Submodular Minimax Optimization: Finding Effective Sets

Loay Mualem
University of Haifa

Ethan R. Elenberg
ASAPP

Moran Feldman
University of Haifa

Amin Karbasi
Yale University

Abstract

Despite the rich existing literature about minimax optimization in continuous settings, only very partial results of this kind have been obtained for combinatorial settings. In this paper, we fill this gap by providing a characterization of submodular minimax optimization, the problem of finding a set (for either the min or the max player) that is effective against every possible response. We show when and under what conditions we can find such sets. We also demonstrate how minimax submodular optimization provides robust solutions for downstream machine learning applications such as (i) prompt engineering in large language models, (ii) identifying robust waiting locations for ride-sharing, (iii) kernelization of the difficulty of instances of the last setting, and (iv) finding adversarial images. Our experiments show that our proposed algorithms consistently outperform other baselines.

1 INTRODUCTION

Many machine learning tasks, ranging from data selection to decision making, are inherently combinatorial and thus, require combinatorial optimization techniques that work at scale. Even though, in general, solving such problems is notoriously hard, practical problems are very often endowed with extra structures that lend them to optimization techniques. One common structure is submodularity, a condition that holds either exactly or approximately in a wide range of machine learning applications, including: dictionary selection (Krause and Cevher, 2010), sparse recovery, feature selection (Das and Kempe, 2011), neural network interpretability (Elenberg et al., 2017), crowd teaching (Singla et al., 2014), human brain mapping (Salehi et al., 2017), data summarizarion (Lin and Bilmes, 2011; Mualem and Feldman, 2022a), among many others. Submodular functions are often considered to be discrete analogs of concave functions, and like concave functions they can be (approximately) maximized. At the same time, submodular functions can also be exactly minimized as they can be extended into an efficiently computable continuous convex function (known as the Lovász extension). These optimization properties of submodular functions has been often exploited in scalable machine learning algorithms.

While scalable optimization methods are desirable, they are not the only requirements for ML algorithm deployment. Very often, it is also important to get solutions that are robust with respect to noise, outliers, adversarial examples, etc. Problems looking for solutions that are robust with respect to worst-case scenarios have usually been expressed as minimax optimization. Accordingly, recent years have witnessed a large body of work addressing minimax optimization in the continuous settings (see, e.g., Diakonikolas et al. (2021); Ibrahim et al. (2020); Lin et al. (2020); Mokhtari et al. (2020)). This line of research has given rise to a myriad of algorithms, and an ever increasing list of applications such as adversarial attack generation (Wang et al., 2021), robust statistics (Agarwal and Zhang, 2022) and multi-agent systems (Li et al., 2019), to name a few. To ensure feasibility of finding a saddle point, one has to make some structural assumptions. For instance, many of the above-mentioned works assume that the minimization is taken over a convex function, and the maximization over a concave function.

Despite the rich existing literature about minimax optimization in continuous settings, very few works have managed to obtain similar results for combinatorial settings. Staib et al. (2019) and Adibi et al. (2022) considered hybrid settings in which the maximization is done with respect to a (discrete) submodular function, but the minimization is still done over a continuous domain. Krause et al. (2008), Torrico et al. (2021) and

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

Iyer (2019, 2020) considered settings in which both the maximization and the minimization are discrete, but one of them is done over a small domain that can be efficiently enumerated. In this paper we provide the first systematic study of the natural case of fully discrete minimax optimization with maximization and minimization domains that can both be large. To the best of our knowledge, the only previous works relevant to this case are works of Bogunovic et al. (2017) and Orlin et al. (2018), who studied the maximization of a monotone submodular function subject to a cardinality constraint in the presence of a worst case (represented by a minimization) removal of a small number of elements from the chosen solution.

As submodular functions cannot be maximized exactly, there is no hope to get a saddle point in our setting. Instead, like Adibi et al. (2022), we take a game theoretic perspective on the setting. From this point of view, there are two players. Each player selects a set, and the objective function value is determined by the sets selected by both players. One of the players aims to minimize the objective function, while the other player wishes to maximize it. Our task is to select for one of the players (either the minimization or the maximization player) a set that is *effective* in the sense that it guarantees a good objective value regardless of the set chosen by the other player.

We map the tractability and approximability of the above minimax submodular optimization task as function of various properties, such as: the player considered (minimization or maximization), the constraints (if any) on the sets that can be chosen by the players, and whether the objective function is submodular as a whole, or for each player separately. We refer the reader to Section 2 for our exact results. However, in a nutshell, we have fully mapped the approximability for the minimization player, and we also have non-trivial results for the maximization player.

Our proposed algorithms for minimax submodular optimization can lead to finding of robust solutions for down-stream machine learning applications, including efficient prompt engineering, ride-share difficulty kernalization, adversarial attacks on image summarization and robust ride-share optimization. Empirical evaluation of our algorithms in the context of all the above applications can be found in Section 3.

1.1 Related Work

Submodular Minimization The first polynomial time algorithm for (unconstrained) submodular minimization was obtained by Grötschel et al. (1981) using the ellipsoids method. Almost twenty years later, Schrijver (2000) and Iwata et al. (2001) obtained, independently, the first strongly polynomial time (and combinatorial) algorithms for the problem. Further works have improved over the time complexities of the last algorithms, and the current state-of-the-art algorithm was described by Lee et al. (2015) (see also Axelrod et al. (2020) for a faster approximation algorithm for the problem).

All the above results apply to unconstrained submodular minimization. Unfortunately, constrained submodular minimization often (provably) admits only very poor approximation guarantees even when the constraint is as simple as a cardinality constraint (see, for example, Goel et al. (2010); Svitkina and Fleischer (2011)). Nevertheless, there are rare examples of constraints that allow for efficient submodular minimization, such as the constraint requiring the output set to be of even size (Goemans and Ramakrishnan, 1995).

Submodular Maximization A simple greedy algorithm obtains the optimal approximation ratio of $1 - 1/e$ for maximization of a monotone submodular function subject to a cardinality constraint (Nemhauser and Wolsey, 1978; Nemhauser et al., 1978). The same approximation ratio was later obtained for general matroid constraints via the continuous greedy algorithm (Călinescu et al., 2011). The best possible approximation ratio for unconstrained maximization of a non-monotone submodular function is $1/2$ (Feige et al., 2011; Buchbinder et al., 2015), even for deterministic algorithms (Buchbinder and Feldman, 2018). However, the approximability of constrained maximization of such functions is not as well understood. Following a long line of works (Buchbinder et al., 2014; Ene and Nguyen, 2016; Feldman et al., 2011; Lee et al., 2009; Oveis Gharan and Vondrák, 2011; Vondrák, 2013), the state-of-the-art algorithm for maximizing a non-monotone submodular function subject to a cardinality or matroid constraint guarantees 0.385-approximation (Buchbinder and Feldman, 2019), while the best inapproximability result for these constraints only shows that one cannot obtain 0.478-approximation for them (Gharan and Vondrák, 2011; Qi, 2022).

It is also worth mentioning a line of work (Mirzasoleiman et al., 2017; Mitrovic et al., 2017) aiming to find a small core set such that even if some elements are adversarially chosen for deletion, it is still possible to produce a good solution based on the core set. Note that this line of work differs from the maximization player point of view in our setting, in which the aim is to find a single solution for the maximization player that is good against every choice of the minimization player.

Additional related work relevant to some of our applications can be found in Appendix A.

2 NOTATION AND OUR THEORETICAL CONTRIBUTION

Let us describe the formal model for our setting. There are two (disjoint) ground sets $\mathcal{N}_1$ and $\mathcal{N}_2$, one ground set for each one of the players. For each ground set $\mathcal{N}_i$, we also have a constraint $\mathcal{F}_i \subseteq 2^{\mathcal{N}_i}$ specifying the sets that can be chosen from this ground set. Finally, there is a non-negative objective set function $f\colon 2^{\mathcal{N}_1 \uplus \mathcal{N}_2} \to \mathbb{R}_{\geq 0}$.[1] The minimization player gets to pick a set X from $\mathcal{F}_1$, and wishes to minimize the value of f, while the maximization players picks a set Y from $\mathcal{F}_2$, and aims to maximize the value of f. Our task is to find for each player a set S that yields the best value for f assuming the other player chooses the best response against S. In other words, for the minimization player we want to find a set $X \in \mathcal{F}_1$ that (approximately) minimizes $\max_{Y \in \mathcal{F}_2} f(X \uplus Y)$, and for the maximization player we should find a set $Y \in \mathcal{F}_2$ that (approximately) maximizes $\min_{X \in \mathcal{F}_1} f(X \uplus Y)$. As optimization of general set functions cannot be done efficiently, we must assume that the objective function f obeys some properties. Two common properties that are often considered in the literature are submodularity and monotonicity. However, to assume these properties, we first need to discuss what they mean in our setting.

Let us begin with the property of submodularity. In the following, given an element u and a set S we use $f(u \mid S) \triangleq f(S \cup \{u\}) - f(S)$ to denote the marginal contribution of u to the set S.[2] According to the standard definition of submodularity,[3] f is submodular if the inequality $f(u \mid S) \geq f(u \mid T)$ holds for every two sets $S \subseteq T \subseteq \mathcal{N}_1 \uplus \mathcal{N}_2$ and element $u \in (\mathcal{N}_1 \uplus \mathcal{N}_2) \setminus T$. Since this definition of submodularity treats $\mathcal{N}_1$ and $\mathcal{N}_2$ as two parts of one ground set, in the rest of this paper we call a function that obeys it *jointly-submodular*. However, since the ground sets $\mathcal{N}_1$ and $\mathcal{N}_2$ play very different roles in our problems, it makes sense to consider also functions that are submodular when restricted to one ground set. We say that f is submodular when restricted to $\mathcal{N}_1$ if it becomes a submodular function when we fix the set of elements of $\mathcal{N}_2$ chosen. More formally, f is submodular when restricted to $\mathcal{N}_1$ if the inequality $f(u \mid S \cup A_2) \geq f(u \mid T \cup A_2)$ holds for every $S \subseteq T \subseteq \mathcal{N}_1$ and $u \in \mathcal{N}_1 \setminus T, A_2 \subseteq \mathcal{N}_2$. The definition of being submodular when restricted to $\mathcal{N}_2$ is analogous, and we say that f is *disjointly-submodular* if it is submodular when restricted to either $\mathcal{N}_1$ or $\mathcal{N}_2$.

[1] We use $\uplus$ to denote the union of disjoint sets.

[2] Similarly, given sets S and T, $f(T \mid S) \triangleq f(S \cup T) - f(S)$ denotes the marginal contribution of T to S.

[3] A set function $g\colon \mathcal{N} \to \mathbb{R}$ is submodular if $g(u \mid S) \geq g(u \mid T)$ for every $S \subseteq T \subseteq \mathcal{N}$ and $u \in \mathcal{N} \setminus T$.

Unfortunately, submodular minimization admits very poor approximation guarantees even subject to simple constraints such as cardinality (see Section 1.1 for more details). Therefore, we restrict attention to the case of $\mathcal{F}_1 = 2^{\mathcal{N}_1}$. Given this restriction, we cannot assume that f is monotone[4] since this will guarantee that the best choice for the set X is always $\varnothing$. However, some of our results assume that f is monotone with respect to the elements of $\mathcal{N}_2$. In other words, we say that f is $\mathcal{N}_2$-*monotone* if the inequality $f(u \mid S) \geq 0$ holds for every $S \subseteq \mathcal{N}_1 \uplus \mathcal{N}_2$ and $u \in \mathcal{N}_2 \setminus S$.

Table 1 summarizes the theoretical results proved in this paper. When the table states that we have an inapproximability result of c for a problem, it means that no polynomial time algorithm can produce a value that with probability at least $2/3$ approximates the exact value of this problem up to a factor of c. For example, if we look at the optimization problem $\min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y)$, then an inapproximability result of c means that no polynomial time algorithm can produce a value v that obeys

$$v \leq c \cdot \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y) \leq c \cdot v \tag{1}$$

with probability at least $2/3$. In contrast, we hold our algorithms to a higher standard. Specifically, when Table 1 states that we have a c-approximation algorithm for a problem, it means that the algorithm is able to produce with probability at least $2/3$ two things: a value v of the above kind, and a solution set S for the external min or max operation that leads to c-approximation when the internal min or max is optimality solved. For example, given the above optimization problem, a c-approximation algorithm produces with probability at least $2/3$ a value v obeying Equation (1), and a set $S \subseteq \mathcal{N}_1$ such that $\min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y) \leq \max_{Y \in \mathcal{F}_2} f(S \uplus Y) \leq c \cdot \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y)$.

The success probability of $2/3$ in the above definitions can always be increased via repetitions. However, such repetitions can usually be avoided since our algorithms are typically either deterministic or naturally have a high success probability.

As is standard in the literature, we assume that access to the objective function f is done via a value oracle that given a set $S \subseteq \mathcal{N}_1 \uplus \mathcal{N}_2$ returns $f(S)$. Furthermore, given a set S and element u, we use $S + u$ and $S - u$ to denote $S \cup \{u\}$ and $S \setminus \{u\}$, respectively.

2.1 Results for $\max\min$ Optimization

For $\max\min$ expressions (the problem of the maximization player) we have a good understanding of the

[4] A set function $g\colon 2^{\mathcal{N}} \to \mathbb{R}$ is monotone if $g(S) \leq g(T)$ for every two sets $S \subseteq T \subseteq \mathcal{N}$.

Table 1: Our theoretical results. We denote by α the approximation ratio that can be obtained for maximizing a non-negative submodular function subject to $\mathcal{F}_2$. If f happens to be $\mathcal{N}_2$-monotone, then α can be improved to be the approximation ratio that can be obtained for maximizing a non-negative *monotone* submodular function subject to $\mathcal{F}_2$.

Expression to approximate	**Assumptions**	**Result proved**
$\max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$	jointly-submodular	$(\alpha+\varepsilon)$-approx. alg. (Thm 2.1)
$\max_{Y \subseteq \mathcal{N}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$	disjointly-submodular	No finite approximation ratio possible unless $BPP = NP$ (Thms 2.2 and 2.3)
$\max_{Y \subseteq \mathcal{N}_2, \lvert Y\rvert \leq k} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$	disjointly-submodular, $\mathcal{N}_2$-monotone	
$\min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2} f(X \cup Y)$	disjointly-submodular	$(4+\varepsilon)$-approx. alg. (Thm 2.5)
$\min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \cup Y)$	jointly-submodular, $\varnothing \in \mathcal{F}_2$	$O(\alpha\sqrt{\lvert\mathcal{N}_1\rvert})$-approx. alg. (Thm. 2.6)
$\min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \cup Y)$	disjointly-submodular, $\{u\} \in \mathcal{F}_2 \ \forall u \in \mathcal{N}_2$	$O(\lvert\mathcal{N}_2\rvert)$-approx. alg. (Thm 2.4)

approximability, and it turns out that this approximability strongly depends on the kind of submodularity guaranteed for f. If f is jointly-submodular, then the problem admits roughly the same approximation ratio as the maximization problem obtained by omitting the min operation.

Theorem 2.1. *Assume that there exists an α-approximation algorithm ALG for the problem of maximizing a non-negative submodular function g subject to $\mathcal{F}_2$. Then, for every polynomially small $\varepsilon \in (0, \alpha]$, there exists a polynomial time algorithm that (i) outputs a set $\hat{Y} \in \mathcal{F}_2$ and the value $\min_{X \subseteq \mathcal{N}_1} f(X \cup \hat{Y})$; and (ii) guarantees that, with probability at least $2/3$, $\min_{X \subseteq \mathcal{N}_1} f(X \cup \hat{Y})$ falls within the range $[\tau/(\alpha+\varepsilon), \tau]$, where $\tau = \max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$. Furthermore, if f is $\mathcal{N}_2$-monotone, then it suffices for ALG to obtain α-approximation when g is guaranteed to be monotone (in addition to being non-negative and submodular).*

We note that by assuming in Theorem 2.1 that ALG is an α-approximation algorithm, we only mean that the expected value of the solution of ALG is smaller than the value of the optimal solution by at most a factor of α. In other words, we do not make any high probability assumption on ALG. The proof of Theorem 2.1 is based on the observation that the joint-submodularity of f implies that $\min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$ is a submodular function of Y. See Section B.1 for details.

Unfortunately, it turns out that when f is only disjointly submodular, there is little an algorithm can guarantee. The following theorems show this for two basic special cases: unconstrained maximization, and maximization subject to a cardinality constraint of an $\mathcal{N}_2$-monotone function (the special case of unconstrained maximization of an $\mathcal{N}_2$-monotone function is trivial since it is always optimal to set $Y = \mathcal{N}_2$ in this case). Both theorems are proved using a reduction showing that the minimization over X can be replaced with a minimization over multiple functions, which allows us to capture well-known NP-hard problems with max min expressions. See Section B.2 for details.

Theorem 2.2. *When f is only guaranteed to be non-negative and disjointly submodular, no polynomial time algorithm for calculating $\max_{Y \subseteq \mathcal{N}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$ has a finite approximation ratio unless $BPP = NP$.*

Theorem 2.3. *When f is only guaranteed to be non-negative, $\mathcal{N}_2$-monotone and disjointly submodular, no polynomial time algorithm for calculating $\max_{Y \subseteq \mathcal{N}_2, |Y| \leq \rho} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$, where ρ is a parameter of the problem, has a finite approximation ratio unless $BPP = NP$.*

2.2 Results for $\min\max$ Optimization

We also have some results for min max expressions (the problem of the minimization player), although our understanding of the approximability of such expressions is worse than for max min expressions. We begin with the following theorem, which shows that when all the singleton subsets of $\mathcal{N}_1$ are feasible choices for the min operation (which is the case, for example, when the constraint is down-closed), it is possible to get a finite approximation (specifically, $O(|\mathcal{N}_2|)$-approximation) for min max. The proof of Theorem 2.4 can be found in Section C.1. In a nutshell, it is based on the observation for a set $X \subseteq \mathcal{N}_1$ the sum $f(X) + \sum_{u \in \mathcal{N}_2} f(X \cup \{u\})$ is an easy to calculate submodular function of X that gives $O(|\mathcal{N}_2|)$-approximation for $\max_{Y \subseteq \mathcal{N}_2} f(X \cup Y)$.

Theorem 2.4. *Assuming $\{u\} \in \mathcal{F}_2$ for every $u \in \mathcal{N}_2$, there is a polynomial time algorithm that, given a non-negative disjointly submodular function $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$, returns a set $\hat{X} \subseteq \mathcal{N}_1$ and a value v such that both $\max_{Y \in \mathcal{F}_2} f(\hat{X} \cup Y)$ and v fall within the range $[\tau, (|\mathcal{N}_2| + 1) \cdot \tau]$, where $\tau \triangleq \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \cup Y)$.*

The approximation ratio of the last theorem can be improved to a constant when the max operation is unconstrained (like the min operation). The following theorem states this formally, and its proof can be found in Section C.2. The proof is based on using samples of $\mathcal{N}_2$ to construct a random easy to calculate submodular function of X approximating $\max_{Y \subseteq \mathcal{N}_2} f(X \uplus Y)$ up to a factor of roughly 4.

Theorem 2.5. *For every constant $\varepsilon \in (0, 1)$, there exists a polynomial time algorithm that given a non-negative disjointly submodular function $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ returns a set $\hat{X} \subseteq \mathcal{N}_1$ and a value v such that the expectations of both $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \uplus Y)$ and v fall within the range $[\tau, (4+\varepsilon)\tau]$, where $\tau \triangleq \min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2} f(X \uplus Y)$. Furthermore, the probability that both $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \uplus Y)$ and v fall within this range is at least $1 - O(|\mathcal{N}_2|^{-1})$.*

The factor of $4+\varepsilon$ in the last theorem improves to $2+\varepsilon$ when f is symmetric with respect to $\mathcal{N}_2$, i.e., when $f(X \uplus Y) = f(X \uplus (\mathcal{N}_2 \setminus Y))$ for every two sets $X \subseteq \mathcal{N}_1$ and $Y \subseteq \mathcal{N}_2$.

It is interesting to note that the last two results show a separation between min max and max min optimization as no finite approximation guarantee can be obtained for disjointly submodular functions in the later case (Theorems 2.2 and 2.3). Our last result obtains a sub-linear approximation guarantee for an (almost) general constraint $\mathcal{F}_2$; however, this comes at the cost of requiring f to be jointly-submodular.

Theorem 2.6. *Assuming $\varnothing \in \mathcal{F}_2$, there exists a polynomial time algorithm that gets as input (i) a non-negative jointly submodular function $f \colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$, and (ii) an oracle that given a set $X \subseteq \mathcal{N}_1$ returns a set $Y \in \mathcal{F}_2$ that maximizes $f(X \uplus Y)$ up to a factor of $\alpha \geq 1$ among such sets,[5] and given this input returns a set $\hat{X} \subseteq \mathcal{N}_1$ and a value v such that both $\max_{Y \in \mathcal{F}_2} f(\hat{X} \uplus Y)$ and v are lower bounded by τ and upper bounded by $O(\alpha\sqrt{|\mathcal{N}_1|}) \cdot \tau$, where $\tau \triangleq \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y)$.*

Below, we prove Theorem 2.6. In this proof, we denote by X^* an arbitrary set in $\arg\min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y)$. Notice that the definitions of X^* and τ imply together that $\tau = \max_{Y \in \mathcal{F}_2} f(X^* \uplus Y)$. Thus, $f(X^* \uplus Y) \leq \tau$ for every set $Y \in \mathcal{F}_2$, and in particular, since $\varnothing \in \mathcal{F}_2$ by assumption, $f(X^*) \leq \tau$.

[5]Theorem 2.6 assumes an oracle that never fails. Such an oracle can be implemented by a deterministic α-approximation algorithm, or a randomized algorithm that maximizes $f(X \uplus Y)$ up to a factor of α with high probability (in the later case, the algorithm guaranteed by the theorem also succeeds only with high probability). If one only has a randomized algorithm guaranteeing α-approximation in expectation, then repetitions should be used to get an oracle that maximizes $f(X \uplus Y)$ up to a factor of $\alpha + \varepsilon$ with high probability. Note that when $\varepsilon > 0$ is only polynomially small, this requires only a polynomial number of repetitions since we may assume that $\alpha \leq |\mathcal{N}_2|$ (otherwise, Theorem 2.4 already provides a better approximation).

Algorithm 1: Iterative X Growing

Use an algorithm for submodular minimization to find a set $X_0 \in \arg\min_{X \subseteq \mathcal{N}_1} f(X)$.
for $i = 1$ **to** $n_1 + 1$ **do**
  Use the given oracle to find a set $Y_i \in \mathcal{F}_2$ maximizing $f(X_{i-1} \uplus Y_i)$ up to a factor of α among all sets in $\mathcal{F}_2$.
  Use an algorithm for submodular minimization to find a set $X'_i \in \arg\min_{X \subseteq \mathcal{N}_1} [\sqrt{n_1} \cdot f(X \cup X_{i-1}) + f(X \uplus Y_i)]$.
  if $X'_i \subseteq X_{i-1}$ **then return** the set X_{i-1} and the value $\alpha \cdot f(X_{i-1} \uplus Y_i)$.
  else Let $X_i \leftarrow X_{i-1} \cup X'_i$.

The algorithm that we use to prove Theorem 2.6 is Algorithm 1. Below, we use n_1 as a shorthand for $|\mathcal{N}_1|$, and use I to denote the number of iterations completed by the loop of this algorithm. Since the size of X_i increases following every completed iteration, $I \leq n_1$. Note that iteration $I+1$ started, but stopped before completion since X'_{I+1} was a subset of X_I. Hence, X_I is the output set of Algorithm 1. We begin the analysis of Algorithm 1 with the following lemma.

Lemma 2.7. *For every integer $1 \leq i \leq I$,*

$$f(X_i) = f(X_{i-1} \cup X'_i) \leq f((X^* \cap X'_i) \cup X_{i-1}) + \tau/\sqrt{n_1} \enspace .$$

Proof. By the choice of X'_i, we have

$$\begin{aligned}
&\sqrt{n_1} \cdot f(X'_i \cup X_{i-1}) + f(X'_i \uplus Y_i) \\
\leq{}& \sqrt{n_1} \cdot f((X^* \cap X'_i) \cup X_{i-1}) + f((X^* \cap X'_i) \uplus Y_i) \\
\leq{}& \sqrt{n_1} \cdot f((X^* \cap X'_i) \cup X_{i-1}) + f(X^* \uplus Y_i) + f(X'_i \uplus Y_i) \\
\leq{}& \sqrt{n_1} \cdot f((X^* \cap X'_i) \cup X_{i-1}) + \tau + f(X'_i \uplus Y_i) \enspace ,
\end{aligned}$$

where the second inequality follows from the submodularity and non-negativity of f, and the last inequality follows from the definition of X^*. The lemma now follows by rearranging the last inequality. ☐

Corollary 2.8. *The output set X_I of Algorithm 1 obeys $f(X_I) \leq O(\sqrt{n_1}) \cdot \tau$.*

Proof. If $I = 0$, then $X_I = X_0$, and by definition we have $f(X_I) \leq f(X^*) \leq \tau$. Therefore, we may assume from now on $I \geq 1$. Using Lemma 2.7, we now get

$$\begin{aligned}
f(X_I) - f(X_0) &\leq \sum_{i=1}^{I} [f(X_i) - f(X_{i-1})] \\
&\leq \sum_{i=1}^{I} [f(X^* \cap X'_i \mid X_{i-1}) + \tau/\sqrt{n_1}]
\end{aligned}$$

$$
\begin{aligned}
&\leq \sum_{i=1}^{I} f(X^* \cap X'_i \mid X^* \cap X_{i-1}) + \sqrt{n_1} \cdot \tau \\
&= f(X^* \cap X_I \mid X^* \cap X_0) + \sqrt{n_1} \cdot \tau \\
&\leq f(X^* \cap X_I) - f(X_0) + \sqrt{n_1} \cdot \tau \enspace ,
\end{aligned}
$$

where the penultimate inequality holds by the submodularity of f and the observation that $I \leq n_1$, and the last inequality holds by the definition of X_0. Adding $f(X_0)$ to be both sides of the last inequality yields

$$
\begin{aligned}
&f(X_I) - \sqrt{n_1} \cdot \tau \leq f(X^* \cap X_I) \leq f(X^*) + f(X_I) \\
&- f(X^* \cup X_I) \leq \tau + f(X_I) - f(X^* \cup X_I) \enspace ,
\end{aligned}
$$

where the second inequality follows from the submodularity of f, and last inequality follows from the definition of X^*. To lower bound the term $f(X^* \cup X_I)$, we observe that since $I \geq 1$, the definition of X'_I implies

$$
\begin{aligned}
f(X^* \cup X_I) &= f((X^* \cup X'_I) \cup X_{I-1}) \\
&\geq f(X'_I \cup X_{I-1}) - \frac{f(X^* \mid X'_I \uplus Y_I)}{\sqrt{n_1}} \\
&\geq f(X_I) - \frac{f(X^*)}{\sqrt{n_1}} \geq f(X_I) - \tau/\sqrt{n_1} \enspace ,
\end{aligned}
$$

where the second inequality uses the submodularity and non-negativity of f, and the last inequality holds by the definition of X^*. The corollary now follows by combining this inequality with the previous one. $\square$

We are now ready to prove Theorem 2.6.

Proof of Theorem 2.6. Since Algorithm 1 outputted the set X_I, we must have $X'_{I+1} \subseteq X_I$. Furthermore, by the choices of Y_{I+1} and X'_{I+1},

$$
\begin{aligned}
&\alpha^{-1} \cdot \max_{Y \in \mathcal{F}_2} f(X_I \uplus Y) \\
&\leq f(X_I \uplus Y_{I+1}) = f(X_I) + f(Y_{I+1} \mid X_I) \\
&\leq f(X_I) + f(Y_{I+1} \mid X'_{I+1}) \\
&\leq f(X_I) + [\sqrt{n_1} \cdot f(X^* \cup X_I) + f(X^* \uplus Y_{I+1}) \\
&\quad - \sqrt{n_1} \cdot f(X'_{I+1} \cup X_I) - f(X'_{I+1})] \\
&\leq f(X_I) + \sqrt{n_1} \cdot f(X^* \mid X_I) + f(X^* \uplus Y_{I+1}) \enspace ,
\end{aligned}
$$

where the second inequality follows from the submodularity of f, and the last inequality holds by the non-negativity of f. Observe now that Corollary 2.8 guarantees $f(X_I) \leq O(\sqrt{n}) \cdot \tau$, and the definition of X^* guarantees $f(X^* \cup Y_{I+1}) \leq \tau$. Furthermore, using the submodularity and non-negativity of f, we also get $f(X^* \mid X_I) \leq f(X^*) \leq \tau$. Plugging all these observations into the previous inequality yields

$$
\begin{aligned}
&\alpha^{-1} \cdot \max_{Y \in \mathcal{F}_2} f(X_I \uplus Y) \leq f(X_I \uplus Y_{I+1}) \\
&\leq O(\sqrt{n_1}) \cdot \tau + \sqrt{n_1} \cdot \tau + \tau = O(\sqrt{n_1}) \cdot \tau \enspace .
\end{aligned}
$$

Multiplying the last inequality by α, we get the upper bound on $\max_{Y \in \mathcal{F}_2} f(X_I \uplus Y)$ and $\alpha \cdot f(X_I \uplus Y_{I+1})$ (the value outputted by Algorithm 1) promised in the theorem. The promised lower bound on these expressions also holds since the definition of Y_I implies

$$
\begin{aligned}
&\alpha \cdot f(X_I \uplus Y_{I+1}) \geq \max_{Y \in \mathcal{F}_2} f(X_I \uplus Y) \\
&\geq \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \uplus Y) = \tau \enspace . \qquad \square
\end{aligned}
$$

3 APPLICATIONS

In this section and Appendix D we discuss five machine-learning applications: efficient prompt engineering, ride-share difficulty kernelization, adversarial attack on image summarization, robust ride-share optimization, and prompt engineering for dialog state tracking. Each one of these applications necessitates either max-min or min-max optimization on a jointly submodular function[6] (with a cardinality constraint on the maximization part). To demonstrate the robustness of our suggested methods in this work, we empirically compare them against a few benchmarks.

In the max-min optimization applications, we compare the algorithm from Theorem 2.1 (named below Min-as-Oracle) against 4 benchmarks: (i) "Random" choosing a random set of k elements from $\mathcal{N}_2$ as the set Y; (ii) "Max-Only" using a maximization algorithm to find the a set Y that is (approximately) optimal against $X = \varnothing$; (iii) "Top-k" selecting a set Y consisting of the top k singletons $y \in \mathcal{N}_2$, where each singleton is evaluated based on the corresponding worst case set X; and (iv) "Best-Response" simulating a best response dynamic between the minimization and maximization players, and outputting the set used by the maximization player after a given number of iterations. The Best-Response method is a widely used concept in game theory and optimization, first introduced in the seminal work by Von Neumann and Morgenstern (1947).

In the min-max optimization applications, we study the algorithm from Theorem 2.4 (named below Min-by-Singletons) and a slightly modified version of the algorithm from Theorem 2.6 (named below Iterative-X-Growing). Out of the above 4 benchmarks, the Random and Best-Response benchmarks still make sense in min-max settings with the natural adaptations. It was also natural to try to replace the Max-Only benchmark with a "Min-Only" benchmark, but such a benchmark would always output the empty set in our applications. Thus, we use instead a benchmark called "Max-and-then-Min" that returns a set X that is optimal against

[6]We consider only jointly submodular functions in our experiments since our theoretical results for disjointly submodular functions are, unfortunately, mostly negative.

the set Y returned by Max-Only. See Appendix E for further detail about the various benchmarks, and the implementations of our algorithms.

3.1 Efficient Prompt Engineering

Consider the problem of designing efficient prompts for zero-shot in-context learning. Following Si et al. (2023), we consider an open-domain question answering task: the goal is to answer questions from the SQuAD dataset (Rajpurkar et al., 2016) by prompting a large language model with k relevant passages of text taken from a large corpus of Wikipedia articles. To get for each question an initial set of relevant candidate passages, 21 million Wikipedia passages were embedded using a pretrained Contriever model (Izacard et al., 2022) and indexed using FAISS.[7] Then, for each question, the top 100 passages were kept as candidates.

Large language models such as OpenAI's ChatGPT offer very impressive performance on natural language tasks via a public API. As the cost of making a prediction depends on the length of the input prompt, we propose to reduce the cost by *jointly answering* similar questions with a common prompt, and thus, a single query to the GPT-3.5-turbo language model. To use this approach, we need to select a subset of passages that are effective on the set of *answerable* questions, which we formulate as a combinatorial optimization problem. Specifically, let $\mathcal{N}_1$ be a batch of questions and let $\mathcal{N}_2$ be the union of all candidate passages. (In general, $100 \leq |\mathcal{N}_2| \leq 100 \cdot |\mathcal{N}_1|$ since there may be significant overlap among candidates for questions on the same topic.) Let $0 \leq s_{u,v} \leq 1$ be the cosine similarity between passage embedding u and question embedding v. Then, we define

$$\begin{aligned} f(X \uplus Y) = & \sum_{v \in \mathcal{N}_1 \setminus X} \max_{u \in Y} s_{u,v} \\ & + \beta \cdot \sum_{u \in \mathcal{N}_1 \setminus X} \sum_{v \in Y} s_{u,v} + \lambda \cdot |X| \ . \end{aligned} \tag{2}$$

Here $\lambda \geq 0$ and $\beta \geq 0$ are regularization parameters. The first term represents how well the passages of Y cover the questions in $\mathcal{N}_1 \setminus X$. For small values of β, the second term ensures f increases in $|Y|$, and the last term controls the size of X. The following lemma is proved in Section F.1.

Lemma 3.1. *The objective function* (2) *is a non-negative jointly-submodular function.*

By solving the max-min optimization $\max_{Y \subseteq \mathcal{N}_2, |Y| \leq k} \min_{X \in \mathcal{N}_1} f(X \uplus Y)$, we get the set X of answerable questions, and a small set Y of effective passages. In our experiments we set $\beta = 10^{-3}$, $\lambda = 0.8$, and $k = 10$, and we grouped the SQuAD test set into batches of 25 questions. In addition to the heterogeneity introduced by crude batching, we removed $\delta = 25\%$ of the candidates from $\mathcal{N}_2$, leading to some questions having no relevant passages.

Table 2 shows the performance of the prompts for GPT-3.5-turbo obtained by Min-as-Oracle and various benchmarks. Each method is evaluated in terms of exact match accuracy and F1 score between predicted and ground truth answers. As a baseline, we also consider using the common prompt returned by the retrieval algorithm, but making a *separate prediction* for each question in the cluster. We see our proposed joint prediction with a single prompt increases accuracy while on average requiring only 5.3% of the tokens per question compared to separate prediction. Moreover, Min-as-Oracle has the highest accuracy among all retrieval algorithms used for joint prediction. Figure 1 shows a qualitative example of joint prediction for a batch of questions.

3.2 Ride-Share Difficulty Kernelization

Consider a regulator overseeing the taxi companies licensed to operate within a given city. The regulator wants to make sure that the taxi companies give a fair level of service to all parts of the city, rather than concentrating on the most profitable neighborhoods. However, checking that this is indeed the case is not trivial since often the limited number of taxis available implies that some locations must remain poorly served. Our objective in this section is to give the regulator a small set (kernel) of locations that that capture the difficulty of the problem faced by the taxi company in the sense that the locations in the set cannot be served well (on average) regardless of how the taxi companies choose the waiting locations for their taxis.

Formally, given a set $\mathcal{N}_1$ of (client) pickup locations, and a set $\mathcal{N}_2$ of potential waiting locations for taxies, we define the following score function to capture the convenience of serving all the locations of $\mathcal{N}_1 \setminus X$ by locating taxis at locations Y.[8]

$$\begin{aligned} f(X \uplus Y) = \sum_{v \in \mathcal{N} \setminus X} \max_{u \in Y} s_{u,v} - \frac{1}{|\mathcal{N}_2|} \sum_{u \in Y} \sum_{v \in Y} s_{u,v} \\ + \lambda \cdot |X| \ . \end{aligned} \tag{3}$$

Here, $s_{u,v}$ is a "convenience score" which, given a customer location $u = (x_u, y_u)$ and a waiting driver location $v = (x_v, y_v)$,[9] represents the ease of accessing u

[7] https://github.com/facebookresearch/faiss

[8] It would have been more natural to define X as the set of locations to service. However, this would have resulted in an objective function that is only disjointly submodular.

[9] Each location is specified by a (latitude, longitude) coordinate pair.

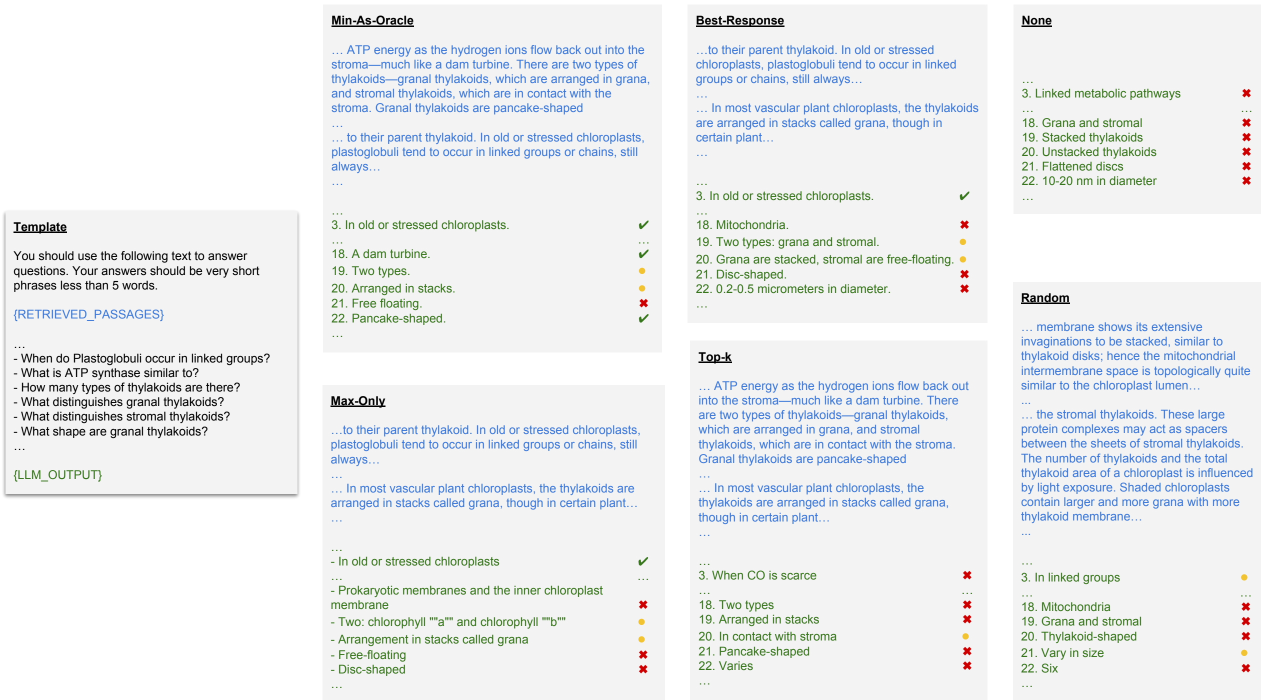


Figure 1: Example of our proposed max-min formulation for jointly answering a batch of questions from the SQuAD dataset using GPT-3.5-turbo. Template prompt (left), followed by excerpts from the retrieved passages (blue) and generated answers (green). Exact match, partial match, and incorrect answer are denoted ✔, ●, and ✖, respectively. Min-as-Oracle retrieved two passages that are relevant to Questions 3, 18, 19, 20, 21, and 22, while the other selection algorithms retrieved only one or neither of them. Consequently, **Min-As-Oracle is best aligned with the ground truth answers, having the highest number of exact matches and the fewest number of hallucinations.**

from v. Following Mitrovic et al. (2018), we set $s_{u,v} \triangleq 2 - \frac{2}{1+e^{-200d(u,v)}}$, where $d(u,v) = |x_u - x_v| + |y_u - y_v|$ is the Manhattan distance between the two points. The value $\lambda \in [0,1]$ is a regularization parameter whose use is discussed below. Some properties of this objective function are given by the next lemma, proved in Appendix F.2.

Lemma 3.2. *The objective function* (3) *is a non-negative jointly-submodular function.*

Recall that we are looking for a kernel set $\mathcal{N}_1 \setminus X$ of pickup locations that cannot be served well (on average) by any choice of k locations for taxis (k is determined by the number of taxis available). To do that, we need to solve the max-min optimization problem given by $\min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2, |Y| \leq k} f(X \cup Y)$. The regularization parameter λ can now be used to control the size the kernel set returned.

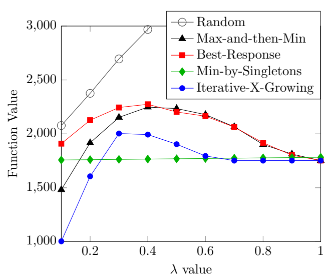
In our experiments for this application, we have used the Uber data set (Uber), which includes real-life Uber pickups in New York City during the month of April in the year 2014. To ensure computational tractability, in each execution of our experiments, we randomly selected from this data set a subset of $|\mathcal{N}_1| = 6{,}000$ pickup locations within the region of Manhattan. Then, we randomly selected a subset of 400 pickup locations from the set $\mathcal{N}_1$ to constitute the set $\mathcal{N}_2$ (we treat locations in $\mathcal{N}_1$ and $\mathcal{N}_2$ as distinct even if they are identical, to guarantee that $\mathcal{N}_1$ and $\mathcal{N}_2$ are disjoint).

In the first experiment, we fixed the maximum number of waiting locations to be 8, and varied λ. Figure 2a depicts the outputs for this experiment for Min-by-Singletons, Iterative-X-Growing (with $\beta = 0.5$) and three benchmarks (averaged over 10 executions of the experiment). One can observe that both Iterative-X-Growing and Min-by-Singletons surpasses the performance of all benchmarks for almost all values of λ. In both this experiment and the next one the standard error of the mean is less than 10 for all data points.

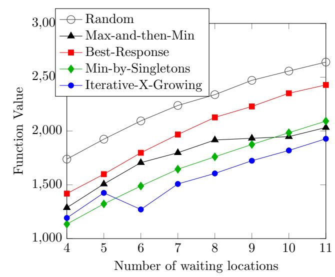
In the second experiment, we fixed λ to 0.2 and varied the number of allowed waiting locations. The results of this experiment are depicted by Figure 2b (averaged over 10 executions of the experiment). Once again, Iterative-X-Growing and Min-by-Singletons demonstrate superior performance compared to the bench-

Table 2: Open-domain question answering on SQuAD using GPT-3.5-turbo. Best values are in **bold**.

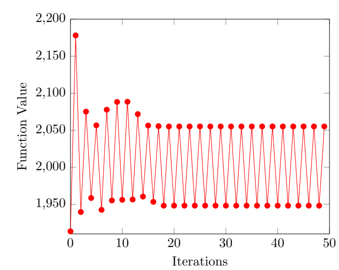
Prompting Method	Retrieval Algorithm	Exact Match % ↑	F1% ↑	Avg Tokens/Question ↓
Joint Prediction	Random	18.6	29.7	73.2
	None	22.5	33.9	20.0
	Top-k	25.0	37.0	**72.8**
	Max-Only	25.9	37.5	73.2
	Best-Response	25.6	37.0	73.2
	Min-as-Oracle	**26.1**	**37.8**	73.2
Separate Prediction	Random	9.7	17.8	1356.3
	None	15.9	29.1	40.1
	Top-k	21.3	31.6	1338.8
	Max-Only	25.3	36.4	1348.7
	Best-Response	25.2	36.2	1348.7
	Min-as-Oracle	25.2	36.3	1348.7



(a) Results for 8 waiting locations



(b) Results for $\lambda = 0.2$



(c) Behavior of Best-Response for $\lambda = 0.2$ and 8 waiting locations

Figure 2: Empirical results for ride-share difficulty kernelization. Figures (a) and (b) compare the performance of our algorithms Min-by-Singletons and Iterative-X-Growing with 3 benchmarks for different value of λ and bounds on the number of weighting locations. Figure (c) depicts the value of the output of the Best-Response method as a function of the number of iterations performed.

marks for almost all values of k. Please refer to Figure 6 in Appendix D.4 for a visual depiction of the results.

As the third experiment for this application, we conducted a more in depth analysis of Best-Response. Figure 2c graphically presents the objective function value obtained by a typical execution of Best-Response after a varying number of iterations (for $\lambda = 0.5$ and an upper bound of 20 on the number of waiting locations). It is apparent that Best-Response does not converge for this execution. Furthermore, both our suggested algorithms demonstrate better performance even with respect to the best performance of Best-Response for any number of iterations between 1 and 50.

4 CONCLUSION

In this paper we have initiated the systematical study of minimax optimization for combinatorial (discrete) settings with large domains. We have fully mapped the theoretical approximability of max-min submodular optimization, and also obtained some understanding of the approximability of min-max submodular optimization. The above theoretical work has been complemented with empirical experiments demonstrating the value of our technique for the machine-learning tasks of efficient prompt engineering, ride-share difficulty kernelization, adversarial attacks on image summarization, and robust ride-share optimization.

We hope future work will lead to a fuller understanding of minimax submodular optimization, and will also consider classes of discrete functions beyond submodularity. A natural class to consider in that regard is the class of weakly-submodular functions (Das and Kempe, 2011), which extends the class of submodular functions and has many machine learning applications (Khanna et al., 2017; Qian and Singer, 2019; Chen et al., 2018; El Halabi et al., 2022). However, minimax optimization of this class seems to be difficult because no algorithm is currently known even for plain minimization of weakly-submodular functions. Another open problem is to prove a performance guarantee for Best-Response.

5 ACKNOWLEDGMENTS

The work of Loay Mualem and Moran Feldman has been partially supported by Israel Science Foundation (ISF) grant number 459/20. Amin Karbasi acknowledges funding in direct support of this work from NSF (IIS-1845032) and the AI Institute for Learning-Enabled Optimization at Scale (TILOS).

References

Arman Adibi, Aryan Mokhtari, and Hamed Hassani. Minimax optimization: The case of convex-submodular. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151 of *Proceedings of Machine Learning Research*, pages 3556–3580. PMLR, 2022. URL `https://proceedings.mlr.press/v151/adibi22a.html`.

Alekh Agarwal and Tong Zhang. Minimax regret optimization for robust machine learning under distribution shift. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory (COLT)*, volume 178 of *Proceedings of Machine Learning Research*, pages 2704–2729. PMLR, 2022. URL `https://proceedings.mlr.press/v178/agarwal22b.html`.

Brian Axelrod, Yang P. Liu, and Aaron Sidford. Near-optimal approximate discrete and continuous submodular function minimization. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 837–853. SIAM, 2020. doi: 10.1137/1.9781611975994.51. URL `https://doi.org/10.1137/1.9781611975994.51`.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014. doi: 10.1137/1.9781611973402.110. URL `https://doi.org/10.1137/1.9781611973402.110`.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL `https://doi.org/10.5281/zenodo.5297715`.

Ilija Bogunovic, Slobodan Mitrovic, Jonathan Scarlett, and Volkan Cevher. Robust submodular maximization: A non-uniform partitioning approach. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 508–516. PMLR, 2017. URL `http://proceedings.mlr.press/v70/bogunovic17a.html`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Trans. Algorithms*, 14(3):32:1–32:20, 2018. doi: 10.1145/3184990. URL `https://doi.org/10.1145/3184990`.

Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Math. Oper. Res.*, 44(3):988–1005, 2019. doi: 10.1287/moor.2018.0955. URL `https://doi.org/10.1287/moor.2018.0955`.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In Chandra Chekuri, editor, *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1433–1452. SIAM, 2014. doi: 10.1137/1.9781611973402.106. URL `https://doi.org/10.1137/1.9781611973402.106`.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015. doi: 10.1137/130929205. URL `https://doi.org/10.1137/130929205`.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *EMNLP*, pages 5016–5026, 2018. doi: 10.18653/v1/D18-1547. URL `https://aclanthology.org/D18-1547`.

Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011. doi: 10.1137/080733991. URL `https://doi.org/10.1137/080733991`.

Lin Chen, Moran Feldman, and Amin Karbasi. Weakly submodular maximization beyond cardinality constraints: Does randomization help greedy? In Jennifer Dy and Andreas Krause, editors, *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 804–813. PMLR, Jul 2018.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL `https://arxiv.org/abs/2107.03374`.

Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In Lise Getoor and Tobias Scheffer, editors, *International Conference on Machine Learning (ICML)*, pages 1057–1064. Omnipress, 2011. URL `https://icml.cc/2011/papers/542_icmlpaper.pdf`.

Jelena Diakonikolas, Constantinos Daskalakis, and Michael I. Jordan. Efficient methods for structured nonconvex-nonconcave min-max optimization. In Arindam Banerjee and Kenji Fukumizu, editors, *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proceedings of Machine Learning Research*, pages 2746–2754. PMLR, 2021. URL `http://proceedings.mlr.press/v130/diakonikolas21a.html`.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey for in-context learning. *arXiv preprint* *arXiv:2301.00234*, 2022. URL https://arxiv.org/abs/2301.00234.

Marwa El Halabi, Suraj Srinivas, and Simon Lacoste-Julien. Data-efficient structured pruning via submodular optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36613–36626. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ed5854c456e136afa3faa5e41b1f3509-Paper-Conference.pdf.

Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. Streaming weak submodularity: Interpreting neural networks on the fly. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4044–4054, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/c182f930a06317057d31c73bb2fedd4f-Abstract.html.

Alina Ene and Huy L. Nguyen. Constrained submodular maximization: Beyond $1/e$. In Irit Dinur, editor, *FOCS*, pages 248–257. IEEE Computer Society, 2016. doi: 10.1109/FOCS.2016.34. URL https://doi.org/10.1109/FOCS.2016.34.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, 2011. doi: 10.1137/090779346. URL https://doi.org/10.1137/090779346.

Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In Rafail Ostrovsky, editor, *FOCS*, pages 570–579. IEEE Computer Society, 2011. doi: 10.1109/FOCS.2011.46. URL https://doi.org/10.1109/FOCS.2011.46.

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In Dana Randall, editor, *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1098–1116. SIAM, 2011. doi: 10.1137/1.9781611973082.83. URL https://doi.org/10.1137/1.9781611973082.83.

Gagan Goel, Chinmay Karande, Pushkar Tripathi, and Lei Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. *SIGecom Exch.*, 9(1):8, 2010. doi: 10.1145/1980534.1980542. URL https://doi.org/10.1145/1980534.1980542.

Michel X. Goemans and V. S. Ramakrishnan. Minimizing submodular functions over families of sets. *Comb.*, 15(4):499–513, 1995. doi: 10.1007/BF01192523. URL https://doi.org/10.1007/BF01192523.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, June 1981. doi: 0.1007/BF02579273.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. In-context learning for few-shot dialogue state tracking. In *Findings of EMNLP*, 2022.

Adam Ibrahim, Waïss Azizian, Gauthier Gidel, and Ioannis Mitliagkas. Linear lower bounds and conditioning of differentiable games. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 4583–4593. PMLR, 2020. URL http://proceedings.mlr.press/v119/ibrahim20a.html.

Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001. doi: 10.1145/502090.502096. URL https://doi.org/10.1145/502090.502096.

Rishabh K. Iyer. A unified framework of robust submodular optimization. *CoRR*, abs/1906.06393, 2019. URL http://arxiv.org/abs/1906.06393.

Rishabh K. Iyer. Robust submodular minimization with applications to cooperative modeling. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 451–458. IOS Press, 2020. doi: 10.3233/FAIA200125. URL https://doi.org/10.3233/FAIA200125.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. In *TMLR*, 2022.

Rajiv Khanna, Ethan Elenberg, Alex Dimakis, Sahand Negahban, and Joydeep Ghosh. Scalable Greedy Feature Selection via Weak Submodularity. In Aarti Singh and Jerry Zhu, editors, *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 1560–1568. PMLR, Apr 2017. URL https://proceedings.mlr.press/v54/khanna17b.html.

Andreas Krause and Volkan Cevher. Submodular dictionary selection for sparse representation. In Johannes Fürnkranz and Thorsten Joachims, editors, *International Conference on Machine Learning (ICML)*, pages 567–574. Omnipress, 2010. URL https://icml.cc/Conferences/2010/papers/366.pdf.

Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(12), 2008.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In Michael Mitzenmacher, editor, *STOC*, pages 323–332. ACM, 2009. doi: 10.1145/1536414.1536459. URL https://doi.org/10.1145/1536414.1536459.

Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1049–1065, 2015. doi: 10.1109/FOCS.2015.68. URL https://doi.org/10.1109/FOCS.2015.68.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2022.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Conference on Artificial Intelligence (AAAI)*, pages 4213–4220. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014213. URL `https://doi.org/10.1609/aaai.v33i01.33014213`.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Hui Lin and Jeff A. Bilmes. A class of submodular functions for document summarization. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 510–520. The Association for Computer Linguistics, 2011. URL `https://aclanthology.org/P11-1052/`.

Tianyi Lin, Chi Jin, and Michael I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 6083–6093. PMLR, 2020. URL `http://proceedings.mlr.press/v119/lin20a.html`.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022.

Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Deletion-robust submodular maximization: Data summarization with “the right to be forgotten”. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 2449–2458. PMLR, 2017. URL `http://proceedings.mlr.press/v70/mirzasoleiman17a.html`.

Marko Mitrovic, Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Data summarization at scale: A two-stage submodular approach. In *International Conference on Machine Learning*, pages 3596–3605. PMLR, 2018.

Slobodan Mitrovic, Ilija Bogunovic, Ashkan Norouzi-Fard, Jakub Tarnawski, and Volkan Cevher. Streaming robust submodular maximization: A partitioned thresholding approach. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 4557–4566, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3baa271bc35fe054c86928f7016e8ae6-Abstract.html`.

Aryan Mokhtari, Asuman E. Ozdaglar, and Sarath Pattathil. Convergence rate of O(1/k) for optimistic gradient and extragradient methods in smooth convex-concave saddle point problems. *SIAM J. Optim.*, 30 (4):3230–3251, 2020. doi: 10.1137/19M127375X. URL `https://doi.org/10.1137/19M127375X`.

Loay Mualem and Moran Feldman. Resolving the approximability of offline and online non-monotone dr-submodular maximization over general convex sets. *arXiv preprint arXiv:2210.05965*, 2022a.

Loay Mualem and Moran Feldman. Using partial monotonicity in submodular maximization. *arXiv preprint arXiv:2202.03051*, 2022b.

George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978. doi: 10.1287/moor.3.3.177. URL `https://doi.org/10.1287/moor.3.3.177`.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Math. Program.*, 14(1):265–294, 1978. doi: 10.1007/BF01588971. URL `https://doi.org/10.1007/BF01588971`.

James B. Orlin, Andreas S. Schulz, and Rajan Udwani. Robust monotone submodular function maximization. *Math. Program.*, 172(1-2):505–537, 2018. doi: 10.1007/s10107-018-1320-2. URL `https://doi.org/10.1007/s10107-018-1320-2`.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In Dana Randall, editor, *SODA*, pages 1098–1116. SIAM, 2011. doi: 10.1137/1.9781611973082.83. URL `https://doi.org/10.1137/1.9781611973082.83`.

Benjamin Qi. On maximizing sums of non-monotone submodular and linear functions. In Sang Won Bae and Heejin Park, editors, *International Symposium on Algorithms and Computation (ISAAC)*, volume 248 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.ISAAC.2022.41. URL `https://doi.org/10.4230/LIPIcs.ISAAC.2022.41`.

Sharon Qian and Yaron Singer. Fast parallel algorithms for statistical subset selection problems. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5073–5082, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/ae587cfeea5ac21a8f1c1ea51027fef0-Abstract.html`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2016.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

Mehraveh Salehi, Amin Karbasi, Dustin Scheinost, and R. Todd Constable. A submodular approach to create individualized parcellations of the human brain. In Maxime Descoteaux, Lena Maier-Hein, Alfred M. Franz, Pierre Jannin, D. Louis Collins, and Simon Duchesne, editors, *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 10433 of *Lecture Notes in Computer Science*, pages 478–485. Springer, 2017. doi: 10.1007/978-3-319-66182-7_55. URL `https://doi.org/10.1007/978-3-319-66182-7_55`.

Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000. doi: 10.1006/jctb.2000.1989. URL `https://doi.org/10.1006/jctb.2000.1989`.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, 2020.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting GPT-3 to be reliable. In *ICLR*, 2023.

Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *International Conference on Machine Learning (ICML)*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 154–162. JMLR.org, 2014. URL `http://proceedings.mlr.press/v32/singla14.html`.

Matthew Staib, Bryan Wilder, and Stefanie Jegelka. Distributionally robust submodular maximization. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 506–516. PMLR, 2019. URL `http://proceedings.mlr.press/v89/staib19a.html`.

Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput.*, 40(6):1715–1737, 2011. doi: 10.1137/100783352. URL `https://doi.org/10.1137/100783352`.

Alfredo Torrico, Mohit Singh, Sebastian Pokutta, Nika Haghtalab, Joseph (Seffi) Naor, and Nima Anari. Structured robust submodular maximization: Offline and online algorithms. *INFORMS J. Comput.*, 33(4):1590–1607, 2021. doi: 10.1287/ijoc.2020.0998. URL `https://doi.org/10.1287/ijoc.2020.0998`.

Sebastian Tschiatschek, Rishabh K. Iyer, Haochen Wei, and Jeff A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1413–1421, 2014. URL `https://proceedings.neurips.cc/paper/2014/hash/a8e864d04c95572d1aece099af852d0a-Abstract.html`.

Uber. Uber pickups in new york city, 2015. `https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city`.

John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior, 2nd rev.* Princeton university press, 1947.

Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013. doi: 10.1137/110832318. URL `https://doi.org/10.1137/110832318`.

Jingkang Wang, Tianyun Zhang, Sijia Liu, Pin-Yu Chen, Jiacen Xu, Makan Fardad, and Bo Li. Adversarial attack generation empowered by min-max optimization. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 16020–16033, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/85e5526a360b0bcf082d8d42e7bf100b-Abstract.html`.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *EMNLP*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *CoRR*, abs/2302.03668, 2023. URL `http://arxiv.org/abs/2302.03668`.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022. URL `https://arxiv.org/abs/2211.01910`.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
    (a) Citations of the creator If your work uses existing assets. [Yes]
    (b) The license information of the assets, if applicable. [Not Applicable]
    (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
    (d) Information about consent from data providers/curators. [Not Applicable]
    (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
    (a) The full text of instructions given to participants and screenshots. [Not Applicable]
    (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
    (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A ADDITIONAL RELATED WORK FOR PROMPT ENGINEERING FOR NATURAL LANGUAGE PROCESSING

In-context learning (Dong et al., 2022) has emerged as a powerful technique to leverage very large language models (Brown et al., 2020; Chen et al., 2021; Ouyang et al., 2022) for Natural Language Processing (NLP) tasks to new tasks without fine-tuning. Recent works, such as Min et al. (2022); Wang et al. (2022); Wei et al. (2022), show the importance of crafting good natural language prompts for these models.

Our prompt engineering experiments build on related works which use a neural retrieval model to prompt large language models for open-domain question answering (Si et al., 2023) and dialog state tracking (Hu et al., 2022). While these previous works only use the Top-k candidates based on embedding similarity, we formulate a novel combinatorial optimization problem for each application.

Some works suggested algorithmic approaches to prompt engineering that learn parameters using gradient-based optimization (Lester et al., 2022; Li and Liang, 2021; Shin et al., 2020; Wen et al., 2023). More recently, Zhou et al. (2022) designed prompts by ranking generations from a secondary language model combined with iterative Monte Carlo search. All of these methods are complex, computationally expensive, and challenging to interpret.

B PROOFS OF SECTION 2.1

B.1 Proof of Theorem 2.1

In this section we prove Theorem 2.1, which we repeat here for convenience.

Theorem 2.1. *Assume that there exists an α-approximation algorithm ALG for the problem of maximizing a non-negative submodular function g subject to $\mathcal{F}_2$. Then, for every polynomially small $\varepsilon \in (0, \alpha]$, there exists a polynomial time algorithm that (i) outputs a set $\hat{Y} \in \mathcal{F}_2$ and the value $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y})$; and (ii) guarantees that, with probability at least $2/3$, $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y})$ falls within the range $[\tau/(\alpha + \varepsilon), \tau]$, where $\tau = \max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cupdot Y)$. Furthermore, if f is $\mathcal{N}_2$-monotone, then it suffices for ALG to obtain α-approximation when g is guaranteed to be monotone (in addition to being non-negative and submodular).*

The majority of the section is devoted to proving the slightly different version of the last theorem given by Proposition B.1. If *ALG* is a deterministic algorithm, then the algorithm whose existence is guaranteed by Proposition B.1 is also deterministic, and immediately implies Theorem 2.1. However, if *ALG* is a randomized algorithm, then it might be necessary to use repetitions to get the result stated in Theorem 2.1. Specifically, by a Markov-like argument, the probability that $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y}) \geq \tau/(\alpha + \varepsilon)$ must be at least ε/α^2, and therefore, by executing the algorithm from Proposition B.1 $O(\alpha^2/\varepsilon)$ times, the probability of getting a set $\hat{Y}$ for which $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y}) \geq \tau/(\alpha + \varepsilon)$ can be made to be at least $2/3$.

Proposition B.1. *Assume that there exists an α-approximation algorithm ALG for the problem of maximizing a non-negative submodular function g subject to $\mathcal{F}_2$, then there exists a polynomial time algorithm that outputs a set $\hat{Y} \in \mathcal{F}_2$ and the value $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y})$, and guarantees that (i) $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y}) \leq \tau$, and (ii) the expectation of $\min_{X \subseteq \mathcal{N}_1} f(X \cupdot \hat{Y})$ is at least τ/α. Furthermore, if f is $\mathcal{N}_2$-monotone, then it suffices for ALG to obtain α-approximation when g is guaranteed to be monotone (in addition to being non-negative and submodular).*

To prove Proposition B.1, let us define, for every set $Y \subseteq \mathcal{N}_2$, $g(Y) = \min_{X \subseteq \mathcal{N}_1} f(X \cupdot Y)$. It is well-known that g is a submodular function, and we prove it in the next lemma for completeness (along with additional properties of g).

Lemma B.2. *The function $g \colon 2^{\mathcal{N}_2} \to \mathbb{R}_{\geq 0}$ is a non-negative submodular function, and there exists a polynomial time implementation of the value oracle of g. Furthermore, if f is $\mathcal{N}_2$-monotone, then g is monotone (in addition to being non-negative and submodular).*

Proof. We begin the proof by considering Algorithm 2. One can observe that this algorithm describes a way to implement a value oracle for g because, by the definitions of X' and h_Y,

$$f(X' \cup Y) = h_Y(X') = \min_{X \subseteq \mathcal{N}_1} h_Y(X) = \min_{X \subseteq \mathcal{N}_1} f(X \cup Y) = g(Y) \enspace .$$

Algorithm 2: Value oracle implementation (Y)

Define $h_Y(X) \triangleq f(X \mathbin{\dot\cup} Y)$ for every set $X \subseteq \mathcal{N}_1$.
Find a set $X' \subseteq \mathcal{N}_1$ minimizing $h_Y(X')$.
return $f(X' \mathbin{\dot\cup} Y)$.

Furthermore, Algorithm 2 can be implemented to run in polynomial time using any polynomial time algorithm for unconstrained submodular minimization because h_Y is a submodular function.

The non-negativity of g follows from the definition of g and the non-negativity of f. Proving that g is also submodular is more involved. Let Y_1 and Y_2 be two arbitrary subsets of $\mathcal{N}_2$, and let us choose a set $X_i \in \arg\min_{X \subseteq \mathcal{N}_1} f(X \mathbin{\dot\cup} Y_i)$ for every $i \in \{1, 2\}$. Then,

$$\begin{aligned}
g(Y_1) + g(Y_2) &= f(X_1 \mathbin{\dot\cup} Y_1) + f(X_2 \mathbin{\dot\cup} Y_2)\\
&\geq f((X_1 \cap X_2) \mathbin{\dot\cup} (Y_1 \cap Y_2)) + f((X_1 \cup X_2) \mathbin{\dot\cup} (Y_1 \cup Y_2))\\
&\geq \min_{X \subseteq \mathcal{N}_1} f(X \mathbin{\dot\cup} (Y_1 \cap Y_2)) + \min_{X \subseteq \mathcal{N}_1} f(X \mathbin{\dot\cup} (Y_1 \cup Y_2)) = g(Y_1 \cap Y_2) + g(Y_1 \cup Y_2) \enspace,
\end{aligned}$$

where the first inequality holds by the submodularity of f since $X_1 \cup X_2 \subseteq \mathcal{N}_1$ is disjoint from $Y_1 \cup Y_2 \subseteq \mathcal{N}_2$. This completes the proof that g is submodular.

It remains to prove that g is monotone whenever f is $\mathcal{N}_2$-monotone. Therefore, in the rest of the proof we assume that f indeed has this property. Then, if the sets Y_1 and Y_2 obey the inclusion $Y_1 \subseteq Y_2$, then they also obey

$$g(Y_2) = f(X_2 \cup Y_2) \geq f(X_2 \cup Y_1) \geq f(X_1 \cup Y_1) = g(Y_1) \enspace,$$

where the first inequality follows from the $\mathcal{N}_2$-monotonicity of f, and the second inequality follows from the definition of X_1. □

We are now ready to prove Proposition B.1.

Proof of Proposition B.1. Note that Lemma B.2 implies that g has all the properties necessary for *ALG* to guarantee α-approximation for the problem of $\min_{Y \in \mathcal{F}_2} g(Y)$. Therefore, we can use *ALG* to implement in polynomial time the procedure described by Algorithm 3 (since *ALG* runs in polynomial time given a polynomial time value oracle implementation for the objective function). Since the definition of g implies $\max_{Y \in \mathcal{F}_2} g(Y) =$

Algorithm 3: Approximate using *ALG*

Use *ALG* to get a set $Y' \in \mathcal{F}_2$ such that $\alpha^{-1} \cdot \max_{Y \in \mathcal{F}_2} g(Y) \leq \mathbb{E}[g(Y')] \leq \max_{Y \in \mathcal{F}_2} g(Y)$.
return the set Y' and the value $g(Y')$.

$\max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_2} f(X \mathbin{\dot\cup} Y) = \tau$, the value $g(Y') = \min_{X \subseteq \mathcal{N}_1} f(X \mathbin{\dot\cup} Y') \leq \max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_2} f(X \mathbin{\dot\cup} Y)$ produced by Algorithm 3 is at most τ and in expectation at least τ/α. Therefore, Algorithm 3 has all the properties guaranteed by Proposition B.1. □

B.2 Proofs of Theorems 2.2 and 2.3

In this section we prove the inapproximability results stated in Theorems 2.2 and 2.3. The proofs of both theorems are based on the reduction described by the following proposition. Below, we use $\mathbb{N}_0$ and $\mathbb{N}$ to denote the set of natural numbers with and without 0, respectively. Additionally, recall that for a non-negative integer i, $[i] = \{1, 2, \dots, i\}$. In particular, this implies that $[0] = \varnothing$, which is a property we employ later in the section.

Proposition B.3. *Fix any family F_2 of pairs of ground set $\mathcal{N}_2$ and constraint $\mathcal{F}_2 \subseteq 2^{\mathcal{N}_2}$. Additionally, let $\alpha\colon \mathbb{N}_0 \times F_2 \to [1, \infty)$ be an arbitrary function (intuitively, for every pair $(\mathcal{N}_2, \mathcal{F}_2) \in F_2$, $\alpha(m, \mathcal{N}_2, \mathcal{F}_2)$ is an approximation ratio that we assign to this pair when the ground set $\mathcal{N}_1$ has a size of m). Assume that there exists* *a (possibly randomized) polynomial time algorithm ALG which, given a ground set $\mathcal{N}_1$, a pair $(\mathcal{N}_2, \mathcal{F}_2) \in F_2$, and a non-negative disjointly submodular function $f\colon 2^{\mathcal{N}_1 \cup \mathcal{N}_2} \to \mathbb{R}_{\geq 0}$, outputs a value v such that, with probability at least 2/3,*

$$\tfrac{1}{\alpha(|\mathcal{N}_1|, (\mathcal{N}_2, \mathcal{F}_2))} \cdot \max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y) \leq v \leq \max_{Y \in \mathcal{F}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y) \enspace .$$

Then, there also exists a polynomial time algorithm that given a pair $(\mathcal{N}_2, \mathcal{F}_2) \in F_2$ and non-negative submodular functions $g_1, g_2, \dots, g_m\colon 2^{\mathcal{N}_2} \to \mathbb{R}_{\geq 0}$ outputs a value v such that, with probability at least 2/3,

$$\tfrac{1}{\alpha(m-1, (\mathcal{N}_2, \mathcal{F}_2))} \cdot \max_{Y \in \mathcal{F}_2} \min_{1 \leq i \leq m} g_i(Y) \leq v \leq \max_{Y \in \mathcal{F}_2} \min_{1 \leq i \leq m} g_i(Y) \enspace .$$

Furthermore, if the functions $g_1, g_2, \dots, g_m\colon 2^{\mathcal{N}_2} \to \mathbb{R}_{\geq 0}$ are all guaranteed to be monotone (in addition to being non-negative and submodular), then it suffices for ALG to have the above guarantee only when f is $\mathcal{N}_2$-monotone (in addition to being non-negative and disjointly submodular).

Before proving Proposition B.3, let us show that it indeed implies Theorems 2.2 and 2.3.

Theorem 2.2. *When f is only guaranteed to be non-negative and disjointly submodular, no polynomial time algorithm for calculating $\max_{Y \subseteq \mathcal{N}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$ has a finite approximation ratio unless $BPP = NP$.*

Proof. Fix the family $F_2 = \{([k], 2^{[k]}) \mid k \in \mathbb{N}\}$. Assume that there exists a polynomial time algorithm for calculating $\max_{Y \subseteq \mathcal{N}_2} \min_{X \subseteq \mathcal{N}_1} f(X \cup Y)$ that has a polynomial approximation ratio. By plugging this algorithm and the family F_2 into Proposition B.3, we get that there exists a polynomial time algorithm ALG and a polynomial function $\alpha\colon \mathbb{N} \times \mathbb{N} \to [1, \infty)$ such that, given integer $k \in \mathbb{N}$ and m non-negative monotone submodular functions $g_1, g_2, \dots, g_m$, the algorithm ALG produces a value v such that, with probability at least 2/3,

$$\tfrac{1}{\alpha(m,k)} \cdot \max_{Y \subseteq [k]} \min_{1 \leq i \leq m} g_i(Y) \leq v \leq \max_{Y \subseteq [k]} \min_{1 \leq i \leq m} g_i(Y) \enspace .$$

In particular, ALG answers correctly with probability at least 2/3 whether the expression $\max_{Y \subseteq [k]} \min_{1 \leq i \leq m} g_i(Y)$ is equal to zero. Therefore, to prove the theorem it suffices to show that that exists some NP-hard problem such that every instance I of this problem can be encoded in polynomial time as an expression of the form $\max_{Y \subseteq [k]} \min_{1 \leq i \leq m} g_i(Y)$ that takes the value 0 if and only if the correct answer for the instance I is “No”.

In the rest of this proof, we show that this is indeed the case for the NP-hard problem `SAT`. Every instance of `SAT` consists a CNF formula ϕ over n variables $x_1, x_2, \dots, x_n$ that has ℓ clauses. To encode this instance, we need to construct $n + \ell$ functions over the ground set $[2n]$. Intuitively, for every integer $1 \leq i \leq n$ the elements $2i - 1$ and $2i$ of the ground set correspond to the variable x_i of ϕ. The element $2i - 1$ corresponds to an assignment of 1 to this variable, and the element $2i$ corresponds to an assignment of 0. For every integer $1 \leq i \leq n$, the objective of the function g_i is to make sure that exactly one value is assigned to x_i. Formally, this is done by defining $g_i(Y) \triangleq |\{2i - 1, 2i\} \cap Y| \bmod 2$ for every $Y \subseteq [2n]$. One can note that $g_i(Y)$ takes the value 1 only when exactly one of the elements $2i - 1$ or $2i$ belongs to Y. Furthermore, one can verify that g_i is non-negative and submodular.

Next, we need to define the functions $g_{n+1}, g_{n+2}, \dots, g_{n+\ell}$. To define these functions, let us denote by $c_1, c_2, \dots, c_\ell$ the clauses of ϕ. Additionally we denote by $c_j(x_i = v)$ an indicator that gets the value 1 if assigning the value v to x_i guarantees that the clause c_j is satisfied. In other words, $c_j(x_i = v)$ equals 1 only if $v = 1$ and c_j includes the positive literal x_i, or $v = 0$ and c_j includes the negative literal $\bar{x}_j$. For every integer $1 \leq j \leq \ell$, the function $g_{n+j}(Y)$ corresponds to the clause w_j and takes the value 1 only when this clause is satisfied by some element of Y. Formally,

$$g_{n+j}(Y) = \max_{i \in Y} c_j(x_{\lceil i/2 \rceil} = (i \bmod 2))$$

(notice that $x_{\lceil i/2 \rceil}$ is the index of the variable corresponding to element i, and $i \bmod 2$ is the value assigned to this variable by the element i). One can verify that $g_{n+j}(Y)$ is a non-negative submodular (and even monotone) function.

Let us now explain why $\max_{Y \subseteq [2n]} \min_{1 \leq i \leq m} g_i(Y)$ takes the value 0 if and only if ϕ is not satisfiable. First, if there exists a satisfying assignment a for ϕ, then one can construct a set $Y \subseteq [2n]$ that encodes a. Specifically, for every integer $1 \leq i \leq n$, Y should include $2i - 1$ (and not $2i$) if a assigns the value 1 to x_i, and otherwise Y should include $2i$ (and not $2i - 1$). Such a choice of Y will make all the above functions $g_1, g_2, \dots, g_{n+\ell}$ take the value 1, and therefore, $\max_{Y \subseteq [2n]} \min_{1 \leq i \leq m} g_i(Y) = 1$ in this case. Consider now the case in which ϕ does not have a satisfying assignment. Then, for every set $Y \subseteq [2n]$ we must have one of the following. The first option is that Y includes either both $2i-1$ and $2i$, or neither of these elements, for some integer i, which makes g_i evaluate to 0 on Y. The other option is that Y corresponds to some legal assignment a of values to $x_1, x_2, \ldots, x_n$ that violates some clause c_j, and thus, g_{n+j} evaluates to 0 on Y. In both cases $\min_{1 \leq i \leq m} g_i(Y) = 0$. □

Theorem 2.3. *When f is only guaranteed to be non-negative, $\mathcal{N}_2$-monotone and disjointly submodular, no polynomial time algorithm for calculating $\max_{Y \subseteq \mathcal{N}_2, |Y| \leq \rho} \min_{X \subseteq \mathcal{N}_1} f(X \uplus Y)$, where ρ is a parameter of the problem, has a finite approximation ratio unless $BPP = NP$.*

Proof. The proof of this theorem is very similar to the proof of Theorem 2.2, and therefore, we only describe here the differences between the two proofs. First, the family F_2 should be chosen this time as $\mathcal{F}_2 = \{([2k], \{Y \subseteq [2k] \mid |Y| \leq k\} \mid k \in \mathbb{N}\}$. This modification implies that we now need to encode ϕ as an instance of

$$\max_{\substack{Y \subseteq [2n] \\ |Y| \leq n}} \min_{1 \leq i \leq m} g_i(Y) \enspace ,$$

where the functions $g_i(Y)$ are all non-negative monotone submodular functions over the ground set $[2n]$. We do this using $n + \ell$ functions like in the proof of Theorem 2.2. Moreover, the functions $g_{n+1}, g_{n+2}, g_{n+\ell}$ are defined exactly like in the proof of Theorem 2.2.

For every integer $1 \leq i \leq n$, the function g_i still corresponds to the variable x_i, but now the role of g_i is only to guarantee that x_i gets at least a single value. This is done by setting $g_i(Y) = \min\{|Y \cap \{2i-1, 2i\}|, 1\}$, which means that g_i takes the value 1 only when at least one of the elements $2i-1$ or $2i$ belongs to Y. Note that g_i is indeed non-negative, monotone and submodular, as necessary. The main observation that we need to make is that if Y is a set of size at most n for which all the functions $g_1, g_2, \ldots, g_n$ return 1, then Y must include at least one element of the pair $\{2i-1, 2i\}$ for every integer $1 \leq i \leq n$. Since these are n disjoint pairs, and Y contains at most n elements, we get that Y contains *exactly* one element of each one of the pairs $\{2i-1, 2i\}$. In other words, $\min_{1 \leq i \leq n} g_i(Y) = 1$ if and only if Y corresponds to assigning exactly one value to every variable x_i, which is exactly the property that the functions $g_1, g_2, \ldots, g_n$ need to have to allow the rest of the proof of Theorem 2.2 to go through. □

Remark. The above proof of Theorem 2.3 plugs into Proposition B.3 the observation that an expression of the form $\max_{Y \subseteq \mathcal{N}_2, |Y| \leq k} \min_{1 \leq i \leq m} g_i(Y)$ can capture an NP-hard problem. The last observation was already shown by Theorem 3 of Krause et al. (2008) (for the Hitting-Set problem). Thus, Theorem 2.3 can also be obtained as a corollary of Proposition B.3 and Theorem 3 of Krause et al. (2008). However, for completeness and consistency, we chose to provide a different proof of Theorem 2.3 that closely follows the proof of Theorem 2.2.

We now get to the proof of Proposition B.3. One can observe that to prove this proposition it suffices to show the following lemma (the algorithm whose existence is guaranteed by Proposition B.3 can be obtained by simply applying ALG to the ground set $\mathcal{N}_1$ and function f defined by Lemma B.4).

Lemma B.4. *Given non-negative submodular functions $g_1, g_2, \ldots, g_m \colon 2^{\mathcal{N}_2} \to \mathbb{R}_{\geq 0}$, there exists a ground set $\mathcal{N}_1$ and a non-negative disjointly submodular function $f \colon 2^{\mathcal{N}_1 \uplus \mathcal{N}_2} \to \mathbb{R}_{\geq 0}$ such that*

- *the size of the ground set $\mathcal{N}_1$ is $m - 1$.*
- *given sets $X \subseteq \mathcal{N}_1$ and $Y \subseteq \mathcal{N}_2$, it is possible to evaluate $f(X \uplus Y)$ in polynomial time.*
- *for every set $Y \subseteq \mathcal{N}_2$, $\min_{X \subseteq \mathcal{N}_1} f(X \uplus Y) = \min_{1 \leq i \leq m} g_i(Y)$.*
- *when the functions $g_1, g_2, \ldots, g_m$ are all monotone (in addition to being non-negative and submodular), then the function f is guaranteed to be $\mathcal{N}_2$-monotone (in addition to being non-negative and disjointly submodular).*

The rest of this section is devoted to proving Lemma B.4. Let us start by describing how the ground set $\mathcal{N}_1$ and the function f are constructed. We assume without loss of generality that $\mathcal{N}_2 \cap [m-1] = \varnothing$, which allows us to choose $\mathcal{N}_1 = [m-1]$. Given a set $X \subseteq [m-1]$, let us define $c(X) \triangleq \max\{i \in \mathbb{N}_0 \mid [i] \subseteq X\}$ (in other words, $c(X)$ is the largest integer such that all the numbers 1 to i appear in X). Additionally, we choose M to be a number obeying $g_i(Y) \leq M/2$ for every $i \in [m]$ and $Y \subseteq \mathcal{N}_2$ (such a number can be obtained in polynomial time by running the 2-approximation algorithm of Buchbinder and Feldman (2018) for unconstrained submodular maximization on the functions $g_1, g_2, \ldots, g_m$, and then setting M to be four times the largest number returned). Using this notation, we can now define, for every two sets $X \subseteq \mathcal{N}_1$ and $Y \subseteq \mathcal{N}_2$,

$$f(X \cup Y) \triangleq g_{c(X)+1}(Y) + (|X| - c(X)) \cdot M \enspace .$$

The following observation states some properties of f that immediately follow from the definition of f and the fact that $c(X)$ is at most $|X|$ by definition.

Observation B.5. *The function f is non-negative and can be evaluated in polynomial time. Furthermore, f is $\mathcal{N}_2$-monotone when the functions $g_1, g_2, \ldots, g_m$ are monotone because $g_{c(X)+1}(Y) + (|X| - c(X)) \cdot M$ is a monotone function of Y for any fixed set $X \subseteq \mathcal{N}_1$.*

The following two lemmata prove additional properties of f.

Lemma B.6. *The function f is disjointly submodular, i.e., it is submodular when restricted to either $\mathcal{N}_1$ or $\mathcal{N}_2$.*

Proof. For every fixed set $X \subseteq \mathcal{N}_1$, there exists a value $i \in [m]$ and another value c, both depending only on X, such that $f(X \cup Y) = g_i(Y) + c$. Since adding a constant to a submodular function does not affect its submodularity, this implies that f is submodular when restricted to $\mathcal{N}_2$. In the rest of the proof we concentrate on showing that f is also submodular when restricted to $\mathcal{N}_1$.

Consider now an arbitrary element $i \in \mathcal{N}_1$. For every two sets $X \subseteq \mathcal{N}_1 - i$ and $Y \subseteq \mathcal{N}_2$,

$$f(i \mid X \cup Y) = g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) + (1 + c(X) - c(X+i)) \cdot M \enspace .$$

To show that f is submodular when restricted to $\mathcal{N}_1$, we need to show that the last expression is a down-monotone function X, i.e., that its value does not increase when elements are added to X. To do that, it suffices to show that the addition to X of any single element $j \in \mathcal{N}_1 \setminus (X + i)$ does not increase the value of this expression; which we show below by considering a few cases.

The first case we need to consider is the case of $[i-1] \not\subseteq X + j$. Clearly, in this case $c(X) = c(X+i)$ and $c(X+j+i) = c(X+j)$, and therefore,

$$\begin{aligned} f(i \mid (X+j) \cup Y) &= g_{c(X+j+i)+1}(Y) - g_{c(X+j)+1}(Y) + (1 + c(X+j) - c(X+j+i)) \cdot M \\ &= M = g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) + (1 + c(X) - c(X+i)) \cdot M = f(i \mid X \cup Y) \enspace . \end{aligned}$$

The second case we consider the case in which $[i-1] \subseteq X + j$, but $[i-1] \not\subseteq X$. In this case

$$\begin{aligned} f(i \mid (X+j) \cup Y) &= g_{c(X+j+i)+1}(Y) - g_{c(X+j)+1}(Y) + (1 + c(X+j) - c(X+j+i)) \cdot M \\ &\leq g_{c(X+j+i)+1}(Y) - g_{c(X+j)+1}(Y) \leq g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) + M \\ &= g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) + (1 + c(X) - c(X+i)) \cdot M = f(i \mid X \cup Y) \enspace , \end{aligned}$$

where the first inequality holds since the definition of the case implies $c(X+j+i) \geq i = 1 + c(X+j)$, the second inequality follows from the definition of M, and the penultimate equality holds since the definition of the case implies $c(X) = c(X+i)$.

The third case we need to consider is when $[i-1] \subseteq X$ and $c(X+i) = c(X+i+j)$. Since we also have in this case $c(X) = i - 1 = c(X+j)$, we get

$$\begin{aligned} f(i \mid (X+j) \cup Y) &= g_{c(X+j+i)+1}(Y) - g_{c(X+j)+1}(Y) + (1 + c(X+j) - c(X+j+i)) \cdot M \\ &= g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) + (1 + c(X) - c(X+i)) \cdot M = f(i \mid X \cup Y) \end{aligned}$$

The last case we need to consider is when $[i-1] \subseteq X$ and $c(X+i) < c(X+j+i)$. In this case

$$\begin{aligned} f(i \mid (X+j) \cup Y) &= g_{c(X+j+i)+1}(Y) - g_{c(X+j)+1}(Y) + (1 + c(X+j) - c(X+j+i)) \cdot M \\ &\leq g_{c(X+j+i)+1}(Y) - g_{c(X+j)+1}(Y) - M \leq g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) \\ &= g_{c(X+i)+1}(Y) - g_{c(X)+1}(Y) + (1 + c(X) - c(X+i)) \cdot M = f(i \mid X \cup Y) \enspace , \end{aligned}$$

where the first inequality holds since the definition of the case implies $c(X+j) = i-1 = c(X+i)-1 < c(X+j+i)-1$, the second inequality follows from the definition of M, and the penultimate equality holds since the definition of the case implies $c(X) = i - 1 = c(X+i) - 1$. $\square$

Lemma B.7. *For every set $Y \subseteq \mathcal{N}_2$, $\min_{X \subseteq \mathcal{N}_1} f(X \cup Y) = \min_{1 \leq i \leq m} g_i(Y)$.*

Proof. Observe that for every integer $1 \leq i \leq m$, we have $f([i-1] \cup Y) = g_i(Y)$ because $|[i-1]| = c([i-1]) = i-1$. Therefore,

$$\min_{1 \leq i \leq m} f([i-1] \cup Y) = \min_{1 \leq i \leq m} g_i(Y) \ . \tag{4}$$

Consider now an arbitrary subset X of $\mathcal{N}_1$ that is not equal to $[i-1]$ for any integer $1 \leq i \leq m$. For such a subset we must have $c(X) \leq |X| - 1$, and therefore,

$$f(X \cup Y) = g_{c(X)+1}(Y) + (|X| - c(X)) \cdot M \geq g_{c(X)+1}(Y) + M \geq M \geq \min_{1 \leq i \leq m} g_i(Y) \ ,$$

where the second inequality follows from the non-negativity of $g_{c(X)+1}$, and the last inequality holds by the definition of M. Combining this inequality with Equation (4) completes the proof of the lemma. $\square$

Lemma B.4 now follows by combining Observation B.5, Lemma B.6 and Lemma B.7.

C OMITTED PROOFS OF SECTION 2.2

C.1 Proof of Theorem 2.4

In this section we prove Theorem 2.4, which we repeat here for convenience.

Theorem 2.4. *Assuming $\{u\} \in \mathcal{F}_2$ for every $u \in \mathcal{N}_2$, there is a polynomial time algorithm that, given a non-negative disjointly submodular function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$, returns a set $\hat{X} \subseteq \mathcal{N}_1$ and a value v such that both $\max_{Y \in \mathcal{F}_2} f(\hat{X} \cup Y)$ and v fall within the range $[\tau, (|\mathcal{N}_2| + 1) \cdot \tau]$, where $\tau \triangleq \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \cup Y)$.*

The algorithm that we use to prove Theorem 2.4 is given as Algorithm 4. We note that the function g defined by this algorithm is the average of $|\mathcal{N}_2| + 1$ submodular functions (since f is submodular once the subset of $\mathcal{N}_2$ in the argument set is fixed), and therefore, g is also submodular. As written, Algorithm 4 is good only for the case in which $\varnothing \in \mathcal{F}_2$, and for simplicity, we assume throughout the section that this is indeed the case. If $\varnothing \notin \mathcal{F}_2$, then the term $f(X)$ should be dropped from the definition of g in Algorithm 4, which allows the proof to go through.

Algorithm 4: Estimating the min-max using singletons

1 Define a function $g\colon 2^{\mathcal{N}_1} \to \mathbb{R}_{\geq 0}$ as follows. For every set $X \subseteq \mathcal{N}_1$, $g(X) \triangleq f(X) + \sum_{u \in \mathcal{N}_2} f(X \cup \{u\})$.
2 Use an unconstrained submodular minimization algorithm to find $X' \subseteq \mathcal{N}_1$ minimizing $g(X')$.
3 **return** the set X' and the value $g(X')$.

The analysis of Algorithm 4 is based on the observation that $g(X)$ provides an approximation for $\max_{Y \in \mathcal{F}_2} f(X \cup Y)$.

Lemma C.1. *For every set $X \subseteq \mathcal{N}_1$, $\max_{Y \in \mathcal{F}_2} f(X \cup Y) \leq g(X) \leq (|\mathcal{N}_2| + 1) \cdot \max_{Y \in \mathcal{F}_2} f(X \cup Y)$.*

Proof. Let Y' be the set in $\mathcal{F}_2$ maximizing $f(X \cup Y')$, then the disjoint submodularity of f guarantees that

$$\begin{aligned}\max_{Y \in \mathcal{F}_2} f(X \cup Y) = f(X \cup Y') &\leq f(X) + \sum_{u \in Y'} f(u \mid X) \\ &\leq f(X) + \sum_{u \in Y'} f(X \cup \{u\}) \leq f(X) + \sum_{u \in \mathcal{N}_2} f(X \cup \{u\}) = g(X) \ ,\end{aligned}$$

where the second and last inequalities hold by the non-negativity of f. This completes the proof of the first inequality of the lemma. To see why the other inequality holds as well, we note that $g(X)$ is the sum of $|\mathcal{N}_2| + 1$ terms, each of which is individually upper bounded by $\max_{Y \in \mathcal{F}_2} f(X \cup Y)$. $\square$

Using the last lemma, we can now prove Theorem 2.4.

Proof of Theorem 2.4. Let X^* be the set minimizing $\max_{Y \in \mathcal{F}_2} f(X^* \cup Y)$. Then, by Lemma C.1 and the choice of X' by Algorithm 4,

$$\begin{aligned}\tau = {} & \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \cup Y) \leq \max_{Y \in \mathcal{F}_2} f(X' \cup Y) \leq g(X') \leq g(X^*) \\ & \leq (|\mathcal{N}_2| + 1) \cdot \max_{Y \in \mathcal{F}_2} f(X^* \cup Y) = (|\mathcal{N}_2| + 1) \cdot \min_{X \subseteq \mathcal{N}_1} \max_{Y \in \mathcal{F}_2} f(X \cup Y) = (|\mathcal{N}_2| + 1)\tau \ .\end{aligned}$$

□

C.2 Proof of Theorem 2.5

In this section we would like to prove Theorem 2.5. However, the majority of the section is devoted to proving the following slightly different theorem, which implies Theorem 2.5.

Theorem C.2. *For every constant $\varepsilon \in (0, 1/2)$, there exists a polynomial time algorithm that given a non-negative disjointly submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ returns a set $\hat{X}$ and a value v such that*

- *v's expectation falls within the range $[\tau, (4 + \varepsilon/2)\tau]$, where $\tau \triangleq \min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2} f(X \cup Y)$, and*
- *with probability at least $1 - \varepsilon/[8(|\mathcal{N}_2| + 1)]$, both v and $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ fall within the range $[\tau, (4 + \varepsilon/2)\tau]$.*

Before getting to the proof of Theorem C.2, let us show that it indeed implies Theorem 2.5, which we repeat here for convenience.

Theorem 2.5. *For every constant $\varepsilon \in (0, 1)$, there exists a polynomial time algorithm that given a non-negative disjointly submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ returns a set $\hat{X} \subseteq \mathcal{N}_1$ and a value v such that the expectations of both $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ and v fall within the range $[\tau, (4 + \varepsilon)\tau]$, where $\tau \triangleq \min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2} f(X \cup Y)$. Furthermore, the probability that both $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ and v fall within this range is at least $1 - O(|\mathcal{N}_2|^{-1})$.*

Proof. Since the guarantee of Theorems 2.5 becomes stronger as ε becomes smaller, it suffices to prove the theorem for $\varepsilon \in (0, 1/2)$. Furthermore, the only way in which the algorithm guaranteed by Theorem C.2 might not obey the properties described in Theorem 2.5 is if the expectation of $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ for its output set $\hat{X}$ does not fall within the range $[\tau, (4 + \varepsilon)\tau]$. Thus, to prove Theorem 2.5 it is only necessary to show how to modify the output set $\hat{X}$ of Theorem C.2 in a way that does not violate the other properties guaranteed by this theorem, but makes the expectation of $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ fall into the right range. We do that using Algorithm 5. This algorithm uses a deterministic polynomial time algorithm that obtains 2-approximation for unconstrained submodular maximization. Such an algorithm was given by Buchbinder and Feldman (2018).

Algorithm 5: Best of two (ε)

---

1 Execute the algorithm guaranteed by Theorem C.2. Let X' denote its output set.
2 Use an algorithm for unconstrained submodular maximization to find a set $Y' \subseteq \mathcal{N}_2$ such that $\max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y) \leq 2f(X' \cup Y') \leq 2 \cdot \max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y)$.
3 Execute the algorithm guaranteed by Theorem 2.4. Let X'' denote its output set.
4 Use an algorithm for unconstrained submodular maximization to find a set $Y'' \subseteq \mathcal{N}_2$ such that $\max_{Y \subseteq \mathcal{N}_2} f(X'' \cup Y) \leq 2f(X'' \cup Y'') \leq 2 \cdot \max_{Y \subseteq \mathcal{N}_2} f(X'' \cup Y)$.
5 **if** $f(X' \cup Y') \leq 2f(X'' \cup Y'')$ **then return** X'.
6 **else return** X''.

Let us denote the output set of Algorithm 5 by $\hat{X}$, and observe that the choice of the output set in the last five lines of Algorithm 5 guarantees that whenever $\hat{X} = X''$, we also have

$$\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y) = \max_{Y \subseteq \mathcal{N}_2} f(X'' \cup Y) \leq 2f(X'' \cup Y'') \leq f(X' \cup Y') \leq \max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y) \ .$$

Since the inequality $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y) \leq \max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y)$ trivially applies also when $\hat{X} = X'$, we get that this inequality always hold, and therefore, with probability at least $1 - \varepsilon/[8(|\mathcal{N}_2| + 1)]$ we must have

$$\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y) \leq (4 + \varepsilon/2)\tau$$

because Theorem C.2 guarantees that this inequality holds with at least this probability when $\hat{X}$ is replaced with X'. Furthermore, since we always have $\tau = \min_{X\subseteq\mathcal{N}_1} \max_{Y\subseteq\mathcal{N}_2} f(X \cup Y) \leq \max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y)$, the inequality $\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y) \leq \max_{Y\subseteq\mathcal{N}_2} f(X' \cup Y)$ also shows that $\hat{X}$ falls within the range $[\tau, (4+\varepsilon)\tau]$ whenever X' falls within the this range.

Next, we need to prove a second upper bound on $\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y)$. By the choice of the output set in the last five lines of Algorithm 5, when this output set is X', we have

$$\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y) = \max_{Y\subseteq\mathcal{N}_2} f(X' \cup Y) \leq 2f(X' \cup Y') \leq 4f(X'' \cup Y'') \leq 4 \cdot \max_{Y\subseteq\mathcal{N}_2} f(X'' \cup Y) \enspace .$$

Since the non-negativity of f implies that the inequality $\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y) \leq 4 \cdot \max_{Y\subseteq\mathcal{N}_2} f(X'' \cup Y)$ applies also when $\hat{X} = X''$, we get by Theorem 2.4,

$$\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y) \leq 4 \cdot \max_{Y\subseteq\mathcal{N}_2} f(X'' \cup Y) \leq 4(|\mathcal{N}_2| + 1)\tau \enspace .$$

We are now ready to prove that the expectation of the expression $\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y)$ falls within the range $[\tau, (4+\varepsilon)\tau]$ as is guaranteed by Theorem 2.5. The expectation is at least the lower end of this range because, as mentioned above, it always holds that $\tau = \min_{X\subseteq\mathcal{N}_1} \max_{Y\subseteq\mathcal{N}_2} f(X \cup Y) \leq \max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y)$. Additionally, by the law of total expectation and the two above proved upper bounds on $\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y)$,

$$\begin{aligned}\mathbb{E}\left[\max_{Y\subseteq\mathcal{N}_2} f(\hat{X} \cup Y)\right] &\leq \left(1 - \frac{\varepsilon}{8(|\mathcal{N}_2|+1)}\right) \cdot \left(4 + \frac{\varepsilon}{2}\right)\tau + \frac{\varepsilon}{8(|\mathcal{N}_2|+1)} \cdot 4(|\mathcal{N}_2|+1)\tau \\ &= \left[\left(4 + \frac{\varepsilon}{2}\right) + \frac{\varepsilon}{2}\right]\tau = (4+\varepsilon)\tau \enspace . \qquad \square\end{aligned}$$

It remains to prove Theorem C.2. The algorithm that we use for this purpose is given as Algorithm 6. We note that the function g defined by this algorithm is the average of m submodular functions (since f is submodular once the subset of $\mathcal{N}_2$ in the argument set is fixed), and therefore, g is also submodular.

Algorithm 6: Estimating the min-max via random subsets (ε)

1. Let $n_1 = |\mathcal{N}_1|$ and $n_2 = |\mathcal{N}_2|$, and pick $m = \lceil 3200\varepsilon^{-2}[(n_1+1)\ln 2 + \ln(n_2+1) + \ln(8/\varepsilon)]\rceil$ uniformly random (and independent) subsets $Y_1, Y_2, \ldots, Y_m$ of $\mathcal{N}_2$.
2. Define a function $g\colon 2^{\mathcal{N}_1} \to \mathbb{R}_{\geq 0}$ as follows. For every $X \subseteq \mathcal{N}_1$, $g(X) \triangleq \frac{1}{m}\sum_{i=1}^{m} f(X \cup Y_i)$.
3. Use an unconstrained submodular minimization algorithm to find a set $X' \subseteq \mathcal{N}_1$ minimizing $g(X')$.
4. **return** the set X' and the value $(4+\varepsilon/2) \cdot g(X')$.

The analysis of Algorithm 6 uses the following known lemma.

Lemma C.3 (Lemma 2.2 of Feige et al. (2011))**.** *Given a submodular function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ and two sets $A, B \subseteq \mathcal{N}$, if $A(p)$ and $B(q)$ are independent random subsets of A and B, respectively, such that $A(p)$ includes every element of A with probability p (not necessarily independently), and $B(q)$ includes every element of B with probability q (again, not necessarily independently), then*

$$\mathbb{E}[f(A(p) \cup B(q))] \geq (1-p)(1-q) \cdot f(\varnothing) + p(1-q) \cdot f(A) + (1-p)q \cdot f(B) + pq \cdot f(A \cup B) \enspace .$$

Given a vector $\mathbf{x} \in [0,1]^{\mathcal{N}_2}$, we define $\mathsf{R}(\mathbf{x})$ to be a random subset of $\mathcal{N}_2$ that includes every element $u \in \mathcal{N}_2$ with probability x_u, independently. Given a set $S \subseteq \mathcal{N}_2$, it will also be useful to denote by $\mathbf{1}_S$ the characteristic vector of S, i.e., the vector in $[0,1]^{\mathcal{N}_2}$ that has 1 in the coordinates corresponding to the elements of S, and 0 in the other coordinates. Using this notation, we can now define a function $h\colon 2^{\mathcal{N}_1} \to \mathbb{R}_{\geq 0}$ as follows. For every set $X \subseteq \mathcal{N}_1$, $h(X) \triangleq \mathbb{E}[f(X \cup \mathsf{R}(1/2 \cdot \mathbf{1}_{\mathcal{N}_2}))]$. The following lemma shows that $h(X)$ is related to $\max_{Y\subseteq\mathcal{N}_2} f(X \cup Y)$.

Lemma C.4. *For every set $X \subseteq \mathcal{N}_1$, $\max_{Y\subseteq\mathcal{N}_2} f(X \cup Y) \leq 4h(X)$.*

Proof. Let us denote by $Y'(X)$ an arbitrary set in $\arg\max_{Y\subseteq\mathcal{N}_2} f(X \cup Y)$, and define $r_X(Y) \triangleq f(X \cup Y)$. Then,

$$\begin{aligned}h(X) &= \mathbb{E}[f(X \cup \mathsf{R}(1/2 \cdot \mathbf{1}_{\mathcal{N}_2})] = \mathbb{E}[r_X(\mathsf{R}(1/2 \cdot \mathbf{1}_{Y'(X)}) \cup \mathsf{R}(1/2 \cdot \mathbf{1}_{\mathcal{N}_2 \setminus Y'(X)}))] \\ &\geq \tfrac{1}{4} r_X(Y'(X)) = \tfrac{1}{4} f(X \cup Y'(X)) = \tfrac{1}{4} \cdot \max_{Y\subseteq\mathcal{N}_2} f(X \cup Y) \enspace ,\end{aligned}$$

where the inequality follows from Lemma C.3 and the observation that for any fixed set $X \subseteq \mathcal{N}_1$ the function r_X is a non-negative submodular function. $\square$

The last lemma shows that the function h is useful. The following lemma complements the picture by showing that the function g defined by Algorithm 6 is a good approximation of h.

Lemma C.5. *With probability at least $1 - \varepsilon/[8(n_2 + 1)]$, for every set $X \subseteq \mathcal{N}_1$ (at the same time) we have $|g(X) - h(X)| \leq (\varepsilon/20) \cdot h(X)$.*

Proof. Fix some set $X \subseteq \mathcal{N}_1$, and let us define $Z_i \triangleq f(X \cup Y_i)$ for every integer $1 \leq i \leq m$. We would like to study the properties of the random variables $Z_1, Z_2, \ldots, Z_m$. First, note that these random variables are independent since the sets $Y_1, Y_2, \ldots, Y_m$ are chosen independently by Algorithm 6. Second, by the definition of h and the distribution of Y_i, $\mathbb{E}[Z_i] = \mathbb{E}[f(X \cup Y_i)] = h(X)$. We would also like to bound the range of values that the random variables $Z_1, Z_2, \ldots, Z_m$ can take. On the one hand, these random variables are non-negative since f is non-negative. On the other hand, $Z_i = f(X \cup Y_i) \leq \max_{Y \subseteq \mathcal{N}_2} f(X \cup Y) \leq 4h(X)$, where the second inequality holds by Lemma C.4.

Given the above proved properties of the random variables $Z_1, Z_2, \ldots, Z_m$, Hoeffding's inequality shows that

$$\Pr[|g(X) - h(X)| \leq (\varepsilon/20) \cdot h(X)] = \Pr\left[\left|\tfrac{1}{m}\sum_{i=1}^{m} Z_i - \mathbb{E}\left[\tfrac{1}{m}\sum_{i=1}^{m} Z_i\right]\right| > (\varepsilon/20) \cdot h(X)\right]$$
$$\leq 2e^{-\frac{2m^2[(\varepsilon/20)\cdot h(X)]^2}{\sum_{i=1}^{m}[4h(X)]^2}} = 2e^{-\frac{m\varepsilon^2}{3200}} \leq 2e^{-(n_1+1)\ln 2 - \ln(n_2+1) - \ln(8/\varepsilon)} = 2^{-n_1} \cdot \frac{\varepsilon}{8(n_2+1)} \enspace,$$

where the last inequality follows from the definition of m. The lemma now follows from the last inequality by the union bound since X was chosen as an arbitrary subset of $\mathcal{N}_1$, and there are only 2^{n_1} such subsets. $\square$

Using the above lemmata, we can now prove Theorem C.2.

Proof of Theorem C.2. Let us denote by X^* a set minimizing $\max_{Y \subseteq \mathcal{N}_2} f(X^* \cup Y)$. By the definition of g and the choice of X' by Algorithm 6,

$$g(X') \leq g(X^*) = \tfrac{1}{m}\sum_{i=1}^{m} f(X^* \cup Y_i) \leq \max_{Y \subseteq \mathcal{N}_2} f(X^* \cup Y) = \min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2} f(X^* \cup Y) = \tau \enspace.$$

Let us now denote by $\mathcal{E}$ the event that $|g(X) - h(X)| \leq (\varepsilon/20) \cdot h(X)$ for every set $X \subseteq \mathcal{N}$. By Lemma C.5, $\mathcal{E}$ happens with probability at least $1 - \varepsilon/[8(n_2 + 1)]$. Furthermore, conditioned on $\mathcal{E}$, we have

$$\max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y) \leq 4h(X') \leq \frac{4g(X')}{1 - \varepsilon/20} \leq \frac{4\tau}{1 - \varepsilon/20} \leq (4 + \varepsilon/2)\tau \enspace, \tag{5}$$

where the first inequality hold by Lemma C.4, and the last inequality holds for $\varepsilon \in (0, 1/2)$. Since we always also have $\max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y) \geq \min_{X \subseteq \mathcal{N}_1} \max_{Y \subseteq \mathcal{N}_2} f(X \cup Y) = \tau$, the above inequality already proves that $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ falls within the range $[\tau, (4 + \varepsilon/2)\tau]$ whenever the event $\mathcal{E}$ happens.

We now would like to show that the output value $(4 + \varepsilon/2) \cdot g(X')$ of Algorithm 6 also falls within this range when the event $\mathcal{E}$ happens. Since we already proved that $g(X') \leq \tau$, all we need to show is that $(4 + \varepsilon/2) \cdot g(X')$ is at least τ condition on $\mathcal{E}$. This is indeed the case since Inequality (5) implies

$$(4 + \varepsilon/2) \cdot g(X') \geq \frac{4}{1 - \varepsilon/20} \cdot g(X') \geq \max_{Y \subseteq \mathcal{N}_2} f(X' \cup Y) \geq \tau \enspace,$$

where the first inequality holds for $\varepsilon \in (0, 1/2)$. In conclusion, we have shown that when the event $\mathcal{E}$ happens the value $(4 + \varepsilon/2) \cdot g(X')$ returned by Algorithm 6 and the expression $\max_{Y \subseteq \mathcal{N}_2} f(\hat{X} \cup Y)$ both fall within the range $(4 + \varepsilon/2)$. Since the probability of the event $\mathcal{E}$ is at least $1 - \varepsilon/[8(n_2 + 1)]$, to complete the proof of the theorem it only remains to show that the expectation of $(4 + \varepsilon/2) \cdot g(X')$ falls within the range $[\tau, (4 + \varepsilon/2)\tau]$, which is what we do in the rest of this proof.

The inequality $\mathbb{E}[(4+\varepsilon/2)\cdot g(X')] \leq (4+\varepsilon/2)\tau$ follows immediately from the above proof that we deterministically have $g(X') \leq \tau$. Using Inequality (5), we can also get that, conditioned on $\mathcal{E}$,

$$\begin{aligned} g(X') &\geq \frac{(1-\varepsilon/20)\cdot \max_{Y\subseteq \mathcal{N}_2} f(X'\cup Y)}{4} \\ &\geq \frac{(1-\varepsilon/20)\cdot \min_{X\subseteq \mathcal{N}_1}\max_{Y\subseteq \mathcal{N}_2} f(X\cup Y)}{4} = \frac{(1-\varepsilon/20)\tau}{4} \enspace . \end{aligned}$$

Thus, we can use the law of total expectation to get

$$\mathbb{E}[g(X')] \geq \Pr[\mathcal{E}]\cdot \mathbb{E}[g(X') \mid \mathcal{E}] \geq \left(1-\frac{\varepsilon}{8(n_2+1)}\right)\cdot \frac{(1-\varepsilon/20)\tau}{4} \geq \left(\frac{1-9\varepsilon/80}{4}\right)\tau \enspace ,$$

which implies

$$\mathbb{E}[(4+\varepsilon/2)\cdot g(X')] \geq \frac{(4+\varepsilon/2)\cdot(1-9\varepsilon/80)}{4}\cdot \tau \geq \tau \enspace ,$$

where the second inequality holds for $\varepsilon \in [0, 1/2]$. □

D ADDITIONAL APPLICATIONS

D.1 Adversarial Attack on Image Summarization

In this section we consider the application of "Adversarial Attack on Image Summarization", which is an attack version of an application studied by many previous works (see, e.g., Mitrovic et al. (2018); Mualem and Feldman (2022b); Tschiatschek et al. (2014)). The setting for this application includes a collection of images from ℓ disjoint categories (such as birds, airplanes or cats), and a user that specifies $r \in [\ell]$ categories of interest. In the classical version of this application, the objective is to construct a subset of k images summarizing the images belonging to the categories specified by the user. However, here we are interested in mounting an attack against this summarization task. Specifically, our goal is to add a few additional images to the original set of images in a way that undermines the quality of any subsequently chosen summarizing subset.

Formally, we have in this application a (completed) similarity matrix M comprising similarity scores for both the set $\mathcal{N}_2$ of original images and the set $\mathcal{N}_1$ of images that the attacker may add. We aim to choose a set $X \subseteq \mathcal{N}_1$ of images such that adding the images of $\mathcal{N}_1 \setminus X$ simultaneously minimizes the value every possible summarizing subset Y. The value of a summarizing set Y is given by the following objective function.

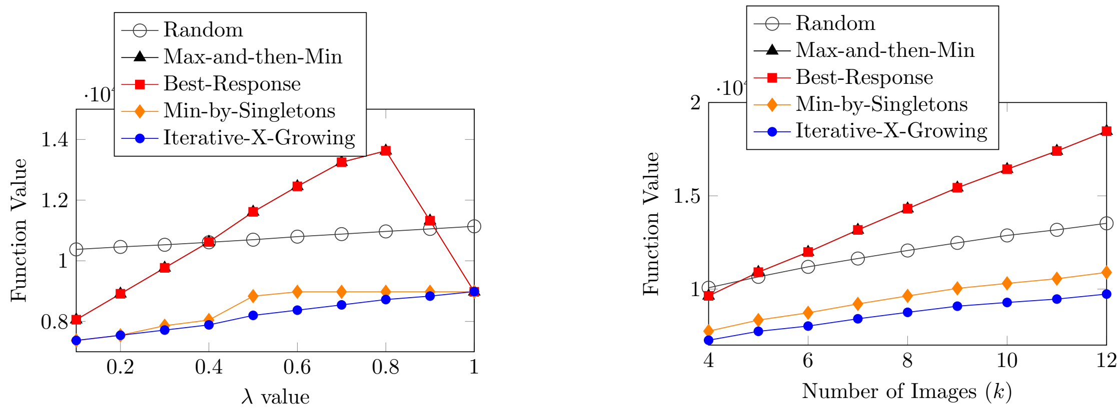
$$f(X\cup Y) = \sqrt[3]{\sum_{v\in\mathcal{N}\setminus X}\sum_{u\in Y} M_{u,v}^3} - \frac{1}{|\mathcal{N}_2|}\sqrt[3]{\sum_{u\in Y}\sum_{v\in Y} M_{u,v}^3} + \lambda\cdot|X|\cdot\sqrt[3]{k} \enspace . \tag{6}$$

Here, $M_{u,v}$ is the similarity score between images u and v, which is assumed to be non-negative and symmetric (i.e., $M_{u,v} = M_{v,u} \geq 0$); and $\lambda \in [0, 1]$ is a regularization parameter affecting the number of elements added by the adversary. Choosing a larger value for λ results in a larger set X, and thus, less adversarial images being added. The objective function f is jointly-submodular and non-negative (the proof is very similar to the proof that the function in Equation (7) has these properties, and therefore, we omit it). Since we are interested in finding an attacker set X that is good against the best summary set Y of size k, the optimization problem that we aim to solve is

$$\min_{X\subseteq \mathcal{N}_1}\max_{\substack{Y\subseteq \mathcal{N}_2\\ |Y|\leq k}} f(X\cup Y) \enspace .$$

Our experiments for this application are based on a subset of the CIFAR-10 data set (Krizhevsky, 2009). This subset includes 10,000 tiny images belonging to 10 classes. Each image consists of 32×32 RGB pixels, and is thus, represented by a 3,072 dimensional vector, and the cosine similarity method was used to compute similarities between images. In order to keep the running time computationally tractable, we randomly sampled from the data set in each experiment disjoint sets $\mathcal{N}_1$ and $\mathcal{N}_2$ of sizes $|\mathcal{N}_1| = 2{,}000$ and $|\mathcal{N}_2| = 250$.

In our experiments, we study the change in the quality of the summaries obtained by the various algorithms and benchmarks as a function of the allowed number k of images and the regularization parameter λ. Figure 3a presents



(a) Results for $k = 5$

(b) Results for $\lambda = 0.5$

Figure 3: Empirical results for adversarial attack on image summarization. Both plots compares the performance of our algorithms Min-by-Singletons and Iterative-X-Growing with 3 benchmarks for different value of the regularization parameter λ and the cardinality parameter k.

the outputs of our algorithms Min-by-Singletons and Iterative-X-Growing (with $\beta = 0.2$) and three benchmarks for $k = 5$ and a varying regularization parameter λ. Figure 3b presents the outputs of the same algorithms and benchmarks for $\lambda = 0.5$ and a varying limitation k on the number of images in the summary. One can observe that both of our algorithms consistently outperform the benchmarks of Best-Response, Max-and-then-Min and Random, with the more involved algorithm Iterative-X-Growing tending to do better than the simpler algorithm Min-by-Singletons. Both figures are based on averaging 400 executions of the algorithms, leading to a standard error of the mean of less than 10 for all data points. It is also worth noting that the basic scarecrow benchmark "Random" outperforms the Best-Response and Max-and-then-Min benchmarks in many cases. This hints that the last heuristics are unreliable despite being natural, and highlights the significance of the methods we propose.

D.2 Robust Ride-Share Optimization

In the "Robust Ride-Share Optimization" application, our primary objective is to determine the most suitable waiting locations for idle taxi drivers based on taxi order history. This problem was previously formalized as a traditional facility location problem (Mitrovic et al., 2018). However, in the current work, we look for a more robust set of waiting locations. Often some locations are inaccessible (for example, due to road maintenance). Hence, we wish to find a robust set of waiting locations that effectively minimizes the distance between each customer and her closest driver even when some of the locations are inaccessible.

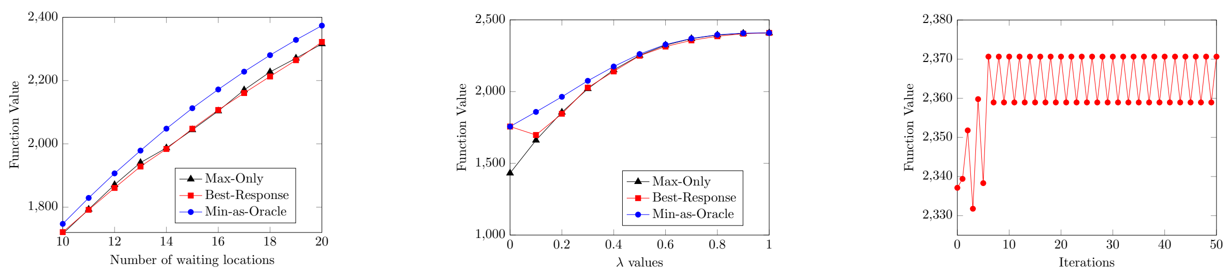
The objective function we use to solve the above problem is technically identical to the jointly-submodular function given by (3). However, now $\mathcal{N}_1$ represents the (client) pickup locations that might be inaccessible due to traffic (while $\mathcal{N}_2$ remains the set of potential waiting locations for idle drivers). Furthermore, we now need to perform max-min optimization on this objective function since we look for a set Y of up to k waiting locations that is good regardless of which pickup locations become inaccessible.

In our experiments, we used again the Uber data set (Uber) (see Section 3.2). To ensure computational tractability, in each execution of our experiments, we randomly selected from this data set a subset of $|\mathcal{N}| = 6{,}000$ pickup locations within the region of Manhattan. Then, we chose the set $\mathcal{N}_1$ to consist of all the pickup locations that have a latitude value greater than 40.8, or less than 40.73. This set represents the pickup locations that are potentially unavailable (for example, due to traffic). Furthermore, we randomly selected a subset of 400 pickup locations from the set $\mathcal{N}$ to constitute the set $\mathcal{N}_2$. This set represents the potential waiting locations for idle drivers.

In the first experiment, we fixed the regularization parameter λ to 0.35 and varied the number of allowed waiting locations. Figure 4a depicts the outputs for this experiment for our algorithm Min-as-Oracle and two benchmarks (averaged over 10 executions of the experiment). One can observe that Min-as-Oracle consistently surpasses the two benchmarks. The two other benchmarks (Random and Top-k) where also included in this experiment and the next one, but are excluded from the figures since their outputs are worse by a factor of at least 2 comapred to the presented methods. We also note that in both experiments the standard error of the mean is less than 10 for all data points.

In the second experiment, we fixed the maximum number of waiting locations to be 15, and varied λ. The results of this experimented are depicted by Figure 4b (again, averaged over 10 executions of the experiment). Once again, our proposed method, Min-as-Oracle, demonstrates superior performance compared to the bechnmarks, with the gap being significant for lower values of λ.

As the third experiment for this application, we conducted a more in depth analysis of the Best-Response technique. Figure 4c graphically presents the objective function value obtained by a typical execution of Best-Response after a varying number of iterations (for $\lambda = 0.35$ and an upper bound of 20 on the number of waiting locations). It is apparent that Best-Response does not converge for this execution. Furthermore, Min-as-Oracle demonstrates better performance even with respect to the best performance of Best-Response for any number of iterations between 1 and 50.



(a) $\lambda = 0.35$

(b) 15 waiting locations

(c) Behavior of Best-Response for $\lambda = 0.35$ and 20 waiting locations

Figure 4: Empirical results for robust ride-share optimization. Figures (a) and (b) compare the performance of our algorithm Min-as-Oracle with 2 benchmarks for different value of λ and bounds on the number of weighting locations. Figure (c) depicts the value of the output of the Best-Response method as a function of the number of iterations performed.

Our last experiment for this section aims to give a more intuitive point of view on the performance of our algorithm (Min-as-Oracle). Figure 5 depicts the results of this algorithm on maps of Manhattan for three different values of λ (0.2, 0.4 and 0.8). To make the maps easy to read, we allowed the algorithm to select only 6 waiting locations for idle drivers, and the locations suggested by the algorithm are marked with red triangles on the maps. We have also marked on the maps the pick up locations of $\mathcal{N}$. The black dots represent the waiting locations that are inaccessible, while the light gray dots indicate the accessible pickup locations. Intuitively, the regularization parameter λ captures in this application the probability of pickup locations in $\mathcal{N}_1$ to be accessible. For example, when $\lambda = 0$, it is assumed that all locations in $\mathcal{N}_1$ are inaccessible, whereas $\lambda = 1$ means that all locations in $\mathcal{N}_1$ are assumed to be accessible. This intuitive role of λ is demonstrated in Figure 5 in the following sense. As the value of λ increases, the number of red triangles in the figure inside the areas of the black dots tends to increase, and furthermore, the locations of theses triangles are pushed deeper into these areas.

D.3 Prompt Engineering for Dialog State Tracking

In this section, we consider the problem of selecting example (input, output) pairs for zero-shot in-context learning. In this application, the objective is to design prompts for the task of dialog state tracking (DST) on the MultiWOZ 2.4 data set (Budzianowski et al., 2018). Following Hu et al. (2022), we recast this as a text-to-SQL problem in order to prompt the GPT-Neo (Black et al., 2021) and OpenAI Codex (`code-davinci-002`) (Chen et al., 2021) code generation models. These base models are adapted to DST with a combination of subset selection and in-context learning. First, a corpus of (previous dialog state, current dialog turn, SQL query) tuples is constructed from the training dialogs. Given a new input u_0, our prompt consists of 1) tabular representations of the dialog state ontology, 2) natural language instructions to query these tables using valid SQL given a task-oriented dialog turn, and 3) examples selected from the corpus by maximizing an objective function.

(a) $\lambda = 0.2$ (b) $\lambda = 0.4$ (c) $\lambda = 0.8$

Figure 5: The results of Min-as-Oracle for 3 different values of λ. The red triangles represent waiting locations chosen by the algorithm, the light gray dots represent always accessible pick-up locations, and the black dots represent possibly inaccessible pick-up locations.

Let u_0 be the input query to a large language model. Each input u_0 contains a list of (key, value) pairs representing the previous dialog state predictions along with the text of the current dialog turn. We would like its prompt to be robust to incorrect predictions of the previous dialog states, as well as text variation such as misspellings. Let $\mathcal{N}_1$ be a set of *perturbed* inputs drawn from a small neighborhood around u_0. These perturbed inputs are constructed by randomly editing up to 2 slots and/or values in the dialog state, and additionally dropping up to 15% of tokens from the most recent dialog turn. In cases where u_0 is initially incorrect, examples that are similar to the perturbed inputs from $\mathcal{N}_1$ improve the final prompt. Let $\mathcal{N} = \mathcal{N}_1 \cup \{u_0\}$, and let $\mathcal{N}_2$ be the ground set of candidate examples. Given a set of examples $Y \subseteq \mathcal{N}_2$ and a set of perturbed inputs $X \subseteq \mathcal{N}_1$, we define the following score function.

$$f(X \uplus Y) = \sum_{u \in \mathcal{N} \setminus X} \sum_{v \in Y} s_{u,v} - \frac{\alpha}{|\mathcal{N}_2|} \cdot \sum_{v \in Y} \sum_{u \in Y} s_{u,v} + \lambda \cdot |X| + |\mathcal{N}_2| \ . \tag{7}$$

Here, $0 \leq s_{u,v} \leq 1$ is the symmetric similarity score between examples u, v (the similarity score is computed by embedding both examples with a pretrained SBERT model (Reimers and Gurevych, 2019), and then computing cosine similarity of the two embeddings), and $\lambda \geq 0$ and $0 \leq \alpha \leq 1$ are regularization parameters. The parameter α explicitly trades off recommendation quality and diversity. Since we are interested in finding a set of candidate examples Y that is good against the worst case set X of perturbed inputs, we would like to optimize $\max_{Y \subseteq \mathcal{N}_2, |Y| \leq k} \min_{X \in \mathcal{N}_1} f(X \uplus Y)$, where k is an upper bound on the number of examples to include in the prompt. By an argument similar to the proof of Lemma 3.2, the objective function (7) is a non-negative jointly-submodular function.

Initially, the GPT-Neo-Small generative model was evaluated with all possible combination of values for the regularization parameters from the grid $\lambda \in \{0, 0.5, 0.75, 0.9, 2.5\}$ and $\alpha \in \{0, 0.1, 0.3, 0.5, 0.7\}$. The best parameters ($\lambda = 0.9, \alpha = 0.5$) were then used for the other generative models. Following Section 5 of Hu et al. (2022), all retrieval models were evaluated on inputs obtained by randomly sampling 10% of the MultiWOZ validation set, and all results were averaged over 3 different candidate sets, which are randomly sampled 5%

subsets of the MultiWOZ training set. We set ($k = 5$, $|\mathcal{N}_1| = 20$) for GPT-Neo models and ($k = 10$, $|\mathcal{N}_1| = 4$) for the OpenAI Codex model.

In our experiments, we have compared Min-as-Oracle with the our standard max-min benchmarks Top-k, Max-Only and Best-Response, and also with a baseline termed "Non-robust Top-k" from Hu et al. (2022). For Top-k, Max-Only, Best-Response and Min-as-Oracle, we first retrieved a ground set of size $|\mathcal{N}_2| = 100k$ candidates using the precomputed KD Tree, and only then selected the output set Y using the retrieval algorithm. Table 3 shows the Joint F1 score for each of the above-mentioned methods. Results for GPT-Neo models are averaged over 4 random seeds. One can observe that prompting with our robust formulation outperforms the Non-Robust Top-k baseline by as much as 1.5%. Among the algorithms using the robust formulation, our proposed algorithm Min-as-Oracle is consistently the best or 2nd best. Table 3 also shows that Min-as-Oracle achieves the highest objective value in all cases. Note that Min-as-Oracle has theoretical guarantees for both its convergence and approximation ratio, whereas Sections 3.2 and D.2 demonstrate that the Best-Response heuristic diverges for some instances.

Table 3: Dialog state tracking performance and objective values for different language models and retrieval algorithms. Best values are in **bold**.

Generative Model	Retrieval Algorithm	Joint F1	Objective Value
GPT-Neo-Small	Random	0.0480	25.259
	Non-robust Top-k (Hu et al., 2022)	0.3249	26.165
	Top-k	0.2787	26.125
	Max-Only	0.2783	26.134
	Best-Response	**0.3251**	26.165
	Min-as-Oracle	0.3022	**26.168**
GPT-Neo-Large	Random	0.2275	25.254
	Non-robust Top-k (Hu et al., 2022)	0.4872	26.164
	Top-k	0.4821	26.127
	Max-Only	0.4830	26.134
	Best-Response	0.4845	26.165
	Min-as-Oracle	**0.5020**	**26.168**
Codex-Davinci	Random	0.8273	17.410
	Non-robust Top-k (Hu et al., 2022)	0.8929	19.021
	Top-k	0.8974	18.954
	Max-Only	0.8913	18.953
	Best-Response	0.8972	19.022
	Min-as-Oracle	**0.8991**	**19.027**

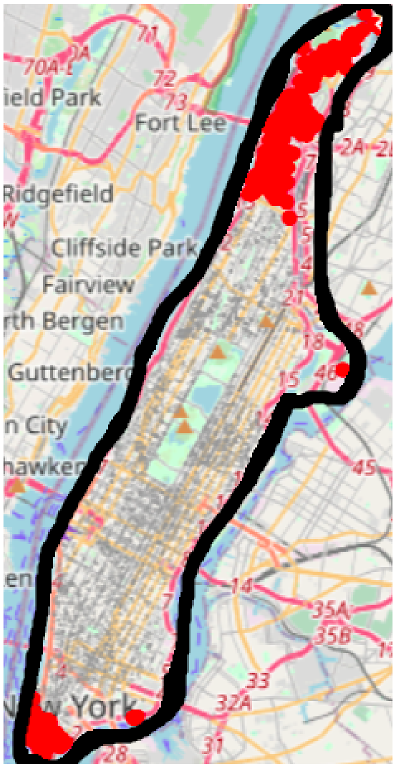
D.4 Additional Figures For Ride-Share Difficulty Kernelization

In this section, we provide a graphical representation of the experimental outcomes discussed in Section 3 for the Ride-Share Difficulty Kernelization application. Figure 6 demonstrates that the locations chosen by our algorithm, based on the objective function (3) used for the application, create a spatial arrangement resembling a "frame" encompassing the Manhattan area. This agrees with the intuitive expectation from a well-structured kernelization, demonstrating that the objective function (3) is a good fit for the Ride-Share Difficulty Kernelization application.

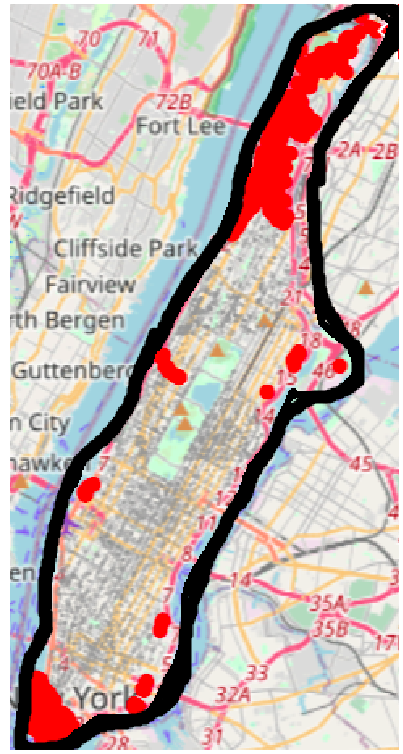
E BENCHMARKS AND ALGORITHM IMPLEMENTATIONS

In this section we define all the benchmarks that we compare in Section 3 and Section D against our algorithms. We then discuss the implementation details of these benchmarks and our algorithms.

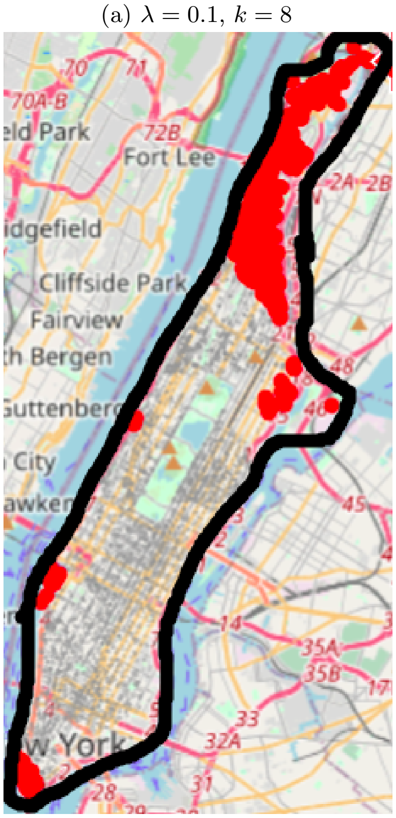
- Random: Returns a random feasible solution. In the max-min setting this means random k elements from the ground set $\mathcal{N}_1$, and in the min-max setting this means a random subset of $\mathcal{N}_2$.
- Max-Only: This benchmark makes sense only in the max-min setting. It uses a submodular maximization algorithm to find a feasible set Y (approximately) maximizing the objective for $X = \varnothing$.



(a) $\lambda = 0.1$, $k = 8$



(b) $\lambda = 0.25$, $k = 8$



(c) $\lambda = 0.1$, $k = 4$

(d) $\lambda = 0.25$, $k = 4$

Figure 6: The results of our algorithm Iterative-X-Growing for different values of λ (the regularization parameter) and k (the number of taxis). The red dots represent the pick-up locations in the difficulty kernel chosen by the algorithm (the other pick-up locations are marked by light gray dots).

- Max-and-then-Min: A variant of Max-Only for use in the min-max setting. It returns a set X minimizing the objective given the set Y chosen by Max-Only. Note that this is essentially equivalent to a single iteration of Best-Response.
- Top-k: This benchmark makes sense only in the max-min setting. It returns the k singletons from $\mathcal{N}_2$ with the maximum value, where the value of every singleton $u \in \mathcal{N}_2$ is defined as $\min_{X \subseteq \mathcal{N}_1} f(X \cup \{u\})$.
- Best-Response: This benchmark proceeds in iterations. In the first iteration, one obtains a subset $Y \in \mathcal{F}_2$ (approximately) maximizing $f(Y)$ through the execution of a maximization algorithm, which is followed by finding a set $X \subseteq \mathcal{N}_2$ minimizing $f(X \cup Y)$ by running a minimization algorithm. Subsequent iterations are similar to the first iteration, except that the set Y chosen in these iterations is a set that (approximately) maximizes $f(X \cup Y)$, where X is the minimizing set chosen in the previous iteration. The output is then the last set Y in the max-min setting, and the last set X in the min-max setting.

In most of our applications, we aim to optimize objectives that are not $\mathcal{N}_2$-monotone, which requires a procedure for (approximate) maximization of non-monotone submodular functions. As mentioned in Section 1.1, the state-of-the-art approximation guarantee for the case in which the objective function f is not guaranteed to be monotone is currently 0.385 (Buchbinder and Feldman, 2019). However, the algorithm obtaining this approximation ratio is quite involved, which limits its practicality. Arguably, the state-of-the-art approximation ratio obtained by a "simple" algorithm is $1/e$-approximation obtained by an algorithm called Random Greedy (Buchbinder et al., 2014). In practice, the performance of this algorithm is comparable to that of the standard greedy algorithm, despite the last algorithm not having any approximation guarantee for non-monotone objective functions. Hence, throughout the experiments, the maximization component used in all the relevant benchmarks and algorithms is either the standard greedy algorithm or an accelerated version of it (suggested by Badanidiyuru and Vondrák (2014)) named Threshold Greedy.

In our experiment we often report the values of the objective function corresponding to the output sets produced by the various benchmarks and algorithms. In the max-min setting, given an output set X, computing the objective value is done by utilizing an efficient minimizing algorithm to identify a minimizing set X. In the min-max setting, the situation is more involved as calculating the true objective value for given an output set X cannot be done efficiently in sub-exponential time (as it corresponds to maximizing a submodular function subject to a cardinality constraint). Therefore, we use Threshold Greedy algorithm mentioned above to find a set Y that approximately maximize the objective with respect to X, and then report the value corresponding to this set Y as a proxy for the true objective value.

Our experiments for the min-max setting use a slightly modified version of Iterative-X-Growing (Algorithm 1). Specifically, we make two modifications to the algorithm.

- Iterative-X-Growing grows a solution in iterations. As written, it outputs the set obtained after the last iteration. However, we chose to output instead the best set obtained after any number of iterations. This is a standard modification often used when applying to practice an iterative theoretical algorithm.
- Line 4 of Iterative-X-Growing looks for a set X'_i that minimizes an expression involving two terms. The first of these terms $\sqrt{n_1} \cdot f(X \cup X_{i-1})$ has the large coefficient $\sqrt{n_1}$. The value of this coefficient was chosen to fit the largest number of possible iterations that the algorithm may perform $(n_1 + 1)$. However, in practice we found that the algorithm usually makes very few iterations. Thus, the use of the large coefficient $\sqrt{n_1}$ becomes sub-optimal. To truly show the empirical performance of Iterative-X-Growing, we replaced the coefficient $\sqrt{n_1}$ with a parameter β whose value is chosen based on the application in question.

All prompt engineering experiments were run using a single NVIDIA A10 GPU. Running inference with the Contriever and GPT-Neo-Large models required a server with 128GB of memory, and all other parts of the pipeline required less than 16GB of memory.

All other experiments were run using 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz CPU, requiring less than 32GB of CPU memory and no GPU.

F ADDITIONAL OMITTED PROOFS

F.1 Proof of Lemma 3.1

In this section we prove Lemma 3.1, which we repeat here for convenience.

Lemma 3.1. *The objective function* (2) *is a non-negative jointly-submodular function.*

Proof. Observe that the objective function (2) is a conical combination of three terms. Below we explain why each one of these terms is non-negative and jointly-submodular, which immediately implies that the entire objective function also has these properties.

The second term is $\sum_{u\in\mathcal{N}_1\setminus X}\sum_{v\in Y} s_{u,v}$. This term can be viewed as the cut function of a directed bipartite graph in which the elements of $\mathcal{N}_1$ and $\mathcal{N}_2$ form the two sides of the graph, and for every $u\in\mathcal{N}_1$ and $v\in\mathcal{N}_2$ the graph includes an edge from v to u whose weight is $s_{u,v}$. Directed graph functions are known to be non-negative and submodular over the set of elements of the graph, which translates into joint-submodularity in our terminology since the both the elements of $\mathcal{N}_1$ and $\mathcal{N}_2$ are vertices of the graph.

The third term is $|X|$, which is a non-negative linear function, and thus, also jointly-submodular.

It remains to consider the first term, namely $\sum_{v\in\mathcal{N}_1\setminus X}\max_{u\in Y} s_{u,v}$. This term is clearly non-negative, so we concentrate below on proving that it is jointly submodular. Recall that f is jointly-submodular if

$$f(u \mid Y' \cup X') \geq f(u \mid Y \cup X) \qquad \forall\, X' \subseteq X \subseteq \mathcal{N}_1, Y' \subseteq Y \subseteq \mathcal{N}_2, u \in (\mathcal{N}_1 \cup \mathcal{N}_2) \setminus (X \cup Y) \enspace .$$

To prove that our objective function obeys this inequality, there are two scenarios to consider, based on whether u belongs to the set $\mathcal{N}_1$ or $\mathcal{N}_2$.

Case 1: The element u belongs to $\mathcal{N}_1$. Here,

$$f(u \mid Y' \cup X') - f(u \mid Y \cup X) = \max_{v\in Y} s_{u,v} - \max_{v\in Y'} s_{u,v} \geq 0 \enspace .$$

Case 2: the element u belongs to $\mathcal{N}_2$. Observe that, in this case,

$$\begin{aligned} f(u \mid Y' \cup X') &= \sum_{\mathcal{N}_1\setminus X'} \max\{0, s_{u,v} - \max_{v'\in Y'} s_{u,v'}\} \\ &\geq \sum_{\mathcal{N}_1\setminus X} \max\{0, s_{u,v} - \max_{v'\in Y} s_{u,v}\} = f(u \mid Y \cup X) \enspace . \end{aligned}$$

□

F.2 Proof of Lemma 3.2

In this section we prove Lemma 3.2, which we repeat here for convenience.

Lemma 3.2. *The objective function* (3) *is a non-negative jointly-submodular function.*

Proof. First, we shall establish that the objective function is non-negative by demonstrating that the first term of the function is consistently greater than the subsequent term. This is established through the following inequality.

$$\sum_{v\in\mathcal{N}\setminus X}\max_{u\in Y} s_{u,v} \geq \sum_{v\in\mathcal{N}_2}\max_{u\in Y} s_{u,v} \geq \frac{1}{|Y|}\cdot\sum_{v\in\mathcal{N}_2}\sum_{u\in Y} s_{u,v} \geq \frac{1}{|\mathcal{N}_2|}\cdot\sum_{v\in Y}\sum_{u\in Y} s_{u,v} \enspace .$$

Next, we demonstrate that the objective function $f(X, Y)$ is jointly-submodular. Recall that f is jointly-submodular if

$$f(u \mid Y' \cup X') \geq f(u \mid Y \cup X) \qquad \forall\, X' \subseteq X \subseteq \mathcal{N}_1, Y' \subseteq Y \subseteq \mathcal{N}_2, u \in (\mathcal{N}_1 \cup \mathcal{N}_2) \setminus (X \cup Y) \enspace .$$

To prove that our objective function obeys this inequality, there are two scenarios to consider, based on whether u belongs to the set $\mathcal{N}_1$ or $\mathcal{N}_2$.

Case 1: The element u belongs to $\mathcal{N}_1$. Here, $f(u \mid Y' \cup X') - f(u \mid Y \cup X) = \max_{v \in Y} s_{u,v} - \max_{v \in Y'} s_{u,v} \geq 0$.

Case 2: The element u belongs to $\mathcal{N}_2$. Let $\phi(Y, w, J) = \sum_{v \in J}(\max_{u \in Y+w} s_{u,v} - \max_{u \in Y} s_{u,v})$; and note that, for every two sets $Y' \subseteq Y \subseteq \mathcal{N}_2$, set $J \subseteq \mathcal{N}_1$ and element $u \in \mathcal{N}_2 \setminus T$, $\phi(Y', u, J) \geq \phi(Y, u, J)$. Using this notation, we get that in this case (the case of $u \in \mathcal{N}_2$)

- $f(u \mid Y' \cup X') = \phi(Y', \{u\}, \mathcal{N} \setminus X') - \frac{1}{|\mathcal{N}_2|}\big(2 \cdot \sum_{v \in Y'} s_{u,v} + s_{u,u}\big)$, and
- $f(u \mid Y \cup X) = \phi(Y, \{u\}, \mathcal{N} \setminus X) - \frac{1}{|\mathcal{N}_2|}\big(2 \cdot \sum_{v \in Y} s_{u,v} + s_{u,u}\big)$.

Thus,

$$f(u \mid Y' \cup X') - f(u \mid Y \cup X) = \phi(Y', \{u\}, \mathcal{N} \setminus X') - \phi(Y, \{u\}, \mathcal{N} \setminus X) + \frac{2}{|\mathcal{N}_2|} \cdot \sum_{v \in Y \setminus Y'} s_{u,v} \geq 0 \enspace ,$$

where the last inequality holds since $\phi(Y', \{u\}, \mathcal{N} \setminus X') - \phi(Y, \{u\}, \mathcal{N} \setminus X) \geq 0$ and $s_{u,v} \geq 0$ for any $u, v \in \mathcal{N}$. $\square$