

Submodular Maximization over a Matroid k -Intersection: Multiplicative Improvement over Greedy

Moran Feldman¹ ✉ 🏠 

Department of Computer Science, University of Haifa, Israel

Justin Ward ✉ 🏠 

School of Mathematical Sciences, Queen Mary University of London, UK

Abstract

We study the problem of maximizing a non-negative monotone submodular objective f subject to the intersection of k arbitrary matroid constraints. The natural greedy algorithm guarantees $(k + 1)$ -approximation for this problem, and the state-of-the-art algorithm only improves this approximation ratio to k . We give a $\frac{2k \ln 2}{1 + \ln 2} + O(\sqrt{k}) < 0.819k + O(\sqrt{k})$ approximation algorithm for this problem. Our result is the first multiplicative improvement over the approximation ratio of the greedy algorithm for general k . We further show that our algorithm can be used to obtain roughly the same approximation ratio also for the more general problem in which the objective is not guaranteed to be monotone (the sublinear term in the approximation ratio becomes $O(k^{2/3})$ rather than $O(\sqrt{k})$ in this case).

All of our results hold also when the k -matroid intersection constraint is replaced with a more general matroid k -parity constraint. Furthermore, unlike the case in many of the previous works, our algorithms run in time that is independent of k and polynomial in the size of the ground set. Our algorithms are based on a hybrid greedy local search approach recently introduced by Singer and Thiery [38] for the weighted matroid k -intersection problem, which is a special case of the problem we consider. Leveraging their approach in the submodular setting requires several non-trivial insights and algorithmic modifications since the marginals of a submodular function f , which correspond to the weights in the weighted case, are not independent of the algorithm's internal randomness. In the special weighted case studied by [38], our algorithms reduce to a variant of the algorithm of [38] with an improved approximation ratio of $(k + 1) \ln 2 + O(\varepsilon) < 0.694k + 0.694 + O(\varepsilon)$, compared to an approximation ratio of $\frac{k+1}{2 \ln 2} \approx 0.722k + 0.722$ guaranteed by Singer and Thiery [38].

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Approximation algorithms analysis

Keywords and phrases Submodular function, matroid k -parity, matroid intersection, local search, greedy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.89

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://arxiv.org/pdf/2602.08473> [15]

Acknowledgements We would like to thank an anonymous reviewer for pointing out an error in a previous version of this paper.

1 Introduction

An important branch of combinatorial optimization studies the maximization of linear and submodular functions subject to natural families of combinatorial constraints. One large such family is *matroid k -intersection* – the family of constraints that can be represented as the

¹ This work was done while the author was visiting Queen Mary University of London.



© Moran Feldman and Justin Ward;

licensed under Creative Commons License CC-BY 4.0

53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).

Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;

Article No. 89; pp. 89:1–89:23



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



intersection of k matroid constraints. In 1978, Fisher, Nemhauser and Wolsey [16] showed that the natural greedy algorithm guarantees $(k + 1)$ -approximation for the problem of maximizing a monotone submodular function subject to a matroid k -intersection constraint, and for linear functions the above approximation ratio improves to k [25, 22].

Improving over the above guarantees of the greedy algorithm has been an open question for many years, and until recently success has mostly been limited to special cases such as k -dimensional matching constraints. Here, we are given a hypergraph $H = (V, E)$ such that the set V of vertices can be partitioned into k disjoint subsets (or parts) $V_1, V_2, \dots, V_k$, and each hyperedge $e \in E$ contains at most 1 vertex from each one of the subsets V_i . Such hypergraphs are known as k -partite hypergraphs. The k -dimensional matching constraint requires that we select a set of vertex disjoint hyperedges from E (i.e., a matching in H). This constraint can be captured by an intersection of k partition matroid constraints, each enforcing that no vertex of a particular set V_i is contained in two hyperedges.

Following a long line of works (including [1, 7, 3, 4, 34, 36, 40]), Neuwöhner [35] obtained an approximation ratio of $0.4986k + O(1)$ for weighted k -dimensional matching (i.e., the problem of maximizing a linear function subject to a k -dimensional matching constraint).² In the more general case of monotone submodular objectives, Ward [42] obtained a slightly weaker approximation ratio of $k/2 + O(1)$. In the less general case of a cardinality objective (i.e., when the goal is to maximize the number of elements in the solution), the best-known approximation is $k/3 + O(1)$ [39, 11, 17]. On the hardness side, Lee, Svensson and Thiery [27] have shown that even in the unweighted case, no polynomial time algorithm can obtain better than $k/12$ -approximation, improving on a previous lower bound of $\Omega(k/\log k)$ due to Hazan, Safra and Schwartz [19].

All the above algorithmic results apply even for the more general case of k -set packing constraints, which are similar to k -dimensional matching constraints, but allow the graph H to be a general k -uniform hypergraph rather than a k -partite hypergraph. Unlike k -dimensional matching constraints, k -set packing constraints are not captured by matroid k -intersection, but they have a different crucial property, namely, that multiple individual exchanges between two sets can be carried out simultaneously as a single valid exchange. This property holds for partition matroids and, more generally, for *strongly base orderable* matroids, but does not necessarily hold even for a single *general* matroid constraint.

For general matroid k -intersection constraints, Lee, Sviridenko and Vondrák [28] gave a local search algorithm with an approximation ratio of $k - 1$ for linear functions and k for monotone submodular functions. In the special case of a cardinality objective, the same authors [29] gave a different local search algorithm with an approximation ratio of $k/2 + \varepsilon$. We note that while the improvement in approximation for the cardinality objective has been by a multiplicative factor of 2, the improvements for submodular and linear functions have, until recently, been only by an additive term. This gap stems from the fact that the analysis presented in [29] is based on global arguments that are appropriate for the unweighted case in which all elements are equally valuable, but obtaining a result for a weighted problem requires more local arguments.

Recently, Singer and Thiery [38] have overcome this obstacle to give a $\frac{k}{2\ln 2} + O(1) \approx 0.722k + O(1)$ -approximation for linear objective functions by combining greedy and local search techniques. Like the result of Lee, Sviridenko and Vondrák [29] for unweighted matroid k -intersection, the algorithm of [38] works in fact even for the more general class of *matroid*

² We note that Thiery [41] derived better approximation results for small to moderate values of k , but for large k values he only obtains a weaker guarantee of $0.5k$.

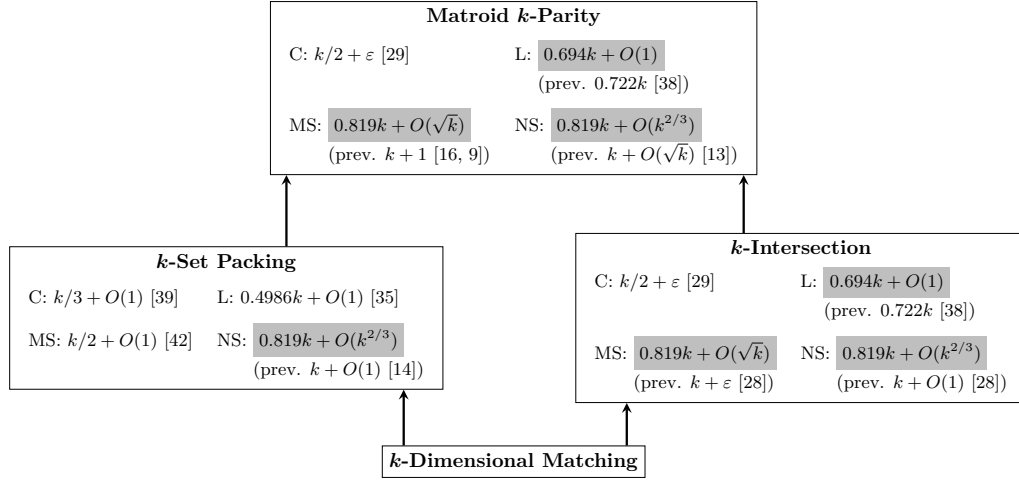


Figure 1 The four families of constraints discussed in Section 1. An arrow from family A of constraints to family B indicates that family B generalizes family A. Below each constraint family, we list the state-of-the-art approximation ratios for maximizing cardinality (C), linear (L), monotone submodular (MS) and/or non-monotone submodular (NS) objectives subject to constraints of this family. New results of this paper are marked with a grey background (and the results they improve over appear below them in parentheses). To avoid repetition, the approximation ratios for k -Dimensional Matching constraints are omitted as they are identical to the ratios for the more general k -Set Packing constraints.

k-parity constraints. This class may be viewed as a common generalization of matroid k -intersection and k -set packing constraints. See Figure 1 for a graphical depiction of the relationships between the different families of constraints we have discussed. The figure also states the state-of-the-art result for each combination of constraint family and class of objective functions.

1.1 Our Results

In this paper, we obtain the first result that multiplicatively improves over the approximation guarantee of the greedy algorithm for the problem of maximizing a monotone *submodular* objective function subject to a *general* matroid k -intersection constraint. Our technique is based on the one of Singer and Thiery [38], and like theirs, works also for the more general family of matroid k -parity constraints.

Let us briefly review the algorithm of Thiery and Singer. Given some $\delta > 0$,³ Thiery and Singer define a sequence of *markers* (or thresholds), of the form $m_i = W(1 - \delta)^{-i}$, where W is the maximum weight of any element and $i \in \mathbb{Z}_{\geq 0}$. These are used to partition the elements into classes according to their weight: an element e appears in the i -th weight class if $w(e) \in [m_i, m_{i-1})$. The algorithm processes weight classes in descending order of weight and maintains a solution A , which is initially empty. When processing a weight class C , the algorithm applies local search to find a feasible approximately maximum weight extension $X \subseteq C$ to the current solution A . The elements of X are then permanently added to A , and the algorithm continues to the next weight class. Intuitively, since all elements in C have similar weights, one mainly cares about maximizing the number of elements in X . Let

³ In our algorithm, we fix $\delta = 1/2$, but we leave it as a parameter here for consistency with [38].

X^* denote the largest possible extension X . Guarantees for the local search procedure for the unweighted matroid k -parity problem can be leveraged to show that $|X| \gtrsim (2/k) \cdot |X^*|$. However, the size of X^* might be smaller than the number of elements of the optimal solution within the class C because some elements of the optimal solution might be “blocked” by elements from heavier weight classes that are already in A . In order to bound the weight of the blocked elements, Thiery and Singer note that each selected element e can block at most k elements in total. Thus, if each blocked element has weight of at most $c \cdot w(e)$ for some $c < 1$, then one obtains $c \cdot k$ approximation for blocked elements. Unfortunately, c might be very nearly 1, in which case the guarantee degenerates to k . However, this can happen only when the blocking element e falls near the boundary of its weight class. To avoid this, Singer and Thiery multiply all markers by a single uniformly random shift τ . They then show that for each element, the probability some marker m_i lands near its weight is small, and thus the expected approximation for all blocked elements can indeed be bounded by $c \cdot k$ for some $c < 1$.

A major obstacle for adapting the algorithm of Singer and Thiery to submodular functions is that, under such functions, there is no longer a static notion of weight. Previous works [28, 42, 6, 8] have adapted local search algorithms for weighted maximization to submodular objectives by using the marginal or incremental contributions of an element e as a surrogate weight. In other words, if f is the objective function, then the adaptations implicitly define the weight of e to be $w(e) \triangleq f(A \cup \{e\}) - f(A)$, where A is the current solution.⁴ However, applying this idea to the algorithm of Singer and Thiery [38] breaks their solution for the problem of bounding the weight of the blocked elements. Recall that Singer and Thiery obtain this bound by applying a random shift, which guarantees that the probability that the weight of an element e is close to the boundary of its weight class is small. However surrogate weights $w(e)$ defined as above depend on the solution A constructed by the algorithm, and indirectly also on the random shift. This creates correlations between the weights and the shift, which make it difficult to bound the probability that the weight of an element is close to the boundary of its weight class.

Our solution for the above hurdle is to introduce an additional auxiliary weight $u(e)$ for every element e in the optimal solution O . Unlike the surrogate weights $w(e)$, the weights $u(e)$ depend only on O , and not on the solution set A , and thus are independent of the random shift of the algorithm. However, as these weights depend on the unknown set O , they are unavailable to the algorithm, which must continue to use the weights $w(e)$. This limitation of the algorithm is not important when $w(e)$ and $u(e)$ roughly agree, and thus, the output of the algorithm is good (in expectation) in this case. In order to obtain a result that holds in general, we further show that whenever the two weights disagree significantly, this discrepancy can be used to obtain an alternative bound on the value of the output of the algorithm.

Along the way, we observe that since the weight classes are defined using exponentially decreasing thresholds, it is natural to draw the random shift used by the algorithm to set these thresholds from an exponential distribution. Using this distribution for the shift instead of the uniform distribution employed by [38] leads to some improvement also for linear objective functions compared to the approximation ratio of roughly $0.722k + O(1)$ obtained by [38]. More formally, we prove the following theorem.

⁴ In some cases, A is in fact some appropriate subset of the current solution, but this distinction is unimportant for the sake of this high-level discussion.

► **Theorem 1.1.** *For every $\varepsilon > 0$, there exists an algorithm that runs in $\text{Poly}(|E|, \varepsilon^{-1})$ time and guarantees an approximation ratio of $\frac{2k \ln 2}{1 + \ln 2} + O(\sqrt{k}) \leq 0.819k + O(\sqrt{k})$ for maximizing a non-negative monotone submodular function $f: 2^E \rightarrow \mathbb{R}_{\geq 0}$ under a matroid k -parity constraint $(E, \mathcal{I})$. Moreover, if f is linear, the approximation ratio of the algorithm improves to $(k + 1) \ln 2 + O(\varepsilon) \leq 0.694k + 0.694 + O(\varepsilon)$.*

We note that the algorithm of [38] considers exchanges of size proportional to k in its local search, which leads to a time complexity that is polynomial only when k is constant. In contrast, the algorithm of Theorem 1.1 considers only exchanges of constant size, and therefore, runs in polynomial time regardless of the value of k .

We also consider the maximization of a non-monotone submodular function subject to a matroid k -parity constraint. Algorithms for maximizing monotone submodular functions usually still produce for a non-monotone submodular function f a guarantee of the form $f(A) \geq \alpha \cdot f(A \cup O)$, where A is the output set of the algorithm, O is the optimal solution and α is positive number. Gupta, Roth, Schoenebeck and Talwar [18] described a general framework for translating such a guarantee into a real approximation ratio, and their framework was improved by subsequent works [33, 13]. Unfortunately, because of the use of our novel auxiliary weights, when our algorithm from Theorem 1.1 is analyzed in the context of a non-monotone submodular function f , it yields a guarantee of the more involved form $f(A) \geq \alpha \cdot f(A \cup O) - \beta \cdot f(O)$ for some $\beta < \alpha$. Nevertheless, we are able to adapt the version of the framework used by [13] to transform guarantees of our more involved form into standard approximation ratios, which yields the following theorem.

► **Theorem 1.2.** *There exists an algorithm that runs in $\text{Poly}(|E|)$ time and guarantees an approximation ratio of $\frac{2k \ln 2}{1 + \ln 2} + O(k^{2/3}) \leq 0.819k + O(k^{2/3})$ for maximizing a non-negative (not necessarily monotone) submodular function $f: 2^E \rightarrow \mathbb{R}_{\geq 0}$ under a matroid k -parity constraint $(E, \mathcal{I})$.*

1.2 Additional Related Work

When $k = 2$, one can exactly optimize a linear function over a matroid k -intersection constraint via the classical algorithms of Edmonds [12] or Cunningham [10]. Similarly, a 2-set packing constraint is simply a standard matching constraint, and one can exactly optimize linear functions over it in polynomial time. Lawler [26] introduced the family of matroid parity constraints as a common generalization of both these families of constraints. Here, we are given a set E of disjoint pairs of elements from V and a matroid $\mathcal{M}$ on V , and the goal is to find a collection of pairs from E whose union is independent for $\mathcal{M}$. In contrast to 2-set packing and matroid 2-intersection, it is impossible to maximize even a cardinality objective under a matroid parity constraint in polynomial time when the matroid is given as an independence oracle [31, 23]. The problem is also known to be NP-hard (see, e.g., [37, §43.9]), but Lee, Sviridenko and Vondrák [29] showed that it admits a PTAS.

When $k \geq 3$, optimizing over both matroid k -intersection and k -set packing constraints is NP-hard as both these constraint families generalize k -dimensional matching constraints, and optimizing over the last family of constraints is NP-hard even for cardinality objectives [24]. All these types of constraints are generalized by the family of matroid k -parity constraints, and this family also generalizes the above-mentioned family of matroid parity constraints in the sense that matroid parity constraints are equivalent to matroid 2-parity constraints. The formal definition of matroid k -parity constraints can be found in Section 2, and we refer the reader to [29] for a more detailed discussion of the relationship between these and other families of constraints.

Finally, we note that improved results are known in several special cases. For matroid 2-intersection constraints, Huang and Sellier [20] give a $\frac{3}{2}$ -approximation in the special case that the objective is a vertex cover function. For matroid k -parity constraints, there are several results for the special case that the matroid is linear and a representation is given. In this setting, when $k = 2$ an exact polynomial time algorithm for cardinality objectives was given by Lovász [30], and Iwata and Kobayashi [21] later gave an exact polynomial time algorithm for maximizing linear objectives. For larger k values, in the same setting, exact FPT algorithms are known for both cardinality and linear objectives, with exponential dependence on k and the rank of the matroid or the number of blocks in the solution [2, 32]. In contrast, in this paper, we give approximation algorithms that require no assumptions on the structure of the underlying matroid and have a running time that is independent of k .

Paper Structure. Section 2 describes the definitions and notation that we use, as well as some known results. Our main algorithm is then presented in Section 3, and Section 4 shows that this algorithm has the approximation guarantees stated by Theorem 1.1 for linear and monotone submodular functions. Finally, Section 5 explains how to use the main algorithm to prove also our result for non-monotone submodular functions (Theorem 1.2).

2 Preliminaries

In this section, we present preliminaries consisting of definitions, notations and known results used in this paper. The preliminaries are loosely grouped based on whether they are more related to the objective function or to the constraint of the problem we consider.

2.1 Constraint Preliminaries

Matroids. A combinatorial constraint is given by a pair $(E, \mathcal{I})$, where E is a ground set of elements and $\mathcal{I} \subseteq 2^E$ is the collection of all subsets of E that are feasible according to the constraint. The constraint $\mathcal{M} = (E, \mathcal{I})$ is a *matroid* if $\mathcal{I}$ is non-empty and also obeys the following two properties.

- Down-closedness: for every two sets $S \subseteq T \subseteq E$, if $T \in \mathcal{I}$ then also $S \in \mathcal{I}$.
- Augmentation: for every two sets $S, T \in \mathcal{I}$, if $|S| < |T|$, then there exists an element $e \in T \setminus S$ such that $S \cup \{e\} \in \mathcal{I}$.

It is customary to refer to the sets in the collection $\mathcal{I}$ as the *independent sets* of $\mathcal{M}$. Notice that the augmentation property of matroids implies that all the inclusion-wise maximal independent sets of $\mathcal{M}$ have the same size. This size is known as the *rank* of $\mathcal{M}$. Like in the last definition, we often need in this paper to add or remove a single element from a set. For this purpose, given a set S and an element e , it is convenient to use $S + e$ and $S - e$ as shorthands denoting $S \cup \{e\}$ and $S \setminus \{e\}$, respectively.

Matroid k -Parity. A matroid k -parity constraint is given by a matroid $\mathcal{M} = (V, \mathcal{I}_{\mathcal{M}})$ together with a partition E of the elements of V into disjoint sets of size at most k . Borrowing terminology from hypergraphs, we refer to the elements of E as edges and the elements of V as vertices. The ground set of the matroid k -parity constraint is the set E of edges and a subset $S \subseteq E$ is feasible according to the constraint if the union of the edges in S is independent in $\mathcal{M}$. More formally, if we denote by $v(e)$ the set of up to k vertices of V appearing in e , then the matroid k -parity constraint is the combinatorial constraint $(E, \mathcal{I})$, where $\mathcal{I} \triangleq \{S \subseteq E \mid \bigcup_{e \in S} v(e) \in \mathcal{I}_{\mathcal{M}}\}$. It is often useful to extend the function v to sets of

edges in the natural way, namely, $v(S) \triangleq \cup_{e \in S} v(e)$ for every set $S \subseteq E$. Note that every matroid k -parity constraint is *down-closed*; i.e., if $A \in \mathcal{I}$ then $B \in \mathcal{I}$ for all $B \subseteq A$. This follows immediately from the fact that the matroid $\mathcal{M}$ is down-closed by definition.

In Section 1, it was mentioned that matroid k -parity constraints generalize the intersection of k matroid constraints. Let us now briefly explain why that is the case. Consider an arbitrary combinatorial constraint $(X, \mathcal{I})$ that can be obtained as the intersection of k matroids, i.e., there exist k matroids $(X, \mathcal{I}_1), (X, \mathcal{I}_2), \dots, (X, \mathcal{I}_k)$ such that $\mathcal{I} = \cap_{i=1}^k \mathcal{I}_i$. We can represent this as a matroid k -parity constraint as follows. Define an edge for each element $x \in X$, and let the k vertices of this edge be denoted by $(x, 1), (x, 2), \dots, (x, k)$. Intuitively, the element (x, i) represents the element x in the matroid $(X, \mathcal{I}_i)$. Based on this intuition, we construct a matroid $\mathcal{M}$ over the resulting set V of vertices by defining that a subset $S \subseteq V$ is independent in $\mathcal{M}$ if $\{e \in E \mid (e, i) \in S\} \in \mathcal{I}_i \quad \forall i \in [k]$. One can verify that $\mathcal{M}$ is a matroid, and so $(E, \mathcal{I})$ is a matroid k -parity constraint given by E and $\mathcal{M}$ that is equivalent to the original matroid intersection constraint.

We use in this paper the exchange property for matroid k -parity constraints given by Lemma 2.1. This property was implicitly proved by [38], and the full version of this paper [15] provides a different, and arguably simpler, proof of it. Lemma 2.1 proves that given two feasible sets A and B , one can construct for every element $b \in B$ a set N_b consisting of A elements that intuitively block the addition of b to A . Claims 2 and 3 of Lemma 2.1 show that for some sets $B' \subseteq B$, adding the elements of B' to A and removing the elements of $\cup_{b \in B'} N_b$ is an exchange that preserves feasibility. In general, designing sets N_b that allow for such exchanges is not very difficult. For example, one could set $N_b = A \setminus B$ for all $b \in B$, which would allow an exchange for every $B' \subseteq B$. Thus, to make Lemma 2.1 interesting (and useful), it is necessary to have constraints on the structure of the sets N_b guaranteeing that these sets are not “too rich”. Claims 1 and 4 of Lemma 2.1 play the role of these constraints.

► **Lemma 2.1.** *Consider a matroid k -parity constraint $(E, \mathcal{I})$, and let $A, B \in \mathcal{I}$ be two sets of edges that are feasible for this constraint. Then, there exists a collection of sets $\{N_b \subseteq A \mid b \in B\}$ such that*

1. *For every element $b \in B \cap A$, $N_b = \{b\}$; and for every element $b \in B \setminus A$, $N_b \subseteq A \setminus B$.*
2. *$A \cup \{b \in B : N_b = \emptyset\} \in \mathcal{I}$.*
3. *For every $a \in A$, $(A - a) \cup \{b \in B : N_b = \{a\}\} \in \mathcal{I}$.*
4. *For every $a \in A$, there are at most k distinct $b \in B$ such that $a \in N_b$.*

Independence/Feasibility Oracle. In most applications of combinatorial constraints, it is useful to have algorithms whose time complexities are polynomial in the size of the ground set E rather than the size of the collection $\mathcal{I}$ of feasible sets, which might be exponential in $|E|$. Therefore, algorithms that operate on such constraints usually assume that $\mathcal{I}$ is not part of the input, but can be accessed through an oracle, known as *independence* or *feasibility oracle*, that takes as input a set $S \subseteq E$ and answers whether $S \in \mathcal{I}$. The algorithms in this paper implicitly follow this standard assumption.

2.2 Objective Function Preliminaries

A set function $f: 2^E \rightarrow \mathbb{R}$ is a function that assigns a value to every subset of a given ground set E . Given a set S and an element e , we denote by $f(e \mid S) \triangleq f(S + e) - f(S)$ the marginal contribution of e to S . Similarly, given an additional set T , we denote by $f(T \mid S) \triangleq f(S \cup T) - f(S)$ the marginal contribution of T to S . The set function f is called *monotone* if $f(e \mid S) \geq 0$ for every set $S \subseteq E$ and element $e \in E$. The set function f is

called *submodular* if $f(e \mid S) \geq f(e \mid T)$ for every two sets $S \subseteq T$ and element $e \in E \setminus T$. In other words, f is submodular if the marginal contribution of an element e to a set S can only diminish when additional elements are added to the set S . An alternative, equivalent, characterization of submodularity states that f is submodular if and only if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for every two sets $S, T \subseteq E$.

In the case that $f(S) + f(T) = f(S \cup T) + f(S \cap T)$ for all $S, T \subseteq E$, we say that f is *modular*. Equivalently, a modular set function can be defined as a function of the form $f(S) = w_0 + \sum_{e \in S} w_e$, where w_0 and $\{w_e \mid e \in E\}$ are arbitrary real values. In the special case that $w_0 = 0$ (and thus, $f(\emptyset) = 0$), we say that f is *linear*.

Our algorithms optimize a submodular set function f . However, as in the case of combinatorial constraints, it often does not make sense to assume that the algorithm has an explicit description of f because this description can be exponential in the size of the ground set E . Instead, our algorithms implicitly make the standard assumption that f is accessed through a *value oracle* that given a set $S \subseteq E$ returns $f(S)$.

3 Main Algorithm

In this section, we present our main algorithm, given below as Algorithm 1. In Section 4, we prove that Algorithm 1 provides the approximation guarantees for linear and monotone submodular functions given in Theorem 1.1. Algorithm 1 is also a central component in the proof of Theorem 1.2 for non-monotone submodular objectives. As stated, Algorithm 1 is not necessarily a polynomial time algorithm as it might have a large number of iterations. In the full version of this paper [15], we describe a polynomial time implementation of this algorithm that skips over unnecessary iterations.

Algorithm 1 gets as input a ground set E , a submodular function $f: 2^E \rightarrow \mathbb{R}$ obeying $f(\emptyset) \geq 0$, the collection $\mathcal{I}$ of the feasible sets of some matroid k -parity constraint over the ground set E and an error parameter $\varepsilon \in (0, 1)$. It then iteratively constructs sets $A_1, A_2, \dots$ whose union becomes the algorithm's output. For ease of notation, Algorithm 1 uses $A_{\leq i}$ to denote the union $A_1 \cup A_2 \cup \dots \cup A_i$. When thinking in terms of the intuition given in Section 1, A_1 is the part of the solution of the algorithm chosen from the highest “weight class”, A_2 is the part of the solution of the algorithm chosen from the second highest “weight class”, and so on. Accordingly, Algorithm 1 adds to the set A_i only elements with large enough marginal values. More specifically, Algorithm 1 defines a series of exponentially decreasing thresholds m_i , and adds to A_i only elements whose marginal contribution is at least m_i .

To understand the details of the construction of the sets A_i by Algorithm 1, we need the following definition.

► **Definition 3.1** ((θ, ε) -improvement). *Let $\theta > 0$ and $\varepsilon \in (0, 1)$ be two given values, and $A \in \mathcal{I}$ be a feasible solution. We say that a pair of sets $S \subseteq E \setminus A$ and $N \subseteq A$ is a (θ, ε) -improvement for A if $(A \cup S) \setminus N \in \mathcal{I}$ and one of the following holds:*

1. $S = \{x\}, N = \emptyset$, and $f(x \mid A) \geq \theta$.
2. $S = \{x\}, N = \{y\}$, $f(x \mid A) \geq \theta$, and $f(A + x - y) \geq f(A) + \varepsilon\theta$.
3. $|S| = 2, N = \{y\}$ and $f(x_1 \mid A) \geq \theta$, $f(x_2 \mid A + x_1) \geq \theta$ for at least one of the two possible ways to assign the labels x_1, x_2 to the elements of S .

At the beginning of iteration i of Algorithm 1, the sets $A_1, A_2, \dots, A_{i-1}$ are already finalized. Algorithm 1 then tries to construct a set A_i that is a good addition to these sets. It does so by initializing A_i to be the empty set, and then improving $A_{\leq i-1} \cup A_i$ via a series of (m_i, ε) -improvements. Notice that, by definition, the elements added by these improvements

to A_i have marginals of at least m_i , as promised above. Since the sets $A_1, A_2, \dots, A_{i-1}$ are finalized before the i -th iteration of Algorithm 1, and must not be modified at this iteration, the algorithm considers only (m_i, ε) -improvements whose set N of removed elements is a subset of the current set A_i . Whenever an improvement of this kind is found, it is applied by setting $A_i \leftarrow (A_i \cup S) \setminus N$, and once no more such (m_i, ε) -improvements can be found, the i -th iteration of Algorithm 1 terminates.

■ **Algorithm 1** GREEDY/LOCAL-SEARCH HYBRID ALGORITHM $(E, f, \mathcal{I}, \varepsilon)$.

```

1 Let  $W \leftarrow \max_{e \in E} f(e \mid \emptyset)$ .
2 Let  $\alpha$  be a uniformly at random value from  $(0, 1]$ , and let  $\tau \leftarrow 2^\alpha$ .
3 Define  $m_i \triangleq W\tau 2^{-i}$  for every integer  $i \geq 0$ .
4 Let  $i \leftarrow 0$  and  $A_{\leq 0} \leftarrow \emptyset$ .
5 while there exists  $e \in E \setminus A_{\leq i}$  such that  $f(e \mid A_{\leq i}) > 0$  and  $A_{\leq i} + e \in \mathcal{I}$  do
6   | Update  $i \leftarrow i + 1$ .
7   | Let  $A_i \leftarrow \emptyset$ .
8   | while there is some  $(m_i, \varepsilon)$ -improvement  $(S, N)$  for  $A_{\leq i-1} \cup A_i$  with  $N \subseteq A_i$  do
9   |   |  $A_i \leftarrow (A_i \setminus N) \cup S$ .
10  | Let  $A_{\leq i} \leftarrow A_{\leq i-1} \cup A_i$ .
11 return  $A_{\leq i}$ .
```

As mentioned above, in the full version of this paper [15], we show that Algorithm 1 can in fact be simulated in polynomial time. In order to derive the approximation guarantees of Theorem 1.1, it is sufficient to prove here the weaker claim that Algorithm 1 terminates. In each iteration i , Algorithm 1 repeatedly searches for and applies (m_i, ε) -improvements for some current solution A . Each such improvement either increases $|A|$, or leaves $|A|$ unchanged and increases $f(A)$. Since there are only finitely many possible distinct values for $|A|$ and $f(A)$, it follows that each iteration of Algorithm 1 must terminate. The following observation implies that Algorithm 1 can perform at most finitely many iterations before terminating.

► **Observation 3.2.** *For any input to Algorithm 1, there is an integer $L \geq 0$ such that $f(e \mid A_{\leq L}) \leq 0$ for every $e \in E \setminus A_{\leq L}$ with $A_{\leq L} + e \in \mathcal{I}$, and furthermore, if $L \neq 0$, then $A_L \neq \emptyset$.*

Proof. First, we note that any iteration i in which at least one (m_i, ε) -improvement is found, must end with $|A_i| > 0$. This follows directly from the structure of (θ, ε) -improvements, which never decrease the size of the set A that is being improved. Since the sets A_i produced across different iterations are disjoint by construction, there can be at most $|E|$ iterations in which the algorithm finds at least one (θ, ε) -improvement. Let L be the index of the last such iteration, or 0 if there are no such iterations. Suppose, for the sake of contradiction, that there is some $e \in E \setminus A_{\leq L}$ with $A_{\leq L} + e \in \mathcal{I}$ and $f(e \mid A_{\leq L}) > 0$. Since no (θ, ε) -improvements are found after iteration L , we must have $A_{\leq i} = A_{\leq L}$ for all $i > L$. But, for sufficiently large $i \geq 1 + \log_2 W - \log_2 f(e \mid A_{\leq L})$, we then have $e \in E \setminus A_{\leq i}$ with $f(e \mid A_{\leq i}) = f(e \mid A_{\leq L}) \geq m_i$ and $A_{\leq i} + e = A_{\leq L} + e \in \mathcal{I}$. Thus, $(\{e\}, \emptyset)$ would be a valid (m_i, ε) -improvement in iteration $i > L$, contradicting the fact that no such improvements are found by Algorithm 1. ◀

Observation 3.2 implies that Algorithm 1 terminates after L iterations. Therefore, in the rest of our analysis we use $A_{\leq L}$ to denote the output set of Algorithm 1. Note that the condition on Line 8 of the algorithm implies that the set $A_{\leq i} = A_{\leq i-1} \cup A_i$ finalized at the

end of each iteration i is locally optimal with respect to (m_i, ε) -improvements with $N \subseteq A_i$. That is, for every $i \in [L]$, there is no (m_i, ε) improvement (S, N) for $A_{\leq i}$ with such N . In particular, considering (m_i, ε) -improvements of the first kind gives the following observation.

► **Observation 3.3.** *For every $i \in [L]$, the inequality $f(e \mid A_{\leq i}) < m_i$ holds for every $e \in E \setminus A_{\leq i}$ that obeys $A_{\leq i} + e \in \mathcal{I}$ at the end of iteration i .*

Proof. Suppose $A_{\leq i} + e \in \mathcal{I}$ for some $e \in E \setminus A_{\leq i}$ at the end of iteration i . If $f(e \mid A_{\leq i}) \geq m_i$, then $(\{e\}, \emptyset)$ would be a valid (m_i, ε) -improvement, contradicting the fact that the loop on Line 8 of Algorithm 1 terminated. ◀

Before concluding this section, let us intuitively explain two aspects of Algorithm 1. The first aspect is the choice of powers of 2 to define the thresholds m_i . This choice guarantees that it is always beneficial (or at least non-detrimental) to replace a single element of A_i with two, which is a property that naturally holds in the unweighted version of the problem, and is necessarily for borrowing arguments from this version. The second aspect is that unlike many local-search-based algorithms for submodular maximization, Algorithm 1 considers a very restricted set of possible improvements, namely, improvements that only involve the removal of up to a single element and the addition of up to two elements. The restriction to add at most two elements is a natural consequence of the restriction to remove at most a single element because, as discussed above, if removing a single element enables the addition of two or more elements, then this is already beneficial (or at least non-detrimental) to add two of them. Thus, it only remains to understand why Algorithm 1 only considers improvements removing at most a single element. The reason for that is that the exchange property that we use (Lemma 2.1) only guarantees the existence of such improvements, and therefore, taking advantage in the analysis of improvements removing multiple elements would require proving a stronger exchange property.

4 Analysis of the Main Algorithm

In this section, we bound the approximation guarantee of Algorithm 1 when $f: 2^E \rightarrow \mathbb{R}$ is a general submodular function with $f(\emptyset) \geq 0$. We then use this bound to prove Theorem 1.1. Fix a matroid k -parity constraint $(E, \mathcal{I})$. In the following, we consider an execution of Algorithm 1 on this matroid k -parity constraint and the above function f . Recall that $A_{\leq L}$ is the output of this algorithm, and thus our objective is to lower bound $f(A_{\leq L})$ in terms of $f(O)$, where O is an optimal solution for our given instance (i.e., a set in $\mathcal{I}$ maximizing f). Without loss of generality, we may restrict our analysis to optimal solutions that satisfy the following property.

► **Definition 4.1.** *A feasible set O is strictly down-monotone (with respect to f) if $f(x \mid O - x) > 0$ for every $x \in O$.*

To see that such a restriction is without loss of generality, note that as long as $f(x \mid O - x) \leq 0$ for some element $x \in O$, removing this element from O gives a new feasible solution that is smaller, but has at least the same value as O and thus is also optimal. Hence, any minimal size optimal solution is strictly down-monotone.

Our analysis below does not use any property of O beside its strict down-monotonicity. In particular, it does not use the optimality of O , and therefore, applies also when O is a strictly down-monotone set that is not necessarily optimal. This property of the analysis is used in Section 5. The analysis consists of a few steps. First, in Section 4.1, we partition the elements of O into sets $\{O_i \mid i \in [L]\}$. Intuitively, each O_i is a subset of O whose contribution

is bounded in terms of the contribution of the elements of A_i , i.e., the elements selected in iteration i of Algorithm 1. We quantify these contributions in Section 4.2 by defining weights w and $\bar{w}$ for the elements of $A_{\leq L}$ and O , respectively. Section 4.3 then classifies the elements of O into three different types of elements, and Section 4.5 separately bounds the total weight of each type in terms of the weight of $A_{\leq L}$ via appropriate charging arguments.

In the linear case, the weights w and $\bar{w}$ of elements are identical and independent of the sets Algorithm 1 constructs. This makes our charging scheme essentially the same as the one used in [38]. However, in the submodular case, the weights depend on the final set $A_{\leq L}$ returned by the algorithm. In order to analyze the effect of the random shift τ selected by Algorithm 1, we must thus introduce, in Section 4.4, a set of auxiliary weights u quantifying the value of O in a fashion that is independent of $A_{\leq L}$. Our analysis then explicitly considers how these weights u differ from $\bar{w}$. Specifically, we show that if there is a large discrepancy between u and $\bar{w}$, then we can handle the resulting loss in the analysis by balancing with an alternative lower bound on $f(A_{\leq L})$ that gains from such a discrepancy.

4.1 Partitioning the Elements of O

This section describes a partition of O into disjoint sets $O_1, O_2, \dots, O_L$. Analogously to the sets $A_{\leq i}$, for every integer $0 \leq i \leq L$, we use $O_{\leq i}$ to denote the union $\cup_{j=1}^i O_j$. Intuitively, each set O_i represents elements from O whose values can be bounded in terms of the elements selected as A_i in iteration i of Algorithm 1 and thus need not be considered when later iterations of this algorithm are analyzed. In line with this intuition, we prove below that the elements of $O \setminus O_{\leq i}$, whose values need to be bounded in terms of elements selected by later iterations, can all be feasibly added to the solution $A_{\leq i}$ that Algorithm 1 has after iteration i .

We can now formally define the partition of O . Recall that $O_{\leq 0} = \emptyset$ by definition. For every $i \in [L]$, the set O_i is given by the recursive formula $O_i \triangleq \{o \in O \setminus O_{\leq i-1} \mid N_o^{(i)} \neq \emptyset\}$, where $\{N_o^{(i)} \subseteq A_{\leq i} \mid o \in A_{\leq i-1} \cup (O \setminus O_{\leq i-1})\}$ is the collection of sets obtained by applying Lemma 2.1 with the sets $A_{\leq i}$ and $A_{\leq i-1} \cup (O \setminus O_{\leq i-1})$ as the sets A and B of the lemma, respectively. Recall that applying Lemma 2.1 with these sets requires them to be feasible. The feasibility of $A_{\leq i}$ is clear. The feasibility of $A_{\leq i-1} \cup (O \setminus O_{\leq i-1})$ follows from the next observation, which can be proven by induction on i using claims 1 and 2 of Lemma 2.1.

► **Observation 4.2.** *For every integer $0 \leq i \leq L$, $A_{\leq i} \cup (O \setminus O_{\leq i})$ is feasible and $A_{\leq i} \cap (O \setminus O_{\leq i}) = \emptyset$.*

4.2 Defining Weights for the Charging Argument

We now define the weights we use to quantify the values of elements in O and $A_{\leq L}$. An element's weight is based on the *incremental value* it adds to a given set, and in the linear case it corresponds exactly to the weight of this element in the definition the objective function f (as long as this weight is non-negative). In order to define the incremental values, we order the elements of $A_{\leq L}$ as $a_1, a_2, \dots, a_{|A_{\leq L}|}$, in the order in which they were last added by Algorithm 1 to any of the sets $A_1, A_2, \dots, A_L$. Applying (θ, ε) -improvements of the third kind involves adding 2 elements to A_i at the same time, which creates an ambiguity. To resolve the ambiguity, recall that in such improvements, Definition 3.1 specifies that there must be a way to assign the two added elements the labels x_1 and x_2 so that $f(x_1 \mid A) \geq \theta$ and $f(x_2 \mid A + x_1) \geq \theta$, where A is the set improved by the (θ, ε) -improvement. In our

ordering of $A_{\leq L}$, we suppose that the element corresponding to x_1 is added to A_i immediately before the element corresponding to x_2 .⁵ Given our ordering $a_1, \dots, a_{|A_{\leq L}|}$, we define the weight of each element $a_j \in A_{\leq L}$ as $w(a_j) \triangleq f(a_j \mid \{a_1, a_2, \dots, a_{j-1}\})$.

We proceed similarly for elements in O . Let us denote the elements of O by $o_1, o_2, \dots, o_{|O|}$ in an arbitrary order.⁶ Then, for any $o_j \in O$, we define

$$\bar{w}(o_j) \triangleq \max\{0, f(o_j \mid (A_{\leq L} - o_j) \cup \{o_1, o_2, \dots, o_{j-1}\})\} .$$

Note that we ensure that the $\bar{w}$ weights are always non-negative. Let us briefly provide some intuition for the second expression in the maximum. For any $o_j \in O \setminus A_{\leq L}$, this term is equal to $f(o_j \mid A_{\leq L} \cup \{o_1, \dots, o_{j-1}\})$, and thus the weights $\bar{w}$ capture the incremental values obtained when adding all such elements sequentially to the set $A_{\leq L}$. For any element $o_j \in O \cap A_{\leq L}$, we could define $\bar{w}(o_j) = 0$, since it provides no such incremental value. However, it is useful in the linear case to ensure that all the weights we are defining are equal to the natural notion of an element's weight (i.e., $f(\{x\})$) whenever this weight is non-negative. The chosen expression ensures that this is the case even for $x \in O \cap A_{\leq L}$.

It is convenient to extend the above-defined weights to sets. We do this in the natural way, i.e., for any set $T \subseteq A_{\leq L}$, we define $w(T) \triangleq \sum_{e \in T} w(e)$, and for any set $T \subseteq O$, we define $\bar{w}(T) \triangleq \sum_{e \in T} \bar{w}(e)$. The following lemma proves some basic properties of the weights w and $\bar{w}$. The first three claims of the lemma show that an element's weight must fall into a particular range defined by the values m_i , depending on which (if any) set A_i or O_i this element appears in. The final claim relates the two weights w and $\bar{w}$ given to elements that appear in both $A_{\leq L}$ and O .

► **Lemma 4.3.** *The weights w and $\bar{w}$ satisfy the following.*

1. For every $i \in [L]$ and element $a \in A_i$, $0 < m_i \leq w(a) \leq m_{i-1}$.
2. For every $i \in [L]$ and element $o \in O_i$, $\bar{w}(o) \leq m_{i-1}$.
3. For all $o \in O \setminus O_{\leq L}$, $\bar{w}(o) = 0$.
4. For every element $o \in O \cap A_{\leq L}$, $\bar{w}(o) \leq w(o)$.

Proof. We prove each claim in turn.

Claim 1. Consider the moment that $a \in A_i$ was last added to A_i by Algorithm 1, and let $\bar{A}_i$ denote the value of this set just before the element a was added by some (m_i, ε) -improvement (if the (m_i, ε) -improvement adds two elements, and a is the one labeled x_2 in this improvement, then $\bar{A}_i$ is assumed to already contain the element labeled as x_1 by the improvement). Since a is added by an (m_i, ε) -improvement for the set $\bar{A}_i \cup A_{\leq i-1}$,

$$m_i \leq f(a \mid \bar{A}_i \cup A_{\leq i-1}) \leq w(a) ,$$

where the second inequality follows from the submodularity of f because all the elements of $A_{\leq L}$ that were last added to the sets $A_1, A_2, \dots, A_L$ before the last time in which a was added to A_i must also belong to $\bar{A}_i \cup A_{\leq i-1}$.

For the upper bound of the claim, first suppose that $a \in A_1$. Then, by submodularity and the definition of W , $w(a) \leq f(a \mid \emptyset) \leq W \leq W\tau = m_0$. In the general case, suppose $a \in A_i$ for some $i > 1$. Then, $a \in E \setminus A_{\leq i-1}$ with $\bar{A}_i \cup A_{\leq i-1} + a \in \mathcal{I}$. Since $\mathcal{I}$ is down-closed,

⁵ If both ways to label the added elements obey the above inequalities, then we arbitrary select one of them.

⁶ While the order $o_1, o_2, \dots, o_{|O|}$ is arbitrary, it is essential that this order is not chosen in a way that depends on the random bits of Algorithm 1.

$A_{\leq i-1} + a \in \mathcal{I}$. Observation 3.3 then implies that $m_{i-1} > f(a \mid A_{\leq i-1}) \geq w(a)$, where the final inequality holds by submodularity since all the elements of $A_{\leq i-1}$ were added to the algorithm's solution before a was added to A_i .

Claim 2. We have $o \in O_i \subseteq O \setminus O_{\leq i-1}$. By Observation 4.2, $(O \setminus O_{\leq i-1}) \cap A_{\leq i-1} = \emptyset$ and $(O \setminus O_{\leq i-1}) \cup A_{\leq i-1} \in \mathcal{I}$; and thus $o \in E \setminus A_{\leq i-1}$ with $A_{\leq i-1} + o \in \mathcal{I}$. If $i > 1$, then by Observation 3.3, we must have $f(o \mid A_{\leq i-1}) < m_{i-1}$. If $i = 1$, we get the slightly weaker inequality $f(o \mid A_{\leq i-1}) \leq f(o \mid \emptyset) \leq W \leq m_0 = m_{i-1}$ by the submodularity of f . In either case, since m_{i-1} is positive, $\max\{0, f(o \mid A_{\leq i-1})\} \leq m_{i-1}$. Suppose that o is the j -th element o_j in our ordering of O . Then, by submodularity,

$$\begin{aligned} \bar{w}(o) = \bar{w}(o_j) &= \max\{0, f(o_j \mid (A_{\leq L} - o_j) \cup \{o_1, o_2, \dots, o_{j-1}\})\} \\ &\leq \max\{0, f(o_j \mid A_{\leq i-1})\} \leq m_{i-1} . \end{aligned}$$

Claim 3. By Observation 4.2, $O \setminus O_{\leq L}$ and $A_{\leq L}$ are disjoint and $(O \setminus O_{\leq L}) \cup A_{\leq L} \in \mathcal{I}$. Thus, for every $o_j \in O \setminus O_{\leq L}$ we have $o_j \in E \setminus A_{\leq L}$ and $A_{\leq L} + o_j \in \mathcal{I}$. Observation 3.2 then implies that $f(o_j \mid A_{\leq L}) \leq 0$. Thus, by the submodularity of f ,

$$\bar{w}(o_j) = \max\{0, f(o_j \mid (A_{\leq L} - o_j) \cup \{o_1, o_2, \dots, o_{j-1}\})\} \leq \max\{0, f(o_j \mid A_{\leq L})\} = 0 .$$

Claim 4. Suppose $o = o_j \in A_{\leq L} \cap O$. Then, we must have $o = a_{j'}$ for some $a_{j'} \in A_{\leq L}$. The first claim of the present lemma yields $w(a_{j'}) > 0$ since $a_{j'}$ must appear in some set A_i . Thus, by submodularity,

$$\begin{aligned} w(o) = w(a_{j'}) &= \max\{0, w(a_{j'})\} = \max\{0, f(a_{j'} \mid \{a_1, \dots, a_{j'-1}\})\} \\ &\geq \max\{0, f(a_{j'} \mid (A_{\leq L} - a_{j'}) \cup \{o_1, \dots, o_{j-1}\})\} \\ &= \max\{0, f(o_j \mid (A_{\leq L} - o_j) \cup \{o_1, \dots, o_{j-1}\})\} = \bar{w}(o) . \quad \blacktriangleleft \end{aligned}$$

4.3 Classifying the Elements of O for Charging

In our charging scheme, we consider three different types of elements of O . The first is the elements that belong to some O_i , have weights approximately equal to the weights of the elements in A_i (up to a factor of $m_{i-1}/m_i = 2$) and can charge to only a single element of A_i . We denote below by $O^{(s)}$ the set of these elements. Intuitively, our analysis uses an argument borrowed from the $(k/2 + \varepsilon)$ -approximation algorithm for the unweighted matroid k -parity problem [29] to show that the elements of $O^{(s)}$ only contribute an additive constant to the approximation guarantee. The next type is the elements of $O \setminus O_{\leq L}$, which are “left over” at the end of Algorithm 1. The contributions of these elements are small, and can be essentially ignored. The final type is all remaining elements, i.e., $O_{\leq L} \setminus O^{(s)}$.

To formally define the set $O^{(s)}$ of the first type of elements of O , we define a set N_o for every element $o \in O_{\leq L}$ as follows. For any $o \in O_{\leq L}$, let i be the single integer in $[L]$ for which $o \in O_i$, and define $N_o \triangleq N_o^{(i)}$ (recall that $N_o^{(i)}$ is defined in Section 4.1). Intuitively, one can think of N_o as the set of elements of $A_{\leq L}$ that are responsible for the inclusion of o in O_i . We then define for every $i \in [L]$ a subset $O_i^{(s)} \subseteq O_i$ given by $O_i^{(s)} \triangleq \{o \in O_i \mid \bar{w}(o) > m_i \text{ and } |N_o| = 1\}$. Then, we can define $O^{(s)} \triangleq \bigcup_{i=1}^L O_i^{(s)}$. Notice that this definition is consistent with the above intuition in the following senses. First, each element $o \in O^{(s)}$ has indeed only a single element in N_o to “blame” for its inclusion in O_i and thus only a single element to charge to. Second, the weight of o is comparable to the weight of this single element because $N_o \subseteq A_i$ (see Observation 4.4 below), which implies by claim 1 of Lemma 4.3 that the weight of every element in N_o is in the range $[m_i, m_{i-1}]$.

► **Observation 4.4.** For every $i \in [L]$, $N_o = N_o^{(i)} \subseteq A_i$ for every $o \in O_i$.

Proof. Claim 1 of Lemma 2.1 guarantees that $N_o^{(i)}$ can intersect $A_{\leq i-1}$ only on the element o itself. However, $o \in O_i$ and thus does not belong to $A_{\leq i-1}$ by Observation 4.2. ◀

The following lemma states some properties of the sets defined above. The first two claims of the lemma show that every element $o \in O^{(s)}$ “blames” a different element of $A_{\leq L}$ for its inclusion in $O_{\leq L}$, and furthermore, they show that this element is almost as valuable as o . Claim 3 of the lemma shows that every element of $A_{\leq L}$ is “blamed” by at most k elements of $O_{\leq L}$.

► **Lemma 4.5.** The above defined sets obey the following.

1. For every $o \in O^{(s)}$, $\bar{w}(o) \leq (1 + \varepsilon) \cdot w(N_o)$.
2. For every two elements $o, o' \in O^{(s)}$, $N_o \cap N_{o'} = \emptyset$.
3. Every $a \in A_{\leq L}$ appears in N_o for at most k distinct $o \in O_{\leq L}$.

Proof. First, we show that all $o \in O_i^{(s)} \setminus A_{\leq i}$ must have relatively large marginal contribution with respect to the set $A_{\leq i}$. By definition, any $o_j \in O_i^{(s)}$ has $0 < m_i < \bar{w}(o_j) = f(o_j \mid (A_{\leq L} - o_j) \cup \{o_1, \dots, o_{j-1}\})$. If $o_j \notin A_{\leq i}$, then $(A_{\leq L} - o_j) \cup \{o_1, \dots, o_{j-1}\} \supseteq A_{\leq i} \cup O'$ for any $O' \subseteq \{o_1, \dots, o_{j-1}\}$ and thus, by submodularity,

$$m_i < \bar{w}(o_j) \leq f(o_j \mid A_{\leq i}) \quad \text{and} \quad m_i < \bar{w}(o_j) \leq f(o_j \mid A_{\leq i} + o_{j'}), \quad (1)$$

for any $j' < j$. Intuitively, these inequalities show that any such o_j has marginal contribution large enough to be part of an (m_i, ε) -exchange at the end of the iteration i . Using these observations, we now prove each claim of the lemma in turn.

Claim 1. Consider any $o \in O^{(s)}$ and let $i \in [L]$ be the (unique) value such that $o \in O_i^{(s)}$. Then, since $o \in O_i \subseteq O \setminus O_{\leq i-1}$, we must have $o \notin A_{\leq i-1}$ by Observation 4.2. If $o \in A_i \cap O_i^{(s)} \subseteq A_i \cap O_i$, then by claim 1 of Lemma 2.1, we are guaranteed that $N_o = \{o\}$, which implies $w(o) = w(N_o)$, and claim 4 of Lemma 4.3 further gives $\bar{w}(o) \leq w(o)$. Thus, $\bar{w}(o) \leq (1 + \varepsilon) \cdot \bar{w}(o) \leq (1 + \varepsilon) \cdot w(o) = (1 + \varepsilon) \cdot w(N_o)$, as required.

Now, suppose that $o \in O_i^{(s)} \setminus A_i$. Then, $o \notin A_{\leq i}$, and $N_o = N_o^{(i)} = \{a\}$ for some element $a \in A_i$. By the properties of $N_o^{(i)}$ (specifically, Lemma 2.1, claim 3), we have $A_{\leq i} - a + o \in \mathcal{I}$. As shown above in (1), $f(o \mid A_{\leq i}) > m_i$. Thus, $(\{o\}, \{a\})$ satisfies all the properties required of an (m_i, ε) improvement of the second type, except for the inequality $f(A_{\leq i} - a + o) - f(A_{\leq i}) \geq \varepsilon m_i$. Since Algorithm 1 terminated iteration i , and finalized the set $A_{\leq i}$, without applying this improvement, it must have been the case that this inequality was not satisfied, i.e.,

$$\begin{aligned} \varepsilon m_i &> f(A_{\leq i} - a + o) - f(A_{\leq i}) = f(o \mid A_{\leq i} - a) - f(a \mid A_{\leq i} - a) \\ &\geq f(o \mid A_{\leq i}) - f(a \mid A_{\leq i} - a) \geq \bar{w}(o) - f(a \mid A_{\leq i} - a) \geq \bar{w}(o) - w(a), \end{aligned}$$

where the second inequality follows from submodularity, the third inequality holds by the first inequality of (1), and the last inequality follows from submodularity and the definition of w . Rearranging, we have $\bar{w}(o) \leq w(a) + \varepsilon m_i \leq (1 + \varepsilon) \cdot w(a) = (1 + \varepsilon) \cdot w(N_o)$, where the second inequality follows from claim 1 of Lemma 4.3 since $a \in A_i$.

Claim 2. Consider any distinct $o_j, o_{j'} \in O^{(s)}$, where $j' < j$. We show below that if $N_{o_j} \cap N_{o_{j'}} \neq \emptyset$, then an (m_i, ε) -improvement of the third form exists for some $A_{\leq i}$, contradicting the fact that iteration i of Algorithm 1 terminated and finalized the set $A_{\leq i}$ without applying this improvement.

To this end, suppose that $N_{o_{j'}} \cap N_{o_j} \neq \emptyset$. Since both $o_{j'}$ and o_j are in $O^{(s)}$, we must have $N_{o_{j'}} = N_{o_j} = \{a\}$ for some single element a , and this element a must belong to exactly one set A_i (for $i \in [L]$) since the sets A_i are disjoint by construction. Moreover, since $N_o^{(i')} \subseteq A_{i'}$ for all $i' \in [L]$, we further must have $N_{o_j} = N_{o_{j'}}^{(i)} = \{a\}$, $N_{o_{j'}} = N_{o_j}^{(i)} = \{a\}$, and $o_j, o_{j'} \in O_i^{(s)}$ for the same $i \in [L]$. Recall that the sets $\{N_o^{(i)} \mid o \in A_{\leq i-1} \cup (O \setminus O_{\leq i-1})\}$ were constructed to satisfy the properties of Lemma 2.1 with respect to $A_{\leq i}$ and $A_{\leq i-1} \cup (O \setminus O_{\leq i-1})$. Since $N_{o_{j'}}^{(i)} = N_{o_j}^{(i)} = \{a\}$, the third such property implies that $(A_{\leq i} - a) \cup \{o_{j'}, o_j\} \in \mathcal{I}$. Additionally, since $o_{j'} \neq o_j$ but $N_{o_{j'}}^{(i)} = N_{o_j}^{(i)}$, the first such property implies that $o_{j'}, o_j \in (A_{\leq i-1} \cup (O \setminus O_{\leq i-1})) \setminus A_{\leq i}$. Hence, $o_{j'}, o_j \in O_i^{(s)} \setminus A_{\leq i}$ and by the inequalities of (1), they must obey $f(o_{j'} \mid A_{\leq i}) \geq m_i$ and $f(o_j \mid A_{\leq i} + o_{j'}) \geq m_i$. Thus, $(\{o_{j'}, o_j\}, \{a\})$ is a valid (m_i, ε) -improvement for $A_{\leq i}$, yielding the desired contradiction.

Claim 3. Fix any $a \in A_{\leq L}$ and let $O' = \{o \in O_{\leq L} \mid a \in N_o\}$. By the same argument as in the previous claim, since the sets A_i are disjoint and $N_o = N_o^{(i)} \subseteq A_i$ for all $o \in O_i$, there must be some single $i \in [L]$ such that $a \in A_i$ and $N_o = N_o^{(i)}$ for all $o \in O'$. For this i , the sets $\{N_o^{(i)} \mid o \in A_{\leq i-1} \cup (O \setminus O_{\leq i-1})\}$ were constructed to satisfy the properties of Lemma 2.1. By the fourth such property, a appears in at most k of them, and hence, $|O'| \leq k$. ◀

4.4 Defining Auxiliary Weights Independent of $A_{\leq L}$

The weights, sets and properties given in the preceding subsections are sufficient to carry out a charging argument analogous to the one presented for the linear case in [38]. As in their proof, this charging argument would use the random shift τ to argue that, in expectation, an element $o \in O_{\leq L} \setminus O^{(s)}$ has value significantly less than all elements $a \in N_o$. The main difficulty in extending the argument to the submodular case is that the weights w and $\bar{w}$ depend crucially on the set $A_{\leq L}$ constructed by the algorithm, and hence, also on the random choice of τ . To circumvent this difficulty, we consider the incremental contribution of each element $o \in O$ to $f(O)$ alone. Formally, we order the elements $o_1, \dots, o_{|O|}$ of O as in the definition of $\bar{w}$, and then, for every $o_j \in O$, define the auxiliary weight $u(o_j) \triangleq f(o_j \mid \{o_1, \dots, o_{j-1}\})$. Note that $\sum_{o \in O} u(o) = f(O) - f(\emptyset)$. Moreover, the auxiliary weights u are independent of the execution of Algorithm 1, and in particular, the random choice of τ . The following observation relates the weights $\bar{w}$ and u . It follows from the strict down-monotonicity of O and the submodularity/linearity of f .

► **Observation 4.6.** For every $o_j \in O$, $u(o_j) > 0$ and $u(o_j) \geq \bar{w}(o_j)$. When f is linear, the last inequality holds as an equality.

For general submodular functions, we may have a strict inequality $u(o) > \bar{w}(o)$ for some $o \in O$. In the following section, this potential discrepancy results in a loss proportional to $u(o) - \bar{w}(o)$ in our charging analysis. However, the following lemma shows that this discrepancy can also be used to derive an alternative lower bound on $f(A_{\leq L})$. To get our results, we balance this lower bound against the bounds obtained in Section 4.5 via our charging scheme.

► **Lemma 4.7.** It holds that $\sum_{o \in O} [u(o) - \bar{w}(o)] \leq f(A_{\leq L}) - f(A_{\leq L} \mid O)$.

Proof. By the definitions of the weights $u(o)$ and $\bar{w}(o)$,

$$\begin{aligned}
\sum_{o \in O} [u(o) - \bar{w}(o)] &= \sum_{j=1}^{|O|} f(o_j \mid \{o_1, \dots, o_{j-1}\}) \\
&\quad - \sum_{o_j \in O} \max\{0, f(o_j \mid (A_{\leq L} - o_j) \cup \{o_1, \dots, o_{j-1}\})\} \\
&\leq \sum_{j=1}^{|O|} f(o_j \mid \{o_1, \dots, o_{j-1}\}) - \sum_{o_j \in O \setminus A_{\leq L}} f(o_j \mid A_{\leq L} \cup \{o_1, \dots, o_{j-1}\}) \\
&= [f(O) - f(\emptyset)] - [f(O \cup A_{\leq L}) - f(A_{\leq L})] \leq f(A_{\leq L}) - f(A_{\leq L} \mid O) . \quad \blacktriangleleft
\end{aligned}$$

4.5 The Charging Scheme

Recall that Section 4.3 classified the elements of O into three types: the elements of $O^{(s)}$, the elements of $O \setminus O_{\leq L}$, and the remaining elements (i.e., the elements of $O_{\leq L} \setminus O^{(s)}$). Each one of the first three lemmata of this section upper bounds $u(o)$ for the elements $o \in O$ of one of these types. Since $u(o)$ is the marginal contribution of o to O , later in the section we are able to derive a bound on $f(O)$ by combining the upper bounds on $u(o)$ proved by the three lemmata. As noted previously, all our upper bounds on $u(o)$ include a loss term that is proportional to $u(o) - \bar{w}(o)$. This loss term arises due to the possible discrepancy between the weights $\bar{w}$ and the auxiliary weights u we are bounding. To get our final bound on $f(O)$, we need also Lemma 4.7, which gives a bound on $f(O)$ that improves as $\sum_{o \in O} [u(o) - \bar{w}(o)]$ grows and thus allows us to cancel the above-mentioned loss terms.

► **Lemma 4.8.** *For each $o \in O^{(s)}$, $u(o) \leq (1 + \varepsilon) \cdot w(N_o) + [u(o) - \bar{w}(o)]$. Moreover, $\sum_{o \in O^{(s)}} w(N_o) \leq f(A_{\leq L} \mid \emptyset)$.*

Proof. Claim 1 of Lemma 4.5 immediately implies that for any $o \in O^{(s)}$,

$$u(o) = \bar{w}(o) + [u(o) - \bar{w}(o)] \leq (1 + \varepsilon) \cdot w(N_o) + [u(o) - \bar{w}(o)] .$$

Claim 2 of Lemma 4.5 further implies that every element of $A_{\leq L}$ appears in N_o for at most one element $o \in O^{(s)}$. Together with the fact that $w(a) > 0$ for every $a \in A_{\leq L}$ (claim 1 of Lemma 4.3), this implies that $\sum_{o \in O^{(s)}} w(N_o) \leq \sum_{a \in A_{\leq L}} w(a) = f(A_{\leq L} \mid \emptyset)$. $\blacktriangleleft$

► **Lemma 4.9.** *For each $o \in O \setminus O_{\leq L}$, $u(o) = u(o) - \bar{w}(o)$.*

Proof. This follows immediately from claim 3 of Lemma 4.3, which states that $\bar{w}(o) = 0$ for all $o \in O \setminus O_{\leq L}$. $\blacktriangleleft$

Next, we would like to bound $u(o)$ for the elements in $O_{\leq L} \setminus O^{(s)}$. Unlike the bounds in the previous two lemmata, this bound depends on the random shift τ selected by the algorithm. To formulate this bound, let us define, for each $o \in O$, $m^{(o)} \triangleq \min\{m_i \mid i \geq 0, m_i \geq u(o)\}$ to be the value of the smallest threshold m_i that is at least $u(o)$ (observe that $m^{(o)}$ is well-defined since $u(o) > 0$ by Observation 4.6 and $u(o) \leq W \leq m_0$ by submodularity). Additionally, we define the ratio $r_o \triangleq m^{(o)} / u(o)$. Notice that r_o is always in the range $[1, 2]$ since $u(o) \leq m^{(o)}$ and $m_{i-1} / m_i = 2$ for all i . Given this notation, we can now state the promised bound as Lemma 4.10. This lemma introduces a parameter $d \geq 1$, which is later used to control the balance between the bound of this lemma and the aforementioned bound from Lemma 4.7.

► **Lemma 4.10.** *For every $o \in O_{\leq L} \setminus O^{(s)}$ and any $d \geq 1$, $\rho_{o,d} \cdot u(o) \leq w(N_o) + d[u(o) - \bar{w}(o)]$, where $\rho_{o,d} \triangleq r_o$ when f is linear, and $\rho_{o,d} \triangleq \min \left\{ r_o, \frac{1 - \frac{1}{2}(1-1/d)}{1 - r_o^{-1}(1-1/d)} \right\}$ when f is a general submodular function.*

Proof. We begin the proof by showing that for all $o \in O_{\leq L} \setminus O^{(s)}$, there exist some threshold m_j such that $\bar{w}(o) \leq m_j \leq w(N_o)$. Since $o \in O_{\leq L}$, we must have $o \in O_i$ for some unique $i \in [L]$, and thus N_o is well-defined and obeys $|N_o| \geq 1$. Since $o \notin O^{(s)}$, this implies that either $\bar{w}(o) \leq m_i$ or $|N_o| \geq 2$. In the first case, $\bar{w}(o) \leq m_i \leq |N_o| \cdot m_i \leq w(N_o)$, where the last inequality holds since $N_o \subseteq A_i$ and $w(a) \geq m_i$ for every $a \in A_i$ (by claim 1 of Lemma 4.3). In the second case,

$$\bar{w}(o) \leq m_{i-1} = 2m_i \leq |N_o| \cdot m_i \leq w(N_o) ,$$

where the first inequality holds by claim 2 of Lemma 4.3 because $o \in O_i$, and the last inequality again holds since $N_o \subseteq A_i$ and $w(a) \geq m_i$ for every $a \in A_i$ (claim 1 of Lemma 4.3). Thus, in all cases there is indeed always a threshold m_j with $\bar{w}(o) \leq m_j \leq w(N_o)$.

We now consider two cases based on the relationship between $u(o)$ and the threshold m_j . If $u(o) \leq m_j$, then because $m^{(o)}$ is the smallest threshold that is at least $u(o)$, we get $u(o) = \frac{m^{(o)}}{r_o} \leq \frac{m_j}{r_o} \leq \frac{w(N_o)}{r_o}$. Rearranging this inequality yields

$$r_o \cdot u(o) \leq w(N_o) \leq w(N_o) + d[u(o) - \bar{w}(o)] , \quad (2)$$

where the last inequality follows from $u(o) \geq \bar{w}(o)$ (Observation 4.6). When the function f is linear, this case is the only possible case because $u(o) = \bar{w}(o) \leq m_j$ for such f (recall that $u(o) > 0$ by Observation 4.6). This completes the proof of the lemma for linear functions.

For general submodular functions, we may also have $u(o) > m_j$. In this case, $m^{(o)}$ must be at least m_{j-1} , which implies $m_j = \frac{1}{2}m_{j-1} \leq \frac{1}{2}m^{(o)} = \frac{u(o) \cdot r_o}{2}$. Thus,

$$\begin{aligned} \left(1 - \frac{1-1/d}{2}\right) \cdot u(o) &= m_j - \frac{1-1/d}{2} \cdot u(o) + [u(o) - m_j] \\ &\leq m_j - \frac{(1-1/d)}{2} \cdot \frac{2m_j}{r_o} + [u(o) - m_j] = (1 - r_o^{-1}(1-1/d))m_j + [u(o) - m_j] . \end{aligned}$$

Note that, since $r_o \geq 1$, it must hold that $1 - r_o^{-1}(1-1/d) \geq 1/d$. Thus, dividing both sides of the above inequality by $1 - r_o^{-1}(1-1/d)$ and simplifying yields

$$\begin{aligned} \frac{1 - \frac{1}{2}(1-1/d)}{1 - r_o^{-1}(1-1/d)} \cdot u(o) &\leq m_j + (1 - r_o^{-1}(1-1/d))^{-1} [u(o) - m_j] \\ &\leq m_j + d[u(o) - m_j] \leq w(N_o) + d[u(o) - \bar{w}(o)] , \quad (3) \end{aligned}$$

where we have used the fact that $\bar{w}(o) \leq m_j \leq w(N_o)$ in the last inequality. The guarantee of the lemma for general submodular functions now follows by taking the smaller of the two lower bounds on $w(N_o) + d[u(o) - \bar{w}(o)]$ given by (2) and (3) for the two cases. ◀

Combining all the above bounds, we obtain the following proposition.

► **Proposition 4.11.** *Assuming f is a submodular function such that $f(\emptyset) \geq 0$, the output set $A_{\leq L}$ of Algorithm 1 obeys, for every two values $d \geq 2$ and $d' \leq k+1$ and any strictly down-monotone $O \in \mathcal{I}$,*

$$(k+1+2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L})] \geq d' \cdot f(O) - d \cdot \sum_{o \in O} \mathbb{E}[u(o) - \bar{w}(o)] - \sum_{o \in O} [d' - \mathbb{E}[\rho_{o,d}]] \cdot u(o) ,$$

where $\rho_{o,d} \triangleq r_o$ when f is linear and $\rho_{o,d} \triangleq \min \left\{ r_o, \frac{1 - \frac{1}{2}(1-1/d)}{1 - r_o^{-1}(1-1/d)} \right\}$ when f is a general submodular function.

89:18 Submodular Maximization over a Matroid k -Intersection

Proof. Combining $\rho_{o,d}$ times the bounds from Lemmata 4.8 and 4.9 with the bound from Lemma 4.10, we obtain

$$\begin{aligned}
\sum_{o \in O} \rho_{o,d} \cdot u(o) &\leq \sum_{o \in O^{(s)}} 2 \cdot u(o) + \sum_{o \in O \setminus O_{\leq L}} 2 \cdot u(o) + \sum_{o \in O_{\leq L} \setminus O^{(s)}} \rho_{o,d} \cdot u(o) \\
&\leq 2 \cdot \sum_{o \in O^{(s)}} ((1 + \varepsilon)w(N_o) + [u(o) - \bar{w}(o)]) + 2 \cdot \sum_{o \in O \setminus O_{\leq L}} [u(o) - \bar{w}(o)] \\
&\quad + \sum_{o \in O_{\leq L} \setminus O^{(s)}} (w(N_o) + d[u(o) - \bar{w}(o)]) \\
&\leq \sum_{o \in O_{\leq L}} w(N_o) + (1 + 2\varepsilon) \sum_{o \in O^{(s)}} w(N_o) + d \cdot \sum_{o \in O} [u(o) - \bar{w}(o)] \\
&\leq \sum_{a \in A_{\leq L}} k \cdot w(a) + (1 + 2\varepsilon) \sum_{o \in O^{(s)}} w(N_o) + d \cdot \sum_{o \in O} [u(o) - \bar{w}(o)] \\
&\leq (k + 1 + 2\varepsilon) \cdot f(A_{\leq L} \mid \emptyset) + d \cdot \sum_{o \in O} [u(o) - \bar{w}(o)] ,
\end{aligned}$$

where the first inequality uses the facts that $\rho_{o,d} \leq r_o \leq 2$ and $u(o) > 0$ (by Observation 4.6); the third inequality follows from $d \geq 2$ and the inequality $u(o) - \bar{w}(o) \geq 0$ (which holds by Observation 4.6); the fourth inequality follows from the fact that each $a \in A_{\leq L}$ appears in at most k sets N_o (claim 3 of Lemma 4.5) and obeys $w(a) \geq 0$ (claim 1 of Lemma 4.3); and the final inequality follows from the inequality $\sum_{o \in O^{(s)}} w(N_o) \leq f(A_{\leq L} \mid \emptyset)$ proved by Lemma 4.8.

Taking expectations on both sides, and noting that $f(O)$ and $u(o)$ do not depend on the randomness of the algorithm, we obtain

$$\begin{aligned}
d' \cdot f(O) &= d' \cdot f(\emptyset) + \mathbb{E} \left[\sum_{o \in O} \rho_{o,d} \cdot u(o) \right] + \mathbb{E} \left[\sum_{o \in O} (d' - \rho_{o,d}) \cdot u(o) \right] \\
&\leq d' \cdot f(\emptyset) + (k + 1 + 2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L} \mid \emptyset)] + d \cdot \sum_{o \in O} \mathbb{E}[u(o) - \bar{w}(o)] \\
&\quad + \sum_{o \in O} [d' - \mathbb{E}[\rho_{o,d}]] \cdot u(o) \\
&\leq (k + 1 + 2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L})] + d \cdot \sum_{o \in O} \mathbb{E}[u(o) - \bar{w}(o)] + \sum_{o \in O} [d' - \mathbb{E}[\rho_{o,d}]] \cdot u(o) .
\end{aligned}$$

Rearranging this inequality gives the claimed result. $\blacktriangleleft$

Using Proposition 4.11 we can now prove our main results for linear and monotone submodular functions. We begin with the simpler case of linear functions.

► Theorem 4.12. *When f is a linear function, the approximation guarantee of Algorithm 1 is at most $(k + 1) \ln 2 + O(\varepsilon)$.*

Proof. For linear functions, $u(o) = \bar{w}(o) = f(\{o\})$ for all $o \in O$, and hence, the terms $\mathbb{E}[u(o) - \bar{w}(o)]$ in Proposition 4.11 all vanish. Then, since $\rho_{o,d} = r_o$ for linear functions, we have

$$(k + 1 + 2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L})] \geq d' \cdot f(O) - \sum_{o \in O} [d' - \mathbb{E}[r_o]] \cdot u(o) .$$

Let $d' = \ln^{-1} 2$ (notice that $d' \leq 2 \leq k+1$). Below, we prove that $\mathbb{E}[r_o] = d'$ for every $o \in O$, which implies that the last term on the rightmost side of the above inequality is 0. Rearranging the above inequality then yields

$$\mathbb{E}[f(A_{\leq L})] \geq \frac{d'}{k+1+2\varepsilon} \cdot f(O) = \frac{\ln^{-1} 2}{k+1+2\varepsilon} \cdot f(O) = \frac{f(O)}{(k+1) \ln 2 + O(\varepsilon)} .$$

Recalling that there must exist at least one optimal solution O that is also strictly down-monotone, the theorem follows from this inequality. The rest of the proof is devoted to proving that indeed $\mathbb{E}[r_o] = d'$.

Recall that for each $o \in O$, r_o is the ratio $m^{(o)}/u(o)$, where $m^{(o)} = \min\{m_i \mid i \geq 0, m_i \geq u(o)\}$ is the lowest value of a threshold m_i that is at least $u(o)$. We also recall that, for each i , $m_i = W\tau 2^{-i} = W2^{-i+\alpha}$, where α is chosen uniformly at random from $(0, 1]$. Intuitively, this implies that $r_o = 2^\beta$ for some $\beta \in [0, 1)$ that is obtained by cyclically shifting α , and is thus, distributed uniformly at random from $[0, 1)$. We give a formal proof of this fact in the full version of this paper [15]. Given this fact,

$$\mathbb{E}[r_o] = \int_0^1 2^\beta d\beta = \frac{2^\beta}{\ln 2} \Big|_{\beta=0}^{\beta=1} = \ln^{-1} 2 = d' . \quad \blacktriangleleft$$

The following proposition gives a guarantee for Algorithm 1 when the objective is a general submodular (rather than linear) function. Our main result for monotone submodular functions readily follows from this proposition (see Corollary 4.14), but the proposition itself applies also to non-monotone submodular functions, and is used in Section 5 to derive our main result for such functions.

► **Proposition 4.13.** *When f is a non-negative submodular function, the output set $A_{\leq L}$ of Algorithm 1 obeys, for every $d \geq 2$ and any strictly down-monotone $O \in \mathcal{I}$,*

$$(k+d+1+2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L})] \geq d' \cdot f(O) + d \cdot \mathbb{E}[f(A_{\leq L} \mid O)] ,$$

where $d' = \frac{1-1/d}{2 \ln 2} + \frac{d+1}{2d}$.

Proof. Note that $d' = \frac{1-1/d}{2 \ln 2} + \frac{d+1}{2d} \leq \frac{1}{2 \ln 2} + \frac{3d/2}{2d} < 1 + 3/4 < 2$. Thus, Proposition 4.11 and Lemma 4.7 give together

$$\begin{aligned} (k+1+2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L})] &\geq d' \cdot f(O) - d \cdot \mathbb{E} \left[\sum_{o \in O} u(o) - \bar{w}(o) \right] - \sum_{o \in O} [d' - \mathbb{E}[\rho_{o,d}]] \cdot u(o) \\ &\geq d' \cdot f(O) - d \cdot \mathbb{E}[f(A_{\leq L}) - f(A_{\leq L} \mid O)] - \sum_{o \in O} [d' - \mathbb{E}[\rho_{o,d}]] \cdot u(o) . \end{aligned}$$

Below we prove that $\mathbb{E}[\rho_{o,d}] = d'$ for every $o \in O$, which implies that the last term on the rightmost side of the above inequality is 0. Notice that, given this observation, the proposition follows by using the linearity of expectation and rearranging the resulting inequality. The rest of the proof is devoted to proving that indeed $\mathbb{E}[\rho_{o,d}] = d'$.

Recall that the full version of this paper [15] proves that $r_o = 2^\beta$ for β that is distributed uniformly at random from $[0, 1)$. This property of r_o underlies the calculation below.

$$\begin{aligned} \mathbb{E}[\rho_{o,d}] &= \mathbb{E} \left[\min \left\{ r_o, \frac{1 - \frac{1}{2}(1-1/d)}{1 - r_o^{-1}(1-1/d)} \right\} \right] = \int_0^1 \min \left\{ 2^\beta, \frac{1 - \frac{1}{2}(1-1/d)}{1 - 2^{-\beta}(1-1/d)} \right\} d\beta \\ &= \int_0^{\log_2(1+\frac{1}{2}(1-1/d))} 2^\beta d\beta + \int_{\log_2(1+\frac{1}{2}(1-1/d))}^1 \frac{1 - \frac{1}{2}(1-1/d)}{1 - 2^{-\beta}(1-1/d)} d\beta \end{aligned}$$

$$\begin{aligned}
&= \frac{2^\beta}{\ln 2} \Big|_{\beta=0}^{\beta=\log_2(1+\frac{1}{2}(1-1/d))} + \frac{1 - \frac{1}{2}(1-1/d)}{\ln 2} \cdot \ln(2^\beta - (1-1/d)) \Big|_{\beta=\log_2(1+\frac{1}{2}(1-1/d))}^{\beta=1} \\
&= \frac{1-1/d}{2\ln 2} + \frac{1 - \frac{1}{2}(1-1/d)}{\ln 2} \cdot (\ln(1+1/d) - \ln(\frac{1}{2}(1+1/d))) \\
&= \frac{1-1/d}{2\ln 2} + 1 - \frac{1}{2}(1-1/d) = \frac{1-1/d}{2\ln 2} + \frac{d+1}{2d} = d' . \quad \blacktriangleleft
\end{aligned}$$

► **Corollary 4.14.** *When f is a non-negative monotone submodular function, Algorithm 1 has an approximation guarantee of at most $\frac{2k \ln 2}{1+\ln 2} + O(\sqrt{k}) \leq 0.819k + O(\sqrt{k})$.*

Proof. In the case that f is monotone, the expression $f(A_{\leq L} \mid O)$ in Proposition 4.13 is always non-negative. Thus, for any $d \geq 2$, we have

$$(k + d + 1 + 2\varepsilon) \cdot \mathbb{E}[f(A_{\leq L})] \geq d' \cdot f(O) ,$$

where $d' = \frac{1-1/d}{2\ln 2} + \frac{d+1}{2d}$. Substituting d' into this inequality, we get that the ratio $f(O)/\mathbb{E}[f(A_{\leq L})]$ is at most

$$\frac{k + d + 1 + 2\varepsilon}{\frac{1-1/d}{2\ln 2} + \frac{d+1}{2d}} = \frac{k + d + 1 + 2\varepsilon}{\frac{1+\ln 2}{2\ln 2} - d^{-1} \cdot \frac{1-\ln 2}{2\ln 2}} \leq \frac{(k + d + 1 + 2\varepsilon)(1 + 2d^{-1} \cdot \frac{1-\ln 2}{1+\ln 2})}{\frac{1+\ln 2}{2\ln 2}} ,$$

where the inequality uses the fact that for every two positive values a, b obeying $2b \leq a$, we have $\frac{1}{a-b} \leq \frac{1+2b/a}{a}$. Selecting $d = 2\sqrt{k}$ (notice that $d \geq 2$) makes the above expression equal to $\frac{2k \ln 2}{1+\ln 2} + O(\sqrt{k})$, which proves the corollary since there must exist some optimal solution O that is also strictly down-monotone. $\blacktriangleleft$

In the full version of this paper [15], we show that it is possible to simulate Algorithm 1 in $O(\varepsilon^{-1}|E|^4) = \text{Poly}(|E|, \varepsilon^{-1})$ time. Together with Theorems 4.12 and Corollary 4.14, this immediately gives the following.

► **Theorem 1.1.** *For every $\varepsilon > 0$, there exists an algorithm that runs in $\text{Poly}(|E|, \varepsilon^{-1})$ time and guarantees an approximation ratio of $\frac{2k \ln 2}{1+\ln 2} + O(\sqrt{k}) \leq 0.819k + O(\sqrt{k})$ for maximizing a non-negative monotone submodular function $f: 2^E \rightarrow \mathbb{R}_{\geq 0}$ under a matroid k -parity constraint $(E, \mathcal{I})$. Moreover, if f is linear, the approximation ratio of the algorithm improves to $(k+1)\ln 2 + O(\varepsilon) \leq 0.694k + 0.694 + O(\varepsilon)$.*

5 Algorithm for Non-monotone Submodular Functions

In this section, our goal is to prove Theorem 1.2. As mentioned in Section 1.1, we do that using the framework of [13] for adapting algorithms for maximizing non-negative monotone submodular functions to work also for functions that are not necessarily monotone. Such algorithms often have guarantees that depend on $f(B \cup O)$, where B is their output set and O is an optimal solution, which is a good thing when f is monotone (since $f(B \cup O) \geq f(O)$ in this case), but problematic in the non-monotone case. Notice that the guarantee of Algorithm 1 given by Proposition 4.13 indeed conforms to this general trend as it involves the term $f(A_\ell \mid O) = f(A_\ell \cup O) - f(O)$.

The framework of [13] (and the earlier similar frameworks of [18, 33]) is based on the observation that the dependence on the term $f(B \cup O)$ is only bad when $f(B \cup O)$ is significantly smaller than $f(O)$, or in other words, when the elements of B significantly “damage” the optimal solution. This observation motivates the following meta-algorithm: execute the algorithm for monotone functions to get a solution B_1 , remove the elements of

B_1 from the ground set, then execute the algorithm for monotone functions again to get a second solution B_2 and then return the better among the two solutions. For the first run of the algorithm, we obtain a guarantee relating $f(B_1)$ to $f(B_1 \cup O)$, and for the second run we obtain a guarantee relating $f(B_2)$ to $f(B_2 \cup (O \setminus B_1))$. Since B_1 and B_2 are disjoint, submodularity implies that $f(B_1 \cup O) + f(B_2 \cup (O \setminus B_1)) \geq f(O \setminus B_1) + f(B_1 \cup B_2 \cup O)$, and so the better of $B_1 \cup O$ and $B_2 \cup (O \setminus B_1)$ is at least $\frac{1}{2}f(O \setminus B_1)$. If $f(O \setminus B_1)$ is comparable to $f(O)$, then we are done. In the remaining case, $f(O \setminus B_1)$ is significantly smaller than $f(O)$, which by submodularity implies that $f(O \cap B_1)$ must be large. To handle this case as well, we can apply an approximation algorithm for *unconstrained* submodular maximization to the set B_1 to obtain a third solution $B'_1 \subseteq B_1$ whose value is at least $\frac{1}{2} \cdot f(B_1 \cap O)$. Here, we use the Double Greedy algorithm of Buchbinder et al. [5] for this purpose.⁷

To get the best guarantee via this general framework, it is necessary to repeat the above steps multiple times, which generates a series of disjoint solutions $B_1, B_2, \dots$ and a series of additional solutions $B'_1 \subseteq B_1, B'_2 \subseteq B_2, \dots$. Algorithm 2 implements this in our context. It receives as input a non-negative (not necessarily monotone) submodular function $f: 2^E \rightarrow \mathbb{R}_{\geq 0}$, a matroid k -parity constraint $(E, \mathcal{I})$, and a positive integer parameter ℓ determining the number of times that the algorithm for monotone functions (Algorithm 1) is executed. As expected, the i -th execution of Algorithm 1 (by Line 3 of Algorithm 2) is done with respect to a restriction of the input instance to the ground set E_{i-1} obtained by removing the elements of $\cup_{j=1}^{i-1} B_j$ from E . We can use Proposition 4.13 to analyze the guarantee of Algorithm 1 on this restricted instance because its objective function $f|_{E_i}$ inherits the submodularity of f and the restricted constraint $(E_{i-1}, 2^{E_{i-1}} \cap \mathcal{I})$ remains a matroid k -parity constraint.

■ **Algorithm 2** REPETITIONS ALGORITHM $(E, f, \mathcal{I}, \ell)$.

```

1 Let  $E_0 \leftarrow E$ .
2 for  $i = 1$  to  $\ell$  do
3   Execute Algorithm 1 on the function  $f|_{E_i}$ , the constraint  $(E_{i-1}, 2^{E_{i-1}} \cap \mathcal{I})$ , and an
   arbitrary constant value  $\varepsilon \in (0, 1)$ . Let  $B_i$  be the output set of this algorithm.
4   Execute the Double Greedy algorithm of [5] on the function  $f|_{B_i}$ , and let  $B'_i$  be
   the output set of this algorithm.
5   Let  $E_i \leftarrow E_{i-1} \setminus B_i$ .
6 return the set  $B^* \in \{B_i \mid i \in [\ell]\} \cup \{B'_i \mid i \in [\ell]\}$  with maximum value  $f(B^*)$ .
```

Due to space constraints, the formal analysis of Algorithm 2, showing that it indeed has the properties stated by Theorem 1.2, is deferred to the full version of this paper [15].

References

- 1 Esther M. Arkin and Refael Hassin. On local search for weighted k -set packing. *Mathematics of Operations Research*, 23(3):640–648, 1998. doi:10.1287/MOOR.23.3.640.
- 2 Alexander I. Barvinok. New algorithms for linear k -matroid intersection and matroid k -parity problems. *Mathematical Programming*, 69:449–470, 1995. doi:10.1007/BF01585571.

⁷ Formally, Double Greedy gets a non-negative submodular function $f: 2^E \rightarrow \mathbb{R}_{\geq 0}$ and returns a set $T \subseteq E$ such that $\mathbb{E}[f(T)] \geq \frac{1}{2} \cdot \max_{S \subseteq E} f(S)$. To employ this algorithm for our purpose, we pass to it the restriction of f to the ground set B_1 , which guarantees that its output set B'_1 is a subset B_1 and that $\mathbb{E}[f(B'_1)] \geq \frac{1}{2} \cdot \max_{S \subseteq B_1} f(S) \geq \frac{1}{2} \cdot f(B_1 \cap O)$.

- 3 Piotr Berman. A $d/2$ approximation for maximum weight independent set in d -claw free graphs. *Nordic J. Computing*, 7(3):178–184, 2000.
- 4 Piotr Berman and Piotr Krysta. Optimizing misdirection. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 192–201, 2003. URL: <http://dl.acm.org/citation.cfm?id=644108.644142>.
- 5 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. *SIAM J. Computing*, 44(5):1384–1402, 2015. doi:10.1137/130929205.
- 6 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015. doi:10.1007/S10107-015-0900-7.
- 7 Barun Chandra and Magnús M. Halldórsson. Greedy local improvement and weighted set packing approximation. *J. Algorithms*, 39(2):223–240, 2001. doi:10.1006/JAGM.2000.1155.
- 8 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming ICALP*, pages 318–330, 2015. doi:10.1007/978-3-662-47672-7_26.
- 9 Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Computing*, 40(6):1740–1766, 2011. doi:10.1137/080733991.
- 10 William H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM J. Computing*, 15(4):948–957, 1986. doi:10.1137/0215066.
- 11 Marek Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *IEEE Symposium on Foundations of Computer Science, FOCS*, pages 509–518, 2013. doi:10.1109/FOCS.2013.61.
- 12 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization — Eureka, You Shrink!: Papers Dedicated to Jack Edmonds*, pages 11–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. doi:10.1007/3-540-36478-1_2.
- 13 Moran Feldman, Christopher Harshaw, and Amin Karbasi. How do you want your greedy: Simultaneous or repeated? *J. Machine Learning Research*, 24:72:1–72:87, 2023. URL: <https://jmlr.org/papers/v24/21-0782.html>.
- 14 Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k -exchange systems - (extended abstract). In *European Symposium on Algorithms (ESA)*, pages 784–798, 2011. doi:10.1007/978-3-642-23719-5_66.
- 15 Moran Feldman and Justin Ward. Submodular maximization over a matroid k -intersection: Multiplicative improvement over greedy. *CoRR*, abs/2602.08473, 2026. doi:10.48550/arXiv.2602.08473.
- 16 Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. An analysis of approximations for maximizing submodular set functions – II. *Mathematical Programming Studies*, 8:73–87, 1978.
- 17 Martin Fürer and Huiwen Yu. Approximating the k -set packing problem by local improvements. In *International Symposium on Combinatorial Optimization ISCO*, pages 408–420, 2014. doi:10.1007/978-3-319-09174-7_35.
- 18 Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics (WINE)*, pages 246–257, 2010. doi:10.1007/978-3-642-17572-5_20.
- 19 Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k -set packing. *Computational Complexity*, 15(1):20–39, 2006. doi:10.1007/S00037-006-0205-6.
- 20 Chien-Chung Huang and François Sellier. Matroid-constrained vertex cover. *Theoretical Computer Science*, 965:113977, 2023. doi:10.1016/j.TCS.2023.113977.
- 21 Satoru Iwata and Yusuke Kobayashi. A weighted linear matroid parity algorithm. *SIAM J. Computing*, 51(2):17–238, 2022. doi:10.1137/17M1141709.

- 22 T.A. Jenkyns. The efficacy of the “greedy” algorithm. In *Southeastern Conf. on Combinatorics, Graph Theory and Computing*, pages 341–350, 1976.
- 23 Per M. Jensen and Bernhard Korte. Complexity of matroid property algorithms. *SIAM J. Computing*, 11(1):184–190, 1982. doi:10.1137/0211014.
- 24 Richard M. Karp. Reducibility among combinatorial problems. In *Symposium on the Complexity of Computer Computations*, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 25 Bernhard Korte and Dirk Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Mathematics*, 2:65–74, 1978. doi:10.1016/S0167-5060(08)70322-4.
- 26 E. Lawler. *Combinatorial optimization - networks and matroids*. Holt, Rinehart and Winston, New York, 1976.
- 27 Euiwoong Lee, Ola Svensson, and Theophile Thiery. Asymptotically optimal hardness for k -set packing and k -matroid intersection. In *ACM Symposium on Theory of Computing (STOC)*, pages 54–61, 2025. doi:10.1145/3717823.3718142.
- 28 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010. doi:10.1287/MOOR.1100.0463.
- 29 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Matroid matching: The power of local search. *SIAM J. Computing*, 42(1):357–379, 2013. doi:10.1137/11083232X.
- 30 László Lovász. Matroid matching and some applications. *J. of Combinatorial Theory, Series B*, 28(2):208–236, 1980. doi:10.1016/0095-8956(80)90066-0.
- 31 László Lovász. The matroid matching problem. In L. Lovász and V. T. Sós, editors, *Algebraic Methods in Graph Theory, Vol. II*, pages 495–517. North-Holland, Amsterdam, 1981.
- 32 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009. doi:10.1016/J.TCS.2009.07.027.
- 33 Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning (ICML)*, pages 1358–1367, 2016. URL: <http://proceedings.mlr.press/v48/mirzasoleiman16.html>.
- 34 Meike Neuwohner. An improved approximation algorithm for the maximum weight independent set problem in d -claw free graphs. In *International Symposium on Theoretical Aspects of Computer Science, STACS*, pages 53:1–53:20, 2021. doi:10.4230/LIPIcs.STACS.2021.53.
- 35 Meike Neuwohner. Passing the limits of pure local search for weighted k -set packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1090–1137, 2023. doi:10.1137/1.9781611977554.CH41.
- 36 Meike Neuwohner. The limits of local search for weighted k -set packing. *Mathematical Programming*, 206(1):389–427, 2024. doi:10.1007/S10107-023-02026-3.
- 37 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg, 2003.
- 38 Neta Singer and Theophile Thiery. Better approximation for weighted k -matroid intersection. In *ACM Symposium on Theory of Computing (STOC)*, pages 1142–1153, 2025. doi:10.1145/3717823.3718219.
- 39 Maxim Sviridenko and Justin Ward. Large neighborhood local search for the maximum set packing problem. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 792–803, 2013. doi:10.1007/978-3-642-39206-1_67.
- 40 Theophile Thiery. *Approximation Algorithms for Independence Systems*. Ph.D. thesis, Queen Mary University of London, London, United Kingdom, 2023.
- 41 Theophile Thiery and Justin Ward. An improved approximation for maximum weighted k -set packing. In *ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1138–1162, 2023. doi:10.1137/1.9781611977554.CH42.
- 42 Justin Ward. A $(k+3)/2$ -approximation algorithm for monotone submodular k -set packing and general k -exchange systems. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 42–53, 2012. doi:10.4230/LIPIcs.STACS.2012.42.