Robust Monotone Submodular Function Maximization

James B. Orlin · Andreas S. Schulz · Rajan Udwani

Abstract We consider a robust formulation, introduced by Krause et al. (2008), of the classical cardinality constrained monotone submodular function maximization problem, and give the first constant factor approximation results. The robustness considered is w.r.t. adversarial removal of up to τ elements from the chosen set. For the fundamental case of $\tau = 1$, we give a deterministic $(1 - 1/e) - 1/\Theta(m)$ approximation algorithm, where m is an input parameter and number of queries scale as $O(n^{m+1})$. In the process, we develop a deterministic $(1-1/e)-1/\Theta(m)$ approximate greedy algorithm for bi-objective maximization of (two) monotone submodular functions. Generalizing the ideas and using a result from Chekuri et al. (2010), we show a randomized $(1 - 1/e) - \epsilon$ approximation for constant τ and $\epsilon \leq \frac{1}{\tilde{\Omega}(\tau)}$, making $O(n^{1/\epsilon^3})$ queries. Further, for $\tau \ll \sqrt{k}$, we give a fast and practical 0.387 algorithm. Finally, we also give a black box result result for the much more general setting of robust maximization subject to an Independence System.

1 Introduction

A set function $f : 2^N \to \mathbb{R}$ on the ground set N is called submodular if,

$$f(A + a) - f(A) \leq f(B + a) - f(B) \text{ for all } B \subseteq A \subseteq N \text{ and } a \in N \setminus A.$$

The function is monotone if $f(B) \leq f(A)$ for all $B \subseteq A$. We also impose $f(\emptyset) = 0$, which combined with monotonicity implies non-negativity. Optimization problems with submodular objective functions have received a lot of interest due to several applications where instances of these problems arise naturally. However, unlike the (unconstrained) minimization of submodular functions, for which polytime algorithms exist [20, 27], even the simplest maximization versions are NP-hard [11–13, 31]. In fact, they encompass several fundamental hard problems, such as max-cut, max-k-coverage, max-dicut and variations of max-SAT and max-welfare.

{jorlin,schulz,rudwani@mit.edu}

A long line of beautiful work has culminated in fast and tight approximation algorithms for many settings of the problem. As an example, for unconstrained maximization of non-monotone submodular functions, Feige et al. in [13], provided an algorithm with approximation ratio of 0.4 and showed an inapproximability threshold of 1/2 in the value-oracle model. Extensions by Gharan and Vondrák [16] and subsequently by Feldman et al. [15] led to further improvement of the guarantee (roughly 0.41 and 0.42, respectively). Finally, Buchbinder et al. in [8] gave a tight randomized 1/2 approximation algorithm, and this was recently derandomized [7].

Here we are interested in the problem of maximizing a monotone submodular function subject to a cardinality constraint, written as: $P1 := \max_{A \subseteq N, |A| \leq k} f(A)$. The problem has been well studied and instances of $P1$ arise in several important applications, two of them being:

Sensor Placement [19, 21–23]: Given a large number of locations N, we would like to place up to k sensors at certain locations so as to maximize the *coverage*. Many commonly used coverage functions measure the cumulative information gathered in some form, and are thus monotone (more sensors is better) and submodular (decreasing marginal benefit of adding a new sensor).

However, as highlighted in [22], it is important to ask what happens if some sensors were to fail. Will the remaining sensors have good coverage regardless of which sensors failed, or is a small crucial subset responsible for most of the coverage?

Feature Selection [17,22,24,29]: In many machine learning models, adding a new feature to an existing set of features always improves the modeling power (monotonicity) and the marginal benefit of adding a new feature decreases as we consider larger sets (submodularity). Given a large set of features, we would like to select a small subset such that, we reduce the problem dimensionality while retaining most of the information.

However, as discussed in [17, 22], in situations where the nature of underlying data is uncertain, leading to non-stationary feature distributions, it is important to not have too much dependence on a few features. Taking a concrete example from [17], in document classification, features may take not standard values due to small sample effects or in fact, the test and training data may come from different distributions. In other cases, a feature may even be deleted at some point, due to input sensors failures for example. Thus, similar questions arise here too and we would like to have an 'evenly spread' dependence on the set of chosen features. With such issues in mind, consider the following *robust* variant of the problem, introduced in [22],

$$P2 := \max_{A \subseteq N, |A| \leq k} \quad \min_{Z \subseteq A, |Z| \leq \tau} f(A - Z).$$

Note that the parameter τ controls the degree of robustness of the chosen set since the larger τ is, the larger the size of subset Z that can be adversarially removed from the chosen set A. For $\tau = 0$, $P2$ reduces to the $P1$. Since this formulation optimizes the worst case scenario, a natural variation is to optimize the average case failure scenario [18]. However, this is not suitable for some applications. For instance, we may have no prior on the failure/deletion mechanism and furthermore, in critical applications, such as sensor placement for outbreak detection [22,23], we want protection against the worst case. This form of worst case analysis has been of great interest in operations research and beyond, under the umbrella of *robust optimization* (e.g., [2–5]). The idea is to formulate the uncertainty in model parameters through a deterministic *uncertainty set*. While much work in this area assumes that the uncertainty set is a connected, if not convex set, the uncertainty set in $P2$, when $\tau = 1$ for instance, is the disconnected set of canonical unit vectors $e_i \in \mathbb{R}^N$ (1 at entry i, 0 otherwise).

Previous work on $P1$ and $P2$. The first rigorous analysis of $P1$ was by Nemhauser et al. [25, 26] in the late 70's, where they showed that the greedy algorithm gives a guarantee of $(1 - 1/e)$ and that this is best possible in the value-oracle model. Later, Feige [12] showed that this is also the best possible under standard complexity assumptions (through the special case of Max-k-cover). On the algorithmic side, Badanidiyuru and Vondrák [1] recently gave a faster algorithm for $P1$ that improved the quadratic query complexity of the classical greedy algorithm to nearly linear complexity, by trading off on the approximation guarantee. However, the optimality (w.r.t. approximation guarantee) of the greedy algorithm is rather coincidental, and for many complex settings of the problem (monotone or not), the greedy algorithm tends to be sub-optimal (there are exceptions, like [28]). An approach first explored by Calinescu et al. [9], that has been very effective, is to perform optimization on the *multilinear extension* of the submodular function, followed by clever *rounding* to get a solution to the original problem. Based on this framework, tremendous progress has been made over the last decade for both monotone and non-monotone versions with various kinds of constraints [9,10,14,30–32]. In fact, a general framework for establishing hardness of many of these variants [11,31], also relies intricately on properties of this relaxation.

Moving on to $P2$, as we will see, the well known greedy algorithm and also the above mentioned *continuous* greedy approach for $\tau = 0$, can be arbitrarily bad even for $\tau = 1$. In fact, many natural approaches do not have a constant factor guarantee for $\tau \geq 1$. The paper by Krause et al. [22], which formally introduced the problem, actually gives a bi-criterion approximation to the much more general and inapproximable problem: $\max\limits_{A \subseteq N, |A| \leq k} \min\limits_{i \in \{1,2,\ldots,m\}} f_i(A)$, where $f_i(.)$ is monotone submodular for every i. Their algorithm, which is based on the greedy algorithm for $P1$, when specialized to $P2$, guarantees optimality by allowing sets up to size $k(1 + \Theta(\log(\tau k \log n)))$ and has runtime exponential in τ. To the best of our knowledge, no stronger/constant factor approximation results were known for $P2$ prior to our work.

Our contributions: We work in the value oracle model and give constant factor guarantees for $P2$ with combinatorial, 'greedy like' algorithms. To ease presentation, we will usually ignore factors of the form $\left(1 - \frac{O(1)}{k}\right)$ in the approximation guarantees and thus, most of the results presented here are asymptotic in k. Our initial focus is on a restricted case, where we construct relatively simple algorithms and get insights that generalize. For this special case, we give a $(1 - 1/e)$ algorithm for $\tau = o(k)$.

In the non-restricted setting, we first focus on generalizing results for $\tau = 1$, for which we propose a fast and practical 0.5547 approximation and later an asymptotically $(1 - 1/e) - 1/\Theta(m)$ algorithm (with runtime exponential in m, which is an input parameter). This relies on developing a new $(1 - 1/e) - 1/\Theta(m)$ approximate greedy algorithm for the bi-objective maximization of monotone submodular functions subject to cardinality constraint, of which the problem of maximizing the minimum of two arbitrary monotone submodular functions is a special case. We conjecture that the greedy algorithm can be generalized to work for multi-objective maximization of a fixed number of monotone submodular functions. There has been previous work on the multi-objective problem for a constant number of monotone submodular functions and a randomized $(1 - 1/e) - \epsilon$ algorithm for (a much more general version of) this problem was given in [10]. It uses the continuous greedy algorithm from [32], along with an innovative dependent rounding scheme called *swap rounding*, and we use this algorithm as a subroutine to get a randomized $(1-1/e)-\epsilon$ approximation for the robust problem when τ is constant, however the runtime scales as $O(n^{1/\epsilon^3})$. Finally, for $\tau = o\big(\sqrt{\frac{k}{c(k)}}\big)$, we give a fast $0.387\big(1 - \frac{1}{\Theta(c(k))}\big)$ algorithm where $c(k) \xrightarrow{k\to\infty} \infty$ is an input parameter that governs the trade off between how large τ can be and how fast the guarantee converges to 0.387.

In the more general case, where we wish to find a robust set A in an independence system, we extend some of the ideas from the cardinality case into an enumerative procedure that yields an $\alpha/(\tau + 1)$ approximation using an α approximation algorithm for $\tau = 0$ as a subroutine. However, the runtime scales as $n^{\tau+1}$.

The outline for the rest of the paper is as follows: In Section 2, we introduce some notation and see how several natural ideas fail to give any approximation guarantees. In Section 3, we start with a special case and slowly build up to an algorithm with asymptotic guarantee $(1-1/e)-\epsilon$ for constant τ, covering the other results in the process. Finally, in Section 4, we extend some of the ideas to more general constraints. Section 5 concludes with some open questions.

2 Preliminaries

2.1 Definitions

We denote an instance of $P2$ on ground set N with cardinality constraint parameter k and robustness parameter τ by (k, N, τ). Subsequently, we use $OPT(k, N, \tau)$ to denote an optimal set for the instance (k, N, τ). For any given set A, we call a subset Z_τ a *minimizer* if $f(A - Z_\tau) = \min_{B\subseteq A;|B|=\tau} f(A - B)$.

Also, let $\mathcal{Z}_\tau(A)$ be the set of minimizers of A. When $\tau = 1$, we often use the letter z for minimizers and when τ is otherwise clear from the context and fixed during the discussion we use the shorthand $Z, \mathcal{Z}(.)$. Based on this we also introduce a key function $g_\tau(A) = f(A - Z_\tau)$. Again, we simply use $g(.)$, when τ is clear from context. We generally refer to singleton sets without braces $\{\}$ and use $+, -$ and $\cup, \setminus$ interchangeably. Also, define the marginal increase in value due to a set X, when added to the set A as $f(X|A) = f(A \cup X) - f(A)$. Similarly, $g(X|A) = g(A \cup X) - g(A)$.

Let $\beta(\eta, \alpha) = \frac{e^\alpha - 1}{e^\alpha - \eta} \in [0,1]$ for $\eta \in [0,1]$, $\alpha \geq 0$. Note that $\beta(0,1) = (1 - 1/e)$. This function appears naturally in our analysis and will be useful for expressing approximation guarantees of the algorithms. Next, recall the widely popular greedy algorithm for $P1$:

Algorithm 1 Greedy Algorithm

```
1: Initialize $A = \emptyset$
2: **while** $|A| < k$ **do** $A = A + \underset{x \in N - A}{\mathrm{argmax}} f(x|A)$
3: Output: $A$
```

Let $k = n$ in the above and denote the i-th element added by a_i. Using this we index the elements in N in the order they were added, so $N = \{a_1, a_2, \ldots, a_n\}$.

Also, recall the following theorem:

Lemma 1 (Nemhauser, Wolsey [25, 26]) *For all $\alpha \geq 0$, greedy algorithm terminated after αk steps yields a set A with $f(A) \geq \beta(0, \alpha) f(OPT(k, N, 0))$.*

For the sake of completeness, we include the proof in Appendix A.1.

In addition, the following lemma, which compares the optimal value of (k, N, τ) with $(k - \tau, N, 0)$ will be very useful:

Lemma 2 *For instances (k, N, τ), we have:*

$$g_\tau(OPT(k, N, \tau)) \leq f(OPT(k - \tau, N - X, 0)) \leq f(OPT(k - \tau, N, 0)),$$

for all $X \subseteq N, |X| \leq \tau$.

Proof We focus on the first inequality since the second follows by definition. Let $x = \big|X \cap OPT(k, N, \tau)\big| \leq \tau$, then note that $g_\tau(OPT(k, N, \tau)) \leq g_{\tau - x}(OPT(k, N, \tau) - X)$, since the RHS represents the value of some subset of $OPT(k, N, \tau)$ of size $k - \tau$, which upper bounds the LHS by definition. Now, $g_{\tau-x}(OPT(k, N, \tau) - X) \leq g_{\tau - x}(OPT(k - x, N - X, \tau - x))$ by definition. Finally, note that $g_{\tau-x}(OPT(k - x, N - X, \tau - x)) \leq f(OPT(k - \tau, N - X, 0))$ since the LHS represents the value of a set of size $k - x - (\tau - x) = k - \tau$ that does not include any element in X, giving us the desired. $\square$

Finally, it is natural to expect that we cannot approximate $P2$ better than $P1$ (which is approximable up to a factor of $\beta(0,1)$) and this is indeed true, as stated below. The proof is included in Appendix A.2.

Lemma 3 *There exists no polytime algorithm with approximation ratio greater than $(1 - 1/e)$ for $P2$ unless $P = NP$. For the value oracle model, we have the same threshold, but for algorithms that make only a polynomial number of queries.*

2.2 Negative Results

The example below demonstrates why the greedy algorithm that does well for instances of $P1$, fails for $P2$. However, the weakness will also indicate a property which will guide us towards better guarantees later.

Example: Consider a ground set N of size $2k$ such that $f(a_1) = 1$, $f(a_i) = 0$, $\forall\, 2 \leq i \leq k$ and $f(a_j) = \frac{1}{k}$, $\forall j \geq k+1$. Also, for all $j \geq k+1$, let $f(a_j|X) = \frac{1}{k}$ if $X \cap \{a_1, a_j\} = \emptyset$ and 0 otherwise. Consider the set $S = \{a_{k+1}, \cdots, a_{2k}\}$ and let the set picked by the greedy algorithm (with arbitrary tie-breaking) be $A = \{a_1, \cdots, a_k\}$. Then we have that $f(A - a_1) = 0$ and $f(S - a_j) = 1 - \frac{1}{k}$ for every $a_j \in S$. The insight here is that greedy may select a set where only the first few elements contribute most of the value in the set, which makes it non-robust. However, as we discuss more formally later, such a concentration of value implies that only the first two elements, $\{a_1, a_2\}$, are critical and protecting against removal of those suffices for best possible guarantees.

In fact, many natural variations fail to give an approximation ratio better than $1/(k-1)$. Indeed, a guarantee of this order, i.e. $1/(k-\tau)$, is achievable for any τ by the following naïve algorithm: *Pick the k largest value elements.* It is also important to examine if the function g is super/sub-modular, since that would make existing techniques useful. It turns out not. However, it is monotonic. Despite this, it is interesting to examine a natural variant of the greedy algorithm, where we greedily pick w.r.t g, but that variant can also be arbitrarily bad if we pick just one element at each iteration.

3 Main Results

Before we start, remember that the focus of these results is on asymptotic performance guarantee (for large k). In some cases, the results can be improved for small k but we generally ignore these details.

Additionally, in every algorithm that uses the greedy algorithm (Algorithm 1) as a subroutine, especially the fast and practical algorithms 2 ,4 and 7, we can replace the greedy addition rule of adding $x = \arg\max_{x \in S_1} f(x|S_2)$ for some S_1, S_2, by the more efficient thresholding rule in [1], where, given a threshold w, we add a new element $x \in S_1$ if $f(x|S_2) \geq w$. This improves the query/run time to $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$, at the cost of a factor of $(1-\epsilon)$ in the guarantee.

3.1 Special case of "copies"

We first consider a special case, which will serve two purposes. First, it will simplify things and the insights gained for this case will generalize well. Secondly, since this case may arise in practical scenarios, it is worthwhile to discuss the special algorithms as they are much simpler than the general algorithms discussed later.

Given an element $x \in N$, we call another element x' a *copy* of element x if,

$$f(x') = f(x) \text{ and } f(x'|x) = 0.$$

This implies $f(x|x') = f(\{x, x'\}) - f(x') = f(x) + f(x'|x) - f(x') = 0$. In fact, more generally, $f(x'|A) = f(x|A)$ for every $A \subseteq N$, since $f(A + \{x, x'\}) = f(A + x) = f(A + x')$. *This is a useful case for robust sensor placement, if we were allowed to place/replicate multiple sensors at certain locations that are critical for coverage.* Assume that each element in N has τ copies. In the next section we discuss algorithms for this special case when $\tau = 1$. This will help build a foundation, even though the results therein are superseded by the result for $\tau = o(k)$.

3.1.1 Algorithms for $\tau = 1$ in presence of "copies"

Let a'_i denote the copy of element a_i. As briefly indicated previously, we would like to make our set robust to removal of critical elements. In the presence of copies, adding a copy of these elements achieves that. So as a first step, consider a set that includes a copy of each element, and so is unaffected by single element removal. One way to do this is to run the greedy algorithm for $k/2$ iterations and then add a copy of each of the $k/2$ elements. Then, it follows from Lemma 1 that $g(S) = f(S) \geq \beta(0, \frac{k/2}{k-1}) f(OPT(k-1, N, 0))$ and then from Lemma 2 we have, $g(S) \geq (1 - \frac{1}{\sqrt{e}})\, g(OPT(k, N, 1))$, where we use the fact that $\beta(0, \frac{k/2}{k-1}) > \beta(0, 0.5) = (1 - \frac{1}{\sqrt{e}})$. Hence, we have ≈ 0.393-approximation and the bound is tight. We can certainly improve upon this, one way to do better is to think about whether we really need to copy all $k/2$ elements. It turns out that copying just a_1 and a_2 is enough! Intuitively, if the greedy set has value nicely spread out, we could get away without copying anything but nevertheless, in such a case copying just two elements does not affect the value much. Otherwise, as in the example from Section 2, if greedy concentrates its value on the first few elements, then copying them is enough.

Before we state and prove this formally, consider the below corollary:

Corollary 4 *Let A be the final set obtained by running the greedy algorithm for l steps on an initial set S. Then we have,*

$$f(A - S|S) \geq \beta\Big(0, \frac{l}{k}\Big) f(OPT(k, N, 0)|S) \geq \beta\Big(0, \frac{l}{k}\Big) \Big(f\big(OPT(k, N, 0)\big) - f(S)\Big)$$

Proof Follows from Lemma 1, since $f(.|S)$ is monotone submodular for any fixed set S and union of greedy on $f(.|S)$ with S is the same as doing greedy on $f(.)$ starting with S. $\square$

Suppose A, referred to as the greedy set, is the output of Algorithm 1. We now state and prove a simple yet key lemma, which will allow us to quantify how the ratio $\frac{f(A)}{f(OPT(k,N,0))}$ improves over $(1 - 1/e)$, as a function of how concentrated the value of the greedy set is on the first few elements.

Lemma 5 *Starting with initial set S, run l iterations of the greedy algorithm and let A be the output (so $|A| = l + |S|$). Then given $f(S) \geq cf(A)$ for some $c \leq 1$ (high concentration of value on S implies large c), we have,*

$$f(A) \geq \beta\Big(c, \frac{l}{k}\Big) f(OPT(k, N, 0)), \text{ for all } k \geq 1$$

In a typical application of this lemma, we will have S be the first $s = |S|$ elements of the greedy algorithm on $\emptyset$ and $c = s\eta$ for some $\eta \leq 1/s$. Additionally, $|A|$ can be greater than k.

Proof Denote $f(OPT(k, N, 0))$ by OPT and with the sets A, S as defined above, let $\delta = \frac{f(A)}{OPT}$, which implies $f(S) \geq c\delta OPT$ (by assumption). Also, since we allow $|A|$ to be larger than k, assume $\delta \leq 1$ (otherwise statement is true by default).

Now from Corollary 4 we have, $f(A) = f(S) + f(A - S|S) \geq f(S) + \beta(0, \frac{l}{k})(OPT - f(S))$ and thus,

$$\begin{aligned}
\delta OPT \geq (1 - \beta(0, \tfrac{l}{k})) \times c\delta f(OPT) + \beta(0, \tfrac{l}{k}) f(OPT) \quad &\text{[substitution]}\\
\delta(1 - c(1 - \beta(0, \tfrac{l}{k}))) &\geq \beta(0, \tfrac{l}{k})\\
\delta(1 - c\tfrac{1}{e^{l/k}}) &\geq \tfrac{e^{l/k}-1}{e^{l/k}}\\
\delta &\geq \tfrac{e^{l/k}-1}{e^{l/k}-c} = \beta\Big(c, \tfrac{l}{k}\Big)\\
\implies f(A) &\geq \beta\Big(c, \tfrac{l}{k}\Big) f(OPT(k, N, 0)). \quad \square
\end{aligned}$$

Let us understand the above lemma with a quick example. Consider the greedy set of the first k elements $A = \{a_1, \ldots, a_k\}$ and let $f(OPT(k, N, 0)) = 1$. Now, clearly $f(a_1) \geq \frac{f(A)}{k}$. So using the lemma with $S = \{a_1\}$ and $c = 1/k$ gives us simply that $f(A) \geq \beta\Big(\frac{1}{k}, \frac{k-1}{k}\Big) \approx (1 - 1/e)$, as expected. Next, if $f(a_1) \geq f(A)/2$, then we have that $f(A) \geq \beta\Big(\frac{1}{2}, \frac{k-1}{k}\Big) \approx 0.77$, asymptotically much better than $(1 - 1/e) \approx 0.63$. Similarly, if $f(\{a_1, a_2\}) \geq f(A)/2$ we again have $f(A) \geq \beta\Big(\frac{1}{2}, \frac{k-2}{k}\Big) \approx 0.77$. Additionally, we could compare the value $f(A)$ to $f(OPT(k-1, N, 0))$ instead, and then replacing k by $k-1$ in the denominator, we have $f(A) \geq \beta\Big(\frac{1}{2}, \frac{k-2}{k-1}\Big) f(OPT(k-1, N, 0))$.

Now, consider the algorithm that copies the first two elements and thus outputs: $\{a_1, a_1', a_2, a_2', a_3, \ldots, a_{k-2}\}$. Call this the 2-Copy algorithm. Using the above lemma, we show that this algorithm gives us the best possible guarantee asymptotically, aligning with the intuitive argument we presented earlier. Preceding the actual analysis, we show an elementary lemma that we will use frequently.

Lemma 6 *For $0 \leq \eta \leq \frac{1}{3}$ and arbitrary α, $\underset{\eta}{argmin}(1 - \eta)\beta(3\eta, \alpha) = 0$.*

Proof

$$\begin{aligned}(1-\eta)\beta(3\eta,\alpha) &= \frac{1}{3}(3-3\eta)\frac{e^\alpha-1}{e^\alpha-3\eta} = \frac{1}{3}(e^\alpha-1)\Big(1+\frac{3-e^\alpha}{e^\alpha-3\eta}\Big)\\ &\geq \frac{1}{3}(e^\alpha-1)\Big(1+\frac{3-e^\alpha}{e^\alpha}\Big) = (1-1/e^\alpha) = \beta(0,\alpha)\end{aligned}$$

Theorem 7 *For the case with copies, 2-Copy is* $\beta\big(0,\frac{k-5}{k-1}\big) \xrightarrow{k\to\infty} (1-1/e)$ *approximate.*

Proof First, denote the output as $A=\{a_1,a_1',a_2,a_2',a_3,\ldots,a_{k-2}\}$ and notice that $f(A)=f(\{a_1,a_2,\ldots,a_{k-2}\})$. As a warm-up, using Lemma 1 we get,

$$f(A)\geq\beta\Big(0,\frac{k-2}{k-1}\Big)f(OPT(k-1,N,0))\geq\beta\Big(0,\frac{k-2}{k-1}\Big)g(OPT(k,N,1)),(1)$$

where the second inequality follows from Lemma 2.

Let z be a minimizer of A. We can assume that $z\notin\{a_1,a_1',a_2,a_2'\}$ since all of these have 0 marginal. Now let $f(z|A-z)=\eta f(A)$. We have due to greedy additions and submodularity, $f(a_3|\{a_1,a_2\})\geq f(z|\{a_1,a_2\})\geq f(z|A-z)$ and similarly, $f(\{a_1,a_2\})\geq 2f(z|A-z)$. This implies that $f(\{a_1,a_2,a_3\})\geq 3\eta f(A)$, which relates the value removed by a minimizer z to the value concentrated on the first 3 elements $\{a_1,a_2,a_3\}$. The higher the value removed, the higher the concentration and closer the value of $f(A)$ to $f(OPT(k-1,N,0))$. More formally, with $S=\{a_1,a_1',a_2,a_2',a_3\}$, k replaced by $k-1$, $l=k-5$ and $c=3\eta$ ($\eta\leq 1/3$), we have from Lemma 5,

$$f(A)\geq\beta\Big(3\eta,\frac{k-5}{k-1}\Big)f(OPT(k-1,N,0))\geq\beta\Big(3\eta,\frac{k-5}{k-1}\Big)g(OPT(k,N,1)).$$

Which implies that $g(A)=(1-\eta)f(A)\geq(1-\eta)\beta(3\eta,\frac{k-5}{k-1})g(OPT(k,N,1))$. Applying Lemma 6 with $\alpha=\frac{k-5}{k-1}$ finishes the proof. As an example, for $k\geq 55$ the value of the ratio is ≥ 0.6. Additionally, we can use more precise bounds of the greedy algorithm for small k to get better guarantees in that regime. □

The result also implies that for large k, if the output set A, of Algorithm 1 (the greedy algorithm), has a minimizer a_i with $i\geq 3$, then $g(A)=f(A-a_i)\geq(1-1/e)g(OPT(k,N,1))$, i.e. the greedy algorithm is $(1-1/e)$ approximate for the robust problem in this situation. This is because, for such an instance we can assume that the set contains copies of a_1,a_2 without changing anything and then Theorem 7 applies.

Moreover, if instead of just the first two, we copy the first i elements for $i\geq 3$, we get the same guarantee but with worse asymptotics, so copying more than first two does not result in a gain. On the other hand, copying just one element, a_1, gives a tight guarantee of 0.5 (proof omitted).

Since $(1-1/e)$ is the best possible guarantee achievable asymptotically, we now shift focus to case of $\tau>1$ but $\ll k$, and generalize the above ideas to get an asymptotically $(1-1/e)$ approximate algorithm in presence of copies.

3.1.2 $(1-1/e)$ *Algorithms for* $\tau = o(k)$ *in the presence of "copies"*

Assume we have τ copies available for each $a_i \in N$. As we did for $\tau = 1$, we would like to determine a small critical set of elements, copy them (possibly many times) and then add the rest of the elements greedily to get a set of size k. In order to understand how large the critical set should be, recall that in the proof of Theorem 7, we relied on the fact that $f(\{a_1, a_2\})$ is at least twice as much as the value removed by the minimizer, and then we could use Lemma 5 to get the desired ratio. To get a similar concentration result on the first few elements for larger τ, we start with an initial set of size 2τ and in particular, we can start with $A_{2\tau} = \{a_1, a_2, \ldots, a_{2\tau}\}$. Additionally, similar to the 2-Copy algorithm, we also want the set to be unaffected by removal of up to τ elements from $A_{2\tau}$. We do this by adding τ copies of each element in $A_{2\tau}$. More concretely, consider the algorithm that greedily picks the set $A_{k-2\tau^2} = \{a_1, a_2, \ldots, a_{k-2\tau^2}\}$, and copies each element in $A_{2\tau} = \{a_1, a_2, \ldots, a_{2\tau}\} \subseteq A_{k-2\tau^2}$, τ times. Denote the set of copies by $C(A_{2\tau})$ and we have $|C(A_{2\tau})| = 2\tau^2$. To summarize, the algorithm outputs the set

$$A = A_{2\tau} \cup C(A_{2\tau}) \cup \{a_{2\tau+1}, \ldots, a_{k-2\tau^2}\}.$$

Observe that when $\tau = 1$, this coincides with the 2-Copy algorithm.

We next show that this algorithm is $\beta(0, \frac{k-2\tau^2-3\tau}{k-\tau}) \xrightarrow{k\to\infty} (1-1/e)$ approximate.

Proof We can assume that $Z \cap (A_{2\tau} \cup C(A_{2\tau})) = \emptyset$ or alternatively $Z \subseteq A_{k-2\tau^2} - A_{2\tau}$, since there are $\tau + 1$ copies of every element (counting the element itself) and Z cannot remove all. Recall that in the analysis of the 2-Copy algorithm, we showed $f(\{a_1, a_2, a_3\}) \geq 3f(z|A - z)$ and then used Lemmas 5 and 6 to get the desired. Analogously, here we would like to show and $f(A_{3\tau}) \geq 3f(Z|A - Z)$ and do the same.

Next, index elements in Z from 1 to τ, with the mapping $\pi : \{1, \ldots, \tau\} \to \{2\tau+1, \ldots, k-2\tau^2\}$, such that $\pi(i) > \pi(j)$ for $i > j$ and $a_{\pi(i)}$ is the i-th element in Z. Then with $A_i = \{a_1, \ldots, a_i\}$ for all i, we have by submodularity, the following set of inequalities,

$$\begin{aligned} f(a_{\pi(i)}|A_{\pi(i)-1}) &= f(a_{\pi(i)}|A - \{a_{\pi(i)}, \ldots, a_{k-2\tau^2}\}) \\ &\geq f(a_{\pi(i)}|A - \{a_{\pi(i)}, a_{\pi(i+1)}, \ldots, a_{\pi(\tau)}\}) \quad \forall i \\ \implies \sum_{i=1}^{\tau} f(a_{\pi(i)}|A_{\pi(i)-1}) &\geq f(Z|A - Z) \end{aligned} \tag{2}$$

where the RHS in (2) is by definition.

Note that $\pi(i) > 2\tau$, and for arbitrary $i \in \{1, \ldots, \tau\} = [\tau]$, $j \in \{1, \ldots, 2\tau\} = [2\tau]$, due to greedy iterations we have that $f(a_{\pi(i)}|A_{\pi(i)-1}) \leq f(a_j|A_{j-1})$. So consider any injective mapping from $i \in [\tau]$ to 2 distinct elements $i_1, i_2 \in [2\tau]$, for instance $i_1 = i, i_2 = i + \tau$. We rewrite the previous inequality as,

$$2f(a_{\pi(i)}|A_{\pi(i)-1}) \leq f(a_{i_1}|A_{i_1-1}) + f(a_{i_2}|A_{i_2-1})$$

Summing over all i, along with (2) above gives,

$$2f(Z|A-Z) \leq f(A_{2\tau}) \tag{3}$$

where the RHS is by injective nature of mapping and definition of $f(.|.)$.

In fact, we have $\pi(i) \geq 2\tau+i$ and thus, $f(a_{\pi(i)}|A_{\pi(i)-1}) \leq f(a_{2\tau+i}|A_{2\tau+i-1})$. From this, we have for the set $A_{3\tau} - A_{2\tau} = \{a_{2\tau+1}, \ldots, a_{3\tau}\}$ that,

$$f(Z|A-Z) \leq f(A_{3\tau} - A_{2\tau}|A_{2\tau}) \tag{4}$$

(3) and (4) combined give us,

$$f(A_{3\tau}) \geq 3f(Z|A-Z)$$

We have from Lemma 2 that $f(OPT(k-\tau, N, 0)) \geq g(OPT(k, N, \tau))$ and combined with using Lemma 5, with k replaced by $k-\tau$, $S = A_{3\tau} \cup C(A_{2\tau})$, $s = 3$ and $l = k - |S| = k - 2\tau^2 - 3\tau$ and $f(Z|A-Z) = \eta f(A)$ gives us,

$$f(A) \geq \beta\Big(3\eta, \frac{k-2\tau^2-3\tau}{k-\tau}\Big) g(OPT(k, N, \tau)).$$

Thus $g(A) = (1-\eta)f(A) \geq \beta\Big(0, \frac{k-2\tau^2-3\tau}{k-\tau}\Big) g(OPT(k, N, \tau))$, follows using Lemma 6. $\square$

Note that while we could ignore the asymptotic factors in approximation guarantee for $\tau = 1$, here we cannot, and this is where the upper bound on τ comes in. Recall that we compare the value $g(OPT(k, N, \tau))$, which is the $f(.)$ value of a set of size $k - \tau$, to the $f(.)$ value of a set of size $k - \Theta(\tau^2)$ (since the $\Theta(\tau^2)$ elements added as copies do not contribute any real value). Now $\frac{k-\Theta(\tau^2)}{k}$ converges to 1 only for $\tau \ll \sqrt{k}$ and it is this degradation that creates the threshold of $o(\sqrt{k})$.

However, it turns out that we don't need to add τ copies of each element in $A_{2\tau}$. Intuitively, the first few elements in $A_{2\tau}$ are more important and for those we should add τ copies, but the later elements are not as important and we can add fewer copies. In fact, we can geometrically decrease the number of copies we add from τ, to 1, over the course of the 2τ elements, resulting in a total of $\Theta(\tau \log \tau)$ copies. The resulting approximation ratio converges to $(1-1/e)$ for $\tau = o(k)$.

More concretely, consider the following algorithm,

Algorithm 2 $(1-1/e)$ Algorithm for copies when $\tau = o(k)$

```
1: Initialize $A = A_{2\tau}, i = 1$
2: **while** $i \leq \lceil \log 2\tau \rceil$ **do**
3:     $A = A \cup (\lceil \tau/2^{i-1} \rceil$ copies for each of $\{a_{2^i-1}, \ldots, a_{2^{i+1}-2}\} \cap A_{2\tau})$; $i = i + 1$
4: **while** $|A| < k$ **do** $A = A + \underset{x \in N-A}{\text{argmax}} f(x|A)$
5: Output: $A$
```

Essentially, we start with the set $A_{2\tau}$, add τ copies each for $\{a_1, a_2\}$, $\tau/2$ copies for each of $\{a_3, \ldots, a_6\}$, $\tau/4$ copies for each of $\{a_7, \ldots, a_{14}\}$ and so on, finally adding the rest of the $k - \Theta(\tau \log \tau)$ elements greedily. Notice that we could get the best possible guarantee with a minimizer oblivious algorithm i.e., the output is independent of the minimizer at any stage of the algorithm.

Theorem 8 *Algorithm 2 is* $\beta\Big(0, \frac{k-\Theta(\tau\log\tau)}{k-\tau}\Big) \xrightarrow{k\to\infty} (1-1/e)$ *approximate for* $\tau = o(k)$.

Proof The basic outline of the analysis is similar to that of the previous one for $\tau = o(\sqrt{k})$, so we focus only on the differences here.

Let A_1 denote the subset of A obtained in the final greedy phase (step 4 in Algorithm 2). Note that if $Z \cap A_{2\tau} = \emptyset$, then we have (3) and (4) as before and thus $f(A_{3\tau}) \geq 3f(Z|A-Z)$. By applying Lemma 5 with $l = k - \Theta(\tau\log\tau)$ this time, we get the desired. However, unlike the previous analysis, here $Z \cap A_{2\tau}$ need not be empty since we have less than τ copies for many elements. We would like to show that $f(A_{2\tau} - Z) \geq 2f(Z|A-Z)$ regardless. Let $m = \lceil \log 2\tau \rceil - 1$ and in fact, for ease of presentation we assume that $\tau = 2^m$. Also let $B_i = \{a_{2^i-1}, \ldots, a_{2^{i+1}-2}\} \cap A_{2\tau}$ i.e., the elements for which we add $\tau/2^{i-1}$ copies in the algorithm and let C_i denote the set of copies of these elements. Note that $|B_i| = 2^i$ and $|C_i| = 2\tau \geq 2|B_i|$ for all $i \leq m$, for $i = m+1$, $B_{m+1} = \{a_{2\tau-1}, a_{2\tau}\}$ and $|C_{m+1}| = 2$. We can assume that for every element in $A_{2\tau}$ included in minimizer Z, all copies of the element are also present in Z, hence Z removes at most $\lfloor \frac{\tau}{1+\tau/2^{i-1}} \rfloor = \lfloor \frac{|B_i|}{2}\frac{\tau}{\tau+2^{i-1}} \rfloor \leq |B_i|/2 - 1$ elements from $B_i, \forall i \leq m$. So we assume w.l.o.g. $|Z \cap B_1| = 0$.

Let us first examine the case where $Z \subset B_i \cup C_i$ for some $2 \leq i \leq m$ (the case of $i = m+1$ follows rather easily). Notice that $\sum_{j=1}^{i-1} |B_j| = 2(|B_i|/2 - 1)$, then from the observation above we have $2|Z \cap B_i| \leq 2\sum_{j=1}^{i-1}|B_j|$. This implies that we can injectively map any $|B_i|/2 - 1 = 2^{i-1} - 1$ elements in B_i, to two distinct elements with lower indices in $\cup_{j=1}^{i-1} B_j$. Then, just as we showed in (4), we have an injective mapping from every element in $Z \cap A_{2\tau}$ to two distinct elements in $\cup_{j=1}^{i-1} B_j$, and this gives us $f(\cup_{j=1}^{i-1} B_j) \geq 2f(Z|A-Z)$ which further implies $f(A_{2\tau}) \geq f(\cup_{j=1}^{i-1} B_j) + f(Z|\cup_{j=1}^{i-1} B_j) \geq 3f(Z|A-Z)$ and applying Lemmas 5 and 6 completes the case.

For the general case, let $x_j = |Z \cap B_j|, \forall j \in \{2, \ldots, m+1\}$ (with $x_{m+1} \leq 2$) and let $x_{m+2} = |Z \cap A_1|$. From $|Z| = \tau$ and the fact that Z contains all copies of every element in $|Z \cap A_{2\tau}|$, we have $\sum_{j=2}^{m+1} x_j(1 + \frac{\tau}{2^{j-1}}) + x_{m+2} \leq \tau$. Observe that to show the existence of the desired injective mapping, it suffices to show,

$$\begin{aligned} 2x_i &\leq |\cup_{j=1}^{i-1} B_j - Z| - \sum_{j=1}^{i-1} 2x_j \\ &= |\cup_{j=1}^{i-1} B_j| - \sum_{j=1}^{i-1} 3x_j, \forall i \in \{2, \ldots, m+2\} \end{aligned} \tag{5}$$

Consider the polytope given by $X = (x_2, \ldots, x_{m+2})$ and constraints $0 \leq x_j \leq \lfloor \frac{|B_i|}{2}\frac{\tau}{\tau+2^{i-1}} \rfloor \forall j$ and $\sum_{j=1}^{m+1} x_j(1 + \frac{\tau}{2^{j-1}}) + x_{m+2} \leq \tau$. Then the extreme points correspond exactly to the special cases we considered so far i.e. (i)$Z \cap$

$A_{2\tau} = \emptyset$ and (ii) $Z \subset B_i \cap C_i$, and we showed that the conditions (5) are satisfied for these cases. This implies the (linear) conditions (5) are satisfied for every point in the polytope and that completes the proof. $\square$

3.2 Algorithms in the possible absence of "copies"

To recap, thus far we chose a set of elements greedily and treated a suitably large subset of the initial few elements as 'critical' and added 'enough' copies of these elements to ensure that we keep a copy of each critical element in the set, even after adversarial removal. We aim to follow a similar scheme even in the absence of copies. In the general case however, we need to figure out a new way to ensure that our set is robust to removal of the first few critical elements chosen greedily. We first discuss how to approach this for $\tau = 1$.

3.2.1 Algorithms for $\tau = 1$ in the possible absence of "copies"

We start by discussing how one could construct a greedy set that is robust to the removal of a_1. In the case of copies, we would simply add a copy a'_1 of a_1, to accomplish this. Here, one approach would be to pick a_1 and then pick the rest of the elements greedily while ignoring a_1. Note that this would add a copy of a_1, if it were available, and if not it will permit the selection of elements which have small marginal on any set containing a_1, but possibly large marginal value in the absence of a_1. Such an element need not be selected by the standard Algorithm 1 that doesn't ignore a_1. Formally,

Algorithm 3 0.387 Algorithm

```
1: Initialize $A = \{a_1\}$
2: while $|A| < k$ do $A = A + \underset{x \in N-A}{\text{argmax}} f(x|A - \{a_1\})$
3: Output: $A$
```

This simple algorithm is in fact, asymptotically 0.387 approximate and the bound is tight (proof in Appendix A.3). However, a possible issue with the algorithm is that it is oblivious to the minimizers of the set at any iteration. It ignores a_1 throughout, even if a_1 stops being the minimizer after a few iterations. Thus, if after every iteration, we check the minimizer and stop ignoring a_1 once it is not a minimizer (i.e. proceed with standard greedy iterations after such a point), we achieve a performance guarantee of 0.5 (proof omitted). Note that this matches the guarantee obtained by copying just a_1, in presence of copies, and so in this sense, we can build a set that is robust to removal of a_1 in the general setting.

As we saw for the case of copies, in order to get even better guarantees, we need to look at the set of the first two elements, $\{a_1, a_2\}$. A direct generalization of line 2 in Algorithm 3, to a rule that ignores both a_1 and a_2, i.e. $\underset{x \in N-A}{\text{argmax}} f(x|A - \{a_1, a_2\})$, can be shown to have a performance bound less than 0.5. In fact, many natural rules that were tried resulted in upper bounds

≤ 0.5. Algorithm 4 avoids looking at both elements simultaneously and instead ignores a_1 until its marginal becomes sufficiently small and then does the same for a_2, if required.

Algorithm 4 $0.5547-\Omega(1/k)$ Algorithm

```
1: Initialize $A = \{a_1, a_2\}$
   **Phase 1:**
2: **while** $|A| < k$ **and** $f(a_1|A - a_1) > \frac{f(A)}{3}$ **do** $A = A + \underset{x \in N-A}{\mathrm{argmax}} f(x|A - a_1)$
   **Phase 2:**
3: **while** $|A| < k$ **and** $f(a_2|A - a_2) > \frac{f(A)}{3}$ **do** $A = A + \underset{x \in N-A}{\mathrm{argmax}} f(x|A - a_2)$
   **Phase 3:**
4: **while** $|A| < k$ **do** $A = A + \underset{x \in N-A}{\mathrm{argmax}} f(x|A)$
5: Output:$A$
```

The algorithm is asymptotically 0.5547-approximate (as an example, guarantee > 0.5 for $k \geq 50$) and note that it's minimizer oblivious and only uses greedy as subroutine, which makes it fast and easy to implement (recall that greedy can be replaced by thresholding). The analysis is presented in Appendix A.4.

Next, recall that we want to make the final set robust to removal of either one of the two elements $\{a_1, a_2\}$. In order to improve upon the guarantee of Algorithm 4, which deals with the two elements one at a time, first ignoring a_1 and then a_2 if required, we devise a way to add new elements while paying attention to both a_1 and a_2 simultaneously. To this end, consider the following addition rule,

$$\underset{|S| \leq m; S \subseteq N-A}{\mathrm{argmax}} \; g(S|A) = \underset{|S| \leq m; S \subseteq N-A}{\mathrm{argmax}} \Big[\min\big\{f(S + A - a_1), f(S + A - a_2)\big\} - \min\big\{f(A - a_1), f(A - a_2)\big\}\Big],$$

i.e. greedily adding m tuples but w.r.t. to the $g(.)$ function now instead of $f(.)$, while $z \in \{a_1, a_2\}$, for suitable $m \geq 1$. We need to resort to m-tuples instead of singletons because, for $m = 1$ we cannot guarantee improvements at each iteration, as there need not be any element that adds marginal value on both a_1 and a_2. However, for larger m we can show improving guarantees. More concretely, consider an instance where $f(a_1) = f(a_2) = 1$, a_1 has a copy a_1' and additionally both a_1 and a_2 have 'partial' copies, $f(a_i^j) = \frac{1}{k}$ and $f(a_i^j|a_i) = 0$ for $j \in \{1, \ldots, k\}$, $i \in \{1, 2\}$. Also, let there be a set G of $k - 2$ garbage elements with $f(G) = 0$. Finally, let $f(\{a_1, a_2\}) = 2$ and $f(a_i^j|X) = \frac{1}{k}$ if $a_i \notin X$ (and also $a_1' \notin X$ for $i = 1$). Running the algorithm with: (i) $m = 1$ outputs $\{a_1, a_2\} \cup G$ in the worst case, with (ii) $m = 2$ outputs a_1^j, a_2^j on step j of Phase 1 and thus, 'partially' copies both a_1 and a_2. Instead, if we run the algorithm with (iii) $m = 3$, the algorithm picks up a_1' and then copies a_2 almost completely with $\{a_2^1, \ldots, a_2^{k-3}\}$. In fact, we will show that while $\{a_1, a_2\}$ are minimizers, adding m-tuples in this manner allows us to guarantee that at each step we increase $g(A)$ by $\frac{m-1}{m}\frac{1}{k}$ times the difference from optimal. Thus, when m is large enough that $\frac{m-1}{m} \approx 1$, we effectively add value at the 'greedy' rate of $\frac{1}{k}$ times the difference from optimal (ref. Lemma 1). However, this is while $z \in \{a_1, a_2\}$, so we need to address the case when $\{a_1, a_2\}$ are not minimizers. One approach would be to follow along the lines of Algorithm 4 by adding singletons greedily w.r.t. f (similar to Phase 3) once the minimizer falls out of $\{a_1, a_2\}$. This is what we do in the algorithm below, recall that $\mathcal{Z}(A)$ is the set of minimizers of set A.

Algorithm 5 A $(1 - 1/e) - 1/\Theta(m)$ Algorithm for $\tau = 1$

```
input: $m$
1: Initialize $A = \{a_1, a_2\}$
   Phase 1:
2: while $|A| < k$ and $\mathcal{Z}(A) \subseteq \{a_1, a_2\}$ do
3:     $l = \min\{m, k - |A|\}$
4:     $A = A \cup \operatorname{argmax}_{|S|=l; S \subseteq N-A} g(S|A)$
   Phase 2:
5: while $|A| < k$ do $A = A + \operatorname{argmax}_{x \in N-A} f(x|A)$
6: Output: $A$
```

Before analyzing the approximation guarantee of Algorithm 5, we first need to show that in Phase 1, at each step we greedily add an m-tuple with marginal value $\frac{m-1}{k}$ times the difference from optimal. We do this by showing a more general property below,

Lemma 9 *Given two monotone submodular functions, f_1, f_2 on ground set N. If there exists a set S of size k, such that $f_i(S) \geq V_i, \forall i$, then for every m ($2 \leq m \leq k$), there exists a set $X \subseteq S$ with size m, such that,*

$$f_i(X) \geq \frac{m-1}{k} V_i, \forall i.$$

Proof First, we show that it suffices to prove this statement for two modular functions h_1, h_2. Note that we can reduce our ground set to S. Now, consider an arbitrary indexing of elements in $S = \{s_1, \dots, s_k\}$ and let $S_j = \{s_1, \dots, s_j\}, \forall j \in [k]$. Consider modular functions such that value of element s_j is $h_i(s_j) := f_i(s_j|S_{j-1})$. Note that $h_i(S) = f_i(S)$ and additionally, by submodularity, we have that $h_i(X)$ is a lower bound on $f_i(X)$ i.e. for every set $X \subseteq S$, $f_i(X) = \sum_{j: s_j \in X} f_i(s_j|X \cap S_{j-1}) \geq h_i(X)$. Also, we can assume w.l.o.g. that $h_i(S) = 1, \forall i$ and so it suffices to show that $h_i(X) \geq \frac{m-1}{k}, \forall i$.

We proceed by induction on m. For the base case of $m = 2$, we can pick elements $e_1, e_2 \in S$ such that $h_i(e_i) \geq \frac{1}{k}$ for $i \in \{1, 2\}$, and we are done. Now assume that the property holds for $m \leq p$ and we show it for $m = p + 1$ by contradiction. Consider an arbitrary set X_0 of size p, such that $h_i(X_0) \geq \frac{p-1}{k}$. Such a set exists by assumption, and note that if for some i, say $i = 1$, $h_1(X_0) \geq \frac{p}{k}$, we are done, since we can add an element $e \in S$ to X_0 such that $h_2(e + X_0) \geq \frac{p}{k}$. So we assume that $\frac{p-1}{k} \leq h_i(X_0) < \frac{p}{k}, \forall i$ to set up the contradiction.

Now, consider the reduced ground set $S - X_0$. Then we have that $h_i(S - X_0) > 1 - \frac{p}{k}, \forall i$ and since $|S - X_0| = k - p$, we have for the reduced ground set $S - X_0$, that there exists a set X_1 of size p such that $h_i(X_1) \geq \frac{p-1}{k-p} h_i(S - X_0) > \frac{p-1}{k}$, by the induction assumption. Using our second assumption (for contradiction), we have that $h_i(X_1) < \frac{p}{k}$. We repeat this until $|S - \cup_j X_j| \leq p$. Let $X' = S - \cup_j X_j$, then since $h_i(S) = 1 \, \forall i$, we have $0 < |X'| = p' \leq p$ and $h_i(X') > \frac{p'}{k} \, \forall i$. Now using the induction assumption for $m = p + 1 - p'$ and ground set $S - X'$, we have a set Y with $|Y| = p + 1 - p'$, such that $h_i(X' \cup Y) \geq h_i(X') + \frac{p-p'}{k-p'}(1 - h_i(X')) > \frac{p}{k}$, $\forall i$, yielding a contradiction. □

Based on this lemma, consider the below generalized greedy algorithm,

Algorithm 6 Generalized Greedy Algorithm

```
input: $m, V_1, V_2$
1: Initialize $A = \emptyset$
2: while $|A| < k$ do
3:     $m = \min\{m, k - |A|\}$
4:     Find $\Big\{X \,|\, f_i(X|A) \geq \frac{m-1}{k}[V_i - f_i(A)], X \subseteq N - A, |X| = m\Big\}$ by enumeration
5:     $A = A \cup X$
6: Output:$A$
```

Theorem 10 *If there exists a set S, $|S| = k$, such that $f_i(S) \geq V_i$, then Algorithm 6 finds a set A, $|A| = l \leq k$, such that,*

$$f_i(A) \geq \Big(\beta\Big(0, \frac{l}{k}\Big) - 1/\Theta(m)\Big) V_i,$$

by making $O(n^{m+1})$ queries.

Proof The analysis closely follows that of Lemma 1 in [25, 26] and Appendix A.1. Let A_j be the set of size $m \times j$ after iteration j, then using Lemma 9 for monotone submodular functions $f_i'(.) = f_i(.|A_j)$ with $V_i' = V_i - f_i(A_j)$ we have that $\exists X, |X| \leq m$, such that $f_i(X|A_j) \geq \frac{m-1}{k}[V_i - f_i(A_j)]$. Hence, $f_i(A_{j+1}) \geq f_i(A_j) + \frac{m-1}{k}[V_i - f_i(A_j)]$. Now similar to what is shown in [25,26], this gives us that after l iterations,

$$\begin{aligned} f_i(A_l) \geq \Big(1 - \Big(1 - \frac{m-1}{k}\Big)^l\Big) V_i &= \beta\Big(0, \frac{ml}{k}\Big)(1 - 1/\Theta(m)) V_i \\ &= \Big(\beta\Big(0, \frac{ml}{k}\Big) - 1/\Theta(m)\Big) V_i . \square \end{aligned}$$

Thus, Algorithm 6 gives a deterministic $(1 - 1/e) - 1/\Theta(m)$ approximation for bi-objective maximization of monotone submodular functions subject to cardinality constraints. We define this problem more generally and formally in Section 3.2.3, but for now, note that the problem of maximizing the minimum of two monotone submodular functions is a special case of the above, where we don't need to input V_i, and hence for Algorithm 5, we have that while in

Phase 1, after l iterations,

$$\begin{aligned} g(A) &= \min\{f(A-a_1), f(A-a_2)\} \\ &\geq \Big(\beta\Big(0, \frac{l}{k}\Big) - 1/\Theta(m)\Big) \min\{f(OPT(k,N,1)-a_1), f(OPT(k,N,1)-a_2)\} \\ &\geq \Big(\beta\Big(0, \frac{l}{k}\Big) - 1/\Theta(m)\Big) g(OPT(k,N,1)) \end{aligned} \tag{6}$$

Note that for Algorithm 6, Lemma 5 applies, albeit with the additive $-1/\Theta(m)$ term. We now present the analysis of Algorithm 5.

Theorem 11 *Given $m \geq 2$, Algorithm 5 is $\beta(0, \frac{k-2m-2}{k}) - 1/\Theta(m) \xrightarrow{k\to\infty} (1-1/e) - 1/\Theta(m)$ approximate, and makes $O(n^{m+1})$ queries.*

Proof Let $A_0 = \{a_1, a_2\}$. Consider the function $g^0(S) = \min_{i\in\{1,2\}}\{f(S-a_i)\}$ and let z be a minimizer of output set A as usual and define $z^0(S) = \arg\min_{i\in\{1,2\}}\{f(S-a_i)\}$. Note that if $z^0(S) \in \mathcal{Z}(S)$ then $g^0(S) = g(S)$. Also, with the standard definition of marginal, note that for any set $S \cap A_0 = \emptyset$, $g^0(S|X) \geq \min_{i\in\{1,2\}}\{f(S|X-a_i)\} \geq f(S|X)$. Let U be the set added during Phase 1 and similarly W during Phase 2. Also, let $U = \{u_1, \ldots, u_p\}$, where each u_i is a set of size m and similarly $W = \{w_1, \ldots, w_r\}$, where each w_i is a singleton. Let $OPT = g(OPT(k,N,1))$ and note that if $r = 0$ i.e., Phase 2 doesn't occur, we have that $z \in A_0$ and using (6), $g(A) \geq \big[\beta(0, \frac{k-2}{k-1}) - 1/\Theta(m)\big]OPT$. So assume $r > 0$ and also $p \geq 2$, since the case $p = 1$ will be easy to handle later. Now, let $f(z|A-z) = \eta f(A)$, and so $g(A) = (1-\eta)f(A)$. Using analysis similar to Lemma 5, we will focus on showing that,

$$f(A) \geq \Big[\beta\Big(3\eta, \frac{k-2m}{k-1}\Big) - 1/\Theta(m)\Big]OPT \tag{7}$$

Lemma 6 then gives $g(A) \geq \big[\beta(0, \frac{k-2m}{k-1}) - 1/\Theta(m)\big]OPT$.

During the rest of the proof, we sometimes ignore the $1/\Theta(m)$ term with the understanding that it is present by default. To show (7), let $a_i = z^0(A_0 \cup U)$. Then, observe that,

$$\begin{aligned} f(A) = g^0(A_0 \cup (U-u_p)) &+ g^0(u_p|A_0 \cup (U-u_p)) \\ &+ f(a_i|(A_0\cup U) - a_i) + f(W|A_0 \cup U) \end{aligned} \tag{8}$$

For the first term in (8), we have

$$g^0(A_0 \cup (U-u_p)) = g^0(A_0 \cup u_1) + g^0(\{u_2, \ldots, u_{p-1}\}|A_0 \cup u_1).$$

Then, due to the greedy nature of Phase 1, we have using Theorem 10 (ignoring $1/\Theta(m)$ term),

$$g^0(\{u_2,\ldots,u_{p-1}\}|A_0 \cup u_1) \geq \beta\Big(0, \frac{mp-2m}{k-1}\Big)(OPT - g^0(A_0 \cup u_1)) \tag{9}$$

As usual, the k in the denominator was replaced by $k-1$ because we compare the value to a set of size $k-1$ ($OPT = f(OPT(k, N, 1) - z)$). Similarly, for the last term in (8), we have,

$$f(W|A_0 \cup U) \geq \beta\big(0, \frac{r}{k-1}\big)(OPT - f(A_0 \cup U)) \tag{10}$$

Now we make some substitutions, let $\Delta = g^0(A_0 \cup u_1) + g^0(u_p|A_0 \cup (U - u_p)) + f(a_i|(A_0 \cup U) - a_i)$, $\alpha_p = \frac{(m-1)(p-2)}{k-1}$, $\alpha_r = \frac{r}{k-1}$. Then using $k = mp + r + 2$ we get, $\alpha_p + \alpha_r = \frac{k-2m}{k-1} - 1/\Theta(m)$. Also, $f(A_0 \cup U) = \Delta + g^0(\{u_2, \ldots, u_{p-1}\}|A_0 \cup u_1)$. This gives us,

$$\begin{aligned}
&f(A) = \Delta + g^0(\{u_2, \ldots, u_{p-1}\}|A_0 \cup u_1) + f(W|A_0 \cup U) \\
&\overset{(a)}{\geq} \Delta + g^0(\{u_2, \ldots, u_{p-1}\}|A_0 \cup u_1) + \beta(0, \alpha_r)[OPT - f(A_0 \cup U)] \\
&\geq \Delta + (1 - \beta(0, \alpha_r))g^0(\{u_2, \ldots, u_{p-1}\}|A_0 \cup u_1) + \beta(0, \alpha_r)(OPT - \Delta) \\
&\overset{(b)}{\geq} \Delta + (1/e^{\alpha_r})\beta(0, \alpha_p)(OPT - g^0(A_0 \cup u_1)) + \beta(0, \alpha_r)(OPT - \Delta) \\
&\geq \Delta + (\beta(0, \alpha_r) + \beta(0, \alpha_p)/e^{\alpha_r})[OPT - \Delta] \\
&= \Delta + \Big[\beta\Big(0, \frac{k-2m}{k-1}\Big) - 1/\Theta(m)\Big][OPT - \Delta]
\end{aligned}$$

where the last equality holds asymptotically, (a) comes from (10) and (b) from (9). Assume for the time being, that for any x, $3f(x|A-x) \leq \Delta$, which implies that $\Delta \geq 3\eta f(A)$. Armed with these inequalities, the same simple algebra as in the proof of Lemma 5, gives us (7). More concretely, ignoring the $1/\Theta(m)$ term, we have,

$$\begin{aligned}
f(A) &\geq \Delta + \beta\Big(0, \frac{k-2m}{k-1}\Big)[OPT - \Delta] \\
&\geq 3\eta f(A)\Big(1 - \beta\Big(0, \frac{k-2m}{k-1}\Big)\Big) + \beta\Big(0, \frac{k-2m}{k-1}\Big)OPT
\end{aligned}$$

$$\begin{aligned}
(1 - 3\eta e^{-\frac{k-2m}{k-1}})f(A) &\geq (1 - e^{-\frac{k-2m}{k-1}})OPT \\
\implies f(A) &\geq \beta\Big(3\eta, \frac{k-2m}{k-1}\Big)OPT
\end{aligned}$$

To finish the proof, we need to show $3f(x|A-x) \leq \Delta$. We break this down by first showing in two steps that for all x, $2f(x|A-x) \leq g^0(A_0 \cup u_1) = f(a_2) + g^0(u_1|A_0)$, followed by proving that $f(x|A-x) \leq g^0(u_p|A_0 \cup (U - u_p)) + f(a_i|(A_0 \cup U) - a_i)$.

Step 1: for all x, $f(x|A-x) \leq f(a_2)$. To see this for $x \neq a_1$, note that $f(x|A-x) \leq f(x|A_0 - x)$ and further, $f(x|A_0 - x) \leq f(a_2|a_1) \leq f(a_2)$, where the first inequality is because a_2 adds maximum marginal value to a_1. For $x = a_1$, since Phase 1 ends, we have that $f(a_1|A - a_1) \leq f(a_1|a_2 + U) \leq f(y|(A_0 \cup U) - y)$ for some y not in A_0 and then we have $f(y|(A_0 \cup U) - y) \leq f(a_2)$.

Step 2: $f(x|A-x) \leq g^0(u_1|A_0)$. For $x \notin A_0$, we have that $f(x|A-x) \leq f(x|A_0) \leq g^0(x|A_0) \leq g^0(u_1|A_0)$. For $x \in A_0$, from the fact that $A_0 \cup U$ has a minimizer $y \notin A_0$, we have that $f(x|A-x) \leq f(x|(A_0 \cup U)-x) \leq f(y|(A_0 \cup U)-y)$ and further $f(y|(A_0 \cup U)-y) \leq f(y|A_0) \leq g^0(y|A_0) \leq g^0(u_1|A_0)$.

Finally, we show that $f(x|A-x) \leq g^0(u_p|A_0 \cup (U-u_p)) + f(a_i|(A_0 \cup U)-a_i)$, for all x. Let $a_j = z^0(A_0 \cup (U-u_p))$ and observe that,

$$\begin{aligned}
&g^0(A_0 \cup U) - g^0(A_0 \cup (U-u_p)) + f(a_i|(A_0 \cup U)-a_i) \\
&= f((A_0 \cup U)-a_i) - f((A_0 - a_j) \cup (U-u_p)) + f(a_i|(A_0 \cup U)-a_i) \\
&= f(A_0 \cup U) - f((A_0 - a_j) \cup (U-u_p)) \\
&\geq f(a_j|(A_0 - a_j) \cup (U-u_p))
\end{aligned}$$

Now, before adding u_p, we have that $g^0(.) = g(.)$ and in fact, a_j is a minimizer of $A_0 \cup (U-u_p)$, so clearly, for all $x \in A_0 \cup (U-u_p)$, the desired is true. For $x \in u_p$, it is true since $f(u_p|A_0 \cup (U-u_p)) \leq g^0(u_p|A_0 \cup (U-u_p))$. For $x \in W$, we have that $f(x|A-x) \leq f(x|A_0 \cup (U-u_p)) \leq g^0(x|A_0 \cup (U-u_p)) \leq g^0(u_p|A_0 \cup (U-u_p)))$, and we are done.

The case $p=1$ can be dealt with same as above, except that now $\Delta = g^0(A_0 \cup u_1) + f(a_i|(A_0 \cup u_1)-a_i) + f(w_1|A_0 \cup u_1) = f(A_0 \cup u_1 \cup w_1)$. □

Before discussing a similar result for constant $\tau \geq 1$, we first describe a fast 0.387 approximation for $\tau = o(\sqrt{k})$, which we will use when showing a $(1-1/e)-\epsilon$ approximation for fixed τ.

3.2.2 0.387 Algorithm for $\tau \ll \sqrt{k}$ in the possible absence of "copies"

One can recast the first algorithm in Section 3.1.2, which greedily chooses $\{a_1, \ldots, a_{k-2\tau^2}\}$ elements and adds τ copies for each of the first 2τ elements. as greedily choosing 2τ elements, ignoring them and choosing another 2τ greedily (which will be copies of the first 2τ) and repeating this τ times in total, leading to a set which contains $A_{2\tau}$ and $\tau-1$ copies of each element in $A_{2\tau}$. Then, ignoring this set, we greedily add till we have k elements in total. Thus, the algorithm essentially uses the greedy algorithm as a sub-routine $\tau+1$ times. Based on this idea, we now propose an algorithm for $\tau = o(\sqrt{k})$, which can also be viewed as an extension of the 0.387 algorithm for $\tau = 1$. To be more precise, it achieves an asymptotic guarantee of 0.387 for $\tau = o(\sqrt{\frac{k}{c(k)}})$, where $c(k)$ is an input parameter that governs the trade off between how fast the guarantee approaches 0.387 as k increases and how large τ can be for the guarantee to still hold. In fact, the guarantee is $0.387\left(1 - \frac{1}{\Theta(c(k))}\right)$ with $c(k)$ being a function monotonically increasing in k and approaching ∞ as $k \to \infty$. The factor also degrades proportionally to $1 - \frac{\tau^2 c(k)}{k}$, as τ approaches $\sqrt{\frac{k}{c(k)}}$.

Algorithm 7 Algorithm for $\tau = o\Big(\sqrt{\frac{k}{c(k)}}\Big)$

```
1: Initialize $\tau' = c(k)\tau^2, A_0 = A_1 = X = \emptyset$.
2: while $|A_0| < \tau'$ do
3:     while $|X| < \tau'/\tau$ do $X = X + \underset{x \in N-(A_0 \cup X)}{\operatorname{argmax}} f(x|X)$
4:     $A_0 = A_0 \cup X$; $X = \emptyset$
5: while $|A_1| < k - \tau'$ do $A_1 = A_1 + \underset{x \in N-(A_0 \cup A_1)}{\operatorname{argmax}} f(x|A_1)$
6: Output: $A_0 \cup A_1$
```

Theorem 12 *Algorithm 7 has an approximation ratio of* $\frac{e-1}{2e-1+\frac{e-1}{c(k)}} = \frac{e-1}{2e-1}\big(1-\frac{1}{\Theta(c(k))}\big) \xrightarrow{k\to\infty} 0.387$ *for* $\tau = o\Big(\sqrt{\frac{k}{c(k)}}\Big)$.

Proof Let $A = A_0 \cup A_1$ be the output with A_0, A_1 as in the algorithm. Define $Z_0 = A_0 \cap Z$ and $Z_1 = Z - Z_0 = A_1 \cap Z$. Let $OPT(k-\tau, N - Z_0, 0) = A_0' \cup X$ where $A_0' = OPT(k-\tau, N-Z_0, 0) \cap A_0$ and $X \cap A_0' = \emptyset$. Now note that,

$$\begin{aligned} f(A_0 - Z_0) + f(OPT(k-\tau, N-A_0, 0)) &\geq f(A_0') + f(X) \\ &\geq f(OPT(k-\tau, N-Z_0, 0)) \\ &\geq g(OPT(k, N, \tau)) \quad [\because \text{Lemma 2}] \end{aligned}$$

Which implies,

$$f(A_0 - Z_0) \geq g(OPT(k,N,\tau)) - f(OPT(k-\tau, N-A_0, 0)) \tag{11}$$

In addition,

$$f(A_1) \geq \beta\Big(0, \frac{k-\tau'}{k-\tau}\Big) f(OPT(k-\tau, N-A_0, 0)) \tag{12}$$

Next, index disjoint subsets of A_0 based on the loop during which they were added. So the subset added during loop i is denoted by A_0^i, where $i \in \{1, \dots, \tau\}$. So the last subset consisting of $\tau c(k)$ elements is A_0^τ.

Now, if Z_0 includes at least one element from each A_0^i then $Z_1 = \emptyset$ and for this case we have from (11) and (12) above,

$$\begin{aligned} f(A - Z) &\geq \max\{f(A_0 - Z_0), f(A_1)\} \\ &\geq \max\{g(OPT(k,N,\tau)) - f(OPT(k-\tau, N-A_0, 0)), \\ &\qquad \beta\Big(0, \frac{k-\tau'}{k-\tau}\Big) f(OPT(k-\tau, N-A_0, 0))\} \\ &\geq \frac{\beta\Big(0, \frac{k-\tau'}{k-\tau}\Big)}{1+\beta\Big(0, \frac{k-\tau'}{k-\tau}\Big)} g(OPT(k,N,\tau)) \\ &\xrightarrow{k\to\infty} \frac{e-1}{2e-1} g(OPT(k,N,\tau)) \end{aligned}$$

Next, suppose that $|Z_1| > 0$, then there is some A_0^j such that $A_0^j \cap Z = \emptyset$. Further let $f(Z_1|A-Z) = \eta f(A_1)$, then since $|A_0^j| \geq c(k)|Z_1|$, similar to (3), we have due to greedy iterations and submodularity, $f(A_0^j) \geq c(k) f(Z_1|A-Z) = c(k)\eta f(A_1)$. Also note that,

$$f(A-Z) \geq f(A_1 - Z_1) \geq f(A_1) - f(Z_1|A-Z) \geq (1-\eta) f(A_1) \quad (13)$$

Moreover, let A_1' be the set of first τ elements of A_1. Then, due to greedy iterations we have $f(A_1') \geq f(Z_1|A_1 - Z_1) \geq \eta f(A_1)$. Thus, from Lemma 5, with N replaced by $N - A_0$, k replaced by $k-\tau$, $S = A_1'$ with $c = \eta$ and $l = k - |S| = k - \tau' - \tau$, we have,

$$f(A_1) \geq \beta(\eta, \frac{k-\tau'-\tau}{k-\tau}) f(OPT(k-\tau, N-A_0, 0)) \quad (14)$$

From (11), (13) and (14),

$$\begin{aligned} f(A-Z) &\geq \max\{f(A_0 - Z_0), (1-\eta) f(A_1), f(A_0^j)\} \\ &\geq \max\{f(A_0 - Z_0), (1-\eta) f(A_1), c(k)\eta f(A_1)\} \\ &\overset{(a)}{\geq} \max\{g(OPT(k,N,\tau)) - f(OPT(k-\tau, N-A_0, 0)), \\ &\qquad \frac{c(k)}{1+c(k)}\beta(\frac{1}{1+c(k)}, \frac{k-\tau'-\tau}{k-\tau}) f(OPT(k-\tau, N-A_0, 0))\} \\ &\geq \frac{\frac{c(k)}{1+c(k)}\beta\left(\frac{1}{1+c(k)}, \frac{k-\tau'-\tau}{k-\tau}\right)}{1 + \frac{c(k)}{1+c(k)}\beta\left(\frac{1}{1+c(k)}, \frac{k-\tau'-\tau}{k-\tau}\right)} g(OPT(k,N,\tau)) \\ &\xrightarrow{k\to\infty} \frac{e-1}{2e-1} g(OPT(k,N,\tau)) \end{aligned}$$

where (a) follows by substituting $\eta = \frac{1}{1+c(k)}$. $\square$

3.2.3 $(1-1/e)-\epsilon$ *algorithm for constant* τ

In Section 3.2.1, inspired by the 2-Copy algorithm, we derived a $(1-1/e) - 1/\Theta(m)$ approximation for the general case, using a phase wise approach. In the first phase, while the minimizers are restricted to the set $A_0 = \{a_1, a_2\}$, we build a set robust to removal of either of these elements by deriving and using an algorithm for bi-objective maximization of monotone submodular functions. In the second and final phase, we filled in the rest of the set with standard greedy iterations (like Algorithm 1).

This phase wise approach however, doesn't generalize well for $\tau > 1$ since we can have a minimizer that intersects with the initial set but is not a subset of the initial set, unlike for $\tau = 1$ where a minimizer z is either in $\{a_1, a_2\}$ or not. Instead, an alternative approach comes from reinterpreting the result from $\tau = 1$, where, we want to build a set that has a large value on both $f_1(.) = f(.|a_1)$ and $f_2(.) = f(.|a_2)$ simultaneously, to deal with the scenarios when either of these elements is a minimizer. Also, we can capture the notion of continuing greedily w.r.t. $f(.)$ once the set becomes robust to removal of either of a_1 or a_2, by considering a third function $f_3(.) = f(.|\{a_1, a_2\})$. Thus, instead of separate phases, we can think about a single multi-objective problem over three monotone submodular functions f_1, f_2, f_3, and try to add a set A_1 to $A_0 = \{a_1, a_2\}$, such that $f_i(A_1) \geq (1 - 1/e)f_i(OPT(k, N, 1)), \forall i$. To see why this serves our purposes, consider the scenario where a_2 is a minimizer for the final set A,

$$\begin{aligned} g(A) = f(A_1 + A_0 - a_2) &= f_1(A_1) + f(a_1) \\ &\geq f(a_1) + (1 - 1/e)(f(OPT(k, N, 1)) - f(a_1)) \\ &\geq (1 - 1/e)g(OPT(k, N, 1)). \end{aligned}$$

Generalizing this for larger τ, we start with the set A_0 obtained by running Algorithm 7 with $\tau' = 3\tau^2$, implying $|A_0| = 3\tau^2$. Now, consider the monotone submodular functions $f_i(.) = f(.|Y_i)$ for every possible subset Y_i ($|Y_i| \geq 3\tau^2 - \tau$) of A_0 and denote the set of functions by $\mathcal{L}$ ($|\mathcal{L}| = 2^{\tilde{\Theta}(\tau)}$, large but constant).

Assuming there exists a set S of size $k - 3\tau^2$, such that $f_i(S) \geq (1 - \Theta(\frac{1}{k}))\big[g(OPT(k, N, \tau)) - f(Y_i)\big]$, we would like to solve an instance of a multi-objective submodular maximization problem (made more precise later) to find a set A_1 of size $k - 3\tau^2$ such that $f_i(A_1) \geq \beta(0, 1)(1 - \Theta(\frac{1}{k}))\big[g(OPT(k, N, \tau)) - f(Y_i)\big]$. We know that $OPT(k, N, \tau)$ is a set such that, $f_i(OPT(k, N, \tau)) \geq f(OPT(k, N, \tau)) - f(Y_i) \geq g(OPT(k, N, \tau)) - f(Y_i)$, however it is a set of size k. So before we can puzzle out how to find set A_1, we need the proof of existence of set S.

Lemma 13 *Given a constant number of monotone submodular functions $\mathcal{L} = \{f_i, \ldots, f_l\}$ and values $\mathcal{V} = \{V_i, \ldots, V_l\}$ and a set S' of size k, such that $f_i(S') \geq V_i, \forall i$. There exists a set S of size $k - p$ such that,*

$$f_i(S) \geq \Big(1 - \frac{l}{k - (p + l - 1)}\Big)^p V_i = \Big(1 - \Theta(1/k)\Big)V_i$$

for constant $l, p \ll k$.

Proof This is clearly true for $l = 1$ since $\exists S \subset S', |S| = k - p$, such that, $f(S) \geq \frac{k-p}{k} f(S')$. More generally, we show that $\exists S_1 \subset S', |S_1| = k - 1$, such that $f_i(S_1) \geq \frac{k-2l}{k-l} f_i(S')$. Then we can reapply this on S_1 to get $S_2 \subset S_1, |S_2| = k - 2$ and $f_i(S_2) \geq \frac{k-2l-1}{k-l-1} f_i(S_1)$. Repeating this p times over all, we get $S_p \subset S', |S_p| = k - p$ and $f_i(S_p) \geq \Pi_{j=0}^{p-1} \frac{k-j-2l}{k-j-l} V_i \geq \big(\frac{k-p-2l+1}{k-p-l+1}\big)^p V_i$, which gives the desired. So, it remains to show the claimed lower bound on $f_i(S_1)$.

Let $S' = \{s'_1, \ldots, s'_k\}$ and let $S'_j = \{s'_1, \ldots, s'_j\}$. Similar to the argument in Lemma 9, it suffices to show this for modular functions h_i where $h_i(S') = 1$ and $h_i(s'_j) = f_i(s'_j|S'_{j-1}), \forall i$. W.l.o.g., assume that S' is indexed such that $h_1(s'_j) \leq h_1(s'_{j+1})$. Consider the set $S'_{1+\lceil \frac{l-1}{l} k \rceil}$ of the $1 + \lceil \frac{l-1}{l} k \rceil$ smallest elements elements w.r.t. h_1. There are at most $(k - \lceil \frac{l-1}{l} k \rceil)(l - 1) \leq \lceil \frac{l-1}{l} k \rceil < |S'_{1+\lceil \frac{l-1}{l} k \rceil}|$ distinct elements which are in the top $(k - \lceil \frac{l-1}{l} k \rceil) = \lfloor \frac{k}{l} \rfloor$ for some function $h_i, i \geq 2$. Hence $\exists j_0 \leq 1 + \lceil \frac{l-1}{l} k \rceil$ such that s'_{j_0} is one of the $\lceil \frac{l-1}{l} k \rceil$ smallest elements for each function $h_i, i \geq 2$. This implies that if we remove s'_{j_0}, we have $h_i(S' - s'_{j_0}) \geq 1 - \frac{1}{\lfloor \frac{k}{l} \rfloor} \geq \frac{k-2l}{k-l}, \forall i$. □

Now that we know that set S of size $k - 3\tau^2$ satisfying the desired inequalities exists, we need an algorithm to find a set A_1 that $(1-1/e)$-approximates the value of S for each f_i. While Algorithm 6 only works for up to two functions, as mentioned in the beginning, a randomized $(1-1/e-\epsilon)$ algorithm for a more general problem was given in [10] and we use it here.

Lemma 14 (Chekuri, Vondrák, Zenklusen [10]) *Given constant number of monotone submodular functions $\mathcal{L} = \{f_i, \dots, f_l\}$ and values $\mathcal{V} = \{V_i, \dots, V_l\}$. If there exists a set S ($|S| \leq k$) with $f_i(S) \geq V_i, \forall i$, there is a polynomial time algorithm which finds a set X of size $\leq k$ such that $f_i(X) \geq (1-1/e-\epsilon)V_i, \forall i$, with constant probability, for a constant $\epsilon \leq \frac{1}{\log l}$, making $O(n^{1/\epsilon^3})$ queries. If a set S with the value lower bounds given by $\mathcal{V}$ doesn't exist, then the algorithm gives a certificate of non-existence.*

Denote the algorithm by $\mathcal{A}$ and let the set output be $\mathcal{A}(\mathcal{V}, \mathcal{L})$, for inputs $\mathcal{V}, \mathcal{L}$ as described above. To ease notation, from here on we generally ignore the ϵ term.

A final hurdle in using the above is that we need to input the values $V_i = g(OPT(k, N, \tau)) - f(Y_i)$ and hence, we need an estimate of $OPT = g(OPT(k, N, \tau))$. We can overestimate OPT as long as the problem remains feasible. However, underestimating OPT results in a loss in guarantee. Using Algorithm 7, we can quickly find lower and upper bounds lb, ub such that $lb \leq OPT \leq ub = lb/0.387$ and then run the multi-objective maximization algorithm above with a geometrically increasing sequence of $O(1/\log(1+\delta))$ many values to get an estimate OPT' within factor $(1 \pm \delta)$ of OPT.

In summary, our scheme starts with the set A_0 of size $3\tau^2$, obtained by running Algorithm 7 with $\tau' = 3\tau^2$, then uses the algorithm for multi-objective optimization as a subroutine to find the estimate $OPT' \geq (1-\delta)OPT$ and simultaneously a set $A_1 = \mathcal{A}(\{OPT' - f(Y_i)\}_i, \mathcal{L})$ of size $k - 3\tau^2$, such that $f_i(A_1) \geq (1-1/e)(1-\Theta(1/k))(OPT' - f(Y_i))$.

The final output is $A = A_0 \cup A_1$ and we next show that this is asymptotically $(1-1/e-\epsilon)$ approximate.

Proof We ignore the ϵ and $\Theta(1/k)$ terms to ease notation. First, note that if $Z \subseteq A_0$, then $f(.|A_0 - Z) \in \mathcal{L}$ gives us, $f(A_1|A_0 - Z) \geq (1-1/e)(OPT' - f(A_0 - Z))$. Hence,

$$\begin{aligned} g(A) &= f(A_0 - Z) + f(A_1|A_0 - Z) \\ &\geq f(A_0 - Z) + (1-1/e)(OPT' - f(A_0 - Z)) \\ &\geq (1-1/e)OPT'. \end{aligned}$$

If $Z \not\subseteq A_0$, then let $Z_1 = Z \cap A_1$ and $Z_0 = Z - Z_1$. Similar to the proof of Theorem 12, let A_0^i denote the ith set of 3τ elements greedily chosen for constructing A_0, $i \leq \tau$. Since $|Z_0| < \tau$, $\exists i$ such that $A_0^i \cap Z_0 = \emptyset$. Then analogous to the proof of Theorem 12, we have due to greedy additions,

$$f(A_0 - Z_0) \geq f(A_0^i) \geq 3f(Z_1|A - Z) \tag{15}$$

(in contrast with $c(k)f(Z_1|A - Z)$ in Theorem 12). Now, since $f(.|A_0 - Z_0)$ is one of the functions in $\mathcal{L}$, we have $f(A_1|A_0 - Z_0) \geq (1-1/e)(OPT' - f(A_0 - Z_0))$ which implies,

$$f(A - Z_0) = f(A_0 - Z_0) + f(A_1|A_0 - Z_0) \geq f(A_0 - Z_0) + (1-1/e)(OPT' - f(A_0 - Z_0)).$$

Further, letting $f(Z_1|A - Z) = \eta f(A - Z_0)$ and using (15),

$$\begin{aligned} f(A - Z_0) &\geq \frac{3\eta}{e} f(A - Z_0) + (1 - 1/e)OPT' \\ &\geq \beta(3\eta, 1)OPT'. \end{aligned}$$

Now, using Lemma 6 we have $g(A) = f(A - Z) \geq (1 - \eta)f(A - Z_0) \geq \beta(0,1)OPT'$. □

Finally, consider the following generalization of Lemma 6,

Conjecture 15 *Given $l \geq 1$ (treated as a constant) monotone submodular functions $f_1, \ldots, f_l$ on ground set N, a set $S \subseteq N$ of size k, such that $f_i(S) \geq V_i$ for all $i \in [l]$ and an arbitrary m with $k \geq m \geq l$, there exists a set $X \subseteq S$ of size m, such that,*

$$f_i(X) \geq \frac{m - \Theta(1)}{k} V_i, \quad \forall i \in [l]$$

If true, the above would give a greedy deterministic $(1 - 1/e) - 1/\Theta(m)$ approximation for the multi-objective optimization problem and thus also for our robust problem, for constant τ.

4 Extension to general constraints

So far, we have looked at a robust formulation of $P1$, where we have a cardinality constraint. However, there are more sophisticated applications where we find instances of budget or even matroid constraints. In particular, consider the generalization $\max_{A \in \mathcal{I}} \min_{|B| \leq \tau} f(A \backslash B)$, for some independence system $\mathcal{I}$. By definition, for any feasible set $A \in \mathcal{I}$, all subsets of the form $A \backslash B$ are feasible as well, so the formulation is sensible. Let's briefly discuss the case of $\tau = 1$ and suppose that we are given an α approximation algorithm $\mathcal{A}$, with query/run time $O(R)$ for the $\tau = 0$ case. Let G_0 denote its output and z_0 be a minimizer of G_0. Consider the restricted system $\mathcal{I}_{z_0} = \{A : z_0 \in A, A \in \mathcal{I}\}$. Now, in order to be able to pick elements that have small marginal on z_0 but large value otherwise, we can generalize the notion of ignoring z_0 by maximizing the monotone submodular function $f(.\backslash z_0)$ subject to the independence system I_{z_0}. However, unlike the cardinality constraint case, where this algorithm gives a guarantee of 0.387, the algorithm can be arbitrarily bad in general (because of severely restricted I_{z_0}, for instance). We tackle this issue by adopting an enumerative procedure.

Let $\mathcal{A}_j$ denote the algorithm for $\tau = j$ and let $\mathcal{A}_j(N, Z)$ denote the output of $\mathcal{A}_j$ on ground set N and subject to restricted system $\mathcal{I}_Z$. Finally, let $\hat{z}(A) = \underset{x \in A}{\text{argmax}} f(x)$. With this, we have for general constraints:

Algorithm 8 $\mathcal{A}_\tau : \frac{\alpha}{\tau+1}$ for General Constraints

```
1: Initialize $i = 0, Z = \emptyset$
2: while $N - Z \neq \emptyset$ do
3:     $G_i = \mathcal{A}_0(N - Z, \emptyset)$
4:     $z_i \in \hat{z}(G_i)$;    $Z = Z \cup z_i$
5:     $M_i = z_i \cup \mathcal{A}_{\tau-1}(N - Z, z_i)$;    $i = i + 1$
6: Output: argmax$\{g_\tau(S) | S \in \{G_j\}_{j=0}^i \cup \{M_j\}_{j=0}^i\}$
```

To understand the basic idea behind the algorithm, assume that z_0 is in an optimal solution for the given τ. Then, given the algorithm $\mathcal{A}_{\tau-1}$, if a minimizer of the set $M_0 = z_0 \cup \mathcal{A}_{\tau-1}(N - z_0, z_0)$ includes z_0, it only removes $\tau - 1$ elements from $\mathcal{A}_{\tau-1}(N - z_0, z_0)$. On the other hand, if a minimizer doesn't include z_0, $g_\tau(M_0) \geq f(z_0) \geq \frac{f(M_0) - g_\tau(M_0)}{\tau}$. These two cases yield the desired ratio, however, since z_0 need not be in an optimal solution, we systematically enumerate.

Theorem 16 *Given an α approximation algorithm $\mathcal{A}$ for $\tau = 0$ with query time $O(R)$, algorithm $\mathcal{A}_\tau$ described above guarantees ratio $\frac{\alpha}{\tau+1}$ for general τ with query time $O(n^\tau R + n^{\tau+1})$*

Proof We proceed via induction on $j \in \{0, \ldots, \tau\}$. Clearly, for $j = 0$, $\mathcal{A}_0 \equiv \mathcal{A}$, and the statement holds. Assume true for $j \in \{0, 1, \ldots, \tau - 1\}$, then we show validity of the claim for $\mathcal{A}_\tau$. The query time claim follows easily since the while loop runs for at most n iterations and each iteration makes $O(n^\tau + n^{\tau-1} R)$ queries (by assumption on query time of $\mathcal{A}_{\tau-1}$) and updating the best solution at the end of each iteration (counts towards the final output step) takes $O(n^\tau)$ time to find the minimizer by brute force for two sets G_i and M_i.

Now, let $OPT(\mathcal{I}, N, \tau)$ denote an optimal solution to $\max_{A \in \mathcal{I}} \min_{|B| = \tau} f(A - B)$ on ground set N and assume that $z_0 \in OPT(\mathcal{I}, N, \tau)$. For any minimizer B of A, we have for every element $z \in \hat{z}(A)$, $f(z) \geq \frac{f(B)}{\tau} \geq \frac{f(A) - f(A - B)}{\tau}$. Let Z_0 denote a minimizer of G_0. Hence, if $z_0 \notin Z_0$, we have that $g_\tau(G_0) \geq f(z_0) \geq \frac{f(G_0) - g_\tau(G_0)}{\tau}$, giving us $g_\tau(G_0) \geq \frac{f(G_0)}{\tau+1}$. Instead if z_0 is in the minimizer of G_0 and if $f(G_0 - Z_0) < \frac{f(G_0)}{\tau+1}$, then we have that $f(Z_0 | G_0 - Z_0) \geq \frac{\tau}{\tau+1} f(G_0)$, implying that $f(z_0) \geq \frac{f(G_0)}{\tau+1}$. Now, let Z_0' denote the minimizer of M_0 and note that if $z_0 \notin Z_0'$, we are done. Else, we have that,

$$\begin{aligned} g_\tau(M_0) = g_{\tau-1}(M_0 - z_0) &\geq \frac{\alpha}{\tau} g_{\tau-1}(OPT(\mathcal{I}_{z_0}, N - z_0, \tau - 1)) \\ &\geq \frac{\alpha}{\tau} g_{\tau-1}(OPT(\mathcal{I}, N, \tau) - z_0) \\ &\geq \frac{\alpha}{\tau} g_\tau(OPT(\mathcal{I}, N, \tau)) > \frac{\alpha}{\tau+1} g_\tau(OPT(\mathcal{I}, N, \tau)) \end{aligned}$$

Where the first inequality stems from the induction assumption, the second and third by our assumption on z_0. This was all true under the assumption that $z_0 \in OPT(\mathcal{I}, N, \tau)$, if that is not the case, we remove z_0 from the ground set and repeat the same process. The algorithm takes the best set out of all the ones generated, and hence there exists some iteration l such that $z_l \in OPT(\mathcal{I}, N, \tau)$ and analyzing that iteration as we did above, gives us the desired.

Finally, for the cardinality constraint case, we can avoid enumeration altogether and the simplified algorithm has runtime polynomial in (n, τ) and guarantee that scales as $\frac{1}{\tau}$, which for $\Omega(\sqrt{k}) \leq \tau = o(k)$, is a better guarantee than the naïve one of $\frac{1}{k-\tau}$ from Section 2.2. □

5 Conclusion, Open Problems and Further Work

We looked at a robust version of the classical monotone submodular function maximization problem, where we want sets that are robust to the removal of any τ elements. We introduced the special, yet insightful case of copies, for which we gave a fast and asymptotically $(1 - 1/e)$ approximate algorithm for $\tau = o(k)$.

For the general case, where we may not have copies, we gave a deterministic asymptotically $(1 - 1/e - 1/\Theta(m))$ algorithm for $\tau = 1$, with the runtime scaling as n^{m+1}. As a byproduct, we also developed a deterministic $(1 - 1/e) - 1/\Theta(m)$ approximation algorithm for bi-objective monotone submodular maximization, subject to cardinality constraint. For larger but constant τ, we gave a randomized $(1-1/e)-\epsilon$ approximation and conjectured that this could be made deterministic. Additionally, we also gave a fast and practical 0.387 algorithm for $\tau = o(\sqrt{k})$. Note that here, unlike in the special case of copies, we could not tune the algorithm to work for larger τ and in fact, there has been further work in this direction since the appearance of the conference and arXiv version of this paper. Notably, [6] generalizes the notion of geometrically reducing the number of copies from Section 3.1.2, and achieves a 0.387 approximation for $\tau = o(k)$. It is still open whether one can go all the way up to $(1-1/e)$ and no constant factor approximation or inapproximability result is known for $\tau = \Theta(k)$.

Finally, similar robustness versions can be considered for maximization subject to independence system constraints and we gave an enumerative black box approach that leads to an $\frac{\alpha}{\tau+1}$ approximation algorithm with query time scaling as $n^{\tau+1}$, given an α approximation algorithm for the non-robust case.

Acknowledgement

The authors would like to thank all the anonymous reviewers for their useful suggestions and comments on all the versions of the paper so far. In addition, RU would also like to thank Jan Vondrák for a useful discussion and pointing out a relevant result in [10].

References

1. A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *SODA '14*, pages 1497–1514. SIAM, 2014.
2. A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.
3. D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.
4. D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3):49–71, 2003.
5. D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.
6. I. Bogunovic, S. Mitrovic, J. Scarlett, and V. Cevher. Robust submodular maximization: A non-uniform partitioning approach. In *ICML*, 2017.
7. N. Buchbinder and M. Feldman. Deterministic algorithms for submodular maximization problems. *CoRR*, abs/1508.02157, 2015.
8. N. Buchbinder, M. Feldman, J.S. Naor, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. FOCS '12, pages 649–658, 2012.
9. G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.
10. C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS 10*, pages 575–584. IEEE, 2010.
11. S. Dobzinski and J. Vondrák. From query complexity to computational complexity. In *STOC '12*, pages 1107–1116. ACM, 2012.
12. U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
13. U. Feige, V.S. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.
14. M. Feldman, J.S. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS '11*, pages 570–579. IEEE.
15. M. Feldman, J.S. Naor, and R. Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm. In *Automata, Languages and Programming*, pages 342–353. Springer, 2011.
16. S.O. Gharan and J. Vondrák. Submodular maximization by simulated annealing. In *SODA '11*, pages 1098–1116. SIAM.
17. A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM, 2006.
18. D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artificial Intelligence Research*, 2011.
19. C. Guestrin, A. Krause, and A.P. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.
20. S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

21. A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.
22. A. Krause, H B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801, 2008.
23. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.
24. Y. Liu, K. Wei, K. Kirchhoff, Y. Song, and J. Bilmes. Submodular feature selection for high-dimensional acoustic score spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7184–7188. IEEE, 2013.
25. G.L. Nemhauser and L.A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.
26. G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.
27. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.
28. M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.
29. M. Thoma, H. Cheng, A. Gretton, J. Han, HP. Kriegel, A.J. Smola, L. Song, S.Y. Philip, X. Yan, and K.M. Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *SDM*, pages 1076–1087. SIAM, 2009.
30. J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC '08*, pages 67–74. ACM.
31. J. Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304, 2013.
32. J. Vondrák, C. Chekuri, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *STOC '11*, pages 783–792. ACM, 2011.

A Appendix

A.1

Lemma 17 (Nemhauser, Wolsey [25, 26]) *For all $\alpha \geq 0$, greedy algorithm terminated after αk steps yields a set A with $f(A) \geq \beta(0, \alpha) f(OPT(k, N, 0))$.*

Proof Let A_i be the set at iteration i of the greedy algorithm. Then by monotonicity, we have,

$$f(A_i \cup OPT(k, N, 0)) \geq f(OPT(k, N, 0))$$

and by submodularity,

$$\sum_{e \in OPT(k,N,0) - A_i} f(e|A_i) \geq f(OPT(k, N, 0)|A_i) \geq f(OPT(N, k, 0)) - f(A_i)$$

Hence, there exists an element e in $OPT(k, N, 0) - A_i$ such that,

$$f(e|A_i) \geq (f(OPT(k, N, 0)) - f(A_i))/k$$

Hence, we get the recurring inequality,

$$f(A_{i+1}) \geq f(A_i + e) \geq f(A_i) + (f(OPT(k, N, 0)) - f(A_i))/k$$

The above implies that the difference between the value of the greedy set and optimal solution decreases by a factor of $(1 - 1/k)$ at each step, so after αk steps,

$$\begin{aligned} f(A_{\alpha k}) &\geq (1 - (1 - 1/k)^{\alpha k}) f(OPT(k, N, 0)) \\ \implies f(A_{\alpha k}) &\geq \beta(0, \alpha) f(OPT(k, N, 0)) \end{aligned}$$

A.2

Lemma 18 *There exists no polytime algorithm with approximation ratio greater than* $(1 - 1/e)$ *for* $P2$ *unless* $P = NP$*. For the value oracle model, we have the same threshold, but for algorithms that make only a polynomial number of queries.*

Proof We will give a strict reduction from the classical problem $P1$ (for which the above hardness result holds [12,26]) to the robust problem $P2$. Consider an instance of $P1$, denoted by $(k, N, 0)$. We intend to reduce this to an instance of $P2$ on an augmented ground set $N \cup X$ i.e. $(k + \tau, N \cup X, \tau)$.

The set $X = \{x_1, \cdots, x_\tau\}$ is such that $f(x_i) = (k + 1)f(a_1)$ (and recall that $f(a_1) \geq f(a_i), \forall a_i \in N$) and $f(x_i|S) = f(x_i)$ for every i and $S \subset N \cup X$ not containing x_i. We will show that $g(OPT(k + \tau, N \cup X, \tau)) = f(OPT(k, N, 0))$.

First, note that for an arbitrary set $S = S_N \cup S_X$, such that $|S| = k + \tau$ and $S_X = S \cap X$, we have that every minimizer contains S_X. This follows by definition of X, since for any two subsets P, Q of S with $|P| = |Q| = k$ and P disjoint with X but $Q \cap X \neq \emptyset$, we have by monotonicity $f(Q) \geq f(x_i) = (k + 1)f(a_1) > kf(a_1)$ and by submodularity $kf(a_1) \geq f(P)$. This implies that X is the minimizer of $OPT(k, N, 0) \cup X$ and hence $f(OPT(k, N, 0)) \leq g(OPT(k + \tau, N \cup X, \tau))$

For the other direction, consider the set $OPT(k + \tau, N \cup X, \tau)$ and define,

$$M = OPT(k + \tau, N \cup X, \tau) \cap X$$

Next, observe that carving out an arbitrary set B of size $\tau - |M|$ from $OPT(k + \tau, N \cup X, \tau) - M$ will give us the set

$$C = OPT(k + \tau, N \cup X, \tau) - M - B$$

of size $k + \tau - (|M| + \tau - |M|) = k$. Also note that by design, $C \subseteq N$ and hence $f(C) \leq f(OPT(k, N, 0))$, but by definition, we have that $g(OPT(k + \tau, N \cup X, \tau)) \leq f(C)$. This gives us the other direction and we have $g(OPT(k + \tau, N \cup X, \tau)) = f(OPT(k, N, 0))$.

To complete the reduction we need to show how to obtain an α-approximate solution to $(k, N, 0)$ given an α-approximate solution to $(k + \tau, N \cup X, \tau)$. Let $S = S_N \cup S_X$ be such a solution i.e. a set of size $k+\tau$ with $S_X = S \cap X$, such that $g(S) \geq \alpha g(OPT(k+\tau, N \cup X, \tau))$. Now consider an arbitrary subset S'_N of S_N of size $\tau - |S_X|$. Observe that $|S_N - S'_N| = |S| - |S_X| - (\tau - |S_X|) = k$ and further $f(S_N - S'_N) \geq g(S) \geq \alpha g(OPT(k + \tau, N \cup X, \tau)) = \alpha f(OPT(k, N, 0))$, by definition. Hence the set $S_N - S'_N \subseteq N$ is an α-approximate solution to $(k, N, 0)$ that, given S, can be obtained in polynomial time/queries. □

A.3 Tight analysis of Algorithm 3

Theorem 19 *The 0.387-algorithm is* $\frac{1}{2}\beta(0.5, \frac{k-2}{k-1})$($> 0.387$ *asymptotically) approximate.*

Proof Let $OPT = g(OPT(k, N, 1))$, A be the output of the 0.387-algorithm and a'_1 be the first element added to A apart from a_1. The case $z = a_1$ is straightforward since $f(A - a_1) \geq \beta(0, 1)f(OPT(k - 1, N - a_1, 0)) \geq \beta(0, 1)OPT$ where the last inequality follows from Lemma 2. So assume $z \neq a_1$. Further, let $f(z|A - a_1 - z) = \eta f(A - a_1)$ which implies that $f(a'_1) \geq f(z) \geq f(z|A - a_1 - z) = \eta f(A - a_1)$ and now from Lemma 5 with N replaced by $N - a_1$, A replaced by $A - a_1$ and thus k replaced by $k-1$, $S = a_1'$ with $s = 1$ and $l = k - 1 - |S| = k - 2$, we get,

$$f(A - a_1) \geq \beta(\eta, \frac{k-2}{k-1}) f(OPT(k-1, N - a_1, 0))$$

This together with Lemma 2 implies, $f(A - a_1) \geq \beta(\eta, \frac{k-2}{k-1})OPT$. Also, we have by definition,

$$f(A - a_1 - z) = (1-\eta) f(A - a_1) \geq (1-\eta)\beta(\eta, \frac{k-2}{k-1})OPT$$

Further, we have,

$$\begin{aligned} g(A) &\geq \max\{f(a_1), f(A - a_1 - z)\} \\ &\geq \max\{f(z|A - a_1 - z), f(A - a_1 - z)\} \\ &\geq \max\{\eta\beta(\eta, \frac{k-2}{k-1}), (1-\eta)\beta(\eta, \frac{k-2}{k-1})\}OPT \\ &\geq 0.5\beta(0.5, \frac{k-2}{k-1})OPT \quad [\text{for } \eta = 0.5] \\ &\xrightarrow{k\to\infty} 0.387OPT \end{aligned}$$

We now give an instance where the above analysis is tight. Let the algorithm start with a maximum value element a_1, then pick a_2, and then add the set C, such that the output of the algorithm is $a_1 \cup a_2 \cup C$, with C being a set of size $k-2$. Let $f(a_1) = 1, f(a_2) = 1, f(C) = 1$ with $f(a_1 + C) = 1, f(a_1 + a_2) = 2, f(a_2 + C) = 2$ i.e. C copies a_1. Hence $f(a_1 + a_2 + C) = 2$ and $g(a_1 + a_2 + C) = f(a_1 + C) = 1$.

Let $OPT(k, N, 1)$ include a_2, a copy a_2' of a_2 (so $f(a_2') = 1, f(a_2 + a_2') = 1$) and a set D of $k-2$ elements of value $\frac{1}{(k-2)\beta(0,1)}$ each, such that $f(OPT(k, N, 1)) = 1 + (k-2)\frac{1}{(k-2)\beta(0,1)} = 1 + \frac{e}{e-1} = \frac{2}{\beta(0.5,1)}$. Observe that the small value elements are all minimizers and $g(OPT(k, N, 1)) \approx \frac{2}{\beta(0.5,1)}$ as k becomes large. Note that $f(D) = \frac{f(C)}{\beta(0,1)}$ and we can have sets C and D as above based on the worst case example for the greedy algorithm given in [26].This proves that the inequality in Lemma 5 is tight.

A.4 Analysis of Algorithm 4

Theorem 20 *Algorithm 4 is* $0.5547 - \Omega(1/k)$ *approximate.*

Proof Let A denote the output and $A_0 \subset A$ denote $\{a_1, a_2\}$. Due to submodularity, there exists at most two distinct $x \in A$ with $f(x|A - x) > \frac{f(A)}{3}$. Additionally, for every $x \notin A_0$, we have that $f(x|S) \leq f(a_1)$ and $f(x|S) \leq f(a_2|a_1)$ for arbitrary subset S of A containing A_0 and $x \notin S$. This implies that that $2f(x|S) \leq f(A_0) \leq f(S)$, which gives us that $f(x|S) \leq \frac{f(S+x)}{3}$.

Note that due to condition in Phase 1, the algorithm ignores a_1 even if it is not a minimizer, as long as its marginal is more than a third the value of the set at that iteration. At the end of Phase 1, if a_2 has marginal more than third of the set value, then it is ignored until its contribution/marginal decreases. Phase 3 adds greedily (without ignoring any element added). As argued above, no element other than a_1, a_2 can have marginal more than a third of the set value at any iteration, so during Phase 2 we have that a_2 is also a minimizer.

We will now proceed by splitting into cases. Denote $g(OPT(k, N, 1))$ as OPT and recall from Lemma 2, $OPT \leq f(OPT(k-1, N - a_i, 0))$ for $i \in \{1, 2\}$. Also, let the set of elements added to A_0 during Phase 1 be $U = \{u_1, \ldots, u_p\}$, similarly elements added during Phases 2 and 3 be $V = \{v_1, \ldots, v_q\}$,$W = \{w_1, \ldots, w_r\}$ respectively, with indexing in order of addition to the set. Finally, let $\alpha_p = \frac{p-2}{k-1}$, $\alpha_q = \frac{q-1}{k-1}$, $\alpha_r = \frac{r}{k-1}$, $\alpha = \alpha_p + \alpha_q + \alpha_r = \frac{k-5}{k-1}$, and assume $k \geq 8$.

Case 1: Phase 2,3 do not occur i.e. $p = k-2, q = r = 0$.
Since we have,

$$\begin{aligned} f(A - a_1) &\overset{(a)}{\geq} f(a_2) + \beta\Big(0, \frac{k-2}{k-1}\Big)(f(OPT(k-1, N-a_1, 0)) - f(a_2)) \\ &\geq \beta\Big(0, \frac{k-2}{k-1}\Big)f(OPT(k-1, N-a_1, 0)) \\ &\overset{(b)}{\geq} \beta\Big(0, \frac{k-2}{k-1}\Big)OPT, \end{aligned} \tag{16}$$

where (a) follows from Lemma 1 and (b) from Lemma 2. This deals with the case $z = a_1$. If $z = u_p$, we have,

$$\begin{aligned} f(A - u_p) \geq f(A - u_p - a_1) &\overset{(c)}{\geq} \beta\Big(0, \frac{k-3}{k-1}\Big)f(OPT(k-1, N-a_1, 0)) \\ &\geq \beta\Big(0, \frac{k-3}{k-1}\Big)OPT, \end{aligned}$$

where (c) is like (16) above but with $k-2$ replaced by $k-3$ when using Lemma 1. Finally, let $z \notin \{a_1, u_p\}$, then due to the Phase 1 termination criteria, we have $f(a_1|A - a_1 - u_p) \geq f(A - u_p)/3$, which implies that,

$$2f(a_1|A - a_1 - u_p) \; \geq f(A - a_1 - u_p) \tag{17}$$

Now letting $\eta = \frac{f(z|A-a_1-u_p-z)}{f(A-a_1-u_p)}$, we have by submodularity $f(z|A-z) \leq \eta f(A - a_1 - u_p)$ and by definition $f(A - a_1 - u_p - z) = (1-\eta)f(A - a_1 - u_p)$. Using the above we get,

$$\begin{aligned} &f(A - z) \overset{(d)}{\geq} \max\{f(a_1), f(A - u_p - z)\} \\ &\geq \max\{f(z|A - a_1 - u_p - z), f(A - a_1 - u_p - z) + f(a_1|A - a_1 - u_p - z)\} \\ &\geq \max\{\eta f(A - a_1 - u_p), (1-\eta)f(A - a_1 - u_p) + f(a_1|A - a_1 - u_p)\} \\ &\geq \max\{\eta, (1-\eta) + \frac{1}{2}\}f(A - a_1 - u_p) \end{aligned} \tag{18}$$

where (d) follows from monotonicity and the fact that $a_1 \in A - z$ and $A - u_p - z \subset A - z$. Now, from Lemma 5 with $S = \{a_2, u_1\}$, $l = p-2$, k replaced by $k-1$, N by $N - a_1$ and $s = 1$, we have, $f(A - a_1 - u_p) \geq \beta(\eta, \alpha_p)f(OPT(k-1, N-a_1, 0)) \geq \beta(\eta, \alpha_p)OPT$. Substituting this in (18) above we get,

$$\begin{aligned} f(A - z) \;\; &\max\{\eta, \frac{3}{2} - \eta\}\beta(\eta, \alpha_p)OPT \\ &\geq \frac{3}{4}\beta\Big(\frac{3}{4}, \alpha_p\Big)OPT \quad [\eta = 3/4] \\ &> \beta(0, \alpha_p)OPT = \beta\Big(0, \frac{k-4}{k-1}\Big)OPT \end{aligned} \tag{19}$$

Case 2: Phase 2 occurs, 3 doesn't i.e. $p + q = k-2$ and $q > 0$.

As stated earlier, during Phase 2, a_2 is the minimizer of $A_0 \cup U \cup (V - v_q)$. We have $g(A) \geq g(A - v_q) = f(A - v_q - a_2) = f(a_1 + U) + f(V - v_q|a_1 + U)$. Further, since the addition rule in Phase 2 ignores a_2, we have from Lemma 1, $f(V - v_q|a_1 + U) \geq \beta(0, \alpha_q)(OPT - f(a_1 + U))$, and $f(a_1 + U) \geq \beta(0, \alpha_p)OPT$ follows from the previous case (to see this, suppose that the algorithm was terminated after Phase 1 and note that $z = a_2$ falls under the scenario

$z \notin \{a_1, u_p\}$). Using this,

$$\begin{aligned} g(A - v_q) &\geq f(a_1 + U) + \beta(0, \alpha_q)(OPT - f(a_1 + U)) \\ \frac{f(A - z)}{OPT} &\geq (1 - \beta(0, \alpha_q))\beta(0, \alpha_p) + \beta(0, \alpha_q) \\ &= \beta(0, \alpha) = \beta\Big(0, \frac{k-5}{k-1}\Big) \end{aligned} \tag{20}$$

Case 3: Phase 3 occurs i.e., $r > 0$.

We consider two sub-cases, $z \in A - W$ and $z \in W$. Suppose $z \in A - W$. Due to $f(z|A-W-z) \leq f(A-W)/3$ we have, $f(A-W) \leq \frac{3}{2}f(A-W-z)$. Also, $f(W|A-W-z) \geq f(W|A-W) \geq \beta(0, \alpha_r)(OPT - f(A-W))$. Then using this along with the previous cases,

$$\begin{aligned} f(A - z) &= f(A - W - z) + f(W|A - W - z) \\ &\geq f(A - W - z) + \beta(0, \alpha_r)(OPT - f(A - W)) \\ &\geq f(A - W - z) + \beta(0, \alpha_r)(OPT - \frac{3}{2}f(A - W - z)) \\ &\geq (1 - \frac{3}{2}\beta(0, \alpha_r))f(A - W - z) + \beta(0, \alpha_r)OPT \\ &\geq (1 - \frac{3}{2}\beta(0, \alpha_r))\beta(0, \alpha_p + \alpha_q)OPT + \beta(0, \alpha_r)OPT \quad \text{[from (19),(20)]} \\ \frac{f(A - z)}{OPT} &\geq 0.5 - \frac{3}{2e^{\alpha}} + \frac{1}{2}(e^{-(\alpha_p + \alpha_q)} + e^{-\alpha_r}) \\ &\geq 0.5 - \frac{3}{2e^{\alpha}} + e^{-\alpha/2} \quad \text{[for } \alpha_r = \alpha_p + \alpha_q = \alpha/2\text{]} \\ &= 0.5 - \frac{3}{2e^{\frac{k-4}{k-1}}} + e^{-\frac{k-4}{2(k-1)}} \xrightarrow{k \to \infty} 0.5547 \end{aligned}$$

Now, suppose $z \in W$, then note that for $p + q \geq 6$ we have either $p \geq 3$, and hence due to greedy additions $f(z|A - z) \leq f(\{u_1, u_2, u_3\})/3 \leq f(A - W)/3$, or $q \geq 3$, and again due to greedy additions $f(z|A - z) \leq f(a_1 + U \cup \{v_1, v_2\})/3 \leq f(A - W)/3$.

If $q > 0$, then note that $f(A - W) \geq f(A - W - v_q) \geq \frac{3}{2}f(A - W - v_q - a_2)$ due to the Phase 2 termination conditions. Now we reduce the analysis to look like the previous sub-case through the following,

$$\begin{aligned} f(A - z) &= f(A - W) + f(W|A - W) - f(z|A - z) \\ &\geq f(A - W) + \beta(0, \alpha_r)(OPT - f(A - W)) - f(z|A - z) \\ &\geq (1 - \beta(0, \alpha_r))f(A - W) + \beta(0, \alpha_r)OPT - f(A - W)/3 \\ &\geq \Big(1 - \frac{3}{2}\beta(0, \alpha_r)\Big)\frac{2}{3}f(A - W) + \beta(0, \alpha_r)OPT \\ &\geq \Big(1 - \frac{3}{2}\beta(0, \alpha_r)\Big)f(A - W - v_q - a_2) + \beta(0, \alpha_r)OPT \end{aligned}$$

Which since $f(A - W - v_q - a_2) \geq \beta\Big(0, \frac{p+q-3}{k-1}\Big)OPT$ from (20), leads to the same ratio asymptotically as when $z \in A - W$. The case $q = 0$ can be dealt with similarly by using $f(A - W) \geq f(A - W - u_p) \geq \frac{3}{2}f(A - W - u_p - a_1)$

If $p + q < 6$, then let $f(z|A - z) = \eta f(A)$. Now we have, $f(A - W) \geq f(A_0) = f(a_1) + f(a_2|a_1) \geq 2f(z|A-z) = 2\eta f(A)$, which further implies that $f(A-W+w_1) \geq 3\eta f(A)$ since $z \in W$. Then proceeding as in Lemma 5 with k replaced by $k - 1$, $S = A - W + w_1$ and hence $s = 3$ and finally $l = k - |S| = k - (p + q + 2 + 1) \leq k - 8$ gives us,

$$f(A) \geq \beta(3\eta, \frac{k-8}{k-1})f(OPT(k - 1, N, 0)) \geq \beta(3\eta, \frac{k-8}{k-1})OPT$$

Then using Lemma 6 we have, $f(A - z) \geq (1 - \eta)\beta(3\eta, \frac{k-8}{k-1})OPT \geq \beta(0, \frac{k-8}{k-1})OPT$. □