Determinant Maximization via Matroid Intersection Algorithms

Adam Brown
School of Mathematics
Georgia Institute of Technology
Atlanta, USA
ajmbrown@gatech.edu

Aditi Laddha
College of Computing
Georgia Institute of Technology
Atlanta, USA
aladdha6@gatech.edu

Madhusudhan Pittu
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
mpittu@andrew.cmu.edu

Mohit Singh
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, USA
msingh94@gatech.edu

Prasad Tetali
Mathematical Sciences
Carnegie Mellon University
Pittsburgh, USA
ptetali@cmu.edu

Abstract—Determinant maximization problem gives a general framework that models problems arising in as diverse fields as statistics [1], convex geometry [2], fair allocations [3], combinatorics [4], spectral graph theory [5], network design, and random processes [6]. In an instance of a determinant maximization problem, we are given a collection of vectors $U = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$, and a goal is to pick a subset $S \subset U$ of given vectors to maximize the determinant of the matrix $\sum_{i \in S} v_i v_i^T$. Additionally, the set $S$ of picked vectors must satisfy additional combinatorial constraints such as cardinality constraint ($|S| \leq k$) or matroid constraint ($S$ is a basis of a matroid defined on the vectors).

In this paper, we give a polynomial-time deterministic algorithm that returns a $\tilde{O}(r)$-approximation for any matroid of rank $r \leq d$. This improves previous results that give $e^{O(r)}$-approximation algorithms relying on $e^{O(r)}$-approximate estimation algorithms [4], [7]–[9] for any $r \leq d$. All previous results use convex relaxations and their relationship to stable polynomials and strongly log-concave polynomials or non-convex relaxations for the problem [10]. In contrast, our algorithm builds on combinatorial algorithms for matroid intersection, which iteratively improve any existing solution by finding an alternating negative cycle in the exchange graph defined by the matroids. While the $\det(.)$ function is not linear, we show that taking appropriate linear approximations at each iteration suffice to give the improved approximation algorithm.

Index Terms—Computations on discrete structures, Linear approximation, Combinatorial algorithms.

I. INTRODUCTION

Determinant maximization problem gives a general framework that models problems arising in as diverse fields as statistics [1], convex geometry [2], fair allocations [3], combinatorics [4], spectral graph theory [5], network design and random processes [6]. In an instance of a determinant maximization problem, we are given a collection of vectors $U = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$, and a goal is to pick a subset $S \subset U$ of given vectors to maximize the determinant of the matrix $\sum_{i \in S} v_i v_i^T$. Additionally, the set $S$ of picked vectors must satisfy additional combinatorial constraints such as cardinality constraint ($|S| \leq k$) or matroid constraint ($S$ is a basis of a matroid defined on the vectors).

Apart from its modeling strength, from a technical perspective, determinant maximization has brought interesting connections between areas such as combinatorial optimization, convex analysis, geometry of polynomials, graph sparsification and complexity of permanent and other counting problems [2], [3], [8], [11].

Applications. Observe that when the number of vectors picked is exactly $d$, the objective is precisely the square of the volume of the parallelepiped spanned by the selected vectors. The problem of finding the largest volume parallelepiped in a collection of given vectors has been studied [2], [12], [13] for over three decades. Another interesting application is the determinantal point processes [6], where a probability distribution over subsets of vectors is defined. The probability of selecting a subset is defined to be proportional to the squared volume of the parallelepiped defined by them. These distributions display nice properties of negative correlation. Finding sets with the largest probability mass is exactly the determinant maximization problem. We refer the reader to [5] for applications in experimental design and to [3] for application to fair allocations.

The computational complexity of the determinant maximization depends crucially on the combinatorial set family which constrains the set of feasible collection of vectors. The simplest constraint being the cardinality constraint, wherein the number of vectors is fixed, has been the most widely studied variant. For this, a variety of methods including convex programming based methods [11], [13]–[15], combinatorial methods - such as local search and greedy [2], [16], [17] -
as well as close relationships to graph sparsification [11] have been exploited to obtain efficient approximation algorithms with very good guarantees. Overall, these results give a very clear understanding of the computational complexity of the problem.

The more general case when the combinatorial constraints are defined by a matroid constraint has recently received extensive focus [3], [4], [7]-[9]. This is especially interesting since some of the applications are naturally modeled as matroid constraints, in particular, as partition constraints. Unfortunately, there is a big gap between estimation algorithms and approximation algorithms in this case! Indeed, one can approximately estimate the value of an optimal solution with a good guarantee, however, finding such a solution is much more challenging, leading to an exponential loss in the approximation factor. For example, even for the special case of the partition matroid, there is an \( e^d \)-approximate estimation algorithm but the best known approximation algorithms return a solution with an approximation factor of \( e^{O(d^2)} \), an exponential blow-up\(^1\). A fundamental reason for this gap is the reliance on the relationship between convex programming relaxations for the problem and the theory of stable polynomials and its generalization to strongly log-concave polynomials. Unfortunately, these methods are inherently non-algorithmic and do not give a simple way to obtain efficient algorithms with the same guarantees that match the estimation bounds. A second approach [10] relies on non-convex optimization that gives algorithmic results for special cases of matroids with guarantees depending on the size of the ground set.

A. Our Results and Contributions

In this work, we introduce new combinatorial methods for deterministic maximization under a matroid constraint and give an \( O(d^{O(d)}) \)-deterministic approximation algorithm. While previous works have used a convex programming approach and the theory of stable polynomials, our approach builds on the classical matroid intersection algorithm. Our first result focuses on the case when the rank of the matroid is exactly \( d \), i.e., the output solution will contain precisely \( d \) vectors.

**Theorem I.1.** There is a polynomial-time algorithm which, given a collection of vectors \( v_1, \ldots, v_n \in \mathbb{R}^d \) and a matroid \( M = ([n], I) \) of rank \( d \), returns a set \( S \in I \) such that

\[
\det \left( \sum_{i \in S} v_i v_i^\top \right) = \Omega \left( \frac{1}{n^{O(d)}} \right) \max_{S' \in I} \det \left( \sum_{i \in S'} v_i v_i^\top \right).
\]

Our results improve the \( e^{O(d)} \)-approximation algorithm which relies on the \( e^{O(d)} \)-estimation algorithm [4], [8], [9]. Our algorithm builds on the matroid intersection algorithm and is an iterative algorithm that starts at any feasible solution and improves the objective in each step. To maintain feasibility in the matroid constraint, each step of the algorithm is an exchange of multiple elements as found by an alternating cycle of an appropriately defined exchange graph.

**Result for \( r \leq d \).** We also generalize the result when the rank \( r \) of the matroid is at most \( d \). Observe that the solution matrix \( \sum_{i \in S} v_i v_i^\top \) is a \( d \times d \) matrix of rank at most \( r \) and, therefore, the appropriate objective to consider is the product of its largest \( r \) eigenvalues, or equivalently, the elementary symmetric function of order \( r \) of its eigenvalues. Let \( \text{sym}_r(M) \) be the \( r \)-th elementary symmetric function of the eigenvalues of the \( d \times d \) matrix \( M \). Thus, our objective is to maximize \( \text{sym}_r \left( \sum_{i \in S} v_i v_i^\top \right) \).

**Theorem I.2.** There is a polynomial-time algorithm which, given a collection of vectors \( v_1, \ldots, v_n \in \mathbb{R}^d \) and a matroid \( M = ([n], I) \) of rank \( r \leq d \), returns a set \( S \in I \) such that

\[
\text{sym}_r \left( \sum_{i \in S} v_i v_i^\top \right) = \Omega \left( \frac{1}{r^{O(r)}} \right) \max_{S' \in I} \text{sym}_r \left( \sum_{i \in S'} v_i v_i^\top \right).
\]

This again improves the best bound of \( e^{O(r^2)} \)-approximation algorithm based on \( e^{O(r)} \)-approximate estimation algorithms. The proof of Theorem I.2 is presented in the arxiv version of the paper [18].

**Technical Overview.** For intuition, let \( vol(S) \) denote the volume of the parallellepiped spanned by the vectors in \( S \). Then \( vol(S)^2 = \det \left( \sum_{i \in S} v_i v_i^\top \right) \), for any \( S \subseteq U \) with \( |S| = d \), so we can think of \( vol(S) \) as an equivalent objective function. First, observe that the feasibility problem of checking whether there is a set \( S \in I \) such that \( vol(S) > 0 \) can be reduced to matroid intersection. Indeed, the feasibility problem is equivalent to checking if there is a common basis of the matroid \( M \) and the linear matroid defined by the vectors \( \{v_1, \ldots, v_n\} \). Since we aim to maximize \( vol(S) \) over all independent sets \( S \), a natural approach is to use the weighted matroid intersection algorithm. Unfortunately, our weights are not linear, i.e., \( vol(S) \) does not equal \( \sum_{i \in S} w_i \) or log-linear \( \prod_{i \in S} w_i \) for some weights \( w \) on the vectors. Nonetheless, the matroid intersection algorithm forms the backbone of our approach.

**Overview of Matroid Intersection.** Before we describe our algorithm, let us review a classical algorithm to find a maximum weight common basis of two matroids. Given \( U = \{1, \ldots, n\} \), a weight function \( w : U \rightarrow \mathbb{R} \) and two matroids \( M_1 = (U, I_1) \) and \( M_2 = (U, I_2) \), the goal is to find a common basis \( S \) of maximum weight \( w(S) := \sum_{i \in S} w_i \). We assume that there exists a common basis of the two matroids. Consider the following simple algorithm that also introduces some of the basic ingredients necessary for our algorithm. The algorithm will take as an input a common basis \( S \) and either certify that \( S \) is a maximum weight common basis or return a new common basis \( \hat{S} \) of higher weight. To explain the algorithm, we recall the important concept of the exchange graph. Given the set \( S \), we construct a directed bipartite graph \( G(S) \) with bipartitions given by \( U \setminus S \) and \( S \). For any \( u \in U \setminus S \) and \( v \in S \), \( G(S) \) contains an arc from \( u \) to \( v \) if \( S - v + u \) is a basis in \( M_2 \) and an arc from \( v \) to \( u \) if \( S - v + u \) is a basis in \( M_1 \). For convenience, we use
$S - v + u$ to refer to the set $(S \cup \{u\}) \setminus \{v\}$. Moreover, give each vertex $u \in U \setminus S$ a weight $-w_u$ and each vertex $v \in S$ a weight of $w_v$. A nice fact from matroid theory is that $S$ is a maximum weight basis if and only if there is no negative weight cycle in this directed graph (Chapter 41, Theorem 41.5 [19]). Moreover, if $C$ is a directed negative weight cycle with minimum hops, then $S\Delta C$ forms a common basis of the two matroids whose weight is strictly larger than the weight of $S$. Thus, the algorithm finds a maximum weight basis by iteratively finding a negative weight cycle in such an exchange graph.

With the above algorithm as a guiding tool, we describe our algorithm. The two matroids are precisely the constraint matroid $M$ and the linear matroid defined over the vectors. A first challenge is that the objective function $\text{vol}(S)^2 = \det \left( \sum_{i \in S} e_i e_i^\top \right)$ is not linear. Thus it is not possible to define the vertex weights as was done above. But a natural function to work with instead is the function $\log \text{vol}(S)$, which is known to be submodular. While we do not use submodularity explicitly, our algorithm takes linear approximations of this function at each iteration while searching for improvements as in the matroid intersection algorithm. We use the geometric relationship between $\text{vol}$ and $\log$ closely. The first new ingredient in our algorithm is to introduce arc weights rather than vertex weights in the exchange graph $G(S)$. Indeed for the forward arcs $(u, v)$ for $u \not\in S$ and $v \in S$ that correspond to the linear matroid, we introduce a weight of $- \log \frac{\text{vol}(S-v+u)}{\text{vol}(S)}$. We also introduce a weight of 0 for the backward arcs, which correspond to the arcs for the constraint matroid $M$. The crucial observation is the following interpretation of the weight $\log \frac{\text{vol}(S)}{\text{vol}(S-v+u)}$: write the vector $u \not\in S$ in the basis $S$, i.e. $u = \sum_{v \in S} a_{uv} e_v$, for some $a_{uv} \in \mathbb{R}$ for each $v \in S$. Then $\frac{\text{vol}(S)}{\text{vol}(S-v+u)} = |a_{uv}|$ (See Lemma II.2). Such relationships between the ratio of volumes and coefficients in expressing the vectors in basis given by $S$ play an important role.

Our first Lemma shows that if the volume of the current solution is much smaller than the optimal solution, then there must be a cycle such that the sum of weights of the arcs on the cycle is significantly negative.

**Lemma I.3** (Determinate to Cycle). Let $S$ be a basis of $M$ and $OPT$ be the basis of $M$ maximizing $\text{vol}(OPT)$. If $\text{vol}(OPT) \geq e^{5d \log d} \cdot \text{vol}(S)$, then there exists a directed cycle $C$ of $2\ell$ hops for some $\ell > 0$ in the exchange graph $G(S)$ such that

$$\prod_{(u,v) \in C, u \not\in S, v \in S} |a_{uv}| \geq 2(\ell)!^3 = : f(\ell).$$

We call such a cycle an $f$-violating cycle. Observe that such a cycle can be found as a negative weight 2-hop cycle when weights are updated to $w_T(u,v) = \frac{1}{2} \log f(\ell) - \log |a_{uv}|$ for a forward arc $(u,v)$ where $u \not\in S$ and $v \in S$. The lemma relies on the following observation. Abusing notation slightly, let $T$ and $S$ be matrices whose columns are the vectors in $OPT$ and $S$, respectively. Writing each vector in $OPT$ in the basis given by $S$ we obtain $T = SA^\top$ for some matrix $A$. The condition in the lemma implies that $\det(A) \geq e^{5d \log d}$. Also observe that the weight of any $(u,v)$ where $u \in OPT$ and $v \in S$ is exactly $-\log |a_{uv}|$ where $a_{uv}$ is the $(u,v)$th entry in $A$. Combining these facts, we can show there exists a cycle satisfying the conditions of the lemma.

The next step in the algorithm is to find an $f$-violating cycle and then update the solution to $T = S\Delta C$. Again, we relate the change in objective $\text{vol}(T)$ to the coefficients. While the entries $|a_{uv}|$ of the cycle are large, the objective of the new solution $T$ depends not only on the weight of the edges of the cycle but the weight of all arcs between all vertices in $C \setminus S$ and $C \cap S$. Indeed, consider a square matrix $B$ with rows and columns indexed by $C \setminus S$ and $S \cap C$ respectively with entry $(u,v)$ as $a_{uv}$. Recall $a_{uv}$ is the coefficient of vector $v$ when $u$ is expressed in basis $S$. Then $\text{vol}(T) = |\det(B)| \cdot \text{vol}(S)$ (Lemma II.5). Thus it remains to lower bound the determinant of $B$. The entries on the diagonal of the matrix $B$ exactly correspond to entries that define the weights of the forward arcs on the cycle $C$. Thus Lemma I.3 implies that the product of the diagonal entries of $B$ is large. In the next lemma, we show that if the cycle $C$ is the minimum hop $f$-violating cycle, then we can in fact lower bound the determinant of the matrix.

**Lemma I.4** (Cycle to Determinant). If $C$ is a minimum hop $f$-violating cycle in the exchange graph $G(S)$, then $\text{vol}(S \Delta C) \geq 2 \cdot \text{vol}(S)$. Moreover, $S \Delta C$ is a basis of $M$.

This lemma crucially uses the fact that $C$ is a minimum hop $f$-violating cycle as in the case for matroid intersection algorithms. Indeed, off-diagonal entries of the matrix $B$ correspond to arcs that form chords of the cycle $C$. The minimality of $C$ allows us to show upper bounds on all the off-diagonal entries of the matrix $B$. A technical calculation then allows us to lower bound the determinant.

**B. Related Work**

**Determinant Maximization under Cardinality Constraints.** Determinant maximization under a cardinality constraint have been studied widely [2], [11], [13]-[16]. Currently, the best approximation algorithm for the case $r \leq d$ is an $e^r$-approximation due to Nikolov [14] and for $r > d$, there is an $e^d$-approximation [15]. It turns out that the problem gets significantly easier when $r >> d$, and there is a $(1+\epsilon)^d$-approximation when $r \geq d + \frac{d}{4}$ [11], [16], [17]. These results use local search methods and are closely related to the algorithm discussed in this paper, as the cycle improving algorithm will always find a 2-cycle when the matroid is defined by the cardinality constraint.

**Determinant Maximization under Matroid Constraints.** As mentioned earlier, determinant maximization under a matroid constraint is considerably challenging and the bounds also depend on the rank $r$ of the constraint matroid. There are $e^{O(r)}$ estimation algorithms when $r \leq d$ [7], [8], [20] and a $\min\{e^{O(r)}, O(d^2d)}$ estimation algorithm when $r \geq d$ [9]. The output of these algorithms is a random feasible set whose
objective is at least \( \min \{ e^{O(r)}, O(d^{O(d)}) \} \) of the objective of a convex programming relaxation, in expectation. Since the approximation guarantees are exponential, it can happen that the output set has objective zero almost always. To convert them into deterministic algorithms (or randomized algorithms that work with high probability), additional loss in approximation factor is incurred. These results imply an \( e^{O(d^2)} \)-approximation algorithm when \( r \leq d \), and an \( O(d^{O(d)}) \)-approximation algorithm [9] for \( r \geq d \). Approximation algorithms are also known where the approximation factor is exponential in the size of the ground set for special classes of matroids [10].

**Nash Social Welfare and its generalizations.** A special case of the determinant maximization problem is the Nash Social Welfare problem [21]. In the Nash Social Welfare problem, we are given \( m \) items and \( d \) players and there is a valuation function \( v_i : 2^{|m|} \rightarrow \mathbb{R}_+ \) for each player \( i \in [d] \) that specifies value obtained by a player when given a bundle of items. The goal is to find an assignment of items to players to maximize the geometric mean of the valuations of each of the players. When the valuation functions are additive, the problem becomes a special case of the determinant maximization and this connection can be utilized to give an \( \epsilon \)-approximation algorithm [3]. Other methods including rounding algorithms [22], [23] as well as primal-dual methods [24] have been utilized to obtain improved bounds. The problem has been studied when the valuation function is more general [25]–[28] and a constant-factor approximation is known when the valuation function is submodular [29].

**Other Spectral Objectives.** While we focus on the determinant objective, the problem is also interesting when considering other spectral objectives including minimizing the trace or the maximum eigenvalue of the matrix \( \left( \sum_{i \in S} v_i v_i^\top \right)^{-1} \). These problems have been studied for the cardinality constraint [11], [30]. For the case of partition matroid, the problem of maximizing the minimum eigenvalue is closely related to the Kadison-Singer problem [31].

**II. ALGORITHM FOR PARTITION MATROID**

We first show the algorithm and the analysis for a partition matroid with rank \( d \). This allows us to show the basic ideas without going into the details of matroid theory. The generalizations to general matroid are quite standard. We detail them in Section III.

Consider a partition matroid \( \mathcal{M} \) with \( d \) parts \( \mathcal{P}_1, \ldots, \mathcal{P}_d \) where \( \mathcal{P}_i \) contains \( n_i \) vectors \( v_{1i}, \ldots, v_{ni} \in \mathbb{R}^d \). Our goal is to find a set \( S \) which provides a good approximation to the objective

\[
\max \left\{ \det \left( \sum_{i \in S} v_i v_i^\top \right) : |S| = d, |S \cap \mathcal{P}_i| = 1 \ \forall i \right\}.
\]

Let \( OPT \) denote the optimal solution set. The following theorem is a specialization of Theorem I.1 to the case of partition matroid.

\[
Theorem II.1. \text{Given a partition matroid } \mathcal{M} \text{ with } d \text{ parts, let } OPT \text{ be the optimal solution to the determinant maximization problem on } \mathcal{M}. \text{ Then, there is a polynomial-time deterministic algorithm that outputs a feasible set } S \subseteq \mathcal{M} \text{ such that}
\]

\[
\det \left( \sum_{i \in S} v_i v_i^\top \right) \geq e^{-10d \log(d)} \cdot \det \left( \sum_{i \in OPT} v_i v_i^\top \right).
\]

**A. Algorithm**

We begin by formally defining the exchange graph, the different weight functions, and then the algorithm which helps establish Theorem I.1 for the case of partition matroids.

**Definition 1 (Exchange Graph).** Formally, for a subset of vectors \( S = \{v_1, v_2, \ldots, v_d\} \) with \( v_i \in \mathcal{P}_i \) for all \( i \), we define the exchange graph of \( S \), denoted by \( G(S) \), as a bipartite graph, where the right-hand side consists of vectors in \( S \), i.e., \( R = \{v_1, v_2, \ldots, v_d\} \) and the left-hand side consists of all the vectors \( L = \bigcup_{i=1}^d \mathcal{P}_i \setminus \{v_i\} \) (See Figure 1). Each \( v_i \in R \) has an edge to every \( u \in \mathcal{P}_i \setminus \{v_i\} \), i.e., all the vectors in the same part as \( v_i \). The vertices on the left-hand side have forward edges to every vertex in \( S \).

We define a family of weight functions on the exchange graph. The basic weight function will be denoted by \( w_0 : A(G(S)) \rightarrow \mathbb{R} \) and, in addition, we define weight functions \( w_i \) for each \( 1 \leq i \leq d \). To define these weights, we use the function \( f : [d] \rightarrow \mathbb{Z}_+ \) with \( f(i) = 2(i!)^3 \) for each \( i > 0 \).
Definition 2 (Weight functions on the Exchange graph). We first define weight function \( w_0 \). All the backward arcs from any \( v_i \in S \) to every \( u_j \in P \setminus \{ v_i \} \), have weight 0. For \( u_j \in L \), let \( u_j = \sum_{i=1}^{d} a_{ij} \cdot v_i \) be expression for \( u_j \) in the basis \( S \) where \( a_{ij} \in \mathbb{R} \) for each \( i \). Then the forward arc \( (u_j, v_i) \) has weight \( w_0(u_j, v_i) := -\log(\langle a_{ij} \rangle) \) for each \( i \in [d] \) and each \( u_j \in L \).

Now we define the weight function \( w_1 \) on the arcs for any \( 1 \leq \ell \leq d \). All backward arcs still have weight 0 but every forward edge \( (u, v) \) has weight \( w_1(u, v) := \frac{\log(f(\ell))}{\log(f(1))} + w_0(u, v) \).

The following lemma gives the intuition behind the weight function \( w_0 \) defined above. It shows that the weight on arc \( (u_j, v_i) \) exactly measures the change in the objective when we replace element \( v_i \) with \( u_j \) in \( S \). The proof appears in the arxiv version [18].

Lemma II.2. Let \( S \) be a solution with \( \text{vol}(S) > 0 \) and \( u \notin S \). Then for any \( v \in S \), we have \( w_0(u, v) = -\log \frac{\text{vol}(S+u-v)}{\text{vol}(S)} \).

Proof: Let \( S = \{ v_1, \ldots, v_d \} \) so that \( v = v_1 \) and write \( u = \sum_{i=1}^{d} a_i v_i \). We can also write \( v = v^+ + \sum_{i=2}^{d} a_i v_i \) where \( v^+ \) is orthogonal to \( S \setminus \{ v \} \). Then \( a = a_1 v_1^+ + \sum_{i=2}^{d} (a_i b_i + a_i) v_i \). For \( X \subseteq \mathbb{R}^d \) with \( |X| = k \leq d \), let \( \text{vol}(X) \) denote the \( k \)-dimensional volume of the parallelepiped spanned by \( X \). Then

\[
\text{vol}(S) = \text{vol}(S-v) \cdot \| v^+ \|,
\]

while

\[
\text{vol}(S+u-v) = \text{vol}(S-v) \cdot |a_1| \| v^+ \|,
\]

since the change in volume from adding a single new vector is proportional to the length of the component of that vector which is orthogonal to our current set. Thus

\[
-\log \frac{\text{vol}(S+u-v)}{\text{vol}(S)} = -\log \frac{\text{vol}(S-v) \cdot |a_1| \| v^+ \|}{\text{vol}(S-v) \cdot \| v^+ \|} = -\log |a_1|.
\]

While we will be specific about which weight function to use, if it is not specified, then we refer to the weight function \( w_0 \).

Definition 3 (Cycle Weight). The weight of a cycle \( C \) in \( G(S) \) is defined as \( w_0(C) = \sum_{e \in C} w_0(e) \).

Observe that the weight of a cycle depends only on the weight of the forward edges as backward edges have a weight 0.

We want to move from the current set \( S \) to a set with higher volume by exchanging on cycles in \( G(S) \). But we want to exchange only on cycles that satisfy certain nice properties. For this purpose, we define \( f \)-Violating Cycles and Minimal \( f \)-Violating Cycles. The algorithm will always exchange on a Minimal \( f \)-Violating Cycle.

Definition 4 (\( f \)-Violating Cycle). A cycle in \( G(S) \) is called an \( f \)-violating cycle if

\[
w_0(C) < -\log f(|C|/2),
\]

where \( |C| \) is the number of arcs in \( C \).

We have the following simple observation regarding \( f \)-violating cycle.

Observation 1. If \( C \) is a \( f \)-violating cycle then

\[
\prod_{(u,v) \in C} |a_{uv}| > 2 \left( \left( \frac{|C|}{2} \right)! \right)^3.
\]

Proof: If \( C \) is \( f \)-violating then \( \ell(C) = -\log f(|C|/2) \), where \( \ell(C) \) is the sum of the \( w_0 \) edge weights in \( C \), and \( f(|C|/2) = 2(|C|/2)! \). Note that \(|C \cap R| = |C \cap L| = |C|/2 \), so \( f(|C|/2) = 2(|C \cap R|)! \). By expanding \( \ell(C) \) we see that

\[
\ell(C) = \sum_{(u,v) \in C} w_0(u,v) = -\log \prod_{(u,v) \in C} |a_{uv}|.
\]

Since \( \ell(C) < -\log f(|C|/2) \), we can take the exponential to remove the logarithms and attain the desired inequality.

Definition 5 (Minimal \( f \)-Violating Cycle). A cycle \( C \) in \( G(S) \) is called a minimal \( f \)-violating cycle if

- \( C \) is an \( f \)-violating cycle, and
- for all cycles \( C' \) such that \( V(C') \subset V(C) \), \( C' \) is not an \( f \)-violating cycle.

Note that finding an \( f \)-violating cycle with \( 2i \) arcs is equivalent to finding a negative cycle with \( 2i \) arcs in \( G(S) \) with weights \( w_i \). We use the following simple algorithm to find a minimal \( f \)-violating cycle in \( G(S) \) (if one exists), where we iterate on the number of arcs in the cycle.

Algorithm 1 Finding minimal \( f \)-violating cycle

for \( i = 1, \ldots, d \) do

if there is a negative cycle \( C \) with exactly \( 2i \) arcs in \( G(S) \) with weight function \( w_i \) then

Return \( C \)

end if

end for

The following lemma is immediate.

Lemma II.3. Algorithm 1 finds the minimal \( f \)-violating cycle in \( G(S) \).

Proof: In the \( i \)-th iteration of Algorithm 1 we determine if there is a negative cycle in \( G(S) \) with weights \( w_i \) and \( 2i \) hops, as follows. For each vertex of \( G(S) \), we start an instance of Bellman-Ford (See Chapter 8, Section 8.3, [19]) with that vertex as the root, and proceed for \( 2i \) iterations. For source \( u \), after \( 2i \) iterations, we check whether the distance from \( u \) to \( u \) is negative. If so, we have found a negative cycle with at most \( 2i \) hops. Note that for weights \( w_i \), any negative cycle with at most \( 2i \) hops is an \( f \)-Violating cycle. Since the \( (i-1) \)-th iteration of the algorithm ensured that there are no \( f \)-Violating
cycles with at most $2(i - 1)$ hops, a negative cycle in the $i$-th iteration (if any) must have exactly $2i$ hops.

Suppose there is an $f$-violating cycle $C$ in $G(S)$, so that $\ell = |C|/2$. Then, with weight $w_f$, the total weight of the cycle $C$ is

$$w_\ell(C) = \sum_{(u,v) \in C} w_f(u,v) = \sum_{(u,v) \in C} \log(f(\ell))/\ell + w_0(u,v) = \log(f(\ell)) + w_0(C).$$

Since $C$ is $f$-violating we know that $\log f(\ell) < -w_0(C)$, so the above calculation shows that $C$ has negative total weight with weights $w_f$. This guarantees that Algorithm 1 will return an $f$-violating cycle whenever one exists.

Now suppose that $C$ is the cycle returned by Algorithm 1 and we must show that $C$ is minimal $f$-violating. Let $C'$ be another cycle such that $V(C') \subset V(C)$. Then $C'$ has fewer hops than $C$, but it was not returned in iteration $|C'|/2$, so we know that $C'$ must not be $f$-violating. Thus $C$ is indeed minimal.

After finding a minimal $f$-violating cycle, $C$, we modify the current set $S$ to $S \Delta C$ and repeat. Observe that $S \Delta C$ is always a feasible set as it will pick exactly one element from each part. The main idea is that if $\text{vol}(S)$ is small compared to $\text{vol}(OPT)$, i.e., $\text{vol}(S) < \text{vol}(OPT) \cdot e^{-\Omega(d \log(d))}$, then there is always an $f$-violating cycle in $G(S)$ (see Lemma II.4). Moreover, if $C$ is a minimal $f$-violating cycle, then $\text{vol}(S \Delta C) \geq 2 \cdot \text{vol}(S)$ (see Lemma II.6). If we initialize $S$ to any solution with non-zero determinant, then the ratio $\text{vol}(OPT)/\text{vol}(S)$ is at most $2^{1+\sigma}$ where $\sigma$ is the encoding length of our problem input (Chapter 3, Theorem 3.2 [32]). This implies that we need only modify the set $S$ polynomially many times before $\text{vol}(S)$ becomes greater than $\text{vol}(OPT) \cdot e^{-\Theta(d \log(d))}$, which gives Theorem II.1. Such an initialization can be obtained by finding a basis of $\mathbb{R}^d$ that picks exactly one vector from each part. As discussed above, this problem can be solved by the matroid intersection algorithm over the partition matroid and the linear matroid defined by the vectors.

**Algorithm 2** Algorithm to find an approximation to OPT

1. $S \leftarrow \text{set with } |S| = d, |S \cap \mathcal{P}_i| = 1$ for all $i$, and $\text{vol}(S) > 0$.
2. while there exists an $f$-violating cycle in $G(S)$ do
   1. $C = \text{minimal } f$-violating cycle in $G(S)$
   2. $S = S \Delta C$
3. end while

Return $S$

**Lemma II.4.** For any set $S$ with $|S| = d$ and $\text{vol}(S) > 0$, if $\text{vol}(S) < \text{vol}(OPT) \cdot e^{-5d \log(d)}$, then there exists an $f$-violating cycle in $G(S)$.

**Proof:** Index the sets $OPT = \{u_1, u_2, \ldots, u_d\}$ and $S = \{v_1, v_2, \ldots, v_d\}$ such that $u_i, v_i \in \mathcal{P}_i$ for all $i \in [d]$. Observe that $(v_i, u_i)$ is an arc in the exchange graph for each $i$ since $u_i$ and $v_i$ belong to the same part.

Abusing notation slightly, let $T$ and $S$ be matrices whose columns are the vectors in $OPT$ and $S$, respectively. Let $A$ be the coefficient matrix of $T$ w.r.t. $S$, i.e., $T = SA^T$. Then

$$\text{vol}(OPT)^2 = \det(TT^T) = \det(SA^TAS^T) = \det(SS^T) \cdot |\det(A)|^2.$$  

Let $X = OPT \setminus S, Y = S \setminus OPT$, and $|X| = |Y| = k$. Without loss of generality, let $Y = \{v_1, \ldots, v_k\}$ and $X = \{u_1, \ldots, u_k\}$. Then $A = [A_k \ A']$, where $A_k$ is the sub-matrix of $A$ corresponding to rows in $X$ and columns in $Y$. Then $\det(A) = \det(A_k).

As per the hypothesis in the lemma, we have $\det(SS^T) < \det(TT^T) \cdot e^{-10d \log(d)}$. Therefore,

$$|\det(A_k)| > e^{5d \log(d)} \geq e^{5k \log(k)}.$$  

(1)

By the Leibniz formula,

$$\det(A_k) = \sum_{\sigma \in S_k} \text{sign}(\sigma) \prod_{i=1}^k a_{\sigma(i)}.$$  

Taking absolute values gives

$$|\det(A_k)| \leq \sum_{\sigma \in S_k} |a_{\sigma(i)}|.$$  

Since $|S_k| = k! \leq e^{k \log(k)}$, there exists a permutation $\sigma \in S_k$ such that

$$\prod_{i=1}^k |a_{\sigma(i)}| > |\det(A_k)| \cdot e^{-k \log(k)} \geq e^{4k \log(k)}.$$  

(2)

Let the disjoint cycle decomposition of this $\sigma$ be $\sigma = \{C_1, C_2, \ldots, C_t\}$. Then each $C_j$ corresponds to a unique cycle in $G(S)$ with $2|C_j|$ hops by considering the forward arcs $(u_i, v_{\sigma(i)})$ for each $i$ on the cycle and the backward arcs $(v_i, u_i)$ for each $i$ in $C_j$. We claim that at least one of these cycles is an $f$-violating cycle. If not, then by the definition of $f$-violating cycles, we have $\prod_{i \in C_j} |a_{\sigma(i)}| \leq 2(|C_j|!)^3$. Multiplying over all cycles in $\sigma$ gives

$$\prod_{i=1}^k |a_{\sigma(i)}| = \prod_{j=1}^t \prod_{i \in C_j} |a_{\sigma(i)}| \leq 2^t (|C_j|!)^3 < 2^t k!^3 < e^{4k \log(k)},$$  

where the second last inequality follows from $\sum_{j=1}^t |C_j| = k$. This contradicts eq. (2), so $G(S)$ must contain an $f$-violating cycle.

The requirement in Lemma II.4 that $\text{vol}(S) < \text{vol}(OPT) \cdot e^{-5d \log(d)}$ is tight, up to the coefficient in the exponent. Consider the case where $d$ is a power of two (or more)

\footnote{Given $u_i \neq v_i$}
generally, any \(d\) for which a Hadamard matrix of order \(d\) is known to exist, \(S = \{e_1, \ldots, e_d\}\) consists of the standard basis vectors, and \(L = H = \{h_1, \ldots, h_d\}\) consists of the columns of the \(d \times d\) Hadamard matrix. The entries of \(H\) are all \(\pm 1\), and \(h_i^T h_j = 0\) for \(i \neq j\). Then \(\text{vol}(S) = 1\), and the optimal solution is \(\text{OPT} = H\), which has objective value

\[
\text{vol}(H) = \prod_{i=1}^{d} ||h_i|| = d^{d/2} = e^{\frac{d}{2} \log(d)} \cdot \text{vol}(S),
\]

since the vectors in \(H\) are orthogonal. Meanwhile, the exchange matrix in this case is \(A = H^T\). Since all the entries of \(A\) are \(\pm 1\), we know that the product of the entries along any cycle will have an absolute value of 1. Thus, we cannot find an \(f\)-violating cycle in the same way, despite the fact that \(\text{vol}(S) \leq \text{vol}(\text{OPT}) \cdot e^{-\frac{1}{2} \log(d)}\).

### B. Cycle Exchange and Determinant

Now we show that exchanging on a minimal \(f\)-violating cycle \(C\) increases the objective of the output set by at least a factor of two. The proof relies on two technical lemmas. First, observe that the arc weights given by \(w_0(u, v)\) are exactly how much the objective will change if switch from the solution \(S\) to \(S + u - v\) in the solution. But switching on a cycle will switch multiple elements at the same time. Since our function \(\text{vol}()\) (or more appropriately \(\log \text{vol}()\)) is not additive, it is not clear what the change in the objective. The following lemma characterizes exactly how the objective changes when we switch a large set.

Consider our current solution \(S\). Let \(C\) be the minimal cycle found and \(\ell = |C|/2\). Let \(X = C \cap L\) and \(Y = C \cap S\). Thus the output set \(T = (S \cup X) \setminus Y\). We will also abuse notation to let \(X, Y\) and \(S\) represent the matrices whose columns are the vectors in their respective sets. Note that \(S\) is \(d \times d\) while both \(X\) and \(Y\) are \(d \times \ell\). Observe that \(\text{vol}(S)^2 = \det(SS^T)\) and \(\text{vol}(T)^2 = \det(TT^T) = \det(SS^T + XX^T - YY^T)\). Crucially, we show that the matrix consisting of coefficients \(a_{uv}\) that define the weights on the arcs of the exchange graph for \(u \in X\) and \(v \in Y\) also defines the change in objective value.

**Lemma II.5.** Let \(S\) be a basis, let \(X\) and \(Y\) be sets with \(|X| = |Y| = \ell\) and \(Y \subseteq S\). Let \(A\) be the \(\ell \times d\) matrix of coefficients so that \(X = SA^T\), and let \(A_C\) be the \(\ell \times \ell\) submatrix of only the coefficients corresponding to columns in \(Y\). If \(T = (S \cup X) \setminus Y\) then \(\text{vol}(T)^2 = \text{vol}(S)^2 \cdot \det(A_C A_C^T)\).

**Proof:** We will abuse notation slightly to let \(S, X, Y\) also denote the matrices with columns from their respective sets. Order the columns of \(S\) so that \(Y\) makes up the first \(\ell\) columns of \(S\). Let \(A'\) be the \((d-\ell) \times \ell\) submatrix of \(A\) consisting of the remaining columns not already in \(A_C\). Then

\[
T = S \begin{bmatrix} A_C & A' \\ 0 & I_{d-\ell} \end{bmatrix}^T,
\]

which implies that

\[
\det(T) = \det(S) \cdot \det(A_C).
\]

Without loss of generality, let \(C = (v_0 \rightarrow u_1 \rightarrow v_1 \rightarrow u_2 \rightarrow v_2 \rightarrow \ldots \rightarrow u_\ell \rightarrow v_0)\) so that \(X = \{u_1, \ldots, u_\ell\}\) and \(Y = \{v_1, \ldots, v_{\ell-1}, v_0\}\), and order the columns of \(A_C\) accordingly so that the \(\ell\)-th column corresponds to \(v_0\). Observe that diagonal entries of the \(A_C\) correspond to coefficient of \(v_i\) when expressing \(u_i\) in basis of \(S\) and thus equals \(a_{ii}\). \(C\) being \(f\)-violating implies that the product of the diagonal entries \(\prod_{i=1}^\ell |a_{ii}| > f(\ell)\). To show that the volume of \(T\) is large, we need to show \(|\det(A_C)|\) is large. To this end, we utilize crucially that \(C\) is the minimal \(f\)-violating cycle. Observe that the off-diagonal entries \(a_{ij}\) exactly correspond to the weight on chords of the cycle. Since each chord introduces a cycle with smaller number of arcs, by minimality we know that it is not \(f\)-violating. This allows us to prove upper bounds on the off-diagonal entries of the matrix \(A_C\). Finally, a careful argument allows us to give a lower bound on the determinant of any matrix with such bounds on the off-diagonal entries. We now expand on the above outline below.

**Lemma II.6.** If \(C\) is a minimal \(f\)-violating cycle in \(G(S)\), then \(\text{vol}(S\Delta C) \geq 2 \cdot \text{vol}(S)\).

**Proof:** Let \(C = (v_0 \rightarrow u_1 \rightarrow v_1 \rightarrow u_2 \rightarrow v_2 \rightarrow \ldots \rightarrow u_\ell \rightarrow v_0)\) where \(v_i, u_{i+1}\) belong to the same part and \(v_i \in S\) (See Figure 2).

By the Lemma II.5, we know that

\[
\text{vol}(S\Delta C) = \det(A_C A_C^T)^{1/2} \cdot \text{vol}(S) = |\det(A_C)| \cdot \text{vol}(S).
\]

We will index the entries of \(A_C\) according to the indices of \(u_i\) and \(v_j\) where the last column corresponds to \(v_0\). Since \(C\) has \(2\ell\) hops, \(A_C\) is an \(\ell \times \ell\) matrix.

We now bound each entry of the matrix \(A_C\) in terms of the its diagonal entries, \(a_{ii}\) for \(i = 1, \ldots, n\). We show upper bounds on the absolute value of each entry as a function of the diagonal entries. Consider the \(i, j\)-th entry of \(A_C\). For

![Fig. 2. The cycle C](image-url)
is a cycle with $f$-violating. Therefore,

$$|a_{i,j}| \cdot \prod_{s=1}^{i-1} |a_{s,s}| \cdot \prod_{s=j+1}^{\ell} |a_{s,s}| < f(\ell - j + i). \tag{4}$$

Since $C$ is an $f$-violating cycle, we also have

$$\prod_{s=1}^{\ell} |a_{s,s}| > f(\ell). \tag{5}$$

Combining (4) and (5) gives

$$|a_{i,j}| < \frac{f(\ell - j + i)}{f(\ell)} \cdot \prod_{s=i}^{j} |a_{s,s}|.$$

Let $B_\ell$ be the matrix obtained by applying the following operations to $A_C$

- Multiply the last column by $a_{1,1}$ and for $j < \ell$, divide the $j$-th column by $\prod_{s=2}^{j} a_{s,s}$
- Divide the first row by $a_{1,1}$ and for $i > 1$, multiply the $i$-th row by $\prod_{s=2}^{i-1} a_{s,s}$
- Divide the last column by $f(\ell)$ and, if needed, flip the sign of the last column so that $a_{\ell,\ell} > 0$.

Then $|\det(A_C)| = f(\ell) \cdot |\det(B_\ell)|$, and $B_\ell$ satisfies the following properties:

- $b_{i,i} = 1$ for all $i \in [\ell - 1], b_{\ell,\ell} \geq 1$,
- $|b_{i,j}| \leq f(i-j)$ for all $j < i \leq \ell$, and
- $|b_{i,j}| \leq f(\ell - j + i)/f(\ell)$ for all $i < j \leq \ell$.

For $\ell \geq 2$, we have the following claim:

**Claim II.1.** $\det(B_2) \geq 0.75$ and $\det(B_\ell) > 0.1$ for all $\ell \geq 3$.

With this claim in hand, it implies that $|\det(A_C)| > 0.1 \cdot f(\ell) > 2$ for all $\ell \geq 3$. For $\ell = 2$, $\det(B_2) \geq 0.75$, and $\det(A_C) \geq 0.75 \cdot f(2) > 2$. Therefore, $\mathrm{vol}(S\Delta C) \geq |\det(A_C)| \cdot \mathrm{vol}(S) \geq 2 \cdot \mathrm{vol}(S)$.

**Proof of Claim II.1:** Consider the following process on $B_\ell$:

**Algorithm 3 Gaussian Elimination Process (Column Operations)**

```plaintext
for s = 1, \ldots, \ell do
  for j = s + 1, \ldots, \ell do
    b_{i,j} = b_{i,j} - b_{i,s} \cdot \frac{b_{s,j}}{b_{s,s}}
  end for
end for
```

Note that $\det(B_2) \geq 0.75, \det(B_3) \geq 0.73,$ and $\det(B_4) \geq 0.83$ (detailed calculations can be found in the arxiv version [18]). From hereafter, we will assume that $\ell \geq 5$.

The output of the Algorithm 3 is a lower triangular matrix. Let $b_{i,j}(s)$ denote the value of $b_{i,j}$ before the $s$-th iteration of the outer loop of Gaussian Elimination. For example, $b_{i,j}(1) = b_{i,j}$ for all $i, j$.

For any $i < j$, $b_{i,j}$ becomes 0 at the end of the $i$-th iteration of the outer loop of the algorithm, and does not change after that. So, the final value of $b_{i,j}$, before it becomes 0, is $b_{i,j}(i)$.
Similarly, for $i \geq j$, the value of $b_{i,j}$ does not change after the $(j - 1)$-th iteration of the outer loop, and therefore the final value of $b_{i,j}$, i.e., $b_{i,j}(\ell) = b_{i,j}(j)$.

Since this process does not change the determinant of $B_{\ell}$, we have $\det(B_{\ell}) = \prod_{j=1}^{\ell} b_{j,j}(j)$. By Lemma II.7, $b_{j,j}(j) > 1 - 0.92/\ell$ for $j < \ell$ and $b_{\ell,\ell}(\ell) > 0.303$. Therefore,

$$\det(B_{\ell}) = \prod_{j=1}^{\ell} b_{j,j}(j) \geq \left(1 - \frac{0.92}{\ell}\right)^{\ell-1} \cdot 0.303.$$  

The function $\left(1 - \frac{0.92}{\ell}\right)^{\ell-1}$ is a decreasing function of $\ell$, but has a horizontal asymptote at $\sim 0.39$. Thus, $\left(1 - \frac{0.92}{\ell}\right)^{\ell-1} \geq 0.39$ and this gives

$$\det(B_{\ell}) > 0.39 \times 0.303 > 0.1.$$  

Lemma II.7. For $\ell \geq 5$, the final values of entries of $B_{\ell}$ after Algorithm 3 are bounded as follows:

1. $|b_{j,j}(j)| < \left(\frac{\ell}{j}\right) \cdot f(i - j) \text{ for } 1 < j < i,$
2. $|b_{i,j}(i)| < 1.5 \cdot f(\ell - j + 1)/f(\ell) \text{ for } i < j < \ell,$
3. $|b_{\ell,\ell}(\ell)| < 2.84 \cdot f(\ell)/f(\ell) \text{ for } i < \ell,$
4. $b_{j,j}(j) > 1 - \frac{0.92}{\ell} \text{ for all } j < \ell,$
5. $b_{\ell,\ell}(\ell) > 0.303$.

Proof: We will prove the lemma by induction on $j$, the column index. Note that Algorithm 3 does not change the values of the first column of $B_{\ell}$, and it also does not change the values of the first row of $B_{\ell}$ before they become 0. So, the bounds are trivially true for the first column and the first row.

For $i \geq j$,

$$b_{i,j}(j) = b_{i,j}(1) - \sum_{s=1}^{j-1} b_{s,i}(s) \cdot \frac{b_{s,j}(s)}{b_{s,s}(s)}.$$  

Taking absolute values gives

$$|b_{i,j}(j) - b_{i,j}(1)| \leq \sum_{s=1}^{j-1} |b_{s,i}(s)| \cdot \left|\frac{b_{s,j}(s)}{b_{s,s}(s)}\right|.  \quad(7)$$  

The induction hypothesis implies that for all $s < j$, $|b_{s,s}(s)| < \left(\frac{j}{s}\right) \cdot f(i - s)$, $|b_{s,j}(s)| < 1.5 \cdot f(\ell - j + 1)/f(\ell)$, and $b_{s,s}(s) > 1 - 0.92/\ell \geq 0.816$ (since $\ell \geq 5$). Plugging these bounds in (7), we get

$$|b_{i,j}(j) - b_{i,j}(1)| < \frac{1.5}{0.816} \cdot \sum_{s=1}^{j-1} \left(\frac{i}{s}\right) \cdot f(i - s) \cdot \frac{f(\ell - j + s)}{f(\ell)}. \quad(8)$$  

Note that

$$\frac{f(i - s) \cdot f(\ell - j + s)}{f(i - j) \cdot f(\ell)} = \frac{(i - s)!}{(i - j)!} \cdot \frac{((\ell - j) + s)!}{(\ell)!}.$$  

For any $1 \leq s \leq j - 1$, $\frac{((\ell - j)!}{(\ell - j + 1)!} \leq \frac{(i - j + 1)!}{(i)!}$. Therefore,

$$\frac{f(i - s) \cdot f(\ell - j + s)}{f(i - j) \cdot f(\ell)} \leq \left(\frac{\ell + i - j}{i - s}\right)^2.$$  

Plugging this in (8) gives

$$\frac{|b_{i,j}(j) - b_{i,j}(1)|}{f(i - j)} \leq f(i - j) \left(1.84 \cdot \frac{(i + j)!}{\ell^2} \cdot \frac{\ell - i + j}{(\ell - i + j)!} \sum_{s=1}^{\ell - j} \left(\frac{(\ell - j)!}{s!}\right)\right)$$

$$= f(i - j) \left(1.84 \cdot \frac{(i + j)!}{\ell^2} \cdot \frac{\ell - i + j}{(\ell - i + j)!} \sum_{s=1}^{\ell - j} \left(\frac{(\ell - j)!}{s!}\right)\right). \quad(9)$$  

For positive integers $a, b, x$ with $x \leq a \leq b$,

$$\left(\frac{a}{x}\right) + \left(\frac{a + 1}{x}\right) + \ldots + \left(\frac{b}{x}\right) = \left(\frac{b + 1}{x + 1}\right) - \left(\frac{a}{x + 1}\right). \quad(10)$$  

Using (10) with $a = \ell - j + 1$, $b = \ell - 1$, and $x = \ell - j$ gives

$$\sum_{j=1}^{\ell - j + 1} \left(\frac{\ell - j + s}{s}\right) \leq \left(\frac{\ell - j + 1}{\ell - j}\right)$$

and from (9),

$$\frac{|b_{i,j}(j) - b_{i,j}(1)|}{f(i - j)} \leq f(i - j) \left(1.84 \cdot \frac{(i + j)!}{\ell^2} \cdot \frac{\ell - i + j}{(\ell - i + j)!} \sum_{s=1}^{\ell - j} \left(\frac{(\ell - j)!}{s!}\right)\right)$$

$$= f(i - j) \left(1.84 \cdot \frac{i}{j} \cdot \frac{(i - j + 1)!}{\ell^2(j - 1)!} \sum_{s=1}^{j - 1} \left(\frac{(\ell - j)!}{s!}\right)\right). \quad(11)$$  

Using (10) with $a = \ell - j + 1$, $b = \ell - 1$, and $x = \ell - j$ gives

$$\sum_{j=1}^{\ell - j + 1} \left(\frac{\ell - j + s}{s}\right) \leq \left(\frac{\ell - j + 1}{\ell - j}\right)$$

and from (9),

$$\frac{|b_{i,j}(j) - b_{i,j}(1)|}{f(i - j)} \leq f(i - j) \left(1.84 \cdot \frac{i}{j} \cdot \frac{(\ell - j + 1)!}{\ell^2(j - 1)!} \sum_{s=1}^{j - 1} \left(\frac{(\ell - j)!}{s!}\right)\right). \quad(12)$$  

Since $(\ell - j + 1)!j$ is maximized at $j = (\ell + 1)/2$, we have

$$\frac{(\ell + 1)!}{2 \ell} \leq 0.36$$

for any $\ell \geq 5$.

Plugging this in (12) gives

$$|b_{i,j}(j)| \leq |b_{i,j}(j)| + f(i - j) \cdot 0.6624 \cdot \left(\frac{i}{j}\right)$$

$$\leq f(i - j) \left(1 + 0.6624 \cdot \left(\frac{i}{j}\right)\right).$$  

263
Now we will restrict ourselves to the case when \( i > j \).

For \( i = 2 \), \( j \) can only be 1 and this corresponds to an entry in the first column for which the bounds are trivially true. So, we only need to consider \( i \geq 3 \). Since \( 1 \leq j < i \), we have \( \binom{i}{j} \geq i \). Furthermore, since \( \ell \geq 5 \) and \( i \geq 3 \), we have \( 1 \leq 0.3376 \cdot i < 0.3376 \binom{i}{j} \). This gives

\[
|b_{i,j}(j)| \leq f(i-j) \left( 0.3376 \cdot \binom{i}{j} + 0.6624 \cdot \binom{i}{j} \right) 
\leq f(i-j) \cdot \binom{i}{j}.
\]

This concludes the proof of part 1.

For \( j > i \), we have

\[
|b_{i,j}(i) - b_{i,j}(1)| \leq \sum_{s=1}^{i-1} \frac{|b_{i,s}(s)|}{|b_{i,s}(s)|}.
\]  \hspace{1cm} (13)

By the induction hypothesis, \( |b_{i,j}(s)| < 1.5 \cdot f(\ell - j + s)/f(\ell) \) and \( b_{i,s}(s) > 1 - 0.92/\ell \geq 0.816 \). Plugging these bounds in (13), we get

\[
|b_{i,j}(i) - b_{i,j}(1)| \leq 1.5 \cdot \sum_{s=1}^{i-1} \frac{f(\ell - j + s)}{f(\ell)} \cdot \binom{i}{j} \cdot f(i-s).
\]  \hspace{1cm} (14)

Note that

\[
\frac{f(\ell - j + s) \cdot f(i-s)}{f(\ell - j + i)} = 2(\ell - j + s)!^3 \cdot ((i-s)!)(\ell - j + i)! = 2 \cdot \frac{1}{(\ell - j + s)!^3}. \]

For any \( 1 \leq s \leq i-1, \frac{1}{(\ell - j + s)!^3} \leq \frac{1}{(\ell - j + i)!^3} \). Therefore,

\[
\frac{f(\ell - j + i-s) \cdot f(s)}{f(\ell - j + i)} \leq \frac{1}{(\ell - j + s)!^3} \cdot \frac{2}{(\ell - j + i)!^3}.
\]

Plugging this in (14) gives

\[
|b_{i,j}(i) - b_{i,j}(1)| \leq \frac{f(\ell - j + i)}{f(\ell)} \cdot \frac{3.68}{(\ell - j + i)!^3} \left( \sum_{s=1}^{i-1} \frac{1}{(\ell - j + s)!^3} \right) \leq \frac{f(\ell - j + i)}{f(\ell)} \cdot \frac{3.68 \cdot i!}{(\ell - j + i)!^3}. \]

Using (10) again, we get

\[
|b_{i,j}(i) - b_{i,j}(1)| \leq \frac{f(\ell - j + i)}{f(\ell)} \cdot \frac{3.68 \cdot i!}{(\ell - j + i)!^3} \leq \frac{f(\ell - j + i)}{f(\ell)} \cdot \frac{3.68 \cdot i!}{(\ell - j + i)^2 \cdot (\ell - j + i)!}.
\]

The function \( \frac{1}{(\ell - j + i)^2} \) is maximized at \( i = \ell - j \). So for any \( j < \ell \), we have

\[
|b_{i,j}(i) - b_{i,j}(1)| \leq \frac{f(\ell - j + i)}{f(\ell)} \cdot \frac{3.68 \cdot i!}{(\ell - j + i)^2 \cdot (\ell - j + i)!} \leq \frac{0.5}{f(\ell)}.
\]

Using the fact that \( |b_{i,j}(1)| \leq f(\ell - j + i)/f(\ell) \), we have \( |b_{i,j}(i)| \leq 1.5 \cdot f(\ell - j+i)/f(\ell) \) for \( i \leq j < \ell \).

For \( j = \ell \) and \( i \geq 2 \), (15) gives

\[
|b_{i,\ell}(i) - b_{i,\ell}(1)| \leq \frac{f(\ell)}{f(\ell)} \cdot \frac{3.68 \cdot i!}{1.84 \cdot f(\ell)} \leq 1.84 \cdot f(\ell) / f(\ell),
\]

and therefore \( |b_{i,\ell}(i)| \leq 2.84 \cdot f(\ell) / f(\ell) \). This concludes the proof of parts 2 and 3.

For \( i = j = \ell \), using (11), we get

\[
|b_{\ell,\ell}(\ell)| \leq \frac{1.84 \cdot f(\ell - 1)}{2\ell^2} \leq \frac{1.84}{2\ell^2} \leq 0.92 / \ell.
\]

For \( i = j = \ell \), by (8) and the induction hypothesis,

\[
|b_{\ell,\ell}(\ell) - b_{\ell,\ell}(1)| \leq \frac{2.84}{1 - 0.92/\ell} \sum_{s=1}^{\ell-1} \frac{f(\ell-s)}{f(\ell)}.
\]

Following the proof outline of (11) gives \( |b_{\ell,\ell}(\ell) - b_{\ell,\ell}(1)| \leq 2.84 \cdot \frac{1}{1 - 0.92/\ell} \leq 0.97 \leq 0.97. \) Since \( b_{\ell,\ell}(1) = 1 \), we have \( b_{\ell,\ell}(1) \geq 0.97 \geq 0.303 \).

III. UPDATE STEP FOR GENERAL MATROIDS

Consider the case when \( \mathcal{M} = ([n], \mathcal{I}) \) is a general matroid of rank \( d \). When we exchange on a cycle \( C \) and update \( S \leftarrow S \Delta C \), the resulting set is guaranteed to be independent in the linear matroid because of the determinant bounds in Lemma II.6, but it is not clear that it would be independent in the general constraint matroid, \( \mathcal{M} \), when \( \mathcal{M} \) is not a partition matroid. However, by exchanging on a minimal \( f \)-violating cycle in our algorithm, we can make the same guarantee.

In this section, we prove the existence of an \( f \)-violating cycle for any matroid \( \mathcal{M} \) with rank \( d \) when the current basis \( S \) is sufficiently smaller in volume than the optimal solution \( OPT \). We also prove that exchanging on a minimal \( f \)-violating cycle preserves independence in \( \mathcal{M} \).

**Theorem III.1.** For any basis \( S \) with \( |S| = d \) and \( \text{vol}(S) > 0 \), if \( \text{vol}(S) < \text{vol}(OPT) \cdot e^{-5d \log(d)} \), then there exists an \( f \)-violating cycle in \( G(S) \).
Proof: Since $S$ and $OPT$ are independent and $|S| = |OPT|$, there exists a perfect matching between $OPT \setminus S$ and $S \setminus OPT$ using the backward arcs in $G(S)$ (Chapter 39, Corollary 39.12a, [19]). Let $X = OPT \setminus S$, $Y = S \setminus OPT$, and $|X| = |Y| = k$. Without loss of generality, let $Y = \{v_1, \ldots, v_k\}$ and $X = \{u_1, \ldots, u_k\}$ such that $(v_i \to u_i)$ is an arc in $G(S)$ for all $i \in [k]$.

Let $T$ and $S$ be matrices whose columns are the vectors in $OPT$ and $S$, respectively. Let $A$ be the coefficient matrix of $T$ w.r.t. $S$, i.e., $T = S A^T$. Then $A = \begin{bmatrix} A_k & A' \\ 0 & I_{d-k} \end{bmatrix}$, where $A_k$ is the sub-matrix of $A$ corresponding to rows in $X$ and columns in $Y$. Then by the same proof as in II.4, there exists a permutation $\sigma \in S_k$ such that

$$\prod_{i=1}^k |a_{i\sigma(i)}| > |\det(A)| \cdot e^{-k \log(k)} \geq e^{4k \log(k)}.$$ (16)

Let the cycle decomposition of $\sigma$ be $\sigma = \{C_1, C_2, \ldots, C_l\}$ where $C_i = (i_1 \to i_2 \to \cdots i_j \to i_1)$. Since there is an edge from $v_{i_j} \to u_{i_1}$ for all $j$, every cyclic permutation $C_i$ corresponds to a cycle $(u_{i_1} \to v_{i_2} \to u_{i_3} \to \cdots u_{i_j} \to v_{i_1} \to u_{i_2} \to \cdots \to u_{i_1})$ in $G(S)$. We claim that at least one of these cycles is an $f$-violating cycle. If not, then by the definition of $f$-violating cycles, we have $\prod_{i \in C_j} |a_{i\sigma(i)}| \leq 2(|C_j|!)^3$ for all $j \leq \ell$. Multiplying over all the cycles in $\sigma$ gives

$$\prod_{i=1}^k |a_{i\sigma(i)}| = \prod_{j=1}^\ell \prod_{i \in C_j} |a_{i\sigma(i)}| \leq \prod_{j=1}^\ell 2(|C_j|!)^3 \leq e^{4k \log(k)},$$

where the last inequality follows from $\sum_{j=1}^\ell |C_j| = k$. This contradicts (16), so $G(S)$ must contain an $f$-violating cycle.

Lemma III.2. If $C$ is a minimal $f$-violating cycle in $G(S)$, then $S \Delta C$ is independent in $\mathcal{M}$.

Proof: For clarity, let $V(C)$ denote the vertex set of $C$. Let $T := \Delta S V(C)$ and let $|C| = 2\ell$. Let consider the graph $G(S)$ with weights $w_T$, and define $w_D := \sum_{e \in D} w_T(e)$ for any cycle $D$. Since $C$ is an $f$-violating cycle, $w_T(C) = w_T(C) + \log(f(\ell)) < 0$.

Let the set of backward arcs in $C$ be $N_2$, and the set of forward arcs be $N_1$. For the sake of contradiction, assume that $T \notin T$. Then, there exists a matching $N_1'$ on $V(C)$ consisting of only backward arcs such that $N_1 \neq N_1'$ (Chapter 39, Theorem 39.13, [19]). Let $A$ be a multiset of arcs consisting of all arcs in $N_2$ twice and all arcs $N_1$ and $N_1'$ (with arcs in $N_1 \cap N_1'$ appearing twice). Consider the directed graph $D = (V(C), A)$. Since $N_1 \neq N_1'$, $D$ contains a directed circuit $C_1$ with $V(C_1) \subseteq V(C)$. Every vertex in $V(C)$ has exactly two in-edges and two out-edges in $A$. Therefore, $D$ is Eulerian, and we can decompose $A$ into directed circuits $C_1, \ldots, C_{\ell}$. Since only arcs in $N_2$ have non-zero weights, we have $\sum_{C_j} w_T(C_j) = 2w_T(C)$. Because $V(C_1) \subseteq V(C)$, at most one cycle $C_j$ can have $V(C_j) = V(C)$. If for some $j$, $V(C) = V(C_j)$, then $w_T(C_j) = w_T(C)$, which must contain every edge in $N_2$.

So, $\sum_{j \neq j} w_T(C_j) = w_T(C) < 0$ and there exists a cycle $C_i$ such that $V(C_i) \subseteq V(C)$ and $w_T(C_i) < 0$. Otherwise $V(C_j) \subseteq V(C')$ for all $j$ and $\sum_{j} w_T(C_j) = 2w_T(C) < 0$. Again, there exists a cycle $C_i$ such that $V(C_i) \subseteq V(C)$ and $w_T(C_i) < 0$.

Let $C'$ be the directed cycle such that $V(C') \subseteq V(C)$ and $w_T(C') \leq w_T(C) \leq 0$. Define $y := |C'|/2$. Thus $w_T(C') = y \cdot \log(f(\ell))/\ell + w_T(C') < 0$. Since $y \ll \ell$, $\log(f(\ell))/\ell \leq \log(f(\ell))/\ell$. Therefore $w_T(C') \leq -y \cdot \log(f(\ell))/\ell \leq -\log(f(\ell))$. So $C'$ is an $f$-violating cycle with $V(C') \subseteq V(C)$, which contradicts the fact that $C$ is a minimal $f$-violating cycle.

References

[1] F. Pukelsheim, Optimal design of experiments. SIAM, 2006.
[2] L. G. Khachiyan, “Rounding of polytopes in the real number model of computation,” Mathematics of Operations Research, vol. 21, no. 2, pp. 307–320, 1996.
[3] N. Anari, S. O. Gharan, A. Saberi, and M. Singh, “Nash social welfare, matrix permanent, and stable polynomials,” in Proceedings of Conference on Innovation in Theoretical Computer Science, 2016.
[4] N. Anari, S. O. Gharan, and C. Vinzant, “Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids,” in 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 35–46, IEEE, 2018.
[5] A. Nikolov, M. Singh, and U. T. Tangipongpitap, “Proportional volume sampling and approximation algorithms for A-optimal design,” in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1369–1386, SIAM, 2019.
[6] A. Kulesza and B. Taskar, “Determinantal point processes for machine learning,” Foundations and Trends in Machine Learning, vol. 5, no. 2–3, pp. 123–286, 2012.
[7] A. Nikolov and M. Singh, “Maximizing determinants under partition constraints,” in ACM symposium on Theory of computing, pp. 192–201, 2016.
[8] N. Anari and S. O. Gharan, “A generalization of permanent inequalities and applications in counting and optimization,” in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pp. 384–396, ACM, 2017.
[9] V. Madan, A. Nikolov, M. Singh, and U. Tangipongpitap, “Maximizing determinants under matroid constraints,” in 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pp. 565–576, IEEE, 2020.
[10] J. B. Ebrahimi, D. Straszak, and N. K. Vishnoi, “Subdeterminant maximization via nonconvex relaxations and anti-concentration,” in 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 1020–1031, IEEE, 2017.
[11] Z. Allen-Zhu, Y. Li, A. Singh, and Y. Wang, “Near-optimal discrete optimization for experimental design: A regret minimization approach,” arXiv preprint arXiv:1711.05174, 2017.
[12] A. Nikolov, “Randomized rounding for the largest simplex problem,” in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, pp. 861–870, ACM, 2015.
[13] M. D. Summa, F. Eisenbrand, Y. Faenza, and C. Moldenhauer, “On largest volume simplices and sub-determinants,” in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 315–323, Society for Industrial and Applied Mathematics, 2015.
[14] A. Nikolov, “Randomized rounding for the largest simplex problem,” in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, pp. 861–870, ACM, 2015.
[15] M. Singh and W. Xie, “Approximate positive correlated distributions and approximation algorithms for D-optimal design,” in Proceedings of SODA, 2018.
[16] V. Madan, M. Singh, U. Tangipongpitap, and W. Xie, “Combinatorial algorithms for optimal design,” in Conference on Learning Theory, pp. 2210–2258, 2019.
[17] L. C. Lau and H. Zhou, “A local search framework for experimental design,” in Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1039–1058, SIAM, 2021.
A. Brown, A. Laddha, M. Pittu, M. Singh, and P. Tetali, “Determinant maximization via matroid intersection algorithms,” 2022.

A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, vol. 24. Springer Science & Business Media, 2003.

N. Anari, K. Liu, S. O. Gharan, and C. Vinzant, “Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid,” in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1–12. ACM, 2019.

I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, “The unreasonable fairness of maximum nash welfare,” in *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC ’16, (New York, NY, USA), pp. 305–322. ACM, 2016.

R. Cole and V. Gkatzelis, “Approximating the nash social welfare with indivisible items,” in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 371–380, 2015.

R. Cole, N. Devanur, V. Gkatzelis, K. Jain, T. Mai, V. V. Vazirani, and S. Yazdanbod, “Convex program duality, fisher markets, and nash social welfare,” in *Proceedings of the 2017 ACM Conference on Economics and Computation*, pp. 459–460, 2017.

S. Barman, S. K. Krishnamurthy, and R. Vaish, “Finding fair and efficient allocations,” in *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 557–574, ACM, 2018.

J. Garg, M. Hoefer, and K. Mehlhorn, “Approximating the nash social welfare with budget-additive valuations,” in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2326–2340, SIAM, 2018.

S. Barman, S. K. Krishnamurthy, and R. Vaish, “Greedy algorithms for maximizing nash social welfare,” in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 7–13, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

N. Anari, T. Mai, S. O. Gharan, and V. V. Vazirani, “Nash social welfare for indivisible items under separable, piecewise-linear concave utilities,” in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2274–2290, SIAM, 2018.

J. Garg, E. Husi, and L. A. Végh, “Approximating nash social welfare under rado valuations,” in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1412–1425, 2021.

W. Li and J. Vondrak, “A constant-factor approximation algorithm for nash social welfare with submodular valuations,” in *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 25–36, IEEE, 2022.

A. Nikolov, M. Singh, and U. T. Tantipongpipat, “Proportional volume sampling and approximation algorithms for a-optimal design,” *Proceedings of SODA 2019*, 2019.

A. W. Marcus, D. A. Spielman, and N. Srivastava, “Interlacing families ii: Mixed characteristic polynomials and the kadison—singer problem,” *Annals of Mathematics*, pp. 327–350, 2015.

A. Schrijver, *Theory of Linear and Integer Programming*. Wiley-Interscience, 2000.