Subquadratic Submodular Maximization with a General Matroid Constraint

Yusuke Kobayashi
Research Institute for Mathematical Sciences, Kyoto University, Japan

Tatsuya Terao
Research Institute for Mathematical Sciences, Kyoto University, Japan

Abstract

We consider fast algorithms for monotone submodular maximization with a general matroid constraint. We present a randomized $(1 - 1/e - \epsilon)$-approximation algorithm that requires $\tilde{O}_\epsilon(\sqrt{r}n)$ independence oracle and value oracle queries, where n is the number of elements in the matroid and $r \leq n$ is the rank of the matroid. This improves upon the previously best algorithm by Buchbinder-Feldman-Schwartz [Mathematics of Operations Research 2017] that requires $\tilde{O}_\epsilon(r^2 + \sqrt{r}n)$ queries.

Our algorithm is based on continuous relaxation, as with other submodular maximization algorithms in the literature. To achieve subquadratic query complexity, we develop a new rounding algorithm, which is our main technical contribution. The rounding algorithm takes as input a point represented as a convex combination of t bases of a matroid and rounds it to an integral solution. Our rounding algorithm requires $\tilde{O}(r^{3/2}t)$ independence oracle queries, while the previously best rounding algorithm by Chekuri-Vondrák-Zenklusen [FOCS 2010] requires $O(r^2t)$ independence oracle queries. A key idea in our rounding algorithm is to use a directed cycle of arbitrary length in an auxiliary graph, while the algorithm of Chekuri-Vondrák-Zenklusen focused on directed cycles of length two.

2012 ACM Subject Classification Theory of computation → Algorithm design techniques; Theory of computation → Submodular optimization and polymatroids

Keywords and phrases submodular maximization, matroid constraint, approximation algorithm, rounding algorithm, query complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.100

Category Track A: Algorithms, Complexity and Games

Funding This work was partially supported by the joint project of Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society", by JST ERATO Grant Number JPMJER2310, and by JSPS KAKENHI Grant Numbers JP20K11692, JP22H05001, JP24KJ1494, and JP24K02901.

Acknowledgements The authors thank the three anonymous reviewers for their valuable comments.

1 Introduction

1.1 Submodular Maximization

Submodular maximization is a fundamental and well-studied problem in theoretical computer science and combinatorial optimization. This is because a number of important problems can be regarded as special cases of submodular maximization, including maximum coverage, generalized assignment, and facility location. Furthermore, submodular maximization has many practical applications in machine learning, economics, and many other areas. In the submodular maximization problem, the input consists of a (monotone) submodular set function $f\colon 2^V \to \mathbb{R}_+$ and a feasible region $\mathcal{F} \subseteq 2^V$ specified by some constraints, and the aim is to find a set $S \in \mathcal{F}$ maximizing $f(S)$.

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 100; pp. 100:1–100:19
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The study of submodular maximization was initiated by a seminal work of Fisher, Nemhauser, and Wolsey in the 1970's [26,34,35]. They showed that, for monotone submodular maximization, the greedy algorithm achieves $(1-1/e)$-approximation for a cardinality constraint and $\frac{1}{2}$-approximation for a matroid constraint. The advantage of their algorithm is that it is very simple and fast, indeed, it runs in quadratic time. It is known that unless $\mathrm{P}=\mathrm{NP}$, for any $\varepsilon > 0$, there is no $(1-1/e+\varepsilon)$-approximation algorithm for the maximum coverage problem [24], which is a special case of monotone submodular maximization with a cardinality constraint or a matroid constraint. Thus, the factor $1-1/e$ is optimal for a cardinality constraint.

To obtain an optimal $(1-1/e)$-approximation algorithm for a matroid constraint, Calinescu-Chekuri-Pál-Vondrák [13] developed a framework based on continuous optimization and rounding technique. In their algorithm, they first solve the continuous optimization problem of maximizing the multilinear extension F of f, a natural continuous extension of f. By using a continuous greedy algorithm, they obtain a $(1-1/e)$-approximation solution for the continuous optimization problem. In order to round the obtained fractional solution to an integral one, they use a variant of the pipage rounding technique of Ageev-Sviridenko [1]. Consequently, their algorithm achieves the optimal $(1-1/e)$-approximation. Note that, although their algorithm runs in polynomial time, its running time is very high.

Since submodular maximization has a number of applications, providing efficient approximation algorithms is a fundamental task both in theory and in practice. Thus, it has received considerable attention to develop fast submodular maximization algorithms that achieve an approximation close to the optimal factor, typically with an approximation factor of $1-1/e-\varepsilon$ for any $\varepsilon > 0$.

In the submodular maximization problem with a general matroid constraint, it is standard to suppose that the objective function f is given as a value oracle, and the feasible region $\mathcal{F} \subseteq 2^V$ is given as an independence oracle of a matroid. In such a case, the efficiency of an algorithm is usually measured by the number of value and independence oracle queries used in it.

Badanidiyuru-Vondrák [3] presented a fast algorithm that achieves an almost optimal approximation factor $1-1/e-\varepsilon$, for any $\varepsilon > 0$, for a matroid constraint. Their algorithm uses $O\left(\frac{rn}{\varepsilon^4}\log^2\left(\frac{n}{\varepsilon}\right)\right)$ value oracle queries and $O\left(\frac{n}{\varepsilon^2}\log\left(\frac{n}{\varepsilon}\right)+\frac{r^2}{\varepsilon}\right)$ independence oracle queries, where n is the number of elements in the matroid and r is the rank of the matroid. To achieve this query complexity, they developed a fast implementation of the continuous greedy algorithm that uses $\tilde{O}_\varepsilon(rn)$ value oracle queries and $\tilde{O}_\varepsilon(n)$ independence oracle queries.[1] The output of their continuous greedy algorithm is a fractional solution represented as a convex combination of $1/\varepsilon$ bases. Then, they apply the swap rounding algorithm of Chekuri-Vondrák-Zenklusen [18] to round the obtained fractional solution to an integral solution, which requires $O(r^2/\varepsilon)$ independence oracle queries.

Buchbinder-Feldman-Schwartz [12] presented a $(1-1/e-\varepsilon)$-approximation algorithm that has a trade-off between the number of value oracle queries and the number of independence oracle queries used in the algorithm. In their algorithm, they combine a variant of the residual random greedy algorithm of Buchbinder-Feldman-Naor-Schwartz [11] and the fast continuous greedy algorithm of Badanidiyuru-Vondrák described above. Then, for a parameter $\lambda \in [1, r]$, their algorithm uses $\tilde{O}_\varepsilon(r\lambda + \frac{rn}{\lambda})$ value oracle queries and $\tilde{O}_\varepsilon(\lambda n + r^2)$ independence oracle queries. We note that the $\tilde{O}_\varepsilon(r^2)$ term in the independence query complexity is due to

[1] The $\tilde{O}_\varepsilon$ notation hides polylogarithmic factors in n and polynomial factors in ε^{-1}.

the rounding algorithm in the same way as the algorithm of Badanidiyuru-Vondrák. If we evaluate the algorithm by the total number of queries regardless of their types, then the query complexity is minimized when $\lambda = \Theta(\sqrt{r})$. In this case, their algorithm uses $\tilde{O}_\varepsilon(r^2 + \sqrt{r}n)$ value and independence oracle queries. This query complexity is better than that of Badanidiyuru-Vondrák [3] when $r = o(n)$, but a quadratic number of queries is still required when r is large.

Recently, for several important classes of matroids, faster algorithms for monotone submodular maximization with a matroid constraint have been investigated. Ene-Nguyễn [22] presented a $(1 - 1/e - \varepsilon)$-approximation algorithm for graphic matroid and partition matroid constraints in time nearly-linear in the size of their representation. Henzinger-Liu-Vondrák-Zheng [27] presented a $(1 - 1/e - \varepsilon)$-approximation algorithm for laminar matroid and transversal matroid constraints in nearly-linear time. A key ingredient in these algorithms is a fast dynamic data structure for maintaining an (approximate) maximum weight basis of the matroid.

1.2 Our Results

This paper focuses on the monotone submodular maximization problem with a general matroid constraint. In the problem, this input consists of a monotone submodular set function $f\colon 2^V \to \mathbb{R}_+$ given as a value oracle, and a matroid $\mathcal{M} = (V, \mathcal{I})$ given as an independence oracle. The objective is to find an independent set $S \in \mathcal{I}$ that maximizes $f(S)$. For $\alpha \in [0, 1]$, a randomized algorithm is said to be an *α-approximation algorithm* if it returns a solution $S \in \mathcal{I}$ with $\mathbb{E}[f(S)] \geq \alpha \cdot \max\{f(T) \mid T \in \mathcal{I}\}$. A randomized algorithm is often called simply an algorithm throughout the paper. Our main result is to give a first $(1 - 1/e - \varepsilon)$-approximation algorithm for this problem that requires a subquadratic number of queries. Recall that $n = |V|$ and r is the rank of $\mathcal{M}$.

▶ **Theorem 1.** *For any $\varepsilon > 0$, there is a randomized algorithm that achieves $(1 - 1/e - \varepsilon)$-approximation for maximizing a monotone submodular function subject to a matroid constraint and uses $O(\sqrt{r}n \operatorname{poly}(1/\varepsilon, \log n))$ value and independence oracle queries.*

It is worth mentioning that, for the case of $r = \Theta(n)$, our algorithm uses $\tilde{O}_\varepsilon(n^{3/2})$ oracle queries, whereas the algorithm of Buchbinder-Feldman-Schwartz [12] uses $\tilde{O}_\varepsilon(n^2)$ oracle queries.

Our algorithm is based on continuous relaxation and rounding technique in the same way as previous algorithms [3, 12, 13]. In this framework, currently, the bottleneck of the query complexity comes from the rounding algorithm. Indeed, the swap rounding algorithm by Chekuri-Vondrák-Zenklusen [18] requires $O(r^2 t)$ independence oracle queries if the input point is represented as a convex combination of t bases of the matroid. Then, this rounding algorithm requires a quadratic number of independence oracle queries even when t is small. Therefore, in order to break the quadratic-independence-query barrier in this framework, it is necessary to devise a faster rounding algorithm.

The key technical contribution of this paper is to develop a new rounding algorithm that uses $o(r^2 t)$ independence oracle queries.

▶ **Theorem 2.** *For any $\varepsilon > 0$, there is a randomized algorithm satisfying the following conditions:*

- *the input consists of a matroid $\mathcal{M} = (V, \mathcal{I})$ given as an independence oracle and a point x in the base polytope of $\mathcal{M}$ that is represented as a convex combination of t bases,*
- *the output is a basis S of $\mathcal{M}$ such that $\mathbb{E}[f(S)] \geq (1 - \varepsilon)F(x)$ for any submodular function $f\colon 2^V \to \mathbb{R}$ and its multilinear extension F, and*
- *it uses $O(r^{3/2} t \log^{3/2}(\frac{rt}{\varepsilon}))$ independence oracle queries.*

By combining this theorem with the submodular maximization algorithm by Buchbinder-Feldman-Schwartz [12], we obtain Theorem 1; see Section 3 for details.

We also show that if the matroid is given as a rank oracle instead of an independence oracle, then we obtain a $(1-1/e-\varepsilon)$-approximation algorithm using $\tilde{O}_\varepsilon(n+r^{3/2})$ value and rank oracle queries.

▶ **Theorem 3.** *For any $\varepsilon > 0$, there is a randomized algorithm that achieves $(1-1/e-\varepsilon)$-approximation for maximizing a monotone submodular function subject to a matroid constraint and uses $O((n+r^{3/2})\operatorname{poly}(1/\varepsilon, \log n))$ value and rank oracle queries.*

1.3 Overview of Our Rounding Algorithm

In this subsection, we give a technical overview of our new rounding algorithm. Since our rounding algorithm is based on that of Chekuri-Vondrák-Zenklusen [18], we first review their algorithm and then explain the key ideas behind ours.

Swap Rounding Algorithm of Chekuri-Vondrák-Zenklusen. The rounding algorithm by Chekuri-Vondrák-Zenklusen is called the swap rounding algorithm. Their algorithm takes as input a point x represented as a convex combination of t bases of $\mathcal{M}$ and returns an integral solution S such that $\mathbb{E}[f(S)] \geq F(x)$ for any submodular function f and its multilinear extension F. In each phase of the algorithm, we pick up two bases in the representation of x and merge them into a single basis. By applying this procedure $t-1$ times, we obtain a single basis of $\mathcal{M}$.

In order to merge two bases, say B_1 and B_2, their swap rounding algorithm uses a *strongly exchangeable pair* of elements, that is, a pair of elements $u \in B_1 \setminus B_2$ and $v \in B_2 \setminus B_1$ such that $B_1 + v - u \in \mathcal{I}$ and $B_2 + u - v \in \mathcal{I}$. Since we can find a strongly exchangeable pair using $O(r)$ independence oracle queries and we need to find such a pair $O(rt)$ times in the algorithm, the total number of queries is $O(r^2 t)$. It is still not clear whether we can develop an algorithm for finding a strongly exchangeable pair using $o(r)$ independence oracle queries, and hence their algorithm is now stuck at $\Omega(r^2 t)$ independence oracle queries.

See Section 4 for details of the swap rounding algorithm of Chekuri-Vondrák-Zenklusen.

Our Faster Rounding Algorithm. We develop a new rounding algorithm that requires $\tilde{O}(r^{3/2}t)$ independence oracle queries with high probability. Our rounding algorithm is based on that of Chekuri-Vondrák-Zenklusen in a sense that we update bases by swapping a pair of elements $O(rt)$ times. Therefore, in each step of our algorithm, we need to update some basis by using only $\tilde{O}(\sqrt{r})$ independence oracle queries. To achieve this, we need substantially new ideas.

First, we introduce a digraph that represents exchangeability of the elements in the matroid (see Definition 8), and provide a new interpretation of the swap rounding algorithm of Chekuri-Vondrák-Zenklusen using this auxiliary graph. Indeed, each step of their algorithm can be seen as an update using a directed cycle of length two in the auxiliary graph. This interpretation motivates us to focus on a directed cycle of arbitrary length in the auxiliary graph instead of a directed cycle of length two. By extending the argument of Chekuri-Vondrák-Zenklusen, we show that we can appropriately update bases using a directed cycle of arbitrary length in the auxiliary graph.

Second, we show that we can find a directed cycle in the auxiliary graph using $o(r)$ independence oracle queries with high probability, which is the most technical part in our argument. To achieve this, we combine sampling technique and binary search technique.

In our algorithm for finding a directed cycle in the auxiliary graph, we first sample $\tilde{O}(\sqrt{r})$ vertices, and define D' as the subgraph induced by the sampled vertex set. If every vertex in D' has an incoming edge, then we can easily find a directed cycle in D' by traversing such directed edges in the opposite direction. Otherwise, by using a vertex with no incoming edge, we find a directed cycle of length two using $\tilde{O}(\sqrt{r})$ independence oracle queries with high probability. We can check whether each vertex in D' has an incoming edge or not using $\tilde{O}(1)$ independence oracle queries with the aid of the binary search technique proposed by Nguyễn [36] and Chakrabarty-Lee-Sidford-Singla-Wong [14]; see Lemma 4 for details. Note that this technique was used in recent studies on fast matroid intersection [8,10,14,36,39] and matroid partition [38] algorithms. Therefore, we obtain an algorithm that finds a directed cycle using $\tilde{O}(\sqrt{r})$ independence oracle queries with high probability.

In our rounding algorithm, we update bases using a directed cycle in the auxiliary graph repeatedly. Since we update bases $O(rt)$ times and each update requires $\tilde{O}(\sqrt{r})$ independence oracle queries, the total number of independence oracle queries is $\tilde{O}(r^{3/2}t)$ with high probability.

1.4 Related Work

We have mentioned several recent studies on fast submodular maximization with matroid constraints in Section 1.1. Other than these, there are a lot of studies on fast submodular maximization algorithms in the literature [2,3,15,20,25,30,32].

Badanidiyuru-Vondrák [3] developed a $(1-1/e-\varepsilon)$-approximation algorithm using $O(\frac{n}{\varepsilon}\log(\frac{n}{\varepsilon}))$ value oracle queries for the cardinality constraint. Mirzasoleiman-Badanidiyuru-Ashwinkumar-Karbasi-Vondrák-Krause [32] developed a $(1-1/e-\varepsilon)$-approximation algorithm using $O(n\log(1/\varepsilon))$ value oracle queries for the cardinality constraint. Ene-Nguyễn [20] developed a $(1-1/e-\varepsilon)$-approximation algorithm using $(1/\varepsilon)^{O(1/\varepsilon^4)}n\log^2 n$ value oracle queries for the knapsack constraint. Filmus-Ward [25] presented a combinatorial $(1-1/e)$-approximation algorithm for monotone submodular maximization with a matroid constraint, which uses $O(n^7r^2)$ oracle queries. They also obtain a $(1-1/e-O(\varepsilon))$-approximation algorithm that uses $O(\varepsilon^{-3}n^4r)$ value oracle queries and $O(\varepsilon^{-1}n^2r\log n)$ independence oracle queries.

Studies on fast submodular maximization algorithms have developed also in the direction of parallelized settings [4,5,16,21,23], distributed settings [7,31], and dynamic settings [6,19,29,33].

Chekuri-Quanrud-Torres [17] developed a fast swap rounding algorithm for graphic matroid constraints to obtain fast approximation algorithms for the Bounded Degree MST problem and the Crossing Spanning Tree problem.

1.5 Paper Organization

The remaining of this paper is organized as follows. In Section 2, we give some preliminaries. In Section 3, we show how to derive Theorem 1 from our fast rounding algorithm in Theorem 2. In Section 4, we describe the swap rounding algorithm by Chekuri-Vondrák-Zenklusen [18] in detail, because it is the basis of our rounding algorithm. In Section 5, we describe our fast rounding algorithm and prove Theorem 2, which is the main technical part of this paper. In Section 6, we discuss the rank oracle setting and prove Theorem 3.

2 Preliminaries

Basic Notation. Let $\mathbb{R}_+$ denote the set of non-negative real numbers. Throughout this paper, let V be a finite set and let n denote its cardinality. For a set $A \subseteq V$ and an element $v \in V$, we will often write $A+v := A \cup \{v\}$ and $A-v := A \setminus \{v\}$. For two sets $A, B \subseteq V$, their symmetric difference is denoted by $A \triangle B := (A \setminus B) \cup (B \setminus A)$. For $A \subseteq V$, the *characteristic vector* of A is defined as the vector $x \in \{0,1\}^V$ with $x_v = 1$ for $v \in A$ and $x_v = 0$ for $v \in V \setminus A$. We will denote by $\mathbf{1}_A$ the characteristic vector of A. For $v \in V$, we will write $\mathbf{1}_v := \mathbf{1}_{\{v\}}$.

Submodular Functions and Multilinear Extension. Let $f : 2^V \to \mathbb{R}_+$ be a set function on a finite ground set V of size n. The function is *submodular* if $f(A)+f(B) \geq f(A \cup B)+f(A \cap B)$ for any two subsets $A, B \subseteq V$. The function is *monotone* if $f(A) \leq f(B)$ for any subsets $A \subseteq B \subseteq V$. In this paper, we only consider monotone submodular functions.

For a function $f : 2^V \to \mathbb{R}_+$, we define its *multilinear extension* $F : [0,1]^V \to \mathbb{R}_+$ by

$$F(x) = \sum_{S \subseteq V} f(S) \prod_{v \in S} x_v \prod_{v \in V \setminus S} (1 - x_v)$$

for $x \in [0,1]^V$. Note that this value is equal to $\mathbb{E}[f(R(x))]$, where $R(x)$ is a random set that contains each element $v \in V$ independently with probability x_v. In particular, $F(\mathbf{1}_S) = f(S)$ for any $S \subseteq V$.

Matroids. A pair $\mathcal{M} = (V, \mathcal{I})$ of a finite set V and a non-empty set family $\mathcal{I} \subseteq 2^V$ is called a *matroid* if the following properties are satisfied.

(Downward closure property) if $S \in \mathcal{I}$ and $S' \subseteq S$, then $S' \in \mathcal{I}$.
(Augmentation property) if $S, S' \in \mathcal{I}$ and $|S'| < |S|$, then there exists $v \in S \setminus S'$ such that $S' + v \in \mathcal{I}$.

A set $S \subseteq V$ is called *independent* if $S \in \mathcal{I}$ and *dependent* otherwise. The *rank* of $\mathcal{M}$ is defined as the size of a largest independent set. In addition, for a subset $S \subseteq V$, the rank of S is defined as the size of a largest independent set contained in S. Inclusionwise maximal independent sets are called *bases*. Note that every basis has the same size. For an independent set $S \in \mathcal{I}$, let $\mathcal{M}/S = (V \setminus S, \mathcal{I}')$ be the matroid obtained by contracting S in $\mathcal{M}$, that is, $S' \in \mathcal{I}'$ if and only if $S' \cup S \in \mathcal{I}$.

Let $\mathcal{B}$ be the set of all bases of a matroid $\mathcal{M} = (V, \mathcal{I})$ and let $B, B' \in \mathcal{B}$ be two bases. It is well-known that, for any $u \in B \setminus B'$, there exists $v \in B' \setminus B$ such that $B - u + v \in \mathcal{B}$ and $B' - v + u \in \mathcal{B}$ (see e.g., [37, Theorem 39.12]). This property is called *strong basis exchange property* of matroids.

Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid whose rank function and basis family are denoted by $r_{\mathcal{M}}$ and $\mathcal{B}$, respectively. The *matroid polytope* $P(\mathcal{M})$ is defined as the convex hull of the characteristic vectors of all the independent sets of $\mathcal{M}$. The *matroid base polytope* $B(\mathcal{M})$ is defined as the convex hull of the characteristic vectors of all the bases of $\mathcal{M}$. It is well-known that $P(\mathcal{M})$ and $B(\mathcal{M})$ are described as follows (see e.g., [37, Section 40.2]):

$$P(\mathcal{M}) := \mathrm{conv}\{\mathbf{1}_I \mid I \in \mathcal{I}\} = \left\{ x \in \mathbb{R}_+^V \;\middle|\; \sum_{v \in S} x_v \leq r_{\mathcal{M}}(S) \text{ for any } S \subseteq V \right\},$$

$$B(\mathcal{M}) := \mathrm{conv}\{\mathbf{1}_B \mid B \in \mathcal{B}\} = \left\{ x \in P(\mathcal{M}) \;\middle|\; \sum_{v \in V} x_v = r_{\mathcal{M}}(V) \right\}.$$

Oracles. When we consider the submodular maximization problem, we assume that the submodular function f is given as a value oracle, which takes as input any subset $S \subseteq V$ and outputs $f(S)$. We also assume that we access a matroid $\mathcal{M}$ through an oracle. Given a subset $S \subseteq V$, an *independence oracle* outputs whether $S \in \mathcal{I}$ or not. Given a subset $S \subseteq V$, a *rank oracle* outputs the rank of S, i.e., the size of a largest independent set contained in S. Note that the rank oracle is more powerful than the independence oracle, since one query of the rank oracle can determine whether a given subset is independent or not.

Binary Search Technique. For a matroid $\mathcal{M} = (V, \mathcal{I})$, an independent set $S \in \mathcal{I}$, an element $u \in V \setminus S$, and $T \subseteq S$, we consider a procedure that finds an element $v \in T$ with $S + u - v \in \mathcal{I}$ if one exists. Chakrabarty et al. [14] and Nguyễn [36] independently proved that this procedure can be implemented efficiently by using the binary search technique in the independence oracle model. Their result is formally described as follows.

▶ **Lemma 4** ([14, 36]). *There is an algorithm* `FindExchangeElement` *which, given a matroid $\mathcal{M} = (V, \mathcal{I})$, an independent set $S \in \mathcal{I}$, an element $u \in V \setminus S$, and $T \subseteq S$, finds an element $v \in T$ such that $S + u - v \in \mathcal{I}$ or otherwise determines that no such element exists, and uses $O(\log |T|)$ independence oracle queries.*

3 Submodular Maxmization Algorithm (Proof of Theorem 1)

In this section, we give a proof of Theorem 1 by combining the algorithm of Buchbinder-Feldman-Schwartz [12] and our rounding algorithm in Theorem 2. Note that a proof of Theorem 2 is given in Section 5 later.

For monotone submodular maximization with a matroid constraint, Buchbinder-Feldman-Schwartz presented a $(1 - 1/e - \varepsilon)$-approximation algorithm that has a trade-off between the number of value oracle queries and the number of independence oracle queries used in the algorithm. The main part of their algorithm is to solve the continuous relaxation of the submodular maximization problem efficiently.

Let $\lambda \in [1, r]$ be a parameter that controls the trade-off. In their algorithm for solving the continuous relaxation problem, they first apply a variant of the residual random greedy algorithm of Buchbinder-Feldman-Naor-Schwartz [11]. This residual random greedy algorithm outputs $S \subseteq V$ and uses $\tilde{O}_\varepsilon(r\lambda + n)$ value oracle queries and $\tilde{O}_\varepsilon(\lambda n)$ independence oracle queries; see [12, Lemma 3.3]. Then they apply a variant of the fast continuous greedy algorithm of Badanidiyuru-Vondrák [3]. This continuous greedy algorithm outputs a point x' represented as a convex combination of $1/\varepsilon$ bases of $\mathcal{M}/S$ and uses $\tilde{O}_\varepsilon(\frac{rn}{\lambda})$ value oracle queries and $\tilde{O}_\varepsilon(n)$ independence oracle queries; see [12, Corollary 3.1]. Then $x = \mathbf{1}_S \vee x'$ is an approximate solution for the continuous relaxation problem, which can be represented as a convex combination of $1/\varepsilon$ bases of $\mathcal{M}$. Here, for vectors y and z, let $y \vee z$ denote the vector such that $(y \vee z)_i = \max\{y_i, z_i\}$ for all i.

Overall, Buchbinder-Feldman-Schwartz [12] presented an efficient algorithm for solving the continuous relaxation problem, which is formally stated as follows.

▶ **Theorem 5** (follows from [12, Corollary 3.1] and [12, Lemma 3.3]). *Given a non-negative monotone submodular function $f\colon 2^V \to \mathbb{R}_+$, a matroid $\mathcal{M} = (V, \mathcal{I})$ of rank r, and parameters $\varepsilon > 0$ and $\lambda \in [1, r]$, there is an algorithm satisfying the following conditions:*

- *the algorithm outputs a point $x \in B(\mathcal{M})$ represented as a convex combination of $1/\varepsilon$ bases such that $\mathbb{E}[F(x)] \geq (1 - 1/e - \varepsilon) \cdot \max\{f(T) \mid T \in \mathcal{I}\}$ holds, where $F\colon [0,1]^V \to \mathbb{R}_+$ is the multilinear extension of f,*

- *it uses* $O\left(r\lambda + \dfrac{rn}{\lambda\varepsilon^5}\log^2\left(\dfrac{n}{\varepsilon}\right)\right)$ *value oracle queries, and*
- *it uses* $O\left(\dfrac{\lambda n}{\varepsilon^2}\log\left(\dfrac{n}{\varepsilon}\right)\right)$ *independence oracle queries.*

Suppose that $x \in B(\mathcal{M})$ is a point as in Theorem 5. In the submodular maximization algorithm of Buchbinder-Feldman-Schwartz, they round x to an integral solution with the aid of the swap rounding algorithm of Chekuri-Vondrák-Zenklusen [18] using $O(r^2/\varepsilon)$ independence oracle queries. Therefore, their entire algorithm requires $\tilde{O}_\varepsilon(r\lambda + \frac{rn}{\lambda})$ value oracle queries and $\tilde{O}_\varepsilon(\lambda n + r^2)$ independence oracle queries.

▶ Remark 6. Theorem 5 is not explicitly stated in the paper by Buchbinder-Feldman-Schwartz [12], because they do not separately evaluate the query complexity for solving the continuous relaxation problem and for the rounding algorithm. Indeed, they just state that the entire algorithm requires $O(\frac{r^2}{\varepsilon} + \frac{\lambda n}{\varepsilon^2}\log\left(\frac{n}{\varepsilon}\right))$ independence oracle queries; see [12, Theorem 1.1]. The $O(\frac{r^2}{\varepsilon})$ term in this query complexity comes from [12, Corollary 3.1], which states that the continuous greedy algorithm together with the rounding algorithm requires $O(\frac{n}{\varepsilon^2}\log(\frac{n}{\varepsilon}) + \frac{r^2}{\varepsilon})$ independence oracle queries. This corollary is a direct consequence of [3, Claim 4.4], whose proof shows that the $O(\frac{r^2}{\varepsilon})$ term comes from the rounding algorithm, while the $O(\frac{n}{\varepsilon^2}\log(\frac{n}{\varepsilon}))$ term comes from the continuous greedy algorithm. Therefore, the $O(\frac{r^2}{\varepsilon})$ term is not needed to solve the continuous relaxation problem.

We now show that our submodular maximization algorithm with subquadratic query complexity is derived from Theorems 2 and 5.

Proof of Theorem 1. Let $\lambda = \Theta(\sqrt{r})$ and let $\varepsilon' = \varepsilon/2$. Note that we can compute r using $O(n)$ independence oracle queries by a greedy algorithm. We first run the algorithm in Theorem 5 with parameters λ and ε' to obtain a point $x \in B(\mathcal{M})$, in which we use $O(\sqrt{r}n \operatorname{poly}(1/\varepsilon, \log n))$ value and independence oracle queries. For the obtained point x, we apply our fast rounding algorithm in Theorem 2 with an error parameter ε'. Then, we obtain a basis S of $\mathcal{M}$ such that

$$\begin{aligned}
\mathbb{E}[f(S)] &\geq (1-\varepsilon') \cdot \mathbb{E}[F(x)] \\
&\geq (1-\varepsilon') \cdot (1 - 1/e - \varepsilon') \cdot \max\{f(T) \mid T \in \mathcal{I}\} \\
&\geq (1 - 1/e - \varepsilon) \cdot \max\{f(T) \mid T \in \mathcal{I}\}.
\end{aligned}$$

Since x is represented as a convex combination of $1/\varepsilon'$ bases of $\mathcal{M}$ by Theorem 5, our rounding algorithm requires $O(r^{3/2} \operatorname{poly}(1/\varepsilon, \log n))$ independence oracle queries by Theorem 2. Therefore, we obtain a $(1 - 1/e - \varepsilon)$-approximation algorithm that uses $O(\sqrt{r}n \operatorname{poly}(1/\varepsilon, \log n))$ value and independence oracle queries, which completes the proof. ◀

4 Swap Rounding Algorithm in Previous Work

In this section, we describe the swap rounding algorithm of Chekuri-Vondrák-Zenklusen [18] for a matroid base polytope, which we denote `SwapRound`. As described in Section 1.3, our new rounding algorithm is based on `SwapRound`. In `SwapRound`, we are given a point $x \in B(\mathcal{M})$ that is represented as a convex combination of the characteristic vectors of t bases of $\mathcal{M}$. The output is a single basis S of $\mathcal{M}$ such that $\mathbb{E}[f(S)] \geq F(x)$ for any submodular function $f\colon 2^V \to \mathbb{R}$ and its multilinear extension F. In each phase of `SwapRound`, we pick up two bases in the representation of x and merge them into a basis. By applying this procedure $t-1$ times, `SwapRound` finally returns a single basis of $\mathcal{M}$; see Algorithm 1.

The procedure for merging two bases is denoted by `MergeBases` (Algorithm 2). The input of `MergeBases` consists of two bases B_1 and B_2 together with their coefficients β_1 and β_2. In the procedure, until B_1 and B_2 coincide, we repeatedly update B_1 and B_2 so that $|B_1 \setminus B_2|$ decreases monotonically. In each update of B_1 and B_2, we need a *strongly exchangeable pair* of elements, that is, a pair of elements $u \in B_1 \setminus B_2$ and $v \in B_2 \setminus B_1$ such that $B_1 + v - u \in \mathcal{I}$ and $B_2 + u - v \in \mathcal{I}$. As described in `UpdateViaStrongBasisExchange` (Algorithm 3), for a strongly exchangeable pair u and v, we apply $B_1 \leftarrow B_1 + v - u$ with probability $\frac{\beta_2}{\beta_1+\beta_2}$ and apply $B_2 \leftarrow B_2 + u - v$ with the remaining probability. Note that, in `UpdateViaStrongBasisExchange`, B_1 and B_2 are updated to B_1' and B_2' so that $\mathbb{E}[\beta_1 \mathbf{1}_{B_1'} + \beta_2 \mathbf{1}_{B_2'}] = \beta_1 \mathbf{1}_{B_1} + \beta_2 \mathbf{1}_{B_2}$, which is a key property to show the validity of the algorithm.

The most time consuming part in `MergeBases` is to find a strongly exchangeable pair. By the strong basis exchange property of matroids, we can find such a pair of elements using $O(r)$ independence oracle queries in the following way: fix an element $u \in B_1 \setminus B_2$ arbitrarily and check the conditions for each element $v \in B_2 \setminus B_1$ one by one. Since we update the bases $|B_1 \setminus B_2| = O(r)$ times, `MergeBases` requires $O(r^2)$ independence oracle queries in total. Hence, `SwapRound` requires $O(r^2 t)$ independence oracle queries.

It is not clear whether we can develop an algorithm that finds a strongly exchangeable pair using $o(r)$ independence oracle queries. Therefore, their implementation of `SwapRound` is now stuck at $\Omega(r^2 t)$ independence oracle queries.

Algorithm 1 `SwapRound`$(x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i})$.

$C_1 \leftarrow B_1$
$\gamma_1 \leftarrow \beta_1$
for $i = 1$ to $t - 1$ **do**
  $C_{i+1} \leftarrow$ `MergeBases`$(\gamma_i, C_i, \beta_{i+1}, B_{i+1})$
  $\gamma_{i+1} \leftarrow \gamma_i + \beta_{i+1}$
return C_t

Algorithm 2 `MergeBases`$(\beta_1, B_1, \beta_2, B_2)$.

while $B_1 \neq B_2$ **do**
  Pick arbitrary $u \in B_1 \setminus B_2$
  Find $v \in B_2 \setminus B_1$ such that $B_1 + v - u \in \mathcal{I}$ and $B_2 + u - v \in \mathcal{I}$
  `UpdateViaStrongBasisExchange`$(\beta_1, B_1, \beta_2, B_2, v, u)$
return B_1

5 Faster Rounding Algorithm

In this section, we present our fast rounding algorithm. We first show the following theorem, and then prove Theorem 2 using this theorem.

▶ **Theorem 7.** *There is a randomized algorithm satisfying the following conditions:*

- *the input consists of a matroid $\mathcal{M} = (V, \mathcal{I})$ given as an independence oracle and a point $x \in B(\mathcal{M})$ represented as a convex combination of t bases,*
- *the output is a basis S of $\mathcal{M}$ such that $\mathbb{E}[f(S)] \geq F(x)$ for any submodular function $f : 2^V \to \mathbb{R}$ and its multilinear extension F, and*
- *it uses $O(r^{3/2} t \log^{3/2}(rt))$ independence oracle queries with probability at least $1 - (rt)^{-1}$.*

Algorithm 3 `UpdateViaStrongBasisExchange`$(\beta_1, B_1, \beta_2, B_2, v, u)$.

Input: $\beta_1, \beta_2 \in \mathbb{R}_+$, two bases B_1, B_2, and elements $v \in B_2 \setminus B_1$ and $u \in B_1 \setminus B_2$ such that $B_1 + v - u \in \mathcal{I}$ and $B_2 + u - v \in \mathcal{I}$

Flip a coin with `Heads` probability $\dfrac{\beta_2}{\beta_1 + \beta_2}$

if *coin flipped* `Heads` **then**

  $B_1 \leftarrow B_1 + v - u$

else

  $B_2 \leftarrow B_2 + u - v$

To show this theorem, we propose an algorithm that merges the bases one by one in the same way as `SwapRound`. Our contribution is to improve `MergeBases`, that is, we give a faster algorithm for merging two bases into a single basis. The following auxiliary graph plays an important role in our algorithm.

▶ **Definition 8.** *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid, and let B_1, B_2 be two bases of $\mathcal{M}$. Then we define the bipartite directed graph $D_{\mathcal{M}}(B_1, B_2)$ whose vertex set and edge set are $B_1 \triangle B_2$ and $E_1(B_1, B_2) \cup E_2(B_1, B_2)$, respectively, where*

$$E_1(B_1, B_2) = \{(u, v) \mid u \in B_1 \setminus B_2, v \in B_2 \setminus B_1, B_1 + v - u \in \mathcal{I}\},$$
$$E_2(B_1, B_2) = \{(v, u) \mid u \in B_1 \setminus B_2, v \in B_2 \setminus B_1, B_2 + u - v \in \mathcal{I}\}.$$

In terms of this auxiliary graph, each step of `MergeBases` can be interpreted as follows: it finds a directed cycle of length two (or a bidirected edge) in $D_{\mathcal{M}}(B_1, B_2)$ and updates the bases B_1 and B_2 using this directed cycle as in `UpdateViaStrongBasisExchange`. Note that we use $O(r)$ independence oracle queries to find a directed cycle of length two.

A key idea in our algorithm is to focus on a directed cycle of arbitrary length in $D_{\mathcal{M}}(B_1, B_2)$ instead of a directed cycle of length two. More precisely, our contribution consists of the following two technical results.

1. We can find a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$ using $o(r)$ independence oracle queries with high probability.
2. We can appropriately update the bases using a directed cycle of arbitrary length in $D_{\mathcal{M}}(B_1, B_2)$.

We discuss the first and second technical results in Sections 5.1 and 5.2, respectively. Then, we describe the entire algorithm and give proofs for Theorems 2 and 7 in Section 5.3.

5.1 Finding a Directed Cycle

The objective of this subsection is to show the following proposition, which states that we can find a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$ using $o(r)$ independence oracle queries with high probability.

▶ **Proposition 9.** *Suppose we are given two bases B_1 and B_2 of a matroid $\mathcal{M}$ and an integer $t \geq 2$. Then, we can find a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$ using $O(\sqrt{r} \log^{3/2}(rt))$ independence oracle queries with probability at least $1 - (rt)^{-2}$.*

To show this proposition, we first show that a directed cycle of length two can be found efficiently if we have an element whose indegree is small in $D_{\mathcal{M}}(B_1, B_2)$.

▶ **Lemma 10.** *Suppose we are given two bases B_1 and B_2 of a matroid $\mathcal{M}$, and an element $a \in B_1 \triangle B_2$ whose indegree is d in $D_{\mathcal{M}}(B_1, B_2)$. Then, we can find a directed cycle of length two in $D_{\mathcal{M}}(B_1, B_2)$ using $O(d \log r)$ independence oracle queries.*

Proof. By symmetry, it suffices to consider the case when $a \in B_1 \setminus B_2$.

We give an algorithm that finds an element $v \in B_2 \setminus B_1$ such that $B_1 + v - a \in \mathcal{I}$ and $B_2 + a - v \in \mathcal{I}$. In our algorithm, let $A \subseteq B_2 \setminus B_1$ denote the set of elements v such that we have already checked that $B_1 + v - a \notin \mathcal{I}$. We initialize $A = \emptyset$.

In each step of our algorithm, by applying Lemma 4 in which $u = a$, $S = B_2$, and $T = B_2 \setminus (B_1 \cup A)$, we can find an element $v \in T$ such that $B_2 + a - v \in \mathcal{I}$ if it exists. For such v, we check whether $B_1 + v - a \in \mathcal{I}$ holds or not. If $B_1 + v - a \in \mathcal{I}$ holds, then a and v induce a desired directed cycle. Otherwise, we add v to A, and repeat the procedure.

This algorithm finds a directed cycle correctly by the strong basis exchange property. Since we apply Lemma 4 at most d times and $|T| \leq r$, this algorithm uses $O(d \log r)$ independence oracle queries. ◀

We now describe our algorithm for finding a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$. In our algorithm, we first sample $2\sqrt{r \log(rt)}$ elements from $B_1 \setminus B_2$ (resp. $B_2 \setminus B_1$) uniformly at random with replacement, where the base of the logarithm is e, and let L (resp. R) be the sampled vertex set, ignoring the multiplicity. Note that $1 \leq |L| \leq 2\sqrt{r \log(rt)}$ and $1 \leq |R| \leq 2\sqrt{r \log(rt)}$ as we ignore the multiplicity. Let D' be the subgraph of $D_{\mathcal{M}}(B_1, B_2)$ induced by $L \cup R$.

For each vertex u in D', we find a directed edge in D' that enters u or conclude that such a directed edge does not exist. This can be done by calling `FindExchangeElement` exactly once for each u. If every vertex in D' has an incoming edge, then we can easily find a directed cycle in D' by traversing such directed edges in the opposite direction. Otherwise, we pick up a vertex a in $L \cup R$ that has no incoming edge in D', and then apply Lemma 10 with this vertex a to find a directed cycle of length two.

Since the correctness of this algorithm is clear, it suffices to analyze the independence query complexity. We use the following lemma in our analysis.

▶ **Lemma 11.** *Let $u \in B_1 \triangle B_2$ be an element whose indegree in $D_{\mathcal{M}}(B_1, B_2)$ is at least $2\sqrt{r \log(rt)}$. Then, the probability that $D_{\mathcal{M}}(B_1, B_2)$ has no directed edge from $L \cup R$ to u is at most $(rt)^{-4}$.*

Proof. By symmetry, it suffices to consider the case when $u \in B_1 \setminus B_2$. Let $N = \{v \in B_2 \setminus B_1 \mid (v, u) \in E(B_1, B_2)\}$. Since R is obtained by sampling $2\sqrt{r \log(rt)}$ vertices from $B_2 \setminus B_1$ and $r \geq |B_2 \setminus B_1| \geq |N| \geq 2\sqrt{r \log(rt)}$, we have the following:

$$\begin{aligned}
\Pr\left[\{v \in R \mid (v, u) \in E(B_1, B_2)\} = \emptyset\right] &= \Pr\left[N \cap R = \emptyset\right] \\
&= \left(1 - \frac{|N|}{|B_2 \setminus B_1|}\right)^{2\sqrt{r \log(rt)}} \\
&\leq \left(1 - \frac{2\sqrt{r \log(rt)}}{r}\right)^{2\sqrt{r \log(rt)}} \\
&\leq \left(e^{-1}\right)^{4 \log(rt)} \\
&= (rt)^{-4},
\end{aligned}$$

which completes the proof. ◀

We are now ready to prove Proposition 9.

Proof of Proposition 9. We analyze the independence query complexity of the algorithm described above. First, since we call `FindExchangeElement` for each vertex $u \in L \cup R$ exactly once to find an incoming edge in D', the number of calls of `FindExchangeElement` is $|L \cup R| = O(\sqrt{r \log(rt)})$. Hence, by Lemma 4, the number of independence oracle queries used in this part is $O(\sqrt{r \log(rt)} \log r)$.

We next analyze the number of independence oracle queries when there exists a vertex $a \in L \cup R$ that has no incoming edge in D'.

We call a vertex $u \in L \cup R$ *bad* if D' has no directed edge entering u and $D_{\mathcal{M}}(B_1, B_2)$ has at least $2\sqrt{r \log(rt)}$ directed edges entering u. By Lemma 11, for each $u \in L \cup R$, the vertex u is bad with probability at most $(rt)^{-4}$. Thus, by taking the union bound over all vertices in $L \cup R$, we see that there exists a bad vertex in $L \cup R$ with probability at most $(rt)^{-2}$.

We now consider the case where there is no bad vertex in $L \cup R$. Suppose that there exists a vertex $a \in L \cup R$ that has no incoming edge in D'. Then, since a is not bad, the indegree of a is at most $2\sqrt{r \log(rt)}$ in $D_{\mathcal{M}}(B_1, B_2)$. Therefore, we can apply Lemma 10 with a using $O(\sqrt{r \log(rt)} \log r)$ independence oracle queries.

Therefore, the total number of independence oracle queries used in the algorithm is $O(\sqrt{r \log(rt)} \log r)$ with probability at least $1 - (rt)^{-2}$, which completes the proof. ◀

5.2 Update with a Directed Cycle

In this subsection, we describe how to update the bases using a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$. Let C be a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$ that traverses $u_0, v_0, u_1, v_1, \ldots, v_{l-1}$ in this order, where $u_i \in B_1 \setminus B_2$ and $v_i \in B_2 \setminus B_1$ for each i. In our algorithm, we first choose B_1 with probability $\frac{\beta_2}{\beta_1+\beta_2}$ and choose B_2 with the remaining probability. If we choose B_1, then we pick up an index i uniformly at random from $\{0, \ldots, l-1\}$ and update B_1 by $B_1 \leftarrow B_1 + v_i - u_i$. If we choose B_2, then we pick up an index i uniformly at random from $\{0, \ldots, l-1\}$ and update B_2 by $B_2 \leftarrow B_2 + u_{i+1} - v_i$, where we denote $u_l = u_0$. The pseudocode of this algorithm is shown in `UpdateWithCycle` (Algorithm 4). We note that, if the length of the directed cycle is two, then `UpdateWithCycle` coincides with `UpdateViaStrongBasisExchange`.

Algorithm 4 `UpdateWithCycle`$(\beta_1, B_1, \beta_2, B_2, C)$.

Input: $\beta_1, \beta_2 \in \mathbb{R}_+$, two bases B_1, B_2, and a directed cycle C in the bipartite directed graph $D_{\mathcal{M}}(B_1, B_2)$

Denote by $V(C) = \{u_0, v_0, u_1, v_1, \ldots, v_{l-1}\}$ the vertices in C in this order (with $u_i \in B_1 \setminus B_2$ and $v_i \in B_2 \setminus B_1$ for each i)
Flip a coin with `Heads` probability $\dfrac{\beta_2}{\beta_1 + \beta_2}$
if *coin flipped* `Heads` **then**
  Pick an index i uniformly at random from $\{0, \ldots, l-1\}$
  $B_1 \leftarrow B_1 + v_i - u_i$
else
  Pick an index i uniformly at random from $\{0, \ldots, l-1\}$
  $B_2 \leftarrow B_2 + u_{i+1} - v_i$ `// We define` $u_l = u_0$.

In order to show the validity of the algorithm, we use the following two lemmas.

▶ **Lemma 12.** *Given two bases B_1 and B_2 and a directed cycle C in the bipartite directed graph $D_{\mathcal{M}}(B_1, B_2)$, the procedure* `UpdateWithCycle` *updates B_1 and B_2 to B_1' and B_2', respectively, so that $\mathbb{E}[\beta_1 \mathbf{1}_{B_1'} + \beta_2 \mathbf{1}_{B_2'}] = \beta_1 \mathbf{1}_{B_1} + \beta_2 \mathbf{1}_{B_2}$.*

Proof. Recall that C traverses $u_0, v_0, u_1, v_1, \dots, v_{l-1}$ in this order, where $u_i \in B_1 \setminus B_2$ and $v_i \in B_2 \setminus B_1$ for each i. In the procedure `UpdateWithCycle`, we obtain $B_1' = B_1 + v_i - u_i$ for some $i \in \{0, \dots, l-1\}$ and $B_2' = B_2$ with probability $\beta_2/(\beta_1 + \beta_2)$, and we obtain $B_1' = B_1$ and $B_2' = B_2 + u_{i+1} - v_i$ for some $i \in \{0, \dots, l-1\}$ with probability $\beta_1/(\beta_1 + \beta_2)$. Thus, we have the following equation:

$$\begin{aligned}\mathbb{E}[\beta_1 \mathbf{1}_{B_1'} + \beta_2 \mathbf{1}_{B_2'}] =& \frac{\beta_2}{\beta_1 + \beta_2} \left(\beta_1 \left(\mathbf{1}_{B_1} + \frac{1}{l} \sum_{i=0}^{l-1} (\mathbf{1}_{v_i} - \mathbf{1}_{u_i}) \right) + \beta_2 \mathbf{1}_{B_2} \right) \\ &+ \frac{\beta_1}{\beta_1 + \beta_2} \left(\beta_1 \mathbf{1}_{B_1} + \beta_2 \left(\mathbf{1}_{B_2} + \frac{1}{l} \sum_{i=0}^{l-1} (\mathbf{1}_{u_{i+1}} - \mathbf{1}_{v_i}) \right) \right) \\ =& \beta_1 \mathbf{1}_{B_1} + \beta_2 \mathbf{1}_{B_2}.\end{aligned}$$

This completes the proof. ◀

▶ **Lemma 13** ([18, Lemma VI.2])**.** *Let $x \in \mathbb{R}_+^n$ be a non-negative vector and $\mathbf{X} = (X_1, \dots, X_n)$ be a non-negative vector-valued random variable satisfying the following properties:*

- *$\mathbb{E}[\mathbf{X}] = x$, and*
- *$\mathbf{X} - x$ has at most one positive coordinate and at most one negative coordinate.*

Then, we have $\mathbb{E}[F(\mathbf{X})] \geq F(x)$ for any function F that is a multilinear extension of some submodular function.

By combining these lemmas, we obtain the following proposition, which shows the validity of `UpdateWithCycle`.

▶ **Proposition 14.** *Let $x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}$ be a point represented by a convex combination of the characteristic vectors of t bases of a matroid $\mathcal{M}$. Suppose that the procedure* `UpdateWithCycle` *updates B_1 and B_2 to B_1' and B_2' using a directed cycle in $D_{\mathcal{M}}(B_1, B_2)$. Let $B_i' = B_i$ for $i \in \{3, \dots, t\}$ and let $x' = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i'}$. Then, we obtain $\mathbb{E}[F(x')] \geq F(x)$ for any function F that is a multilinear extension of some submodular function.*

Proof. It is obvious that $x' - x$ has at most one positive coordinate and at most one negative coordinate, since only two coordinate are involved in `UpdateWithCycle`, and exactly one of them increases and the other decreases. We also see that $\mathbb{E}[x'] = x$ holds by Lemma 12. Therefore, Lemma 13 shows that $\mathbb{E}[F(x')] \geq F(x)$ for any function F that is a multilinear extension of some submodular function. ◀

5.3 Whole Algorithm

We now prove Theorem 7 by giving our fast swap rounding algorithm. See `FastMergeBases` (Algorithm 5) for the pseudocode of our algorithm.

Proof of Theorem 7. Suppose that $x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}$ is a point represented by a convex combination of the characteristic vectors of t bases of a matroid $\mathcal{M}$. We pick up two bases, say B_1 and B_2, in the representation and merge them into a single basis in the following way: until B_1 and B_2 coincide, we find a directed cycle C in $D_{\mathcal{M}}(B_1, B_2)$ using Proposition 9, and update B_1 and B_2 by `UpdateWithCycle` using C. Our algorithm repeats this process $t - 1$ times so that all the bases are merged into a single basis.

Since the correctness of this algorithm is shown by Proposition 14, it remains to analyze the independence query complexity of this rounding algorithm.

For merging two bases into a single basis, since we apply Proposition 9 at most r times, we require $O(r^{3/2}\log^{3/2}(rt))$ independence oracle queries with probability at least $1-r^{-1}t^{-2}$. Furthermore, since we apply this procedure $t-1$ times in our swap rounding algorithm, the entire algorithm requires $O(tr^{3/2}\log^{3/2}(rt))$ independence oracle queries with probability at least $1-(rt)^{-1}$. This completes the proof of Theorem 7. ◀

We can remove the condition "with probability at least $1-(rt)^{-1}$" by losing a sufficiently small approximation factor $\varepsilon > 0$. That is, we obtain Theorem 2, which we restate here.

▶ **Theorem 2.** *For any $\varepsilon > 0$, there is a randomized algorithm satisfying the following conditions:*

- *the input consists of a matroid $\mathcal{M} = (V, \mathcal{I})$ given as an independence oracle and a point x in the base polytope of $\mathcal{M}$ that is represented as a convex combination of t bases,*
- *the output is a basis S of $\mathcal{M}$ such that $\mathbb{E}[f(S)] \geq (1-\varepsilon)F(x)$ for any submodular function $f\colon 2^V \to \mathbb{R}$ and its multilinear extension F, and*
- *it uses $O(r^{3/2}t\log^{3/2}(\frac{rt}{\varepsilon}))$ independence oracle queries.*

Proof. Recall that the algorithm in Theorem 7 (Algorithm 5) uses $O(r^{3/2}t\log^{3/2}(rt))$ independence oracle queries with probability at least $1-(rt)^{-1}$. If Algorithm 5 returns a basis using $O(r^{3/2}t\log^{3/2}(rt))$ independence oracle queries, then we say that it *succeeds*. Otherwise, we say that it *fails*. By a slight modification, when the algorithm fails, we suppose that it uses $O(r^{3/2}t\log^{3/2}(rt))$ independence oracle queries and terminates without returning a basis. This modified algorithm is denoted by Algorithm 5'. Note that Algorithm 5' fails with probability at most $(rt)^{-1}$.

Let $q := \lceil \log_{(rt)^{-1}} \varepsilon \rceil = \lceil \frac{\log(1/\varepsilon)}{\log rt} \rceil = O\left(\frac{\log(rt/\varepsilon)}{\log rt}\right)$. In our algorithm, we run Algorithm 5' q times. If at least one execution of Algorithm 5' succeeds, then our algorithm returns a basis that is obtained in the first successful execution of Algorithm 5'. If all the executions of Algorithm 5' fail, then our algorithm returns an arbitrary basis. Then, we use $O(r^{3/2}t\log^{3/2}(\frac{rt}{\varepsilon}))$ independence oracle queries in total. Furthermore, the probability that all the executions of Algorithm 5' fail is at most $(rt)^{-q} \leq \varepsilon$. Therefore, the output S satisfies $\mathbb{E}[f(S)] \geq (1-\varepsilon)F(x)$ for any submodular function f and its multilinear extension F. This completes the proof. ◀

6 Submodular Maximization with Rank Oracle

In this section, we present a fast submodular maximization algorithm in the rank oracle model and prove Theorem 3. In the rank oracle setting, the input consists of a monotone submodular set function $f\colon 2^V \to \mathbb{R}_+$ given as a value oracle, and a matroid $\mathcal{M} = (V, \mathcal{I})$ given as a rank oracle. The objective is to maximize $f(S)$ subject to $S \in \mathcal{I}$. We restate Theorem 3 here.

▶ **Theorem 3.** *For any $\varepsilon > 0$, there is a randomized algorithm that achieves $(1-1/e-\varepsilon)$-approximation for maximizing a monotone submodular function subject to a matroid constraint and uses $O((n+r^{3/2})\operatorname{poly}(1/\varepsilon, \log n))$ value and rank oracle queries.*

In the same way as the independence oracle setting, our algorithm is based on continuous relaxation and rounding technique.

Algorithm 5 `FastMergeBases`$(\beta_1, B_1, \beta_2, B_2)$.

```
while B_1 \neq B_2 do
    Sample a set L of 2\sqrt{r \log(rt)} elements drawn uniformly and independently from
      B_1 \setminus B_2 with replacement.
    Sample a set R of 2\sqrt{r \log(rt)} elements drawn uniformly and independently from
      B_2 \setminus B_1 with replacement.
    a \leftarrow \emptyset
    E \leftarrow \emptyset
    for u \in L do
        v \leftarrow FindExchangeElement(\mathcal{M}, B_2, u, R)
        if v = \emptyset then
            a \leftarrow u
        else
            E \leftarrow E \cup \{(u, v)\}
    for v \in R do
        u \leftarrow FindExchangeElement(\mathcal{M}, B_1, v, L)
        if u = \emptyset then
            a \leftarrow v
        else
            E \leftarrow E \cup \{(v, u)\}
    if a = \emptyset then
        Find a directed cycle C in the bipartite directed graph (L \cup R, E)
        UpdateWithCycle(\beta_1, B_1, \beta_2, B_2, C)
    else
        if a \in B_1 \setminus B_2 then
            A \leftarrow \emptyset
            while v = FindExchangeElement(\mathcal{M}, B_2, a, B_2 \setminus (B_1 \cup A)) satisfies v \neq \emptyset
             do
                if B_1 + v - a \in \mathcal{I} then
                    UpdateViaStrongBasisExchange(\beta_1, B_1, \beta_2, B_2, v, a)
                    break
                A \leftarrow A + v
        else
            A \leftarrow \emptyset
            while u = FindExchangeElement(\mathcal{M}, B_1, a, B_1 \setminus (B_2 \cup A)) satisfies u \neq \emptyset
             do
                if B_2 + u - a \in \mathcal{I} then
                    UpdateViaStrongBasisExchange(\beta_1, B_1, \beta_2, B_2, a, u)
                    break
                A \leftarrow A + u
return B_1
```

Algorithm for the Continuous Relaxation Problem. Let F be the multilinear extension of f and let $P(\mathcal{M})$ be the matroid polytope of $\mathcal{M}$. Ene-Nguyễn [22] presented a framework to solve the continuous optimization problem $\max_{x \in P(\mathcal{M})} F(x)$ in near-linear time for several important classes of matroids. In their algorithm, they use a data structure for maintaining a maximum weight basis of the matroid, where each element has a weight and the weights are updated. In each update, the weight of exactly one element decreases, while all the other weights do not change. The data structure supports an operation that decreases the weight of an element and updates the current basis to a maximum weight basis with respect to the updated weights. This operation is called the *maximum weight basis data structure operation.* With this terminology, their result is stated as follows.

▶ **Lemma 15** (follows from Lemmas 8 and 9 in the arXiv version of [22]). *Given a non-negative monotone submodular function $f\colon 2^V \to \mathbb{R}_+$, a matroid $\mathcal{M} = (V, \mathcal{I})$ of rank r, and a parameter $\varepsilon > 0$, there is a randomized algorithm satisfying the following conditions:*

- *the algorithm finds a point $x \in P(\mathcal{M})$ represented as a convex combination of $1/\varepsilon$ bases such that $\mathbb{E}[F(x)] \geq (1 - 1/e - \varepsilon) \cdot \max\{f(T) \mid T \in \mathcal{I}\}$, where $F\colon [0,1]^V \to \mathbb{R}_+$ is the multilinear extension of f,*
- *it uses $O\left(\frac{n}{\varepsilon^5} \log^2\left(\frac{n}{\varepsilon}\right)\right)$ value oracle queries,*
- *it uses $O\left(\frac{n}{\varepsilon} \log\left(\frac{n}{\varepsilon}\right)\right)$ independence oracle queries, and*
- *it uses $O\left(\frac{r}{\varepsilon} \log^2\left(\frac{n}{\varepsilon}\right)\right)$ maximum weight basis data structure operations.*

Maximum Weight Basis Data Structure Operation. To implement a maximum weight basis data structure operation by using rank oracle queries efficiently, we use the following lemma obtained by the binary search technique of Nguyễn [36] and Chakrabarty et al. [14]; see also [39, Lemma 2].

▶ **Lemma 16** ([14, Lemma 10]; see also [39, Lemma 2] and [10]). *There is an algorithm* `FindFreeElement` *which, given a matroid $\mathcal{M} = (V, \mathcal{I})$, a weight function $w\colon V \to \mathbb{R}$, an independent set $S \in \mathcal{I}$, and $T \subseteq V \setminus S$, finds an element $u \in T$ maximizing $w(u)$ such that $S + u \in \mathcal{I}$ or otherwise determines that no such element exists, and uses $O(\log |T|)$ rank oracle queries.*

This lemma shows that the maximum weight basis data structure operation can be easily implemented in the rank oracle model as follows.

▶ **Lemma 17.** *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid given as a rank oracle and let $w\colon V \to \mathbb{R}$ be a weight function. Let $w'\colon V \to \mathbb{R}$ be a weight function such that $w'(v) < w(v)$ for some $v \in V$ and $w'(u) = w(u)$ for any $u \in V - v$. Given a maximum weight basis B of $\mathcal{M}$ with respect to w, we can compute a maximum weight basis B' of $\mathcal{M}$ with respect to w' using $\tilde{O}(1)$ rank oracle queries.*

Proof. If $v \notin B$, then $B' := B$ is a desired basis, because $w'(v) < w(v)$. Otherwise, we apply `FindFreeElement` with the weight function w' in which $S = B - v$ and $T = (V \setminus B) \cup \{v\}$. Let u be the element found by the procedure (possibly, $u = v$). Then our algorithm returns a basis $B' := B - v + u$, which is a maximum weight basis with respect to w' (see e.g., [28, Lemma 3.1] and Section 6 of the arXiv version of [9]). By Lemma 16, this algorithm requires $\tilde{O}(1)$ rank oracle queries. ◀

Putting Them Together (Proof of Theorem 3). We now prove Theorem 3. Lemma 17 shows that we can execute the maximum weight basis data structure operation using $\tilde{O}(1)$ rank oracle queries without a sophisticated data structure. Hence, by Lemma 15, we can solve the continuous optimization problem $\max_{x \in P(\mathcal{M})} F(x)$ using $\tilde{O}_\varepsilon(n)$ value and rank oracle queries, where we note that the rank oracle is more powerful than the independence oracle.

For the obtained point x, we apply our fast rounding algorithm given in Theorem 2 to obtain an integral solution. Note again that the rank oracle is more powerful than the independence oracle, and hence this rounding algorithm requires $\tilde{O}_\varepsilon(r^{3/2})$ value and rank oracle queries.

By replacing ε with $\varepsilon/2$ in the same way as in the proof of Theorem 1, we obtain a $(1 - 1/e - \varepsilon)$-approximation algorithm that uses $\tilde{O}_\varepsilon(n + r^{3/2})$ value and rank oracle queries, which completes the proof. ◀

References

1 Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8:307–328, 2004. doi:10.1023/B:JOCO.0000038913.96607.c2.

2 Yossi Azar and Iftah Gamzu. Efficient submodular function maximization under linear packing constraints. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP 2012)*, pages 38–50, 2012. doi:10.1007/978-3-642-31594-7_4.

3 Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 1497–1514, 2014. doi:10.1137/1.9781611973402.110.

4 Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. *Operations Research*, 70(5):2967–2981, 2022. doi:10.1287/opre.2021.2170.

5 Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 1138–1151, 2018. doi:10.1145/3188745.3188752.

6 Kiarash Banihashem, Leyla Biabani, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Dynamic algorithms for matroid submodular maximization. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 3485–3533, 2024. doi:10.1137/1.9781611977912.125.

7 Rafael da Ponte Barbosa, Alina Ene, Huy L Nguyễn, and Justin Ward. A new framework for distributed submodular maximization. In *Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS 2016)*, pages 645–654, 2016. doi:10.1109/FOCS.2016.74.

8 Joakim Blikstad. Breaking $O(nr)$ for matroid intersection. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 198, pages 31:1–31:17, 2021. doi:10.4230/LIPIcs.ICALP.2021.31.

9 Joakim Blikstad, Sagnik Mukhopadhyay, Danupon Nanongkai, and Ta-Wei Tu. Fast algorithms via dynamic-oracle matroids. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, pages 1229–1242, 2023. arXiv version is arXiv:2302.09796. doi:10.1145/3564246.3585219.

10 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, 2021. doi:10.1145/3406325.3451092.

11 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 1433–1452, 2014. doi:10.1137/1.9781611973402.106.

12 Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query trade-off in submodular maximization. *Mathematics of Operations Research*, 42(2):308–329, 2017. doi:10.1287/moor.2016.0809.

13 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011. doi:10.1137/080733991.

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

15 Chandra Chekuri, TS Jayram, and Jan Vondrák. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS 2015)*, pages 201–210, 2015. doi:10.1145/2688073.2688086.

16 Chandra Chekuri and Kent Quanrud. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019)*, pages 78–89, 2019. doi:10.1145/3313276.3316406.

17 Chandra Chekuri, Kent Quanrud, and Manuel R Torres. Fast approximation algorithms for bounded degree and crossing spanning tree problems. In *Proceedings of the 24th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2021)*, volume 207, pages 24:1–24:21, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.24.

18 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS 2010)*, pages 575–584, 2010. doi:10.1109/FOCS.2010.60.

19 Xi Chen and Binghui Peng. On the complexity of dynamic submodular maximization. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (FOCS 2022)*, pages 1685–1698, 2022. doi:10.1145/3519935.3519951.

20 Alina Ene and Huy L. Nguyễn. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 53:1–53:12, 2019. doi:10.4230/LIPIcs.ICALP.2019.53.

21 Alina Ene and Huy L Nguyễn. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the 30-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 274–282, 2019. doi:10.1137/1.9781611975482.18.

22 Alina Ene and Huy L. Nguyễn. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 54:1–54:14, 2019. arXiv version is arXiv:1811.07464. doi:10.4230/LIPIcs.ICALP.2019.54.

23 Alina Ene, Huy L Nguyễn, and Adrian Vladu. Submodular maximization with matroid and packing constraints in parallel. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019)*, pages 90–101, 2019. doi:10.1145/3313276.3316389.

24 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

25 Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS 2012)*, pages 659–668, 2012. doi:10.1109/FOCS.2012.55.

26 ML Fisher, GL Nemhauser, and LA Wolsey. An analysis of approximations for maximizing submodular set functions–II. *Mathematical Programming Studies*, 8:73–87, 1978. doi:10.1007/BFb0121195.

27 Monika Henzinger, Paul Liu, Jan Vondrák, and Da Wei Zheng. Faster submodular maximization for several classes of matroids. In *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261, pages 74:1–74:18, 2023. doi:10.4230/LIPIcs.ICALP.2023.74.

28 Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis. Recoverable robust optimization with commitment, 2023. arXiv:2306.08546.

29 Silvio Lattanzi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakub M Tarnawski, and Morteza Zadimoghaddam. Fully dynamic algorithm for constrained submodular optimization. *Advances in Neural Information Processing Systems 33: Proceedings of the 34th Annual Conference on Neural Information Processing Systems (Neurips 2020)*, 33:12923–12933, 2020.

30 Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. *Advances in Neural Information Processing Systems 35: Proceedings of the 36th Annual Conference on Neural Information Processing Systems (Neurips 2022)*, 35:17473–17487, 2022.

31 Paul Liu and Jan Vondrák. Submodular optimization in the MapReduce model. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69, pages 18:1–18:10, 2019. doi:10.4230/OASIcs.SOSA.2019.18.

32 Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, volume 29(1), 2015. doi:10.1609/aaai.v29i1.9486.

33 Morteza Monemizadeh. Dynamic submodular maximization. *Advances in Neural Information Processing Systems 33: Proceedings of the 34th Annual Conference on Neural Information Processing Systems (Neurips 2020)*, 33:9806–9817, 2020.

34 George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978. doi:10.1287/moor.3.3.177.

35 George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14:265–294, 1978. doi:10.1007/BF01588971.

36 Huy L Nguyễn. A note on Cunningham's algorithm for matroid intersection. *arXiv preprint arXiv:1904.04129*, 2019. doi:10.48550/arXiv.1904.04129.

37 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

38 Tatsuya Terao. Faster matroid partition algorithms. In *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261, pages 104:1–104:20, 2023. doi:10.4230/LIPIcs.ICALP.2023.104.

39 Ta-Wei Tu. Subquadratic weighted matroid intersection under rank oracles. In *Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC 2022)*, volume 248, pages 63:1–63:14, 2022. doi:10.4230/LIPIcs.ISAAC.2022.63.