Submodular Order Functions and Assortment Optimization

Rajan Udwani [1]

Abstract

We define a new class of set functions that in addition to being monotone and subadditive, also admit a very limited form of submodularity defined over a permutation of the ground set. We refer to this permutation as a submodular order. We give fast algorithms with strong approximation guarantees for maximizing submodular order functions under a variety of constraints. Applying this new notion to the problem of constrained assortment optimization in fundamental choice models, we obtain new algorithms that are both faster and have stronger approximation guarantees (in some cases, first algorithm with constant factor guarantee). We also show an intriguing connection to the maximization of monotone submodular functions in the streaming model, where we recover best known approximation guarantees as a corollary of our results.

1. Introduction

Given a ground set N, the problem of finding a (feasible) subset S that maximizes a monotone[1] and submodular[2] set function $f : 2^N \to \mathbb{R}$, subject to some constraint, has received significant and continued interest in ML and many other communities due to a variety of applications, such as, feature selection (Das & Kempe, 2011; Wei et al., 2015), sensor placement (Krause et al., 2008b; Leskovec et al., 2007), influence maximization (Krause et al., 2008a), data summarization (Badanidiyuru et al., 2014), and many others. Despite this richness of submodular functions, there are important subset selection problems where the objective (f) is *not* submodular.

Of particular interest to us is the setting of *assortment optimization* where the objective is not submodular[3]. The problem arises widely in many industries including retailing and online advertising, where given a universe of substitutable products, the seller wishes to select a subset of products (called an assortment) with maximum expected revenue. The effect of substitution between products is captured by a *choice model* $\phi : N \times 2^N \to [0, 1]$. Given assortment S, the probability that customer chooses product $i \in S$ is given by $\phi(i, S)$. Customer may choose an *outside option* that is not in S with probability $1 - \sum_{i \in S} \phi(i, S)$. Given (fixed) prices $(r_i)_{i \in N}$, the expected revenue is,

$$R_\phi(S) = \sum_{i \in S} r_i \phi(i, S).$$

We assume that the choice model is given to us and we are interested in the problem of finding the revenue optimizing assortment subject to certain types of constraints that we will discuss shortly[4]. Unfortunately, for a general choice model Aouad et al. (2018a) show that even the unconstrained assortment optimization problem is hard to approximate. Given this intractability and lack of general structure, the field of assortment optimization has evolved through algorithms that are tailored for individual families of choice models.

Motivated by this, we are interested in finding a unifying structural property (for set functions) that leads to computationally tractable optimization problems and captures both monotone submodular functions and assortment optimization problems as special cases. As a starting point, consider monotone functions that satisfy the following mild condition,

Subadditivity: $f(A) + f(B) \geq f(A \cup B)\ \forall A, B \subseteq N.$

Every non-negative submodular function is subadditive and the revenue function in assortment optimization is also subadditive for a general family of random utility based choice

[1]Department of IEOR, University of California, Berkeley, USA. Correspondence to: Rajan Udwani <rudwani@berkeley.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]A set function $f : 2^N \to \mathbb{R}$ is monotone if $f(A) \geq f(B) \quad \forall B \subseteq A$.

[2]A set function $f : 2^N \to \mathbb{R}$ is submodular if $f(A \cup \{e\}) - f(A) \leq f(B \cup \{e\}) - f(B)\ \forall B \subseteq A \subseteq N$, $e \in N \backslash A$.

[3]In fact, the objective is also non-monotone but we can transform it into a monotone function (see Section 3.2).

[4]In practice, a choice model can either be estimated from historical data (if available) prior to the assortment decision (Kök et al., 2008), or, one may implement a bandit algorithm where data is collected over time and an optimal assortment is computed at each step based on an evolving estimate of the choice model (Agrawal et al., 2019).

models (Kök et al., 2008). Using the (standard) normalization $f(\emptyset) = 0$, monotonicity implies that f is non-negative. Given query access to an oracle for f, we are interested in solving,

$$\arg\max_{S \in \mathcal{F}} f(S),$$

where $\mathcal{F}$ corresponds to one of the following types of constraints that commonly arise in applications,

Cardinality (k)	$\{S \mid \|S\| \leq k\}$,
Budget/Knapsack ($\{b_i\}_{i \in N}, B$)	$\{S \mid b(S) \leq B\}$,
Matroid ($\mathcal{I}$)	$\{S \mid S \in \mathcal{I}\}$,

here $b(S) = \sum_{i \in S} b_i$ and $\mathcal{I}$ is the family of independent sets of a matroid[5]. In the absence of a constraint the ground set N maximizes function value (due to monotonicity). However, constrained optimization is computationally intractable for monotone subadditive functions even with a cardinality constraint.

Theorem 1.1 (Adapted from Theorem 6.1 in (Dobzinski et al., 2010))**.** *Any algorithm that makes polynomial number of queries (in size n of ground set), cannot have approximation guarantee better than $n^{-\Omega(1)}$ for maximizing a monotone subadditive function subject to cardinality constraint.*

In light of this impossibility result, we introduce a mild structural property, namely, a *submodular order*, and show that it drastically alters the optimization landscape by allowing efficient algorithms with constant factor guarantee for all constraint families of interest. To define this new notion we introduce some notation. Given a permutation π over N, we index elements in the order given by π. Let $r_\pi(S)$ denote the element in S with the largest index and $l_\pi(S)$ the element with the smallest index. It helps to visualize the indexing as an increasing order from left to right; $r_\pi(S)$ is the rightmost element of S and $l_\pi(S)$ is the leftmost. We define the marginal value function,

$$f(X|S) = f(X \cup S) - f(S).$$

Submodular order: A permutation π of elements in N is a submodular order if for all sets $B \subseteq A$ and C to the right of A i.e., $l_\pi(C) > r_\pi(A)$, we have

$$f(C \mid A) \leq f(C \mid B). \tag{1}$$

We refer to function with submodular order π simply as *π-submodular ordered* for brevity.

Comparison with submodular functions: For a submodular function f, inequality (1) holds for *every* set C. In fact, it can be shown that a function is submodular if and only if every permutation of the ground set is a submodular order. From an optimization standpoint, there is a substantial difference between submodularity and submodular order. Consider the classic work of Nemhauser et al. (1978), which showed that the greedy algorithm that iteratively builds a solution by adding an element with the largest marginal value, achieves a guarantee of $(1 - 1/e)$ for maximizing monotone submodular functions subject to cardinality constraint. The following example demonstrates that greedy can be arbitrarily bad even for a simple submodular order function.

Example 1: Consider a ground set $\{1, \cdots, 2k+1\}$. Let singleton values $f(\{e\})$ equal 1 for all *good* elements $e \in \{1, \cdots, k\}$, equal ϵ for all *poor* elements $e \in \{k+1, \cdots, 2k\}$ and finally, let $f(\{2k+1\}) = 1 + \epsilon$. Let the first $2k$ elements be modular i.e., the value of a set $S = S_1 \cup S_2$ with subset S_1 of good elements and S_2 of poor elements is simply $|S_1| + \epsilon|S_2|$. Finally, let $f(S_1 \cup S_2 \cup \{2k+1\}) = \max\{|S_1|, 1+\epsilon\} + \epsilon|S_2|$ for every subset S_1 of good elements and S_2 of poor elements. It can be verified that this function is monotone, subadditive, and the natural indexing $\{1, \cdots, 2k+1\}$ is a submodular order. Notice that the set of all good elements has value k and this is the optimal set of cardinality k for every $k \geq 2$. However, the greedy algorithm would first pick element $2k+1$ and subsequently pick $k-1$ poor elements resulting in a total value of $1 + \epsilon\, k$. For $\epsilon \to 0$, this is only a trivial $1/k$ approximation of the optimal value.

What about other algorithms for submodular maximization? One may wonder if a greedy algorithm that chooses m elements at each step for $m \geq 2$, admits better performance. The example above can be easily modified to show that this family of algorithms is arbitrarily bad in the worst case for any constant m. The more general schema of *continuous* greedy algorithms (see Calinescu et al. (2011), Feldman et al. (2011)), is similarly ineffective.

Local search is another well studied family of algorithms for submodular maximization (see (Feige et al., 2011)). These algorithms incrementally improve the value of a solution by swapping elements. For submodular order functions, local search fails to improve the value of a poor solution. In the example above, performing local search by swapping one element at a time (with the objective of improving function value) will not find any improvement on the set $\{k+1, \cdots, 2k+1\}$. This set is a local maxima with value $1/k$ of the optimal. More generally, natural modifications of the example show that local search over m-tuples is arbitrarily bad in the worst case for any constant m. We now discuss our main algorithmic results for submodular order maximization and its applications.

[5]Both budget and matroid constraint include cardinality constraint as a special case.

1.1. Overview of Our Results

All our approximation results hold more strongly under a milder version of submodular order, called *weak* submodular order, defined as follows.

***Weak* submodular order and π nested sets:** Sets $B \subseteq A$ are π-nested if the left most element of $A \backslash B$ is to the right of B i.e., $l_\pi(A \backslash B) > r_\pi(B)$. An order π is a weak submodular order if $f(C \mid A) \leq f(C \mid B)$ for all π-nested sets $B \subseteq A$ and set C with $l_\pi(C) > r_\pi(A)$.

The weak submodular order property does not impose any requirements on sets A, B that are not π-nested. Since this is a milder condition the resulting family of functions is broader than (strong) submodular order functions. An algorithm with guarantee α for weak submodular order functions is also an α approximation for submodular order functions. Conversely, the upper bound on approximation guarantee for submodular order functions also applies to functions with weak submodular order. It is not meant to be obvious a priori but we establish that, in general, the stronger notion does not offer any benefits in terms of approximation guarantee. We establish the following approximation guarantees.

Theorem 1.2 (Approximation Guarantees)**.** *For constrained maximization of a monotone subadditive function f with a (known) weak submodular order, we give algorithms with the following performance guarantees,*

(i) ***Cardinality constraint:*** *There is a $(1-\epsilon)\,0.5$ approximation algorithm with query complexity $O(\frac{n}{\epsilon}\log k)$, for any choice of $\epsilon \in (0,1)$.*

(ii) ***Budget constraint:***

- *(a) There is a $\frac{1}{3} - \epsilon$ approximation algorithm with query complexity $O(\frac{n}{\epsilon}\log n)$, $\forall \epsilon \in (0,1)$.*
- *(b) There is a $0.5 - \epsilon$ approximation algorithm with query complexity $O(\frac{1}{\epsilon}n^{1+\frac{1}{\epsilon}}\log n)$, $\forall \epsilon \in (0, 0.5)$.*

(iii) ***Matroid constraint:*** *There is a 0.25 approximation algorithm with query complexity $O(nd)$, where d is the rank of the matroid.*

The ideas behind the algorithms referenced in Theorem 1.2 are discussed subsequently in Section 2. We also establish that our guarantees for cardinality and budget constraint are essentially tight. This gives a clear separation between the maximization of submodular order functions and the maximization of submodular functions that admit stronger $(1-1/e)$ approximation algorithms (Nemhauser et al., 1978; Sviridenko, 2004; Calinescu et al., 2011).

Theorem 1.3 (Upper Bound)**.** *Given a monotone and subadditive function with (strong) submodular order on a ground set of size n, any algorithm for the cardinality constrained maximization problem that makes at most $poly(n)$ queries cannot have approximation guarantee better than $0.5 + \epsilon$ for any $\epsilon > 0$.*

As a direct corollary of Theorem 1.3, no polynomial query algorithm can have approximation guarantee strictly better than 0.5 for the budget constrained and matroid constrained versions. Table 1 provides a summary of our results.

Remark: Given a noisy value oracle $\hat{f}$ for a submodular order function f, such that $(1-\delta)\,f(S) \leq \hat{f}(S) \leq (1+\delta)\,f(S)\ \forall S \subseteq N$, we show that our approximation guarantees continue to hold but are reduced by an additional multiplicative factor of $\left(1 - O(\frac{n\delta}{1-\delta})\right)$.

Table 1. Results for maximization of (weak) submodular order functions. The first column shows approximation guarantees, the second column shows upper bounds. n denotes the size of the ground set, k the cardinality parameter, and d is the matroid rank. For budget constraint, we also give a fast ($O(\frac{n}{\epsilon}\log n)$ time) $\frac{1}{3} - \epsilon$ approximation.

	LOWER	UPPER	# QUERIES
CARDINALITY	$0.5 - \epsilon$	0.5	$O(\frac{n}{\epsilon}\log k)$
BUDGET	$0.5 - \epsilon$	0.5	$O(\frac{1}{\epsilon}n^{1+\frac{1}{\epsilon}}\log n)$
MATROID	0.25	0.5	$O(nd)$

Applications of submodular order functions: We show that the objective in assortment optimization, while not submodular, is compatible with the weaker notion of submodular order for some fundamental choice models. Consequently, we achieve new results for constrained assortment optimization in these models under a *unified framework*. We also show an intriguing connection to the maximization of monotone submodular functions in the streaming model (Badanidiyuru et al., 2014), where we recover best known approximation guarantees as a corollary of our results.

Outline for rest of the paper. We discuss the main insights behind our algorithms for submodular order maximization in Section 2, where we include the algorithms for cardinality constraint and matroid constraint in detail. We defer the details of the algorithms for budget constraint to Appendix A.1. In Section 3, we present applications of submodular order function maximization in more detail. First, we discuss new results for assortment optimization. Then, in Section 3.3, we discuss connections with streaming maximization of submodular functions. Section 4 concludes the discussion with several directions for further research. Due to space limitation, we present the analysis of our algorithms (Theorem 1.2) and the proof of our upper bound result (Theorem 1.3) in Appendix B, where we start with useful bounds for (weak) submodular order functions that drive the analysis of our algorithms[6]. Proofs for main results on assortment optimization are included in Appendix D.

[6]For Theorem 1.2, we prove part (i) for cardinality constraint in

2. Algorithms for Submodular Order Functions

In Example 1, we saw that in the absence of submodularity, iterative algorithms that augment the solution at each step (such as the family of greedy algorithms) may add elements that have large marginal value but *blind* the algorithm from picking up more "good" elements. Submodularity mutes this problem by guaranteeing that for any sets S and A, if $f(A) > f(S)$ then there exists an element $e \in A \backslash S$ such that, $f(e|S) \geq \frac{f(A)-f(S)}{|A|}$. In general, the absence of submodularity is severely limiting (see Theorem 1.1). However, for functions with submodular order we have a lifeline. Consequently, all our algorithms treat the submodular order as salient and "process" elements in this order. Broadly speaking, we propose two types of algorithms. The first is inspired by greedy algorithms and the second by local search.

2.1. Threshold Based Parsing in Submodular Order

Consider an instance of the cardinality constrained problem for a π submodular function. Fix an optimal solution and given set S, let OPT_S denote the subset of optimal products located to the right of S in the submodular order ($l_\pi(\text{OPT}_S) > r_\pi(S)$). Suppose we have an algorithm that maintains a feasible set at every iteration and let S_j denote this set at iteration j. Using weak submodular order property (and monotonicity), we can show that there exists e to the right of S_j such that

$$f(e \mid S_j) \geq \frac{f(S_j \cup \text{OPT}_{S_j}) - f(S_j)}{k}.$$

Now, if we greedily add an element to S_j, we may add an element very far down in the order. This may substantially shrink the set $\text{OPT}_{S_{j+1}}$ in the next iteration. To address this issue we consider a *paced* approach where we add the first element e to the right of S_j such that

$$f(e \mid S_j) \geq \tau(N, f, k, S_j),$$

where $\tau(N, f, k, S_j)$ is a *threshold* that can depend on the instance N, f, k as well as set S_j. In other words, we take marginal values into consideration by focusing only on elements that have sufficiently large value (but not the maximum value). From this filtered set of candidates we add the closest element to the right of S_j. Thus, minimizing the shrinkage in $\text{OPT}_{S_{j+1}}$.

The simplest possible threshold is a constant independent of the iteration. Quite surprisingly, for cardinality constraint choosing a constant threshold leads to the best possible guarantee for maximizing (strong and weak) submodular order functions. Letting OPT denote the optimal value, any threshold close to $\frac{\text{OPT}}{2k}$ is a good choice. While we do not know OPT a priori, trying a few values on a geometric grid gets us arbitrarily close.

ALGORITHM 1: $\frac{1}{2}$ for Cardinality

Input: Cardinality k, error $\epsilon \in (0, 1)$;
Initialize $\tau = \frac{1}{k} \max_{e \in N} f(\{e\})$;
for $i \in \{1, 2, \cdots, \lceil \log_{1+\epsilon} k \rceil\}$ **do**
  $S_i = \text{Threshold Add}\left(k, (1+\epsilon)^{i-1}\tau\right)$;
end
Output: Best of $\{S_1, S_2, \cdots, S_{\lceil \log_{1+\epsilon} k \rceil}\}$

ALGORITHM 2: Threshold Add(k, τ)

Input: Cardinality k, threshold τ, submodular ordered N;
Initialize $S = \emptyset$;
for $i \in \{1, 2, \cdots, n\}$ *and* $|S| < k$ **do**
  if $f(i|S) \geq \tau$ **then**
    $S \to S \cup \{i\}$;
  end
end
Output: Set S of size at most k;

Remarks: The idea of iteratively adding elements with marginal value above a threshold has a rich history in both submodular optimization and assortment optimization. Badanidiyuru & Vondrák (2014) gave a fast $(1 - 1/e) - \epsilon$ approximation for constrained maximization of submodular functions using adaptive thresholds. Badanidiyuru et al. (2014) proposed an adaptive threshold algorithm for cardinality constraint in the streaming model. In the assortment optimization literature, Désir et al. (2020) introduce a constant threshold algorithm for cardinality and budget constrained assortment optimization in the Markov model.

Generalizing to budget constraint: We discuss the high level ideas here and include the algorithms for budget constraint in Appendix A.1. For budget constraints, a natural generalization of Algorithm 1 is to consider a threshold on the "bang-per-buck" i.e., *marginal value per unit budget*. Formally, we filter out elements e such that $\frac{f(e|S_j)}{b_e} < \tau(N, f, B)$. The resulting algorithm (Algorithm 4 in Appendix A.1) is $(1 - \epsilon)/3$ approximate, matching the guarantee stated in Theorem 1.2 (ii)(a). A standard technique in submodular maximization (and beyond) for obtaining best possible guarantees under budget constraint is *partial enumeration*. For example, Sviridenko (2004) considers all possible sets of size 3 as a starting point for a greedy algorithm that picks an item with best bang-per-buck at each step thereafter. Starting with an initial set X is equivalent to changing the objective function to $f(\cdot \mid X)$. In case

Appendix B.1, part (ii) (a and b) for budget constraint in Appendix B.2, and part (iii) for matroid constraint in Appendix B.3. The proof of Theorem 1.3 is included in Appendix B.4.

of submodular order functions, $f(\cdot \mid X)$ will, in general, not have a weak submodular order even for a small set X. To preserve submodular order we may restrict enumeration to starting sets that are concentrated early in the order. However, this limited enumeration appears to be ineffective. Instead, we propose an algorithm that starts with an empty set and parses elements in submodular order but enumerates over a small set of high budget elements which get "special attention". The final algorithm is included in Appendix A.1 (see Algorithm 6). Its performance guarantee is stated in Theorem 1.2 (ii)(b).

2.2. Local Search Along Submodular Order

When subject to a matroid constraint, threshold based ideas fail to achieve a strong guarantee (see Appendix A.2). To address the novel challenges posed by a matroid constraint, we introduce an algorithm inspired by local search. Given a feasible set S, a local search algorithm tries to make "small" changes to the set in order to improve function value. Often, this is done via *swap operations* where a set X of elements is added to S and a subset Y (can be empty) is removed such that $f(S \cup X \backslash Y) \geq (1+\epsilon) f(S)$, for some $\epsilon > 0$, and the new set $S \cup X \backslash Y$ is feasible[7]. The algorithm terminates at local maxima and for submodular functions the local maxima are within a constant factor of optimal (Feige et al., 2011). As illustrated in Example 1, functions with submodular order have poor local maxima. Therefore, we perform a *directed* local search where we consider elements for swap operations one-by-one in the submodular order. Consider the following algorithm based on this idea,

Parse elements in submodular order and add j to current set S if there exists an element i (could be $\emptyset$) such that $S' := S + j - i \in \mathcal{I}$ and $f(S') \geq (1+\epsilon) f(S)$.

At first, one may prefer to set a small value of ϵ. This would entail slower convergence but is expected to yield stronger approximation guarantee. However, unlike standard local search, the algorithm above parses each element exactly once. It turns out that unless ϵ is large enough, a single pass over the elements may end with a poor chain of swaps $i_1 \to i_2 \cdots \to i_k$. This is demonstrated in the next example.

Example: Consider the independent set $S = \{1, \cdots, m\}$ with $f(S) = 1$ at iteration $m+1$. Let $\{m, m+1\}$ form a circuit and suppose $m+1$ meets the criteria for a swap when $1/\epsilon = O(m^2)$ (quite small). Therefore, we have $S = [m-1] \cup \{m+1\}$ after iteration $m+1$ and let $f(S) = 1+\epsilon$. In subsequent iteration, suppose $m+2$ makes a circuit with element $2 \in S$ and let $f(S \cup \{m+2\} \backslash \{2\}) = f(S)$. Thus, $m+2$ does not meet the criteria for a swap. However, $f(\{m, m+2\}) = 2$. In the next iteration, we swap out $m+1$ in favor of $m+3$ and for the resulting set let $f(S) = 1+2\epsilon$. Next, we find that element $m+4$ does not meet the criteria for a swap but $f(\{m, m+2, m+4\}) = 3$. Inductively, after $2m$ such iterations we have an instance such that the local search solution value is $1 + m\epsilon \approx 1 + 1/m$, while the optimal value is m.

At a high level, we prevent such poor chains of swaps from occurring by constantly adjusting the swap criteria so that the k-th element in a chain $i_1 \to \cdots \to i_k$ of swaps has marginal value greater than equal to the *sum of the marginal values of all previous elements* in the chain.

ALGORITHM 3: $\frac{1}{4}$ for Matroid Constraint

Input: Independence system $\mathcal{I}$, submodular ordered N;
Initialize $S, R = \emptyset$ and values $v_j = 0$ for all $j \in [n]$;
for $j \in \{1, 2, \cdots, n\}$ **do**
  if $S + j \in \mathcal{I}$ **then** initialize $v_j = f(j \mid S \cup R)$ and update $S \to S + \{j\}$;
  else
    Let C denote a circuit in $S + j$ and compute $i^* = \arg\min_{i \in C \backslash \{j\}} v_i$ and $v_C = v_{i^*}$;
    if $f(j | S \cup R) > v_C$ **then**
      $v_j \to v_C + f(j \mid S \cup R)$;
      $S \to S + \{j\} - \{i^*\}$ and $R \to R + \{i^*\}$;
    end
  end
end
Output: Independent set S

3. Applications of Submodular Order

We show applications of submodular order function maximization to assortment optimization in Sections 3.1 – 3.2, and to the streaming maximization of submodular functions in Section 3.3.

3.1. Constrained Assortment Optimization

We start by providing background on assortment optimization and discussing related work. Recall that in assortment optimization, we assume that a choice model $\phi : N \times 2^N \to [0,1]$ is given and we are interested in the problem of finding the revenue optimizing assortment subject to some constraint. Given assortment S, the probability that customer chooses product $i \in S$ is given by $\phi(i, S)$. Customer may choose an *outside option* (not in S) with probability $1 - \sum_{i \in S} \phi(i, S)$. Given (fixed) prices

[7] Other local search techniques exist. For instance, oblivious local search algorithms (see (Filmus & Ward, 2012)) may perform swaps to increase the value of a proxy function instead of the objective.

$(r_i)_{i\in N}$, the expected revenue is,

$$R_\phi(S) = \sum_{i\in S} r_i \phi(i,S).$$

Out of the many choice models that have been introduced in the literature, we are interested in the following well studied choice models.

Multinomial Logit Choice Model (MNL) (Bradley & Terry, 1952; Luce, 1959; McFadden, 1973; Plackett, 1975): This model is defined by parameters $v_i \geq 0$ for $i \in N$. $v_0 \geq 0$ denotes the parameter for the outside option. The probability that a customer chooses product i from assortment S is proportional to v_i. Formally,

$$\phi(i,S) := \frac{v_i}{v_0 + \sum_{e\in S} v_e}.$$

Previous results for MNL optimization: The assortment optimization problem under MNL choice is very well studied. Notice that the objective in this problem is non-monotone (and not submodular). Talluri & Van Ryzin (2004) showed that the unconstrained problem can be solved optimally and the optimal solution includes all products above a price threshold. Rusmevichientong et al. (2010) gave a polynomial time algorithm for the cardinality constrained problem. Davis et al. (2013) and Avadhanula et al. (2016) showed that the MNL optimization problem can solved optimally in polynomial time under totally unimodular constraints (TUM). Désir & Goyal (2014) showed that the budget constrained version is NP hard and they gave a FPTAS for the problem.

Mixture of MNL with Customization (El Housni & Topaloglu, 2022): Consider a population given by m types of customers. Each customer chooses according to a MNL choice model that depends on their type. The choice model of the population is described by a Mixture of MNLs – MMNL choice model (McFadden & Train, 2000). Customer type is revealed on arrival and we offer a *customized* assortment based on the type. An offered assortment can be any subset of the products that we keep in our selection and suppose that we can keep at most k different products in the selection. Suppose that a random customer is type $j \in [m]$ with probability α_j. Thus, we would like to select at most k products to maximize,

$$\max_{S,|S|\leq k} \sum_{j\in[m]} \alpha_j \max_{X\subseteq S} R_{\phi_j}(X),$$

here ϕ_j is the MNL choice model for customer type j. A $\frac{1}{m}$ approximation for this problem can be obtained simply by solving a cardinality constrained MNL assortment problem for each type separately and picking the best of these solutions. El Housni & Topaloglu (2022) gave a substantially stronger $\frac{\Omega(1)}{\log m}$ approximation for this problem and showed that it is NP hard to approximate the optimal assortment better than $\left(1 - \frac{1}{e} + \epsilon\right)$. For constant m, they gave a FPTAS. The unconstrained version of this problem (where S can be any subset of N) can be solved simply by taking the union of optimal unconstrained assortments for each type of MNL.

Markov Choice Model (Blanchet et al., 2016): In this model, the customer choice process is described by a discrete markov chain on the state space of products (and the outside option). Given an assortment S, the customer starts at product $i \in N$ with probability λ_i. A customer at a product $i \notin S$, transitions to product $j \in N$ with probability ρ_{ij} (independent of their actions prior to i). The random process terminates when the customer reaches a product in $S \cup \{\emptyset\}$, which is their final choice. This model generalizes a wide array of choice models (see Blanchet et al. (2016)).

Previous results for Markov optimization: Blanchet et al. (2016) introduced this model and gave a polynomial time algorithm for unconstrained optimization. Désir et al. (2020) consider the problem under cardinality and budget constraints. They show that the problem is APX hard under cardinality constraint and inapproximable under TUM constraints (sharp contrast with MNL). They obtain a $0.5 - \epsilon$ approximation for the cardinality constrained problem and a $1/3 - \epsilon$ approximation for the budget constrained version.

MNL, MMNL, and Markov choice models have received significant attention in the assortment optimization literature and even the simpler of these models i.e., MNL, is very useful for modeling choice behavior in practice (for instance, see Feldman et al. (2021)).

3.2. New Results for Assortment Optimization

An obvious obstacle in the application of our results is the absence of monotonicity in the revenue objective. We solve this issue by redefining the objective as follows,

$$f_\phi(S) = \max_{X\subseteq S} R_\phi(X).$$

In Appendix D, we show that the function f_ϕ is monotone and subadditive under the following (mild) condition,

Substitutability: $\phi(i, S\cup\{j\}) \leq \phi(i,S)$, $\forall i \in S, j \notin S$.

MNL, MMNL, and Markov model, all satisfy this condition. In fact, most choice models are substitutable (Kök et al., 2008; Golrezaei et al., 2014). In order to evaluate $f_\phi(S)$, we need to solve an unconstrained assortment optimization problem on the reduced ground set S. Recall that previous works give polynomial time algorithms for unconstrained optimization under the choice models of interest to us. Therefore, we have efficient implementation of the value oracle for f_ϕ. More generally, if a choice model ϕ is

such that f_ϕ has a submodular order but the unconstrained assortment problem only admits a $1-\delta$ approximation (instead of an efficient optimal algorithm), then the approximation guarantees are reduced by a multiplicative factor of $\left(1 - O(\frac{n\delta}{1-\delta})\right)$. This factor is negligible for small δ and this can be useful in case the unconstrained assortment problem admits a FPTAS.

3.2.1. Results for MNL and MMNL with Customization

When ϕ is MNL, we show that sorting products in descending order of price (breaking ties arbitrarily) is a (strong) submodular order for f_ϕ. Consequently, all our results for constrained optimization of submodular order apply directly. More importantly, using the fact that submodular order functions are closed under addition (see Remark 3.1), we immediately obtain the first constant factor approximation for the problem of assortment optimization with customization under MMNL choice.

Remark 3.1. Given p monotone subadditive functions $f_1, \cdots, f_m$ with the same submodular order π and non-negative real values $\alpha_1, \cdots, \alpha_m$, the function $\sum_{j\in[m]} \alpha_j f_j$ is also monotone subadditive with submodular order π.

Theorem 3.2. *The problem of assortment optimization with customization under MMNL choice is an instance of cardinality constrained submodular order maximization.*

As a direct consequence of Theorem 3.2, we have a $0.5-\epsilon$ approximation for any m. Table 2 provides approximation guarantees under more general constraint on the selection of products (budget and matroid). We establish submodular order property for MNL model and prove Theorem 3.2 in Appendix D.1. In fact, as a corollary of the result for matroid constraint, we obtain an approximation result for the problem of joint *pricing* and customization under MMNL choice (see Appendix D.2 for more details).

3.2.2. Results for Markov Model and Beyond

The order given by descending prices is not a submodular order for every choice model. The following example demonstrates this for the Markov model.

Example: Consider a ground set of 4 items indexed $i \in [4]$ in decreasing order of prices $r_1 = 8, r_2 = 4, r_3 = 4$ and $r_4 = 2$. Customer chooses in a markovian fashion starting at item 2 with probability 1. If item 2 is not available, the customer transits to item j with probability $1/3$ for every $j \in \{1,3,4\}$. If j is also unavailable then the customer departs with probability 1. Consider the sets $B = \{1,2\}$ and $A = \{1,2,3\}$. We have $f(A) = f(B) = 4$. However, if we add item 4 to these sets we get, $f(4 \mid A) = R(\{1,3,4\}) - 4 = 2/3$, whereas, $f(4 \mid B) = R(2) - R(2) = 0$.

Table 2. Results for constrained assortment optimization. New results are stated in bold. Custom MMNL stands for MMNL with customization. m denotes the number of MNL models in the mixture and d is the matroid rank.

	Custom MMNL	Markov
Cardinality	$\frac{\Omega(1)}{\log m} \to \mathbf{0.5-\epsilon}$	$0.5-\epsilon$
Budget	$\frac{\Omega(1)}{\log m} \to \mathbf{0.5-\epsilon}$	$\frac{1}{3}-\epsilon \to \mathbf{0.5-\epsilon}$
Matroid	$\frac{1}{m} \to \mathbf{0.25}$	$\frac{1}{d} \to \mathbf{0.25}$

It is not obvious (to us) if there is an alternative submodular order for Markov choice model. We give a procedure that extracts a *partial* submodular order for any given Markov choice model and show that all our algorithms for submodular order functions apply to assortment optimization in this model without any loss in guarantee. In fact, we show this more generally for any choice model that has the following structure.

Compatible Choice Models: Given a substitutable choice model ϕ, let S be an optimal unconstrained assortment on the ground set N. We say that ϕ is *compatible* if,

$$R_\phi(A \mid C) \geq 0 \qquad \forall A \subseteq S,\, C \subseteq N \tag{2}$$

$$R_\phi(C \mid A) \leq R_\phi(C \mid B) \qquad \forall B \subseteq A \subseteq S,\, C \subseteq N \tag{3}$$

Compatibility is a structural property of optimal unconstrained assortments that gives sufficient conditions (in the absence of a submodular order) for translating algorithms (with guarantees) from submodular order maximization.

Theorem 3.3. *The family of Markov choice models is compatible. For any compatible choice model ϕ that admits a polynomial time algorithm for finding optimal unconstrained assortments, we have an algorithm that matches the guarantee obtained by algorithms for submodular order functions under cardinality, budget, and matroid constraint (see table 2).*

The algorithm (Algorithm 8) is included in Appendix C and it builds on the algorithms for submodular order functions. We prove Theorem 3.3 in Appendix D.3.

Summary of application to assortment optimization: The framework of submodular order functions provides a new algorithmic tool for constrained assortment optimization. We summarize the key steps to check if this tool can be applied to obtain efficient approximations for any given choice model.

(i) Is the unconstrained assortment problem efficiently solvable? If not, is there a FPTAS?
(ii) Is there a (strong or weak) submodular order? Specifically, is the descending order of revenues a submodular order?
(iii) If the unconstrained assortment problem is efficiently solvable but a submodular order is not evident (or does not exist), is the choice model compatible?

3.3. Application to Streaming Submodular Maximization

The streaming model: Consider a setting where the ground set N is ordered in some arbitrary manner, say $\{1, 2, \cdots, n\}$. We have a monotone submodular objective f that we want to maximize subject to some constraint. Due to the large size of N, elements can only be accessed sequentially in order i.e., to access element j we need to parse the data stream of elements from 1 to j. We also have a small working memory. Thus, we seek an algorithm that guarantees a good solution with very few passes (ideally, just one pass) over the data stream and requires very low memory (ideally, at most $O(k)$, where k is the size of the optimal solution). This setting has a variety of applications in processing and summarizing massive data sets (Badanidiyuru et al., 2014). Note that standard algorithms for submodular maximization, such as the greedy algorithm for cardinality constraint, make $\Theta(kn)$ passes over the data stream given small working memory.

Previous work in streaming model: Chakrabarti & Kale (2015) gave a $\frac{1}{4p}$ approximation for maximization subject to intersection of p matroids. (Badanidiyuru et al., 2014) gave a different algorithm with improved guarantee of $0.5 - \epsilon$ for cardinality constraint. Feldman et al. (2020) show that every streaming algorithm with guarantee better than $0.5 + \epsilon$ requires $O(\epsilon n/k^3)$ memory. Huang & Kakimura (2021) gave a $0.4 - \epsilon$ streaming algorithm for budget constrained optimization. Recently, Feldman et al. (2022) gave improved approximation algorithms for matroid constraint. Chekuri et al. (2015) showed results for very general ($p-$matchoid) constraints as well as non-monotone submodular functions. A more comprehensive review of related work can be found in Huang & Kakimura (2021); Feldman et al. (2022).

Connection to submodular order maximization: Consider a function f that is π-submodular order. Given the salience of order π, many of our algorithms for optimizing f are intentionally designed so that they parse elements in the order given by π. In fact, these algorithms can be efficiently implemented such that they parse the ground set exactly once and require very little memory (see Algorithm 1, Algorithm 4, and Algorithm 3). Now, consider an instance of streaming submodular maximization. An α approximate algorithm for submodular order functions that parses the ground set only once (in submodular order) and requires low memory is, by default, an α approximation algorithm for streaming submodular maximization. Consequently, we recover the best known guarantees for streaming submodular maximization in the cardinality and matroid case as a corollary of our results. It is worth noting that Algorithm 6 (given in Appendix A.1), which is $0.5 - \epsilon$ approximate for budget constraint, requires working memory polynomial in n.

While this connection is obvious in hindsight, it raises intriguing new questions. For example, the upper bound of 0.5 for submodular order functions (Theorem 1.3) is a consequence of their milder structure. In contrast, the upper bound of 0.5 for streaming maximization (Feldman et al., 2020) arises out of restrictions on memory. From a purely structural viewpoint, submodularity permits a stronger $(1 - 1/e)$ guarantee. *Is there a precise connection between less structure and memory limitations?*

4. Conclusion

We introduced notions of weak and strong submodular order functions and explored the landscape of constrained maximization problems under this structure. We showed that in the value oracle model, no polynomial algorithm can achieve a guarantee better than 0.5 for maximizing function value subject to a cardinality constraint. We proposed algorithms that achieve this best possible guarantee under cardinality and more generally, budget constraint. We also gave a 0.25 approximation for the problem under matroid constraint. Applying these results, we obtained improved (and first constant factor) guarantees, with unified algorithms, for several constrained assortment optimization problems. We also observed an intriguing algorithmic connection of submodular order maximization with the problem of streaming submodular optimization. The connection raises some interesting questions about the interplay between structure of set functions and computational considerations such as limited memory.

Directions for Further Work

- What is the best possible approximation guarantee under matroid constraint?
- **Multiple submodular orders:** Consider an order π and its reverse order π_r. Does a function with submodular order along both π and π_r admit better approximations? Note that the upper bound of 0.5 in Theorem 1.3 applies to functions with multiple submodular orders but not to functions where an order and its reverse are both submodular orders.
- **Unknown submodular order:** Given a function f and the knowledge that it has a submodular order, can we get interesting guarantees for optimization without knowing the order? Are there natural conditions under which the order could be computed using only value queries? What is the computational complexity of verifying that a given order is a submodular order or certifying the inexistence of (weak) submodular order? Note that testability problems are hard and not completely understood even for classic submodular functions (Seshadhri & Vondrák, 2014).

Recall that knowing the submodular order is not necessary for obtaining algorithmic guarantees. In particular, our algorithm for Markov choice model develops a partial (submodular) order by repeatedly solving an unconstrained version of the problem on smaller and smaller ground sets. Generalizing the scope of such an idea idea is another interesting direction for future work. It is worth noting that the impossibility result given in Theorem 1.1 only applies to functions that do not have any (strong or weak) submodular order.

- Are there other applications where objective is not submodular but has submodular order?

Acknowledgements

The author thanks Ali Aouad, Omar El-Housni, Vineet Goyal, and Danny Segev for helpful discussions and comments on early drafts of this work.

References

Agrawal, S., Avadhanula, V., Goyal, V., and Zeevi, A. Mnl-bandit: A dynamic learning approach to assortment selection. *Operations Research*, 67(5):1453–1485, 2019.

Aouad, A., Farias, V., Levi, R., and Segev, D. The approximability of assortment optimization under ranking preferences. *Operations Research*, 66(6):1661–1669, 2018a.

Aouad, A., Levi, R., and Segev, D. Greedy-like algorithms for dynamic assortment planning under multinomial logit preferences. *Operations Research*, 66(5):1321–1345, 2018b.

Avadhanula, V., Bhandari, J., Goyal, V., and Zeevi, A. On the tightness of an lp relaxation for rational optimization and its applications. *Operations Research Letters*, 44(5): 612–617, 2016.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1497–1514. SIAM, 2014.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 671–680, 2014.

Blanchet, J., Gallego, G., and Goyal, V. A markov chain approximation to choice modeling. *Operations Research*, 64(4):886–905, 2016.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324, 1952.

Calinescu, G., Chekuri, C., Pal, M., and Vondrák, J. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6): 1740–1766, 2011.

Chakrabarti, A. and Kale, S. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1):225–247, 2015.

Chekuri, C., Gupta, S., and Quanrud, K. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming*, pp. 318–330. Springer, 2015.

Das, A. and Kempe, D. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 1057–1064, 2011.

Davis, J., Gallego, G., and Topaloglu, H. Assortment planning under the multinomial logit model with totally unimodular constraint structures, 2013. Technical Report.

Désir, A. and Goyal, V. Near-optimal algorithms for capacity constrained assortment optimization, 2014. Working paper, available online as SSRN report 2543309.

Désir, A., Goyal, V., Segev, D., and Ye, C. Constrained assortment optimization under the markov chain–based choice model. *Management Science*, 66(2):698–721, 2020.

Dobzinski, S., Nisan, N., and Schapira, M. Approximation algorithms for combinatorial auctions with complement-free bidders. *Mathematics of Operations Research*, 35(1): 1–13, 2010.

El Housni, O. and Topaloglu, H. Joint assortment optimization and customization under a mixture of multinomial logit models: On the value of personalized assortments. *Operations Research*, 2022.

Feige, U., Mirrokni, V. S., and Vondrak, J. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Feldman, J., Zhang, D., Liu, X., and Zhang, N. Customer choice models versus machine learning: Finding optimal product displays on alibaba. *Operations Research (Forthcoming)*, 2021.

Feldman, M., Naor, J., and Schwartz, R. A unified continuous greedy algorithm for submodular maximization. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 570–579. IEEE, 2011.

Feldman, M., Norouzi-Fard, A., Svensson, O., and Zenklusen, R. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1363–1374, 2020.

Feldman, M., Liu, P., Norouzi-Fard, A., Svensson, O., and Zenklusen, R. Streaming Submodular Maximization Under Matroid Constraints. In Bojańczyk, M., Merelli, E., and Woodruff, D. P. (eds.), *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 59:1–59:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-235-8.

Filmus, Y. and Ward, J. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pp. 659–668. IEEE, 2012.

Golrezaei, N., Nazerzadeh, H., and Rusmevichientong, P. Real-time optimization of personalized assortments. *Management Science*, 60(6):1532–1551, 2014.

Huang, C.-C. and Kakimura, N. Improved streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 83(3):879–902, 2021.

Kök, A. G., Fisher, M. L., and Vaidyanathan, R. Assortment planning: Review of literature and industry practice. *Retail supply chain management*, pp. 99–153, 2008.

Krause, A., McMahan, H. B., Guestrin, C., and Gupta, A. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(12), 2008a.

Krause, A., Singh, A., and Guestrin, C. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2), 2008b.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, 2007.

Luce, R. D. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

McFadden, D. Conditional logit analysis of qualitative choice behavior. *in P. Zarembka, ed., Frontiers in Econometrics*, 1973.

McFadden, D. and Train, K. Mixed MNL models for discrete response. *Journal of Applied Econometrics*, 15(5): 447–470, 2000.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Panconesi, A. and Srinivasan, A. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

Plackett, R. L. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24 (2):193–202, 1975.

Rusmevichientong, P., Shen, Z.-J. M., and Shmoys, D. B. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research*, 58(6):1666–1680, 2010.

Seshadhri, C. and Vondrák, J. Is submodularity testable? *Algorithmica*, 69(1):1–25, 2014.

Sviridenko, M. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Talluri, K. and Van Ryzin, G. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pp. 1954–1963. PMLR, 2015.

A. Further Discussion on Algorithms for Submodular Order Functions

A.1. Algorithms for Budget Constraint

Algorithm 4 presents a natural generalization of Algorithm 5 (for cardinality constraint) with a threshold on the "bang-per-buck" i.e., *marginal value per unit budget*. Formally, we filter out elements e such that $\frac{f(e|S_j)}{b_e} < \tau(N, f, B)$. We show a guarantee of $1/3$ for Algorithm 4, short of the upper bound of 0.5 implied by Theorem 1.3.

ALGORITHM 4: $\frac{1}{3}$ for Budget

Input: Budget B, error $\epsilon \in (0, 1)$;

Initialize $\tau = \frac{1}{B} \max_{e \in N} f(\{e\})$;
for $i \in \{1, 2, \cdots, \lceil \log_{1+\epsilon} n \rceil\}$ **do**
  $S_i =$ B-Th. Add $\big(B, (1+\epsilon)^{i-1}\tau\big)$;
end

Output: Best of all *singletons* and sets $\{S_1, S_2, \cdots, S_{\lceil \log_{1+\epsilon} k \rceil}\}$

ALGORITHM 5: B-Threshold Add(B, τ)

Input: Budget B, threshold τ, submodular ordered N;

Initialize $S = \emptyset$;
for $i \in \{1, 2, \cdots, n\}$ *and* $b(S) < B$ **do**
  if $b_i < B - b(S)$ *and* $\frac{f(i|S)}{b_i} \geq \tau$ **then** $S \to S \cup \{i\}$;
end

Output: Feasible set S;

ALGORITHM 6: 0.5 for Budget Constraint

Input: Budget B, error $\epsilon \in (0, 1)$, submodular ordered N;

Initialize collection $E = \big\{X \subseteq N \,\big|\, b(X) \leq B,\ |X| \leq 1/\epsilon\big\}$;
for $X \in E$ **do**
  Filter ground set $N \to N \backslash \{e \mid b_e > \min_{i \in X} b_i\}$;
  Initialize $\tau = \frac{1}{B} \max_{e \in N} f(\{e\})$;
  for $i \in \{1, 2, \cdots, \lceil \log_{1+\epsilon} |N| \rceil\}$ **do**
    Initialize $S_i = \emptyset$;
    for $j \in \{1, 2, \cdots, |N|\}$ *and* $b(S_i) < B$ **do**
      if $\frac{f(j|S_i)}{b_i} \geq (1+\epsilon)^{i-1}\tau$ **then**
        if $b_j + b(S_i) \leq B$ **then**
          $S_i \to S_i \cup \{j\}$
        else
          Update $S_i \to$ **Final Add** $(B, \epsilon, S_i \cup \{j\})$;
          Goto next i;
        end
      end
    end
  end
  Let $S(X) =$ Best of all feasible sets S_i;
end

Output: Best of all sets $\{S(X) \mid X \in E\}$

A standard technique in submodular maximization (and beyond) for obtaining best possible guarantees under budget constraint is *partial enumeration*. For example, Sviridenko (Sviridenko, 2004) considers all possible sets of size 3 as a starting point for a greedy algorithm that picks an item with best bang-per-buck at each step thereafter. Starting with an

ALGORITHM 7: Final Add

Input: Budget B, ϵ, set S;
while $b(S) > B$ **do**
  Remove an element i with budget $b_i < \epsilon B$ from S;
  End loop if no such element exists;
end
Output: S;

initial set X is equivalent to changing the objective function to $f(\cdot \mid X)$. In case of submodular order functions $f(\cdot \mid X)$ will, in general, not have a weak submodular order even for a small set X. To preserve submodular order we may restrict enumeration to starting sets that are concentrated early in the order. However, this limited enumeration appears to be ineffective. Instead, we propose an algorithm that starts with an empty set and parses elements in submodular order but enumerates over a small set of high budget elements which get "special attention" via the Final Add subroutine. It is worth noting that unlike Algorithm 4, which only adds elements, Algorithm 6 may discard elements that were added in previous iterations via the Final Add subroutine.

A.2. Failure of Threshold Algorithms under Matroid Constraint

The following example illustrates why threshold based algorithms fail under matroid constraints.

Example: Consider a matroid with rank $2^n - 1$ on the ground set $\{1, \cdots, 2^{n+1} - 2\}$, indexed in submodular order. Let R denote the set $\{1, \cdots, 2^n - 1\}$ i.e., the first half of the ground set. Let P denote the other half. Partition R into ordered sets $\{R_1, \cdots, R_n\}$ such that R_1 is the set $[2^{n-1}]$ of the first 2^{n-1} elements, R_2 of the next 2^{n-2} elements, and so on. Therefore, $|R_i| = 2^{n-i}$ for every $i \in [n]$. Partition P into ordered sets $\{P_1, \cdots, P_n\}$ so that P_i is the set $\{2^n - 1 + 2^{i-1}, 2^n - 2 + 2^i\}$ of 2^{i-1} elements. Observe that $|R_i| = |P_{n-i+1}|$. We define independent sets in the matroid so that the set $R_i \cup \left(\cup_{j=n-i+1}^{n} P_j\right)$ is independent for every $i \in [n]$. Therefore, P is independent and so is the set R_i (and P_i) for every i. Further, let the rank of set $R_i \cup R_{i+j}$ equal $|R_i|$ for every $j \geq 1$. Finally, define a *modular* function f so that P is optimal and $f(P) = 2^n - 1$. Let $f(P_i) = f(P)/n$ and $f(e) = f(P_i)/|P_i|$, for every $i \in [n]$ and $e \in P_i$. Similarly, let $f(R_i) = f(P_{n-i+1}) = f(P)/n$ and $f(e) = f(R_i)/|R_i|$, for every $i \in [n]$ and $e \in R_i$. Now, for any given threshold τ, the algorithm that parses elements in order and selects every element with marginal value exceeding τ (while maintaining independence) will pick a set with value exactly $2f(P)/n$ on this instance (translating to a factor $2/\log n$ for ground set of size n).

B. Main Results for Submodular Order Maximization

In this section, we prove our main results for maximizing functions with (known) submodular order. For submodular functions, given a set A and partition $\{O, E\}$ of A we have,

$$f(A) = f(E) + f(O|E) \;\leq\; f(E) + \sum_{i \in O} f(i \mid E).$$

The inequality above is at the heart of proving tight guarantees for maximizing monotone submodular functions. For functions with submodular order this property need not hold unless all elements of O are located entirely to the right of E in the submodular order. In the following we show stronger upper bounds that play a crucial role in the analysis of all our algorithms. First, we define the notion of interleaved partitions.

Interleaved partitions: Given a set A and an order π over elements, an interleaved partition

$$\{O_1, E_1, O_2, \cdots, O_m, E_m\}$$

of A is given by sets $\{O_\ell\}_{\ell \in [m]}$ and $\{E_\ell\}_{\ell \in [m]}$ that alternate and never cross each other in the order π. Formally, for every $\ell \geq 1$ we have $r_\pi(O_\ell) < l_\pi(E_\ell) \leq r_\pi(E_\ell) < r_\pi(O_{\ell+1})$. To assist the reader we note that letters O and E signify odd and even numbered sets in the partition. As noted previously, for functions with submodular order the upper bound $f(A) \leq f\left(\cup_{\ell \in [m]} E_\ell\right) + f\left(\cup_{\ell \in [m]} O_\ell \mid \cup_{\ell \in [m]} E_\ell\right)$ does not always hold. Instead, we establish a family of (incomparable) upper bounds parameterized by permutation $\sigma : [m] \to [m]$. The flexibility provided by permutation σ is particularly helpful in analyzing Algorithm 6 and Algorithm 3, where elements may be swapped out.

The following set unions will be used extensively in the lemmas that follow. Given permutation σ, let

$$E(j) = \cup_{\ell \le j} E_\ell \text{ and } O_\sigma(j) = \cup_{\ell \,|\, \sigma(\ell) \ge \sigma(j)} O_\ell,$$

with $E(0) := \emptyset$ and $O_\sigma(m+1) := \emptyset$. When $\sigma(\ell) = \ell$, $\forall \ell \in [m]$, we use the shorthand $O(j) = \cup_{\ell \ge j} O_\ell$. In this notation, $E(m) = \cup_{\ell \in [m]} E_\ell$, $O(1) = \cup_{\ell \in [m]} O_\ell$. Thus, $A = O(1) \cup E(m)$. Next, for every $j \in [m]$ define

$$L_\sigma(j) = \bigcup_{\ell \,|\, \ell < j,\, O_\ell \subseteq O_\sigma(j)} O_\ell,$$

which is the union over sets O_ℓ that are to the left of O_j in submodular order and to the right of O_j in permutation σ. Notice that when $\sigma(\ell) = \ell$, $\forall \ell \in [m]$, we have $L_{j,\sigma} = \emptyset$ for every $j \in [m]$.

Lemma B.1. *Given monotone subadditive function f with weak submodular order π, set A with interleaved partition $\{O_\ell, E_\ell\}_{\ell=1}^m$, and a permutation $\sigma : [m] \to [m]$, we have*

$$f(A) \le f\left(E(m)\right) + \sum_{\ell \in [m]} f\left(O_\ell \mid L_\sigma(\ell) \cup E(\ell-1)\right).$$

Remark: The following corollaries of this general bound suffice for all our analyses. In the first corollary we choose σ to be identity. In the second, we let σ be the reverse order permutation.

Corollary B.2. *Given monotone subadditive function f with weak submodular order π, set A with interleaved partition $\{O_\ell, E_\ell\}_{\ell=1}^m$, and the permutation $\sigma(\ell) = \ell$ for every $\ell \in [m]$, we have*

$$f(A) \le f\left(E(m)\right) + \sum_{\ell \in [m]} f\left(O_\ell \mid E(\ell-1)\right).$$

Corollary B.3. *Consider a monotone subadditive function f with weak submodular order π and set A with interleaved partition $\{O_\ell, E_\ell\}_{\ell=1}^m$. Let $L_\ell = \{e \in A \mid \pi(e) < l_\pi(E_\ell)\}$. Then,*

(i) $f(A) \le f\left(E(m)\right) + \sum_{\ell \in [m]} f\left(O_\ell \mid E(\ell-1) \cup O(1) \backslash O(\ell)\right)$,

(ii) $\sum_{\ell \in [m]} f\left(E_\ell \mid L_\ell\right) \le f\left(E(m)\right)$.

Proof. Applying Lemma B.1 with $\sigma[\ell] = m - \ell + 1$ for $\ell \in [m]$, we have

$$f(A) \quad \le \quad f\left(E(m)\right) + \sum_{\ell \in [m]} f\left(O_\ell \mid E(\ell-1) \cup O(1) \backslash O(\ell)\right).$$

To complete the proof we use the decomposition,

$$f(A) = \sum_\ell \left[f\left(E_\ell \mid L_\ell\right) + f\left(O_\ell \mid E(\ell-1) \cup O(1) \backslash O(\ell)\right) \right].$$

□

Proof of Lemma B.1. The following inequalities are crucial for the proof,

$$f\left(O_\sigma(\ell) \cup E(m)\right) \le f\left(O_\ell \mid L_\sigma(\ell) \cup E(\ell-1)\right) + f\left(O_\sigma(\ell) \cup E(m) \backslash O_\ell\right) \quad \forall\, \ell \in [m]. \tag{4}$$

Before we establish (4), observe that by summing these inequalities for $\ell \in [m]$ we get

$$\begin{aligned}
\sum_{\ell \in [m]} f\left(O_\sigma(\ell) \cup E(m)\right) - \sum_{\ell \in [m]} f\left(O_\sigma(\ell) \cup E(m) \backslash O_\ell\right) &\le \sum_{\ell \in [m]} f\left(O_\ell \mid L_\sigma(\ell) \cup E(\ell-1)\right), \\
f\left(O(1) \cup E(m)\right) - f\left(E(m)\right) &\le \sum_{\ell \in [m]} f\left(O_\ell \mid L_\sigma(\ell) \cup E(\ell-1)\right), \\
f(A) - f\left(E(m)\right) &\le \sum_{\ell \in [m]} f\left(O_\ell \mid L_\sigma(\ell) \cup E(\ell-1)\right),
\end{aligned}$$

as desired. It remains to show (4). Consider arbitrary $\ell \in [m]$ and notice that sets $B := L_\sigma(\ell) \cup E(\ell-1)$ and $A := L_\sigma(\ell) \cup E(\ell-1) \cup O_\ell$ are π-nested. Moreover, the set $C := O_\sigma(\ell) \cup E(m) \backslash A$, lies entirely to the right of A. From weak submodular order property we have, $f(C \mid A) \leq f(C \mid B)$. Consequently,

$$\begin{aligned} f\left(O_\sigma(\ell) \cup E(m)\right) &= f(A) + f(C \mid A) \\ &\leq f(B) + f(O_\ell \mid B) + f(C \mid B) \\ &= f(O_\ell \mid B) + f(B \cup C) \\ &= f(O_\ell \mid L_\sigma(\ell) \cup E(\ell-1)) + f\left(O_\sigma(\ell) \cup E(m) \backslash O_\ell\right). \end{aligned}$$

□

B.1. Cardinality Constraint

Proof of Theorem 1.2. We start by calculating the number of queries made by Algorithm 1. The Threshold Add subroutine makes at most n queries. Algorithm 1 calls the Threshold Add subroutine $\lceil \log_{1+\epsilon} k \rceil$ times. This results in $O(n \log_{1+\epsilon} k) = O(\frac{n}{\epsilon} \log k)$ queries.

Next, we establish the approximation guarantee. Let $\tau_i = \tau(1+\epsilon)^{i-1}$. We use OPT to denote both the optimal solution and function value. Notice that for $\text{OPT} \leq 2 \max_{e \in N} f(\{e\})$, we have $f(S_{\lceil \log_{1+\epsilon} k \rceil}) \geq (1-\epsilon)\, 0.5\, \text{OPT}$. So from here on we let $\text{OPT} > 2 \max_{e \in N} f(\{e\})$. Consequently, there exists an i such that

$$(1-\epsilon)\frac{\text{OPT}}{2k} \leq \tau_i \leq \frac{\text{OPT}}{2k}.$$

We show that $f(S_i) \geq (1-\epsilon)\, 0.5\, \text{OPT}$. For convenience, we omit i from the subscript and write S_i simply as S and τ_i as τ.

Let k' denote the cardinality of set S. By definition of Threshold Add,

$$f(S) \geq k'\tau.$$

When $k' = k$ this gives us $f(S) \geq (1-\epsilon)\, 0.5\, \text{OPT}$. So let $k' < k$. From monotonicity of the function we have, $\text{OPT} \leq f(\text{OPT} \cup S)$. So consider $\text{OPT} \cup S$ and its interleaved partition

$$\{O_1, \{s_1\}, O_2, \{s_2\}, \cdots, \{s_{k'}\}, O_{k'+1}\},$$

where s_j denotes the jth element added to S. Set O_{j+1} contains all elements in OPT between s_j and s_{j+1} i.e., $\pi(s_j) < l_\pi(O_{j+1})$ and $r_\pi(O_j) < \pi(s_j)$ for every $j \in [k']$. Note that some sets may be empty. Applying Corollary B.2 on $\text{OPT} \cup S$ with $E_j = s_j$ for $j \in [k']$ we have,

$$\begin{aligned} \text{OPT} &\leq f(E(k')) + \sum_{\ell \in [k']} f(O_\ell \mid E(\ell-1)), \\ &= f(S) + \sum_{\ell \in [k']} f(O_\ell \mid E(\ell-1)). \end{aligned} \tag{5}$$

Using weak submodular order we have for every $\ell \geq 1$,

$$f(O_\ell \mid E(\ell-1)) \leq \sum_{e \in O_\ell} f(e \mid E(\ell-1)) \leq \tau |O_\ell|,$$

where the second inequality follows by definition of Threshold Add. Plugging this into (5) and using the upper bound $\tau \leq 0.5\text{OPT}/k$ we get,

$$\text{OPT} \leq f(S) + k\tau \leq f(S) + 0.5\, \text{OPT}.$$

This completes the proof. Now, suppose we have a noisy function oracle $\hat{f}$ such that, $(1-\delta)f(S) \leq \hat{f}(S) \leq (1+\delta)f(S)\ \forall S \subseteq N$. Observe that $f(e \mid S) \leq \frac{1}{1-\delta}\hat{f}(S+e) - \frac{1}{1+\delta}\hat{f}(S) \leq \hat{f}(e \mid S) + \frac{2\delta}{1-\delta} f(S+e)$. For $k' = k$, we have, $f(S) \geq \frac{1}{1+\delta}\hat{f}(S) \geq \frac{1}{1+\delta}k'\tau \geq \frac{1}{1+\delta}(1-\epsilon)0.5\, \text{OPT}$. For $k' < k$, we have,

$$f(e \mid E(\ell-1)) \leq \hat{f}(e \mid E(\ell-1)) + 2\frac{\delta}{1-\delta} f(e \cup E(\ell-1)) < \tau + 2\frac{\delta}{1-\delta}\text{OPT} \quad \forall e \in O_\ell.$$

Therefore, $f(S) \geq \left(0.5 - \frac{2k\delta}{1-\delta}\right)$ OPT. Overall, in the presence of a noisy oracle we have have approximation guarantee $\min\left\{\frac{0.5}{1+\delta}(1-\epsilon),\, 0.5 - \frac{2k\delta}{1-\delta}\right\} = (1-\epsilon)\left(1 - O(\frac{n\delta}{1-\delta})\right)\, 0.5$.

□

B.2. Budget Constraint

Proof of Theorem 1.2 (i). The query complexity of Algorithm 4 is identical to Algorithm 1. To establish the approximation guarantee, we ignore elements $i \in N$ with $b_i > B$. Also let $\sum_{i\in N} b_i > B$ (otherwise picking the ground set is optimal). Let $\tau_i = \tau(1+\epsilon)^{i-1}$. Notice that for OPT $\leq 2\max_{e\in N} f(\{e\})$ the final solution has value at least 0.5 OPT due to the singleton $\arg\max_{e\in N} f(\{e\})$. So from here on we let OPT $> 2\max_{e\in N} f(\{e\})$. Consequently, there exists an i such that

$$(1-\epsilon)\frac{2\text{OPT}}{3B} \leq \tau_i \leq \frac{2\text{OPT}}{3B}.$$

We will show that $f(S_i) \geq \frac{1-\epsilon}{3}$OPT. Let B_i denote the total budget utilized by set S_i. By definition of B-Threshold Add,

$$f(S_i) \geq B_i\tau_i.$$

Thus, for $B_i \geq \frac{1}{2}B$ we have $f(S_i) \geq \frac{1-\epsilon}{3}$OPT. So let $B_i < \frac{1}{2}B$. Similar to the proof of Theorem 1.2, consider the set OPT $\cup\, S_i$ and its interleaved partition

$$\{O_1, \{s_1\}, O_2, \{s_2\}, \cdots, \{s_{k_i}\}, O_{k_i+1}, E_{k_i+1}, O_{k_i+2}\},$$

where s_j is the j-th element added to S_i. The main difference is the set E_{k_i+1}. This set (may) contain elements of OPT that meet the threshold requirement but cannot be added to S_i due to the budget constraint. Applying Corollary B.2, we have

$$f(\text{OPT} \cup S_i) \leq f(S_i) + f(E_{k_i+1} \mid S_i) + \sum_{\ell\in[k_i+1]} f(O_\ell \mid E(\ell-1)).$$

We consider two cases based on the set E_{k_i+1}. Let ALG denote the value of solution generated by the algorithm.

Case I: $E_{k_i+1} = \emptyset$. Let B_j denote the total budget of set O_j. Using the upper bound $f(O_j \mid E(j-1)) \leq \tau_i B_j$ for every $j \in [k_i+1]$, we have

$$\text{OPT} \leq f(S_i) + \tau_i \sum_{j\in[k_i+1]} B_j = f(S_i) + \frac{2}{3}\text{OPT}.$$

Thus, ALG $\geq f(S_i) \geq \frac{1}{3}$OPT.

Case II: $E_{k_i+1} \neq \emptyset$. Then, there exists i^* such that $f(i^* \mid S_i) \geq \tau_i b_{i^*}$ and $b(S_i) + b_{i^*} > B$. Therefore,

$$f(S_i + \{i^*\}) \geq \tau_i B \geq (1-\epsilon)\frac{2}{3}\text{OPT}. \tag{6}$$

Now, the algorithm outputs the best of S_i and all singletons. Therefore,

$$2\text{ALG} \geq f(i^*) + f(S_i) \geq f(S_i + \{i^*\}),$$

here second inequality follows from subadditivity. Using (6) completes the proof.

Given a noisy function oracle f with multiplicative error $(1 \pm \delta)$, we have, $\hat{f}(S_i) \geq B_i\tau_i$. For $B_i \geq 0.5B$, we get, $f(S_i) \geq \frac{1-\epsilon}{3(1+\delta)}$ OPT. For $B_i < 0.5B$ and $E_{k_i+1} = \emptyset$, we have, $f(O_j \mid E(j-1)) \leq \tau_i B_j + 2\frac{|O_j|\delta}{1-\delta}$OPT for every $j \in [k_i+1]$. Thus, $f(S_i) \geq (1 - \frac{6n\delta}{1-\delta})\,\frac{\text{OPT}}{3}$. Finally, for $E_{k_i+1} \neq \emptyset$, we have, $\hat{f}(S_i + \{i^*\}) \geq \tau_i B$, and therefore, ALG $\geq (1 - O(\delta))\frac{\text{OPT}}{3}$. Overall, in the presence of a noisy oracle we have approximation guarantee $\frac{\min\{(1-O(\delta))(1-\epsilon),\, 1-\frac{6n\delta}{1-\delta}\}}{3} = (1-\epsilon)\left(1 - O(\frac{n\delta}{1-\delta})\right)\,\frac{1}{3}$.

□

Proof of Theorem 1.2(ii). For each X the algorithm makes $O(\frac{n}{\epsilon}\log n)$ queries and there are $O(n^{1/\epsilon})$ possible sets X leading to $O(\frac{1}{\epsilon}n^{1+\frac{1}{\epsilon}}\log n)$ queries in total.

To show the guarantee we focus on set $S(X)$ when $X \subseteq \text{OPT}$ contains the $1/\epsilon$ largest budget elements in OPT (or all of OPT if its cardinality is small). Given X the budget b_e required by any $e \in \text{OPT}\backslash X$ is strictly smaller than ϵB, otherwise $b(X) > B$.

Consider $\tau_i \in \left[(1-\epsilon)\frac{\text{OPT}}{2B}, \frac{\text{OPT}}{2B}\right]$ and let S_i denote the solution returned by the algorithm with X and threshold τ_i. We show that $f(S_i) \geq (0.5-\epsilon)\,\text{OPT}$. The analysis is split in two cases based on how the algorithm terminates.

Case I: Final Add is not invoked. In this case every element not in S_i fails the threshold requirement and the algorithm does not remove elements after they are chosen. Similar to the analysis of Algorithm 1, we have an interleaved partition

$$\{O_1, \{s_1\}, O_2, \{s_2\}, \cdots, \{s_{k_i}\}, O_{k_i+1}\},$$

of $\text{OPT} \cup S_i$ such that s_j is the jth element added to S_i. Using Corollary B.2, we have

$$\text{OPT} \leq f(S_i) + \sum_{j=1}^{k_i+1} f\left(O_j \mid E(j-1)\right).$$

Now, $f\left(O_j \mid E(j-1)\right) \leq \tau_i |O_j|$, $\forall j \in [k_i+1]$. Thus, $\text{ALG} \geq f(S_i) \geq 0.5\,\text{OPT}$.

Case II: Final Add is invoked for some element j. We claim that the set S_i output by Final Add is such that (i) $B \geq b(S_i) \geq (1-\epsilon)\,B$ and (ii) $f(S_i) \geq \tau_i b(S_i)$. Using (i) and (ii), we have for $\epsilon \in [0,1]$,

$$f(S_i) \geq 0.5\,(1-\epsilon)^2\,\text{OPT} \geq\ (0.5-\epsilon)\,\text{OPT}.$$

It remains to show (i) and (ii). Let S_{in} denote the set that is input to Final Add and let $X_i = \{e \mid b_e \geq \epsilon B, e \in S_{in}\}$. Observe that $X_i \subseteq X$, since every element outside X with budget exceeding ϵB is discarded in the beginning. Therefore, $b(X_i) \leq B$ and the set S_i returned by Final Add is feasible (and contains X_i). To see the lower bound on $b(S_i)$, let t denote the last element removed from S_{in} by Final Add. We have, $b_t < \epsilon B$ and $b_t + b(S_i) > B$. Thus, $b(S_i) \geq B - \epsilon B$.

To show (ii) we use Corollary B.3(ii). For simplicity, let $\{1, 2, \cdots, s\}$ denote the elements of S_i in submodular order. Recall that $S_i \subset S_{in}$. Using Corollary B.3(ii) with $A = S_{in}$ and sets $E_k = \{k\}$ for $k \in [s]$, we have

$$\sum_{k\in[s]} f(k \mid L_k) \leq f(S_i),$$

where L_k denotes the set of all elements in S_{in} chosen by the algorithm prior to k. From the threshold requirement, we have $f(k \mid L_k) \geq \tau_i b_k$, $\forall k \in [s]$. Thus, $f(S_i) \geq \tau_i b(S_i)$.

In case of a noisy oracle, similar to previous analyses it can be verified that the approximation guarantee is reduced by a multiplicative factor $\left(1 - O(\frac{n\delta}{1-\delta})\right)$. □

B.3. Matroid Constraint

Lemma B.4. *Given a matroid $\mathcal{M}$ and an independent set A, define $\mathcal{I}_A = \{B \mid A \cup B \in \mathcal{I},\, B \subseteq N\backslash A\}$. For every independent set A' such that $r(A') = r(A \cup A') = r(A)$, we have $\mathcal{I}_A = \mathcal{I}_{A'}$.*

Proof. Consider an arbitrary set $B \in \mathcal{I}_A$. It suffices to show that $B \in \mathcal{I}_{A'}$. There are many ways to show this, in the spirit of submodularity we will prove this by using the submodularity of rank function $r(\cdot)$.

Note that $r(A \cup B) = |A| + |B| = |A'| + |B|$ and consider the sets $A' \cup B$ and $A \cup A'$. We have from submodularity,

$$\begin{aligned} r\left(A' \cup A \cup B\right) + r\left(A'\right) &\leq r\left(A' \cup B\right) + r\left(A \cup A'\right), \\ r\left(A' \cup A \cup B\right) &\leq r\left(A' \cup B\right). \end{aligned}$$

Therefore, $r(A' \cup B) = r(A' \cup A \cup B)$. Since $r(A' \cup B) \leq |A'| + |B| = r(A \cup B)$, we have that $A' \cup B$ is independent and $B \subseteq N\backslash A'$. □

Lemma B.5. *At the end of every iteration, marked by the element $j \in [n]$ that was parsed, the set S maintained by Algorithm 3 is maximally independent i.e., $r(S)$ equals the rank of the set $[j]$ that contains all elements parsed so far.*

Proof. Note that the algorithm always maintains an independent set and the lemma is true after the first iteration. For the sake of contradiction, let $j+1$ be the first element in submodular order where the lemma is not true. Let S_j denote the set at the end of iteration j (after element j is parsed). By definition of j, $S_j \cup \{j+1\}$ is not independent but $r([j+1]) = r(S_j)+1$. Using the fact that S_j is rank maximal subset of $r([j])$, there exists an independent set $S'_j \in [j]$ with the same rank as S_j and such that $S'_j \cup \{j+1\}$ is independent. Applying Lemma B.4 with $A = S_j$ and $A' = S'_j$, we have $\{j+1\} \in \mathcal{I}_{S_j}$, a contradiction.

□

Proof of Theorem 1.2. Algorithm 3 parses the ground set once. When parsing an element, the algorithm makes at most d queries, resulting in total nd queries.

To show the performance guarantee. Let S_j denote the set S maintained in the algorithm at the beginning of iteration j. Similarly, let R_j denote the set R at the beginning of iteration j. Observe that R_j is the set of all elements that were chosen and later swapped out, prior to iteration j. Therefore, the set $S_j \cup R_j$ grows monotonically and includes all elements selected by the algorithm prior to j. We use S and R to denote the final sets when the algorithm terminates. Let OPT denote the optimal set (and value) and ALG denote the algorithm output (and value). By monotonicity, $f(\text{ALG} \cup \text{OPT}) \geq \text{OPT}$. We will "essentially" show that $f(\text{OPT} \mid \text{ALG}) \leq 3\text{ALG}$. The main elements that contribute to this upper bound are,

Swap operations: If an element j is taken out of the set at iteration t, then the elements in OPT parsed between j and t, but never included, may have larger marginal value after the swap. We show that the resulting total increase in marginal value is upper bounded by ALG.

Rejection of elements: Any element that is rejected by the algorithm on parsing (due to insufficient marginal value for swapping) may be an element of OPT. Let j denote the index (in submodular order) of such an element. Since j is rejected we have that $S_j \cup \{j\}$ contains a circuit C_j and the marginal $f(j \mid S_j \cup R_j)$ is upper bounded by v_i for all $i \in C_j$. We show that the total marginal value $\sum_{j \in \text{OPT}\setminus(S\cup R)} f(j \mid S_j \cup R_j)$ of elements in OPT rejected by the algorithm is at most 2ALG.

W.l.o.g., we ignore all elements in $N\setminus(\text{OPT} \cup S \cup R)$. So let $\hat{N} = \text{OPT} \cup S \cup R$ denote our ground set and re-index elements in $\hat{N}$ from 1 to $|\hat{N}|$ (maintaining submodular order). Similarly, we re-index the sets $\{S_j\}$ so that they continue to denote the set maintained by the algorithm when element $j \in \hat{N}$ is first parsed, for every $j \in \{1, 2, \cdots, |\hat{N}|\}$. From monotonicity, $\text{OPT} \leq f(\hat{N})$.

Consider an interleaved partition $\{O_\ell, E_\ell\}$ of $\hat{N}$ such that $E(m) = \cup_{\ell \in [m]} E_\ell := S \cup R$ and $O(1) = \cup_{\ell \in [m]} O_\ell := \text{OPT}\setminus(S \cup R)$. Using Corollary B.2,

$$\begin{aligned} \text{OPT} &\leq f(E(m)) + \sum_{\ell \in [m]} f(O_\ell \mid E(\ell-1)), \\ &\leq f(E(m)) + \sum_{j \in \text{OPT}\setminus(S\cup R)} f(j \mid S_j \cup R_j), \end{aligned}$$

where the second inequality follows from weak submodular order. Next, we apply Corollary B.3(i) to upper bound $f(E(m))$ in terms of ALG. Consider an interleaved partition $\{\bar{O}_\ell, \bar{E}_\ell\}_{\ell \in [\bar{m}]}$ of $E(m)$ such that $\bar{E}(\bar{m}) = S$ and $\bar{O}(1) = R$. We have,

$$\begin{aligned} f(E(m)) &\leq \text{ALG} + \sum_{\ell \in [\bar{m}]} f\left(\bar{O}_\ell \mid \bar{E}(\ell-1) \cup \bar{O}(1)\setminus\bar{O}(\ell)\right), \\ &= \text{ALG} + \sum_{j \in R} f(j \mid S_j \cup R_j). \end{aligned}$$

We upper bound $\sum_{j \in R} f(j \mid S_j \cup R_j)$ by ALG and $\sum_{j \in \text{OPT}\setminus(S\cup R)} f(j \mid S_j \cup R_j)$ by 2ALG. This proves the main claim.

Part I: $\sum_{j \in R} f(j \mid S_j \cup R_j) \leq \text{ALG}$.

Consider an element $j_1 \in R$ that was added to S_j without swapping out any element. Since $j_1 \in R$, there exists an element j_2 that replaced j. Inductively, for $t \geq 2$, let j_t denote the tth element in the chain of swaps $j_1 \to j_2 \to \cdots \to j_t$. The chain terminates at an element in S and we call this a *swap chain*. From the swap criteria in Algorithm 3, we have

$$\sum_{\tau \in [t-1]} f\left(j_\tau \mid S_{j_\tau} \cup R_{j_\tau}\right) \leq f\left(j_t \mid R_{j_t} \cup S_{j_t}\right).$$

Every element in R is part of a unique swap chain and each chain has a unique terminal element in S. Therefore,

$$\sum_{j \in R} f\left(j \mid S_j \cup R_j\right) \leq \sum_{i \in S} f\left(i \mid S_i \cup R_i\right).$$

Applying Corollary B.3(ii) with $A = S \cup R$ and sets $E_i = \{i\}$ for $i \in S$, we have

$$\sum_{i \in S} f\left(i \mid S_i \cup R_i\right) \leq \text{ALG}. \tag{7}$$

Observe that given a noisy function oracle $\hat{f}$, we have,

$$\begin{aligned}
\sum_{j \in R} f\left(j \mid S_j \cup R_j\right) &\leq \sum_{j \in R} \hat{f}\left(j \mid S_j \cup R_j\right) + \frac{O(|R|\delta)}{1-\delta} f(\hat{N}), \\
&\leq \sum_{i \in S} \hat{f}\left(i \mid S_i \cup R_i\right) + \frac{O(|R|\delta)}{1-\delta} f(\hat{N}), \\
&\leq \sum_{i \in S} f\left(i \mid S_i \cup R_i\right) + \frac{O(n\delta)}{1-\delta} f(\hat{N}),
\end{aligned}$$

here the first and third inequalities follow from the definition of noisy oracle and the second inequality follows from the swap criteria. Thus, $\sum_{j \in R} f\left(j \mid S_j \cup R_j\right) \leq \text{ALG} + + O(\frac{n\delta}{1-\delta}) f(\hat{N})$.

Part II: $\sum_{j \in \text{OPT} \setminus (S \cup R)} f\left(j \mid S_j \cup R_j\right) \leq 2\,\text{ALG}$.

From Lemma B.5 we have that the rank $r\left(\text{OPT}\right) \leq r\left(\text{ALG}\right)$. Suppose there exists an injection ϕ from elements in $\text{OPT} \setminus (S \cup R)$ to elements in S such that

$$f\left(j \mid S_j \cup R_j\right) \leq 2 f\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right), \quad \forall j \in \text{OPT} \setminus (S \cup R).$$

Summing up these inequalities and using (7), we are done. It remains to show that ϕ exists. We do this via a graphical construction inspired by (Chekuri et al., 2015).

Consider a graph G with vertices given by $\hat{N}$. In order to define the edges recall that every $j \in \text{OPT} \setminus (S \cup R)$ is rejected by the algorithm on parsing. Thus, we have a unique circuit C_j where $C_j \setminus \{j\} \subseteq S_j$. For our first set of edges we make a directed edge from j to every element in $C_j \setminus \{j\}$ and we do this for all $j \in \text{OPT} \setminus (S \cup R)$. Next, for every $j \in R$, let C_j represent the chain that causes j to be swapped out in the algorithm. We create a directed edge from j to every element in $C_j \setminus \{j\}$. Graph G has the following properties,

(a) The elements of $\text{OPT} \setminus (S \cup R)$ are *source* vertices with no incoming edges. Elements of S are *sinks* with no outgoing edges.

(b) The neighbors of every node in G form a circuit with the node.

(c) Given arbitrary vertex $j \in \text{OPT} \setminus (S \cup R)$, for every i reachable from j we claim that

$$f(j \mid S_j \cup R_j) \leq v_i,$$

where v_i corresponds to the value defined in Algorithm 3. For neighbors of j the claim follows directly from the swap criterion. Also, for any two neighboring vertices $i', i'' \in S \cup R$, we have $v_{i'} \leq v_{i''}$. The general claim follows by using these inequalities for every edge on the path from j to i.

Using (a) and (b) we apply Lemma B.6 to obtain an injection ϕ from $\text{OPT}\backslash(S\cup R)$ to S such that for every $j \in \text{OPT}\backslash(S\cup R)$ there exists a path to $\phi(j) \in S$. Then, from (c) we have, $f(j \mid S_j \cup R_j) \le v_{\phi(j)}$. Recall the notion of a swap chain defined in Part I and let $W(\phi(j))$ denote the set of all preceding elements of the swap chain that terminates at $\phi(j)$. By definition of $v_{\phi(j)}$ and the swap criterion,

$$v_{\phi(j)} = f\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right) + \sum_{i \in W(\phi(j))} f\left(i \mid S_i \cup R_i\right) \le 2f\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right).$$

Given a noisy function oracle $\hat{f}$, by the preceding argument there exists an injection ϕ from $\text{OPT}\backslash(S \cup R)$ to S such that, $\hat{f}\left(j \mid S_j \cup R_j\right) \le 2\hat{f}\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right),\ \forall j \in \text{OPT}\backslash(S \cup R)$. Thus, $\sum_{j\in\text{OPT}\backslash(S\cup R)} f\left(j \mid S_j \cup R_j\right) \le 2\,\text{ALG} + +O(\frac{n\delta}{1-\delta})f(\hat{N})$. Overall, for a noisy oracle we have $\text{ALG} \ge (1 - O(\frac{n\delta}{1-\delta}))\,0.25\,f(\hat{N}) \ge (1 - O(\frac{n\delta}{1-\delta}))\,0.25\,\text{OPT}$. □

Lemma B.6 (Lemma 30 in (Chekuri et al., 2015))**.** *Let $M = (N, \mathcal{I})$ be a matroid, and G a directed acyclic graph over N such that for every non-sink vertex $e \in N$, the outgoing neighbors of e form a circuit with e (alternatively, e is in the span of its neighbors). Let $I \in \mathcal{I}$ be an independent set such that no path in G goes from one element in I to another. Then there exists an injection from I to sink vertices in G such that each $e \in I$ maps into an element reachable from e.*

B.4. Upper Bound of 0.5

The following lemma is useful in verifying the (strong) submodular order property for a given function and order. We use it subsequently in the proof of Theorem 1.3 and again in Section D.

Lemma B.7. *A monotone subadditive function f is π (strongly) submodular ordered if for any two sets $B \subseteq A$ and an element i to the right of A, we have $f(i \mid A) \le f(i \mid B)$.*

Proof. Strong submodular order implies $f(i \mid A) \le f(i \mid B)$ by definition. To see the reverse direction, consider an arbitrary set A, subset B of A, and a set C such that $l_\pi(C) > r_\pi(A)$. Let $C = \{c_1, \cdots, c_k\}$ represent the elements of C in submodular order and define subsets $C_i = \{c_1, \cdots, c_i\}, \forall i \in [k-1]$ and $C_0 = \{\emptyset\}$. We are given,

$$f(c_i \mid A \cup C_{i-1}) \le f(c_i \mid B \cup C_{i-1}), \quad \forall i \in [k].$$

Therefore,

$$f(C \mid A) = \sum_{i\in[k]} f(c_i \mid A \cup C_{i-1}) \le \sum_{i\in[k]} f(c_i \mid B \cup C_{i-1}) = f(C \mid B),$$

where we use telescoping marginal values to rewrite $f(C \mid A)$ and $f(C \mid B)$. □

Proof of Theorem 1.3. At a high level, we construct a family of submodular order functions where the (unique) optimal solution is "well-hidden" i.e., only a small subset of the optimal solution can be found by polynomial number of queries to the value oracle. By design, the value of every feasible solution that has small overlap with the optimal set is at most half the optimal value. In the following, we use non-negative integers n_1, n_2, k_1, k_2, r and real value $\alpha > 0$ that is defined later. Our choice of these values will obey the following rules $n_1 > k_1 > n_2 = k_2$, $\alpha = k_1/k_2$ and $r < \alpha k_2 = k_1$. Let N denote the ground set (size n) with partition $\{N_1, N_2\}$. Let n_i denote the cardinality of N_i for $i \in \{1, 2\}$. For sets $S \subseteq N_1$, we define $f(S) = \min\{|S|, 2k_1\}$. Under this definition, we have a monotone submodular function on the ground set N_1. Notice that, for all sets $S \subseteq N_1$ with $|S| \le 2k_1$, the function value equals $|S|$. Further, subsets of N_1 of the same size have identical value. For sets $S \subseteq N_2$, we define $f(S) = \alpha|S|$. Notice that $f(N_2) = \alpha k_2 = k_1$. Our overall ground set is given by $N_1 \cup N_2$ so it remains to define function value on sets that intersect both N_1 and N_2.

For a special set $A_1 \subseteq N_1$ with $|A_1| = k_1$, we define

$$f(A_1 \cup S_2) := k_1 + \alpha|S_2| - r \quad \forall S_2 \subseteq N_2.$$

Recall that $r < \alpha k_2 = k_1$ thus, $f(A_1 \cup S_2) > \max\{f(A_1), f(S_2)\}$. More generally, for $B_1 \subseteq A_1$ and $S_2 \subseteq N_2$,

$$f(S_2 \mid B_1) := f(S_2)\left(1 - \frac{\min\{r, f(B_1)\}}{k_1}\right) = |S_2|\left(\alpha - \frac{\min\{r, |B_1|\}}{k_2}\right).$$

Next, for arbitrary sets $S_i \subseteq N_i$,

$$f(S_2 \mid S_1) \quad := \quad f(S_2)\left(1 - \min\left\{1, \frac{|S_1 \backslash A_1| + \min\{r, |S_1 \cap A_1|\}}{k_1}\right\}\right). \tag{8}$$

Observe that for $|S_1 \cap A_1| \leq r$, this simplifies to,

$$f(S_2 \mid S_1) = f(S_2)\left(1 - \min\left\{1, \frac{|S_1|}{k_1}\right\}\right).$$

By definition, we have that for every element $e \in A_1$, the marginal $f(e \mid N_2) = 0$. However, given a set $B_1 \subset A_1$ with $|B_1| > r$ and $e \in A_1 \backslash B_1$, we have $f(e \mid B_1 \cup N_2) = 1$, Therefore, f is not a submodular function. We claim that it is monotone subadditive and has a strong submodular order.

To see monotonicity, consider function value in the form $f(S_1) + f(S_2 \mid S_1)$. If an element is added to S_2, this sum does not decrease. Now, if $|S_1| \geq 2k_1$, then $f(S_2|S_1) = 0$ and we also have monotonicity w.r.t. elements added to S_1. Finally, when $|S_1| < 2k_1$, adding an element to S_1 increases $f(S_1)$ by 1 and can decrease $f(S_2 \mid S_1)$ by at most $\frac{f(S_2)}{\alpha k_2} \leq 1$. To see subadditivity, observe that $f(S_2) \geq f(S_2 \mid S_1)$.

Fix some order π where N_1 is entirely to the left of N_2. We show that π is a strong submodular order for f. Consider sets $B \subseteq A$ and an element i to the right of A. Using Lemma B.7, it suffices to show that $f(i \mid A) \leq f(i \mid B)$. We establish this in cases.

Case I: $A \cup \{i\} \subseteq N_j$ for $j \in \{1, 2\}$. Note that $A \subseteq N_j$ implies $B \subseteq N_j$. The inequality follows from the fact that f is monotone and submodular when restricted entirely to either N_1 or N_2.

Case II: $A \subseteq N_1$ and $i \in N_2$. Since $B \subseteq A$, we have, $|B \backslash A_1| \leq |A \backslash A_1|$ and $|B \cap A_1| \leq |A \cap A_1|$. Then from (8), we have that $f(i|B) \leq f(i|A)$ by definition.

Case III: A intersects both N_1 and N_2. Since i is to the right of A in order π, we have that $i \in N_2$. In this case, we claim that,

$$f(i \mid B + e) \leq f(i \mid B), \quad \forall i \in N_2, e \notin B + i. \tag{9}$$

Applying (9) repeatedly gives us the desired. To see (9), consider two cases. First, suppose $e \in N_2$. In this case, observe that the submodularity of f on N_2 gives us the desired. In the second case, $e \in N_1$, and from (8), we see that the marginal value of i does not increase. This establishes the submodular order property for f.

Now, consider the problem of maximizing f subject to cardinality constraint $k := k_1 + k_2$. Observe that the set $A_1 \cup N_2$ is the unique optimum with value $2k_1 - r$. Further, every feasible set with at most r elements from A_1, has value at most k. Choosing $k_1 = n_1^{0.5-\delta}$ (i.e., $o(\sqrt{n_1})$), and $n_2 = k_2 = r = k_1^{1-\delta}$ (i.e., $o(k_1)$) for some small $\delta > 0$, we have that as $k_1 \to +\infty$,

$$k = k_1 + k_2 = k_1 + o(k_1) < (0.5 + \epsilon)\,\mathrm{OPT},$$

for any constant $\epsilon > 0$. Therefore, an algorithm has approximation guarantee strictly greater than 0.5 only if it outputs a set with more than r elements from A_1. We generate A_1 by sampling k_1 distinct elements from N_1 uniformly at random and show that the random set A_1 is "well hidden" from polynomial query algorithms. Formally, we show that when A_1 is generated uniformly randomly, any given polynomial query algorithm will fail to find a feasible set with more than r elements of A_1 with probability approaching 1 in the limit of $k_1 \to \infty$. This proves the main claim.

W.l.o.g., we consider algorithms that query only marginal values $f(X \mid Y)$ for sets $X, Y \subseteq N_i$ for $i \in \{1, 2\}$. A query in any other form can be converted to a constant number of such queries. We call a set S *good*, if

$$S \subseteq N_1,\ |S| < 2k_1,\ \text{and } |S \cap A_1| > r.$$

If at least one of the sets X or Y is good, we call $f(X|Y)$ a *good* query. All other queries are called *bad*. To show that A_1 is well hidden, we show that, (i) Unless a query $f(X|Y)$ is good, we do not get any useful information about finding a good set. (ii) A polynomial (query) algorithm fails to make a good query with probability approaching 1 (as $k_1 \to +\infty$). Combining (i) and (ii), we have that a polynomial algorithm will fail to find any good set with probability approaching 1.

To show (i), notice that queries with $X \cup Y \subseteq N_i$ give no information (on finding a good set). Thus, it suffices to consider queries with $X \subseteq N_i$ and $Y \subseteq N_{-i}$, where $\{i, -i\} = \{1, 2\}$. We consider all possible bad queries through the following cases.

Case I: $X \subseteq N_1$, $Y \subseteq N_2$, and $|X| \geq 2k_1$. In this case, we call X a *large* set. Observe that given a set $Y \subseteq N_2$, we have $f(X \cup Y) = 2k_1$. Thus, marginal value $f(X \mid Y)$ is identical for every X and Y and we obtain no useful information.

Case II: $Y \subseteq N_1$, $X \subseteq N_2$, and $|Y| \geq 2k_1$. In this case, observe that $f(X \mid Y) = 0$ for every $X \subseteq N_2$ and we obtain no useful information.

Case III: $|X \cap A_1| \leq r$ and $|Y \cap A_1| \leq r$. First, for $Y \subseteq N_2$, notice that every set $X \subseteq N_1$ with $|X \cap A_1| \leq r$ has marginal value $f(X \mid Y) = f(X) + f(Y \mid X) - f(Y)$, which only depends on the cardinality of X and Y. Similarly, given $Y \subseteq N_1$ with $|Y \cap A_1| \leq r$, every set $X \subseteq N_2$ of a given cardinality has the same marginal value $f(X \mid Y)$. Thus, we find no useful information on good sets.

Therefore, given a collection of bad queries $\{f(X_i \mid Y_i)\}$, we do not obtain any useful information on finding a good set. Since a query where at least one of X or Y is a large set gives no useful information on good sets, to show (ii), it suffices to consider algorithms that only make queries $f(X|Y)$ such that, $\max\{|X|, |Y|\} < 2k_1$. We call a set with less than $2k_1$ elements *small*.

First, consider the special case of deterministic algorithms. Consider an arbitrary *small* set $Z \subseteq N_1$. Since A_1 is selected uniformly randomly, using a standard concentration bounds for negatively correlated random variables we have that the probability $P[|Z \cap A_1| > r]$, is exponentially small in n_1 (see Lemma B.8 for a proof). Given polynomially many small sets $\{Z_i\}_{i \in [n']}$, using the union bound we have an exponentially small probability of the event that $|Z_i \cap A_1| > r$ for some $i \in [n']$. Thus, a deterministic algorithm with total number of queries polynomial in n_1 will, with probability arbitrarily close to 1, fail to make a good query. Given a randomized algorithm, we condition on its random seed (that is independent of randomness in our instance), to obtain a deterministic algorithm. Given a randomized polynomial algorithm, the conditionally deterministic algorithm must make polynomial number of queries. Thus, it fails to make a good query with probability 1 (asmptotically). Unconditioning on the random seed then gives us the desired bound for every (randomized) polynomial query algorithm. □

Lemma B.8. *Consider a ground set N of n elements. Let A be a subset of $k = \lfloor n^{0.5-\delta} \rfloor$ elements chosen uniformly at random from N for some small $\delta > 0$. Then, for any S with cardinality at most $2k$ and $r = k^{1-\delta}$, we have*

$$P\left[|S \cap A| > r\right] < e^{-\Theta(n^{1-\delta})}.$$

Proof. Let $S = \{1, 2, \cdots, s\}$, where $s \leq 2k$. For every $i \in [s]$, define indicator random variable X_i that is 1 if $i \in A$ and 0 otherwise. Note that $P[X_i = 1] = k/n$ for every $i \in [s]$. We claim that these random variables are negatively correlated such that for every $\hat{S} \subseteq S$,

$$P\left[\bigwedge_{j \in \hat{S}} X_j = 1\right] \leq \prod_{j \in \hat{S}} P\left[X_j = 1\right].$$

We prove this by induction. The claim is true for singleton sets $\hat{S}$. We assume the claim holds for all sets $\hat{S}$ with cardinality $\hat{k}$ or less. Consider a set $\hat{S} + e$ of size $\hat{k} + 1$ with $e \in S \backslash \hat{S}$. We have,

$$P\left[X_e = 1 \,\middle|\, \bigwedge_{j \in \hat{S}} X_j = 1\right] = \frac{k - \hat{k}}{n - \hat{k}} \leq P[X_e = 1],$$

where the inequality follows from $k < n$. Now, a direct application of the generalized Chernoff bound given in Theorem 3.4 in Panconesi and Srinivasan (Panconesi & Srinivasan, 1997), gives us

$$P\left[|S \cap A| > (1 + \epsilon)2n^{-2\delta}\right] < e^{-\frac{\Theta(\epsilon^2)}{n^{2\delta}}},$$

for $\epsilon > 0$. Substituting $\epsilon = O(n^{0.5+0.5\delta})$ completes the proof. □

C. Algorithms for Constrained Assortment Optimization

In this section, we translate the algorithms from submodular order maximization to constrained assortment optimization. Recall that the revenue objective in assortment optimization can be a non-monotone function. So we transform the objective to $f_\phi(S) = \max_{X \subseteq S} R_\phi(S)$, which is monotone and subadditive for choice models that are substitutable. In an ideal scenario, we have a choice model ϕ where the unconstrained assortment optimization can be solved efficiently (or admits a FPTAS) and f_ϕ has a submodular order. If a submodular order is not evident, or does not exist, but the choice model is compatible (a property of optimal unconstrained assortments), we propose a new framework.

Our framework for compatible choice models builds on the algorithms for submodular order functions and sequentially constructs a partial order over the ground set, satisfying a relaxed notion of the submodular order property. To define the framework, we need some notation. Let $\mathcal{A}$ represent an algorithm for constrained submodular order maximization (out of Algorithms 1, 4, 6 and 3). An instance of the assortment optimization problem is given by set N, function f_ϕ, and constraint $\mathcal{F}$. Let $\Gamma_{\mathcal{A}}(N, f_\phi, \mathcal{F})$ define the set of parameter settings that $\mathcal{A}$ enumerates over. We explicitly define this set for each algorithm later. For brevity, we write $\Gamma_{\mathcal{A}}(N, f_\phi, \mathcal{F})$ simply as $\Gamma_{\mathcal{A}}$. Given a parameter setting $\gamma \in \Gamma_{\mathcal{A}}$, let

$$(S, R) := \mathcal{A}_\gamma(f, N, \pi, \mathcal{F}),$$

denote the output of $\mathcal{A}$ when ground set N is ordered according to π. Set S is the feasible solution generated by $\mathcal{A}_\gamma$. R is a minimal set such that every query made by $\mathcal{A}_\gamma$ during its execution is a marginal value $f(\cdot \mid X)$ for some $X \subseteq S \cup R$. Finally, let $U_\phi(X)$ denote an optimal unconstrained assortment on ground set X. Observe that $f_\phi(X)$ is the expected revenue of $U_\phi(X)$. Before stating our new framework, we explicitly define $\Gamma_{\mathcal{A}}$ and R for each algorithm.

$\Gamma_{\mathcal{A}}$ and R for Algorithms 1, 4, and 6: $\Gamma_{\mathcal{A}}$ is the set of all threshold values τ tried in the algorithm. For Algorithm 6, the set also includes sets X of high budget elements used to filter the ground set. In Algorithms 1 and 4 marginal values are evaluated only w.r.t. subsets of the output S. So $R = \emptyset$ for every parameter setting. In Algorithm 6, R is the set of elements discarded by the Final Add subroutine (Algorithm 7).

$\Gamma_{\mathcal{A}}$ and R for Algorithm 3: There is no enumeration in Algorithm 3 so $\Gamma_{\mathcal{A}}$ is a singleton set (the **for** loop executes once). R is the set of all elements that were picked by the algorithm and later swapped out.

ALGORITHM 8: Constrained Assortment Optimization for Compatible Choice Models

Input: Algorithm $\mathcal{A}$, set N, constraint $\mathcal{F}$, oracles for f_ϕ, U_ϕ;
for $\gamma \in \Gamma_{\mathcal{A}}$ **do**
  Initialize $\hat{N} = U_\phi(N)$, $M = \emptyset$ and arbitrary order $\pi_{\hat{N}} : \hat{N} \to \{1, \cdots, \hat{N}\}$;
  while $\hat{N} \backslash M \neq \emptyset$ **do**
    $(S, R) = \mathcal{A}_\gamma\left(f_\phi, \hat{N}, \pi_{\hat{N}}, \mathcal{F}\right)$;
    $M = S \cup R$;
    $N \to (N \backslash \hat{N}) \cup M$;
    $\hat{N} \to U_\phi(N) \cup M$;
    Update $\pi_{\hat{N}}$ by adding elements in $\hat{N} \backslash M$ last in the order;
  end
  $S_\gamma := S$
end
Output: Best of $\{S_\gamma\}_{\gamma \in \Gamma_{\mathcal{A}}}$

At a high level, given a constrained assortment optimization problem without a submodular order, the framework sequentially constructs a (partial) submodular order. It starts with the unconstrained optimal assortment $U_\phi(N)$, denoted as N_1 for brevity. For a compatible choice model, we show that every order π that places elements in N_1 before all other elements, defines a partial submodular order in the following sense,

$$f(C \mid A) \leq f(C \mid B) \qquad \forall\, B \subseteq A \subseteq N_1,\, C \subseteq N.$$

Given this insight and the fact that our algorithms for submodular order functions parse the ground set in submodular order, the framework now executes $\mathcal{A}$ with the truncated ground set N_1. Let $M_1 = S \cup R$ denote the output of $\mathcal{A}$ after this first

phase. The set $N_1 \backslash M_1$ of elements that are parsed and rejected by $\mathcal{A}$ are now discarded and we update $N \to (N \backslash N_1) \cup M_1$, which is the set of remaining elements. Having parsed elements in N_1, the framework now augments the (partial) submodular order by re-solving for an unconstrained optimal assortment on the new ground set N. The new set $U_\phi(N)$ may now include elements that were not previously parsed by $\mathcal{A}$. Let $N_2 = U_\phi(N) \backslash M_1$ denote this new set of elements. The framework adds N_2 to the end of the partial submodular order (to the right of N_1) and executes $\mathcal{A}$ now on the ground set $M \cup N_2$. Repeating this process several times, new elements are incrementally passed into $\mathcal{A}$ in a growing partial order π. The algorithm terminates when no new elements appear in the unconstrained assortment (after at most n phases).

To summarize, Algorithm 8 employs the algorithms for submodular order maximization to solve a constrained assortment optimization when it is not evident that the choice model has a submodular order. Note that this framework can be used to solve constrained assortment optimization for any choice model which admits an efficient algorithm for solving the unconstrained problem. However, the approximation guarantees may not hold for non-compatible choice models. The analysis of Algorithm 8 is presented in Appendix D.3.

D. Proof of Main Results for Assortment Optimization

In this section, we establish the results for assortment optimization discussed in Section 3.2. Recall that to obtain monotonicity we transform the revenue objective $R_\phi(S)$ in assortment optimization to $f_\phi(S) = \max_{X \subseteq S} R_\phi(S)$. We start by proving the subadditivity of this function for substitutable choice models. Recall that a choice model is substitutable if $\phi(i, S) \geq \phi\left(i, S \cup \{j\}\right), \forall i \in S, j \notin S$. It is worth noting again that both MNL and Markov models are substitutable (Blanchet et al., 2016).

Lemma D.1. *For any substitutable choice model ϕ, the function f_ϕ is monotone subadditive.*

Proof. Monotonicity follows by definition. To see subadditivity, consider any two sets S_1 and S_2 and let $Y = \arg\max_{X \subseteq S_1 \cup S_2} R_\phi(X)$. Using substitutability,

$$\begin{aligned} f_\phi(S_1 \cup S_2) = R_\phi(Y) \quad &= \sum_{i \in S_1 \cap Y} r_i \phi(i, Y) + \sum_{i \in S_2 \cap Y} r_i \phi(i, Y) \\ &\leq \sum_{i \in S_1 \cap Y} r_i \phi(i, S_1 \cap Y) + \sum_{i \in S_2 \cap Y} r_i \phi(i, S_2 \cap Y) \\ &\leq f_\phi(S_1) + f_\phi(S_2). \end{aligned}$$

□

Next, we prove the submodular order property for MNL choice models and obtain new results for the joint customization and assortment optimization problem.

D.1. New Results in the MMNL Model with Customization

We start by stating a well known property of MNL models. A proof is provided for completeness.

Lemma D.2. *Given an MNL model ϕ, an assortment S, and an element $i \notin S$, we have $R_\phi(i \mid S) \geq 0 \Leftrightarrow r_i \geq R_\phi(S)$. Further, if $r_i \geq R_\phi(S)$ then $r_i \geq R_\phi(S + i)$.*

Proof. We drop ϕ from notation for convenience. Let $v(S) = v_0 + \sum_{j \in S} v_j$. The following chain proves the lemma,

$$R(i \mid S) = \frac{r_i v_i + R(S) v(S)}{v_i + v(S)} - R(S) = \frac{r_i v_i - R(S) v_i}{v_i + v(S)} = v_i \frac{(r_i - R(S))}{v_i + v(S)} \leq r_i - R(S).$$

□

Lemma D.3. *The order π given by sorting products in descending order of price r_i, breaking ties arbitrarily, is a strong submodular order under MNL choice.*

Proof. We omit reference to choice model ϕ from the subscript. Index elements in N in descending order of prices $r_1 \geq r_2 \cdots \geq r_n$, breaking ties arbitrarily. Call this ordering π and consider two sets A, B with $B \subseteq A$. Using Lemma B.7, it suffices to consider an arbitrary element i to the right of A and show $f(i \mid A) \leq f(i \mid B)$.

First, consider the case $r_i < f(A)$. From Lemma D.2 we have $f(i \mid A) = 0$ and non-negativity of f gives us the desired. Now, let $r_i \geq f(A)$. Since r_i is the lowest price element in $A + i$, repeatedly applying (both parts of) Lemma D.2 we have that $f(A) = R(A)$ and $f(A + i) = R(A + i)$ i.e., the optimal unconstrained assortment on sets A and $A + i$ includes all elements. Similarly, using $f(B) \leq f(A)$ we have, $f(B) = R(B)$ and $f(B + i) = R(B + i)$. Thus, it only remains to show that $R(i \mid A) \leq R(i \mid B)$. Letting $v(S) = v_0 + \sum_{j \in S} v_i$, we have

$$R(i \mid A) = \frac{v_i \left(r_i - R(A)\right)}{v_i + v(A)} \;\leq\; \frac{v_i \left(r_i - R(B)\right)}{v_i + v(B)} = R(i \mid B).$$

Notice that the argument does not work if r_i is not the smallest revenue element in A. So, in general, orders other than non-increasing price order are not submodular orders for MNL. □

Remark: (Aouad et al., 2018b) introduce and show a *restricted* submodularity for MNL model. This notion is equivalent to submodularity enforced over "small" sets. Formally, $f(C \mid A) \leq f(C \mid B)$ for all sets $B \subseteq A$ and $|A \cup C| \leq s$, for some $s \geq 2$. This weakening of submodularity is quite distinct from submodular order property.

Now, recall that in the joint customization and assortment optimization problem we seek a selection S of at most k products to maximize,

$$\sum_{j \in [m]} \alpha_j \max_{X \subseteq S} R_{\phi_j}(X)$$

Next, we prove our main result (Theorem 3.2) for this problem.

Proof of Theorem 3.2. Using the definition of f_{ϕ_j} we reformulate the problem as,

$$\max_{S, |S| \leq k} \sum_{j \in [m]} \alpha_j f_{\phi_j}(S).$$

Using Lemma D.1 and Lemma D.3, every f_{ϕ_j} is monotone subadditive and has (strong) submodular order in the direction of descending prices. From Remark 3.1, the function $F := \sum_{j \in [m]} \alpha_j f_{\phi_i}(S)$ is also monotone subadditive and has (strong) submodular order in the direction of descending product prices. Further, the value oracle for F can be implemented efficiently with runtime linear in m. Thus, a straightforward application of results constrained submodular order maximization gives us the desired (including generalizations to budget and matroid constraint). □

D.2. Joint Customization, Pricing, and Optimization

Consider a setting where the prices of products are not fixed and we can customize both the assortment and product prices for each customer type. Let $P = \{p_1, \cdots, p_r\}$ denote the of possible product prices. Given an assortment S and price vector $\boldsymbol{r}_S = (r_i)_{i \in S} \in P^S$, the probability that type j customer chooses item i is specified by an MNL model $\phi_j(i, S, \boldsymbol{r})$. The joint optimization problem is stated as follows,

$$\max_{S, |S| \leq k} \sum_{j \in [m]} \left(\alpha_j \max_{X \subseteq S, \boldsymbol{r}_S} \sum_{i \in S} r_i \phi_j(i, S, \boldsymbol{r}) \right). \tag{10}$$

Corollary D.4. *The joint customization, pricing, and assortment optimization problem under MNL choice has a 0.25 approximation.*

Proof. We prove the corollary by reducing the problem to an instance of matroid constrained joint assortment optimization and customization problem (with fixed prices). For every type $j \in [m]$, recall that the MNL model ϕ_j is specified by parameters v_{j,i,p_r} for every $i \in N, p_r \in P$. Consider the expanded ground set $N_{m,P} = \{(j, i, p_r) \mid j \in [m], i \in N, p_r \in P\}$. For each type $j \in [m]$, define MNL model $\hat{\phi}_j$ on ground set $N_{m,P}$ such that $\hat{v}_{j',i,p_r} = 0$ for every $j' \neq j$ and $\hat{v}_{j,i,p_r} = v_{j,i,p_r}$. Consider the joint customization and assortment optimization problem (with fixed prices) on ground set

$N_{m,P}$ and choice models $\hat{\phi}_j$ for $j \in [m]$, subject to a matroid constraint that ensures that: (i) we pick at most k products and (ii) for every $j \in [m]$ and $i \in [N]$, we pick at most one product from the set $\{(j, i, p_r) \mid p_r \in P\}$. It can be verified that this is an equivalent reformulation of the original pricing problem problem. □

D.3. New Results for Markov Choice and Beyond

D.3.1. Proof of Theorem 3.3

Theorem 9 has two main claims. The first claim is that when ϕ is compatible Algorithm 8 has the same guarantee as the underlying algorithm $\mathcal{A}$. The second claim is that the Markov choice model is compatible. We prove these claims separately. In Part A, we show approximation guarantees for Algorithm 8 under the assumption of a compatible model. Part B, which can be read independently, establishes the compatibility of Markov model.

A - Analyzing Algorithm 8.

At a high level, the analysis has two parts. In Part A-I, we introduce the notion of piece-wise submodular order and show that this notion is sufficient to generalize the guarantees obtained for submodular order functions. In Part A-II, we show that compatible choice models exhibit the required notion of piece-wise submodularity.

Part A-I: To introduce the notion of a piece-wise submodular order, consider an algorithm $\mathcal{A}$ and fix a parameter setting $\gamma \in \Gamma_{\mathcal{A}}$. Suppose that the **while** loop in Algorithm 8 runs in p phases, each defined by a call to U_ϕ. With $M, \hat{N}$ as defined in the algorithm, consider the beginning of phase $i \in [p]$ and define,

$$\text{(Beginning of phase } i) \qquad N_i := \hat{N} \backslash M.$$

Note that N_i contains all elements of $\hat{N}$ that have not been passed to $\mathcal{A}_\gamma$ prior to phase i. At the end of phase i, let M_i be the set of new elements picked by $\mathcal{A}_\gamma$ i.e.,

$$\text{(End of phase } i) \qquad M_i = M \cap N_i.$$

At the end of the last phase, let N_{p+1} denote the set of elements that were never passed to $\mathcal{A}_\gamma$. Let $M_{p+1} = \emptyset$. Notice that $\{N_i\}_{i \in [p+1]}$ is a partition of the original ground set N. Let π denote an order such that

$$r_\pi(N_i) < l_\pi(N_{i+1}), \quad \forall i \in [p].$$

Proper sets: We say that set A is *k-proper* if $A \cap N_i \subseteq M_i$ for $i \leq k-1$ and $A \cap N_i = \emptyset$ for all $i > k$. Using this definition, every subset of N_k is k-proper and given a k-proper set A, all subsets of A (including the empty set) are also k-proper. Finally, every subset of $\cup_{i \in [p]} M_i$ is $p+1$ proper. We use the term proper set to refer to a set that is k-proper for some $k \in [p+1]$.

Piece-wise submodular order: Order π is a $\{N_i, M_i\}_{i \in [p+1]}$ piece submodular order if for every *proper* set A, set $B \subseteq A$, and C to the right of A, we have $f(C \mid A) \leq f(C \mid B)$.

It is worth noting that for $p = 0$ we recover the notion of submodular order. While this notion is weaker than submodular order, we have the following crucial upper bounds in the same vein as Corollary (B.2) and Corollary (B.3).

Lemma D.5. *Consider a monotone subadditive function f on ground set N and $\{N_i, M_i\}_{i \in [p+1]}$ piece submodular order π. For every set A that has an interleaved partitioned $\{O_\ell, E_\ell\}_{\ell \in [m]}$ such that $E(m) \subseteq \cup_{i \in [p+1]} M_i$, we have*

$$f(A) \leq f(E(m)) + \sum_\ell f(O_\ell \mid E(\ell - 1)).$$

Proof. Recall, $E(j) = \cup_{\ell \leq j} E_\ell$ and $O(j) = \cup_{\ell \geq j} O_\ell$. Also, $O(m+1) = \phi$. The following inequalities are crucial for the proof,

$$f\left(O(\ell) \cup E(m)\right) \leq f\left(O_\ell \mid E(\ell - 1)\right) + f\left(O(\ell + 1) \cup E(m)\right) \quad \forall\, \ell \in [m]. \tag{11}$$

Summing up these inequalities for $\ell \in [m]$ we have,

$$\begin{aligned}
\sum_{\ell\in[m]} f\left(O(\ell)\cup E(m)\right) - \sum_{\ell\in[m]} f\left(O(\ell+1)\cup E(m)\right) &\leq \sum_{\ell\in[m]} f\left(O_\ell \mid \cup E(\ell-1)\right), \\
f\left(O(1)\cup E(m)\right) - f\left(E(m)\right) &\leq \sum_{\ell\in[m]} f\left(O_\ell \mid \cup E(\ell-1)\right), \\
f(A) - f\left(E(m)\right) &\leq \sum_{\ell\in[m]} f\left(O_\ell \mid \cup E(\ell-1)\right).
\end{aligned}$$

It remains to show (11). For any $\ell \in [m]$, the sets $B := E(\ell-1)$ and $A := E(\ell-1)\cup O_\ell$ are proper sets and $B \subseteq A$. Further, the set $C := O(\ell+1)\cup E(m)\backslash B$ lies entirely to the right of A. Thus, from the piece-wise submodular order property, we have $f\left(C \mid A\right) \leq f(C \mid B)$ and

$$\begin{aligned}
f\left(O(\ell)\cup E(m)\right) &= f\left(A\right) + f\left(C \mid A\right) \\
&\leq f\left(B\right) + f\left(O_\ell \mid B\right) + f\left(C \mid B\right) \\
&= f\left(O_\ell \mid B\right) + f\left(B\cup C\right) = f\left(O_\ell \mid E(\ell-1)\right) + f\left(O(\ell+1)\cup E(m)\right).
\end{aligned}$$

□

Lemma D.6. *Consider a monotone subadditive function f on ground set N and $\{N_i, M_i\}_{i\in[p+1]}$ piece submodular order π. We have that f is π-submodular ordered on the restricted ground set $\cup_{i\in[p+1]}M_i$. Thus, given a set $A \subseteq \cup_{i\in[p+1]}M_i$, with interleaved partitioned $\{O_\ell, E_\ell\}_{\ell\in[m]}$ and sets $L_\ell = \{e \in A \mid \pi(e) < l_\pi(E_\ell)\}$, $\forall \ell \in [m]$, we have*

(i) $f(A) \leq f\left(E(m)\right) + \sum_{\ell\in[m]} f\left(O_\ell \mid E(\ell-1)\cup O(1)\backslash O(\ell)\right)$,

(ii) $\sum_{\ell\in[m]} f\left(E_\ell \mid L_\ell\right) \leq f\left(E(m)\right)$.

Proof. It suffices to show that f is π-submodular ordered on $\cup_{i\in[p+1]}M_i$. The rest of the lemma the follows from Corollary B.3. On ground set $M_0 = \cup_{i\in[p+1]}M_i$, consider arbitrary sets $B \subseteq A$ and a set C to the right of A. Observe that A is a proper set on the original ground set N. Therefore, using the piece-wise submodular order property on N gives us the desired. □

Interestingly, (the more general) Lemma B.1 is not valid for piece-wise submodular order. Nonetheless, we recover all the guarantees for submodular order functions as the analysis of *every* algorithm $\mathcal{A}$ in this paper relies only on Corollary (B.2) and Corollary (B.3) (for sets $E(m)$), in addition to monotonicity and subadditivity. The following lemma formalizes this observation. See Appendix D.5 for a proof.

Lemma D.7. *Consider an instance of constrained assortment optimization for a compatible choice model ϕ. Suppose we execute Algorithm 8 with $\mathcal{A}$ given by an appropriate algorithm out of Algorithms 1, 4, 6 and 3. If for every parameter setting $\gamma \in \Gamma_{\mathcal{A}}$, the order π and sets $\{N_i, M_i\}_{i\in[p+1]}$ generated by Algorithm 8 are such that π is a $\{N_i, M_i\}_{i\in[p+1]}$ piece submodular order then, Algorithm 8 retains the guarantee of $\mathcal{A}$ for submodular order maximization.*

This concludes the first part of the analysis. We have shown that if f_ϕ satisfies a more relaxed notion of piece-wise submodular order then, Algorithm 8 has the same guarantee as the underlying algorithm $\mathcal{A}$ (when $\mathcal{A}$ is used for functions with submodular order).

Part A-II: In this part we show that compatibility implies the piece-wise submodular order property. Consider a compatible choice model with optimal unconstrained assortment S. Recall that a choice model is compatible if,

$$\begin{aligned}
R(X \mid Z) &\geq 0 && \forall X \subseteq S,\ Z \subseteq N && (2) \\
R(Z \mid X) &\leq R(Z \mid Y) && \forall Y \subseteq X \subseteq S,\ Z \subseteq N && (3)
\end{aligned}$$

Consider sets A, B, and C such that A is a $k+1$-proper set for some $k \geq 0$, $B \subseteq A$, and C is to the right of A in order π. We need to show that, $f(C \mid A) \leq f(C \mid B)$. First, using properties (2) and (3), we show that,

(i) $f(A \cup C) = \max_{X \subseteq C} R(A \cup X)$.

(ii) $R(C \mid A) \leq R(C \mid B)$.

To prove (i) and (ii) for $k+1$ proper sets, define $N_k^- = \cup_{i \leq k} N_i$, $N_k^+ = N \backslash N_k^-$, and $M_k^- = \cup_{i \leq k} M_i$. We claim that the assortment $N_{k+1} \cup M_k^-$ is an optimal unconstrained assortment on ground set $N_k^+ \cup M_k^-$. Since any $k+1$ proper set is a subset of this assortment, using properties (2) and (3) of compatible choice models we have (i) and (ii) as desired. We prove $N_{k+1} \cup M_k^-$ is optimal on ground set $N_k^+ \cup M_k^-$ by induction. For $k = 0$, $M_0^- = \emptyset$ and N_1 is an optimal assortment on N by definition. Suppose the claim holds for $k \in \{0, \cdots, h\}$. Then, $S := N_{h+1} \cup M_h^-$ is an optimal assortment on ground set $N_h^+ \cup M_h^-$. Using (2) with $A := M_{h+1}^- \subseteq S$ and $C := N_{h+1}^+ \cup M_{h+1}^-$, we have that the assortment $N_{h+2} \cup M_{h+1}^-$ is optimal on ground set $N_{h+1}^+ \cup M_{h+1}^-$. This completes the induction and the proof of (i) and (ii).

The main claim now follows from (i) and (ii). From (i) we have, $f(C \mid A) = \max_{X \subseteq C} R(X \mid A)$ and $f(C \mid B) = \max_{X \subseteq C} R(X \mid B)$ (since $B \subseteq A$ is also a proper set). From (ii) we get, $R(X \mid A) \leq R(X \mid B)$ for every X to the right of A. To complete the proof we note that $\max_{X \subseteq C} R(X \mid A) \leq \max_{X \subseteq C} R(X \mid B)$.

B - Proof of Compatibility of Markov Model.

We omit ϕ from subscript for convenience. Let us with some necessary notation and properties of Markov model. Let $P(i \prec Y)$ denote the probability that i is visited before any element in $Y \cup \{0\} \backslash \{i\}$ in the markov chain, here 0 is the outside option. Let $P_j(i \prec Y)$ denote the probability of i being visited before $Y \cup \{0\} \backslash \{i\}$ when the traversal starts at j. (Désir et al., 2020) introduced an important notion of externality-adjustment, where given disjoint sets X and Y they define,

$$R^X(Y) = R(X \cup Y) - R(X).$$

In the following, we summarize properties of this adjustment and some useful lemmas shown in (Désir et al., 2020):

(i) **X-adjusted markov chain:** $R^X(Y)$ $(= R(Y \mid X))$ is the revenue of set Y in a markov chain where the *reduced* price of every $i \in Y$ is,

$$r_i^X = r_i - \sum_{j \in X} P_i(j \prec X) r_j,$$

and reduced prices $r_j^X = 0$ for every $j \in X$. Transition probabilities are adjusted so that $\rho_{j0}^X = 1$ for $j \in X$ (Lemma 4 and Figure 1 in (Désir et al., 2020)). Therefore,

$$R^X(i) = R(i \mid X) = P(i \prec X) r_i^X.$$

(ii) **Composition of adjustments:** Given disjoint sets X, Y and element $i \notin X \cup Y$, from Lemma 5 in (Désir et al., 2020) we have,

$$r_i^{X \cup Y} = r_i^X - \sum_{j \in Y} P_i(j \prec X \cup Y) r_j^X.$$

(iii) Let $r_0 = 0$. From Lemma 7 in (Désir et al., 2020), we have

$$R(i \mid Y) \geq 0, \quad \text{if } r_i \geq \max_{j \in Y \cup \{0\}} r_j, i \notin Y.$$

(iv) Let S be an unconstrained optimal assortment. Then,

$$r_j^A \geq 0, \quad \forall A \subseteq S, j \in S.$$

Property (iv) is not shown directly in (Désir et al., 2020), so we give a proof before proceeding. By optimality of S, the set $B := S \backslash A$ is an optimal unconstrained assortment in the A-adjusted markov chain. Suppose there exists $j \in B$ such that $r_j^A < 0$. Then, we have a contradiction,

$$R^A(B) = \sum_{i \in B} P(i \prec S) r_i^A < \sum_{i \in B \backslash \{j\}} P(i \prec S \backslash \{j\}) r_i^A = R^A(B \backslash \{j\}),$$

where the inequality uses $P(i \prec S) \leq P(i \prec S\backslash\{j\})$, which follows from substitutability of Markov model.

Now, we show (2) and (3). Let S be a *maximal* optimal unconstrained assortment on the ground set N i.e., $R(i \mid S) < 0$ for every $i \notin S$. Consider sets $B \subseteq A \subseteq S$ and arbitrary set C.

Proving (3)*:* $R(C \mid A) \leq R(C \mid B)$. Equivalently, we wish to show that,

$$\sum_{i \in C} P(i \prec A \cup C) r_i^A \leq \sum_{i \in C} P(i \prec B \cup C) r_i^B.$$

From substitutability, we have $P(i \prec A \cup C) \leq P(i \prec B \cup C)$ for every $i \in C$. Hence, it suffices to show $r_i^A \leq r_i^B$, $\forall i \in C$. Consider the B-adjusted markov chain. From property (iv), we have $r_j^B \geq 0$ for every $j \in S$. Using property (ii),

$$r_i^A \;=\; r_i^B - \sum_{j \in A \backslash B} P_i(j \prec A) r_j^B \;\leq\; r_i^B.$$

Proving (2)*:* $R(A \mid C) \geq 0$. Given some element $e \in S$, we show that $R(e \mid C) \geq 0$ for every set $C \subseteq N\backslash\{e\}$. Applying this for every element in $A\backslash C$ proves the desired. From property (i), we have $R(e \mid C) \geq 0$ if and only if $r_e^C \geq 0$. Let $C_e = C + e$. Using property (ii),

$$\begin{aligned}
r_e^C &= r_e^{C \cap S} - \sum_{j \in C \backslash S} P_e(j \prec C) r_j^{C \cap S} \\
&= r_e^{C \cap S} - \sum_{j \in C \backslash S} P_e(j \prec C) \left(r_j^{C \cap S} - \sum_{i \in S \backslash C_e} P_j(i \prec S\backslash\{e\}) r_i^{C \cap S} \right) \\
&\quad - \sum_{j \in C \backslash S} \sum_{i \in S \backslash C_e} P_e(j \prec C) P_j(i \prec S\backslash\{e\}) r_i^{C \cap S} \\
&\geq r_e^{C \cap S} - \sum_{i \in S \backslash C_e} P_e(i \prec S\backslash\{e\}) r_i^{C \cap S} - \sum_{j \in C \backslash S} P_e(j \prec C) \left(r_j^{C \cap S} - \sum_{i \in S \backslash C_e} P_j(i \prec S\backslash\{e\}) r_i^{C \cap S} \right) \\
&= r_e^{S \backslash \{e\}} - \sum_{j \in C \backslash S} P_e(j \prec C) r_j^{S \backslash \{e\}} \qquad (12)
\end{aligned}$$

We note that the inequality in derivation above uses $r_i^{C \cap S} \geq 0$ for every $i \in S$. Now, we claim that,

$$r_e^{S \backslash \{e\}} \geq r_j^{S \backslash \{e\}}, \quad \forall j \in C \backslash S.$$

Substituting this into (12) then gives us the desired. For the sake of contradiction, let there be a $j \in C\backslash S$ such that $r_j^{S \backslash \{e\}} > r_e^{S \backslash \{e\}}$. From property (i), recall that $r_j^{S \backslash \{e\}} = 0$ for every $j \in S\backslash\{e\}$. Thus, we have $r_j^{S \backslash \{e\}} > \max_{i \in S} r_i^{S \backslash \{e\}}$. Applying property (iii) on the $S\backslash\{e\}$-adjusted markov chain,

$$R^{S \backslash \{e\}}(S + j) \geq R^{S \backslash \{e\}}(S).$$

Thus, $R(S \cup \{j\}) \geq R(S)$, contradicting the maximality of optimal assortment S. This completes the proof of Theorem 3.3.

D.4. Joint Pricing and Assortment Optimization under Markov Choice

As an application of matroid constrained optimization, we obtain the first non-trivial approximation for joint pricing and assortment optimization in the Markov model (see Appendix D.3 for a proof).

Corollary D.8. *There is a 0.25 approximation for joint pricing and assortment optimization in the Markov choice model.*

Proof. Given a ground set N, discrete price ladder $\{p_1, \ldots, p_r\}$, let $N_P = \{(i, p_j) \mid i \in N, p_j \in P\}$ denote an expanded universe of products. Given a Markov choice model over N_P, we can solve the joint pricing and assortment optimization

problem by solving an instance of matroid constrained assortment optimization on ground set N_P. Consider a partition $\{R_1, \cdots, R_n\}$ of N_P, where $R_i = \{(i, p_j) \mid p_j \in P\}$, $\forall i \in N$. Then, the partition matroid where a set S is independent if and only if, $\forall (i, p_j) \in S$ we have $(i, p_k) \notin S$, $\forall p_k \neq p_j$, enforces that each product can be included with at most one price. In fact, combining a cardinality constraint in addition to this partition matroid is still a matroid constraint. This completes the proof. □

D.5. Proof of Lemma D.7

We consider each type of constraint (and the corresponding algorithms) separately. The analysis in each case mimics the analysis for submodular order functions. We omit ϕ from subscripts and denote f_ϕ as simply f. Recall that f is monotone subadditive for substitutable choice models and compatible choice models are substitutable by definition. As shown in Part II of the analysis of Algorithm 8 (see Section D.3.1), for compatible choice models the function f exhibits piece-wise submodular order. For a given $\mathcal{A}_\gamma$, this order is characterized by sets $\{N_i, M_i\}_{i\in[p+1]}$ and ordering π as defined in Section D.3.1.

CARDINALITY CONSTRAINT

$\mathcal{A}$ = Algorithm 1 and parameter γ corresponds to threshold value τ in the algorithm. Let OPT denote both the optimal solution and function value. Notice that for $\text{OPT} \leq 2\max_{e\in N} f(\{e\})$, picking the largest value of τ gives a solution S_γ such that $f(S_\gamma) \geq (1-\epsilon)\,0.5\,\text{OPT}$. So from here on, let $\text{OPT} > 2\max_{e\in N} f(\{e\})$.

Now, fix γ such that $\tau \in [(1-\epsilon)\frac{\text{OPT}}{2k}, \frac{\text{OPT}}{2k}]$ (such a setting exists for $\text{OPT} > 2\max_{e\in N} f(\{e\})$). In the following, we drop γ in the notation for convenience. Let $\{N_i, M_i\}_{i\in[p+1]}$ and π represent the resulting piece wise order generated. Recall that $S = \cup_{i\in[p]} M_i$. We show that $f(S) \geq (1-\epsilon)\,0.5\,\text{OPT}$.

Let k' denote the cardinality of set S. Then, by definition of Threshold Add,

$$f(S) \geq k'\tau.$$

When $k' = k$ this gives us $f(S) \geq (1-\epsilon)\,0.5\,\text{OPT}$. So let $k' < k$. From monotonicity of the function we have, $\text{OPT} \leq f(\text{OPT} \cup S)$. So consider the union $\text{OPT} \cup S$ and its interleaved partition

$$\{o_1, \{s_1\}, o_2, \{s_2\}, \cdots, \{s_{k'}\}, o_{k'+1}\},$$

where every element of S is a singleton s_j for some j and every element of $\text{OPT}\backslash S$ is a singleton o_j for some j. Note that $\{s_1, \cdots, s_\ell\} \subseteq \cup_{i\in[p]} M_i$ for every $\ell \in [k']$. So we apply Lemma D.5 on $\text{OPT} \cup S$ with $E_j := s_j$ for $j \in [k']$, to obtain

$$\text{OPT} \quad \leq \quad f(S) + \sum_{\ell\in[k']} f\left(o_\ell \mid E(\ell-1)\right).$$

By definition of Threshold Add, we have $f\left(o_\ell \mid E(\ell-1)\right) \leq \tau$, for every $\ell \geq 1$. Plugging this into the above inequality and using the upper bound $\tau \leq 0.5\text{OPT}/k$ we get,

$$\text{OPT} \leq f(S) + k\tau \leq f(S) + 0.5\,\text{OPT}.$$

BUDGET CONSTRAINT

$\mathcal{A}$ = Algorithm 6 (the case of Algorithm 4 is similar to cardinality constraint). Parameter γ determines set X and threshold τ. We focus on $X \subseteq \text{OPT}$ that contains the $1/\epsilon$ largest budget elements in OPT (or all of OPT if its cardinality is small). Given this X, the budget b_e required by any $e \in \text{OPT}\backslash X$ is strictly smaller than ϵB, otherwise $b(X) > B$. Let $\tau \in \left[(1-\epsilon)\frac{\text{OPT}}{2B}, \frac{\text{OPT}}{2B}\right]$.

For the above setting of γ, let $\{N_i, M_i\}_{i\in[p+1]}$ and π represent the piece wise submodular order generated. Let $S \cup R \subseteq \cup_{i\in[p]} M_i$ denote the output of $\mathcal{A}_\gamma$ where S is the feasible set generated and R is the set of elements (if any) removed by Final Add subroutine. Consider the following cases based on how $\mathcal{A}_\gamma$ terminates.

Case I: Final Add is not invoked i.e., $R = \emptyset$. In this case elements are not removed after they are chosen and every element not in S fails the threshold requirement. Similar to the case of cardinality constraint, we have an interleaved partition

$$\{o_1, \{s_1\}, o_2, \{s_2\}, \cdots, \{s_k\}, o_{k+1}\},$$

where every element of S is a singleton s_j for some $j \in [k]$ and every element of OPT$\backslash S$ is a singleton o_j for some $j \in [k+1]$. Using Lemma D.5 with $E_j := s_j$, $\forall j \in [k]$,

$$\text{OPT} \leq f(S_i) + \sum_{j=1}^{k_i+1} f\left(o_j \mid E(j-1)\right).$$

Since $f\left(o_j \mid E(j-1)\right) \leq \tau$, $\forall j \in [k+1]$, we have ALG $\geq f(S) \geq 0.5\,$OPT.

Case II: Final Add is invoked for some element j. We claim that the final output S is such that (i) $b(S) \geq (1-\epsilon)\,B$ and (ii) $f(S) \geq \tau b(S)$. Using (i) and (ii), we have for $\epsilon \in [0,1]$,

$$f(S) \geq 0.5\,(1-\epsilon)^2\,\text{OPT} \geq (0.5-\epsilon)\,\text{OPT}.$$

It remains to show (i) and (ii). Let S_{in} denote the set that is input to Final Add and let $\hat{X} = \{e \mid b_e \geq \epsilon B, e \in S_{in} + \{j\}\}$. Observe that $\hat{X} \subseteq X$, since every element outside X with budget exceeding ϵB is discarded in the beginning. Therefore, $b(\hat{X}) \leq B$ and the set S returned by Final Add is feasible (and contains $\hat{X}$).

To see (i), let t denote the last element removed from S_{in} by Final Add. We have, $b_t < \epsilon B$ and $b_t + b(S) > B$. Thus, $b(S) \geq B - \epsilon B$.

To show (ii), recall that f is π submodular ordered on the ground set $M_0 := \cup_{i\in[p]} M_i$. So we use Lemma D.6 (ii) on set $S_{in} \subset S \cup R \subseteq M_0$. For simplicity, let $\{1, 2, \cdots, s\}$ denote the elements of S in submodular order. Recall that $S \subset S_{in}$. Using Lemma D.6 (ii) with $A = S_{in}$ and sets $E_k = \{k\}$ for $k \in [s]$, we have

$$\sum_{k\in[s]} f(k \mid L_k) \leq f(S),$$

where L_k denotes the set of all elements in S_{in} chosen by $\mathcal{A}_\gamma$ prior to k. From the threshold requirement, we have $f(k \mid L_k) \geq \tau b_k$, $\forall k \in [s]$. Thus, $f(S) \geq \tau b(S)$.

MATROID CONSTRAINT

$\mathcal{A}$ = Algorithm 3 and there are no parameters γ. Let S_j denote the set S maintained in the algorithm at the beginning of iteration j. Similarly, let R_j denote the set R at the beginning of iteration j. Observe that R_j is the set of all elements that were chosen and later swapped out, prior to iteration j. Therefore, the set $S_j \cup R_j$ grows monotonically and includes all elements selected by the algorithm prior to j. We use S and R to denote the final sets when the algorithm terminates. The sets $\{N_i, M_i\}_{i\in[p+1]}$ and order π denote the piece-wise order generated. Observe that $S \cup R = \cup_{i\in[p+1]} M_i$.

Let OPT denote the optimal set (and value). Similarly, we use ALG to denote both the value $f(S)$ and the set S output by the algorithm. By monotonicity, $f(\text{ALG} \cup \text{OPT}) \geq \text{OPT}$. We will "essentially" show that $f(\text{OPT} \mid \text{ALG}) \leq 3\text{ALG}$. The main elements that contribute to this upper bound are as discussed in the proof of Theorem 1.2 so we jump directly to the analysis.

W.l.o.g., ignore all elements in $N\backslash(\text{OPT} \cup S \cup R)$. So let $\hat{N} = \text{OPT} \cup S \cup R$ denote our ground set and re-index elements in $\hat{N}$ from 1 to $|\hat{N}|$ (maintaining submodular order). Similarly, we re-index the sets $\{S_j\}$ so that they continue to denote the set maintained by the algorithm when element $j \in \hat{N}$ is first parsed, for every $j \in \{1, 2, \cdots, |\hat{N}|\}$. From monotonicity, OPT $\leq f(\hat{N})$.

Consider an interleaved partition $\{o_\ell, e_\ell\}$ of $\hat{N}$ such that every element in $S \cup R$ is given by singleton e_ℓ for some $\ell \in [m]$ and every element in OPT$\backslash(S \cup R)$ is given by singleton o_ℓ for some $\ell \in [m]$. Using Lemma D.5,

$$\text{OPT} \leq f\left(S \cup R\right) + \sum_{j\in\text{OPT}\backslash(S\cup R)} f\left(j \mid S_j \cup R_j\right),$$

where $E(\ell) = \{e_1, \cdots, e_\ell\}$, $\forall \ell \in [m]$ and $E(m) = S \cup R$. Next, we upper bound $f(S \cup R)$ in terms of ALG. Consider an interleaved partition $\{\bar{o}_\ell, \bar{e}_\ell\}_{\ell\in[\bar{m}]}$ of $S \cup R$ such that elements of S are given by singletons $\bar{e}_\ell$ and singletons $\bar{o}_\ell$ represent elements of R. Since $S \cup R = \cup_{i\in[p+1]} M_i$, mwe apply Lemma D.6 (i) to obtain,

$$f\left(S \cup R\right) \;\leq\; \text{ALG} + \sum_{\ell\in[\bar{m}]} f\left(\bar{o}_\ell \mid \{\bar{o}_1, \bar{e}_1, \cdots, \bar{o}_{\ell-1}, \bar{e}_{\ell-1}\}\right) \;=\; \text{ALG} + \sum_{j\in R} f\left(j \mid S_j \cup R_j\right).$$

In the following, we upper bound $\sum_{j\in R} f\left(j \mid S_j \cup R_j\right)$ by ALG and $\sum_{j\in \text{OPT}\setminus(S\cup R)} f\left(j \mid S_j \cup R_j\right)$ by 2ALG. This proves the main claim.

Part I: $\sum_{j\in R} f\left(j \mid S_j \cup R_j\right) \leq \text{ALG}$.

Consider an element $j_1 \in R$ that was added to S_j without swapping out any element. Since $j_1 \in R$, there exists an element j_2 that replaced j. Inductively, for $t \geq 2$, let j_t denote the tth element in the chain of swaps $j_1 \to j_2 \to \cdots \to j_t$. The chain terminates at an element in S and we call this a *swap chain*. From the swap criteria in Algorithm 3, we have

$$\sum_{\tau\in[t-1]} f\left(j_\tau \mid S_{j_\tau} \cup R_{j_\tau}\right) \leq f\left(j_t \mid R_{j_t} \cup S_{j_t}\right).$$

Every element in R is part of a unique swap chain and each chain has a unique terminal element in S. Therefore,

$$\sum_{j\in R} f\left(j \mid S_j \cup R_j\right) \leq \sum_{i\in S} f\left(i \mid S_i \cup R_i\right).$$

Applying Lemma D.6 (ii) with $A = S \cup R$ and sets $E_i = \{i\}$ for $i \in S$, we have

$$\sum_{i\in S} f\left(i \mid S_i \cup R_i\right) \leq \text{ALG}. \tag{13}$$

Part II: $\sum_{j\in \text{OPT}\setminus(S\cup R)} f\left(j \mid S_j \cup R_j\right) \leq 2\text{ALG}$.

From Lemma B.5 we have that the rank $r\left(\text{OPT}\right) \leq r\left(\text{ALG}\right)$. Suppose there exists an injection ϕ from elements in $\text{OPT}\setminus(S \cup R)$ to elements in S such that

$$f\left(j \mid S_j \cup R_j\right) \leq 2f\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right), \quad \forall j \in \text{OPT}\setminus(S \cup R).$$

Summing up these inequalities and using (13), we are done. It remains to show that ϕ exists. We do this via a graphical construction inspired by (Chekuri et al., 2015).

Consider a graph G with vertices given by $\hat{N}$. In order to define the edges recall that every $j \in \text{OPT}\setminus(S \cup R)$ is rejected by the algorithm on parsing. Thus, we have a unique circuit C_j where $C_j\setminus\{j\} \subseteq S_j$. For our first set of edges we make a directed edge from j to every element in $C_j\setminus\{j\}$ and we do this for all $j \in \text{OPT}\setminus(S \cup R)$. Next, for every $j \in R$, let C_j represent the chain that causes j to be swapped out in the algorithm. We create a directed edge from j to every element in $C_j\setminus\{j\}$. Graph G has the following properties,

(a) The elements of $\text{OPT}\setminus(S \cup R)$ are *source* vertices with no incoming edges. Elements of S are *sinks* with no outgoing edges.

(b) The neighbors of every node in G form a circuit with the node.

(c) Given arbitrary vertex $j \in \text{OPT}\setminus(S \cup R)$, for every i reachable from j we claim that

$$f(j \mid S_j \cup R_j) \leq v_i,$$

where v_i corresponds to the value defined in Algorithm 3. For neighbors of j the claim follows directly from the swap criterion. Also, for any two neighboring vertices $i', i'' \in S \cup R$, we have $v_{i'} \leq v_{i''}$. The general claim follows by using these inequalities for every edge on the path from j to i.

Using (a) and (b) we apply Lemma B.6 to obtain an injection ϕ from $\text{OPT}\setminus(S\cup R)$ to S such that for every $j \in \text{OPT}\setminus(S\cup R)$ there exists a path to $\phi(j) \in S$. Then, from (c) we have, $f(j \mid S_j \cup R_j) \leq v_{\phi(j)}$. Recall the notion of a swap chain defined in Part I and let $W(\phi(j))$ denote the set of all preceding elements of the swap chain that terminates at $\phi(j)$. By definition of $v_{\phi(j)}$ and the swap criterion,

$$v_{\phi(j)} = f\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right) + \sum_{i\in W(\phi(j))} f\left(i \mid S_i \cup R_i\right) \leq 2f\left(\phi(j) \mid S_{\phi(j)} \cup R_{\phi(j)}\right).$$