2} \cdot \mathcal{T}_{\mathsf{ind}})$ algorithm using the "blocking-flow" idea. Lee, Sidford, and Wong [18] gave quadratic algorithms using the cutting-plane method, running in $\widetilde{O}(nr \cdot \mathcal{T}_{\mathsf{rank}} + n^3)$ and $\widetilde{O}(n^2 \cdot \mathcal{T}_{\mathsf{ind}} + n^3)$ times[1], respectively. This also gives rise to the "quadratic barrier" of matroid intersection: most previous algorithms involve building exchange graphs that contain $\Theta(nr)$ edges explicitly and therefore cannot go beyond quadratic time. Chakrabarty, Lee, Sidford, Singla, and Wong [4] were the first to partially break the barrier. They obtained $(1-\epsilon)$-approximation algorithms running in $\widetilde{O}(n/\epsilon \cdot \mathcal{T}_{\mathsf{rank}})$ and $\widetilde{O}(n^{3/2}/\epsilon^{3/2} \cdot \mathcal{T}_{\mathsf{ind}})$ times and an $\widetilde{O}(n\sqrt{r} \cdot \mathcal{T}_{\mathsf{rank}})$ exact algorithm. One of the major components of Chakrabarty et al.'s improvements is to show that edges in exchange graphs can be efficiently discovered using binary search (this was discovered independently by Nguyễn [19]). This technique also allows them to obtain improved $\widetilde{O}(nr \cdot \mathcal{T}_{\mathsf{ind}})$ exact algorithms. Combining the approximation algorithm and a faster augmenting-path algorithm, Blikstad, van den Brand, Mukhopadhyay, and Nanongkai [3] broke the quadratic barrier completely by giving an $\widetilde{O}(n^{9/5} \cdot \mathcal{T}_{\mathsf{ind}})$ exact algorithm. This result was later optimized by Blikstad [2] to $\widetilde{O}(nr^{3/4} \cdot \mathcal{T}_{\mathsf{ind}})$ by improving the approximation algorithm to run in $\widetilde{O}(n\sqrt{r}/\epsilon \cdot \mathcal{T}_{\mathsf{ind}})$ time.

For the weighted case, the blocking flow idea does not seem to apply anymore. Frank [13] obtained an $O(nr^2 \cdot \mathcal{T}_{\mathsf{ind}})$ algorithm by characterizing the optimality of a common independent set using weight splitting. Fujishige and Zhang [14] improved the running time to $\widetilde{O}(nr^{3/2} \log W \cdot \mathcal{T}_{\mathsf{ind}})$ by solving a more general *independent assignment* problem using a scaling framework. The same bound was achieved by Shigeno and Iwata [23] and also by Gabow and Xu [15]. Lee, Sidford, and Wong's [18] algorithms work for the weighted case as well, albeit with an extra factor of $\operatorname{polylog} W$, in $\widetilde{O}(n^2 \log W \cdot \mathcal{T}_{\mathsf{ind}} + n^3 \operatorname{polylog} W)$ and $\widetilde{O}(nr \log W \cdot \mathcal{T}_{\mathsf{rank}} + n^3 \operatorname{polylog} W)$ times. Huang, Kakimura, and Kamiyama [16] obtained a generic framework that transforms any algorithm that solves the unweighted case into one that solves the weighted case with an extra $O(W)$ factor. Plugging in the state-of-the-art algorithms of [4] and [2], we get $\widetilde{O}(n\sqrt{r} \cdot W \cdot \mathcal{T}_{\mathsf{rank}})$ and $\widetilde{O}(nr^{3/4} \cdot W \cdot \mathcal{T}_{\mathsf{ind}})$ algorithms. Chekuri and Quanrud [6] also gave an $\widetilde{O}(n^2/\epsilon^2 \cdot \mathcal{T}_{\mathsf{ind}})$ approximation algorithm which, according to [3], can be improved to subquadratic by applying more recent techniques. A similar $\widetilde{O}(nr^{3/2}/\epsilon \cdot \mathcal{T}_{\mathsf{ind}})$ approximation algorithm was obtained independently by Huang et al. [16].

Our Result. The question of whether weighted matroid intersection can be solved exactly in subquadratic time with polylogarithmic dependence on W under either oracle model remained open. We obtain the first subquadratic algorithm for exact weighted matroid intersection under rank oracles. The formal statement of Theorem 1 is presented as Theorem 5 in Section 2.

▶ **Theorem 1.** *Weighted matroid intersection can be solved in $\widetilde{O}(nr^{3/4} \log W \cdot \mathcal{T}_{\mathsf{rank}})$ time.*

Our algorithm relies on the framework of Fujishige-Zhang [14] and Shigeno-Iwata [23], where they first obtain an approximate solution by adjusting weights of some elements (similar to the "auction" algorithms for bipartite matching [20]) and then refine it by augmenting the solution iteratively.

[1] For function $f(n)$, $\widetilde{O}(f(n))$ denotes $O(f(n) \operatorname{polylog} f(n))$.

We obtain efficient algorithms for these two phases, leading to the final subquadratic algorithm.

2 Preliminaries

Notation. For a set S, let $|S|$ denote the cardinality and 2^S the power set of S. Let $S \setminus R$ consist of elements of S which are not in R. Let $e = (u, v, w)$ denote a weighted directed edge directing from u to v with weight $w = w_E(e)$ and (u, v) be its unweighted counterpart.[2] Let $\text{head}(e) = v$ and $\text{tail}(e) = u$. For an edge set E, let $\text{head}(E) = \{\text{head}(e) \mid e \in E\}$ and $\text{tail}(E) = \{\text{tail}(e) \mid e \in E\}$. For functions f, g mapping from a set V to $\mathbb{R}$, let $f + g$, $f - g$, and $f + c$ for $c \in \mathbb{R}$ denote functions from V to $\mathbb{R}$ with $(f + g)(x) = f(x) + g(x)$, $(f - g)(x) = f(x) - g(x)$, and $(f + c)(x) = f(x) + c$ for each $x \in V$. We often abuse notation and use f to denote the function from 2^V to $\mathbb{R}$ with $f(S) = \sum_{x \in S} f(x)$ for each $S \subseteq V$.

Matroid. Let V be a finite set and $w : V \to \mathbb{Z}$ be a given weight function. For $S \subseteq V$, let $\overline{S} = V \setminus S$. Let $n = |V|$ and $W = \max_{x \in V} |w(x)|$. An ordered pair $\mathcal{M} = (V, \mathcal{I})$ with *ground set* V and a non-empty family $\emptyset \in \mathcal{I} \subseteq 2^V$ is a *matroid* if

M1. for each $S \in \mathcal{I}$ and $R \subseteq S$, it holds that $R \in \mathcal{I}$, and

M2. for each $R, S \in \mathcal{I}$ with $|R| < |S|$, there exists an $x \in S \setminus R$ such that $R \cup \{x\} \in \mathcal{I}$.

Sets in $\mathcal{I}$ are *independent*; sets not in $\mathcal{I}$ are *dependent*. A *basis* is a maximal independent set. A *circuit* is a minimal dependent set. It is well-known from the definition of matroid that all bases are of the same cardinality. For an independent set S and $x \notin S$, $S \cup \{x\}$ contains at most one circuit C and if it does, then $x \in C$ (see [21, Lemma 1.3.3]). The *rank* of $S \subseteq V$, denoted by $\mathsf{rank}(S)$, is the size of the largest $S' \subseteq S$ such that $S' \in \mathcal{I}$. The rank of $\mathcal{M}$ is the rank of V, i.e., the size of the bases of $\mathcal{M}$. Given two matroids $\mathcal{M}_1 = (V, \mathcal{I}_1)$ and $\mathcal{M}_2 = (V, \mathcal{I}_2)$ over the same ground set, the *weighted matroid intersection* problem is to find an $S^* \in \mathcal{I}_1 \cap \mathcal{I}_2$ maximizing $w(S^*)$. Let $r = \max_{S \in \mathcal{I}_1 \cap \mathcal{I}_2} |S|$. In this paper, the two matroids are accessed through *rank oracles*, one for each matroid. Specifically, let $\mathsf{rank}_1(\cdot)$ and $\mathsf{rank}_2(\cdot)$ denote the rank functions of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. We assume that given pointers to a linked list containing elements of S (see, e.g., [5]), the rank oracles compute $\mathsf{rank}_1(S)$ and $\mathsf{rank}_2(S)$ in $O(\mathcal{T}_{\mathsf{rank}})$ time. With the $O(n\sqrt{r} \log n \cdot \mathcal{T}_{\mathsf{rank}})$ unweighted matroid intersection algorithm of Chakrabarty et al. [4], we also assume that $\mathcal{M}_1$ and $\mathcal{M}_2$ are of the same rank and share a common basis $S^{(0)}$ of size r by adjusting the given rank oracles properly.[3] By adding r zero-weight elements to V, we may also assume that each common independent set $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ is contained in a common basis of the same weight.[4] Therefore, it suffices to find a common basis S^* maximizing $w(S^*)$. Note that elements with negative weights can be safely discarded from V.

---

[2] We use $w_E(\cdot)$ to denote edge weights so that they can be unambiguously distinguished from the element weights $w(\cdot)$ introduced later.

[3] We can compute r via the unweighted matroid intersection algorithm and regard all sets of size greater than r as dependent.

[4] In particular, let $Z = \{z_1, \ldots, z_r\}$ be the set of newly added zero-weight elements. For each $i \in \{1, 2\}$, instead of working with $\mathcal{M}_i$, we now work with $\widetilde{\mathcal{M}_i} = (V \cup Z, \widetilde{\mathcal{I}_i})$ such that for each $\widetilde{S} \subseteq V \cup Z$, $\widetilde{S} \in \widetilde{\mathcal{I}_i}$ if and only if $|\widetilde{S}| \leq r$ and $\widetilde{S} \setminus Z \in \mathcal{I}_i$. This change is reflected in the new rank function $\widetilde{\mathsf{rank}}_i(\widetilde{S}) = \min(\mathsf{rank}_i(\widetilde{S} \setminus Z) + |\widetilde{S} \cap Z|, r)$, which can be implemented via the given oracle rank_i.

Weight-Splitting. For weight function $f : V \to \mathbb{R}$, a basis S of matroid $\mathcal{M}$ is *f-maximum* if $f(S) \geq f(R)$ holds for each basis R of $\mathcal{M}$. Let $w^\epsilon = (w_1^\epsilon, w_2^\epsilon)$ with $w_i^\epsilon : V \to \mathbb{R}$ being a weight function for each $i \in \{1, 2\}$. Let $w^\epsilon(x) = w_1^\epsilon(x) + w_2^\epsilon(x)$. We say that w^ϵ is an *ϵ-splitting* (see, e.g., [23]) of w with $\epsilon > 0$ if $w(x) \leq w^\epsilon(x) \leq w(x) + \epsilon$ holds for each $x \in V$. If w^ϵ is an ϵ-splitting of w and S_i is a w_i^ϵ-maximum basis of $\mathcal{M}_i$ for each $i \in \{1, 2\}$, then we call (w^ϵ, S) with $S = (S_1, S_2)$ an *ϵ-partial-solution* of w. Note that by M1, $S_1 \cap S_2$ is a common independent set. If $S_1 = S_2$, then (w^ϵ, S) is an *ϵ-solution* of w. In this case, we may abuse notation and refer to S_1 as simply S.

Matroid Algorithms. The unweighted version of the following lemma was shown in [4] (it was also mentioned in [19]), and it was extended to the weighted case implicitly in [3].

▶ **Lemma 2** ([4, Lemma 13], [19], and [3])**.** *For $i \in \{1, 2\}$, given $S \in \mathcal{I}_i$, $B \subseteq S$ (respectively, $B \subseteq \overline{S}$), $x \in \overline{S}$ (respectively, $x \in S$), and weight function $f : V \to \mathbb{R}$, it takes $O(\log |B| \cdot \mathcal{T}_{\mathsf{rank}})$ time to either obtain a $b \in B$ minimizing/maximizing $f(b)$ such that $(S \setminus \{b\}) \cup \{x\} \in \mathcal{I}_i$ (respectively, $(S \setminus \{x\}) \cup \{b\} \in \mathcal{I}_i$) or report that such an element does not exist in B.*

The main idea of Lemma 2 is to perform binary search on B ordered by $f(\cdot)$. Throughout this paper, we will maintain such an ordered set in a balanced binary search tree where each element holds pointers to its successor and predecessor and each node holds pointers to the first and the last elements in its corresponding subtree. This allows us to perform binary search on the tree and obtain pointers to the linked list containing elements in a consecutive range efficiently.

The following greedy algorithm for finding a maximum-weight basis is folklore.

▶ **Lemma 3** (See, e.g., [9])**.** *It takes $O(n \log n + n\mathcal{T}_{\mathsf{rank}})$ time to obtain a f-maximum basis S of a given matroid $\mathcal{M}$ and weight function $f : V \to \mathbb{R}$.*

2.1 The Framework

The core of our algorithm is the following subroutine.

▶ **Theorem 4.** *Given a 2ϵ-solution $(w^{2\epsilon}, S')$ of w, it takes $O(nr^{3/4} \log n \cdot \mathcal{T}_{\mathsf{rank}})$ time to obtain an ϵ-solution (w^ϵ, S).*

With Theorem 4, the weighted matroid intersection algorithm follows from the standard weight-scaling framework (see, e.g, [14, 23]). Recall that our goal is to find a maximum-weight common basis.

▶ **Theorem 5** (Weighted Matroid Intersection)**.** *Given two matroids $\mathcal{M}_1 = (V, \mathcal{I}_1)$ and $\mathcal{M}_2 = (V, \mathcal{I}_2)$, it takes $O(nr^{3/4} \log n \log (rW) \cdot \mathcal{T}_{\mathsf{rank}})$ time to obtain an $S^* \in \mathcal{I}_1 \cap \mathcal{I}_2$ maximizing $w(S^*)$.*

Proof. Let $w^W = (w_1^W, w_2^W)$ with $w_i^W(x) = \frac{W}{2}$ for each $x \in V$ and the initial common basis $S^{(0)}$ obtained via the unweighted matroid intersection algorithm be a W-solution of w. Repeatedly apply Theorem 4 for $O(\log rW)$ iterations to obtain a $\frac{1}{2r}$-solution $(w^{\frac{1}{2r}}, S^*)$. For each $S \in \mathcal{I}_1 \cap \mathcal{I}_2$, we have

$$w(S) \leq w^{\frac{1}{2r}}(S) \leq w^{\frac{1}{2r}}(S^*) \leq w(S^*) + r \cdot \frac{1}{2r} < w(S^*) + 1.$$

Since $w(S)$ and $w(S^*)$ are integers, S^* is a maximum-weight common basis. The algorithm runs in $O(nr^{3/4} \log n \log (rW) \cdot \mathcal{T}_{\mathsf{rank}})$ time. The theorem is proved. ◀

The rest of the paper proves Theorem 4.

3 The Algorithm

As in [14] and [23], the algorithm of Theorem 4 consists of the following two parts.

3.1 Weight Adjustment

The first part of the algorithm is the following subroutine which computes two bases S_1 and S_2 with a large enough intersection. This part is essentially the same as Shigeno and Iwata's algorithm [23], except that we replace the fundamental (co-)circuit queries in it with calls to Lemma 2.

▶ **Lemma 6.** *Given a 2ϵ-solution $(w^{2\epsilon}, S')$ and a parameter $1 \leq k \leq r$, it takes $O(nk \log n \cdot \mathcal{T}_{\mathsf{rank}})$ time to obtain an ϵ-partial-solution (w^{ϵ}, S) with $|S_1 \cap S_2| \geq \left(1 - \frac{O(1)}{k}\right) r$.*

Since the algorithm and analysis are essentially the same as in [23], here we only describe how we can obtain S_1 and S_2 in the desired time bound. Please refer to [23] or Lemma 11 in Appendix A for the proof of $|S_1 \cap S_2| \geq \left(1 - \frac{O(1)}{k}\right) r$.

Algorithm of Lemma 6. Let $w^{\epsilon} = (w_1^{2\epsilon}, w - w_1^{2\epsilon} + \epsilon)$ be the initial ϵ-splitting and S_i be the w_i^{ϵ}-maximum basis of $\mathcal{M}_i$ obtained by Lemma 3 in $O(n \log n + n\mathcal{T}_{\mathsf{rank}})$ time for each $i \in \{1, 2\}$. Let $p(x) = 0$ for each $x \in V$. Repeat the following *weight adjustment* for an arbitrary $x \in S_1 \setminus S_2$ with $p(x) < k$ until such an x becomes non-existent.

- If $w^{\epsilon}(x) = w(x) + \epsilon$, then set $w_1^{\epsilon}(x) \leftarrow w_1^{\epsilon}(x) - \epsilon$. Apply Lemma 2 to obtain a $y \in V \setminus S_1$ maximizing $w_1^{\epsilon}(y)$ such that $(S_1 \setminus \{x\}) \cup \{y\} \in \mathcal{I}_1$. If $w_1^{\epsilon}(x) < w_1^{\epsilon}(y)$, then set $S_1 \leftarrow (S_1 \setminus \{x\}) \cup \{y\}$.
- Otherwise, set $p(x) \leftarrow p(x) + 1$ and $w_2^{\epsilon}(x) \leftarrow w_2^{\epsilon}(x) + \epsilon$. Apply Lemma 2 to obtain a $y \in S_2$ minimizing $w_2^{\epsilon}(y)$ such that $(S_2 \setminus \{y\}) \cup \{x\} \in \mathcal{I}_2$. If $w_2^{\epsilon}(x) > w_2^{\epsilon}(y)$, then set $S_2 \leftarrow (S_2 \setminus \{y\}) \cup \{x\}$.

Since $p(x)$ is only incremented when $x \in S_1 \setminus S_2$, we have $p(x) \leq k$ for each $x \in V$ when the procedure terminates. Apparently, $w^{\epsilon}(x)$ oscillates between $w(x)$ and $w(x) + \epsilon$, and thus the number of weight adjustments for x is bounded by $2p(x)$. We also have that S_i remains w_i^{ϵ}-maximum for each $i \in \{1, 2\}$ due to the potential exchange of x and y after the adjustment. Each weight adjustment takes $O(\mathcal{T}_{\mathsf{rank}} \log n)$ time by Lemma 2, hence the total running time is $O(nk \log n \cdot \mathcal{T}_{\mathsf{rank}})$.

3.2 Augmentation

With S_1 and S_2 obtained from Lemma 6, we then run "few" augmentations to make these two bases equal. To do so, we need the following notion of exchange graphs, which is slightly different compared to previous algorithms for unweighted matroid intersection (e.g., [3, 4, 7, 17]).

Exchange Graph. Let (w^{ϵ}, S) be an ϵ-partial-solution of w with $S_1 \neq S_2$. The *exchange graph* with respect to (w^{ϵ}, S) is a weighted directed multi-graph $G_{w^{\epsilon},S} = (V \cup \{s, t\}, E)$ with $s, t \notin V$ and $E = E_1 \cup E_2 \cup E_s \cup E_t$, where

$$E_1 = \{(x, y, w_1^{\epsilon}(x) - w_1^{\epsilon}(y)) \mid x \in S_1, y \notin S_1, \text{ and } (S_1 \setminus \{x\}) \cup \{y\} \in \mathcal{I}_1\},$$
$$E_2 = \{(y, x, w_2^{\epsilon}(x) - w_2^{\epsilon}(y)) \mid x \in S_2, y \notin S_2, \text{ and } (S_2 \setminus \{x\}) \cup \{y\} \in \mathcal{I}_2\},$$
$$E_s = \{(s, x, 0) \mid x \in S_1 \setminus S_2\}, \text{ and}$$
$$E_t = \{(x, t, 0) \mid x \in S_2 \setminus S_1\}.$$

Since S_i is w_i^ϵ-maximum for each $i \in \{1, 2\}$, all edge weights are non-negative. Note that this definition of exchange graph is a simplified version of the *auxiliary graph* defined by Fujishige and Zhang [14] to solve the more generalized *independent assignment* problem[5]. We have the following properties of the exchange graph, for which we also provide simplified and more direct proofs for self-containedness in Appendix A.

▶ **Lemma 7** ([14]; See Appendix A)**.** *$G_{w^\epsilon,S}$ admits an st-path.*

Let $d(x)$ be the sx-distance in $G_{w^\epsilon,S}$ for each $x \in V$ (set $d(x)$ to a large number if x is unreachable from s; see Section 3.3 for the exact value) and P be the shortest st-path with the least number of edges.

Let $\widehat{S}_1 = (S_1 \setminus \mathrm{tail}(P \cap E_1)) \cup \mathrm{head}(P \cap E_1)$ and $\widehat{S}_2 = (S_2 \setminus \mathrm{head}(P \cap E_2)) \cup \mathrm{tail}(P \cap E_2)$ be S_1 and S_2 augmented by P. Let $\widehat{w}_1^\epsilon(x) = w_1^\epsilon(x) + d(x)$ and $\widehat{w}_2^\epsilon(x) = w_2^\epsilon(x) - d(x)$. We have that $(\widehat{w}^\epsilon, \widehat{S})$ with $\widehat{w}^\epsilon = (\widehat{w}_1^\epsilon, \widehat{w}_2^\epsilon)$ and $\widehat{S} = (\widehat{S}_1, \widehat{S}_2)$ is a better ϵ-solution (note that $\widehat{w}^\epsilon$ is indeed an ϵ-splitting). In other words, we have the following.

▶ **Lemma 8** ([14]; See Appendix A)**.** *It holds that $(\widehat{w}^\epsilon, \widehat{S})$ is an ϵ-solution with $|\widehat{S}_1 \cap \widehat{S}_2| > |S_1 \cap S_2|$.*

With the above properties and Lemma 6, we finish our algorithm with the following shortest-path procedure. Note that in order to make the algorithm subquadratic, we do not construct the exchange graphs explicitly. Nevertheless, we show that a partial construction suffices to compute the shortest-path trees in them.

▶ **Lemma 9.** *It takes $O(n\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time to obtain $d(x)$ for each $x \in V$ and the shortest st-path with the least number of edges in $G_{w^\epsilon,S}$.*

We are now ready to prove Theorem 4.

Proof of Theorem 4. Apply Lemma 6 with $k = r^{3/4}$ to obtain an ϵ-partial-solution (w^ϵ, S) of w such that $|S_1 \cap S_2| \geq r - O(r^{1/4})$ in $O(nr^{3/4}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time. For $O(r^{1/4})$ iterations, apply Lemmas 8 and 9 to obtain $(\widehat{w}^\epsilon, \widehat{S})$ with $\widehat{S}_1$ and $\widehat{S}_2$ having a larger intersection than S_1 and S_2 do, and set $(w^\epsilon, S) \leftarrow (\widehat{w}^\epsilon, \widehat{S})$ until $S_1 = S_2$. This takes overall $O(nr^{3/4}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time as well. Note that $w^\epsilon(x) = w_1^\epsilon(x) + w_2^\epsilon(x)$ remains the same, completing the proof. ◀

The remainder of this section proves Lemma 9. For the ease of notation, we abbreviate $G_{w^\epsilon,S}$ as G.

Intuitively, we would like to run Dijkstra's algorithm [8] on G to build a shortest-path tree. However, naïve implementation takes $O(nr)$ time since we might need to relax $O(nr)$ edges. This is unlike the BFS algorithm of Chakrabarty et al. [4] for the unweighted case, where we can immediately mark all out-neighbors of the current vertex as "visited", leading to a near-linear running time. To speed things up, note that using Lemma 2, for a vertex x, we can efficiently find the vertex which is "closest" to x. Let F denote the set of visited vertices whose exact distances are known. The closest unvisited vertex to F must be closest to some $x \in F$. Therefore, in each iteration, it suffices to only relax the "shortest" edge from

[5] Specifically, given a bipartite graph $G = (V_1 \cup V_2, E)$ with V_1 and V_2 being copies of V and two matroids $\mathcal{M}_1 = (V, \mathcal{I}_1)$, $\mathcal{M}_2 = (V, \mathcal{I}_2)$ on V, the independent assignment problem aims to find the largest $S_1 \in \mathcal{I}_1$ and $S_2 \in \mathcal{I}_2$ such that G admits a perfect matching between $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$. Analogously, the weighted version of the problem wants to find S_1 and S_2 such that the weight of the maximum-weight perfect matching between S_1 and S_2 is maximized. Clearly, the (weighted) matroid intersection problem is a special case of the (weighted) independence assignment problem with $E = \{(v, v) \mid v \in V\}$.

each $x \in F$. This can be done efficiently by maintaining a set of "recently visited" vertices B of size roughly $\sqrt{n}$ and computing the distance estimate from $F \setminus B$ to all unvisited vertices[6]. In each iteration, we relax the shortest edge from each $x \in B$, and now the vertex with the smallest distance estimate is closest to F and therefore we include it into B (and thus F). When B grows too large, we clear B and recompute the distance estimates from F in $\widetilde{O}(n)$ queries. This leads to a subquadratic algorithm. We now prove the lemma formally.

Proof of Lemma 9. The algorithm builds a shortest-path tree of G using Dijkstra's algorithm. We maintain a distance estimate $\widehat{d}(x)$ for each $x \in V \cup \{s,t\}$. Initially, $\widehat{d}(x) = 0$ for each $x \in (S_1 \setminus S_2) \cup \{s\}$ and $\widehat{d}(x) = \infty$ for other vertices. Edge set E_t is only for the convenience of defining an st-path and thus we may ignore it here. Let F be the set of *visited* vertices whose distance estimates are correct, i.e., $d(x) = \widehat{d}(x)$ holds for each $x \in F$. Initially, $F = \{s\}$. The algorithm runs in at most n iterations, and in the t-th iteration, we visit a new vertex v_t such that $d(v_t) = \widehat{d}(v_t)$ and $d(v_t) \leq d(v)$ for each $v \notin F$. We maintain two *buffers* $B_1 \subseteq F \cap S_1$ and $B_2 \subseteq F \cap \overline{S_2}$ containing vertices in S_1 and $\overline{S_2}$ that are visited "recently". That is, after the t-th iteration, we have $B_1 = \{v_i, \ldots, v_t\} \cap S_1$ or $B_1 = \emptyset$ and $B_2 = \{v_j, \ldots, v_t\} \cap \overline{S_2}$ or $B_2 = \emptyset$ for some $i, j \leq t$. Recall that E_1 and E_2 are the edges in G that correspond to exchange relations in $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. For each $i \in \{1,2\}$ and edge $e = (x,y)$, let $w_{E_i}(x,y) = |w_i^{\epsilon}(x) - w_i^{\epsilon}(y)|$ be the edge weight of e in E_i if $e \in E_i$ and $w_{E_i}(x,y) = \infty$ otherwise. Let $E(B_i) = \{(x,y) \in E_i \mid x \in B_i \text{ and } y \notin F\}$ be edges in E_i directing from B_i to $V \setminus F$ and $E(B) = E(B_1) \cup E(B_2)$. Let $E(F) = \{(x,y) \in E_1 \cup E_2 \mid x \in F \text{ and } y \notin F\}$. For $v \in V \setminus F$ and edge set E' such that $\text{tail}(E') \subseteq F$, let $\widetilde{d}(v, E') = \min_{(x,v) \in E'}\{d(x) + w_E(x,v)\}$ be the shortest distance to v "relaxed" by edges in E' (recall that $w_E(x,v)$ is the weight of the edge (x,v)). We maintain the following invariants after each iteration of the algorithm except the last one.

(i) $d(v) \leq \widehat{d}(v) \leq \widetilde{d}(v, E(F) \setminus E(B))$ holds for each $v \in V \setminus F$.

(ii) There exists a $v \in V \setminus F$ such that $\widehat{d}(v) = d_F^* := \min_{u \in V \setminus F}\{\widetilde{d}(u, E(F))\}$.

Intuitively, Invariant (i) asserts that all edges in $E(F) \setminus E(B)$ are "relaxed" while Invariant (ii) ensures that the distance estimate of the target vertex, i.e., one with the shortest distance from s, is correct. Initially, both invariants are satisfied since $\widehat{d}(x) = 0$ holds for each $x \in S_1 \setminus S_2$. We maintain a priority queue Q containing vertices in $V \setminus F$ ordered by $\widehat{d}(\cdot)$. In the t-th iteration, let v_t be the vertex v with the smallest $\widehat{d}(v)$. By Invariants (i) and (ii), we have $\widehat{d}(v_t) = d_F^*$ and thus $d(v_t) \leq d(v')$ holds for each $v' \in V \setminus F$ according to Dijkstra's algorithm. As such, we push v_t into F and update B_1, B_2 appropriately by checking if v_t belongs to S_1 and $\overline{S_2}$. Now, we would like to modify $\widehat{d}(v)$ for some $v \in V \setminus F$ so that both invariants remain true. For each $i \in \{1,2\}$, depending on the size of B_i, we perform one of the following.

1. If $|B_i| \geq \sqrt{r}$, then we compute $\widetilde{d}_i(v) = \widetilde{d}(v, E(B_1))$ and set $\widehat{d}(v) \leftarrow \min(\widehat{d}(v), \widetilde{d}_i(v))$ for each $v \in V \setminus F$ using Lemma 10 below. For $i = 1$, by definition of G, $\text{head}(E_1) \subseteq V \setminus S_1$ and thus we only need to compute $\widetilde{d}_i(v)$ for $v \in V \setminus S_1$, and therefore Lemma 10 takes $O(n \log n \cdot \mathcal{T}_{\mathsf{rank}})$ time. For $i = 2$, similarly, $\text{head}(E_2) \subseteq S_2$ and thus we only need to compute $\widetilde{d}_i(v)$ for $v \in S_2$, taking $O(r \log n \cdot \mathcal{T}_{\mathsf{rank}})$ time. Then, we set $B_i \leftarrow \emptyset$, and the above modification ensures that Invariant (i) holds since $\widetilde{d}(v, E(F)) = \min(\widetilde{d}(v, E(F) \setminus E(B)), \widetilde{d}(v, E(B)))$.

[6] In the actual algorithm, we maintain two buffers instead of one to further improve the running time to $\widetilde{O}(n\sqrt{r})$ from $\widetilde{O}(n\sqrt{n})$. This makes our weighted matroid intersection algorithm $o(nr)$ as opposed to just $o(n^2)$.

2. If $|B_i| < \sqrt{r}$, then we do not clear B_i, and therefore Invariant (i) trivially holds. For each $b \in B_i$, we find a $v_b \in V \setminus F$ minimizing $d(b) + w_{E_i}(b, v_b)$ via Lemma 2 as follows. If $i = 1$, then we have $w_{E_1}(b, v_b) = w_1^\epsilon(b) - w_1^\epsilon(v_b)$, and thus we find the v_b maximizing $w_1^\epsilon(v_b)$ such that $(S_1 \setminus \{b\}) \cup \{v_b\} \in \mathcal{I}_1$. If $i = 2$, then $w_{E_2}(b, v_b) = w_2^\epsilon(v_b) - w_2^\epsilon(b)$, and thus we find the v_b minimizing $w_2^\epsilon(v_b)$ such that $(S_2 \setminus \{v_b\}) \cup \{b\} \in \mathcal{I}_2$. Then, we set $\widehat{d}(v_b) \leftarrow \min(\widehat{d}(v_b), d(b) + w_{E_i}(b, v_b))$ and update v_b's position in Q appropriately. This takes $O(\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time.

In both cases, as argued above, Invariant (i) holds. We argue that Invariant (ii) holds after the iteration as well. Let $B_1^{(t)}$ be B_1 after the t-th iteration and define $B_2^{(t)}$ and $F^{(t)}$ similarly. Let $E(B^{(t)})$ denote $E(B_1^{(t)}) \cup E(B_2^{(t)})$. Let $v^* = \arg\min_{v \in V \setminus F^{(t)}}\{\widetilde{d}(v, E(F^{(t)}))\}$ be an unvisited vertex after the t-th iteration with the smallest distance from s and let $e^* = (u, v^*)$ be the edge such that $u \in F^{(t)}$ and $d(v^*) = d(u) + w(e^*)$. That is, e^* is the edge connecting v^* and its parent in the shortest-path tree. If $e^* \in E(F^{(t-1)}) \setminus E(B^{(t-1)})$, then Invariant (ii) trivially follows from the end of the $(t-1)$-th iteration. Otherwise, we must have either $e^* \in E(B_1^{(t-1)})$ or $e^* \in E(B_2^{(t-1)})$. Without loss of generality, let's assume $e^* \in E(B_1^{(t-1)})$. If $|B_1^{(t-1)}| + 1 \geq \sqrt{r}$ (i.e., Case 1), then after setting $B_1^{(t)} \leftarrow \emptyset$, Invariant (ii) follows from the fact the Invariant (i) holds for v^* and $\widetilde{d}(v^*, E(F^{(t)}) \setminus E(B^{(t)})) \leq d(u) + w_E(e^*)$ since $e^* \in E(F^{(t)}) \setminus E(B^{(t)})$. If $|B_1^{(t-1)}| + 1 < \sqrt{r}$ (i.e., Case 2), then there must exists a $b \in B_1^{(t)}$ such that $d(b) + w_{E_1}(b, v^*) = \min_v\{d(b) + w_{E_1}(b, v)\}$ and thus we have at least one $v_b \in V \setminus F^{(t)}$ such that $\widehat{d}(v_b) \leq d(b) + w_{E_1}(b, v_b) = d(v^*)$. This shows that Invariant (ii) indeed holds after the t-th iteration. The correctness of the algorithm follows from the two invariants and the analysis of Dijkstra's algorithm.

To bound the total running time, observe that for B_1, Case 1 happens at most $O(r/\sqrt{r}) = O(\sqrt{r})$ times since $|S_1| = r$. Thus, it takes $O(n\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time in total. Similarly, for B_2, Case 1 happens at most $O(n/\sqrt{r})$ time, taking $O(n/\sqrt{r} \cdot r\log n \cdot \mathcal{T}_{\mathsf{rank}}) = O(n\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time in total as well. For Case 2, each iteration takes $O(\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time, contributing a total of $O(n\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time. As a result, the algorithm runs in $O(n\sqrt{r}\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time, as claimed.

Finally, it is easy to maintain balanced binary search trees of elements ordered by w_1^ϵ, w_2^ϵ, $\widehat{d} + w_1^\epsilon$, and $\widehat{d} - w_2^\epsilon$ in $O(n\log n)$ time throughout the procedure so that Lemma 2 can be applied without overhead. The shortest st-path can also be easily recovered by maintaining the optimal parent in the shortest-path tree for each vertex. This proves the lemma. ◀

▶ **Lemma 10.** *For each $i \in \{1, 2\}$, given $B_i \subseteq F$ and $R \subseteq V \setminus F$, it takes takes $O(|R|\log n \cdot \mathcal{T}_{\mathsf{rank}})$ time to compute $\widetilde{d}(v, E(B_i))$ for all $v \in R$.*

Proof. For $i = 1$ and $e = (b, v) \in E(B_i)$, we have $w_E(e) = w_1^\epsilon(b) - w_1^\epsilon(v)$. Therefore, $d(v, E(B_1))$ can be computed by finding the $b \in B$ with the smallest $d(b) + w_1^\epsilon(b)$ such that $(S_1 \setminus \{b\}) \cup \{v\} \in \mathcal{I}_1$ via Lemma 2. Similarly, for $i = 2$, we have $w_E(e) = w_2^\epsilon(v) - w_2^\epsilon(b)$, and thus $d(v, E(B_2))$ can be computed by finding the $b \in B$ with the smallest $d(b) - w_2^\epsilon(b)$. The lemma simply follows by calling Lemma 2 once for each $v \in R$. ◀

3.3 Bounding the Numbers

Finally, to conclude the analysis of our algorithm, we argue that the numbers such as $w_1^\epsilon(x)$ and $w_2^\epsilon(x)$ are bounded by $\widetilde{O}(\mathrm{poly}(nW))$ so that the number of bits needed to store them and the time for a single arithmetic operation only grow by a constant factor. In the weight adjustment stage, each number is adjusted at most $O(r)$ times and each adjustment changes the number by at most $O(W)$ since ϵ is at most W. Therefore, the accumulative change to a number via weight adjustments is at most $O(\text{poly}(nW))$. For growth incurred by augmentations, we first assume that all vertices are reachable from s in $G_{w^\epsilon,S}$. Consider a single run of Lemma 9 and fix an $x \in V$. Let $P_x = \{s, v_1, \ldots, v_k\}$ with $v_k = x$ be the shortest sx-path in $G_{w^\epsilon,S}$. Suppose that $(v_1, v_2), (v_{k-1}, v_k) \in E_1$, then by definition, we have

$$\begin{aligned} d(x) &= w_1^\epsilon(v_1) - w_1^\epsilon(v_2) + w_2^\epsilon(v_3) - w_2^\epsilon(v_2) + \cdots + w_1^\epsilon(v_{k-1}) - w_1^\epsilon(v_k) \\ &\le w_1^\epsilon(v_1) - w(v_2) + (w(v_3) + \epsilon) + \cdots + (w(v_{k-1}) + \epsilon) - w_1^\epsilon(v_k) \\ &\le w_1^\epsilon(v_1) - w_1^\epsilon(v_k) + \left(\sum_{i=2}^{k-1} (-1)^{i+1} w(v_i)\right) + nW. \end{aligned}$$

Since $\widehat{w}_1^\epsilon(x) = w_1^\epsilon(x) + d(x)$ and $\widehat{w}_2^\epsilon(x) = w_2^\epsilon(x) - d(x)$ as defined in Lemma 8, we have

$$|\widehat{w^\epsilon}_1(x)| \le |w_1^\epsilon(v_1)| + 2nW \quad \text{and} \quad |\widehat{w}_2^\epsilon(x)| \le |w_1^\epsilon(v_1)| + 2nW. \tag{1}$$

Similarly, if $(v_{k-1}, v_k) \in E_2$, then

$$\begin{aligned} d(x) &= w_1^\epsilon(v_1) - w_1^\epsilon(v_2) + w_2^\epsilon(v_3) - w_2^\epsilon(v_2) + \cdots + w_2^\epsilon(v_k) - w_2^\epsilon(v_{k-1}) \\ &\le w_1^\epsilon(v_1) - w(v_2) + (w(v_3) + \epsilon) + \cdots + (w(v_{k-2}) + \epsilon) - w(v_{k-1}) + w_2^\epsilon(v_k) \\ &\le w_1^\epsilon(v_1) + w_2^\epsilon(v_k) + \left(\sum_{i=2}^{k-1} (-1)^{i+1} w(v_i)\right) + nW, \end{aligned}$$

implying (1) as well. The case when $(v_1, v_2) \in E_2$ holds similarly, except now we have

$$|\widehat{w}_1^\epsilon(x)| \le |w_2^\epsilon(v_1)| + 2nW \quad \text{and} \quad |\widehat{w}_2^\epsilon(x)| \le |w_2^\epsilon(v_1)| + 2nW. \tag{2}$$

Since the number of augmentations is $\widetilde{O}(r^{1/4})$, we indeed have that $|w_1^\epsilon(x)| = |w_2^\epsilon(x)| = O(\text{poly}(nW)) = \Theta((nW)^k)$ for some constant k. For the case where some vertex x is not reachable from s, we can simply set $d(x)$ to some $c(nW)^{k+1}$ for a large enough constant c and the desired bound still holds.

4 Concluding Remarks

We present a simple subquadratic algorithm for weighted matroid intersection under the rank oracle model, providing a partial yet affirmative answer to one of the open problems raised by Blikstad et al. [3]. Whether the same is achievable under the independence oracle model remains open. It seems that our techniques for computing shortest-path trees do not solely result in a subquadratic augmenting-path algorithm under the independence oracle. Removing the dependence on $\log W$ and making the algorithm run in strongly-polynomial time is also of interest. Finally, as noted in [3], there were very few non-trivial lower bound results for matroid intersection. It would be helpful to see if there is any super-linear lower bound on the number of queries for these problems or even for computing shortest-path trees in the exchange graphs under either oracle model.

References

1 Martin Aigner and Thomas A Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158(1):231–245, 1971. doi:10.2307/1995784.

2 Joakim Blikstad. Breaking o(nr) for matroid intersection. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, pages 31:1–31:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.31.

3 Joakim Blikstad, Jan van den Brand, Sagnik Mukhopadhyay, and Danupon Nanongkai. Breaking the quadratic barrier for matroid intersection. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 421–432. ACM, 2021. doi:10.1145/3406325.3451092.

4 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, Sahil Singla, and Sam Chiu-wai Wong. Faster matroid intersection. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 1146–1168. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00072.

5 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. Subquadratic submodular function minimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1220–1231. ACM, 2017. doi:10.1145/3055399.3055419.

6 Chandra Chekuri and Kent Quanrud. A fast approximation for maximum weight matroid intersection. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 445–457. SIAM, 2016. doi:10.1137/1.9781611974331.ch33.

7 William H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM J. Comput.*, 15(4):948–957, 1986. doi:10.1137/0215066.

8 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

9 Jack Edmonds. Matroids and the greedy algorithm. *Math. Program.*, 1(1):127–136, 1971. doi:10.1007/BF01584082.

10 Jack Edmonds. Matroid intersection. *Annals of Discrete Mathematics*, 4:39–49, 1979. doi:10.1016/S0167-5060(08)70817-3.

11 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization – Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop*, volume 2570 of *Lecture Notes in Computer Science*, pages 11–26. Springer, 2001. doi:10.1007/3-540-36478-1_2.

12 Jack Edmonds. Matroid partition. In *50 Years of Integer Programming 1958-2008 – From the Early Years to the State-of-the-Art*, pages 199–217. Springer, 2010. doi:10.1007/978-3-540-68279-0_7.

13 András Frank. A weighted matroid intersection algorithm. *J. Algorithms*, 2(4):328–336, 1981. doi:10.1016/0196-6774(81)90032-8.

14 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. doi:10.15807/jorsj.38.124.

15 Harold N. Gabow and Ying Xu. Efficient theoretic and practical algorithms for linear matroid intersection problems. *J. Comput. Syst. Sci.*, 53(1):129–147, 1996. doi:10.1006/jcss.1996.0054.

16 Chien-Chung Huang, Naonori Kakimura, and Naoyuki Kamiyama. Exact and approximation algorithms for weighted matroid intersection. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 430–444. SIAM, 2016. doi:10.1137/1.9781611974331.ch32.

17 Eugene L. Lawler. Matroid intersection algorithms. *Math. Program.*, 9(1):31–56, 1975. doi:10.1007/BF01681329.

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

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

20 James B. Orlin and Ravindra K. Ahuja. New scaling algorithms for the assignment and minimum mean cycle problems. *Math. Program.*, 54:41–56, 1992. doi:10.1007/BF01586040.

21 Christopher Price. Combinatorial algorithms for submodular function minimization and related problems. Master's thesis, University of Waterloo, 2015. URL: http://hdl.handle.net/10012/9356.

22 Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

23 Maiko Shigeno and Satoru Iwata. A dual approximation approach to weighted matroid intersection. *Oper. Res. Lett.*, 18(3):153–156, 1995. doi:10.1016/0167-6377(95)00047-X.

A Omitted Proofs

A.1 Proofs of Lemmas in Section 3.1

For self-containedness, we include the proof that the two bases obtained in the weight adjustment phase have a large intersection by Shigeno and Iwata [23] here.

▶ **Lemma 11** ([23]). *Let S_1 and S_2 be obtained from the procedure described in Lemma 6. Then, $|S_1 \cap S_2| \geq \left(1 - \frac{O(1)}{k}\right) r$.*

Proof. Let $p(S)$ denote $\sum_{x \in S} p(x)$. Observe that an element is never moved to $S_2 \setminus S_1$ during weight adjustments, and therefore we have $p(S_2 \setminus S_1) = 0$ and $p(S_1 \setminus S_2) = p(S_1) - p(S_2)$. Recall that S' is a common basis such that $(w^{2\epsilon}, S')$ is a 2ϵ-solution. Since $p(x)$ equals the number of adjustments of $w_2^{\epsilon}(x)$ and each such adjustment is preceded by an adjustment of $w_1^{\epsilon}(x)$, we have

$$p(x) \cdot \epsilon = w_2^{\epsilon}(x) - (w(x) - w_1^{2\epsilon}(x)) \leq w_1^{2\epsilon}(x) - w_1^{\epsilon}(x)$$

for each $x \in V$. Thus,

$$\begin{aligned}
p(S_1 \setminus S_2) \cdot \epsilon &= (p(S_1) - p(S_2)) \cdot \epsilon \\
&\leq \left(w_1^{2\epsilon}(S_1) - w_1^{\epsilon}(S_1)\right) - \left(w_2^{\epsilon}(S_2) - w(S_2) + w_1^{2\epsilon}(S_2)\right) \\
&\overset{\text{(a)}}{\leq} w_1^{2\epsilon}(S_1) - w_1^{\epsilon}(S') - w_2^{\epsilon}(S') + w(S_2) - w_1^{2\epsilon}(S_2) \\
&\overset{\text{(b)}}{\leq} w_1^{2\epsilon}(S_1) - w(S') + w(S_2) - w_1^{2\epsilon}(S_2),
\end{aligned}$$

where (a) is because S_i is w_i^{ϵ}-maximum for each $i \in \{1, 2\}$ and (b) is because $w(S') \leq w^{\epsilon}(S')$ as w^{ϵ} is an ϵ-splitting. Since $(w^{2\epsilon}, S')$ is a 2ϵ-solution, $w_2^{2\epsilon}(S) - 2\epsilon r \leq w(S) - w_1^{2\epsilon}(S) \leq w_2^{2\epsilon}(S)$ holds for each basis S. This combined with the fact that S' is $w_i^{2\epsilon}$-maximum for each $i \in \{1, 2\}$ implies

$$p(S_1 \setminus S_2) \cdot \epsilon \leq 2\epsilon r - w_2^{2\epsilon}(S') + w_2^{2\epsilon}(S_2) \leq 2\epsilon r \implies p(S_1 \setminus S_2) \leq 2r.$$

When the algorithm terminates, we have $p(x) = k$ for all $x \in S_1 \setminus S_2$, implying

$$p(S_1 \setminus S_2) = |S_1 \setminus S_2| \cdot k \leq 2r \implies |S_1 \setminus S_2| \leq \frac{2r}{k}.$$

As a result,

$$|S_1 \cap S_2| = r - |S_1 \setminus S_2| \geq \left(1 - \frac{O(1)}{k}\right) r.$$ ◀

A.2 Proofs of Lemmas in Section 3.2

In this section, we prove the properties of the exchange graphs. The proofs for the more generalized auxiliary graph given by Fujishige and Zhang can be found in [14].

To prove Lemma 7, it would be more convenient to refer to the following definition of a directed bipartite graph based on exchange relationships, which is heavily used in unweighted matroid intersection algorithms. For $S \in \mathcal{I}_1 \cap \mathcal{I}_2$, let $\widetilde{G}_S = (V \cup \{s,t\}, \widetilde{E})$ with $s,t \notin V$ denote the directed graph with $\widetilde{E} = \widetilde{E}_1 \cup \widetilde{E}_2 \cup \widetilde{E}_s \cup \widetilde{E}_t$, where

$$\widetilde{E}_1 = \{(x,y) \mid x \in S, y \notin S, \text{ and } (S \setminus \{x\}) \cup \{y\} \in \mathcal{I}_1\},$$
$$\widetilde{E}_2 = \{(y,x) \mid x \in S, y \notin S, \text{ and } (S \setminus \{x\}) \cup \{y\} \in \mathcal{I}_2\},$$
$$\widetilde{E}_s = \{(s,x) \mid S \cup \{x\} \in \mathcal{I}_1\}, \text{ and}$$
$$\widetilde{E}_t = \{(x,t) \mid S \cup \{x\} \in \mathcal{I}_2\}.$$

▶ **Lemma 12** ([17]). *$\widetilde{G}_S$ for $|S| < r$ admits an st-path.*

We will use the existence of an st-path in $\widetilde{G}_{\widetilde{S}}$ to prove that such a path exists in $G_{w^\epsilon,S}$, for $\widetilde{S} = S_1 \cap S_2$. The following claims certify that $\widetilde{G}_{\widetilde{S}}$ and $G_{w^\epsilon,S}$ are almost the same.

▷ Claim 13. Let $\mathcal{M} = (V,\mathcal{I})$ be a matroid, $S \subseteq S' \in \mathcal{I}$, $x \in S$, and $y \notin S'$ such that $(S \setminus \{x\}) \cup \{y\} \in \mathcal{I}$ but $S \cup \{y\} \notin \mathcal{I}$, then $(S' \setminus \{x\}) \cup \{y\} \in \mathcal{I}$.

Proof. Let C be the unique circuit in $S \cup \{y\}$. Since $C \subseteq S' \cup \{y\}$ and $S' \cup \{y\}$ has only one circuit, C is the unique circuit in $S' \cup \{y\}$ as well. Moreover, $(S \setminus \{x\}) \cup \{y\} \in \mathcal{I}$ if and only if $x \in C$ and therefore $(S' \setminus \{x\}) \cup \{y\} \in \mathcal{I}$. ◁

▷ Claim 14. Let $\mathcal{M} = (V,\mathcal{I})$ be a matroid, $S \subseteq S' \in \mathcal{I}$ where S' is a basis of $\mathcal{M}$, and $x \notin S'$ such that $S \cup \{x\} \in \mathcal{I}$. Then, there exists a $y \in S' \setminus S$ such that $(S' \setminus \{y\}) \cup \{x\} \in \mathcal{I}$.

Proof. Let S'' be an arbitrary basis of $\mathcal{M}$ that contains $S \cup \{x\}$. Since $x \in S'' \setminus S'$, by the strong exchange property (see, e.g., [22, Theorem 39.6]) of bases, there exists a $y \in S' \setminus S'' \subseteq S' \setminus S$ such that $(S' \setminus \{y\}) \cup \{x\} \in \mathcal{I}$, completing the proof. ◁

We are now ready to prove Lemma 7.

Proof of Lemma 7. Let $\widetilde{P} = \{s, v_1, \ldots, v_k, t\}$ be the shortest st-path in $\widetilde{G}_{\widetilde{S}}$ for $\widetilde{S} = S_1 \cap S_2$. The existence of such a path is guaranteed by Lemma 12. We have $\widetilde{S} \cup \{v_i\} \notin \mathcal{I}_1$ and $\widetilde{S} \cup \{v_i\} \notin \mathcal{I}_2$ for each $1 < i < k$ since $\widetilde{P}$ is the shortest path. For an odd $1 \le i < k$, we have $v_i \notin S$ and $v_{i+1} \in S$. If $v_i \notin S_2 \setminus S_1$, then by Claim 13, we have $(v_i, v_{i+1}) \in E(G_{w^\epsilon,S})$. Similarly, for an even $1 \le i < k$, if $v_{i+1} \notin S_1 \setminus S_2$, then we have $(v_i, v_{i+1}) \in E(G_{w^\epsilon,S})$. Suppose that $v_1 \notin S_1 \setminus S_2$, then by Claim 14, we can find a $v_0 \in S_1 \setminus S_2$ such that $(S_1 \setminus \{v_0\}) \cup \{v_1\} \in \mathcal{I}_1$. Similarly, if $v_k \notin S_2 \setminus S_1$, then we can find a $v_{k+1} \in S_2 \setminus S_1$ such that $(S_2 \setminus \{v_{k+1}\}) \cup \{v_k\} \in \mathcal{I}_2$. Therefore, without loss of generality, we may assume that there exists the last vertex $v_i \in S_1 \setminus S_2$ and the first vertex $v_j \in S_2 \setminus S_1$ after v_i. Now, for each $i < k < j$, we have $v_k \notin (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. Therefore, $(v_k, v_{k+1}) \in E(G_{w^\epsilon,S})$ holds for each $i \le k < j$, and we obtain an st-path in $G_{w^\epsilon,S}$ as $P = \{s, v_i, \ldots, v_j, t\}$. This concludes the proof. ◀

Finally, to prove Lemma 8, we need the following results.

▶ **Lemma 15** ([21, Proposition 2.4.1]). *Given a matroid $\mathcal{M} = (V,\mathcal{I})$ and an $S \in \mathcal{I}$. Suppose that $(a_1, \ldots, a_p) \subseteq V \setminus S$ and $(b_1, \ldots, b_p) \subseteq S$ are two sequences satisfying the following conditions:*

1. $(S \setminus \{b_i\}) \cup \{a_i\} \in \mathcal{I}$ *for each* $1 \leq i \leq p$ *and*
2. $(S \setminus \{b_j\}) \cup \{a_i\} \notin \mathcal{I}$ *for each* $1 \leq j < i \leq p$.

Then, $(S \setminus \{b_1, \ldots, b_p\}) \cup \{a_1, \ldots, a_p\} \in \mathcal{I}$ *holds.*

▶ **Lemma 16** ([21, Lemma 2.4.2]). *Let* $\mathcal{M}$*,* S*,* $(a_1, \ldots, a_p)$*, and* $(b_1, \ldots, b_p)$ *be the same as in Lemma 15. Let* $S' = (S \setminus \{b_1, \ldots, b_p\}) \cup \{a_1, \ldots, a_p\}$*. For* $x \in S'$ *and* $y \in V \setminus S'$*, if* $(S' \setminus \{x\}) \cup \{y\} \in \mathcal{I}$ *but either* $y \in S$ *or* $(S \setminus \{x\}) \cup \{y\} \notin \mathcal{I}$*, then there exists* $1 \leq \ell \leq k \leq p$ *such that* $(S \setminus \{x\}) \cup \{a_k\} \in \mathcal{I}$ *and either* $b_\ell = y$ *or* $(S \setminus \{b_\ell\}) \cup \{y\} \in \mathcal{I}$*.*

In essence, Lemma 15 captures the validity of an augmentation while Lemma 16 models the condition in which new exchange relationships emerge in the augmented independent set. The following three claims imply Lemma 8. Recall that $P = \{s, v_1, \ldots, v_k, t\}$ is the shortest st-path with the least number of edges and $d(x)$ is the sx-distance in $G_{w^\epsilon,S}$.

▷ Claim 17. $|\widehat{S}_1 \cap \widehat{S}_2| > |S_1 \cap S_2|$ holds.

Proof. If $k = 2$, then either $v_1 \in \widehat{S}_1 \cap \widehat{S}_2$ or $v_k \in \widehat{S}_1 \cap \widehat{S}_2$ must hold, depending on whether $(v_1, v_2) \in E_1$ or $(v_1, v_2) \in E_2$, and the claim trivially holds in this case. Thus, in the following, we assume that $k > 2$. Since P is the shortest, we may assume that $v_i \in (S_1 \cap S_2) \cup (\overline{S_1} \cap \overline{S_2})$ holds for each $1 < i < k$. Also, for $1 < i < k-1$, if $v_i \in S_1 \cap S_2$, then v_{i+1} must be in $\overline{S_1} \cap \overline{S_2}$ due to the way $G_{w^\epsilon,S}$ is constructed. Similarly, if $v_i \in \overline{S_1} \cap \overline{S_2}$, then we must have $v_{i+1} \in S_1 \cap S_2$. Let $P_{\text{mid}} = \{v_2, \ldots, v_{k-1}\}$, $I = S_1 \cap S_2$, and $O = \overline{S_1} \cap \overline{S_2}$. Clearly, we have

$$|\widehat{S}_1 \cap \widehat{S}_2| - |S_1 \cap S_2| = |P_{\text{mid}} \cap O| - |P_{\text{mid}} \cap I| + [\![v_1 \in \widehat{S}_1 \cap \widehat{S}_2]\!] + [\![v_k \in \widehat{S}_1 \cap \widehat{S}_2]\!]. \tag{3}$$

We prove the claim by considering the following four possible cases.

k is even and $v_2 \in I$. We have $(v_1, v_2) \in E_2$, $(v_{k-1}, v_k) \in E_2$, and $|P_{\text{mid}} \cap I| = |P_{\text{mid}} \cap O|$. Also, $v_1 \in \text{tail}(P \cap E_2)$ and therefore $v_1 \in \widehat{S}_1 \cap \widehat{S}_2$.

k is even and $v_2 \in O$. We have $(v_1, v_2) \in E_1$, $(v_{k-1}, v_k) \in E_1$, and $|P_{\text{mid}} \cap I| = |P_{\text{mid}} \cap O|$. Also, $v_k \in \text{head}(P \cap E_1)$ and therefore $v_k \in \widehat{S}_1 \cap \widehat{S}_2$.

k is odd and $v_2 \in I$. We have $(v_1, v_2) \in E_2$, $(v_{k-1}, v_k) \in E_1$, and $|P_{\text{mid}} \cap I| = |P_{\text{mid}} \cap O| + 1$. Also, $v_1 \in \text{tail}(P \cap E_2)$, $v_k \in \text{head}(P \cap E_1)$ and therefore $v_1, v_k \in \widehat{S}_1 \cap \widehat{S}_2$.

k is odd and $v_2 \in O$. We have $(v_1, v_2) \in E_1$, $(v_{k-1}, v_k) \in E_2$, and $|P_{\text{mid}} \cap I| = |P_{\text{mid}} \cap O| - 1$.

In all cases, we have $|\widehat{S}_1 \cap \widehat{S}_2| > |S_1 \cap S_2|$ via Equation (3), concluding the proof. ◁

We prove the following claims for $i = 1$. The proofs for $i = 2$ follow analogously.

▷ Claim 18. $\widehat{S}_i \in \mathcal{I}_i$ holds for each $i \in \{1, 2\}$.

Proof. Let $P_1 = P \cap E_1 = \{(b_1, a_1), (b_2, a_2), \ldots (b_p, a_p)\}$, where $(S_1 \setminus \{b_i\}) \cup \{a_i\} \in \mathcal{I}_1$ holds for each $1 \leq i \leq p$. Since P is the shortest path,

$$d(b_i) + w_1^\epsilon(b_i) - w_1^\epsilon(a_i) = d(a_i) \implies d(b_i) + w_1^\epsilon(b_i) = d(a_i) + w_1^\epsilon(a_i)$$

holds for each i. Reorder P_1 so that $d(b_1) + w_1^\epsilon(b_1) \leq d(b_2) + w_1^\epsilon(b_2) \leq \cdots \leq d(b_p) + w_1^\epsilon(b_p)$. Moreover, if $d(b_i) + w_1^\epsilon(b_i) = d(b_j) + w_1^\epsilon(b_j)$ for some i, j, then (b_i, a_i) precedes (b_j, a_j) in P_1 if and only if (b_i, a_i) precedes (b_j, a_j) in P. It follows that for each $1 \leq j < i \leq p$, it holds that $(S_1 \setminus \{b_j\}) \cup \{a_i\} \notin \mathcal{I}_1$ since otherwise we would have

$$d(b_j) + w_1^\epsilon(b_j) - w_1^\epsilon(a_i) \geq d(a_i) \implies d(b_j) + w_1^\epsilon(b_j) \geq d(a_i) + w_1^\epsilon(a_i). \tag{4}$$

Because $j < i$, (4) must take equality, but this would contradict with the fact the P has the least number of edges since the edge (b_j, a_i) "jumps" over vertices $a_j, b_{j+1}, \ldots, b_i$ in P and has the same weight as the subpath $b_j, a_j, \ldots, b_i, a_i$. As such, by Lemma 15, the claim is proved. ◁

▷ Claim 19. $\widehat{S}_i$ is $\widehat{w}_i^\epsilon$-maximum for each $i \in \{1,2\}$.

Proof. Let $P_1 = P \cap E_1 = \{(b_1, a_1), \ldots, (b_p, a_p)\}$ be ordered the same way as in the proof of Claim 18. It suffices to show that $\widehat{w}_1^\epsilon(x) \geq \widehat{w}_1^\epsilon(y)$ holds for each $x \in \widehat{S}_1$ and $y \notin \widehat{S}_1$ with $(\widehat{S}_1 \setminus \{x\}) \cup \{y\} \in \mathcal{I}_1$. Consider the following two cases.

1. $(S_1 \setminus \{x\}) \cup \{y\} \in \mathcal{I}_1$: Since $(x, y) \in E_1$, it follows that

$$d(x) + w_1^\epsilon(x) - w_1^\epsilon(y) \geq d(y) \implies \widehat{w}_1^\epsilon(x) = d(x) + w_1^\epsilon(x) \geq d(y) + w_1^\epsilon(y) = \widehat{w}_1^\epsilon(y).$$

2. $(S_1 \setminus \{x\}) \cup \{y\} \notin \mathcal{I}_1$: By Lemma 16, there exists $1 \leq \ell \leq k \leq p$ such that (1) $(S_1 \setminus \{x\}) \cup \{a_k\} \in \mathcal{I}_1$ and either (2.1) $b_\ell = y$ or (2.2) $(S_1 \setminus \{b_\ell\}) \cup \{y\} \in \mathcal{I}_1$. (1) implies that $\widehat{w}_1^\epsilon(x) \geq \widehat{w}_1^\epsilon(a_k)$. If (2.1) holds, then $\widehat{w}_1^\epsilon(x) \geq \widehat{w}_1^\epsilon(a_k) \geq \widehat{w}_1^\epsilon(b_\ell) = \widehat{w}_1^\epsilon(y)$. If (2.2) holds, then $\widehat{w}_1^\epsilon(x) \geq \widehat{w}_1^\epsilon(a_k) \geq \widehat{w}_1^\epsilon(b_\ell) \geq \widehat{w}_1^\epsilon(y)$.

The claim is proved. ◁