

Nearly-Linear Query Algorithm for Submodular Maximization subject to a Matroid Constraint

Tatsuya Terao*

Abstract

We consider monotone submodular maximization subject to a matroid constraint. The seminal continuous greedy algorithm of Calinescu, Chekuri, Pál, and Vondrák [SICOMP'11] achieves the tight approximation of $(1 - 1/e)$. Since then, a line of works by Badanidiyuru and Vondrák [SODA'14], Buchbinder, Feldman, and Schwartz [SODA'15 & MOR'17], Kobayashi and Terao [ICALP'24], Buchbinder and Feldman [FOCS'24 & SICOMP'25], and Buchbinder and Feldman [STOC'25] has developed efficient $(1 - 1/e - \varepsilon)$ -approximation algorithms for this problem.

We present a combinatorial randomized $(1 - 1/e - \varepsilon)$ -approximation algorithm that requires $\tilde{O}_\varepsilon(n)$ independence-oracle and value-oracle queries, where n is the number of elements in the matroid. This is the first $(1 - 1/e - \varepsilon)$ -approximation algorithm with both independence-oracle and value-oracle query complexities nearly-linear in n and polynomial in $1/\varepsilon$. To achieve this, we develop a new local search algorithm, which we call *protected local search*, based on the potential function of Buchbinder and Feldman [FOCS'24 & SICOMP'25].

*Research Institute for Mathematical Sciences, Kyoto University. E-mail: ttatsuya@kurims.kyoto-u.ac.jp

1 Introduction

Submodular maximization is a fundamental and well-studied problem in theoretical computer science and combinatorial optimization. In particular, the problem of maximizing a nonnegative monotone submodular function subject to a matroid constraint is one of the most extensively investigated settings.

The study of submodular maximization was initiated by a seminal work of Fisher, Nemhauser, and Wolsey in the 1970's [NWF78, FNW78, NW78]. They showed that, for monotone submodular maximization, the natural greedy algorithm achieves a $(1 - 1/e)$ -approximation for a cardinality constraint and a $\frac{1}{2}$ -approximation for a matroid constraint. They [NW78] also showed that, for any constant $\delta > 0$, achieving a $(1 - 1/e + \delta)$ -approximation requires exponentially many value-oracle queries, even under a cardinality 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, and Vondrák [CCPV11] developed a new framework based on continuous optimization and a rounding technique. Their algorithm first obtains a $(1 - 1/e)$ -approximate solution to the problem of maximizing the multilinear extension of f , a natural continuous extension of f , via the continuous greedy algorithm, and then rounds it to an integral solution via pipage rounding [AS04]. 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 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 $1 - 1/e - \varepsilon$.

In submodular maximization subject to a matroid constraint, it is standard to assume that the objective function f is accessed through a value oracle and that the family $\mathcal{I} \subseteq 2^N$ of independent sets is accessed through an independence oracle. In this setting, the efficiency of an algorithm is usually measured by the number of queries it makes to these oracles.

Badanidiyuru and Vondrák [BV14] 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 $\tilde{O}_\varepsilon(nr)$ value-oracle queries and $\tilde{O}_\varepsilon(n + r^2)$ independence-oracle queries¹, where n is the number of elements in the matroid and r is the rank of the matroid. Their algorithm combines a fast implementation of the continuous greedy algorithm, which uses $\tilde{O}_\varepsilon(nr)$ value and $\tilde{O}_\varepsilon(n)$ independence oracle queries, with the swap rounding of Chekuri, Vondrák, and Zenklusen [CVZ10], which rounds the resulting convex combination of $1/\varepsilon$ bases using $O(r^2/\varepsilon)$ independence oracle queries.

Buchbinder, Feldman, and Schwartz [BFS17] developed another fast implementation of the continuous greedy algorithm, which exhibits a trade-off between the numbers of value and independence oracle queries. In particular, they obtained a $(1 - 1/e - \varepsilon)$ -approximation algorithm using $\tilde{O}_\varepsilon(n\sqrt{r})$ value oracle queries and $\tilde{O}_\varepsilon(n\sqrt{r} + r^2)$ independence oracle queries. As in the algorithm of Badanidiyuru-Vondrák, the $\tilde{O}_\varepsilon(r^2)$ term in the independence query complexity is due to swap rounding.

Kobayashi and Terao [KT24] improved the independence query complexity of the swap rounding of Chekuri, Vondrák, and Zenklusen [CVZ10] to $\tilde{O}_\varepsilon(r^{3/2})$. Combined with the algorithm of Buchbinder, Feldman, and Schwartz, this yields a $(1 - 1/e - \varepsilon)$ -approximation algorithm using $\tilde{O}_\varepsilon(n\sqrt{r})$ value and independence oracle queries.

In another line of work, Filmus and Ward [FW14] presented the first combinatorial $(1 - 1/e)$ -approximation algorithm, which requires no rounding. Their algorithm is based on *non-oblivious local search*, i.e., local search guided by an auxiliary potential function instead of f , which is estimated by sampling and thus makes the algorithm randomized.

All of the above $(1 - 1/e - \varepsilon)$ -approximation algorithms for a matroid constraint are randomized. In a breakthrough work, Buchbinder and Feldman [BF25a] presented the first deterministic $(1 - 1/e - \varepsilon)$ -approximation algorithm, which is based on the non-oblivious local search and uses $\tilde{O}(nr \cdot 2^{O(1/\varepsilon)})$

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

Reference	Value queries	Independence queries	Type
Calinescu, Chekuri, Pál and Vondrák [CCPV11]	$\text{poly}(n)$	$\text{poly}(n)$	R
Filmus and Ward [FW14]	$\text{poly}(n)$	$\text{poly}(n)$	R
Badanidiyuru and Vondrák [BV14]	$\tilde{O}_\varepsilon(nr)$	$\tilde{O}_\varepsilon(n + r^2)$	R
Buchbinder, Feldman, and Schwartz [BFS17]	$\tilde{O}_\varepsilon(n\sqrt{r})$	$\tilde{O}_\varepsilon(n\sqrt{r} + r^2)$	R
Kobayashi and Terao [KT24]	$\tilde{O}_\varepsilon(n\sqrt{r})$	$\tilde{O}_\varepsilon(n\sqrt{r})$	R
Buchbinder and Feldman [BF25a]	$\tilde{O}(nr \cdot 2^{O(1/\varepsilon)})$	$\tilde{O}_\varepsilon(nr)$	D
Buchbinder and Feldman [BF25a]	$\tilde{O}((n + r\sqrt{n}) \cdot 2^{O(1/\varepsilon)})$	$\tilde{O}_\varepsilon(n + r\sqrt{n})$	R
Buchbinder and Feldman [BF25b]	$\tilde{O}(n \cdot 2^{O(1/\varepsilon^4)})$	$\tilde{O}_\varepsilon(n + r^{3/2})$	R
Theorem 2	$\tilde{O}(n \cdot 2^{O(1/\varepsilon)})$	$\tilde{O}_\varepsilon(n)$	D
Theorem 1	$\tilde{O}_\varepsilon(n)$	$\tilde{O}_\varepsilon(n)$	R

Table 1: Oracle-query complexities for maximizing a nonnegative monotone submodular function subject to a general matroid constraint. Here, D and R denote deterministic and randomized algorithms, respectively. Note that the $\tilde{O}$ notation hides polylogarithmic factors in n , and the $\tilde{O}_\varepsilon$ notation additionally hides polynomial factors in ε^{-1} .

value oracle queries and $\tilde{O}_\varepsilon(nr)$ independence oracle queries. With randomization, these query complexities can be reduced to $\tilde{O}((n + r\sqrt{n}) \cdot 2^{O(1/\varepsilon)})$ and $\tilde{O}_\varepsilon(n + r\sqrt{n})$, respectively.

Subsequently, Buchbinder and Feldman [BF25b] introduced the extended multilinear extension, which enables deterministic implementations of both the fractional maximization step and the rounding step of the continuous greedy framework. As a by-product, they obtained a randomized $(1 - 1/e - \varepsilon)$ -approximation algorithm using $\tilde{O}(n \cdot 2^{O(1/\varepsilon^4)})$ value oracle queries and $\tilde{O}_\varepsilon(n + r^{3/2})$ independence oracle queries, which is nearly-linear in n when $r = O(n^{2/3})$. We note that, unlike the algorithms of [BV14, BFS17, KT24], the value query complexities of these algorithms depend exponentially on $1/\varepsilon$.

Buchbinder and Feldman [BF25a] posed, as an open question, whether the query complexity can be further improved to be nearly-linear in n for all values of r , rather than only for $r = \tilde{O}(n^{2/3})$ as in [BF25b].

1.1 Our Results

Table 1 summarizes the best known complexities of $(1 - 1/e - \varepsilon)$ -approximation algorithms for monotone submodular maximization subject to a matroid constraint, together with our new results. We develop a new local search algorithm, which we call *protected local search*, based on the potential function of Buchbinder and Feldman [BF25a]. Our main result is the following theorem.

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

This is the first $(1 - 1/e - \varepsilon)$ -approximation algorithm with both independence-oracle and value-oracle query complexities nearly-linear in n and polynomial in $1/\varepsilon$. In particular, this resolves the open question posed by Buchbinder and Feldman [BF25a].

Our algorithm can also be derandomized at the cost of an exponential dependence on $1/\varepsilon$ in the value-oracle query complexity.

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

1.2 Additional Related Work

Buchbinder and Feldman [BF25b] also gave a deterministic algorithm that computes a value V with $(1 - 1/e - \varepsilon)f(\text{OPT}) \leq V \leq f(\text{OPT})$ using $\tilde{O}(n \cdot 2^{O(1/\varepsilon^4)})$ value-oracle and $\tilde{O}_\varepsilon(n)$ independence-oracle queries. Unlike our algorithm, however, it only estimates the optimal value and does not output a feasible solution.

Recently, faster algorithms based on the continuous greedy framework have been developed for several important classes of matroids. Specifically, $(1 - 1/e - \varepsilon)$ -approximation algorithms using nearly-linear value-oracle queries and arithmetic operations are known for graphic and partition matroids [EN19] and for laminar and transversal matroids [HLVZ23].

Ganz Rozenman, Kulik, Schwartz, and Singh [GRKSS26] proposed a different approach based on a stochastic Poisson process, which requires neither discretization nor rounding. For partition matroids, their approach yields a $(1 - 1/e - \varepsilon)$ -approximation algorithm using $O(n \cdot \text{poly}(1/\varepsilon))$ value oracle queries, which is linear in n .

2 Preliminaries

Basic Notation. Let $\mathbb{R}_+$ denote the set of non-negative real numbers. Throughout this paper, let $\mathcal{N}$ be a finite set and let n denote its cardinality. For a set $A \subseteq \mathcal{N}$ and an element $v \in \mathcal{N}$, we will often write $A + v := A \cup \{v\}$ and $A - v := A \setminus \{v\}$. For a positive integer k , let $[k] := \{1, \dots, k\}$. For a set function h , a set S , and an element v , we write $h(v \mid S) := h(S + v) - h(S)$. Note that $h(v \mid S) = 0$ if $v \in S$.

We assume without loss of generality that $0 < \varepsilon \leq 1/4$. We say that an event happens *with high probability* (w.h.p.) if it happens with probability at least $1 - (\varepsilon/n)^c$, where $c > 0$ is an arbitrarily large constant that affects only the constant factors in the query complexity.

Submodular Functions. Let $f: 2^{\mathcal{N}} \rightarrow \mathbb{R}_+$ be a set function on a finite ground set $\mathcal{N}$ 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 $A, B \subseteq \mathcal{N}$. The function is *monotone* if $f(A) \leq f(B)$ for any subsets $A \subseteq B \subseteq \mathcal{N}$. In this paper, we only consider monotone submodular functions. For such a function, $f(v \mid A) \geq f(v \mid B)$ holds for any $A \subseteq B \subseteq \mathcal{N}$ and $v \in \mathcal{N}$; we refer to this inequality simply as submodularity.

Matroids. A pair $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ of a finite set $\mathcal{N}$ and a non-empty set family $\mathcal{I} \subseteq 2^{\mathcal{N}}$ 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 $S \subseteq \mathcal{N}$ 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 $S \subseteq \mathcal{N}$, the rank of S is defined as the size of a largest independent set contained in S . Inclusion-wise maximal independent sets are called *bases*. Note that every basis has the same size. Throughout this paper, r denotes the rank of $\mathcal{M}$. Before running the algorithm, we remove all loops of $\mathcal{M}$, i.e., elements $u \in \mathcal{N}$ with $\{u\} \notin \mathcal{I}$. Hence, we assume without loss of generality that $\mathcal{M}$ has no loops.

Problem. We consider the problem of maximizing a monotone submodular function $f: 2^{\mathcal{N}} \rightarrow \mathbb{R}_+$ subject to a matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, that is, $\max\{f(S) \mid S \in \mathcal{I}\}$. We denote by $\text{OPT} \in \mathcal{I}$ an optimal solution.

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 $S \subseteq \mathcal{N}$ and outputs $f(S)$. We also assume that we access a matroid $\mathcal{M}$ through an oracle. Given a subset $S \subseteq \mathcal{N}$, an *independence oracle* outputs whether $S \in \mathcal{I}$ or not.

2.1 Basic Facts on Submodular Functions

We use the following basic inequalities for submodular functions.

Fact 3 (From [BF25a, Lemmas 2.2 and 2.3 in the journal version]). *Let $f: 2^{\mathcal{N}} \rightarrow \mathbb{R}_+$ be a submodular function.*

(a) *If $S \subseteq T \subseteq \mathcal{N}$, then $\sum_{u \in T \setminus S} f(u \mid T - u) \leq f(T) - f(S)$.*

(b) *For all $S, T \subseteq \mathcal{N}$, $\sum_{u \in T} f(u \mid S) \geq f(S \cup T) - f(S)$.*

2.2 Binary Search Technique

Nguyen [Ngu19] and Chakrabarty, Lee, Sidford, Singla, and Wong [CLS⁺19] independently presented the so-called *binary search technique*, which is useful for improving the query complexity of matroid problems.

Lemma 4 (Binary Search Technique [Ngu19, CLS⁺19]). *Let $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ be a matroid. Let $S \in \mathcal{I}, T \subseteq S$, and $v \in \mathcal{N} \setminus S$. If there exists an element $u \in T$ such that $S + v - u \in \mathcal{I}$, then such an element can be found using $O(\log |T|)$ independence oracle queries.*

The idea behind Lemma 4 is to perform binary search on T . We skip the proof in this paper.

2.3 The Lifted Matroid and Potential Function of Buchbinder and Feldman

In this subsection, we describe the lifted matroid and the potential function, which were used in the non-oblivious local search algorithm of Buchbinder and Feldman [BF25a], and present some of their useful properties. Our algorithm also performs local search on this lifted matroid, guided by the same potential function, but it updates the solution in an entirely different way, as we explain in Section 3.

Throughout the rest of this paper, we set $\ell = \lceil 1/\varepsilon \rceil$, so that $1/\ell \leq \varepsilon$ and $\ell = O(1/\varepsilon)$. We use the following coefficients of Buchbinder and Feldman [BF25a]:

$$\alpha_0 = 0, \quad \alpha_k = \frac{(1 + 1/\ell)^{k-1}}{\binom{\ell-1}{k-1}} \quad (k \in [\ell]).$$

We also write $\kappa = (\ell + 1)\alpha_\ell = \ell(1 + 1/\ell)^\ell$. Here, we have $1 - \ell/\kappa = 1 - (1 + 1/\ell)^{-\ell} \geq 1 - 1/e - \varepsilon$ (see Fact 7).

Lifted Matroid. We also use the lifted matroid construction of Buchbinder and Feldman [BF25a]. Let $\mathcal{N}' = \mathcal{N} \times [\ell]$. The element (u, i) is called the *copy* of u with color i . We use u, w for elements of $\mathcal{N}$ and x, y for elements of $\mathcal{N}'$. For $J \subseteq [\ell]$ and $S \subseteq \mathcal{N}'$, let

$$\pi_J(S) = \{u \in \mathcal{N} \mid (u, i) \in S \text{ for some } i \in J\} \quad \text{and} \quad \pi(S) = \pi_{[\ell]}(S).$$

Let $\mathcal{M}' = (\mathcal{N}', \mathcal{I}')$, where $S \in \mathcal{I}'$ if and only if S contains at most one copy of each element of $\mathcal{N}$ and $\pi(S) \in \mathcal{I}$. An independence oracle query to $\mathcal{M}'$ can be answered using at most one independence oracle query to $\mathcal{M}$, since a set containing two copies of the same element does not belong to $\mathcal{I}'$. Note that $|\pi(S)| = |S|$ for every $S \in \mathcal{I}'$, and that $S \times \{j\} \in \mathcal{I}'$ for every $S \in \mathcal{I}$ and $j \in [\ell]$. The following lemma guarantees that the lifting preserves the matroid structure.

Lemma 5 (From [BF25a, Lemma 3.5 in the journal version]). $\mathcal{M}'$ is a matroid, and it has the same rank r as $\mathcal{M}$.

Our algorithm runs on the lifted matroid $\mathcal{M}'$ instead of $\mathcal{M}$: it maintains a basis B of $\mathcal{M}'$ and finally outputs $\pi(B)$, which is a basis of $\mathcal{M}$ since $\pi(B) \in \mathcal{I}$ and $|\pi(B)| = |B| = r$.

Potential Function. We also use the potential function $g: 2^{\mathcal{N}'} \rightarrow \mathbb{R}$ of Buchbinder and Feldman [BF25a],² which is defined for every $S \subseteq \mathcal{N}'$ as

$$g(S) = \sum_{J \subseteq [\ell]} \alpha_{|J|} \cdot f(\pi_J(S)).$$

The following property of g was proved.

Lemma 6 (From [BF25a, Lemma 3.4 in the journal version]). *The function g is non-negative, monotone, and submodular.*

In the analysis, we also use the auxiliary function $z: 2^{\mathcal{N}'} \rightarrow \mathbb{R}$, defined by

$$z(S) = \sum_{J \subseteq [\ell]} \frac{|J|}{\ell} \alpha_{|J|} \cdot f(\pi_J(S)).$$

Since $0 \leq \frac{|J|}{\ell} \alpha_{|J|} \leq \alpha_{|J|}$ for every $J \subseteq [\ell]$ and each function $S \rightarrow f(\pi_J(S))$ is monotone, we have

$$0 \leq z(X + y) - z(X) \leq g(X + y) - g(X) \quad \text{for any } X \subseteq \mathcal{N}' \text{ and } y \in \mathcal{N}'. \quad (1)$$

We use the following properties of the coefficients α_k . Here, recall that $\kappa = \ell(1 + 1/\ell)^\ell$.

Fact 7.

- (a) $(\ell - k)\alpha_{k+1} = k(1 + 1/\ell)\alpha_k$ for $1 \leq k < \ell$.
- (b) For every $i \in [\ell]$, $\sum_{J \ni i} \alpha_{|J|} = \kappa - \ell$. We also have $\sum_J \frac{|J|}{\ell} \alpha_{|J|} = \kappa - \ell$.
- (c) $\sum_J \alpha_{|J|} = O(\ell^2)$.
- (d) $\ell \leq \kappa \leq e\ell$ and $\ell/\kappa \leq 1/e + 1/\ell$. In particular, $1 - \ell/\kappa \geq 1 - 1/e - \varepsilon$.

We defer the proof of Fact 7 to Appendix A.1. Next, we describe two basic properties of the potential function g .

Observation 8. For $x = (u, i) \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, $g(x \mid X - x) = \sum_{J \ni i} \alpha_{|J|} \cdot f(u \mid \pi_J(X - x))$.

Observation 9. For every $X \subseteq \mathcal{N}'$, we have $g(X) \leq O(\ell^2) \cdot f(\pi(X))$. For every $y = (u, i) \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, we have $g(y \mid X) \leq (\kappa - \ell) \cdot f(\{u\})$.

We defer the proofs of Observations 8 and 9 to Appendix A.1.

²This function is denoted by g' in [BF25a].

2.4 Comparison Inequality

In this subsection, we derive a bound on the potential function g , which is the key inequality in the analyses of both the algorithm of Buchbinder and Feldman [BF25a] and our algorithm. We remark that a special case of Corollary 11 also plays an important role in their analysis (see Section 3).

Lemma 10.

- (a) For every $C \in \mathcal{I}'$, $\sum_{x \in C} g(x \mid C - x) + z(C) \leq \kappa \cdot f(\pi(C))$.
- (b) For every $H \subseteq \mathcal{N}'$, $\frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{u \in \text{OPT}} g((u, j) \mid H - (u, j)) \geq (\kappa - \ell) \cdot f(\text{OPT}) - z(H)$.

We defer the proof of Lemma 10 to Appendix A.2. Adding the two inequalities of Lemma 10 yields the following.

Corollary 11. For every $C \in \mathcal{I}'$ and every $H \subseteq \mathcal{N}'$,

$$(\kappa - \ell) \cdot f(\text{OPT}) \leq \kappa \cdot f(\pi(C)) + D_1 + D_2,$$

$$\text{where } D_1 = z(H) - z(C) \text{ and } D_2 = \frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{u \in \text{OPT}} g((u, j) \mid H - (u, j)) - \sum_{x \in C} g(x \mid C - x).$$

2.5 Estimating $g(x \mid X - x)$

Our algorithm needs to evaluate $g(x \mid X - x)$. By Observation 8, $g(x \mid X - x)$ is a sum of $2^{\ell-1}$ terms, each of which can be computed using two value-oracle queries; hence, $g(x \mid X - x)$ can be computed exactly using $2^\ell = 2^{O(1/\varepsilon)}$ value-oracle queries (see Lemma 12). Our deterministic algorithm (Theorem 2) uses this exact computation. To improve the dependence on $1/\varepsilon$ from exponential to polynomial, our randomized algorithm instead estimates $g(x \mid X - x)$ by sampling (see Lemma 13).

Lemma 12 (Exact computation). For any $x \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, the value $g(x \mid X - x)$ can be computed deterministically using 2^ℓ value oracle queries.

Proof. Let $x = (u, i)$. By Observation 8, $g(x \mid X - x)$ is determined by the values $f(\pi_J(X - x) + u)$ and $f(\pi_J(X - x))$ for the $2^{\ell-1}$ sets $J \subseteq [\ell]$ that contain i . $\square$

Lemma 13. There is a randomized algorithm $\text{ESTIMATE}(x, X)$ which, given $x \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, uses $O(\varepsilon^{-3} \log(n/\varepsilon))$ value oracle queries and w.h.p. outputs $\hat{g}$ such that $\frac{g(x \mid X - x)}{1 + \varepsilon} \leq \hat{g} \leq g(x \mid X - x)$. Since each call uses fresh randomness, this holds even if (x, X) is chosen depending on the outcomes of previous calls.

Roughly speaking, the procedure ESTIMATE draws $O(\ell \varepsilon^{-2} \log(n/\varepsilon))$ independent random samples, computes a value for each sample using value oracle queries, and outputs the average of the computed values. We defer the proof of Lemma 13 to Appendix A.3.

We call an estimate *good* if it satisfies the inequalities in Lemma 13.

3 Technical Overview: Comparison with the Local Search of Buchbinder and Feldman [BF25a]

In this section, we first explain the analysis of the local search of Buchbinder and Feldman in terms of Corollary 11, and then outline how our algorithm uses the same corollary in its analysis. Corollary 11 states that, for every $C \in \mathcal{I}'$ and every $H \subseteq \mathcal{N}'$,

$$(\kappa - \ell) \cdot f(\text{OPT}) \leq \kappa \cdot f(\pi(C)) + D_1 + D_2,$$

where $D_1 = z(H) - z(C)$ and $D_2 = \frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{u \in \text{OPT}} g((u, j) \mid H - (u, j)) - \sum_{x \in C} g(x \mid C - x)$.

The Analysis of Local Search of Buchbinder and Feldman [BF25a]. Roughly speaking, they compute a set $S \in \mathcal{T}'$ by local search, and their analysis essentially applies the inequality in Corollary 11 with $C = H = S$. In this case, we have $D_1 = 0$. Moreover, the inequality satisfied by the set S obtained by local search [BF25a, Inequality (3.4)] implies that $D_2 \leq \varepsilon \ell \cdot f(\text{OPT})$.³ Combining these bounds and dividing by $\kappa \geq \ell$, we obtain $f(\pi(S)) \geq (1 - \ell/\kappa - \varepsilon) \cdot f(\text{OPT})$. Since $1 - \ell/\kappa \geq 1 - 1/e - \varepsilon$ by Fact 7(d), this yields a $(1 - 1/e - O(\varepsilon))$ -approximation.

However, their local search for finding a set S satisfying [BF25a, Inequality (3.4)] uses $\tilde{O}(nr \cdot 2^{O(1/\varepsilon)})$ queries, or $\tilde{O}((n + r\sqrt{n}) \cdot 2^{O(1/\varepsilon)})$ queries with randomization, which is not nearly-linear in n when r is large.

The Analysis of Our Protected Local Search. Our *protected local search* performs $T = O(\varepsilon^{-3})$ phases, each of which is a single pass over decreasing thresholds Θ in which every element y is processed at most once: y is swapped in for an element x of the current basis only if removing x costs less than about Θ , and otherwise x is *protected*, i.e., kept in the basis until the end of the phase; see Algorithm 1 for details.

Fix a single phase, and let B_0 and B_1 be the bases at the beginning and the end of this phase, respectively. Let $\Delta = g(B_1) - g(B_0)$ be the increase in g during this phase. We show that $\Delta \geq 0$ and $f(\pi(B_0)) \geq \left(1 - \frac{1}{e} - O(\varepsilon)\right) \cdot f(\text{OPT}) - O(\Delta)$ (Lemma 17). To prove the second inequality, we apply Corollary 11 to the sets C and H at the end of the phase. Since $C \subseteq B_0$ and f is monotone, we have $f(\pi(B_0)) \geq f(\pi(C))$, and hence it suffices to upper-bound $D_1 + D_2$. For the upper bound on $D_1 + D_2$, see the paragraph *Plan of the Proof of Lemma 17* in Section 5.

Lemma 17 means that if $\Delta \leq \varepsilon \cdot f(\text{OPT})$, then $\pi(B_0)$ is a $(1 - 1/e - O(\varepsilon))$ -approximate solution; otherwise, the phase increases g by more than $\varepsilon \cdot f(\text{OPT})$. Since $\Delta = g(B_1) - g(B_0) \geq 0$ for every phase and $g(B) \leq O(\ell^2) \cdot f(\text{OPT})$ for every basis B of $\mathcal{M}'$ (by Observation 9), the total increase of g over T phases is at most $O(\ell^2) \cdot f(\text{OPT})$. Hence, for $T = \ell^3$, some phase satisfies $\Delta \leq O(\ell^2/T) \cdot f(\text{OPT}) = O(\varepsilon) \cdot f(\text{OPT})$ since $\ell \geq 1/\varepsilon$, and its starting basis B_0 yields a $(1 - 1/e - O(\varepsilon))$ -approximation. Our algorithm returns the best among the starting bases of the T phases.

4 Our Protected Local Search Algorithm

4.1 Parameters

We first construct a basis B_{init} of $\mathcal{M}$ greedily using n independence queries, and compute $f_{\max} = \max_{u \in \mathcal{N}} f(\{u\})$ using n value queries. Since we assume that $\mathcal{M}$ has no loops, we have $f_{\max} \leq f(\text{OPT})$. If $f_{\max} = 0$, then $0 \leq f(\emptyset) \leq f(\{u\}) = 0$ by monotonicity, and Fact 3(b) gives $f(X) \leq \sum_{u \in X} f(\{u\}) = 0$ for every $X \subseteq \mathcal{N}$; hence we simply return B_{init} .

The parameters in our algorithm are listed in Table 2. The thresholds are $\Theta_t = \Theta_{\max}(1 + \varepsilon)^{-(t-1)}$ for $t \in [t_{\max}]$, where $t_{\max}$ is the largest t with $\Theta_t \geq \Theta_{\min}$. Note that $\Theta_{t_{\max}} < (1 + \varepsilon)\Theta_{\min}$. Since $\Theta_{\max}/\Theta_{\min} \leq \kappa r \leq e\ell r$ by Fact 7(d) and $\ln(1 + \varepsilon) \geq \varepsilon/2$ for $\varepsilon < 1$, the number of thresholds is $t_{\max} = O\left(\frac{\log(\ell r)}{\varepsilon}\right) = O\left(\frac{\log(n/\varepsilon)}{\varepsilon}\right)$. By Observation 9, we have $g(y \mid X) \leq (\kappa - \ell) \cdot f_{\max} = \Theta_{\max}$ for every $y \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$.

³To see this, note that Inequality (3.4) of [BF25a] states that $\sum_{v \in T} g(v \mid S - v) - \sum_{u \in S} g(u \mid S - u) \leq \varepsilon' \cdot g(\text{OPT}')$ for every $T \in \mathcal{T}'$, where $\varepsilon' = \varepsilon/(e(1 + \ln \ell))$ and OPT' is a set in $\mathcal{T}'$ maximizing g . Applying this with $T = \text{OPT} \times \{j\} \in \mathcal{T}'$ for each $j \in [\ell]$ and averaging over j , we obtain $D_2 \leq \varepsilon' \cdot g(\text{OPT}')$. Combining this with $g(\text{OPT}') \leq e\ell(1 + \ln \ell) \cdot f(\text{OPT})$, shown at the end of the proof of [BF25a, Lemma 3.6], yields the claimed bound $D_2 \leq \varepsilon \ell \cdot f(\text{OPT})$.

Table 2: Parameters of the algorithm.

Symbol	Value	Role
ℓ	$\lceil 1/\varepsilon \rceil$	number of colors
$\Theta_{\max}$	$(\kappa - \ell) \cdot f_{\max}$	upper limit for thresholds
$\Theta_{\min}$	$f_{\max}/r$	lower limit for thresholds
Θ_t	$\Theta_{\max} \cdot (1 + \varepsilon)^{-(t-1)}, t \in [t_{\max}]$	thresholds, in decreasing order
T	ℓ^3	number of phases

Algorithm 1 PHASE(B_0)**Input:** a basis B_0 of $\mathcal{M}'$

```

1:  $A \leftarrow \emptyset$ ;  $C \leftarrow B_0$ ; mark every  $y \in \mathcal{N}'$  as unprocessed
   Throughout:  $B = A \cup C$ ,  $H = B_0 \cup A$ ,  $\gamma(y) = g(y \mid H - y)$ ,  $\lambda(x) = g(x \mid C - x)$ .
2: for  $\Theta = \Theta_1, \Theta_2, \dots, \Theta_{t_{\max}}$  do
3:   for each unprocessed  $y \in \mathcal{N}'$ , in arbitrary order do
4:     if  $\hat{\gamma}(y) < \Theta$  then continue ▷  $y$  stays unprocessed
5:     while  $y \notin A$  and  $A + y \in \mathcal{I}'$  do ▷ processing of  $y$ 
6:       if  $y \in B$  then
7:          $x \leftarrow y$ 
8:       else
9:          $x \leftarrow$  an element of  $C \setminus A$  such that  $B - x + y \in \mathcal{I}'$  ▷ by the binary search
        technique (Lemma 4)
10:      if  $\hat{\lambda}(x) > \Theta/(1 + \varepsilon)^2$  then
11:         $A \leftarrow A + x$  ▷ protect  $x$ 
12:      else
13:         $C \leftarrow C - x$ ;  $A \leftarrow A + y$  ▷ exchange  $x$  for  $y$ 
14:      break
15:   mark  $y$  as processed
16: return  $B_1 = A \cup C$ 

```

4.2 One Phase

Our algorithm consists of T phases, each of which is described in Algorithm 1.

A phase starts from a basis B_0 of $\mathcal{M}'$ and maintains the following sets.

A . The elements that stay in the current basis until the end of the phase; A never shrinks. An element enters A in one of two ways: it is *protected* if it is added to A in Line 11, and it is *inserted* if it is added to A in Line 13.

$C \subseteq B_0$. The elements of B_0 that have not been deleted. C never grows.

$B = A \cup C$. The current basis.

$H = B_0 \cup A$. The set on which gains are evaluated.

The sets A and C may intersect, since a protected element belongs to both. For the current sets, we write

$$\gamma(y) = g(y \mid H - y) \quad (\text{gain of } y) \quad \text{and} \quad \lambda(x) = g(x \mid C - x) \quad (\text{loss of } x).$$

We call $\gamma(y)$ the gain of y also when $y \in H$, and $\lambda(x)$ the loss of x also when $x \notin C$. We write $\hat{\gamma}(y)$ and $\hat{\lambda}(x)$ for the estimates returned by ESTIMATE(y, H) and ESTIMATE(x, C), respectively.

Algorithm 2 Protected local search

```
1: Construct a basis  $B_{\text{init}}$  of  $\mathcal{M}$ ; compute  $f_{\max}$ 
2: if  $f_{\max} = 0$  then return  $B_{\text{init}}$ 
3: set the parameters as in Table 2
4:  $B^{(0)} \leftarrow B_{\text{init}} \times \{1\}$  ▷ every element of the initial basis gets color 1
5: for  $t = 0, 1, \dots, T - 1$  do
6:    $B^{(t+1)} \leftarrow \text{PHASE}(B^{(t)})$ 
7: return  $\pi(B^{(t^*)})$ , where  $t^* \in \arg \max_{0 \leq t < T} f(\pi(B^{(t)}))$ 
```

In words, for each threshold, the phase scans the elements that are still unprocessed. An element $y \in \mathcal{N}'$ whose estimated gain $\widehat{\gamma}(y)$ is below the threshold Θ is skipped, and it is scanned again at the next threshold. Otherwise, the element y is *processed*: as long as $y \notin A$ and $A + y$ is independent, the phase sets $x = y$ if $y \in B$, and otherwise picks an element $x \in C \setminus A$ such that $B - x + y \in \mathcal{I}'$. The phase protects x if the estimated loss $\widehat{\lambda}(x)$ exceeds $\Theta/(1 + \varepsilon)^2$, and otherwise it exchanges x for y and stops. Afterwards, y is marked as processed and is not examined again in this phase.

The following lemma plays a crucial role in the analysis of our algorithm.

Observation 14 (Monotonicity of $\gamma(y)$ and $\lambda(x)$). *During a phase, H only grows and C only shrinks. Hence, by submodularity of g (Lemma 6), the gain $\gamma(y)$ of each y can only decrease, and the loss $\lambda(x)$ of each x can only increase.*

4.3 The Whole Algorithm

The whole algorithm is described in Algorithm 2. It first constructs a basis B_{init} of $\mathcal{M}$, computes $f_{\max}$, and sets the parameters as in Table 2. It then assigns color 1 to every element of B_{init} , which yields a basis $B^{(0)} = B_{\text{init}} \times \{1\}$ of $\mathcal{M}'$, and runs T phases starting from $B^{(0)}$, where each phase takes the basis produced by the previous phase as its input. Finally, it returns the best projected basis $\pi(B^{(t)})$ with respect to f .

4.4 Invariants and Query Complexity of One Phase

In this subsection, we show some invariants maintained during a phase and then bound the number of oracle queries used in a phase. Both the invariants and the bound hold regardless of whether the estimates are good.

Lemma 15 (Invariants). *Throughout a phase, $A \in \mathcal{I}'$, $C \subseteq B_0$, $B = A \cup C$ is a basis of $\mathcal{M}'$, and $C \subseteq B \subseteq H$. Moreover, A and H only grows and C only shrinks, and each iteration of the while loop adds a new element to A . In particular, the body of the while loop is executed at most r times in total during a phase.*

Proof. The sets A and C change only in iterations of the while loop, and the invariants hold at the beginning of the phase, where $A = \emptyset$ and $B = C = H = B_0$. We show that each iteration maintains them. Consider an iteration processing y ; by the loop condition, $y \notin A$ and $A + y \in \mathcal{I}'$.

We first show that the element x is well-defined and satisfies $x \in C \setminus A$. If $y \in B$, then $x = y \in C \setminus A$, because $B = A \cup C$ and $y \notin A$. Now suppose that $y \notin B$. It remains to show that an element x as in Line 9 always exists. To prove this, we use the augmentation property of the matroid $\mathcal{M}'$: by repeatedly adding elements of B to $A + y$, we obtain a basis B' of $\mathcal{M}'$ with $A + y \subseteq B' \subseteq B + y$. Since $|B'| = |B|$ and $y \in B'$, we have $B' = B - x' + y$ for some $x' \in B$, and $x' \notin A$ because $A \subseteq B'$. Hence $x' \in B \setminus A = C \setminus A$ satisfies $B - x' + y \in \mathcal{I}'$, and thus an element x as in Line 9 always exists.

We next consider the update.

- If x is protected (Line 11), then A becomes $A + x \subseteq B$, which is independent. Since $x \in C \subseteq B_0$, none of $B = A \cup C$, C , and $H = B_0 \cup A$ changes.

- If $x \neq y$ is exchanged for y (Line 13), then $y \notin B$, and B becomes $(A+y) \cup (C-x) = B - x + y$. This is a basis of $\mathcal{M}'$ that contains the new set $A+y$. The set H becomes $H+y$, and C becomes $C-x$.
- If $x = y$ is exchanged for y , then y moves from C to A , so B and H do not change, and $A+y \subseteq B$ is independent.

In all cases, $C \subseteq B_0$ and $C \subseteq B \subseteq A \cup B_0 = H$ still hold, A and H do not shrink, and C does not grow. Finally, every iteration adds an element not in A (namely, x or y) to A . Since $A \in \mathcal{I}'$ has at most r elements, there are at most r iterations. $\square$

Lemma 16 (Query complexity of one phase). *PHASE(B_0) makes $O(\varepsilon^{-2}n \log(n/\varepsilon))$ calls to ESTIMATE. Moreover, PHASE(B_0) uses $O(\varepsilon^{-5}n \log^2(n/\varepsilon))$ value-oracle queries and $O(\varepsilon^{-1}n + r \log r)$ independence-oracle queries.*

Proof. We first count the value-oracle queries, which are used only in the calls to ESTIMATE. For each of the $t_{\max}$ thresholds, the gain of each unprocessed element is estimated once, so the number of gain estimates is at most $|\mathcal{N}'| \cdot t_{\max} \leq n \ell t_{\max} = O(\varepsilon^{-2}n \log(n/\varepsilon))$. A loss is estimated once per iteration of the while loop, so by Lemma 15 the number of loss estimates is at most $r \leq n$. Hence, PHASE(B_0) calls ESTIMATE $O(\varepsilon^{-2}n \log(n/\varepsilon))$ times. By Lemma 13, each call uses $O(\varepsilon^{-3} \log(n/\varepsilon))$ value-oracle queries, and thus the number of value-oracle queries is $O(\varepsilon^{-5}n \log^2(n/\varepsilon))$.

We next count the independence-oracle queries. Recall that each independence-oracle query to $\mathcal{M}'$ is answered by at most one independence-oracle query to $\mathcal{M}$. Independence-oracle queries are used only to check the loop condition $A+y \in \mathcal{I}'$ and to find x in Line 9.

- Each check of whether $A+y \in \mathcal{I}'$ either leads to an iteration of the while loop, which happens at most r times by Lemma 15, or ends the processing of y , which happens at most once for each y , i.e., at most $|\mathcal{N}'| \leq n \ell$ times.
- Line 9 is executed at most once per iteration, i.e., at most r times. We apply Lemma 4 to $\mathcal{M}'$ with $S = B$, $T = C \setminus A$, and $v = y$. A desired element exists by the proof of Lemma 15, and $|T| \leq |B| = r$, so each execution uses $O(\log r)$ queries.

Hence, the number of independence-oracle queries is $O(n \ell + r \log r) = O(\varepsilon^{-1}n + r \log r)$. $\square$

5 Analysis of One Phase

Throughout this section, we consider a single phase, and we assume that every estimate made during it is good.

Lemma 17 (Phase lemma). *Let B_0 and B_1 be the starting and ending bases of a phase in which all estimates are good, and let $\Delta = g(B_1) - g(B_0)$. Then $\Delta \geq 0$ and*

$$f(\pi(B_0)) \geq \left(1 - \frac{1}{e} - O(\varepsilon)\right) \cdot f(\text{OPT}) - O(\Delta).$$

Notation. For a processed element y , let $\Theta(y)$ be the threshold at which it is processed. Let E be the set of elements inserted by exchanges (Line 13) during the phase, and let P be the set of protected (Line 11) elements. For $y \in E$, let x_y be the element deleted from C in its exchange, and let

$$a_y = \gamma(y) \quad \text{and} \quad b_y = \lambda(x_y),$$

where γ and λ are evaluated immediately before that exchange. Unless stated otherwise, A , C , H , γ , and λ refer to the end of the phase.

Plan of the Proof of Lemma 17. We apply Corollary 11 to the sets C and H at the end of the phase. Since $C \subseteq B_0$ and f is monotone, we have $f(\pi(B_0)) \geq f(\pi(C))$, and hence it suffices to upper-bound $D_1 + D_2$. We first collect basic properties of a phase: Claim 18 bounds the gain of each element in terms of the thresholds, and Claim 19 shows that each exchange increases g by at least an ε -fraction of the gain of the inserted element, which in particular implies $\Delta \geq 0$. Next, we upper-bound $D_1 = z(H) - z(C)$ by Claim 20: each exchange changes $z(H) - z(C)$ by at most $O(a_y)$ while increasing g by at least $\Omega(\varepsilon) \cdot a_y$ (Claim 19), and hence $D_1 = O(\Delta/\varepsilon)$. Finally, we upper-bound D_2 by Claims 21, 22, and 23. Claim 22 is the heart of the analysis; it plays the role of the approximate local optimality [BF25a, Inequality (3.4)] in the analysis of Buchbinder and Feldman [BF25a], and bounds the total gain of any independent set by the sum of the thresholds at which the elements of A entered A . These thresholds are bounded by the losses of the protected elements (Claim 21) and by Δ for the inserted elements (Claim 20), which yields Claim 23.

Using Observation 14, we first show that the gain of each element is bounded in terms of the thresholds.

Claim 18. *If y is processed at threshold Θ , then $\Theta \leq \gamma(y) \leq (1 + O(\varepsilon)) \cdot \Theta$ throughout its processing. Moreover, every element y that is unprocessed at the end of the phase satisfies $\gamma(y) \leq (1 + O(\varepsilon)) \cdot \Theta_{\min}$ at the end of the phase.*

Proof. While y is processed, $H = B_0 \cup A$ can change only at an exchange in Line 13, after which the processing of y immediately ends. Indeed, a protection adds an element of $C \subseteq B_0$ to A , and hence does not change H . Hence, $\gamma(y) = g(y \mid H - y)$ does not change during the processing of y . The lower bound follows from $\gamma(y) \geq \hat{\gamma}(y) \geq \Theta$, where the second inequality holds because y was not skipped in Line 4. For the upper bound, if $\Theta = \Theta_{\max}$, then it follows from $\gamma(y) \leq \Theta_{\max}$ (see Section 4.1). Otherwise, y was skipped in Line 4 at the previous threshold $(1 + \varepsilon)\Theta$, i.e., $\hat{\gamma}(y) < (1 + \varepsilon)\Theta$ held at that time. Hence, at that time, $\gamma(y) \leq (1 + \varepsilon)\hat{\gamma}(y) < (1 + \varepsilon)^2\Theta$, and by Observation 14, $\gamma(y)$ has not increased since then. Therefore, in both cases, $\Theta \leq \gamma(y) \leq (1 + O(\varepsilon)) \cdot \Theta$ holds throughout the processing of y .

Let y be an element that is unprocessed at the end of the phase. Then, y was skipped in Line 4 at the last threshold $\Theta_{t_{\max}} < (1 + \varepsilon)\Theta_{\min}$, i.e., $\hat{\gamma}(y) < \Theta_{t_{\max}}$ held at that time. Hence, at that time, $\gamma(y) \leq (1 + \varepsilon)\hat{\gamma}(y) < (1 + \varepsilon)^2\Theta_{\min}$, and by Observation 14, $\gamma(y)$ has not increased since then. Therefore, $\gamma(y) \leq (1 + O(\varepsilon)) \cdot \Theta_{\min}$ at the end of the phase. $\square$

For $y \in E$, let Δ_y be the increase of $g(B)$ caused by the exchange that inserts y (Line 13), i.e., $\Delta_y = g(B - x_y + y) - g(B)$, where B denotes the basis just before the exchange. Recall that $\Delta = g(B_1) - g(B_0)$.

Claim 19.

(a) *In each iteration of the while loop during the processing of y , if $y \in B$ at that time, then $x = y$ is chosen in Line 9, and y is protected (Line 11) in that iteration. Consequently, just before each exchange in Line 13, we have $y \notin B$ and $x \neq y$.*

(b) *For every $y \in E$, we have $\Theta(y) \leq a_y$ and $b_y \leq a_y/(1 + \varepsilon)$. Moreover,*

$$\Delta_y \geq a_y - b_y \geq \frac{\varepsilon}{1 + \varepsilon} \cdot a_y > 0.$$

(c) $\Delta = \sum_{y \in E} \Delta_y \geq 0$.

Proof. (a) Suppose that $y \in B$ in an iteration of the while loop during the processing of y ; then $x = y$ by Line 9. Since $y \in B = A \cup C$ and $y \notin A$ by the loop condition, we have $y \in C \subseteq B_0 \subseteq H$. Thus, $C - y \subseteq H - y$, and hence

$$\lambda(y) = g(y \mid C - y) \geq g(y \mid H - y) = \gamma(y) \geq \Theta,$$

where the first inequality follows from the submodularity of g (Lemma 6), and the last inequality follows from Claim 18. Therefore, $\widehat{\lambda}(y) \geq \lambda(y)/(1+\varepsilon) \geq \Theta/(1+\varepsilon) > \Theta/(1+\varepsilon)^2$, and hence the test in Line 10 succeeds for $x = y$, and y is protected in Line 11. Consequently, an exchange is performed only in an iteration with $y \notin B$; since x is chosen from $C \setminus A \subseteq B$ in Line 9, we also have $x \neq y$.

(b) Let $y \in E$, let $x = x_y$, and let B be the basis just before the exchange that inserts y . Since the exchange is performed during the processing of y at threshold $\Theta(y)$, Claim 18 gives $\Theta(y) \leq \gamma(y) = a_y$. Since the exchange is performed only when the test in Line 10 fails, we have $\widehat{\lambda}(x) \leq \Theta(y)/(1+\varepsilon)^2$, and hence $b_y = \lambda(x) \leq (1+\varepsilon)\widehat{\lambda}(x) \leq \Theta(y)/(1+\varepsilon) \leq a_y/(1+\varepsilon)$. Since $x \in B$, we have

$$\Delta_y = g(B - x + y) - g(B) = g(y \mid B - x) - g(x \mid B - x).$$

Since $B = A \cup C \subseteq H$ and $y \notin B$ by (a), we have $B - x \subseteq H - y$; moreover, $C - x \subseteq B - x$. By the submodularity of g , we obtain $g(y \mid B - x) \geq g(y \mid H - y) = a_y$ and $g(x \mid B - x) \leq g(x \mid C - x) = b_y$, and hence $\Delta_y \geq a_y - b_y$. Finally, $a_y - b_y \geq a_y - a_y/(1+\varepsilon) = \varepsilon a_y/(1+\varepsilon)$, which is positive since $a_y \geq \Theta(y) > 0$.

(c) A protection (Line 11) adds an element of C to A , and hence does not change $B = A \cup C$. Thus, B changes only through exchanges, and since each $y \in E$ is inserted by exactly one exchange, we have $\Delta = g(B_1) - g(B_0) = \sum_{y \in E} \Delta_y$. Since $\Delta_y > 0$ for every $y \in E$ by (b), we have $\Delta \geq 0$. $\square$

The next claim upper-bounds $D_1 = z(H) - z(C)$ by tracking its change during the phase: each exchange changes it by at most $O(a_y)$, while increasing g by at least $\Omega(\varepsilon) \cdot a_y$ by Claim 19(b). The same argument also bounds $\sum_{y \in E} \Theta(y)$, since $\Theta(y) \leq a_y$. Recall that $\Theta(y)$ is the threshold at which y is processed.

Claim 20. *At the end of the phase, both D_1 and $\sum_{y \in E} \Theta(y)$ are $O(\Delta/\varepsilon)$.*

Proof. By Claim 19, $\Delta = \sum_{y \in E} \Delta_y \geq \frac{\varepsilon}{1+\varepsilon} \sum_{y \in E} a_y$, so $\sum_{y \in E} a_y = O(\Delta/\varepsilon)$. Since $\Theta(y) \leq a_y$ by Claim 19(b), this gives $\sum_{y \in E} \Theta(y) \leq \sum_{y \in E} a_y = O(\Delta/\varepsilon)$.

To show $D_1 = O(\Delta/\varepsilon)$, we track $z(H) - z(C)$ during the phase. At the beginning of the phase, $z(H) - z(C) = 0$ since $A = \emptyset$ and hence $H = C = B_0$. The protections (Line 11) change neither H nor C , since they only add an element of $C \subseteq B_0$ to A and $H = B_0 \cup A$. Thus, $z(H) - z(C)$ changes only through exchanges. Consider an exchange that inserts $y \in E$ and deletes x , and let H and C denote the sets just before it; the exchange replaces H by $H + y$ and C by $C - x$. By (1) and the submodularity of g , we have

$$\begin{aligned} z(H + y) - z(H) &\leq g(y \mid H) \leq g(y \mid H - y) = a_y, \\ z(C) - z(C - x) &\leq g(x \mid C - x) = b_y \leq a_y, \end{aligned}$$

where the last inequality follows from Claim 19(b). Hence, each exchange increases $z(H) - z(C)$ by at most $2a_y$. Since each $y \in E$ is inserted by exactly one exchange, we obtain $D_1 = z(H) - z(C) \leq 2 \sum_{y \in E} a_y = O(\Delta/\varepsilon)$ at the end of the phase. $\square$

Using Observation 14, we show that every protected element stays in C and keeps a large loss.

Claim 21. *Every protected element $x \in P$ belongs to the final C . Moreover, if x was protected at threshold Θ , then $\Theta \leq (1 + O(\varepsilon)) \cdot \lambda(x)$ for the final C .*

Proof. Let $x \in P$. When x is protected (Line 11), x belongs to C and is added to A . Since an element is deleted from C only if it belongs to $C \setminus A$, and x stays in A , the element x is never deleted; hence x belongs to the final C .

Next, suppose that x was protected at threshold Θ . At that time, we have $\lambda(x) \geq \widehat{\lambda}(x) > \Theta/(1+\varepsilon)^2$. Since $\lambda(x)$ does not decrease afterwards (Observation 14), we obtain $\Theta < (1+\varepsilon)^2 \cdot \lambda(x) \leq (1 + O(\varepsilon)) \cdot \lambda(x)$ for the final C . $\square$

The next claim is the heart of the analysis. It plays the role of the approximate local optimality [BF25a, Inequality (3.4)] in the analysis of Buchbinder and Feldman, and is the main tool for upper-bounding D_2 .

Claim 22. *Let $\theta_1 \geq \theta_2 \geq \dots \geq \theta_{|A|}$ be the thresholds at which the elements of the final A entered A , listed in order of entry, and set $\theta_i = \Theta_{\min}$ for $|A| < i \leq r$. Then, at the end of the phase, every $S \in \mathcal{I}'$ satisfies*

$$\sum_{y \in S} \gamma(y) \leq (1 + O(\varepsilon)) \sum_{i=1}^r \theta_i.$$

The proof of Claim 22 compares the order in which the elements of S are processed with the order in which elements enter A . Let y_i be the i -th processed element of S , and consider the moment when its processing ends. At this moment, each of $y_1, \dots, y_i$ is spanned by A , and hence $|A| \geq i$, i.e., the i -th element to enter A did so no later than this moment. Since thresholds decrease over time, this gives $\theta_i \geq \Theta(y_i)$. On the other hand, Claim 18 gives $\gamma(y_i) \leq (1 + O(\varepsilon)) \cdot \Theta(y_i)$ at the end of the phase. Combining these two inequalities and summing over i yields the inequality of Claim 22.

Proof of Claim 22. We say that y is *spanned* by A if $y \in A$ or $A + y \notin \mathcal{I}'$. The processing of an element y ends only when the loop condition fails (i.e., y is spanned by A) or after an exchange (Line 13) that inserts it into A , so every processed element is spanned by A at the moment its processing ends. It remains spanned afterwards, because A only grows.

Let $y_1, \dots, y_k$ be the elements of S that are processed during the phase (possibly $k < |S|$), in the order in which they are processed. Fix $i \in [k]$. When the processing of y_i ends, the set $\{y_1, \dots, y_i\} \subseteq S$ is independent (as $S \in \mathcal{I}'$) and all of its elements are spanned by A . Since $A \subseteq B$ is independent, if $|A| < i$ at that moment, then the augmentation property gives some $y_h \notin A$ ($h \leq i$) with $A + y_h \in \mathcal{I}'$, which contradicts that y_h is spanned by A . Hence, $|A| \geq i$ when the processing of y_i ends. Since A never shrinks, the i -th element to enter A did so no later than the end of the processing of y_i . Since thresholds never increase over time, $\theta_i \geq \Theta(y_i)$. Therefore, at the end of the phase, $\gamma(y_i) \leq (1 + O(\varepsilon)) \cdot \Theta(y_i) \leq (1 + O(\varepsilon)) \cdot \theta_i$, where the first inequality holds since $\gamma(y_i) \leq (1 + O(\varepsilon)) \cdot \Theta(y_i)$ while y_i is processed (Claim 18) and $\gamma(y_i)$ does not increase afterwards (Observation 14).

Let $y_{k+1}, \dots, y_{|S|}$ be the remaining elements of S , which are unprocessed, in arbitrary order; note that $|S| \leq r$ since $S \in \mathcal{I}'$. For each $k < i \leq |S|$, Claim 18 gives $\gamma(y_i) \leq (1 + O(\varepsilon)) \cdot \Theta_{\min} \leq (1 + O(\varepsilon)) \cdot \theta_i$ at the end of the phase, where the last inequality holds since $\theta_i \geq \Theta_{\min}$ (every threshold is at least $\Theta_{\min}$). Combining these bounds and using $\theta_i \geq 0$ for every i , we obtain $\sum_{y \in S} \gamma(y) = \sum_{i=1}^{|S|} \gamma(y_i) \leq$

$$(1 + O(\varepsilon)) \sum_{i=1}^{|S|} \theta_i \leq (1 + O(\varepsilon)) \sum_{i=1}^r \theta_i. \quad \square$$

Using Claim 22, we obtain the following inequality, which is used to upper-bound D_2 .

Claim 23. *At the end of the phase,*

$$\frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{u \in \text{OPT}} \gamma((u, j)) \leq (1 + O(\varepsilon)) \sum_{x \in C} \lambda(x) + O\left(\sum_{y \in E} \Theta(y) + f_{\max}\right).$$

Proof. For each $j \in [\ell]$, we have $\text{OPT} \times \{j\} \in \mathcal{I}'$, and hence Claim 22 with $S = \text{OPT} \times \{j\}$ gives

$$\sum_{u \in \text{OPT}} \gamma((u, j)) \leq (1 + O(\varepsilon)) \sum_{i=1}^r \theta_i.$$

Each element of A entered A either as a protected element $x \in P$, at a threshold of at most $(1 + O(\varepsilon)) \cdot \lambda(x)$ by Claim 21, or as an inserted element $y \in E$, at threshold $\Theta(y)$. The remaining $r - |A|$ values θ_i are equal to $\Theta_{\min}$, and they sum to at most $r\Theta_{\min} = f_{\max}$. Hence,

$$\sum_{i=1}^r \theta_i \leq (1 + O(\varepsilon)) \sum_{x \in P} \lambda(x) + \sum_{y \in E} \Theta(y) + f_{\max} \leq (1 + O(\varepsilon)) \sum_{x \in C} \lambda(x) + \sum_{y \in E} \Theta(y) + f_{\max},$$

where the last inequality holds since $P \subseteq C$ by Claim 21 and $\lambda(x) \geq 0$ for every $x \in C \setminus P$ by the monotonicity of g . Combining this with the first inequality, we have, for every $j \in [\ell]$,

$$\sum_{u \in \text{OPT}} \gamma((u, j)) \leq (1 + O(\varepsilon)) \sum_{x \in C} \lambda(x) + O\left(\sum_{y \in E} \Theta(y) + f_{\max}\right).$$

Since the right-hand side does not depend on j , summing this inequality over $j \in [\ell]$ and dividing by ℓ yields the claim. $\square$

We now prove Lemma 17 using the claims established above.

Proof of Lemma 17. We have $g(B_1) - g(B_0) = \Delta \geq 0$ by Claim 19(c). We apply Corollary 11 to the final sets $C \in \mathcal{I}'$ (recall that $C \subseteq B_0$) and H . In the notation of this section, $D_2 = \frac{1}{\ell} \sum_j \sum_{u \in \text{OPT}} \gamma((u, j)) - \sum_{x \in C} \lambda(x)$. By Claims 20 and 23,

$$D_1 + D_2 \leq O(\varepsilon) \sum_{x \in C} \lambda(x) + O\left(\frac{\Delta}{\varepsilon}\right) + O(f_{\max}).$$

By Lemma 10(a) and $z(C) \geq 0$, we have $\sum_{x \in C} \lambda(x) \leq \kappa \cdot f(\pi(C))$. Plugging these bounds into Corollary 11 and dividing by κ , we obtain

$$\left(1 - \frac{\ell}{\kappa}\right) \cdot f(\text{OPT}) \leq (1 + O(\varepsilon)) \cdot f(\pi(C)) + O\left(\frac{\Delta}{\varepsilon \kappa}\right) + O\left(\frac{f_{\max}}{\kappa}\right).$$

By Fact 7(d), we have $1 - \ell/\kappa \geq 1 - 1/e - \varepsilon$ and $\kappa \geq \ell \geq 1/\varepsilon$. Together with $f_{\max} \leq f(\text{OPT})$, this gives

$$\left(1 - \frac{1}{e} - O(\varepsilon)\right) \cdot f(\text{OPT}) \leq (1 + O(\varepsilon)) \cdot f(\pi(C)) + O(\Delta).$$

Dividing by $1 + O(\varepsilon)$, we obtain $f(\pi(C)) \geq (1 - 1/e - O(\varepsilon)) \cdot f(\text{OPT}) - O(\Delta)$. Finally, $C \subseteq B_0$ and monotonicity of f give $f(\pi(B_0)) \geq f(\pi(C))$. Therefore, we obtain

$$f(\pi(B_0)) \geq f(\pi(C)) \geq \left(1 - \frac{1}{e} - O(\varepsilon)\right) \cdot f(\text{OPT}) - O(\Delta).$$

$\square$

6 Proof of the Main Theorem

In this section, we present the approximation guarantee and the query complexity of Algorithm 2 (Theorem 24), which implies Theorem 1.

Theorem 24. *Algorithm 2 returns a basis S of $\mathcal{M}$ such that, w.h.p., $f(S) \geq (1 - 1/e - O(\varepsilon)) \cdot f(\text{OPT})$. It uses $O(\varepsilon^{-8} n \log^2(n/\varepsilon))$ value oracle queries and $O(\varepsilon^{-4} n + \varepsilon^{-3} r \log r)$ independence oracle queries.*

Proof. Since π maps bases of $\mathcal{M}'$ to bases of $\mathcal{M}$, the output is a basis of $\mathcal{M}$.

Probability. By Lemma 16, regardless of the outcomes of the estimates, the algorithm calls ESTIMATE $O(T\varepsilon^{-2} n \log(n/\varepsilon)) = \text{poly}(n/\varepsilon)$ times. By Lemma 13, each call is good w.h.p., even conditioned on the previous calls. Hence, by the union bound over the calls, w.h.p. all estimates are good. In the rest of the proof, we assume this.

Approximation Guarantee. Let $\Delta_t = g(B^{(t+1)}) - g(B^{(t)})$, which is non-negative by Lemma 17. We have

$$\sum_{t=0}^{T-1} \Delta_t = g(B^{(T)}) - g(B^{(0)}) \leq g(B^{(T)}) \leq O(\ell^2) \cdot f(\text{OPT}),$$

where the last inequality follows from Observation 9 and $\pi(B^{(T)}) \in \mathcal{I}$. Since $T = \ell^3$, some phase t satisfies $\Delta_t \leq O(\ell^2/T) \cdot f(\text{OPT}) = O(\varepsilon) \cdot f(\text{OPT})$. By Lemma 17 applied to this phase,

$$f(\pi(B^{(t)})) \geq \left(1 - \frac{1}{e} - O(\varepsilon)\right) \cdot f(\text{OPT}) - O(\Delta_t) = \left(1 - \frac{1}{e} - O(\varepsilon)\right) \cdot f(\text{OPT}).$$

Since the algorithm returns the best starting basis, the output also satisfies this bound.

Query Complexity. By Lemma 16 and $T = O(\varepsilon^{-3})$, the T phases use $O(\varepsilon^{-8}n \log^2(n/\varepsilon))$ value oracle queries and $O(\varepsilon^{-4}n + \varepsilon^{-3}r \log r)$ independence oracle queries in total. The preprocessing uses $O(n)$ value and independence oracle queries, and choosing the best starting basis uses T value oracle queries. $\square$

6.1 Deterministic Algorithm.

We obtain Theorem 2 by replacing every call ESTIMATE in Algorithm 2 with the exact computation of $g(x \mid X - x)$ in Lemma 12; all other steps are deterministic. Since exact values are always good, the proof of Theorem 24 applies without the probability argument, and the number of independence oracle queries does not change. Each of the $O(T\varepsilon^{-2}n \log(n/\varepsilon))$ calls to ESTIMATE now uses $2^\ell = 2^{O(1/\varepsilon)}$ value oracle queries, and hence the total number of value oracle queries is $O(T\varepsilon^{-2}n \log(n/\varepsilon)) \cdot 2^{O(1/\varepsilon)} = O(n \log n \cdot 2^{O(1/\varepsilon)})$.

Declaration of Generative AI Use. The main ideas behind our protected local search algorithm and its analysis were developed largely with the assistance of GPT-6 Astra (OpenAI). We also used GPT-6 Astra (OpenAI), Claude Opus 5.5, and Claude Fable 5.1 (Anthropic) to assist in drafting the manuscript and improving its language and presentation. We carefully verified and revised the proofs and take full responsibility for the paper.

References

- [AS04] 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.
- [BF25a] Niv Buchbinder and Moran Feldman. Deterministic algorithm and faster algorithm for submodular maximization subject to a matroid constraint. *SIAM Journal on Computing*, pages FOCS24–85, 2025. Preliminary version in FOCS’24.
- [BF25b] Niv Buchbinder and Moran Feldman. Extending the extension: Deterministic algorithm for non-monotone submodular maximization. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC 2025)*, pages 1130–1141, 2025.
- [BFS17] 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. Preliminary version in SODA’15.
- [BV14] 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.

- [CCPV11] 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.
- [CLS⁺19] 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.
- [CVZ10] 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.
- [DP09] Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.
- [EN19] 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.
- [FNW78] ML Fisher, GL Nemhauser, and LA Wolsey. An analysis of approximations for maximizing submodular set functions–II. *Mathematical Programming Studies*, 8:73–87, 1978.
- [FW14] Yuval Filmus and Justin Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM Journal on Computing*, 43(2):514–542, 2014.
- [GRKSS26] Amit Ganz Rozenman, Ariel Kulik, Roy Schwartz, and Mohit Singh. A Poisson process for submodular maximization. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC 2026)*, pages 2152–2163, 2026.
- [HLVZ23] 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.
- [KT24] Yusuke Kobayashi and Tatsuya Terao. Subquadratic submodular maximization with a general matroid constraint. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297, pages 100:1–100:19, 2024.
- [Ngu19] Huy L Nguyễn. A note on Cunningham’s algorithm for matroid intersection. *arXiv preprint arXiv:1904.04129*, 2019.
- [NW78] 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.
- [NWF78] 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.

A Omitted Proofs

In this appendix, we restate the lemmas whose proofs were omitted from the main body, and provide their proofs.

A.1 Omitted Proofs in Section 2.3

Fact 7.

- (a) $(\ell - k)\alpha_{k+1} = k(1 + 1/\ell)\alpha_k$ for $1 \leq k < \ell$.
- (b) For every $i \in [\ell]$, $\sum_{J \ni i} \alpha_{|J|} = \kappa - \ell$. We also have $\sum_J \frac{|J|}{\ell} \alpha_{|J|} = \kappa - \ell$.
- (c) $\sum_J \alpha_{|J|} = O(\ell^2)$.
- (d) $\ell \leq \kappa \leq e\ell$ and $\ell/\kappa \leq 1/e + 1/\ell$. In particular, $1 - \ell/\kappa \geq 1 - 1/e - \varepsilon$.

Proof of Fact 7. (a) $\alpha_{k+1}/\alpha_k = (1 + 1/\ell) \binom{\ell-1}{k-1} / \binom{\ell-1}{k} = (1 + 1/\ell) k/(\ell - k)$.

(b) There are $\binom{\ell-1}{k-1}$ sets $J \ni i$ of size k in the sum, and $\binom{\ell-1}{k-1} \alpha_k = (1 + 1/\ell)^{k-1}$. Hence, the sum is $\sum_{k=1}^{\ell} (1 + 1/\ell)^{k-1} = \ell((1 + 1/\ell)^{\ell} - 1) = \kappa - \ell$. For the second identity, there are $\binom{\ell}{k}$ sets J of size k in the sum, and $\binom{\ell}{k} \frac{k}{\ell} = \binom{\ell-1}{k-1}$. Hence, the sum equals $\sum_{k=1}^{\ell} \binom{\ell-1}{k-1} \alpha_k = \sum_{k=1}^{\ell} (1 + 1/\ell)^{k-1}$, which is $\kappa - \ell$ as shown above.

(c) There are $\binom{\ell}{k}$ sets of size k in the sum. Since $\binom{\ell}{k} = \frac{\ell}{k} \binom{\ell-1}{k-1}$, we have $\binom{\ell}{k} \alpha_k = \frac{\ell}{k} (1 + 1/\ell)^{k-1} \leq \ell(1 + 1/\ell)^{k-1} \leq e\ell$, where the last inequality uses $(1 + 1/\ell)^{k-1} \leq (1 + 1/\ell)^{\ell} \leq e$. Summing over $k \in [\ell]$ gives the claim.

(d) For every $\ell > 0$, we have $(1 + 1/\ell)^{\ell} \leq e \leq (1 + 1/\ell)^{\ell+1}$. The first inequality gives $\kappa \leq e\ell$, and the second gives $\ell/\kappa = (1 + 1/\ell)^{-\ell} \leq (1 + 1/\ell)/e \leq 1/e + 1/\ell$. $\square$

Observation 8. For $x = (u, i) \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, $g(x \mid X - x) = \sum_{J \ni i} \alpha_{|J|} \cdot f(u \mid \pi_J(X - x))$.

Proof of Lemma 8. By the definition of g ,

$$g(x \mid X - x) = g(X + x) - g(X - x) = \sum_{J \subseteq [\ell]} \alpha_{|J|} \left(f(\pi_J(X + x)) - f(\pi_J(X - x)) \right).$$

We consider the value $f(\pi_J(X + x)) - f(\pi_J(X - x))$. If $i \notin J$, then $\pi_J(X + x) = \pi_J(X - x)$, and hence the term is zero. If $i \in J$, then $\pi_J(X + x) = \pi_J(X - x) + u$, and hence $f(\pi_J(X + x)) - f(\pi_J(X - x)) = f(u \mid \pi_J(X - x))$. Combining these yields the claim. $\square$

Observation 9. For every $X \subseteq \mathcal{N}'$, we have $g(X) \leq O(\ell^2) \cdot f(\pi(X))$. For every $y = (u, i) \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, we have $g(y \mid X) \leq (\kappa - \ell) \cdot f(\{u\})$.

Proof of Observation 9. The first inequality holds since

$$g(X) = \sum_{J \subseteq [\ell]} \alpha_{|J|} \cdot f(\pi_J(X)) \leq \left(\sum_{J \subseteq [\ell]} \alpha_{|J|} \right) \cdot f(\pi(X)) = O(\ell^2) \cdot f(\pi(X)),$$

where the inequality follows from $\alpha_{|J|} \geq 0$ and $f(\pi_J(X)) \leq f(\pi(X))$ (by $\pi_J(X) \subseteq \pi(X)$ and the monotonicity of f), and the equality follows from Fact 7(c). For the second inequality, let $y = (u, i)$. Then

$$g(y \mid X) \leq g(y \mid \emptyset) = \sum_{J \ni i} \alpha_{|J|} \cdot f(u \mid \emptyset) = (\kappa - \ell) \cdot f(u \mid \emptyset) \leq (\kappa - \ell) \cdot f(\{u\}),$$

where the first inequality follows from the submodularity of g (Lemma 6), the first equality follows from Observation 8 and $\pi_J(\emptyset) = \emptyset$, the second equality follows from Fact 7(b), and the last inequality follows from $f(\emptyset) \geq 0$. $\square$

A.2 Omitted Proofs in Section 2.4: Comparison Inequality

Lemma 10.

(a) For every $C \in \mathcal{I}'$, $\sum_{x \in C} g(x \mid C - x) + z(C) \leq \kappa \cdot f(\pi(C))$.

(b) For every $H \subseteq \mathcal{N}'$, $\frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{u \in \text{OPT}} g((u, j) \mid H - (u, j)) \geq (\kappa - \ell) \cdot f(\text{OPT}) - z(H)$.

Proof of Lemma 10. (a) We have

$$\begin{aligned} \sum_{x \in C} g(x \mid C - x) &= \sum_{(u, i) \in C} \sum_{J \ni i} \alpha_{|J|} \cdot f(u \mid \pi_J(C) - u) \\ &= \sum_{J \subseteq [\ell]} \alpha_{|J|} \sum_{i \in J} \sum_{u \in \pi_{\{i\}}(C)} f(u \mid \pi_J(C) - u) \\ &\leq \sum_{J \subseteq [\ell]} \alpha_{|J|} \sum_{i \in J} \left(f(\pi_J(C)) - f(\pi_{J-i}(C)) \right), \end{aligned} \quad (2)$$

where the first equality follows from Observation 8 and $\pi_J(C - x) = \pi_J(C) - u$ for $x = (u, i) \in C$ and $J \ni i$, the second equality follows by exchanging the order of summation, and the inequality follows from Fact 3(a) with $T = \pi_J(C)$ and $S = \pi_{J-i}(C)$, since $\pi_{\{i\}}(C) = \pi_J(C) \setminus \pi_{J-i}(C)$. Here, both $\pi_J(C - x) = \pi_J(C) - u$ and $\pi_{\{i\}}(C) = \pi_J(C) \setminus \pi_{J-i}(C)$ hold because C contains at most one copy of each element.

The right-hand side of the inequality (2) plus $z(C) := \sum_{J \subseteq [\ell]} \frac{|J|}{\ell} \alpha_{|J|} \cdot f(\pi_J(C))$ equals

$$\sum_{J \subseteq [\ell]} |J| \alpha_{|J|} f(\pi_J(C)) - \sum_{J \subseteq [\ell]} \alpha_{|J|} \sum_{i \in J} f(\pi_{J-i}(C)) + \sum_{J \subseteq [\ell]} \frac{|J|}{\ell} \alpha_{|J|} f(\pi_J(C)).$$

The first and third sums together equal $\sum_K |K| (1 + 1/\ell) \alpha_{|K|} f(\pi_K(C))$. For the second sum, the pairs (J, i) with $i \in J$ correspond one-to-one to the pairs (K, i) with $i \notin K$ via $K = J - i$, and hence

$$\sum_J \alpha_{|J|} \sum_{i \in J} f(\pi_{J-i}(C)) = \sum_K \sum_{i \notin K} \alpha_{|K|+1} f(\pi_K(C)) = \sum_K (\ell - |K|) \alpha_{|K|+1} f(\pi_K(C)).$$

Therefore, the right-hand side of the inequality (2) plus $z(C)$ equals $\sum_K c_K \cdot f(\pi_K(C))$ with

$$c_K = |K| \left(1 + \frac{1}{\ell} \right) \alpha_{|K|} - (\ell - |K|) \alpha_{|K|+1}.$$

Setting $\alpha_{\ell+1} = 0$, we obtain

$$\sum_{x \in C} g(x \mid C - x) + z(C) \leq \sum_{K \subseteq [\ell]} \left(|K| (1 + 1/\ell) \alpha_{|K|} - (\ell - |K|) \alpha_{|K|+1} \right) \cdot f(\pi_K(C)).$$

By Fact 7(a), the coefficient vanishes for $1 \leq |K| < \ell$. For $K = \emptyset$, the coefficient is negative. For $K = [\ell]$, the coefficient is $(\ell + 1) \alpha_{\ell} = \kappa$, which completes the proof of (a).

(b) Fix $j \in [\ell]$ and $u \in \text{OPT}$. We have

$$g((u, j) \mid H - (u, j)) = \sum_{J \ni j} \alpha_{|J|} \cdot f(u \mid \pi_J(H - (u, j))) \geq \sum_{J \ni j} \alpha_{|J|} \cdot f(u \mid \pi_J(H)),$$

where the equality follows from Observation 8. For the inequality, if $(u, j) \notin H$, then $H - (u, j) = H$, and hence it holds with equality. If $(u, j) \in H$, then $u \in \pi_J(H)$ for every $J \ni j$, and hence the

right-hand side is 0, while the left-hand side is non-negative by the monotonicity of g (Lemma 6). We also have

$$\sum_{u \in \text{OPT}} f(u \mid \pi_J(H)) \geq f(\text{OPT} \cup \pi_J(H)) - f(\pi_J(H)) \geq f(\text{OPT}) - f(\pi_J(H)),$$

where the first inequality follows from Fact 3(b), and the second follows from the monotonicity of f . Combining the two inequalities above, for each $j \in [\ell]$,

$$\sum_{u \in \text{OPT}} g((u, j) \mid H - (u, j)) \geq \sum_{J \ni j} \alpha_{|J|} (f(\text{OPT}) - f(\pi_J(H))).$$

Summing this inequality over $j \in [\ell]$ and dividing by ℓ , we obtain

$$\begin{aligned} \frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{u \in \text{OPT}} g((u, j) \mid H - (u, j)) &\geq \frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{J \ni j} \alpha_{|J|} (f(\text{OPT}) - f(\pi_J(H))) \\ &= \sum_{J \subseteq [\ell]} \frac{|J|}{\ell} \alpha_{|J|} (f(\text{OPT}) - f(\pi_J(H))) \\ &= (\kappa - \ell) \cdot f(\text{OPT}) - z(H), \end{aligned}$$

where the first equality holds since each $J \subseteq [\ell]$ appears exactly $|J|$ times in the double sum, once for each $j \in J$, and the second equality follows from Fact 7(b) and the definition of z . $\square$

A.3 Omitted Proofs in Section 2.5: Estimating $g(x \mid X - x)$

Lemma 13. *There is a randomized algorithm $\text{ESTIMATE}(x, X)$ which, given $x \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$, uses $O(\varepsilon^{-3} \log(n/\varepsilon))$ value oracle queries and w.h.p. outputs $\hat{g}$ such that $\frac{g(x \mid X-x)}{1+\varepsilon} \leq \hat{g} \leq g(x \mid X-x)$. Since each call uses fresh randomness, this holds even if (x, X) is chosen depending on the outcomes of previous calls.*

To prove Lemma 13, we first show the following lemma.

Lemma 25 (Sampler). *Let $x = (u, i) \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$. Choose $k \in [\ell]$ uniformly at random, then choose J uniformly at random among the subsets of $[\ell]$ of size k that contain i , and let*

$$Y = \ell(1 + 1/\ell)^{k-1} \cdot f(u \mid \pi_J(X - x)).$$

Then, $\mathbb{E}[Y] = g(x \mid X - x)$ and $0 \leq Y \leq e\ell \cdot \mathbb{E}[Y]$.

Proof. Given $|J| = k$, the set J is uniformly distributed over the $\binom{\ell-1}{k-1}$ sets of size k containing i , and $\Pr[|J| = k] = 1/\ell$. Hence, each set $J \ni i$ is chosen with probability $1/(\ell \binom{\ell-1}{|J|-1})$. Since $\binom{\ell-1}{k-1} \alpha_k = (1 + 1/\ell)^{k-1}$ for every $k \in [\ell]$, we obtain

$$\mathbb{E}[Y] = \sum_{J \ni i} \frac{\ell(1 + 1/\ell)^{|J|-1}}{\ell \binom{\ell-1}{|J|-1}} \cdot f(u \mid \pi_J(X - x)) = \sum_{J \ni i} \alpha_{|J|} \cdot f(u \mid \pi_J(X - x)) = g(x \mid X - x),$$

where the last equality follows from Observation 8.

Let $h_0 = f(u \mid \pi_{\{i\}}(X - x))$. Every sampled J contains i , so $\pi_J(X - x) \supseteq \pi_{\{i\}}(X - x)$, and submodularity gives $Y \leq \ell(1 + 1/\ell)^{\ell-1} h_0 \leq e\ell h_0$. On the other hand, $J = \{i\}$ with probability $1/\ell$, and then $Y = \ell h_0$. Since $Y \geq 0$, this gives $\mathbb{E}[Y] \geq \frac{1}{\ell} \cdot \ell h_0 = h_0$, and hence $Y \leq e\ell \cdot \mathbb{E}[Y]$. $\square$

We use the following standard Chernoff bound.

Algorithm 3 ESTIMATE(x, X)

Input: $x \in \mathcal{N}'$ and $X \subseteq \mathcal{N}'$

1: draw $m = \Theta(\ell \varepsilon^{-2} \log(n/\varepsilon))$ independent samples $Y_1, \dots, Y_m$ of Y in Lemma 25

2: **return** $\frac{1}{1+\varepsilon/3} \cdot \frac{1}{m} \sum_{s=1}^m Y_s$

Fact 26 (Chernoff bound; see, e.g., [DP09, Theorem 1.1]). *Let $Y_1, \dots, Y_m$ be independent random variables taking values in $[0, R]$, and let $\mu = \mathbb{E}[\sum_{s=1}^m Y_s]$. Then, for $0 < \eta < 1$,*

$$\Pr\left[\left|\sum_{s=1}^m Y_s - \mu\right| \geq \eta\mu\right] \leq 2 \exp\left(-\frac{\eta^2 \mu}{3R}\right).$$

Proof of Lemma 13. We show that Algorithm 3 satisfies the conditions of Lemma 13. Let $a = g(x \mid X - x)$. If $a = 0$, then $Y = 0$ with probability 1 and the output is 0. Otherwise, by Lemma 25, each sample Y_s satisfies $0 \leq Y_s \leq e\ell a$ and $\mathbb{E}[Y_s] = a$. Hence, Fact 26 with $R = e\ell a$, $\mu = ma$, and $\eta = \varepsilon/3$ shows that the average $\bar{Y}$ of the m samples satisfies $|\bar{Y} - a| < \varepsilon a/3$ with probability at least $1 - 2 \exp(-\Omega(\varepsilon^2 m/\ell))$, which is w.h.p. by the choice of m . In this case, the output $\hat{g} = \bar{Y}/(1 + \varepsilon/3)$ satisfies $\hat{g} \leq a$ and $\hat{g} \geq \frac{1-\varepsilon/3}{1+\varepsilon/3}a \geq \frac{a}{1+\varepsilon}$. Each sample uses two value oracle queries, $f(\pi_J(X - x) + u)$ and $f(\pi_J(X - x))$. Since $\ell = O(1/\varepsilon)$, the number of value oracle queries is $2m = O(\varepsilon^{-3} \log(n/\varepsilon))$. $\square$