MINI-BATCH SUBMODULAR MAXIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We present the first *mini-batch* algorithm for maximizing a non-negative monotone *decomposable* submodular function, $F = \sum_{i=1}^{N} f^i$, under a set of constraints. Our experiments demonstrate that a straightforward uniform mini-batch sampling approach significantly outperforms existing state-of-the-art sparsifier methods, requiring only a fraction of their running time. However, explaining this improvement via worst-case analysis is *impossible*.

Under mild assumptions, we show that uniform sampling is superior to weighted sampling for both the mini-batch and sparsifier approaches. We further verify empirically that these assumptions hold across various datasets. Unlike weighted sampling, uniform sampling is simple to implement and several orders of magnitude faster, making it ideal for handling massive real-world datasets.

1 INTRODUCTION

Submodular functions capture the natural property of *diminishing returns* which often arises in machine learning, graph theory and economics. For example, imagine you are given a large set of images, and your goal is to extract a small subset that best represents the original set (e.g., creating thumbnails for a YouTube video). Intuitively, this is a submodular optimization problem, as the more thumbnails we have, the less we gain by adding an additional thumbnail. Formally, a set function $F : 2^E \to \mathbb{R}^+$, on a *ground set* E, is submodular if for any subsets $S \subseteq T \subseteq E$ and $e \in E \setminus T$, it holds that $F(S + e) - F(S) \geq F(T + e) - F(T)$.

Decomposable submodular functions In many natural scenarios F is *decomposable*: $F = \sum_{i=1}^{N} f^i$, where each $f^i : 2^E \to \mathbb{R}^+$ is a non-negative submodular function on the ground set E with $|E| = n$. That is, F can be written as a sum of "simple" submodular functions. We assume that every f^i is represented by an evaluation oracle that, when queried with $S \subseteq E$, returns the value $f^i(S)$. For ease of notation, we assume that $f^i(\emptyset) = 0$ (our results hold even if this is not the case). Our goal is to maximize F under some set of constraints while minimizing the number of oracle calls to $\{f^i\}$. An excellent survey of the importance of decomposable functions is given in [37], which we summarize below.

Decomposable submodular functions are prevalent in both machine learning and economics. In economics, they play a pivotal role in welfare optimization during combinatorial auctions [8, 11, 12, 31, 44]. In machine learning, they are instrumental in tasks like data summarization, aiming to select a concise yet representative subset of elements. Their utility spans various domains, from exemplar-based clustering by [9] to image summarization [43], recommender systems [32] and document summarization [21]. The optimization of these functions, especially under specific constraints (e.g., cardinality, matroid), has been studied in various data summarization settings [25–27] and differential privacy [5, 28, 36]. In many of the above applications, N (the number of underlying submodular functions) is extremely large, making the evaluation of F prohibitively slow. We illustrate this with a simple example.

Example (Welfare maximization) Imagine you are tasked with deciding on a meal menu for a large group of N people (e.g., all students in a university, all high school students in a country). You need to choose k ingredients to use from a predetermined set (chicken, fish, beef, etc.) of size n. Every student has a specific preference, modeled as a monotone submodular function f^i. Our goal is to maximize the "social welfare", $F(S) = \sum f^i(S)$, over all students. We use this meal menu example for simplicity and ease of presentation, but the same framework applies to more meaningful domains such as public service allocation, medical benefit selection, or emergency response planning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The greedy algorithm Going forward we focus on the case where $\forall i$, f^i is monotone ($\forall S \subseteq T \subseteq E, f^i(T) \geq f^i(S)$), which in turn means that F is monotone. This is applicable to the above example (e.g., students are happier with more food varieties). For ease of presentation, let us first focus on maximizing F under a cardinality constraint k, i.e., $\max F(S), |S| \leq k$. The classical greedy algorithm [30] (presented below) achieves an *optimal* $(1 - 1/e)$-approximation for this problem. Let us define some notation before we continue. For $S, A \subseteq E$ we define $F_S(A) = F(S + A) - F(S)$. We slightly abuse notation and write $F_S(e), F(e)$ instead of $F_S(\{e\}), F(\{e\})$.

`Greedy:` Start by initializing an empty set S_1. Then for each j from 1 to k, do the following: Find the element $e' \in E \setminus S_j$ that maximizes the function F_{S_j} and add it to S_j to form the new set S_{j+1}. After completing all iterations, return S_{k+1}.

When F is decomposable, and each evaluation of f^i is counted as an oracle call, the above algorithm requires $O(Nnk)$ oracle calls. This can be prohibitively expensive if $N \gg n$. Looking back at our example, `Greedy` will require asking every student k questions. That is, we would start by asking all students: "Rate how much you would like to see food x on the menu" for n possible options. We would then need to wait for all of their replies, and continue with "Given that chicken is on the menu, rate how much you would like to see food y on the menu" for $n - 1$ possible options and so on. Even if we do an online poll, this is still very time-consuming (as we must wait for *everyone* to reply in each of the k steps). If we could *sample* a representative subset of the students, we could greatly speed up the above process. Specifically, we would like to eliminate the dependence on N.

A sparsifier-based approach Let $\hat{F} = \sum_{i=1}^{N} w_i f^i$ where $w \in \mathbb{R}^N$. We say that $\hat{F}$ is an ε-sparsifier [37] for F if it satisfies with high probability (w.h.p)[1] that $\forall S \subseteq E, (1 - \varepsilon)F(S) \leq \hat{F}(S) \leq (1 + \varepsilon)F(S)$. The size of the sparsifier is the number of non-zero entries in w. The current state-of-the-art construction [18] sets $w_i = 1/\alpha_i$ with probability $\alpha_i = \min\{p_i \alpha, 1\}$ where $p_i = \max_{e \in E, F(e) \neq 0} \frac{f^i(e)}{F(e)}$ and α is a parameter to control the size of the sparsifier. Assuming the size of the solution is bounded by k, it is possible to construct a sparsifier of size $\widetilde{O}(k^2 n \varepsilon^{-2})$ using $O(Nn)$ oracle calls[2]. Note that, when the size of the solution is bounded by k it is enough for the sparsifier guarantees to only hold for sets of size at most k.

Computing the p_i's is the main bottleneck of this approach, and is treated as a *preprocessing* step in prior work. Nevertheless, it still contributes to the overall query complexity of the algorithm. We count the preprocessing step toward the overall query complexity (i.e., we include all sparsifier-construction queries in the total number of queries performed by the algorithm).

The simplest way to overcome the $O(Nn)$ bottleneck in the sparsifier construction is to use *uniform sampling* (i.e., set $\forall i \in [N], p_i = 1/N$). We experimentally evaluate the uniform sparsifier against the weighted sparsifier and observe that it achieves solutions of equal (or superior) quality while reducing the query complexity by orders of magnitude. Surprisingly, this simple baseline was neglected in previous experimental evaluations [37].

Unfortunately, we cannot get worst-case theoretical guarantees for uniform sampling. Consider the case where only a single f^i takes non-zero values (all other f^i's are always 0). Clearly, uniform sampling will almost surely miss f^i. However, this is a pathological case that is very unlikely to occur in practice. To bridge this gap, we introduce a new structural parameter which we call *singleton density*.

Singleton density We assume, without loss of generality, that $\forall i \in [N], e \in E, f^i(e) \in [0, 1]$ (we can always achieve this by normalization). Let $e^* = \arg\max_{e \in E} F(e)$, breaking ties arbitrarily. We define the *singleton density* of F as $\phi = \frac{1}{N} F(e^*)$. A large ϕ means that the value $F(e^*) = \phi N$ of the best element is accumulated from many components rather than a few, exactly the regime in which a uniform sample estimates it accurately. In the pathological example above, where a single f^i is non-zero, we have $\phi \leq 1/N$, whereas when the components are broadly similar $\phi = \Theta(1)$. Going back to our lunch menu example, ϕ being bounded away from zero simply means that there *exists* a food on the menu that is not universally hated by the students. Finally, this parameterization can also be justified using *smoothed analysis* (discussed in Appendix B).

[1] Probability at least $1 - 1/n^c$ for an arbitrary constant $c > 1$. The value of c does not affect the asymptotics of the results we state (including our own).

[2] Where $\widetilde{O}$ hides $\log n$ factors.

CONTRIBUTION AND TECHNIQUES

Our main contribution is introducing the mini-batch greedy algorithm with uniform sampling (Algorithm 2 in Section 2). This is a natural step, as mini-batch algorithms are widely used in practice (e.g., SGD, mini-batch k-means [39]). Instead of sampling a single sparsifier and executing the algorithm, we sample a new sparsifier (mini-batch) in every iteration. Our algorithm can be seen as an analogue to Sculley's mini-batch k-means algorithm [39], a simple and highly practical method described in a concise two-page paper, which has since been widely adopted in libraries such as scikit-learn [33].

We experimentally observe that the mini-batch algorithm achieves solutions of superior quality to the sparsifier-based approach. We provide theoretical guarantees for both our mini-batch algorithm and the uniform sparsifier.

Theoretical results We start our theoretical analysis (Section 2) by presenting a meta greedy algorithm (Algorithm 1) which captures the behavior of the greedy algorithm under various constraints. We focus on cardinality and p-system constraints. The cardinality constraint was chosen for its simplicity and applicability, while the p-system constraint was chosen for its generality.

It is known that for cardinality and p-system constraints, this algorithm achieves near-optimal approximation guarantees even if we only have a multiplicative approximation to F_{S_j} (the marginal gain given the current partial solution) in every iteration. We generalize these guarantees to hold under an approximation that combines a multiplicative and an *additive* error.

Finally, in Section 3 we show that the mini-batch algorithm with uniform sampling indeed achieves the desired approximation to F_{S_j} in every iteration. Compared to weighted sampling, we eliminate an $O(n)$-multiplicative factor in the query complexity, while incurring a multiplicative $O(1/\phi)$ factor. This gives a smooth transition in the query complexity between the worst-case scenario where $\phi \to 0$ and the well-behaved case where $\phi = \Theta(1)$.

We use similar techniques to show that the uniform sparsifier requires a factor of k more queries than our mini-batch algorithm, as it must approximate F on *sets* rather than on singleton marginals. Furthermore, we introduce a *weighted* mini-batch algorithm and show that it achieves improved *worst-case* guarantees compared to the weighted sparsifier. Our weighted algorithm uses the sampling probabilities of [18], which brings back the $O(Nn)$ bottleneck to the query complexity, making the algorithm impractical. We include it primarily for theoretical completeness.

In the main body of the paper we focus only on cardinality constraints and the uniform mini-batch algorithm for ease of presentation. Nevertheless, this is enough to present our main ideas and contributions. We include a discussion of p-systems in Appendix C, our results for the uniform sparsifier in Appendix E and for the weighted mini-batch algorithm in Appendix F.

We compare our results with the state-of-the-art sparsifier results and the naive algorithm (without sampling or sparsification) in Table 1. **Our main results are the complexity guarantees for uniform sampling parameterized by singleton density, ϕ, which explains why it outperforms weighted sampling in practice**.

Experimental evaluation In Section 4 we compare the uniform mini-batch algorithm against the weighted mini-batch algorithm, the (weighted) sparsifier of [18] and the uniform sparsifier on five datasets: CIFAR100, FashionMNIST, Uber pickups, Discogs, Facebook. We evaluate our algorithm both under a cardinality and a p-system constraint. We observe a considerable performance discrepancy in favor of the uniform mini-batch algorithm for small batch sizes. The weighted methods are orders of magnitude slower due to the $O(Nn)$ factor. We observe similar results when we combine our approach with the popular *stochastic-greedy* algorithm [3, 24].

Empirical validation of the singleton density In addition to evaluating our algorithm, we empirically examine the validity of the assumption that the singleton density is not too small. Specifically, we compute $\phi = \max_e \frac{1}{N} \sum_i f^i(e)$. The empirical values are as follows (dataset, N, ϕ): Facebook, $4{,}039$, 0.04; CIFAR100, $50{,}000$, 0.38; FashionMNIST, $50{,}000$, 0.35; Discogs, $243{,}764$, 0.13; Uber Pickups, $652{,}434$, 0.61. All ϕ values are orders of magnitude above the pathological $1/N$ value.

We then ask a natural follow-up: could real-world datasets support a stronger assumption? In particular, we consider a variant in which F is not too small for *all* $e \in E$, rather than for just a single e^*. To evaluate this, we compute $\min_e \frac{1}{N} \sum_i f^i(e)$, and observe much smaller values:

	Oracle queries	
	Weighted (Worst-case)	Uniform
Naive	$O(Nnk)$ [30]	
Sparsifier	$\widetilde{O}\big(Nn + \frac{k^3n^2}{\varepsilon^2}\big)$ [18]	$\widetilde{O}\big(\frac{k^3n}{\varepsilon^2\phi}\big)$ [this work]
Mini-batch	Card. $\widetilde{O}\big(Nn + \frac{k^2n^2}{\varepsilon^2}\big)$ [this work] p-sys. $\widetilde{O}\big(Nn + \frac{k^2pn^2}{\varepsilon^2}\big)$ [this work]	$\widetilde{O}\big(\frac{k^2n}{\varepsilon^2\phi}\big)$ [this work]
Mini-batch (bounded curvature)	$\widetilde{O}\big(Nn + \frac{kn^2}{(1-c)\varepsilon^2}\big)$ [this work]	-

Table 1: Comparison of the expected number of oracle queries for the greedy algorithm. The results for the uniform column hold in expectation and w.h.p. Results are under both a cardinality constraint and a p-system constraint. Unless explicitly stated, the number of queries is the same for both constraints. All results achieve the near-optimal approximation guarantees of $(1 - 1/e - \varepsilon)$ for a cardinality constraint and $(\frac{1-\varepsilon}{p+1})$ for a p-system constraint.

CIFAR100: 0.34, FashionMNIST: 0.31, Uber Pickup: 2.36×10^{-4}, Discogs: 4.10×10^{-6}, Facebook: 2.5×10^{-4}. For three of the five datasets, the empirical ϕ is essentially zero, indicating that such a strong assumption is unrealistic.

2 THE MINI-BATCH ALGORITHM

Meta greedy algorithm Our starting point is the meta greedy algorithm (Algorithm 1). The algorithm executes for $k \leq n$ iterations, where k is some upper bound on the size of the solution. At every iteration, the set $A_j \subseteq E \setminus S_j$ represents some constraint that limits the choice of potential elements to extend S_j. The algorithm terminates when the solution size reaches k or when no further extensions to the current solution are possible (i.e., $A_j = \emptyset$). Furthermore, the algorithm does not have access to the exact *incremental oracle*, F_{S_j}, at every iteration, but only to some approximation (which may differ between iterations). Before we formally define "approximation", let us note

Algorithm 1: Meta greedy algorithm with an approximate oracle

$S_1 \leftarrow \emptyset$, Let k be an upper bound on the size of the solution
for $j = 1$ *to* k **do**
  Let $A_j \subseteq E \setminus S_j$ `// Problem specific constraint (e.g., ` $A_j = E \setminus S_j$ `for card. constraint)`
  if $A_j = \emptyset$ **then** return S_j
  Let $\hat{F}^j_{S_j}$ be an approximation for F_{S_j} `// Problem specific approximation`
  $e_j = \arg\max_{e \in A_j} \hat{F}^j_{S_j}(e)$
  $S_{j+1} = S_j + e_j$
end
return S_{k+1}

that when we have access to exact values of F_{S_j}, Algorithm 1 captures many variants of the greedy submodular maximization algorithm. For example, setting $A_j = E \setminus S_j$ we get `Greedy`. This meta-algorithm also captures the case of maximization under a *p-system constraint*. For ease of presentation, we defer the discussion about p-systems to Appendix C.

Approximate oracles In many scenarios we do not have access to *exact* values of F_{S_j}, and instead we must make do with an approximation. We start with the notion of an *approximate incremental oracle* introduced in [13]. We say that $\hat{F}^j_{S_j}$ is an $(1 - \varepsilon)$-approximate incremental

oracle if $\forall e \in A_j, (1-\varepsilon)F_{S_j}(e) \leq \hat{F}^j_{S_j}(e) \leq (1+\varepsilon)F_{S_j}(e)$. It was shown in [13, 4][3] that given a $(1-\varepsilon)$-approximate incremental oracle, the greedy algorithm under both a cardinality constraint and a p-system constraint achieves almost the same (optimal) approximation ratio as with an exact oracle.

Theorem 2.1. *Algorithm 1 with a $(1-\varepsilon)$-approximate incremental oracle has the following guarantees: (1) It achieves a $(1-1/e-\varepsilon)$-approximation under a cardinality constraint k. (2) It achieves a $(\frac{1-\varepsilon}{1+p})$-approximation under a p-system constraint.*

We use a weaker notion of approximate incremental oracle, which allows for both a multiplicative and an additive error. Let S^* be some optimal solution for F (under the relevant set of constraints). We say that $\hat{F}^j_{S_j}$ is an $(\varepsilon, \varepsilon')$-*approximate incremental oracle* if $\forall e \in A_j, (1-\varepsilon)F_{S_j}(e) - \varepsilon' F(S^*) \leq \hat{F}^j_{S_j}(e) \leq (1+\varepsilon)F_{S_j}(e) + \varepsilon' F(S^*)$.

This might seem problematic at first glance, as it might be the case that $F(S^*) \gg F_{S_j}(e)$. Luckily, the proofs guaranteeing the approximation ratio are *linear*. Therefore, by the end of the proof, we end up with an expression of the form: $F(S_{k+1}) \geq \beta(1-2\varepsilon-\gamma\varepsilon')F(S^*)$. Where β is the desired approximation ratio and γ depends on the constraints (e.g, $\beta = (1-1/e), \gamma = 2k$ for a cardinality constraint). That is, the multiplicative error costs us a $(1-2\varepsilon)$ factor, while the additive error is amplified by γ. We can achieve the desired result by setting $\varepsilon' = \varepsilon/\gamma$. We state the following theorem (the proofs are similar to those of [13, 4], and we defer them to the Appendix).

Theorem 2.2. *Algorithm 1 with an $(\varepsilon, \varepsilon')$-approximate incremental oracle has the following guarantees: (1) If $\varepsilon' \leq \varepsilon/2k$, it achieves a $(1-1/e-3\varepsilon)$-approximation under a cardinality constraint k. (2) If $\varepsilon' \leq \varepsilon/2kp$, it achieves a $(\frac{1-3\varepsilon}{1+p})$-approximation under a p-system constraint.*

Mini-batch sampling We present our mini-batch algorithm in Algorithm 2. Here, the approximate incremental oracle is constructed via mini-batch sampling with a batch size parameter α. In Section 3 we analyze the relation between α and the type of approximate incremental oracle guaranteed by our sampling procedure. We state the main theorem for Section 3 below (recall that e^* is the singleton maximizing F and S^* is an optimal solution for F).

Theorem 2.3. *When applied to maximize a non-negative monotone submodular function with batch size $\alpha = \Theta\big(\varepsilon^{-2}\,\phi^{-1}\gamma\log n\big)$, the mini-batch greedy algorithm (Algorithm 2) guarantees that w.h.p for every $j \in [k]$, $\hat{F}^j_{S_j}$ is an $(\varepsilon, \varepsilon F(e^*)/\gamma F(S^*))$-approximate incremental oracle, for any $\gamma > 0$. Furthermore, the algorithm performs $O(\alpha k n)$ oracle queries in expectation and w.h.p.*

Algorithm 2: Mini-batch greedy

α is a batch size parameter (where $\alpha \leq N$), k is an upper bound on the size of the solution
$S_1 \leftarrow \emptyset$
for $j = 1$ *to* k **do**
  Let $A_j \subseteq E \setminus S_j$ `// Problem specific constraint`
  if $A_j = \emptyset$ **then** return S_j
  $w \leftarrow \vec{0}$
  $\forall i, w_i \leftarrow N/\alpha$ with probability α/N
  $\hat{F}^j = \sum_{i=1}^{N} w_i f^i$
  $e_j = \arg\max_{e \in A_j} \hat{F}^j_{S_j}(e)$
  $S_{j+1} = S_j + e_j$
end
return S_{k+1}

Finally, we state our main result for this section.

Theorem 2.4. *When applied to maximize a non-negative monotone submodular function the mini-batch greedy algorithm (Algorithm 2) achieves w.h.p a $(1-1/e-\varepsilon)$-approximation under a*

[3] Strictly speaking, both [13, 4] define the approximate incremental oracle to be a function that returns e_j at iteration j of the greedy algorithm such that $\forall e \in A_j, F_{S_j}(e_j) \geq (1-\varepsilon)F_{S_j}(e)$. Our definition guarantees this property while allowing easy analysis of the mini-batch algorithm.

cardinality constraint and a $(\frac{1-\varepsilon}{1+p})$*-approximation under a* p*-system constraint using* $O(\frac{k^2 n \log n}{\varepsilon^2 \phi})$ *oracle evaluations in expectation and w.h.p.*

Proof. Combining Theorem 2.3 (setting $\gamma = 2k$) with Theorem 2.2, and rescaling ε by a constant factor, we get our result for cardinality constraints. For p-systems, we break the proof into cases. If $F(e^*) \geq F(S^*)/p$, Theorem 2.3 with $\gamma \geq 1$ guarantees that $\hat{F}^1_{S_1}(e_1) \geq \hat{F}^1_{S_1}(e^*) \geq (1-\varepsilon)F(e^*) - \varepsilon F(e^*)/\gamma \geq (1-2\varepsilon)F(e^*)$. Applying the upper bound of the oracle to e_1 we get $F(e_1) \geq (\hat{F}^1_{S_1}(e_1) - \varepsilon F(e^*)/\gamma)/(1+\varepsilon) \geq (1-4\varepsilon)F(e^*)$. Hence e_1, the element chosen in the first iteration of the greedy algorithm, is a $(1-4\varepsilon)/p$ approximate solution for F, and the rest follows because F is monotone and by rescaling ε. If $F(e^*) < F(S^*)/p$, Theorem 2.3 guarantees an $(\varepsilon, \varepsilon/\gamma p)$-approximate oracle. Setting $\gamma = 2k$ and applying Theorem 2.2, again rescaling ε by a constant factor, completes the proof. □

2.1 Related work

Approximate oracles Apart from the results of [13, 4] some works use different notions of an approximate oracle. Several works consider an approximate oracle $\hat{F}$, such that $\forall S \subseteq E, \left|\hat{F}(S) - F(S)\right| < \varepsilon F(S)$ [7, 15, 35]. The main difference of these models from our work is that they do not assume the surrogate function, $\hat{F}$, to be submodular. This adds a significant complication to the analysis and degrades the performance guarantees. Closer to the additive component of our notion, Streeter & Golovin [42] analyze greedy selection with a per-step additive error, which is amplified by a factor of k in the final guarantee.

Noisy submodular maximization Our work is related to prior work on optimizing submodular objectives when the algorithm does not have exact access to function values, but instead must rely on noisy feedback [16, 40, 14]. For example, [40] studies i.i.d. noisy evaluations, while [14] considers access to a noisy value oracle for a fixed objective.

While one can treat our mini-batch estimator as a sampling-based proxy oracle and attempt to apply results from the above works, this leads to weaker guarantees and additional assumptions. For example, instantiating [40] yields an additive-approximation guarantee with query complexity depending on additional gap/noise parameters, and the results are specific to cardinality constraints (and do not directly extend to p-system constraints).

Mini-batch methods The closest result resembling our mini-batch approach is the *stochastic-greedy algorithm* [3, 24]. They improve the expected query complexity of the greedy algorithm under a cardinality constraint by only considering a small random subset of $E \setminus S_j$ at the j-th iteration. We note that their approach can be combined into our mini-batch algorithm, reducing our query complexity by a $\tilde{\Theta}(k)$ factor, resulting in an approximation guarantee in expectation instead of w.h.p.

3 Analysis of the mini-batch greedy algorithm

We start by stating the following useful Chernoff bound:

Theorem 3.1 (Chernoff bound [29])**.** *Let* $X_1, ..., X_N$ *be independent random variables in the range* $[0, a]$*. Let* $X = \sum_{i=1}^{N} X_i$*. Then for any* $\varepsilon \in [0, 1]$ *and* $\mu \geq \mathbb{E}[X]$ *it holds that* $\mathbb{P}(|X - \mathbb{E}[X]| \geq \varepsilon\mu) \leq 2\exp\left(-\varepsilon^2\mu/3a\right)$.

The following lemma provides concentration guarantees for $\hat{F}^j$ in Algorithm 2.

Lemma 3.2. *For every* $j \in [k]$ *and* $S \subseteq E$ *(*$\hat{F}^j$ *sampled after* S *is fixed) and for every* $e \in E$ *and* $\mu \geq F_S(e)$*, it holds that* $\mathbb{P}\left[|\hat{F}^j_S(e) - F_S(e)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2 \alpha \phi \mu}{3F(e^*)}\right)$.

Proof. Fix some $e \in E$. Due to the fact that $\forall i \in [N], \mathbb{E}[w_i] = 1$ we have $\mathbb{E}[\hat{F}^j_S(e)] = \mathbb{E}[\sum_{i\in[N]} w_i f^i_S(e)] = F_S(e)$. Applying a Chernoff bound (Theorem 3.1) we have

$\mathbb{P}\left[|\hat{F}_S^j(e) - F_S(e)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2\mu}{3a}\right)$ where $a = \max\{\frac{N}{\alpha} f_S^i(e)\}_{i\in[N]} \leq N/\alpha$, as submodularity gives $f_S^i(e) \leq f^i(e) \leq 1$. Let us upper bound a as a function of $F(e^*)$ where $e^* \in E$ is the singleton maximizing F. Rearranging the singleton density definition we get that $N = F(e^*)/\phi$. We conclude that $a \leq N/\alpha = F(e^*)/\phi\alpha$. Given the above upper bound for a we get

$$\mathbb{P}\left[|\hat{F}_S^j(e) - F_S(e)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi\mu}{3F(e^*)}\right). \qquad \square$$

We are now ready to prove the main theorem for this section.

Theorem 2.3. *When applied to maximize a non-negative monotone submodular function with batch size $\alpha = \Theta\big(\varepsilon^{-2}\,\phi^{-1}\gamma\log n\big)$, the mini-batch greedy algorithm (Algorithm 2) guarantees that w.h.p for every $j \in [k]$, $\hat{F}_{S_j}^j$ is an $(\varepsilon, \varepsilon F(e^*)/\gamma F(S^*))$-approximate incremental oracle, for any $\gamma > 0$. Furthermore, the algorithm performs $O(\alpha k n)$ oracle queries in expectation and w.h.p.*

Proof. The number of oracle evaluations is due to the fact that every w_i is taken to be non-zero with probability α/N, which results in an expected α non-zero entries per iteration. Due to the size of α we also get concentration via a Chernoff bound. The algorithm executes for k iterations and must evaluate $|A_j| \leq n$ elements per iteration.

Let us prove the approximation guarantees. We break the analysis into cases.

$F(e^*) \leq \gamma F_{S_j}(e)$: Setting $\mu = F_{S_j}(e)$ we get:

$$\mathbb{P}\left[|\hat{F}_{S_j}^j(e) - F_{S_j}(e)| \geq \varepsilon F_{S_j}(e)\right] \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi F_{S_j}(e)}{3F(e^*)}\right) \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi}{3\gamma}\right) \leq 1/n^3.$$

$F(e^*) > \gamma F_{S_j}(e)$: Here we can set $\mu = F(e^*)/\gamma \geq F_{S_j}(e)$ and get:

$$\mathbb{P}\left[|\hat{F}_{S_j}^j(e) - F_{S_j}(e)| \geq \varepsilon F(e^*)/\gamma\right] \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi F(e^*)}{3\gamma F(e^*)}\right) \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi}{3\gamma}\right) \leq 1/n^3.$$

In both cases, the last inequality is by setting an appropriate constant in $\alpha = \Theta(\varepsilon^{-2}\phi^{-1}\gamma\log n)$. In the first case the deviation is at most $\varepsilon F_{S_j}(e)$, and in the second it is at most $\varepsilon F(e^*)/\gamma$. Hence, in both cases the deviation is at most $\varepsilon F_{S_j}(e) + \frac{\varepsilon F(e^*)}{\gamma F(S^*)} F(S^*)$, i.e., $\hat{F}_{S_j}^j$ is an $(\varepsilon, \varepsilon F(e^*)/\gamma F(S^*))$-approximate incremental oracle. We take a union bound over all $e \in E$ and $j \in [k]$ (at most n^2 values), which concludes the proof. $\square$

4 Experiments

We perform experiments on the following datasets, where the goal for all datasets is to maximize $F(S)$ subject to $|S| \leq k$. For influence maximization, we also enforce a p-system constraint.

Uber pickups This dataset consists of Uber pickups in New York City in May 2014[4]. The set contains $652,434$ records, where each record consists of a longitude and latitude, representing a pickup location. Following [37] we aim to find k positions for idle drivers to wait from a subset of popular pickup locations. We formalize the problem as follows. We run Lloyd's algorithm on the dataset, X, and find 100 cluster centers. We set these cluster centers to be the ground set E. We define the goal function $F : 2^E \to \mathbb{R}^+$ as $F(S) = \sum_{v\in X} f_v(S)$ where $f_v(S) = \max_{e\in E} d(v,e) - \min_{e\in S} d(v,e)$, and $d(v,e)$ is the Manhattan distance between v and e.

Discogs [20] This dataset[5] provides audio record information structured as a bipartite graph $G = (L, R; E')$. The left nodes represent labels, and the right nodes represent styles. Each edge $(u,v) \in L \times R$ signifies the involvement of a label u in producing a release of a style v. The dataset comprises $|L| = 243,764$ labels, $|R| = 383$ styles, and $|E'| = 5,255,950$ edges. We aim to select k styles that cover the activity of the maximum number of labels. We construct a maximum coverage

[4] https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city

[5] http://konect.cc/networks/discogs_lstyle/

function $F : 2^R \to \mathbb{R}$, where $F(S) = \sum_{u \in L} f_u(S)$, and $f_u(S)$ equals 1 if u is adjacent to some element of S and 0 otherwise.

Exemplar-based clustering We consider the problem of selecting a representative subset of k images from a massive dataset. We present experiments for both the CIFAR100 and the FashionMNIST datasets. For each dataset, we consider a subset of $50,000$ images, denoted by X. We flatten every image into a one-dimensional vector, subtract from it the mean of all images, and normalize it to unit norm. We take the distance between two elements in X as $d(x, x') = \|x - x'\|^2$. Here, the ground set is simply the dataset, $E = X$. Similarly to the Uber pickup dataset we define the goal function $F : 2^E \to \mathbb{R}^+$ as $F(S) = \sum_{v \in X} f_v(S)$ where $f_v(S) = \max_{e \in E} d(v, e) - \min_{e \in S} d(v, e)$.

Influence Maximization We consider the problem of maximizing influence in a social network. We use the Facebook dataset from SNAP [22][6], a simple undirected graph of real user connections with $4{,}039$ nodes and $88{,}234$ edges. We study influence spread under the Independent Cascade Model [17]: each activated node has one chance to activate each neighbor with probability p (here $p = 0.01$). The goal is to select a seed set $S \subseteq V$ (V is the ground set) of size k to maximize the expected total number of activated nodes. Furthermore, we enforce an independence constraint - the nodes in S may not be adjacent. This corresponds to a p-system, where p is the maximum degree in the graph. Formally, let $F(S) = \sum_{v \in V} f_v(S)$ be the expected cascade size where $f_v(S)$ is the expectation that v is activated by S - we approximate this via 500 Monte Carlo runs.

Experimental setup We compare the sparsifier and mini-batch approaches on each of the above datasets. For both, we evaluate the weighted and uniform variants. We introduce a parameter $\beta \in (0, 1)$ which corresponds to the desired fraction of *data* (the elements that constitute F) we wish to sample. We set α for all algorithms such that the expected number of sampled elements is βN in the mini-batch / sparsifier. For example, setting $\beta = 0.01$ means that the sparsifier will sample 1% of the data and execute the greedy algorithm, while the mini-batch algorithm will sample 1% of the data for every iteration of the greedy algorithm.

In practice, the naive algorithm is usually augmented with the following heuristic called *lazy-greedy* [23]. The algorithm initially creates a max heap of E ordered according to a key $\rho(e)$, where initially $\rho(e) = F(e)$. During the j-th iteration, it repeatedly pops e from the top of the heap, updates its key $\rho(e) = F_{S_j}(e)$, and inserts it back into the heap. Due to submodularity, if e remains at the top of the heap after the update, we know it maximizes $F_{S_j}(e)$. While this does not change the asymptotic number of oracle queries, it often leads to significant improvements in practice.[7]

We also consider *stochastic-greedy* [3, 24], which further reduces the number of oracle evaluations by greedily choosing the next element added to S_j from a subset of $E \setminus S_j$ of size $\frac{n}{k} \log \frac{1}{\varepsilon}$ sampled uniformly at random. We apply both of the above to the sparsifier / mini-batch approach and compare them against lazy-greedy and stochastic-greedy.

In Figure 1 we plot results for different β values for both the sparsifier and the mini-batch approach for different values of k. Every point on the graph is the average of 20 executions with the same k, β, and error bars denote the standard error of the mean. For every dataset, we present one plot per value of β showing the relative utility (value of F) of the mini-batch / sparsifier approach compared to lazy-greedy, followed by a single plot showing the relative number of oracle evaluations for all values of β, on a logarithmic scale. The cost plots also show error bars, but these are too small to be visible. We use "u" / "w" prefixes for uniform / weighted sampling. In Figures 2,3 (Appendix G) we compare the sparsifier and mini-batch algorithms (augmented with the sampling scheme of stochastic-greedy) to stochastic-greedy.

Results The first clear observation is that weighted sampling is impractical due to its $O(Nn)$ cost bottleneck. In the cost plots, the curves for all β values collapse into a single line. Surprisingly, the relative performance of uniform sampling is overall better than weighted sampling. Finally, we observe that the mini-batch algorithm is superior to the sparsifier approach for small values of β while using about the same number of queries. The largest performance gap occurs on the Facebook

[6] `https://snap.stanford.edu/data/ego-Facebook.html`

[7] Note that for the mini-batch algorithm the surrogate objective changes between iterations, and therefore heap keys computed in earlier iterations are no longer valid upper bounds on the current marginal gains. As a heuristic we still use the heap as is. Nevertheless, the mini-batch algorithm still outperforms the sparsifier, as we show in our experiments.

dataset. This is partly due to the independence constraint, which complicates cost normalization across methods. Still, we observe that the uniform mini-batch method outperforms the uniform sparsifier for $\beta = 10^{-1}$, where the number of queries is about the same, and even for $\beta = 10^{-3}$, where it attains a considerably better solution with a comparable number of queries.

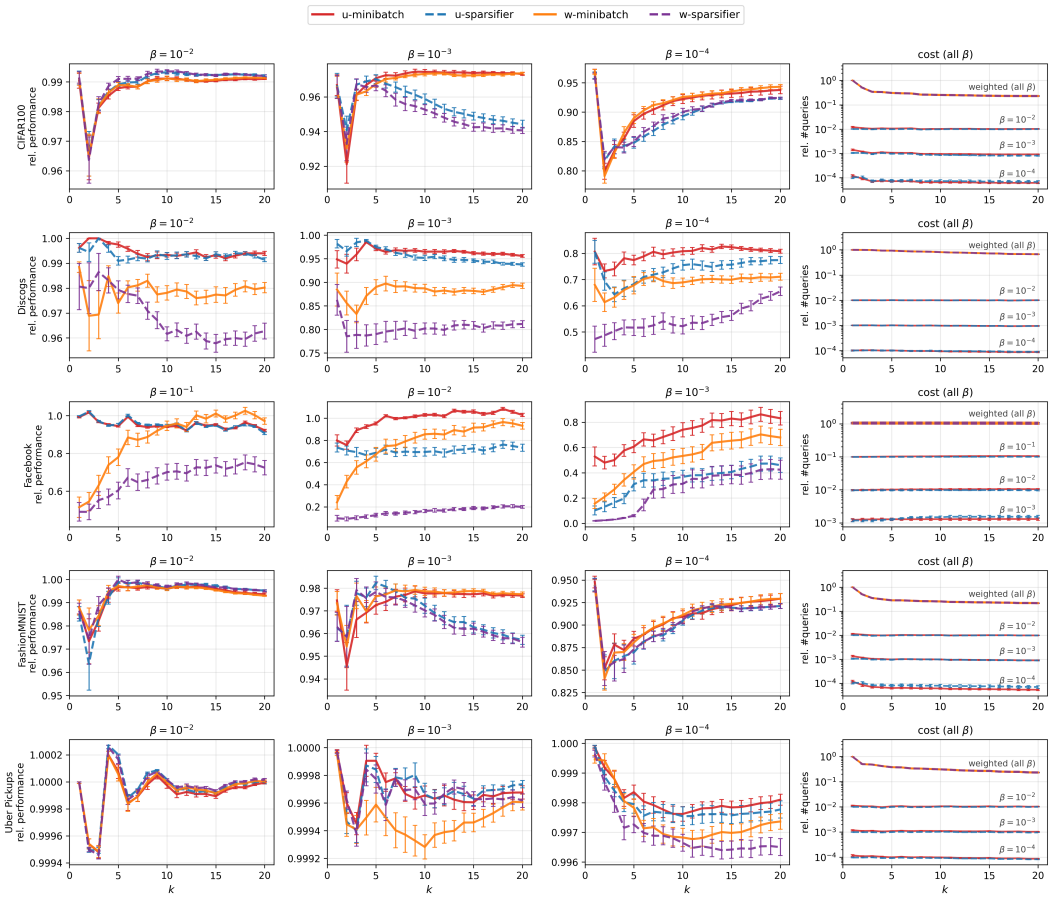

Figure 1: Sparsifier and mini-batch compared using lazy-greedy. Every row is a dataset; the first three plots show the relative utility for a single value of β (indicated above each plot), and the last plot shows the relative number of oracle queries for all values of β. In the cost plot the three lower bands are the uniform algorithms, which lie at relative cost $\approx \beta$, while the weighted algorithms collapse onto a single band for all β, as their cost is dominated by the $O(Nn)$ preprocessing. Error bars denote the standard error of the mean over 20 executions; in the cost plots they are present but too small to be visible.

THE USE OF LARGE LANGUAGE MODELS

In this work, we used a large language model for light copy-editing of the manuscript (grammar, phrasing and formatting). We have not used generative AI tools for research ideation, for developing the theoretical model or proofs, or for producing or analyzing the experimental results. The authors are responsible for all of the content of this paper.

REPRODUCIBILITY STATEMENT

The supplementary material contains our code, along with the scripts needed to reproduce all of the experiments and figures.

REFERENCES

[1] Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. In *STOC*, pp. 429–437. ACM, 2017.

[2] Wenruo Bai, Rishabh K. Iyer, Kai Wei, and Jeff A. Bilmes. Algorithms for optimizing the ratio of submodular functions. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2751–2759. JMLR.org, 2016.

[3] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *SODA*, pp. 1149–1168. SIAM, 2015.

[4] 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.

[5] Anamay Chaturvedi, Huy Le Nguyen, and Lydia Zakynthinou. Differentially private decomposable submodular maximization. In *AAAI*, pp. 6984–6992. AAAI Press, 2021.

[6] Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discret. Appl. Math.*, 7(3):251–274, 1984.

[7] Victoria G. Crawford, Alan Kuhnle, and My T. Thai. Submodular cost submodular cover with an approximate oracle. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1426–1435. PMLR, 2019.

[8] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, pp. 1064–1073. ACM Press, 2006.

[9] Delbert Dueck and Brendan J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *ICCV*, pp. 1–8. IEEE Computer Society, 2007.

[10] Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. *ACM Trans. Algorithms*, 13(2):25:1–25:12, 2017.

[11] Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009.

[12] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *FOCS*, pp. 667–676. IEEE Computer Society, 2006.

[13] Pranava R Goundan and Andreas S Schulz. Revisiting the greedy approach to submodular set function maximization. *Optimization online*, (1984):1–25, 2007.

[14] Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. In *COLT*, volume 65 of *Proceedings of Machine Learning Research*, pp. 1069–1122. PMLR, 2017.

[15] Thibaut Horel and Yaron Singer. Maximization of approximately submodular functions. In *NIPS*, pp. 3045–3053, 2016.

[16] Mohammad Reza Karimi, Mario Lucic, S. Hamed Hassani, and Andreas Krause. Stochastic submodular maximization: The case of coverage functions. In *NIPS*, pp. 6853–6863, 2017.

[17] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory Comput.*, 11:105–147, 2015.

[18] Yotam Kenneth and Robert Krauthgamer. Cut sparsification and succinct representation of submodular hypergraphs. *CoRR*, abs/2307.09110, 2023.

[19] Jannik Kudla and Stanislav Zivný. Sparsification of monotone k-submodular functions of low curvature. *CoRR*, abs/2302.03143, 2023.

[20] Jérôme Kunegis. KONECT: the koblenz network collection. In *WWW (Companion Volume)*, pp. 1343–1350. International World Wide Web Conferences Steering Committee / ACM, 2013.

[21] Hui Lin and Jeff A. Bilmes. A class of submodular functions for document summarization. In *ACL*, pp. 510–520. The Association for Computer Linguistics, 2011.

[22] Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *NIPS*, pp. 548–556, 2012.

[23] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*, pp. 234–243. Springer, 2005.

[24] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pp. 1812–1818. AAAI Press, 2015.

[25] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1358–1367. JMLR.org, 2016.

[26] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *J. Mach. Learn. Res.*, 17:238:1–238:44, 2016.

[27] Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In *NIPS*, pp. 3594–3602, 2016.

[28] Marko Mitrovic, Mark Bun, Andreas Krause, and Amin Karbasi. Differentially private submodular maximization: Data summarization in disguise. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2478–2487. PMLR, 2017.

[29] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[30] 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.

[31] Christos H. Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *FOCS*, pp. 250–259. IEEE Computer Society, 2008.

[32] Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. A coverage-based approach to recommendation diversity on similarity graph. In *RecSys*, pp. 15–22. ACM, 2016.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Sriram V. Pemmaraju. Equitable coloring extends chernoff-hoeffding bounds. In *RANDOM-APPROX*, volume 2129 of *Lecture Notes in Computer Science*, pp. 285–296. Springer, 2001.

[35] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Subset selection under noise. In *NIPS*, pp. 3560–3570, 2017.

[36] Akbar Rafiey and Yuichi Yoshida. Fast and private submodular and k-submodular functions maximization with matroid constraints. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7887–7897. PMLR, 2020.

[37] Akbar Rafiey and Yuichi Yoshida. Sparsification of decomposable submodular functions. In *AAAI*, pp. 10336–10344. AAAI Press, 2022.

[38] Aviad Rubinstein and Junyao Zhao. Budget-smoothed analysis for submodular maximization. In *ITCS*, volume 215 of *LIPIcs*, pp. 113:1–113:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[39] D. Sculley. Web-scale k-means clustering. In *WWW*, pp. 1177–1178. ACM, 2010.

[40] Adish Singla, Sebastian Tschiatschek, and Andreas Krause. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. *CoRR*, abs/1511.07211, 2015.

[41] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[42] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, 2008.

[43] Sebastian Tschiatschek, Rishabh K. Iyer, Haochen Wei, and Jeff A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *NIPS*, pp. 1413–1421, 2014.

[44] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pp. 67–74. ACM, 2008.

A SPARSIFIER BACKGROUND AND RELATED WORK

Rafiey & Yoshida [37] were the first to show how to construct a *sparsifier* for F. That is, given a parameter $\varepsilon > 0$ they show how to find a vector $w \in \mathbb{R}^N$ such that the number of non-zero elements in w is small in expectation and the function $\hat{F} = \sum_{i=1}^{N} w_i f^i$ satisfies with high probability (w.h.p)[8] that $\forall S \subseteq E, (1-\varepsilon)F(S) \leq \hat{F}(S) \leq (1+\varepsilon)F(S)$.

Specifically, every f^i is sampled with probability α_i proportional to $p_i = \max_{S \subseteq E, F(S) \neq 0} \frac{f^i(S)}{F(S)}$. If it is sampled, it is included in the sparsifier with weight $1/\alpha_i$, which implies that $\mathbb{E}[w_i] = 1$. While calculating the p_i's exactly requires exponential time, Rafiey & Yoshida [37] make do with an approximation, which can be calculated using interior point methods [2].

Rafiey & Yoshida [37] show that if all f^i's are non-negative and monotone[9], the above sparsifier can be constructed by an algorithm that requires $poly(N)$ oracle evaluations and the sparsifier will have expected size $O(\varepsilon^{-2} B n^{2.5} \log n)$, where $B = \max_{i \in [N]} B_i$ and B_i is the number of extreme points in the base polyhedron of f^i. They extend their results to matroid constraints of rank r and show that a sparsifier with expected size $O(\varepsilon^{-2} B r n^{1.5} \log n)$ can be constructed.

For the specific case of a cardinality constraint k, this implies a sparsifier of expected size $O(\varepsilon^{-2} B k n^{1.5} \log n)$ can be constructed using $poly(N)$ oracle evaluations. The sparsifier construction is treated as a *preprocessing step*, and therefore the actual execution of `Greedy` on the sparsifier requires only $O(\varepsilon^{-2} B k^2 n^{2.5} \log n)$ oracle evaluations to get a $(1-1/e-\varepsilon)$ approximation. This is an improvement over `Greedy` when $N \gg n, B$.

Recently, Kudla & Zivný [19] showed improved results for the case of *bounded curvature*. The *curvature* of a submodular function F is defined as $c = 1 - \min_{S \subseteq E, e \in E \setminus S} \frac{F_S(e)}{F(e)}$. We say that F has *bounded-curvature* if $c < 1$. Submodular functions with bounded curvature [6] offer a balance

[8] Probability at least $1 - 1/n^c$ for an arbitrary constant $c > 1$. The value of c does not affect the asymptotics of the results we state (including our own).

[9] Rafiey & Yoshida [37] also present results for non-monotone functions, however, Kudla & Zivný [19] point out an error in their calculation and note that the results only hold when all f^i's are monotone.

between modularity and submodularity, capturing the essence of diminishing returns without being too extreme.

They show that when the curvature of all f^i's and of F is constant, it is possible to reduce the preprocessing time to $O(Nn)$ oracle queries and to reduce the size of the sparsifier by a factor of $\sqrt{n}$. Furthermore, their results extend to the much more general case of k-*submodular functions*. While this significantly improves over the number of oracle calls compared to [37], the runtime of the preprocessing step depends on $\log\left(\max_{i\in[N]} \frac{\max_{e\in E} f^i(e)}{\min_{e\in E, f^i(e)>0} f^i(e)}\right)$.

Note that the B factor in [19, 37] can be exponential in n in the worst case.

The current state of the art is due to Kenneth & Krauthgamer [18] where they show that by sampling according to $p_i = \max_{e\in E, F(e)\neq 0} \frac{f^i(e)}{F(e)}$ it is possible to get both a fast sparsifier construction time of $O(Nn)$ and a small sparsifier size of $O(\frac{n^3}{\varepsilon^2})$. Their analysis also implies that if the solution size is bounded by k (e.g., a cardinality constraint), a sparsifier of size $O(\frac{nk^2}{\varepsilon^2})$ is sufficient. They also present results for general submodular functions, however, we only focus on their results for monotone functions, which are relevant for this paper.

B SINGLETON DENSITY THROUGH THE LENS OF SMOOTHED ANALYSIS

Smoothed analysis was introduced by [41] in an attempt to explain the fast runtime of the Simplex algorithm in practice, despite its exponential worst-case runtime. In the smoothed analysis framework, an algorithm is provided with an adversarial input that is perturbed by some random noise. Often, the crux of smoothed analysis is defining a realistic model of noise. To the best of our knowledge, [38] is the only result that considers smoothed analysis in the context of submodular maximization. They consider submodular maximization under a cardinality constraint, where the cardinality parameter k undergoes a perturbation according to some known distribution.

Bounded singleton density can be justified via a very natural smoothing model. As before, we assume, without loss of generality, that $\forall i \in [N], e \in E, f^i(e) \in [0, 1]$ (we can always achieve this by normalization). We treat $\phi \in [0, 1]$ as a *smoothing* parameter and let $d \in [N]$ be a *dependency* parameter.

Smoothing model. The function, $F = \sum_{i=1}^{N} f^i$, we aim to optimize is created as follows. An adversary chooses distributions $D_1, \ldots, D_N$ over monotone submodular functions $f : 2^E \to \mathbb{R}$. Each f^i is sampled *once* from D_i and then fixed. All probability/expectation statements below are with respect to this one-time sampling; the algorithm then runs on the realized instance (i.e., F is fixed during execution).

We assume $N = \Omega\Big(\frac{d}{\phi}\log(nd)\Big)$, and that there **exists** an element $e^* \in E$ such that:

1. For every $i \in [N]$, the random variable $f^i(e^*)$ (induced by the above sampling of f^i) satisfies $\mathbb{E}[f^i(e^*)] \geq \phi$.
2. In the collection $\{f^i(e^*)\}_{i\in[N]}$, each random variable depends on at most d others (equivalently, the dependency graph has maximum degree at most d).

For every $e \neq e^*$, the values $\{f^i(e)\}_{i\in[N]}$ may be chosen *adversarially*.

Intuition Going back to our lunch menu example, Condition (1) means that there *exists* a food on the menu that is not universally hated by the students. Condition (2) means that the preference of a student for that specific food is sufficiently independent of the preferences of other students. Assuming that $N = \Omega(\frac{d}{\phi}\log(nd))$ means that we have a sufficiently large student body – note that increasing N should not change ϕ and d.

Comparison to other models of smoothing Perhaps the most general smoothing approach when dealing with weighted inputs (e.g., in $[0, 1]$) is to assume that the weights are taken from some distribution whose density is upper bounded by a smoothing parameter ϕ [10, 1]. This generalizes the approach of [41], where Gaussian noise was added to the weights. Our approach is even more

general, as the above immediately implies that the expectation is lower bounded by ϕ. Furthermore, we only assume bounded dependency and only *partially* smooth the input.

Singleton density under the smoothing model We show that under the smoothing assumptions it holds that the singleton density of F is at least $\phi/2$ w.h.p. First we introduce the following Chernoff bound.

Theorem B.1 (Bounded dependency Chernoff bound [34])**.** *Let $X_1, ..., X_N$ be identically distributed random variables in the range $[0, 1]$ with bounded dependency d. Let $X = \sum_{i=1}^{N} X_i$. Then for any $\varepsilon \in [0, 1]$ and $\mu = \mathbb{E}[X]$ it holds that $\mathbb{P}(|X - \mathbb{E}[X]| \geq \varepsilon\mu) \leq \Theta(d) \cdot \exp\left(-\Theta(\varepsilon^2\mu/d)\right)$.*

We lower bound $F(e^*)$ using the smoothing assumptions.

Lemma B.2. *Under the smoothing assumptions it holds w.h.p that $F(e^*) \geq N\phi/2$.*

Proof. According to the smoothing assumptions, it holds that $\mathbb{E}[F(e^*)] = \sum_{i\in[N]} \mathbb{E}[f^i(e^*)] \geq N\phi$. Applying the bounded dependency Chernoff bound with $\varepsilon = 1/2, \mu = \mathbb{E}[F(e^*)]$ we get that:

$$\begin{aligned}&\mathbb{P}\left[|F(e^*) - \mathbb{E}[F(e^*)]| \geq \mu/2\right] \\ &\leq \Theta(d)\exp(-\Theta(\mu/d)) \leq \Theta(d)\exp(-\Theta(N\phi/d)) \leq 1/n^2\end{aligned}$$

Where the last inequality is due to the fact that $N = \Omega((d/\phi)\log(nd))$. We conclude that w.h.p $F(e^*) \geq N\phi/2$. □

C p-SYSTEMS

p-systems The concept of p-systems offers a generalized framework for understanding independence families, parameterized by an integer p. We can define a p-system in the context of an independence family $\mathcal{I} \subseteq 2^E$ and $E' \subseteq E$. Let $\mathcal{B}(E')$ be the maximal independent sets within $\mathcal{I}$ that are also subsets of E'. Formally, $\mathcal{B}(E') = \{A \in \mathcal{I} | A \subseteq E' \text{ and no } A' \in \mathcal{I} \text{ exists s.t } A \subset A' \subseteq E'\}$. A distinguishing characteristic of a p-system is that for every $E' \subseteq E$, the ratio of the sizes of the largest to the smallest sets in $\mathcal{B}(E')$ does not exceed p: $\frac{\max_{A\in\mathcal{B}(E')}|A|}{\min_{A\in\mathcal{B}(E')}|A|} \leq p$.

The significance of p-systems lies in their ability to encapsulate a variety of combinatorial structures. For instance, when the intersection of p matroids can be described using p-systems. In graph theory, the collection of matchings in a standard graph can be viewed as a 2-system. Extending this to hypergraphs, where edges might have cardinalities up to p, the set of matchings therein can be viewed as a p-system.

The greedy algorithm for p-systems Formally, the optimization problem can be expressed as: $\max_{S\in\mathcal{I}} F(S)$ where the pair $(E, \mathcal{I})$ characterizes a p-system and $F : 2^E \to \mathbb{R}^+$ denotes a non-negative monotone submodular set function. It was shown by Nemhauser et al. [30] that the natural greedy approach achieves an optimal approximation ratio of $\frac{1}{p+1}$. Setting $A_j = \{e \mid S_j + e \in \mathcal{I}\}$ (i.e., S_j remains an independent set after adding e) in Algorithm 1 we get the greedy algorithm of Nemhauser et al. [30]. Note that for general p-systems it might be that $k = n$, however, there are very natural problems where $k \ll n$. For example, for maximum matching, E corresponds to all edges in the graph, which can be quadratic in the number of nodes, while the solution is at most linear in the number of nodes.

D MISSING PROOFS

Proof of Theorem 2.2 Theorem 2.2 directly follows from the two lemmas below.

Lemma D.1. *Let $\varepsilon' \leq \varepsilon/2k$. Algorithm 1 with an $(\varepsilon, \varepsilon')$-approximate incremental oracle achieves a $(1 - 1/e - 3\varepsilon)$-approximation under a cardinality constraint k.*

Proof. Let S^* be some optimal solution for F. We start by proving that the following holds for every $j \in [k]$:

$$F(S_{j+1}) - F(S_j) \geq \frac{1}{k}((1 - 3\varepsilon)F(S^*) - F(S_j))$$

Fix some $j \in [k]$ and let $S^* \setminus S_j = \{e_1^*, \ldots, e_\ell^*\}$ where $\ell \leq k$. Let $S_t^* = \{e_1^*, \ldots, e_t^*\}$, and $S_0^* = \emptyset$. Let us first use submodularity and monotonicity to upper bound $F(S^*)$.

$$\begin{aligned}
&F(S^*) \leq F(S^* + S_j) \\
&= F(S_j) + \sum_{t=1}^{\ell} [F(S_j + S_t^*) - F(S_j + S_{t-1}^*)] \\
&\leq F(S_j) + \sum_{t=1}^{\ell} F_{S_j}(e_t^*) \leq F(S_j) + \sum_{t=1}^{\ell} \max_{e \in E \setminus S_j} F_{S_j}(e) \\
&\leq F(S_j) + k \max_{e \in E \setminus S_j} F_{S_j}(e) \\
&\leq F(S_j) + \frac{k}{1-\varepsilon} (\max_{e \in E \setminus S_j} \hat{F}_{S_j}^j(e) + \varepsilon' F(S^*))
\end{aligned}$$

Where the last inequality is due to the fact that $\hat{F}_{S_j}^j$ is an $(\varepsilon, \varepsilon')$-approximate incremental oracle, which implies that $F_{S_j}(e) \leq (\hat{F}_{S_j}^j(e) + \varepsilon' F(S^*))/(1-\varepsilon)$ for every e.

Noting that $e_j = \arg\max_{e \in E \setminus S_j} \hat{F}_{S_j}^j(e)$ we get that:

$$\begin{aligned}
&F(S^*) \leq F(S_j) + \frac{k}{1-\varepsilon} (\hat{F}_{S_j}^j(e_j) + \varepsilon' F(S^*)) \\
&\implies \hat{F}_{S_j}^j(e_j) \geq \frac{1-\varepsilon}{k} (F(S^*) - F(S_j)) - \varepsilon' F(S^*)
\end{aligned}$$

The above lower bounds the progress on the j-th mini-batch. Now, let us bound the progress on F. Again, we use the fact that $\hat{F}_{S_j}^j$ is an $(\varepsilon, \varepsilon')$-approximate incremental oracle, which implies that $F_{S_j}(e_j) \geq (\hat{F}_{S_j}^j(e_j) - \varepsilon' F(S^*))/(1+\varepsilon)$.

$$\begin{aligned}
&F(S_{j+1}) - F(S_j) \geq \frac{\hat{F}_{S_j}^j(e_j) - \varepsilon' F(S^*)}{1+\varepsilon} \\
&\geq \frac{1}{1+\varepsilon} \left(\frac{1-\varepsilon}{k} (F(S^*) - F(S_j)) - 2\varepsilon' F(S^*) \right) \\
&\geq \frac{1-2\varepsilon}{k} (F(S^*) - F(S_j)) - 2\varepsilon' F(S^*) \\
&\geq \frac{1}{k} ((1 - 2\varepsilon - 2\varepsilon' k) F(S^*) - F(S_j))
\end{aligned}$$

Where the third transition uses $\frac{1-\varepsilon}{1+\varepsilon} \geq 1 - 2\varepsilon$ and $\frac{1}{1+\varepsilon} \leq 1$, and the last uses $(1-2\varepsilon)F(S_j) \leq F(S_j)$.

Finally, using the fact that $\varepsilon' \leq \varepsilon/2k$ we get:

$$F(S_{j+1}) - F(S_j) \geq \frac{1}{k} ((1 - 3\varepsilon) F(S^*) - F(S_j))$$

Rearranging, the result directly follows using standard arguments.

$$\begin{aligned}
&F(S_{k+1}) \geq \frac{(1-3\varepsilon)}{k} F(S^*) + (1 - \frac{1}{k}) F(S_k) \\
&\geq \frac{(1-3\varepsilon)}{k} F(S^*) \left(\sum_{i=0}^{k-1} (1 - \frac{1}{k})^i \right) + F(\emptyset) \\
&= F(S^*) \frac{(1-3\varepsilon)(1 - (1 - \frac{1}{k})^k)}{k(1 - (1 - \frac{1}{k}))} = (1-3\varepsilon) \left(1 - (1 - \frac{1}{k})^k \right) F(S^*) \\
&\geq (1-3\varepsilon)(1 - 1/e) F(S^*) \geq (1 - 1/e - 3\varepsilon) F(S^*)
\end{aligned}$$

□

Lemma D.2. *Let $\varepsilon' \leq \varepsilon/2kp$. Algorithm 1 with an $(\varepsilon, \varepsilon')$-approximate incremental oracle achieves a $(\frac{1-3\varepsilon}{1+p})$-approximation under a p-system constraint.*

Proof. Let S^* be some optimal solution for F.

Assume without loss of generality that the solution returned by the algorithm consists of k elements $S_{k+1} = \{e_1, \ldots, e_k\}$.

We show the existence of a partition $S_1^*, S_2^*, \ldots, S_k^*$ of S^* such that $F_{S_j}(e_j) \geq \frac{1-2\varepsilon}{p} F_{S_{k+1}}(S_j^*) - 2\varepsilon' F(S^*)$. Note, we allow some of the sets in the partition to be empty.

Define $T_k = S^*$. For $j = k, k-1, ..., 2$ execute: Let $B_j = \{e \in T_j \mid S_j + e \in \mathcal{I}\}$. If $|B_j| \leq p$ set $S_j^* = B_j$; else pick an arbitrary $S_j^* \subset B_j$ with $\left|S_j^*\right| = p$. Then set $T_{j-1} = T_j \setminus S_j^*$ before decreasing j. After the loop set $S_1^* = T_1$. It is clear that for $j = 2, ..., k$, $\left|S_j^*\right| \leq p$.

We prove by induction over $j = 0, 1, ..., k-1$ that $|T_{k-j}| \leq (k-j)p$. For $j = 0$, when the greedy algorithm stops, S_{k+1} is a maximal independent set contained in E, therefore any independent set (including $T_k = S^*$) satisfies $|T_k| \leq p\,|S_{k+1}| = pk$. We proceed to the inductive step for $j > 0$. There are two cases: (1) $|B_{k-j+1}| > p$, which implies that $\left|S_{k-j+1}^*\right| = p$ and using the induction hypothesis we get that $|T_{k-j}| = |T_{k-j+1}| - \left|S_{k-j+1}^*\right| \leq (k-j+1)p - p = (k-j)p$. (2) $|B_{k-j+1}| \leq p$, it holds that $T_{k-j} = T_{k-j+1} \setminus B_{k-j+1}$. Let $Y = S_{k-j+1} + T_{k-j}$. Due to the definition of B_{k-j+1} it holds that S_{k-j+1} is a maximal independent set in Y. It holds that T_{k-j} is independent and contained in Y, therefore $|T_{k-j}| \leq p\,|S_{k-j+1}| = p(k-j)$.

Finally, we get that $|T_1| = |S_1^*| \leq p$. By construction it holds that $\forall j \in [k], \forall e \in S_j^*, S_j + e$ is independent. From the choice made by the greedy algorithm and the fact that $\hat{F}_{S_j}^j$ is an $(\varepsilon, \varepsilon')$-approximate incremental oracle, it follows that for each $e \in S_j^*$:

$$\begin{aligned} F_{S_j}(e_j) &\geq \frac{\hat{F}_{S_j}^j(e_j) - \varepsilon' F(S^*)}{1+\varepsilon} \\ &\geq \frac{\hat{F}_{S_j}^j(e) - \varepsilon' F(S^*)}{1+\varepsilon} \geq \frac{(1-\varepsilon)F_{S_j}(e) - 2\varepsilon' F(S^*)}{1+\varepsilon} \\ &\geq (1-2\varepsilon)F_{S_j}(e) - 2\varepsilon' F(S^*) \end{aligned}$$

Hence,

$$\begin{aligned} \left|S_j^*\right| F_{S_j}(e_j) &\geq \sum_{e \in S_j^*} ((1-2\varepsilon)F_{S_j}(e) - 2\varepsilon' F(S^*)) \\ &\geq (1-2\varepsilon)F_{S_j}(S_j^*) - 2\varepsilon' \left|S_j^*\right| F(S^*) \geq (1-2\varepsilon)F_{S_{k+1}}(S_j^*) - 2\varepsilon' \left|S_j^*\right| F(S^*) \end{aligned}$$

Using submodularity in the last two inequalities.

For all $j \in \{1, 2, ..., k\}$ it holds that $\left|S_j^*\right| \leq p$, and thus $F_{S_j}(e_j) \geq \frac{1-2\varepsilon}{p} F_{S_{k+1}}(S_j^*) - 2\varepsilon' F(S^*)$. Using the partition, we get that:

$$\begin{aligned} F(S_{k+1}) &\geq \sum_{j=1}^{k} F_{S_j}(e_j) \geq \sum_{j=1}^{k} (\frac{1-2\varepsilon}{p} F_{S_{k+1}}(S_j^*) - 2\varepsilon' F(S^*)) \\ &\geq \frac{1-2\varepsilon}{p} F_{S_{k+1}}(S^*) - 2\varepsilon' k F(S^*) \\ &\geq \frac{1-2\varepsilon}{p} (F(S^*) - F(S_{k+1})) - 2\varepsilon' k F(S^*) \end{aligned}$$

Where the second to last inequality is due to submodularity, and the last is due to monotonicity. Rearranging, we get that:

$$F(S_{k+1}) \geq \frac{(1-2\varepsilon-2p\varepsilon' k)}{p+1-2\varepsilon} F(S^*) \geq \frac{(1-2\varepsilon-2p\varepsilon' k)}{p+1} F(S^*)$$

As $\varepsilon' \leq \frac{\varepsilon}{2pk}$ we get the desired result. □

E THEORETICAL GUARANTEES FOR THE UNIFORM SPARSIFIER

Let us start with a definition.

Definition E.1. We say that $\hat{F}$ is an $(\varepsilon, \varepsilon')$-*additive sparsifier* for F if $\forall S \subseteq E, \max\{(1-\varepsilon)F(S) - \varepsilon' F(S^*), 0\} \leq \hat{F}(S) \leq (1+\varepsilon)F(S) + \varepsilon' F(S^*)$, where S^* is an optimal solution for F.

We state the following theorem.

Theorem E.2. *Let $\hat{F}$ be an $(\varepsilon, \varepsilon')$-*additive sparsifier *for F, and $\hat{S}$ be a ψ-approximate solution for $\hat{F}$, for $\psi \in (0, 1]$. Then $\hat{S}$ is a $((1-2\varepsilon)\psi - 2\varepsilon')$-approximate solution for F.*

Proof. Let $\hat{S}^*, S^*$ be optimal solutions for $\hat{F}, F$ respectively, and let $\hat{S}$ be a ψ-approximate solution for $\hat{F}$. We may assume that $(1-2\varepsilon)\psi - 2\varepsilon' \geq 0$, as otherwise the claim holds trivially due to the non-negativity of F; this implies that $\psi(1-\varepsilon-\varepsilon') - \varepsilon' \geq 0$. It holds that

$$\begin{aligned} F(\hat{S}) &\geq \frac{\hat{F}(\hat{S}) - \varepsilon' F(S^*)}{1+\varepsilon} \geq \frac{\psi \hat{F}(\hat{S}^*) - \varepsilon' F(S^*)}{1+\varepsilon} \\ &\geq \frac{\psi \hat{F}(S^*) - \varepsilon' F(S^*)}{1+\varepsilon} \geq \frac{\psi((1-\varepsilon)F(S^*) - \varepsilon' F(S^*)) - \varepsilon' F(S^*)}{1+\varepsilon} \\ &\geq (1-\varepsilon)\left(\psi(1-\varepsilon-\varepsilon') - \varepsilon'\right) F(S^*) \geq ((1-2\varepsilon)\psi - 2\varepsilon') F(S^*) \end{aligned}$$

Where the first and fourth transitions use the definition of the sparsifier, the second uses the fact that $\hat{S}$ is an approximate solution for $\hat{F}$, the third uses the fact that $\hat{S}^*$ is an optimal solution for $\hat{F}$, and the fifth uses $\frac{1}{1+\varepsilon} \geq 1-\varepsilon$ together with the non-negativity of $\psi(1-\varepsilon-\varepsilon') - \varepsilon'$. □

An analogous proof also works for the case where we aim to minimize F. Furthermore, the above holds independently of any constraints.

Next we show that we can construct an $(\varepsilon, \varepsilon')$-additive sparsifier using uniform sampling. Specifically, for a parameter $\alpha \leq N$ we set $w_i = N/\alpha$ with probability α/N and 0 otherwise. The sparsifier is $\hat{F} = \sum_{i=1}^{N} w_i f^i$. We prove an analogue to Lemma 3.2.

Lemma E.3. *For every $S \subseteq E$ such that $|S| \leq k$ ($\hat{F}$ sampled after S is fixed) and $\mu \geq F(S)$, it holds that $\mathbb{P}\left[|\hat{F}(S) - F(S)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2 \alpha \phi \mu}{3kF(e^*)}\right)$.*

Proof. Fix some $S \subseteq E$. Due to the fact that $\forall i \in [N], \mathbb{E}[w_i] = 1$ we have $\mathbb{E}[\hat{F}(S)] = \mathbb{E}[\sum_{i\in[N]} w_i f^i(S)] = F(S)$. Applying a Chernoff bound (Theorem 3.1) we have

$$\mathbb{P}\left[|\hat{F}(S) - F(S)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2\mu}{3a}\right)$$

where $a = \max\{\frac{N}{\alpha} f^i(S)\}_{i\in[N]} \leq kN/\alpha$, as submodularity gives $f^i(S) \leq \sum_{e\in S} f^i(e) \leq |S| \leq k$.

Let us upper bound a as a function of $F(e^*)$ where $e^* \in E$ is the singleton maximizing F. Rearranging the singleton density definition we get that $N = F(e^*)/\phi$. We conclude that $a \leq kN/\alpha = kF(e^*)/\phi\alpha$. Given the above upper bound for a we get:

$$\begin{aligned} &\mathbb{P}\left[|\hat{F}(S) - F(S)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2\mu}{3a}\right) \\ &\leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi\mu}{3kF(e^*)}\right) \end{aligned}$$

□

We are now ready to prove the main theorem for this section.

Theorem E.4. *For any $\gamma > 0$ and $\alpha = \Theta(k^2\gamma\varepsilon^{-2}\phi^{-1}\log n)$, it holds that $\hat{F}$ is an $(\varepsilon, \varepsilon F(e^*)/\gamma F(S^*))$-additive sparsifier for F of size $\Theta(k^2\gamma\varepsilon^{-2}\phi^{-1}\log n)$ in expectation and w.h.p, where $k \leq n$ is an upper bound on any solution for F.*

Proof. Let us prove the approximation guarantees. We break the analysis into cases.

$F(e^*) \leq \gamma F(S)$: Setting $\mu = F(S)$ we get:

$$\mathbb{P}\left[|\hat{F}(S) - F(S)| \geq \varepsilon F(S)\right] \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi F(S)}{3kF(e^*)}\right)$$
$$\leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi}{3k\gamma}\right) \leq 1/n^{k+3}$$

$F(e^*) > \gamma F(S)$: Here we can set $\mu = F(e^*)/\gamma \geq F(S)$ and get:

$$\mathbb{P}\left[|\hat{F}(S) - F(S)| \geq \varepsilon F(e^*)/\gamma\right] \leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi F(e^*)}{3k\gamma F(e^*)}\right)$$
$$\leq 2\exp\left(-\frac{\varepsilon^2\alpha\phi}{3k\gamma}\right) \leq 1/n^{k+3}$$

Where in both cases, the last inequality is by setting an appropriate constant in $\alpha = \Theta(k^2\gamma\varepsilon^{-2}\phi^{-1}\log n)$. In the first case the deviation is at most $\varepsilon F(S)$, and in the second it is at most $\varepsilon F(e^*)/\gamma$. Hence, in both cases the deviation is at most $\varepsilon F(S) + \frac{\varepsilon F(e^*)}{\gamma F(S^*)}F(S^*)$, i.e., $\hat{F}$ is an $(\varepsilon, \varepsilon F(e^*)/\gamma F(S^*))$-additive sparsifier. We take a union bound over all $S \subseteq E$ such that $|S| \leq k$, of which there are at most $\sum_{i=0}^{k}\binom{n}{i} \leq n^{k+1}$, which concludes the proof.

□

While the above yields improved approximation guarantees for arbitrary constraints and arbitrary algorithms, in this paper we focus on the greedy algorithm under cardinality and p-system constraints. Let us state our results for these two constraints by combining Theorem E.2 and Theorem E.4. The result for a p-system constraint follows via a similar case-based analysis as Theorem 2.4.

Theorem E.5. *When applied to maximize a non-negative monotone submodular function, the greedy algorithm executed on $\hat{F}$ constructed with parameter $\alpha = \Theta(k^2\varepsilon^{-2}\phi^{-1}\log n)$ achieves w.h.p a $(1 - 1/e - \varepsilon)$-approximation under a cardinality constraint and a $(\frac{1-\varepsilon}{1+p})$-approximation under a p-system constraint using $O(\frac{k^3 n\log n}{\varepsilon^2\phi})$ oracle evaluations in expectation and w.h.p.*

F WEIGHTED MINI-BATCH GREEDY ALGORITHM

In this section, we define the weighted mini-batch algorithm and prove *worst-case* guarantees, which are superior to those of the sparsifier-based approach.

The weighted mini-batch algorithm is identical to the uniform version, except that we sample according to the following probabilities $\forall i \in [N], p_i = \max_{e\in E, F(e)\neq 0}\frac{f^i(e)}{F(e)}$, which are the same as in the sparsifier algorithm of [18]. That is, instead of sampling a single $\hat{F}$ at the beginning, we sample a new $\hat{F}^j$ (*mini-batch*), for every step of the algorithm.

Similar to the uniform version, we show that when $\hat{F}^j_{S_j}$ is sampled using mini-batch sampling, we indeed get, w.h.p, an (additive) approximate incremental oracle for every step of the algorithm. We present our sampling procedure in Algorithm 3 and the complete mini-batch algorithm in Algorithm 4.

Algorithm 3: Sample($\alpha, \{p_i\}_{i=1}^N$)

$w \leftarrow \vec{0}$
for $i = 1$ *to* N **do**
  $\alpha_i \leftarrow \min\{1, \alpha p_i\}$
  $w_i \leftarrow 1/\alpha_i$ with probability α_i
end
return $\hat{F} = \sum_{i=1}^N w_i f^i$

Algorithm 4: Weighted mini-batch greedy

$\forall i \in [N], p_i \leftarrow \max_{e \in E, F(e) \neq 0} \frac{f^i(e)}{F(e)}$
α is a batch parameter
$S_1 \leftarrow \emptyset$
k is an upper bound on the size of the solution
for $j = 1$ *to* k **do**
  Let $A_j \subseteq E \setminus S_j$
  if $A_j = \emptyset$ **then** return S_j
  Let $\hat{F}^j \leftarrow Sample(\alpha, \{p_i\}_{i=1}^N)$
  $e_j = \arg\max_{e \in A_j} \hat{F}^j_{S_j}(e)$
  $S_{j+1} = S_j + e_j$
end
return S_{k+1}

We analyze the relation between the batch parameter, α, and the type of approximate incremental oracles guaranteed by our sampling procedure.

We start with the following lemma from [18], which bounds the expected size of $\hat{F}$. We present a proof for completeness.

Lemma F.1. *The expected size of $\hat{F}$ is $\alpha \sum_{i=1}^N p_i \leq \alpha n$.*

Proof. We start by showing that $\sum_{i=1}^N p_i \leq n$. Let us divide the range $[N]$ into

$$A_e = \left\{ i \in [N] \mid e = \arg\max_{e' \in E, F(e') \neq 0} \frac{f^i(e')}{F(e')} \right\}$$

If 2 elements in E achieve the maximum value for some i, we assign it to a single A_e arbitrarily.

$$\sum_{i=1}^N p_i = \sum_{i=1}^N \max_{e \in E} \frac{f^i(e)}{F(e)} =$$

$$\sum_{e \in E} \sum_{i \in A_e} \frac{f^i(e)}{F(e)} = \sum_{e \in E} \frac{\sum_{i \in A_e} f^i(e)}{F(e)} \leq \sum_{e \in E} 1 \leq n$$

Let X_i be an indicator variable for the event $w_i > 0$. We are interested in $\sum_{i=1}^N \mathbb{E}[X_i]$. It holds that:

$$\sum_{i=1}^N \mathbb{E}[X_i] = \sum_{i=1}^N \alpha_i \leq \sum_{i=1}^N \alpha p_i = \alpha \sum_{i=1}^N p_i \leq \alpha n$$

□

The following lemma provides concentration guarantees for $\hat{F}$ in Algorithm 3.

Lemma F.2. *For every $S \subseteq E$ ($\hat{F}$ sampled after S is fixed) and for every $e \in E$ and $\mu \geq F_S(e)$, it holds that $\mathbb{P}\left[|\hat{F}_S(e) - F_S(e)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2 \alpha \mu}{3F(e)}\right)$.*

Proof. Fix some $e \in E$. Let $G = \sum_{i \in I} f^i$, where $I = \{i \in [N] \mid \alpha_i = 1\}$. Let $F'_S(e) = F_S(e) - G_S(e)$ and $\hat{F}'_S(e) = \hat{F}_S(e) - G_S(e)$. Let $J = [N] \setminus I$. It holds that:

$$\begin{aligned}
&\mathbb{P}\left[|\hat{F}_S(e) - F_S(e)| \geq \varepsilon\mu\right] \\
&= \mathbb{P}\left[|\hat{F}'_S(e) + G_S(e) - F'_S(e) - G_S(e)| \geq \varepsilon\mu\right] \\
&= \mathbb{P}\left[|\hat{F}'_S(e) - F'_S(e)| \geq \varepsilon\mu\right]
\end{aligned}$$

Due to the fact that $\mathbb{E}[w_i] = 1$ we have $\mathbb{E}[\hat{F}'_S(e)] = \mathbb{E}[\sum_{i\in J} w_i f^i_S(e)] = F'_S(e)$. As f^i's are monotone, it holds that $\mu \geq F_S(e) \geq F'_S(e)$. Applying a Chernoff bound (Theorem 3.1) we have

$$\mathbb{P}\left[|\hat{F}'_S(e) - F'_S(e)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\varepsilon^2\mu/3a\right)$$

where $a = \max\{w_i f^i_S(e)\}_{i\in J}$. Recall that $w_i = 1/\alpha_i$ where $\alpha_i = \min\{1, \alpha p_i\}$ and $\alpha_i < 1$ for all $i \in J$. Let us upper bound a.

$$a = \max_{i\in J} w_i f^i_S(e) = \max_{i\in J} \frac{f^i_S(e)}{\alpha p_i} = \max_{i\in J} \frac{f^i_S(e)}{\alpha \cdot \max_{e'\in E} \frac{f^i(e')}{F(e')}}$$

$$\leq \max_{i\in J} \frac{f^i(e)}{\alpha \cdot \frac{f^i(e)}{F(e)}} = \frac{F(e)}{\alpha}$$

Where the inequality is due to submodularity and non-negativity in the numerator and maximality in the denominator. Note that the above is also correct if we only have some approximation to p_i – i.e., given $p'_i > p_i\lambda$ for $\lambda \in (0, 1)$, we can increase α by a $1/\lambda$ factor and the above still holds. Given the above upper bound for a we get:

$$\mathbb{P}\left[|\hat{F}_S(e) - F_S(e)| \geq \varepsilon\mu\right] \leq 2\exp\left(-\frac{\varepsilon^2\mu}{3a}\right)$$

$$\leq 2\exp\left(-\frac{\varepsilon^2\alpha\mu}{3F(e)}\right)$$

□

We state the main theorem for the section below.

Theorem F.3. *The weighted mini-batch greedy algorithm (Algorithm 4) maximizing a non-negative monotone submodular function has the following guarantees:*

1. *If F has curvature bounded by c, and $\alpha = \Theta(\frac{\log n}{\varepsilon^2(1-c)})$ it holds w.h.p that $\forall j \in [k]$ that $\hat{F}^j_{S_j}$ is an $(1-\varepsilon)$-approximate incremental oracle.*
2. *If $\alpha = \Theta(\varepsilon^{-2}\gamma \log n)$ it holds w.h.p $\forall j \in [k]$ that $\hat{F}^j_{S_j}$ is an $(\varepsilon, \varepsilon/\gamma)$-approximate incremental oracle, for any parameter $\gamma > 0$.*

Furthermore, the number of oracle evaluations during preprocessing is $O(nN)$ and an expected $\alpha(\sum_{i=1}^N p_i)(\sum_{j=1}^k |A_j|) = O(\alpha k n^2)$ during execution.

Combining Theorem F.3 (setting $\gamma = 2k$ for a cardinality constraint and $\gamma = 2kp$ for a p-system constraint) with Theorem 2.1 and Theorem 2.2, rescaling ε by a constant factor, we state our main result.

Theorem F.4. *The weighted mini-batch greedy algorithm maximizing a non-negative monotone submodular function requires $O(nN)$ oracle calls during preprocessing and has the following guarantees:*

1. *If F has curvature bounded by c, it achieves w.h.p a $(1 - 1/e - \varepsilon)$-approximation under a cardinality constraint and $(\frac{1-\varepsilon}{1+p})$-approximation under a p-system constraint with an expected $O(\frac{kn^2 \log n}{\varepsilon^2(1-c)})$ oracle evaluations for both cases.*
2. *It achieves w.h.p a $(1 - 1/e - \varepsilon)$-approximation under a cardinality constraint and $(\frac{1-\varepsilon}{1+p})$-approximation under a p-system constraint with an expected $O(k^2(n/\varepsilon)^2 \log n)$ and $O(k^2 p(n/\varepsilon)^2 \log n)$ oracle evaluations respectively.*

Proof. The number of oracle evaluations is due to Lemma F.1 and the fact that the algorithm executes for k iterations and must evaluate $|A_j| \leq n$ elements per iteration.

Let us prove the approximation guarantees. Let us start with the bounded curvature case. Fix some S_j. As $\hat{F^j}$ is sampled after S_j is fixed, we can fix some $e \in E$ and apply Lemma F.2 with $\mu = F_{S_j}(e)$. We get that:

$$\mathbb{P}\left[|\hat{F}^j_{S_j}(e) - F_{S_j}(e)| \geq \varepsilon F_{S_j}(e)\right] \leq 2\exp\left(-\frac{\varepsilon^2 \alpha F_{S_j}(e)}{3F(e)}\right)$$
$$\leq 2\exp\left(-\frac{\varepsilon^2 \alpha(1-c)}{3}\right) \leq 1/n^3$$

Where the second inequality is because $F_{S_j}(e)/F(e) \geq \min_{S \subseteq E, e' \in E \setminus S} F_S(e')/F(e') = 1 - c$, and the last transition is by setting an appropriate constant in $\alpha = \Theta(\frac{\log(n)}{\varepsilon^2(1-c)})$. When the curvature is not bounded, we break the analysis into cases.

$F(e) \leq \gamma F_{S_j}(e)$**:** Setting $\mu = F_{S_j}(e)$ we get:

$$\mathbb{P}\left[|\hat{F}^j_{S_j}(e) - F_{S_j}(e)| \geq \varepsilon F_{S_j}(e)\right] \leq 2\exp\left(-\frac{\varepsilon^2 \alpha F_{S_j}(e)}{3F(e)}\right)$$
$$\leq 2\exp\left(-\frac{\varepsilon^2 \alpha}{3\gamma}\right) \leq 1/n^3$$

$F(e) > \gamma F_{S_j}(e)$**:** Here we can set $\mu = F(e)/\gamma \geq F_{S_j}(e)$ and get:

$$\mathbb{P}\left[|\hat{F}^j_{S_j}(e) - F_{S_j}(e)| \geq \varepsilon F(e)/\gamma\right] \leq 2\exp\left(-\frac{\varepsilon^2 \alpha F(e)}{3\gamma F(e)}\right)$$
$$\leq 2\exp\left(-\frac{\varepsilon^2 \alpha}{3\gamma}\right) \leq 1/n^3$$

Where in both cases, the last inequality is by setting an appropriate constant in $\alpha = \Theta(\varepsilon^{-2}\gamma \log n)$. In the first case the deviation is at most $\varepsilon F_{S_j}(e)$, and in the second it is at most $\varepsilon F(e)/\gamma \leq \varepsilon F(S^*)/\gamma$, as $\{e\}$ is a feasible solution and F is monotone. Hence, in both cases $\hat{F}^j_{S_j}$ is an $(\varepsilon, \varepsilon/\gamma)$-approximate incremental oracle. For both bounded and unbounded curvature, we take a union bound over all $e \in E$ and $j \in [k]$ (at most n^2 values), which concludes the proof. □

G Additional experiments

Below are additional experimental results.

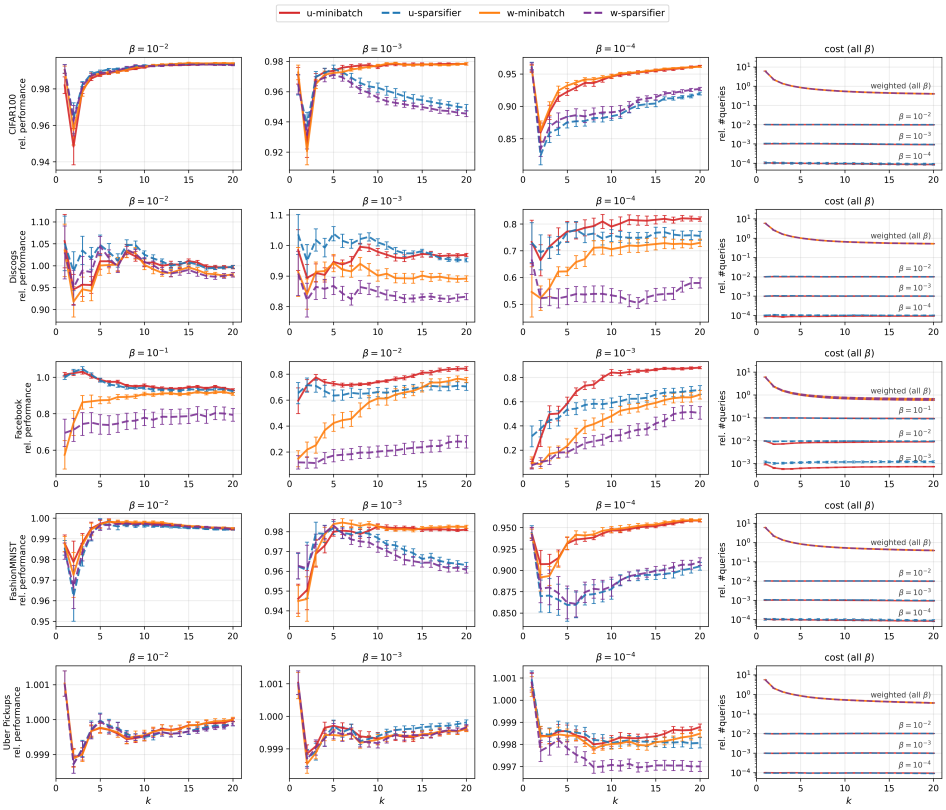

Figure 2: Sparsifier and mini-batch, augmented with the sampling scheme of stochastic-greedy, for $\varepsilon = 0.1$. The plots are as in Figure 1, except that all values are relative to *stochastic-greedy* executed on the exact objective rather than to lazy-greedy; this is why some curves exceed 1.

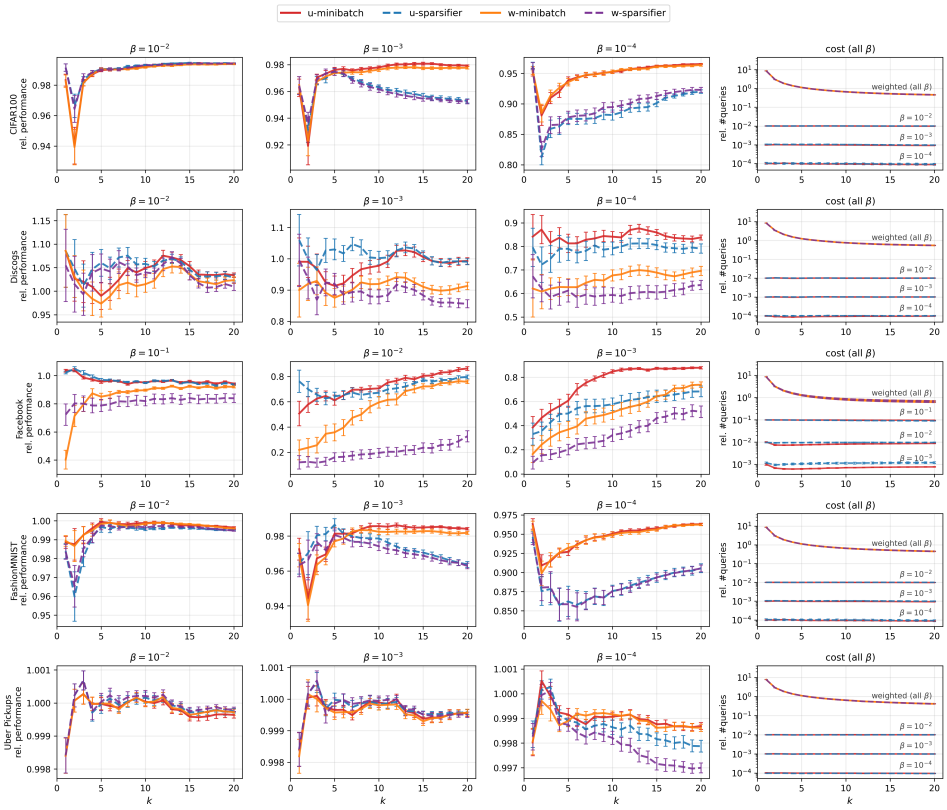

Figure 3: Sparsifier and mini-batch, augmented with the sampling scheme of stochastic-greedy, for $\varepsilon = 0.2$. The plots are as in Figure 1, except that all values are relative to *stochastic-greedy* executed on the exact objective rather than to lazy-greedy; this is why some curves exceed 1.