Revisiting stochastic submodular maximization with cardinality constraint: A bandit perspective

Anonymous authors
Paper under double-blind review

Abstract

In this paper, we focus on the problem of maximizing non-negative, monotone, stochastic submodular functions under cardinality constraint. Recent works have explored continuous optimization algorithms via multi-linear extensions for such problems and provided appropriate approximation guarantees. We take a fresh look into this problem from a discrete, (stochastic) greedy perspective under a probably approximately correct (PAC) setting, i.e., the goal is to obtain solutions whose expected objective value is greater than or equal to $(1-1/e-\epsilon)\text{OPT}-\nu$ with at least $1-\delta$ probability, where OPT is the optimal objective value. Using the theory of multi-armed bandits, we propose novel bandit stochastic greedy (BSG) algorithms in which selection of the next element at iteration i is posed as a (ν_i, δ_i)-PAC best-arm identification problem. Given (ν, δ)-PAC parameters to BSG, we formally characterize a set of per-iteration (ν_i, δ_i)-policies such that any policy from this set guarantees a (ν, δ)-PAC solution for the stochastic submodular maximization problem using BSG. Next, we derive the optimal (ν_i^*, δ_i^*)-policy from this set which incurs the least computational cost in terms of the number of stochastic function calls (N). With the obtained optimal policy, we show that BSG has better complexity in N than the existing approaches. Lastly, we also analyze the inverse fixed-budget problem, i.e., obtaining the best per-iteration policy given a fixed budget $N = N_0$ of stochastic function calls. Experiments on various problems illustrate the efficacy of our approach in terms of optimization quality as well as computational efficiency.

1 Introduction

Submodular functions are those that have the property of diminishing returns, where the incremental gain of adding an element to a set decreases as the size of the set increases. They are often regarded as discrete analogs of convex functions and arise naturally in applications such as document summarization (Bairi et al., 2015; Kim et al., 2016; Gurumoorthy et al., 2019), image summarization (Tschiatschek et al., 2014), data/feature subset selection (Wei et al., 2015), speech data training (Wei et al., 2014), facility location (Mehrotra & Vishnoi, 2023; Karimi et al., 2017), influence maximization in social networks (Kempe et al., 2003; Chen et al., 2009), sensor placement, active learning (Wei et al., 2014), and recommender systems (Mehrotra & Vishnoi, 2023), to name a few. We refer interested readers to the survey (Krause & Golovin, 2014) for further introduction to submodular functions.

Given a ground set $\mathcal{S} = [n]$ consisting of first n natural numbers, a set function $F : 2^{\mathcal{S}} \to \mathbb{R}$ is submodular if it satisfies $F(A) + F(B) \geq F(A \cap B) + F(A \cup B)$ for all $A, B \subseteq \mathcal{S}$. Defining $g(i|A) = F(A \cup \{i\}) - F(A)$ as the incremental gain of adding an element i to the set A, a submodular function equivalently satisfies the diminishing returns property: $g(i|A) \geq g(i|B)$ for any $A \subseteq B \subseteq \mathcal{S}$ and $i \notin B$. The function is monotone non-decreasing if $g(i|A) \geq 0$ for all $i \in \mathcal{S} \setminus A$. The submodular maximization problem is typically stated as identifying the subset $\mathbf{s}^*$ under some constraints that maximizes the submodular function F. In this regard, a common optimization problem is maximizing F under a cardinality constraint, stated as:

$$\max_{\mathbf{s} \subseteq \mathcal{S}, |\mathbf{s}| \leq k} F(\mathbf{s}). \tag{1}$$

As submodular maximization is NP-hard in general, fast algorithms are sought to approximate $\text{OPT} := \max_{\mathbf{s} \subseteq \mathcal{S}, |\mathbf{s}| \leq k} F(\mathbf{s})$ with theoretical guarantees (Nemhauser et al., 1978).

Problem (1) is classically studied in an oracle model, where the access to F is made available via a black box which when fed a subset $\mathbf{s}$ outputs $F(\mathbf{s})$. However, recent works (Karimi et al., 2017; Mokhtari et al., 2018) have also investigated (1) in settings where F is estimated from large amount of data involving stochastic fluctuations, or is not directly accessible and computed only via simulations. Examples of such applications include influence maximization, exemplar-based clustering, recommender systems, prototype selection, etc. In influence maximization, for instance, the goal is to find the most influential seed nodes (of size k) that maximize the propagation of influence through the network (Kempe et al., 2003). The subset of nodes influenced from a seed set is modeled as an expectation of a stochastic process, and may not be computable in closed-form. In facility location, the objective may be viewed as an expected value of similarity between a large number of data points and chosen k centers (with respect to the distribution of data points).

In the above applications, the objective function for stochastic submodular maximization (SSM) problems could be expressed as $F(\mathbf{s}) := \mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s};\mathbf{z})]$, where f is a stochastic function and $\mathbf{z} \in \mathcal{Z}$ is a random variable with distribution P. When $F(\cdot)$ is computed from large number of data points or simulations, standard discrete algorithms such as (lazy) greedy (Minoux, 1978) or (lazy) stochastic greedy (Mirzasoleiman et al., 2015) incur a high computational cost. Recent works (Karimi et al., 2017; Hassani et al., 2017; Mokhtari et al., 2018; 2020) have proposed stochastic continuous optimization algorithms for SSM which leverage the multi-linear extension (Vondrák, 2008; Calinescu et al., 2011) of submodular problems.

In this work, we take a fresh look into non-negative monotone SSM problems with cardinality constraint from a greedy perspective and under provably approximately correct (PAC) setting. In each iteration of the (stochastic) greedy approach, we view the set of candidate elements as *arms* with expected rewards equaling their incremental gains. Hence, choosing the candidate with the highest incremental gain is similar to the best-arm identification problem in the multi-armed bandit (MAB) literature (Even-Dar et al., 2006; Audibert et al., 2010).

Overall, we propose novel bandit stochastic greedy (BSG) algorithms for solving SSM problems with cardinality constraint. In particular, we pose the next element selection problem in each iteration i of BSG as a (ν_i, δ_i)-PAC best-arm identification problem. To concretize our theoretical guarantees for BSG, we answer the following questions.

Q1 Which set of per-iteration (ν_i, δ_i)-policies would guarantee a (ν, δ)-PAC solution for BSG?

Q2 What is the optimal per-iteration policy for BSG in terms of the number of stochastic function f calls?

Q3 Given a fixed budget of the number of stochastic function f calls (N_0), which per-iteration policy incurs the lowest error for BSG?

Our main contributions are as follows.

- For the proposed BSG approach, we first formalize a set of per-iteration (ν_i, δ_i)-policies that guarantee a (ν, δ)-PAC solution for SSM problems with cardinality constraint. We show that this set always contains non-trivial policies. Thus, we prove the correctness of BSG.
- For BSG, we next derive the optimal per-iteration policy in the above formalized set which corresponds to the least computational cost in terms of the number of stochastic function f calls (N). Our results are summarized in Table 1. We observe that the proposed BSG (and bandit greedy BG) algorithms have lower stochastic function call (N) complexity than the existing approaches.
- We also analyze BSG in the resource constrained setting. In particular, given a fixed budget on the number of stochastic function calls $N = N_0$ and a confidence δ, we derive the best per-iteration policy in the above formalized set (in terms of lowest error incurred by BSG).

Empirical results illustrate the efficacy of the proposed approach on applications such as exemplar based clustering and representative sampling from a target set. The proofs of all theoretical results and additional experimental details are provided are in the appendix.

Table 1: Approximation guarantees and number of stochastic function f calls (N) for various algorithms on the SSM problem (2). We define $\gamma(\epsilon) = 1 - 1/e - \epsilon$ and m denotes the maximum number of stochastic function f calls required to compute the function $F(\cdot)$. In exemplar based clustering, for instance, $m = n$. For discrete stochastic methods (disc. stoch.), the expectation over $F(\mathbf{s})$ is over the randomized sampling of the candidate set. For continuous stochastic methods (cont. stoch.), the expectation over $F(\mathbf{s})$ is due to multi-linear extension, stochastic gradients, and randomized pipage rounding. Greedy is the only deterministic method in the table. We observe that the proposed algorithms (BG-ABA, BG-NE, BSG-ABA, and BSG-NE) have lower complexity of N in terms of n.

Algorithm	Type	Approximation guarantee	Number of f calls (N)
Greedy	discrete	$F(\mathbf{s}) \geq \gamma(0)\mathrm{OPT}$	$O(nkm)$
BG-ABA (**this work**)	discrete	$\mathbb{P}[F(\mathbf{s}) \geq \gamma(0)\mathrm{OPT} - \nu] \geq 1 - \delta$	$O(nk^3 \log(\frac{k}{\delta})/\nu^2)$
BG-NE (**this work**)	discrete	$\mathbb{P}[F(\mathbf{s}) \geq \gamma(0)\mathrm{OPT} - \nu] \geq 1 - \delta$	$O(nk^3 \log(\frac{nk}{\delta})/\nu^2)$
Stochastic Greedy (Mirzasoleiman et al., 2015)	disc. stoch.	$\mathbb{E}[F(\mathbf{s})] \geq \gamma(\epsilon)\mathrm{OPT}$	$O(nm \log(1/\epsilon))$
BSG-ABA (**this work**)	disc. stoch.	$\mathbb{P}[\mathbb{E}[F(\mathbf{s})] \geq \gamma(\epsilon)\mathrm{OPT} - \nu] \geq 1 - \delta$	$O(nk^2 \log(\frac{k}{\delta}) \log(\frac{1}{\epsilon})\gamma(\epsilon)^2/\nu^2)$
BSG-NE (**this work**)	disc. stoch.	$\mathbb{P}[\mathbb{E}[F(\mathbf{s})] \geq \gamma(\epsilon)\mathrm{OPT} - \nu] \geq 1 - \delta$	$O(nk^2 \log(\frac{n}{\delta} \log(\frac{1}{\epsilon})) \log(\frac{1}{\epsilon})\gamma(\epsilon)^2/\nu^2)$
SGA (Hassani et al., 2017)	cont. stoch.	$\mathbb{E}[F(\mathbf{s})] \geq (1/2)\mathrm{OPT} - \nu$	$O(n^2k^2/\nu^2)$
SCG (Mokhtari et al., 2018)	cont. stoch.	$\mathbb{E}[F(\mathbf{s})] \geq \gamma(0)\mathrm{OPT} - \nu$	$O(n^{2.5}k^{1.5}/\nu^3)$
SCG++ (Hassani et al., 2019)	cont. stoch.	$\mathbb{E}[F(\mathbf{s})] \geq \gamma(0)\mathrm{OPT} - \nu$	$O(n^2k^4/\nu^2)$

2 Related work

This section briefly reviews the literature on submodular maximization, stochastic submodular maximization, and the best-arm identification (or pure exploration) problem under the multi-armed bandit framework.

2.1 Submodular maximization

Discrete greedy approaches. For maximizing a monotone, non-negative, deterministic submodular function F under cardinality constraint k, the landmark paper (Nemhauser et al., 1978) employed the Greedy algorithm to obtain the $1 - e^{-1}$ approximation guarantee and proved that any algorithm that is allowed to only evaluate F at a polynomial number of sets will not be able to obtain an approximation guarantee better than $1 - e^{-1}$. Exploiting the properties of submodularity, Minoux (1978) proposed an accelerated version of the classical greedy algorithm, popularly known as Lazy Greedy (Leskovec et al., 2007). The Stochastic Greedy algorithm (Mirzasoleiman et al., 2015) provides $1 - e^{-1} - \epsilon$ approximation in expectation using $O(n \log(\frac{1}{\epsilon}))$ function F evaluations. A deterministic $1 - e^{-1} - \epsilon$ approximation algorithm requiring $O(n/\epsilon)$ calls to F has been recently proposed (Li et al., 2022).

Continuous greedy approaches. Existing works (Vondrák, 2008; Calinescu et al., 2011; Chekuri et al., 2014) have explored continuous relaxations of submodular functions via multi-linear extension, which lifts the discrete problem (1) into a continuous domain and perform continuous optimization using gradients. Calinescu et al. (2011) showed that the continuous greedy algorithm obtains the tight $1 - e^{-1}$ approximation bound for monotone submodular functions under a general matroid constraint. Randomized pipage rounding (Calinescu et al., 2011; Karimi et al., 2017) is usually employed to obtain a discrete solution from fractional solution without worsening the objective value in expectation.

2.2 Stochastic submodular maximization (SSM)

While the classical Greedy or the Stochastic Greedy (Mirzasoleiman et al., 2015) algorithms may be employed for solving SSM problem ($F(\mathbf{s}) \coloneqq \mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s};\mathbf{z})]$) defined over large number of data points or simulations,

they compute the function $F(\cdot)$ exactly at each iteration i. Hence, they require the full data at each iteration and incur high computational cost. Recent works have explored stochastic gradient based continuous greedy approaches for SSM as they require only a small batch of data at every iteration.

For maximizing weighted coverage functions, a subclass of stochastic submodular functions, Karimi et al. (2017) proposed a concave relaxation scheme and employed projected stochastic gradient ascent (SGA) algorithm. However, for SSM problems, SGA does not provide tight guarantees as it offers $\mathrm{OPT}/2 - \nu$ lower bound in expectation after $O(n^2k^2/\nu^2)$ iterations (Hassani et al., 2017) Conditional gradient based stochastic continuous greedy (SCG) improves upon SGA and achieves $\left(1 - e^{-1}\right)\mathrm{OPT} - \nu$ guarantee in expectation after $O(n^{2.5}k^{1.5}/\nu^3)$ iterations (Mokhtari et al., 2020). Hassani et al. (2020) proposed variance reduction based SCG, SCG++, which provides the same guarantee as SCG in $O(n^2k^4/\nu^2)$ iterations. Table 1 summarizes the approximation guarantee and the computational cost (in terms of the number of stochastic function f calls) provided by Greedy, Stochastic Greedy, the stochastic gradient based continuous greedy approaches (SGA, SCG, and SCG++), and our proposed algorithms.

2.3 Best-arm identification problem

Consider a set $\mathcal{S}$ of n arms. When an arm $s \in \mathcal{S}$ is sampled (i.e., pulled), a reward $\mu(s)$ is received. This reward is realization of a random variable from an unknown distribution $D(s)$. The pure exploration (or best arm identification) problem (Bubeck et al., 2009; Audibert et al., 2010) is defined as identifying an arm s^* with the highest expected reward: $s^* \in \arg\max_{s\in\mathcal{S}} \mathbb{E}_{D(s)}[\mu(s)]$. Thus, estimating the expected reward of an arm involves sampling them sufficiently. Learning a (ν, δ)-best arm under probably approximately correct (PAC) setting implies identifying an arm s' such that $\mu(s') \geq \mu(s^*) - \nu$ with at least $1 - \delta$ probability. Here, ν and δ are usually termed as the error and confidence parameters, respectively. An interesting problem is to identify a (ν, δ)-best arm while incurring minimal sampling complexity (Jamieson & Nowak, 2014).

Elimination strategy is a common approach for the (ν, δ)-PAC problem, with median elimination (Even-Dar et al., 2006) being one of the most popular algorithms. Elimination framework involves sampling from all the non-eliminated arms at each around and eliminating some of the arms at the end of each round. The celebrated Hoeffding bound provides the following simple yet efficient naive elimination (NE) procedure for obtaining a (ν, δ)-best arm $s^* \in \mathcal{S}$:

1. Given ν, δ parameters, sample each arm $2\log(n/\delta)/\nu^2$ times and record the mean empirical reward $\hat{\mu}(s)$ for each arm $s \in \mathcal{S}$
2. Select the arm $s^* := \arg\max_{s\in\mathcal{S}} \hat{\mu}(s)$.

Recently, Hassidim et al. (2020) proposed the *approximate best arm* (ABA) algorithm that learns a (ν, δ)-best arm with lower number of samples than the median elimination algorithm. The ABA algorithm employs two elimination strategies – aggressive elimination (AE) and NE – one after the other. The aim of AE is to use only a few samples to eliminate a significant number of potentially uninteresting arms with high confidence. The subsequent NE stage takes the surviving arms from AE as well as a randomly chosen subset of $\mathcal{S}$ (to account for the small probability that AE may eliminate good arms) and outputs a (ν, δ)-best arm. We detail the steps of the ABA algorithm in Section G.

3 Proposed approach

Given a ground set $\mathcal{S} = \{1, \ldots, n\}$ of n elements, we consider the problem of maximizing a non-negative, monotone, stochastic submodular function $F : 2^{\mathcal{S}} \to \mathbb{R}_+$, subject to cardinality constraint, in the setting where the function F is defined as $F(\mathbf{s}) := \mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s}; \mathbf{z})]$. The function $f : 2^{\mathcal{S}} \times \mathcal{Z} \to \mathbb{R}_+$ may be given as a stochastic function (as part of an implicit stochastic model for which the distribution P may be unknown) or could be an empirical function (with possibly large domain $\mathcal{Z}$). Overall, the SSM problem is formalized as follows:

$$\max_{\mathbf{s}\subseteq\mathcal{S}, |\mathbf{s}|\leq k} F(\mathbf{s}) (:= \mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s}; \mathbf{z})]). \tag{2}$$

Algorithm 1 Proposed bandit greedy (BG) and bandit stochastic greedy (BSG) algorithms

Input: $\mathcal{Z}, k, \nu, \delta$, and ϵ.
Select a per-iteration (ν_i, δ_i) policy satisfying (6).
Initialize $\mathbf{s}_0 = \phi, i = 1$.
while $i \leq k$ **do**
  1: Greedy: set $\mathcal{R}_i = \mathcal{S} \setminus \mathbf{s}_{i-1}$
  or
  Stochastic greedy: set $\mathcal{R}_i$ as a random subset of $\mathcal{S} \setminus \mathbf{s}_{i-1}$ with $nk^{-1}\log(1/\epsilon)$ elements.
  2: $s = \text{PAC_BEST_ARM}(\mathbf{s}_{i-1}, \mathcal{R}_i, \mathcal{Z}, \nu_i, \delta_i)$.
  3: $\mathbf{s}_i = \mathbf{s}_{i-1} \cup \{s\}$, $i = i + 1$.
end while
Output: $\mathbf{s}_k$.

It should be noted that the stochastic function $f(\mathbf{s}; \mathbf{z})$ need not be submodular (Mokhtari et al., 2018; 2020). Since our analysis involves application of Hoeffding's inequality, we assume that the range of f is $[0, 1]$ for simplified expressions. As with existing works on approximate best-arm identification problem (Bagaria et al., 2018; Hassidim et al., 2020), all theoretical results of this paper can be generalized to any sub-Gaussian distribution.

The classical Greedy algorithm (Nemhauser et al., 1978) for (2) begins with the empty set $\mathbf{s}_0 = \emptyset$. In the i-th iteration, it finds the element s that maximizes the incremental gain g_i, i.e., $s \in \arg\max_{v \in \mathcal{R}_i} g_i(v|\mathbf{s}_{i-1})$, where $\mathcal{R}_i = \mathcal{S} \setminus \mathbf{s}_{i-1}$ is the candidate element set and $g_i(v|\mathbf{s}_{i-1}) = F(\mathbf{s}_{i-1} \cup \{v\}) - F(\mathbf{s}_{i-1})$ as defined earlier. We then update $\mathbf{s}_i \leftarrow \mathbf{s}_{i-1} \cup \{s\}$. On the other hand, Stochastic Greedy (Mirzasoleiman et al., 2015) chooses a smaller (but random) candidate element set $\mathcal{R}_i \subseteq \mathcal{S} \setminus \mathbf{s}_{i-1}$ for the i-th iteration, with $|\mathcal{R}_i| = \frac{n}{k}\log(\frac{1}{\epsilon})$.

Both Greedy and Stochastic Greedy require multiple evaluations of the function $F(\cdot)$, which may be computationally costly or even impractical in several real-world applications (Mokhtari et al., 2018; 2020). We propose novel discrete stochastic greedy based algorithms that require a small number of stochastic function f calls in each iteration and provide approximation guarantees.

3.1 Bandit stochastic greedy (BSG)

Each iteration of the stochastic greedy framework involves greedily selecting an element with the maximum incremental gain. The incremental gain for the SSM problem (2) with respect to an element $s \in \mathcal{R}_i$ may be written as:

$$\begin{aligned} g(s|\mathbf{s}) &= \mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s} \cup \{s\}; \mathbf{z})] - \mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s}; \mathbf{z})] \\ &= \mathbb{E}_{\mathbf{z}\sim P}[g(s|\mathbf{s}; \mathbf{z})], \end{aligned} \tag{3}$$

where $g(s|\mathbf{s}; \mathbf{z}) \coloneqq f(\mathbf{s} \cup \{s\}; \mathbf{z}) - f(\mathbf{s}; \mathbf{z})$ is the stochastic marginal gain (corresponding to the sample $\mathbf{z} \sim P$) when an element s is added to $\mathbf{s}$. Since (3) is an expectation, we may obtain an unbiased estimator of $g(s|\mathbf{s})$ as $\hat{g}(s|\mathbf{s}; \mathcal{Z}_i) = \sum_{\mathbf{z}\in\mathcal{Z}_i} g(s|\mathbf{s}; \mathbf{z})/|\mathcal{Z}_i|$ given a set $\mathcal{Z}_i$ whose elements are sampled from P. We also note that the term $\mathbb{E}_{\mathbf{z}\sim P}[f(\mathbf{s}; \mathbf{z})]$ in (3) is constant with respect to element s. Thus, we may equivalently use the estimator $\hat{h}(s|\mathbf{s}; \mathcal{Z}_i) = \sum_{\mathbf{z}\in\mathcal{Z}_i} f(s|\mathbf{s}; \mathbf{z})/|\mathcal{Z}_i|$ for greedily selecting the next element.

We now propose to employ the best arm selection strategies from the stochastic multi-armed bandit (MAB) framework for greedily selecting the next element. We view the different candidates $s \in \mathcal{R}_i$ as individual arms (agents) with the corresponding reward given by $\hat{h}(\cdot)$. Thus, identifying the best element that maximizes the incremental gain under (ν, δ)-PAC setting may be viewed as a pure exploration problem (Even-Dar et al., 2006; Bubeck et al., 2009). If s_i^* is the element with best incremental gain (3) in the i-th iteration of the (stochastic) greedy framework, a (ν_i, δ_i)-best element s_i satisfies the following condition:

$$\mathbb{P}[g(s_i|\mathbf{s}_{i-1}) \geq g(s^*|\mathbf{s}_{i-1}) - \nu_i] \geq 1 - \delta_i. \tag{4}$$

We formalize our bandit greedy (BG) and bandit stochastic greedy (BSG) approaches in Algorithm 1. The `PAC_BEST_ARM` function denotes a best-arm identification algorithm that returns a (ν_i, δ_i)-best element at

i-th iteration. In this work, we analyze the proposed BG and BSG approaches with `PAC_BEST_ARM` as the NE or the ABA algorithm (discussed in Section 2.3). If ABA is employed in Algorithm 1, we term it BSG-ABA (or BG-ABA) and if NE is employed, we term it BSG-NE (or BG-NE).

In the rest of this section, we discuss the approximation guarantees provided by our BSG algorithms. Analogous results for the BG algorithms are discussed in Section C.

Table 1 summarizes the key theoretical results for the BSG-ABA, BSG-NE, BG-ABA, and BG-NE algorithms. In the stochastic greedy framework with the (stochastic greedy) parameter ϵ, a set $\mathbf{s} \subseteq \mathcal{S}$ (with $|\mathbf{s}| \leq k$) is termed as a (ν, δ)-PAC solution of (2) if the following holds:

$$\mathbb{P}\left(\mathbb{E}\left[F(\mathbf{s})\right] \geq (1 - 1/e - \epsilon)\mathrm{OPT} - \nu\right) \geq 1 - \delta. \tag{5}$$

We begin our analysis by noting the following open questions regarding Algorithm 1.

1. **Correctness of BSG and characterizing a set of feasible per-iteration policies**: Given ν and δ parameters, can Algorithm 1 obtain a (ν, δ)-PAC solution for the SSM problem (2)? If yes, can we characterize a set of such per-iteration (ν_i, δ_i) policies which ensure (ν, δ)-PAC solution for (2)?

2. **Optimal per-iteration policy**: Among the per-iteration policies present in the above set which per-iteration (ν_i^*, δ_i^*) policy would incur the lowest number of stochastic function f calls (N)?

3. **Optimal (lowest) error ν and the corresponding per-iteration policy for fixed budget setting**: Given a fixed budget of the number of stochastic function calls $N = N_0$ and confidence δ, what is the best (ν, δ)-PAC guarantee ensured by Algorithm 1 for the problem (2)? What is the corresponding per-iteration (ν_i, δ_i) policy?

3.2 Characterizing a set of per-iteration policies which ensure (ν, δ)-PAC solution for (2)

Our first result shows that the proposed BSG algorithm indeed learns a (ν, δ)-PAC solution of (2) if the per-iteration (ν_i, δ_i) policy is designed systematically.

Theorem 3.1. *Let* $\mathbf{s}$ *be the solution obtained by BSG (BSG-ABA or BSG-NE) for (2) with given PAC parameters* (ν, δ) *and stochastic greedy parameter* $\epsilon \in (0, 1 - 1/e)$*. Then,* $\mathbf{s}$ *is a* (ν, δ)*-PAC solution of (2) if per-iteration* (ν_i, δ_i) *policy belongs to the following set*

$$\mathcal{A}(\nu, \delta) \coloneqq \{\{(\nu_i, \delta_i)\}_{i=1}^k : \sum_{i=1}^k \delta_i = \delta, (1 - \epsilon) \sum_{i=1}^k \beta^{k-i} \nu_i \leq \nu\}, \tag{6}$$

where $\beta = 1 - (1 - \epsilon)/k$.

Proof. The proof is provided in Section B.1. □

Remark 3.2. The ABA algorithm (Hassidim et al., 2020) requires its confidence parameter to lie in the range $(0, 0.05]$. Hence, we have the following constraint on user defined δ for BSG-ABA: $\delta \in \begin{cases} (0, 0.05k] & k < 20 \\ (0, 1) & k \geq 20 \end{cases}$.

Theorem 3.1 provides various options for designing the per-iteration (ν_i, δ_i) policies for the SSM problem (2). For example, if we model $\nu_i = \nu_0 \ \forall i$ and solve for ν_0 in $(1 - \epsilon) \sum_{i=1}^k \beta^{k-i} \nu_i = \nu$, we obtain $(\nu_i, \delta_i) = (\frac{\nu}{k(1-\beta^k)}, \frac{\delta}{k}) \ \forall i$. Alternatively, since $\beta < 1$, one can upper bound $(1-\epsilon) \sum_{i=1}^k \beta^{k-i} \nu_i$ with $(1-\epsilon) \sum_{i=1}^k \nu_i$. If we equate the latter with ν and solve for ν_0 (with $\nu_i = \nu_0$), we get another feasible policy $(\nu_i, \delta_i) = (\frac{\nu}{k(1-\epsilon)}, \frac{\delta}{k}) \ \forall i$. While Theorem 3.1 guarantees that both these example policies would allow BSG to obtain a (ν, δ)-PAC solution, the computational cost of BSG in terms of number of stochastic function f calls (N) would be different. Between the above two example policies, the former would incur lesser cost. In the following sections, we investigate designing per-iteration policies while taking the computational cost of BSG into consideration.

3.3 Per-iteration (ν_i, δ_i) policy with lowest computational cost

Theorem 3.1 characterizes a family (or set) of $\{(\nu_i, \delta_i)\}_{i=1}^k$ policies $\mathcal{A}(\nu, \delta)$ for BSG which ensure (ν, δ)-PAC solution for the problem (2). However, the computational cost (in terms of the number of stochastic function f calls N) for obtaining a (ν, δ)-PAC solution depends on the chosen policy. For a given per-iteration (ν_i, δ_i) policy, the following lemma provides an expression of N for the proposed BSG algorithms.

Lemma 3.3. *Let N_{ABA} and N_{NE} be (tight) upper bounds on the number of stochastic function f evaluations required by BSG-ABA and BSG-NE (Algorithm 1), respectively. Then,*

$$\begin{aligned} N_{\text{ABA}} &= \sum_{i=1}^{k} \frac{18n}{k\nu_i^2} \log\left(\frac{1}{\delta_i}\right) \log\left(\frac{1}{\epsilon}\right), \\ N_{\text{NE}} &= \sum_{i=1}^{k} \frac{2n}{k\nu_i^2} \log\left(\frac{n}{k\delta_i} \log\left(\frac{1}{\epsilon}\right)\right) \log\left(\frac{1}{\epsilon}\right). \end{aligned} \tag{7}$$

Proof. The proof is provided in Section B.2. □

We next consider the problem of finding which per-iteration policy in the set $\mathcal{A}(\nu, \delta)$ incurs the lowest computational cost N. From the N_{ABA} and N_{NE} expressions (7), we observe that the error ν_i parameters influence them more significantly than the confidence parameters δ_i (quadratic versus logarithmic dependency). Hence, we propose to minimize N_{ABA} or N_{NE} in (7) with respect to the $\{\nu_i\}_{i=1}^k$ parameters under the constraint defined in (6). The confidence parameters $\{\delta_i\}_{i=1}^k$ are set to be uniform across iterations ($\delta_i = \delta/k$) in order to obtain simplified expressions. To this end, in both the ABA and NE cases, we require solving the following convex optimization problem:

$$\begin{aligned} \min_{\nu_i > 0 \forall i} \quad & \sum_{i=1}^{k} \nu_i^{-2} \\ \text{s.t.} \quad & (1-\epsilon) \sum_{i=1}^{k} \beta^{k-i} \nu_i \leq \nu. \end{aligned} \tag{8}$$

Overall, we have the following result.

Theorem 3.4. *In the set of policies defined by the set $\mathcal{A}(\nu, \delta)$ (6), let $\{(\nu_i^*, \delta_i^* = \delta/k)\}_{i=1}^k$ be an optimal per-iteration policy corresponding to the minimum number of stochastic function f calls required by BSG to obtain a (ν, δ)-PAC solution. Then,*

$$\nu_i^* = \frac{\nu(1-\beta^{2/3})\beta^{(i-k)/3}}{(1-\epsilon)(1-\beta^{2k/3})} \ \forall i.$$

The expressions of N_{ABA} and N_{NE} corresponding to (ν_i^*, δ_i^*) are

$$\begin{aligned} N_{\text{ABA}}^* &= \frac{18n(1-\epsilon)^2}{\nu^2} \log\left(\frac{k}{\delta}\right) \log\left(\frac{1}{\epsilon}\right) c(k) \quad \text{and} \\ N_{\text{NE}}^* &= \frac{2n(1-\epsilon)^2}{\nu^2} \log\left(\frac{n}{\delta} \log\left(\frac{1}{\epsilon}\right)\right) \log\left(\frac{1}{\epsilon}\right) c(k), \end{aligned}$$

where $c(k) = \frac{(1-\beta^{2k/3})^3}{k(1-\beta^{2/3})^3}$ and $\beta = 1 - (1-\epsilon)/k$.

Proof. The proof involves solving the problem (8) by using the Hölder inequality. Details are provided in Section B.3. □

Remark 3.5 (**Complexity of BSG-ABA and BSG-NE algorithms**). It can be shown that $c(k) \leq k^2$, resulting in linear $O(n)$ complexity of BSG-ABA and almost linear $O(n \log(n))$ complexity of BSG-NE algorithms (ignoring the constants ν, δ, and ϵ and assuming $k \ll n$). This result is significant as for popular

SSM problems such as exemplar-based clustering, both Greedy and Stochastic Greedy have quadratic $O(n^2)$ complexity. From Table 1, we also observe that existing continuous stochastic greedy approaches SGA, SCG, and SCG++ have at least quadratic $O(n^2)$ complexity.

Remark 3.6. In Theorem 3.4, the optimized choice for $\{\nu_i\}$ is obtained by making the confidence parameters $\{\delta_i\}$ uniform across iterations, i.e., $\delta_i = \delta/k$. In Section D, we show the results obtained by jointly optimizing both $\{\nu_i\}$ and $\{\delta_i\}$. This leads to further improvements in the (minimal) number of stochastic function f calls needed. However, the margin of improvement is quite small.

Remark 3.7. Recently, Tiwari et al. (2020) have proposed the BanditPAM algorithm for exemplar-based clustering (k-medoids) problem. BanditPAM employs a variant of UCB algorithm (Lai & Robbins, 1985; Zhang et al., 2019) and provides guarantees under probably correct setting (i.e., error parameter ν is set to 0). Given the confidence parameter δ, Tiwari et al. (2020) provide the following guarantee: With probability at least $1-\delta$, BanditPAM when restricted to the submodular BUILD step returns the same set of k points as Greedy, and the required number of pairwise distance computations satisfies $\mathbb{E}[N_{\mathrm{BP}}] = O(n^2\delta d_1(k) + n\log(n/\delta)d_2(k))$ (Tiwari et al., 2020, Appendix Remark A1). Tiwari et al. (2020) derive the computational cost of BanditPAM by assuming k to be a constant and does not formally analyze its dependency on k. Hence, we represent this dependency on k as (unknown) functions $d_1(k)$ and $d_2(k)$ in the above expression. However, Tiwari et al. (2020, Appendix Section 2.5) have observed that BanditPAM's computational cost varies from quadratic to cubic on k for some parameter regime. In contrast, as observed in Table 1, our proposed BSG algorithms solve the general SSM problem (2) under a PAC setting and have better computational complexity at a given (ν, δ). Empirically, the computational cost of both BG and BSG algorithms are an order of magnitude lower than BanditPAM. We note that in the exemplar-based clustering problem, each stochastic function f call entails one pairwise distance computation.

3.4 Fixed budget setting

We now discuss the PAC guarantee of BSG in budget setting: given a fixed budget of the number of stochastic function f calls ($N = N_0$) and the overall confidence parameter δ, the aim is to find the best per-iteration policy in the set $\mathcal{A}(\nu, \delta)$. We note that the overall error ν is unknown in this setting and the best per-iteration policy for BSG is the one which achieves the lowest ν for problem (2).

We begin by noting that given a per-iteration policy $\{(\nu_i, \delta_i)\}_{i=1}^k$, with $\delta_i = \delta/k$ (for simplicity), the best (lowest) error achievable by BSG is: $\nu(\nu_1, \ldots, \nu_k) = (1-\epsilon)\sum_{i=1}^k \beta^{k-i}\nu_i$. We also note that (7) in Lemma 3.3 provides expressions of N_{ABA} and N_{NE} for a given per-iteration policy. Hence, to get the best policy under the fixed budget setting, we propose the following convex optimization problem:

$$\begin{aligned} \min_{\nu_i > 0 \forall i} \quad & (1-\epsilon)\sum_{i=1}^{k} \beta^{k-i}\nu_i \\ \text{s.t.} \quad & N_{\mathrm{ABA}} \leq N_0 \text{ (or } N_{\mathrm{NE}} \leq N_0). \end{aligned} \tag{9}$$

Overall, the problem (9) is an optimization problem with the variables $\{\nu_i\}$ and we have the following result.

Theorem 3.8. *Given a fixed budget N_0 (the maximum number of stochastic function f calls allowed), the overall confidence parameter for BSG δ, and the per-iteration confidence as $\delta_i = \delta/k$, BSG-ABA and BSG-NE achieve $(\nu^*_{\mathrm{ABA}}, \delta)$- and $(\nu^*_{\mathrm{NE}}, \delta)$-PAC solutions, respectively, where*

$$\nu^*_{\mathrm{ABA}} = \left(\frac{1-\beta^{2k/3}}{1-\beta^{2/3}}\right)^{3/2} \left(\frac{(1-\epsilon)^2 18n\log(k/\delta)\log(1/\epsilon)}{kN_0}\right)^{1/2},$$

$$\nu^*_{\mathrm{NE}} = \left(\frac{1-\beta^{2k/3}}{1-\beta^{2/3}}\right)^{3/2} \left(\frac{(1-\epsilon)^2 2n\log(n\log(1/\epsilon)/\delta)\log(1/\epsilon)}{kN_0}\right)^{1/2},$$

and $\beta = 1-(1-\epsilon)/k$. The corresponding per-iteration policy of BSG-ABA is given by $\delta^*_i = \delta/k$ and

$$\nu^*_i = \left(\frac{18n\log(k/\delta)\log(1/\epsilon)(1-\beta^{2k/3})}{kN_0(1-\beta^{2/3})}\right)^{1/2} \beta^{(i-k)/3} \ \forall i.$$

Similarly, the per-iteration policy of BSG-NE, corresponding to $(\nu^*_{\text{NE}}, \delta)$*, is given by* $\delta^*_i = \delta/k$ *and*

$$\nu^*_i = \left(\frac{2n\log(n\log(1/\epsilon)/\delta)\log(1/\epsilon)(1-\beta^{2k/3})}{kN_0(1-\beta^{2/3})}\right)^{1/2} \beta^{(i-k)/3} \ \forall i.$$

Proof. The proof involves solving the problem (9) using the Hölder inequality and is detailed in Section B.5. □

3.5 Implementation details

We now discuss a few implementation-related aspects of the proposed BSG approach.

Computing rewards for all the arms on the same sample set. In a given iteration i of BSG (Algorithm 1), let $\mathcal{Z}_i$ be the set of instances $\mathbf{z}$ sampled for the first arm. Then, the same set $\mathcal{Z}_i$ may be used for computing rewards for the remaining candidate arms. This offers several practical advantages without changing any theoretical results: (a) reduces the required number of samples in an iteration (by a factor of the number of arms), which becomes important when acquiring samples is costly, and (b) helps in memory-efficient parallelized computations. However, we note that using the same set $\mathcal{Z}_i$ *does not* reduce the number of stochastic function f calls as rewards needs to be computed for all the candidate arms.

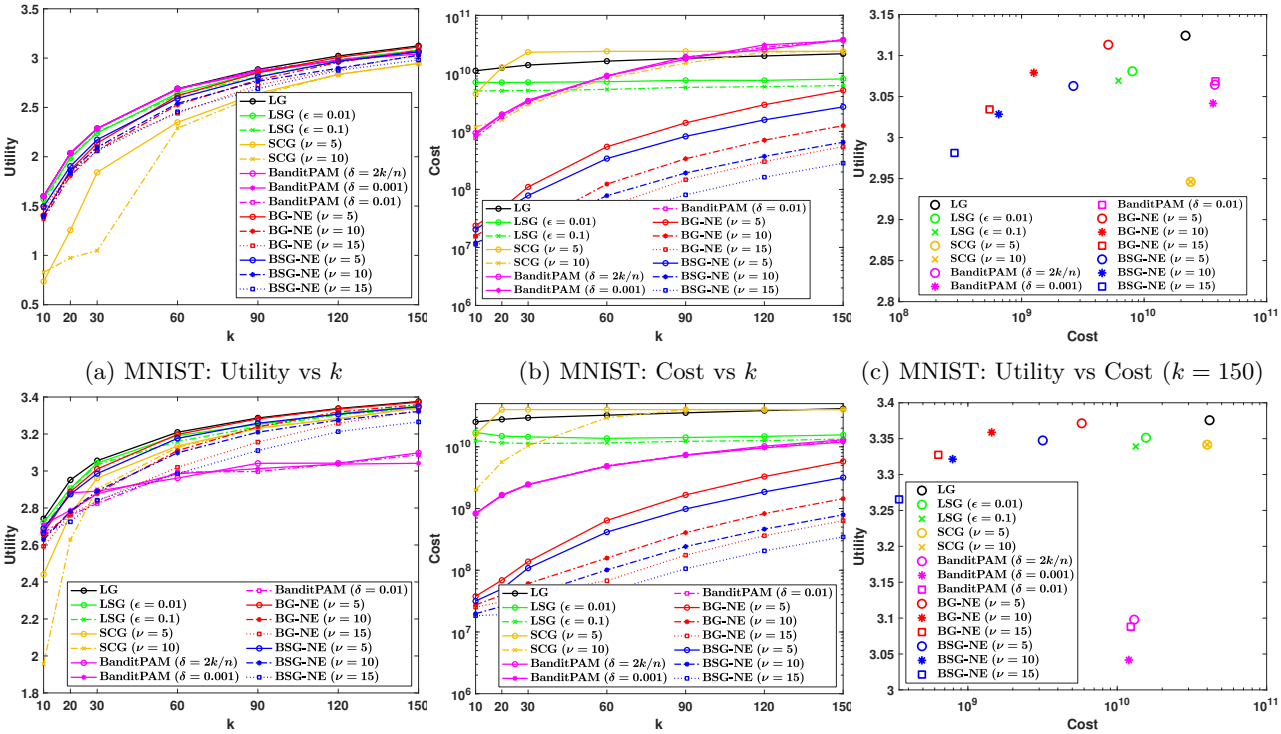
On BSG-ABA versus BSG-NE. In Remark 3.5, we observe that BSG-ABA has lower complexity of $O(n)$ on required number of stochastic function f calls than BSG-NE that has complexity $O(n\log(n))$ for a given (ν, δ)-PAC setting. However, from (7), we note that the constant terms hiding in the O notation of BSG-ABA are larger than those of BSG-NE (N^*_{ABA} has 18 while N^*_{NE} has 2). By setting $N^*_{\text{NE}} \leq N^*_{\text{ABA}}$, we can therefore compute the range $[1, n_0]$ of n over which BSG-NE requires lower number of stochastic function f calls than BSG-ABA. We get $n_0 = \delta(k/\delta)^9/\log(1/\epsilon)$. As discussed in Remark 3.2, $\delta/k \leq 0.05$ for BSG-ABA. Hence, $n_0 \geq 5.12 \times 10^{12}\delta/\log(1/\epsilon)$. For reasonable choice of parameter values $\delta = 0.001$ and $\epsilon = 0.01$, we get $n_0 \geq 2.3 \times 10^9$. Hence, BSG-ABA will have a practical runtime advantage over BSG-NE only in very large scale problem setting.

Overall, we observe that BSG-NE is a well-suited algorithm for the SSM problem (2) as it has lower stochastic function calls complexity than existing SSM algorithms like SGA (Hassani et al., 2017), SCG (Mokhtari et al., 2018), and SCG++ (Hassani et al., 2019). BSG-NE is simple to implement, can take advantage of lazy evaluations (Minoux, 1978), and can parallelize the reward computation across the candidate arms and samples within each iteration. This allows BSG-NE to exploit modern GPGPU computing systems, thereby making BSG-NE our method of choice for experiments discussed in Section 4.

4 Experiments

In this section, we show the benefit of the proposed BSG and BG algorithms on the exemplar-based clustering and representative sampling applications. As discussed, we consider the BSG-NE and BG-NE variants of the proposed Algorithm 1. For both our algorithms, we pick the $\{(\nu_i, \delta_i)\}$ policy (from Theorem 3.4) that leads to the minimal stochastic function f calls for the given (ν, δ) parameters. We consider the following baselines.

1. **LG**: the deterministic Greedy algorithm with lazy evaluations (Minoux, 1978; Leskovec et al., 2007). As it has the best theoretical guarantee, LG is expected to obtain the best generalization performance among the competing methods but with high computational cost.
2. **LSG**: the popular Stochastic Greedy algorithm with lazy evaluations (Mirzasoleiman et al., 2015).
3. **SCG:** the stochastic continuous gradient algorithm proposed in (Mokhtari et al., 2018; 2020).
4. **BanditPAM:** UCB best-arm identification based k-medoids method (Tiwari et al., 2020). We run only the BUILD step of BanditPAM, which corresponds to solving the submodular exemplar-based clustering problem. The authors' C++ code link: `https://github.com/motiwari/BanditPAM`.



(a) MNIST: Utility vs k (b) MNIST: Cost vs k (c) MNIST: Utility vs Cost ($k = 150$)

(d) TIN: Utility vs k (e) TIN: Cost vs k (f) TIN: Utility vs Cost ($k = 150$)

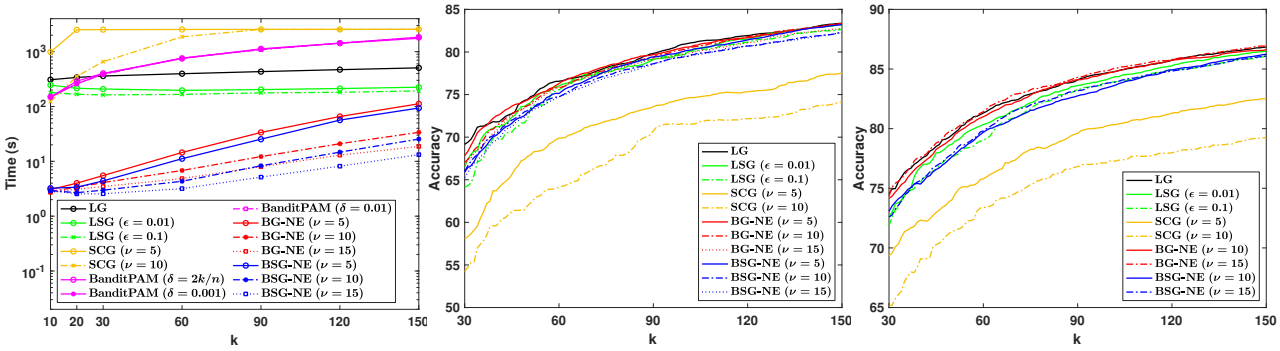
Figure 1: Performance of various algorithms on the exemplar-based clustering problem. Top row: MNIST dataset, Bottom row: TIN dataset. We observe that the proposed algorithms BSG and BG obtain a good utility versus cost trade-off by varying error parameter ν and outperforms other SSM methods such as LSG, SCG, and BanditPAM.

All the algorithms (except BanditPAM) are implemented in Matlab. The experiments are run on Intel Xeon CPU (3.6 GHz) with 6 cores and 64 GB RAM. The following datasets are considered.

- **MNIST** (LeCun et al., 1998) is a handwritten digits dataset with 28×28 pixels greyscale images of 10 classes for digits $\{0, 1, \ldots 9\}$. It has two different sets of 60 000 samples (train) and 10 000 samples (test).
- **TinyImageNet (TIN)** is a smaller version of the ImageNet dataset (Russakovsky et al., 2015) with 200 classes and 500 instances per class (Wu et al., 2017). It consists of images of size 64×64 pixels.

Exemplar-based clustering. Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, the aim in exemplar-based clustering is to select $k \ll n$ relevant data points $\mathbf{s} \subseteq \mathcal{X}$ that best represent the dataset $\mathcal{X}$. A popular approach is the k-medoids formulation (Kaufman & Rousseeuw, 2009) that aims to minimize the pairwise distances between the selected exemplars and the overall dataset, i.e., minimize $L(\mathbf{s}) := \sum_{\mathbf{u} \in \mathcal{X}} (\min_{\mathbf{x} \in \mathbf{s}} \mathrm{dist}(\mathbf{x}, \mathbf{u}))$. Here, $\mathrm{dist}(\mathbf{x}, \mathbf{u})$ computes the distance between the data points $\mathbf{x}$ and $\mathbf{u}$. We can pose the above as an equivalent monotone (stochastic) submodular maximization problem (Gomes & Krause, 2010) via an auxiliary element $\mathbf{u}_0$: $F(\mathbf{s}) := L(\mathbf{u}_0) - L(\mathbf{s} \cup \{\mathbf{u}_0\})$. In our experiments, we set $\mathrm{dist}(\mathbf{x}, \mathbf{u}) = \|\mathbf{x} - \mathbf{u}\|^2$.

For BSG-NE, BG-NE, and SCG, we experiment with $\nu = \{5, 10, 15\}$. As the error parameter ν increases, these methods require lesser number of samples (BSG-NE and BG-NE) or iterations (SCG). The number of iterations for SCG is upper bounded by 20 000. We set $\delta = 0.001$ for BSG-NE and BG-NE and $\epsilon = 0.01$ for BSG-NE. We experiment with LSG in two settings: $\epsilon = 0.01$ and $\epsilon = 0.1$. For BanditPAM, we show results with $\delta = \{0.01, 0.001, 2k/n\}$. The per iteration policy of BanditPAM is $\delta_i = \delta/k$.



(a) TIN: Time vs k

(b) MNIST: prototype selection skew=10%

(c) MNIST: prototype selection skew=50%

Figure 2: (a) Time versus k plot for the exemplar-based clustering problem on the TIN dataset; (b)&(c) Generalization performance achieved by nearest prototype classifier corresponding to each method on representative sampling problem.

Figure 1 top-row and Figure 1 bottom-row show the results on the MNIST and TIN datasets, respectively. From Figures 1(a) & 1(d), we observe that the proposed BSG-NE and BG-NE algorithms' utility is comparable with LG across k, especially at lower values of error parameter ν. BSG-NE and BG-NE methods achieve the lowest cost across k in both the datasets, i.e., in Figures 1(b) & 1(e), where cost is the number of stochastic function f calls. In exemplar-based clustering, this cost is the number of pairwise distance computations done by an algorithm. At a high k value of 150, we observe in Figures 1(c) & 1(f) that BG-NE with $\nu = 5$ obtains the closest utility to LG and is computationally cheaper than both LG and LSG. On both datasets, across k, the proposed BSG-NE and BG-NE methods show a good trade-off between utility and cost with varying error parameter ν. LSG's cost, on the other hand, varies little across k and is much costlier than the proposed approaches at low k values. SCG obtains the worst utility across k on the MNIST dataset and is lower than the proposed approaches on the TIN dataset. BanditPAM, on the other hand, matches LG's utility at low k values on MNIST, but as k increases, it ends up with utility lower than both BG and BSG algorithms. On the TIN dataset, BanditPAM obtains a competitive utility to BG/BSG (with $\nu = 15$) till $k = 20$ but its utility almost flattens for $k > 30$. The computational cost of SCG and BanditPAM are also significantly higher than BSG-NE and BG-NE. In Figures 1(c) & 1(f), we observe that SCG and BanditPAM obtain the worst utility versus cost trade-off on the MNIST and TIN datasets, respectively. In Figure 2(a), we also plot the time taken for different k values on TIN. We observe that BSG-NE and BG-NE take much less time than the baselines, highlighting their practical usefulness.

Representative sampling from a target set. The previous set of experiments show that the proposed algorithms provide a better utility versus cost trade-off compared to the existing approaches. In the above experiments, the quality of the obtained solution is evaluated only in terms of the achieved (submodular) objective. We now evaluate the quality of the proposed approaches in terms of the generalization performance of the obtained solution in a downstream application).

The aim in this application is to select few $k \ll n$ data points from a given *source* set $\mathcal{X}$ that best represents a different given *target* set $\mathcal{Z}$ (Kim et al., 2016). It can also be viewed as an instance of the facility location problem and is formulated as maximizing $\sum_{\mathbf{z} \in \mathcal{Z}} (\max_{\mathbf{x} \in \mathbf{s}_{\mathcal{X}}} w(\mathbf{x}, \mathbf{z}))$, where w computes the similarity between points from $\mathcal{X}$ and $\mathcal{Z}$. Hence, the above representative sampling formulation is an instance of SSM problem. Following (Bien & Tibshirani, 2011; Kim et al., 2016), we evaluate the quality of the selected representative samples (a.k.a. prototypes) via the performance of the corresponding nearest prototype classifier. This is a 1-NN classifier which is parameterized by the selected prototypes.

We evaluate all the methods on the MNIST dataset using the standard experimental protocol (Gurumoorthy et al., 2019; 2021). The source set consists of 5 000 points uniformly sampled from the MNIST test set (of size 10 000). The target set is constructed from the MNIST train set consisting of 60 000 points such that one class

has a skewed $r\%$ representation and others have $(100 - r)/9\%$ representation each. We evaluate the methods in two skew settings: $r = 10$ and $r = 50$. We compare BSG-NE and BG-NE against the baselines LG, LSG, and SCG. BanditPAM is not evaluated as its code is made available only for the k-medoids problem.

The accuracy of the nearest prototype classifier corresponding to each method is plotted in Figures 2(b) & 2(c). We observe that the prototypes selected by the proposed BSG-NE and BG-NE algorithms show good generalization performance and are comparable to LG. This shows that the quality of the solution obtained by the proposed algorithms is similar to LG's solution. However, the prototypes selected by SCG seem to less informative as they obtain $8\% - 15\%$ less accuracy than our approach.

5 Conclusion

The paper takes a fresh look at the stochastic submodular maximization (SSM) problem from a bandit viewpoint. Each iteration of the greedy framework is naturally posed as a best-arm selection (pure exploration) problem. Subsequently, we make use the best-arm identification strategy at each iteration i to provide (ν_i, δ_i)-PAC optimal incremental gain step, where ν_i is the error parameter and δ_i is the confidence parameter. This allows to come out with a set of $\{(\nu_i, \delta_i)\}$ policies which ensure our overall proposed (stochastic) bandit greedy algorithms come with (ν, δ)-PAC guarantees. Within the set of optimal policies, we discuss different ways to optimize $\{(\nu_i, \delta_i)\}$. For example, we obtain expressions for $\{(\nu_i, \delta_i)\}$ that lead to the minimum number of stochastic function f calls. We also derive the expressions for $\{(\nu_i, \delta_i)\}$ that are the best in a given fixed-cost (i.e., the number of stochastic function f calls) setting. Overall, our algorithms offer better utility versus computational cost trade-off than existing SSM approaches in both theory and practice.

References

J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *COLT*, 2010.

V. Bagaria, G. Kamath, V. Ntranos, M. Zhang, and D. Tse. Medoids in almost-linear time via multi-armed bandits. In *AISTATS*, 2018.

Ramakrishna Bairi, Rishabh Iyer, Ganesh Ramakrishnan, and Jeff Bilmes. Summarization of multi-document topic hierarchies using submodular mixtures. In *ACL*, 2015.

Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.

Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *International Conference on Algorithmic Learning Theory*, 2009.

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

Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *SIGKDD international conference on Knowledge discovery and data mining*, 2009.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7(39):1079–1105, 2006.

R. Gomes and A. Krause. Budgeted nonparametric learning from data streams. In *ICML*, 2010.

Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi, and Charu Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 260–269. IEEE, 2019.

Karthik S. Gurumoorthy, Pratik Jawanpuria, and Bamdev Mishra. SPOT: A framework for selection of prototypes using optimal transport. In *ECML*, pp. 535–551, 2021.

H. Hassani, M. Soltanolkotabi, and A. Karbasi. Gradient methods for submodular maximization. In *NeurIPS*, 2017.

H. Hassani, A. Karbasi, A. Mokhtari, and Z. Shen. Stochastic continuous greedy ++: When upper and lower bounds match. In *NeurIPS*, 2019.

H. Hassani, A. Karbasi, A. Mokhtari, and Z. Shen. Stochastic conditional gradient++: (non)convex minimization and continuous submodular maximization. *SIAM Journal on Optimization*, 30(4):3315–3344, 2020.

A. Hassidim, R. Kupfer, and Y. Singer. An optimal elimination algorithm for learning a best arm. In *NeurIPS*, 2020.

K. Jamieson and R. Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Conference on Information Sciences and Systems (CISS)*, 2014.

M. R. Karimi, M. Lucic, H. Hassani, and A. Krause. Stochastic submodular maximization: The case of coverage functions. In *NeurIPS*, 2017.

Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., 2009.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3(71–104):3, 2014.

T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, 2007.

Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17473–17487. Curran Associates, Inc., 2022.

Anay Mehrotra and Nisheeth K Vishnoi. Maximizing submodular functions for recommendation in the presence of biases. In *ACM Web Conference*, 2023.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *JMLR*, 6(38):1099–1125, 2005.

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, 1978.

B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *AAAI*, 2015.

A. Mokhtari, H. Hassani, and A. Karbasi. Conditional gradient method for stochastic submodular maximization: Closing the gap. In *AISTATS*, 2018.

A. Mokhtari, H. Hassani, and A. Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *JMLR*, 21(105):1–49, 2020.

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

Arghya Roy Chaudhuri, Pratik Jawanpuria, and Bamdev Mishra. ProtoBandit: Efficient prototype selection via multi-armed bandits. In *ACML*, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Y. Sun, J. Gao, X. Hong, B. Mishra, and B. Yin. Heterogeneous tensor decomposition for clustering via manifold optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):476–489, 2016. doi: 10.1109/TPAMI.2015.2465901.

Mo Tiwari, Martin J Zhang, James Mayclin, Sebastian Thrun, Chris Piech, and Ilan Shomorony. Banditpam: Almost linear time k-medoids clustering via multi-armed bandits. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10211–10222, 2020.

Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems*, 2014.

J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008.

K. Wei, R. Iyer, and J. Bilmes. Fast multi-stage submodular maximization. In *ICML*, 2014.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pp. 1954–1963. PMLR, 2015.

Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imagenet challenge. *Technical report*, 2017.

Martin Zhang, James Zou, and David Tse. Adaptive Monte Carlo multiple testing via multi-armed bandits. In *ICML*, 2019.

Appendix

A Organization

The appendix is organized as follows.

1. Section B gives the proofs to the theorems and lemmas mentioned in the main paper.
2. Section C discusses the theorems and the proofs related to the correctness of the proposed BG algorithm.
3. Additional discussion on the algorithms is presented in Section E.
4. The experimental details are in Section F.
5. The bandit algorithms for PAC_BEST_ARM in Algorithm 1 are in Section G.

B Proofs related to the BSG algorithm

B.1 Proof of Theorem 3.1

We first note the following lemma which adapts (Mirzasoleiman et al., 2015, Lemma 2) to the (ν, δ)-PAC best arm identification setting and then use this result to prove Theorem 3.1.

Lemma B.1. *Given a current solution* $\mathbf{s}$ *of BSG, let* (ν_0, δ_i) *be the PAC parameters for the next iteration. Then, the following holds with probability* $1-\delta_i$*: the expected gain of BSG (Algorithm 1) in one iteration is at least* $\frac{(1-\epsilon)}{k}\sum_{s\in\mathbf{s}^*\setminus\mathbf{s}}(g(s|\mathbf{s})-\nu_0)$ *where* $\mathbf{s}^*$ *denote the optimal solution of problem (2).*

Proof. Let $\mathcal{R}$ denote the set of candidate elements in the current iteration. We first estimate the probability that $\mathcal{R}\cap(\mathbf{s}^*\setminus\mathbf{s})$ is non-empty. The set $\mathcal{R}$ has $r=\frac{n}{k}\log(\frac{1}{\epsilon})$ randomly sampled elements from set $\mathcal{S}\setminus\mathbf{s}$ (w.l.o.g. with repetition). Hence,

$$\begin{aligned}
\mathbb{P}[\mathcal{R}\cap(\mathbf{s}^*\setminus\mathbf{s})\neq\phi] &= 1-\mathbb{P}[\mathcal{R}\cap(\mathbf{s}^*\setminus\mathbf{s})=\phi] \\
&= 1-\left(1-\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{|\mathcal{S}\setminus\mathbf{s}|}\right)^r \\
&\geq 1-e^{-r\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{|\mathcal{S}\setminus\mathbf{s}|}} && (\because 1-x\leq e^{-x}) \\
&\geq 1-e^{-\frac{r}{n}|\mathbf{s}^*\setminus\mathbf{s}|} && (\because |\mathcal{S}\setminus\mathbf{s}|\leq n).
\end{aligned}$$

Since $1-e^x$ is a concave function and $|\mathbf{s}^*\setminus\mathbf{s}|/k\leq 1$, we have

$$1-e^{-\frac{r}{n}\left((1-\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k})0+(\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k})k\right)} \geq ((1-\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k}))(1-e^{-\frac{r}{n}0})+(\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k})(1-e^{-\frac{r}{n}k}),$$

which reduces to

$$1-e^{-\frac{r}{n}|\mathbf{s}^*\setminus\mathbf{s}|}\geq\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k}(1-e^{-\frac{rk}{n}}).$$

Therefore, we have

$$\mathbb{P}[\mathcal{R}\cap(\mathbf{s}^*\setminus\mathbf{s})\neq\phi]\geq 1-e^{-\frac{r}{n}|\mathbf{s}^*\setminus\mathbf{s}|}\geq\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k}(1-e^{-\frac{rk}{n}})=(1-\epsilon)\frac{|\mathbf{s}^*\setminus\mathbf{s}|}{k}. \tag{10}$$

In a given iteration, BSG's step 2 (`PAC_BEST_ARM`) selects a ν_0-optimal element s with probability $1-\delta_i$. That is,

$$g(s|\mathbf{s}) \geq \max_a g(a|\mathbf{s}) - \nu_0. \tag{11}$$

with probability at least $1-\delta_i$. In the following, we omit repeating the phrase "with probability at least $1-\delta_i$" with every result/implication for brevity. Equation (11) implies

$$g(s|\mathbf{s}) \geq g(s|\mathbf{s}) - \nu_0 \ \forall s \in \mathcal{R} \cap (\mathbf{s}^* \setminus \mathbf{s}) \text{ (if non-empty) with probability } 1-\delta_i.$$

We note that since $\mathcal{R}$ is equally likely to contain any element of $\mathbf{s}^* \setminus \mathbf{s}$, a uniformly random element of $\mathcal{R} \cap (\mathbf{s}^* \setminus \mathbf{s})$ is a uniformly random element of $\mathbf{s}^* \setminus \mathbf{s}$. Taking expectation on both sides of the inequality in (11) and using the above observation, we get

$$\mathbb{E}[g(s|\mathbf{s})] \geq \mathbb{P}[\mathcal{R} \cap (\mathbf{s}^* \setminus \mathbf{s}) \neq \phi] \left(\frac{1}{|\mathbf{s}^* \setminus \mathbf{s}|} \sum_{s \in \mathbf{s}^* \setminus \mathbf{s}} (g(s|\mathbf{s}) - \nu_0) \right)$$

The above inequality holds as the expectation is being taken over the events $\mathbb{P}[\mathcal{R} \cap (\mathbf{s}^* \setminus \mathbf{s}) \neq \phi]$ and $\mathbb{P}[\mathcal{R} \cap (\mathbf{s}^* \setminus \mathbf{s}) = \phi]$ and the product of marginal gain and the latter event is non-negative. By substituting (10) in the right hand side of the above inequality, we get that the following holds with probability at least $1-\delta_i$

$$\mathbb{E}[g(s|\mathbf{s})] \geq \frac{1-\epsilon}{k} \sum_{s \in \mathbf{s}^* \setminus \mathbf{s}} (g(s|\mathbf{s}) - \nu_0).$$

This completes the proof of Lemma B.1. □

Now, the proof of Theorem 3.1 begins.

Let $\mathbf{s}_i = \{s_1, \ldots, s_i\}$ be the solution of BSG after i iterations. Let (ν_i, δ_i) be a feasible per-iteration policy for BSG such that employing it ensures that BSG learns a (ν, δ)-PAC solution. From Lemma B.1, we have the following result:

$$\mathbb{E}[g(s_{i+1}|\mathbf{s}_i)|\mathbf{s}_i] \geq \frac{1-\epsilon}{k} \sum_{s \in \mathbf{s}^* \setminus \mathbf{s}_i} (g(s|\mathbf{s}_i) - \nu_{i+1}), \tag{12}$$

which holds with probability at least $1-\delta_i$. By submodularity, we have the following result: $\sum_{s \in \mathbf{s}^* \setminus \mathbf{s}} g(s|\mathbf{s}_i) \geq g(\mathbf{s}^*|\mathbf{s}_i) \geq F(\mathbf{s}^*) - F(\mathbf{s}_i)$. Applying this in the above result gives

$$\mathbb{E}[g(s_{i+1}|\mathbf{s}_i)|\mathbf{s}_i] \geq \frac{1-\epsilon}{k}(F(\mathbf{s}^*) - F(\mathbf{s}_i)) - (1-\epsilon)\nu_{i+1}.$$

Now we take expectation over $\mathbf{s}_i$ over both the sides and using the law of total expectation, we have

$$\mathbb{E}[g(s_{i+1}|\mathbf{s}_i)] = \mathbb{E}[F(\mathbf{s}_{i+1}) - F(\mathbf{s}_i)] \geq \frac{1-\epsilon}{k}\mathbb{E}[F(\mathbf{s}^*) - F(\mathbf{s}_i)] - (1-\epsilon)\nu_{i+1}.$$

Since $\mathbb{E}[F(\mathbf{s}^*)] = F(\mathbf{s}^*)$, we get

$$(F(\mathbf{s}^*) - \mathbb{E}[F(\mathbf{s}_i)]) - (F(\mathbf{s}^*) - \mathbb{E}[F(\mathbf{s}_{i+1})]) \geq \frac{1-\epsilon}{k}(F(\mathbf{s}^*) - \mathbb{E}[F(\mathbf{s}_i)]) - (1-\epsilon)\nu_{i+1}.$$

Taking $B_{i+1} = F(\mathbf{s}^*) - \mathbb{E}[F(\mathbf{s}_{i+1})]$ and $\beta = (1 - \frac{1-\epsilon}{k})$ we get

$$B_{i+1} \leq \beta B_i + (1-\epsilon)\nu_{i+1}.$$

By induction, we have

$$B_k \leq \beta^k B_0 + (1-\epsilon) \sum_{i=1}^{k} \beta^{k-i} \nu_i. \tag{13}$$

If $(1-\epsilon)\sum_{i=1}^{k}\beta^{k-i}\nu_i \leq \nu$, the inequality (13) implies

$$\mathbb{E}[F(\mathbf{s}_k)] \geq (1-e^{-1}-\epsilon)F(\mathbf{s}^*) - \nu, \tag{14}$$

where we also use the relation that $(1-(1-\frac{1-\epsilon}{k})^k) \geq (1-e^{-(1-\epsilon)}) \geq (1-e^{-1}-\epsilon)$.

Thus, the result (14) is true when both the following conditions hold:

1. $(1-\epsilon)\sum_{i=1}^{k}\beta^{k-i}\nu_i \leq \nu$
2. the result (12) is true for every iteration of BSG (Algorithm 1)

However, (12) depends on the validity of (11). Now, BSG's step 2 returns a ν_i-optimal solution with probability $1-\delta_i$. The probability that BSG's step 2 does not return ν_i-optimal solution in any iteration $i \in \{1,\ldots,k\}$ is $\sum_i \delta_i = \delta$. Hence, the probability that BSG's step 2 return ν_i-optimal solution in every iteration i for $i = 1,\ldots,k$ is $1-\delta$. Therefor, we prove that (14) is true with probability $1-\delta$ if $(1-\epsilon)\sum_{i=1}^{k}\beta^{k-i}\nu_i \leq \nu$ and $\sum_i \delta_i = \delta$, which completes the proof.

B.2 Proof of Lemma 3.3

The i-th iteration of BSG-ABA calls the ABA algorithm once to obtain a ν_i, δ_i-PAC solution. For n arms, the number of samples $\mathbf{z}$ required by ABA algorithm (Hassidim et al., 2020, Theorem 1) to select a ν_i-best arm with probability at least $1-\delta_i$ is at most $\frac{18n}{\nu^2}\log(\frac{1}{\delta})$. One stochastic function f call is required for every sample $\mathbf{z}$. The candidate set $\mathcal{R}_i$ in each iteration i of BSG-ABA has $|\mathcal{R}_i| = \frac{n}{k}\log(\frac{1}{\epsilon})$ elements. Hence, total number of function calls required by BSG-ABA across k iterations is upper bounded by

$$N_{\text{ABA}} = \sum_{i=1}^{k}\frac{18|\mathcal{R}_i|}{\nu_i^2}\log\left(\frac{1}{\delta_i}\right) = \sum_{i=1}^{k}\frac{18n}{k\nu_i^2}\log\left(\frac{1}{\delta_i}\right)\log\left(\frac{1}{\epsilon}\right).$$

For n arms, the number of samples $\mathbf{z}$ required by the NE algorithm to select a ν_i-best arm with probability at least $1-\delta_i$ is $\frac{2n}{\nu^2}\log(\frac{n}{\delta})$. This well-known result can be proved using the Hoeffding bound (Hassidim et al., 2020, Appendix A). The i-th iteration of BSG-NE calls the NE algorithm once to obtain a ν_i, δ_i-PAC solution. Hence, the total number of function calls required by BSG-NE across k iterations is upper bounded by

$$N_{\text{NE}} = \sum_{i=1}^{k}\frac{2|\mathcal{R}_i|}{\nu_i^2}\log\left(\frac{|\mathcal{R}_i|}{\delta_i}\right) = \sum_{i=1}^{k}\frac{2n}{k\nu_i^2}\log\left(\frac{n}{k\delta_i}\log\left(\frac{1}{\epsilon}\right)\right)\log\left(\frac{1}{\epsilon}\right).$$

Both the upper bounds are tight as equality holds for $k = 1$.

B.3 Proof of Theorem 3.4

We first state the following result.

Lemma B.2. *Let $r > 0$, $p = 1 + \frac{1}{r}$, $a_i > 0$ $\forall i = 1,\ldots,k$. Consider the below optimization problem:*

$$\min_{x_j>0, j=1,\ldots,k, \sum_j x_j^r \leq 1} \sum_{i=1}^{k}\frac{a_i^2}{x_i}. \tag{15}$$

The optimal objective of the above problem is $\left(\sum_{i=1}^{k} a_i^{\frac{2}{p}}\right)^p$ which occurs at

$$x_i^* = \left(\frac{a_i^{\frac{2}{p}}}{\sum_j a_j^{\frac{2}{p}}}\right)^{\frac{1}{r}}. \tag{16}$$

Proof. Lemma B.2 can be proved using the Hölder inequality (Micchelli & Pontil, 2005, Lemma 26). □

To prove Theorem 3.4, we make use Lemma B.2 and use the substitution

$$\sqrt{x_i} = (1-\epsilon)\beta^{k-i}\nu_i/\nu.$$

This leads to $1/\nu_i^2 = a_i^2/x_i$, $a_i = (1-\epsilon)\beta^{k-i}/\nu$, $r = 1/2$, and $p = 3$ and ensures that the optimization problem (8) is same as (15). Finally, using (16), we have

$$\sqrt{x_i^*} = a_i^{2/3}/\sum_j a_j^{2/3} = \beta^{2(k-i)/3}/\sum_j \beta^{2(k-j)/3} = \frac{\beta^{2(k-i)/3}(1-\beta^{2/3})}{(1-\beta^{2k/3})}$$
$$\Rightarrow \nu_i^* = \frac{\nu(1-\beta^{2/3})\beta^{(i-k)/3}}{(1-\epsilon)(1-\beta^{2k/3})},$$

plugging the values back in N_{ABA} and N_{NE} completes the proof of Theorem 3.4.

B.4 More details on $c(k) \leq k^2$ in Remark 3.5

The expression of $c(k)$ is $c(k) = \frac{(1-\beta^{2k/3})^3}{k(1-\beta^{2/3})^3}$. We consider the function

$$c_1(k) = (c(k)/k^2)^{1/3} = \frac{(1-\beta^{2k/3})}{k(1-\beta^{2/3})}.$$

It is easy to see that if $c_1(k) < 1$, it implies that $c(k) \leq k^2$. Hence, we aim to prove $c_1(k) < 1$ in the following way.

- The binomial theorem for fractional exponent is given by $(1+x)^{\frac{p}{q}} = 1 + \frac{p}{q}x + \frac{\frac{p}{q}(\frac{p}{q}-1)}{2!}x^2 + \ldots = \sum_{k=0}^{\infty} \binom{p/q}{k} x^k$. Hence,

$$k(1-\frac{1-\epsilon}{k})^{\frac{2}{3}} = k\left[1 - \left(1 - \frac{2}{3}\frac{1-\epsilon}{k} - \frac{2}{3}\frac{1}{3}\frac{1}{2!}\frac{(1-\epsilon)^2}{k^2} - \ldots\right)\right] \geq \frac{2}{3}(1-\epsilon). \tag{17}$$

  We also note that $\beta^{2k/3} = \left(1 - \frac{1-\epsilon}{k}\right)^{2k/3}$ is an increasing function of k, bounded above by $e^{2\frac{2}{3}(1-\epsilon)}$. Hence, $c_2(k) = (1-\beta^{2k/3})$ is a decreasing function of k. We note that $k = 3$, $c_2(3) = 1 - \beta^{2k/3} = 1 - \beta^2 = (1 - 1 - \frac{(1-\epsilon)^2}{9} + \frac{2}{3}(1-\epsilon)) \leq \frac{2}{3}(1-\epsilon)$. Thus, we have shown that for $k \geq 3$,

$$c_2(k) \leq c_2(3) \leq \frac{2}{3}(1-\epsilon). \tag{18}$$

  Using (17) and (18), we have that for $k \geq 3$, $c_1(k) \leq 1 \Rightarrow c(k) \leq k^2$.

- For $k = 1$, $\beta = \epsilon$ and we get $c(1) = 1 = k^2$.
- For $k = 2$, $\beta = (1+\epsilon)/2$ and we get $c_1(2) = \frac{1-\beta^{4/3}}{2(1-\beta^{2/3})} = \frac{1+\beta^{2/3}}{2} \leq 1 \Rightarrow c(2) \leq 2^2$.

B.5 Proof of Theorem 3.8

The proof of this result is similar to that of Theorem 3.4. We simply need to use Lemma B.2 after doing the following substitution:

$$x_i = \frac{1}{\nu_i}\sqrt{\frac{b}{N_0}}, \tag{19}$$

where $b = \frac{18n}{k}\log(\frac{k}{\delta})\log(\frac{1}{\epsilon})$ for BSG-ABA and $b = \frac{2n}{k}\log(\frac{n}{\delta}\log(\frac{1}{\epsilon}))\log(\frac{1}{\epsilon})$ for BSG-NE. The substitution in (19) also implies $a_i^2 = (1-\epsilon)\beta^{k-i}\sqrt{\frac{b}{N_0}}$ in Lemma B.2.

C Theorems and proofs related to the proposed BG algorithm

In this section, we re-state the theorems in the main paper for the proposed BG algorithm. The proofs follow the proof strategy of Section B.

Lemma C.1. *Given a current solution* $\mathbf{s}$ *of BG and let* (ν_0, δ_i) *be the PAC parameters for the next iteration. Then, the following holds with probability* $1-\delta_i$*: the gain of BG (Algorithm 1) in one iteration is at least* $\frac{1}{k}\sum_{s\in\mathbf{s}^*\setminus\mathbf{s}}(g(s|\mathbf{s})-\nu_0)$ *where* $\mathbf{s}^*$ *denote the optimal solution of problem (2).*

Proof. The proof is similar to the proof of Lemma B.1 except that $\mathcal{R}$ is the entire set $\mathcal{S}\setminus\mathbf{s}$. In the given iteration, BG's step 2 (`PAC_BEST_ARM`) selects a ν_0-optimal element s with probability $1-\delta_i$. That is,

$$g(s|\mathbf{s}) \geq \max_a g(a|\mathbf{s}) - \nu_0 \geq \frac{1}{|\mathbf{s}^*\setminus\mathbf{s}|}\sum_{s\in\mathbf{s}^*\setminus\mathbf{s}}(g(s|\mathbf{s})-\nu_0) \geq \frac{1}{k}\sum_{s\in\mathbf{s}^*\setminus\mathbf{s}}(g(s|\mathbf{s})-\nu_0).$$

with probability at least $1-\delta_i$. This completes the poof of the lemma. □

Theorem C.2. *Let* $\mathbf{s}$ *be the solution obtained by BG (BG-ABA or BG-NE) for (2) with given PAC parameters* (ν,δ)*. Then,* $\mathbf{s}$ *is a* (ν,δ)*-PAC solution of (2) if per-iteration* (ν_i,δ_i) *policy is defined as*

$$\textstyle\sum_{i=1}^k \delta_i = \delta \quad \text{and} \quad \sum_{i=1}^k \alpha^{k-i}\nu_i \leq \nu,$$

where $\alpha = 1-1/k$*.*

Proof. The proof follows the proof strategy of Theorem 3.1 and relies on Lemma C.1. The inequality corresponding to (13) is

$$A_{i+1} \leq \alpha A_i + \nu_{i+1},$$

where $A_{i+1} = F(\mathbf{s}^*) - F(\mathbf{s}_{i+1})$ and $\alpha = 1-1/k$. Summing it over from $i=0$ to $i=k-1$, we get $A_k \leq \alpha^k A_0 + \sum_{i=1}^k \alpha^{k-i}\nu_i$. Finally, we have the inequality $F(\mathbf{s}_k) \geq (1-e^{-1})F(\mathbf{s}^*) - \nu$ with $\sum_{i=1}^k \alpha^{k-i}\nu_i \leq \nu$. □

Lemma C.3. *Let* N_{ABA} *and* N_{NE} *be the upper bounds on the number of stochastic function* f *evaluations required by BG-ABA and BG-NE (Algorithm 1), respectively. Then,*

$$\begin{aligned} N_{\text{ABA}} &= \textstyle\sum_{i=1}^k \frac{18n}{\nu_i^2}\log\left(\frac{1}{\delta_i}\right), \\ N_{\text{NE}} &= \textstyle\sum_{i=1}^k \frac{2n}{\nu_i^2}\log\left(\frac{n}{\delta_i}\right). \end{aligned} \tag{20}$$

Proof. The proof is similar to the proof of Lemma 3.3 except $\mathcal{R}_i$ is all the remaining candidates set and $|\mathcal{R}_i|$ is $n-i$ whose upper bound is n that is used in the expressions. □

Theorem C.4. *Let* (ν_i^*, δ_i^*) *be an optimal per-iteration policy corresponding to the minimum number of stochastic function* f *calls required by BG to obtain a* (ν,δ)*-PAC solution. Then,*

$$\nu_i^* = \tfrac{\nu(1-\alpha^{2/3})\alpha^{(i-k)/3}}{(1-\alpha^{2k/3})}$$

and $\delta_i^* = \delta/k$*. The expressions of* N_{ABA} *and* N_{NE} *corresponding to* (ν_i^*, δ_i^*) *are*

$$\begin{aligned} N_{\text{ABA}}^* &= \tfrac{18nk}{\nu^2}\log\left(\tfrac{k}{\delta}\right)c(k) \text{ and} \\ N_{\text{NE}}^* &= \tfrac{2nk}{\nu^2}\log\left(\tfrac{nk}{\delta}\right)c(k), \end{aligned}$$

where $c(k) = \frac{(1-\alpha^{2k/3})^3}{k(1-\alpha^{2/3})^3}$ *and* $\alpha = 1-1/k$*.*

Proof. We follow the proof of Theorem 3.4, make use Lemma B.2, and use the substitution

$$\sqrt{x_i} = \alpha^{k-i}\nu_i/\nu.$$

□

Theorem C.5. *Given a fixed budget N_0 (the maximum number of stochastic function f calls allowed) and confidence δ, BG-ABA and BG-NE achieve $(\nu^*_{\text{ABA}}, \delta)$- and $(\nu^*_{\text{NE}}, \delta)$-PAC solutions, respectively, where*

$$\nu^*_{\text{ABA}} = \left(\frac{1-\alpha^{2k/3}}{1-\alpha^{2/3}}\right)^{3/2} \left(\frac{18n\log(k/\delta)}{N_0}\right)^{1/2}$$
$$\nu^*_{\text{NE}} = \left(\frac{1-\alpha^{2k/3}}{1-\alpha^{2/3}}\right)^{3/2} \left(\frac{2n\log(nk/\delta)}{N_0}\right)^{1/2}$$

and $\alpha = 1 - 1/k$. The corresponding per-iteration policy of BSG-ABA is given by $\delta^*_i = \delta/k$ and

$$\nu^*_i = \left(\frac{18n\log(k/\delta)(1-\alpha^{2k/3})}{N_0(1-\alpha^{2/3})}\right)^{1/2} \alpha^{(i-k)/3}.$$

Similarly, the per-iteration policy of BSG-NE, corresponding to $(\nu^*_{\text{NE}}, \delta)$, is given by $\delta^*_i = \delta/k$ and

$$\nu^*_i = \left(\frac{2n\log(nk/\delta)(1-\alpha^{2k/3})}{N_0(1-\alpha^{2/3})}\right)^{1/2} \alpha^{(i-k)/3}.$$

Proof. We need to use Lemma B.2 after doing the following substitution:

$$x_i = \frac{1}{\nu_i}\sqrt{\frac{b}{N_0}},$$

where $b = 18n\log(\frac{k}{\delta})$ for BG-ABA and $b = 2n\log(\frac{nk}{\delta})$ for BG-NE. The above substitution implies $a_i^2 = \alpha^{k-i}\sqrt{\frac{b}{N_0}}$ in Lemma B.2. □

D On optimal $\{(\nu_i, \delta_i)\}$ policy

In the main sections, we have primarily focused on optimizing the $\{\nu_i\}$ parameters while δ_i is set to δ/k for all i. Here, we show how to optimize for both $\{\nu_i\}$ and $\{\delta_i\}$. Below, we discuss the developments for BSG-ABA. For other BSG and BG variants, a similar approach can be followed.

Fixed (ν, δ) setting. Given (ν, δ), the resulting optimization problem is

$$\begin{array}{ll} \min\limits_{\nu_i, \delta_i > 0 \forall i} & \sum_{i=1}^k -\log(\delta_i)\nu_i^{-2} \\ \text{s.t.} & (1-\epsilon)\sum_{i=1}^k \beta^{k-i}\nu_i \leq \nu \\ & \sum_{i=1}^k \delta_i \leq \delta. \end{array} \tag{21}$$

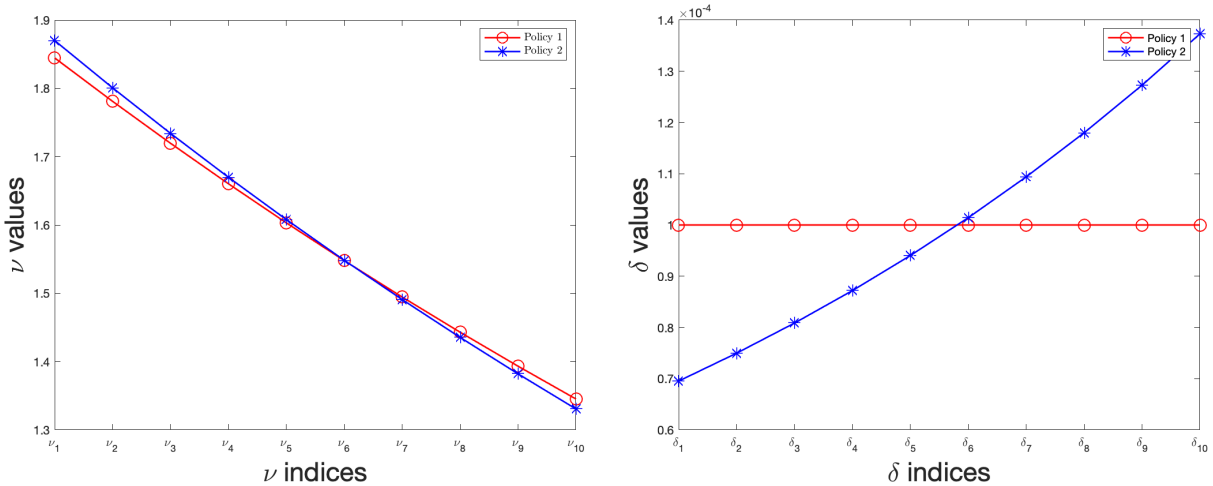
Problem (21) is not jointly convex in $\{\nu_i\}$ and $\{\delta_i\}$ but is individually strongly convex (with simple closed-form solutions). This allows to propose an alternating minimization algorithm efficiently to solve (21). Figure 3 shows the values and the trade-offs obtained by two different policies: one from Theorem 3.4 and the other by solving the problem (21). Although the joint optimization approach leads to a smaller computational cost (in terms of the number of stochastic function calls) than the one proposed in Theorem 3.4, the improvements are only around 0.23%. This also justifies that our approach to only optimize the ν_i parameters (while fixing $\delta_i = \delta/k$) in Section 3.3.

Fixed cost setting. Given a fixed computational cost, the optimization problem that needs to be solved is as follows:

$$\begin{array}{ll} \min\limits_{\nu_i, \delta_i > 0 \forall i} & \sum_{i=1}^k \beta^{k-i}\nu_i \\ \text{s.t.} & \sum_{i=1}^k -\log(\delta_i)\nu_i^{-2} \leq c \\ & \sum_{i=1}^k \delta_i \leq \delta, \end{array} \tag{22}$$

where $c > 0$ and $0 < \delta < 1$ are given. Using the substitution $\mu_i = -\log(\delta_i)\nu_i^{-2} > 0$, the problem (22) is equivalent to

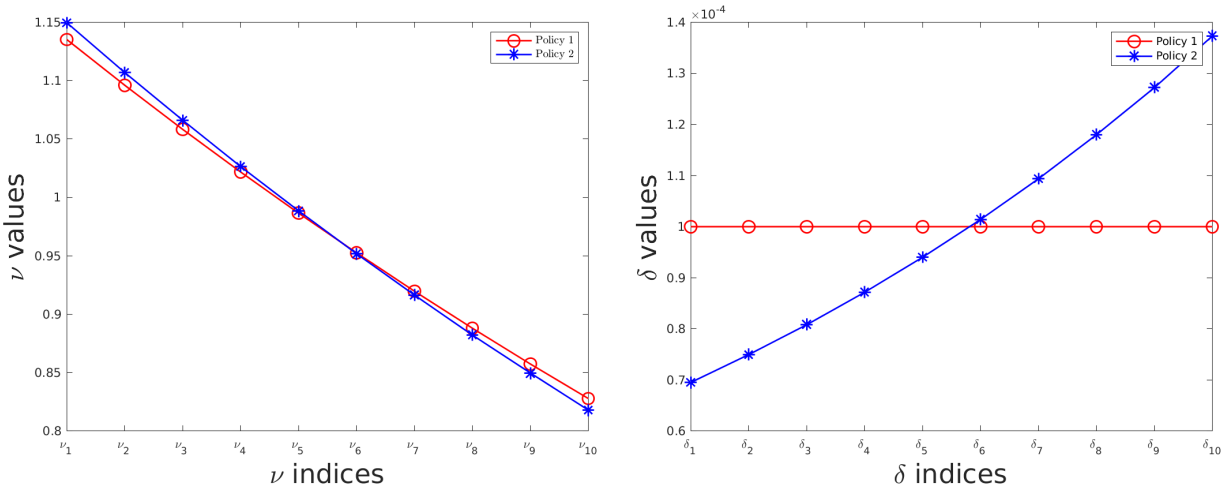
$$\begin{array}{ll} \min\limits_{\mu_i, \delta_i > 0 \forall i} & \sum_{i=1}^k \beta^{k-i}\sqrt{-\log(\delta_i)/\mu_i} \\ \text{s.t.} & \sum_{i=1}^k \mu_i \leq c \\ & \sum_{i=1}^k \delta_i \leq \delta. \end{array} \tag{23}$$



(a) $\{\nu_i\}$ values (b) $\{\delta_i\}$ values

Figure 3: We plot the values of $\{\nu_i\}$ and $\{\delta_i\}$ for two different policies in the fixed (ν, δ) setting. The problem instance has $k = 10$, $\nu = 10$, $\delta = 10^{-3}$, and $\epsilon = 10^{-3}$. Policy 1 refers to the values proposed in Theorem 3.4 and $\delta_i = \delta/k = 10^{-4}$. Policy 2 refers to the solutions obtained by solving the problem (21) which optimizes both $\{\nu_i\}$ and $\{\delta_i\}$.

Problem (23) is non convex and we can resort to an iterative algorithm scheme to empirically solve it. In particular, we make use of manifold optimization on the multinomial manifold (Sun et al., 2016) using the Manopt package (Boumal et al., 2014). Figure 4 shows the values of $\{\nu_i\}$ and $\{\delta_i\}$ obtained by solving the problem (22). Although we obtain a smaller $\nu = \sum_{i=1}^{k} \beta^{k-i}\nu_i$ using joint optimization of $\{\nu_i\}$ and $\{\delta_i\}$ than that from Theorem 3.8, the improvement is around 0.17% for the problem setting in Figure 4.



(a) $\{\nu_i\}$ values (b) $\{\delta_i\}$ values

Figure 4: We plot the values of $\{\nu_i\}$ and $\{\delta_i\}$ for two different policies in the fixed cost setting. The problem instance has $k = 10$, $\delta = 10^{-3}$, $\epsilon = 10^{-3}$, and $c = 100$. Policy 1 refers to the values proposed in Theorem 3.8 and $\delta_i = \delta/k = 10^{-4}$. Policy 2 refers to the solutions obtained by solving the problem (22) which optimizes both $\{\nu_i\}$ and $\{\delta_i\}$.

In the above setting, we are given the confidence parameter δ along with the cost c. However, it might be interesting to look at how the optimized ν (i.e., the argmin solution to (22)) varies with different choices of δ but fixed c. Figure 5 shows the (ν, δ) Pareto optimal curve for a given cost c. Here, $\delta \in \{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.2, 0.3\}$.

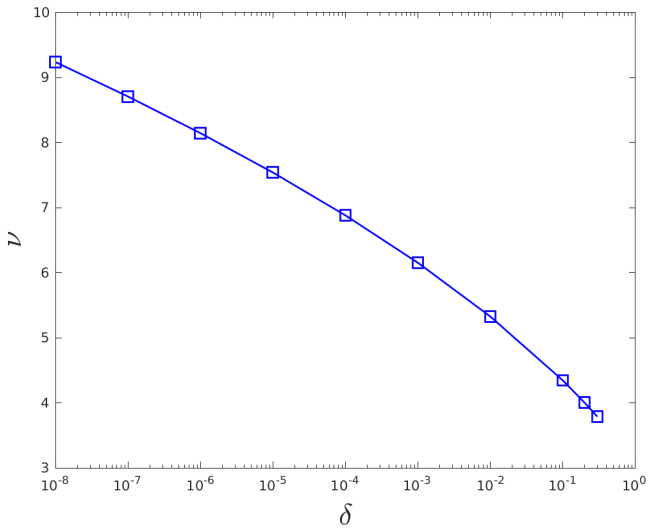


Figure 5: Pareto optimal curve in the fixed cost setting.

E Additional discussion

In this section, we discuss a few other details related to the developments in the main paper.

E.1 On ProtoBandit (Roy Chaudhuri et al., 2022)

The recent work of Roy Chaudhuri et al. (2022) also uses a MAB setting to solve exemplar-based clustering problem. In particular, they also makes use of the ABA algorithm. However, a number of crucial differences exist that are highlighted below.

- The focus in (Roy Chaudhuri et al., 2022) is limited to exemplar-based clustering problem. On the other hand, we have proposed results for general SSM problems of type (2).
- Roy Chaudhuri et al. (2022) do not discuss optimizing the (ν_i, δ_i) policy and use $\nu_i = \nu_0$ (i.e., fixed user-defined ν_i) at every iteration of ProtoBandit. In contrast, we propose how to optimize the per-iteration (ν_i, δ_i) policies in different settings.
- For the exemplar-based clustering problem, the number of pairwise distance computations required by ProtoBandit is $O(nk^3)$, whereas our proposed BSG-ABA algorithm require $O(nk^2)$ pairwise distance computations. Hence, our algorithms show an improvement of a factor k.
- As detailed in (Roy Chaudhuri et al., 2022, Section 3.5), they implement a KL-LUCB algorithm based method with early stopping heuristic for empirical study. However, they do not provide any PAC style analysis of their implemented algorithm.

E.2 Regarding Remark 3.7 on BanditPAM (Tiwari et al., 2020)

For BanditPAM (Tiwari et al., 2020) and the proposed algorithms, the confidence δ mentioned in our draft corresponds to the $1 - \delta$ confidence of the PC guarantee (BanditPAM) and the $1 - \delta$ confidence of the PAC guarantee (proposed algorithms).

The BanditPAM paper mentions a different δ parameter, henceforth referred to as δ_{BP}. The δ_{BP} parameter in BanditPAM's draft (and code) corresponds to $1 - 2n^2\delta_{\mathrm{BP}}$ confidence for every BanditPAM iteration. Tiwari et al. (2020, Remark A1 in the Appendix section) details this. Hence, in terms of the δ_{BP} parameter, BanditPAM guarantees the following across k iterations. Given the parameter δ_{BP}, Tiwari et al. (2020) provide the following guarantee: With probability at least $1 - 2n^2k\delta_{\mathrm{BP}}$, BanditPAM (restricted to the submodular BUILD step) returns the same set of k points as Greedy and the required number of stochastic function f calls (N_{BP}) satisfies $\mathbb{E}[N_{\mathrm{BP}}] = O(n^4 d_1(k)\delta_{\mathrm{BP}} + nd_2(k)\log(1/\delta_{\mathrm{BP}}))$.

E.3 On (Karimi et al., 2017, Proposition 3)

We begin by noting that Karimi et al. (2017) have proposed a multilinear extension based approach for maximizing weighted coverage functions, which is a subclass of the SSM problem (2). As discussed in (Karimi et al., 2017, Section 3), the multilinear extension G of a submodular function F lacks concavity guarantee and may suffer from bad local optima. Thus, they consider concave continuous extensions of G that are efficient to compute and are at most a constant factor away from G to ensure a good quality solution. Weighted coverage functions is one such class of (stochastic) submodular functions where such an extension can be efficiently computed.

For weighted coverage functions, the stochastic function f is submodular. However, as noted in (Mokhtari et al., 2018; 2020), the stochastic function f need not be submodular for SSM problems. Overall, Karimi et al. (2017) propose a concave relaxation scheme and employed projected stochastic gradient ascent (SGA) algorithm. For the SSM problem (2), SGA does not provide tight guarantees as it offers $\mathrm{OPT}/2 - \nu$ lower bound in expectation after $O(n^2k^2/\nu^2)$ iterations (Hassani et al., 2017).

Karimi et al. (2017) have also analyzed a vanilla Hoeffding bound based strategy for maximizing weighted coverage functions. Karimi et al. (2017, Proposition 3) crucially assume the submodularity of the stochastic function f. As discussed above, this assumption is not made in the subsequent SSM literature (Mokhtari et al., 2018; 2020; Hassani et al., 2019; 2020). Following these later works on SSM, we propose an approach for the general stochastic submodular function F in (2) and do not assume submodularity of the stochastic function f.

F Experiment details

Some additional details regarding experimental settings are below.

- For BanditPAM, we use the code from `https://github.com/motiwari/BanditPAM/pull/262` which fixes a critical bug (`https://github.com/motiwari/BanditPAM/issues/252`). In addition, we empirically do not observe much difference in BanditPAM's results with different values of the confidence parameter $\delta = \{0.01, 0.001, 2k/n\}$. In this last setting, the δ is set to different value with different k.

- For SCG, the error parameter ν influences only the number of iterations. In the implementation, we lower bound the number of SCG iterations with 1000 and upper bound it by 20000. With 20000 iterations, SCG takes around 42 minutes on the TIN dataset for $k \geq 90$ and is the slowest baseline in terms of time). Since the number of iterations for SCG with different ν parameters (at high k values) is the same (20000), their plots coincide.

G Bandit algorithms for PAC_BEST_ARM in Algorithm 1

We detail the ABA algorithm (Hassidim et al., 2020) in Algorithm 2. It employs the naive elimination and aggressive elimination strategies, which are detailed in Algorithms 3 and 4, respectively.

Algorithm 2 Approximate best arm (ABA) algorithm

Input: $\mathbf{s}$, $\mathcal{R}$, $\mathcal{Z}$, ν, δ
$\mathbf{t} = \mathcal{R}$.
if $n < \max\{10^5, \delta^{-4}\}$ **then**
  $s \leftarrow \texttt{naiveElimination}(\mathbf{s}, \mathbf{t}, \mathcal{Z}, \nu, \delta)$
else
  $\mathbf{r} \leftarrow \frac{|\mathbf{t}|^{7/8}}{2}$ elements selected uniformly at random from set $\mathcal{R}$.
  $\mathbf{t}_1 \leftarrow \texttt{aggressiveElimination}(\mathbf{s}, \mathbf{t}, \mathcal{Z}, (1-1/e)\nu, \frac{\delta}{2})$
  $s \leftarrow \texttt{naiveElimination}(\mathbf{s}, \mathbf{t}_1 \cup \mathbf{r}, \mathcal{Z}, \frac{\nu}{e}, \frac{\delta}{e})$
end if
Output: s. $\{s$ is a (ν, δ)-best element$\}$

Algorithm 3 Naive elimination algorithm

Input: $\mathbf{s}$, $\mathcal{R}$, $\mathcal{Z}$, ν, δ
$\mathbf{t} = \mathcal{R}$.
Sample a subset $\mathcal{Z}_l \subseteq \mathcal{Z}$ with $l = \left\lceil \frac{2}{\nu^2} \log \frac{|\mathbf{t}|}{\delta} \right\rceil$ samples.
$s = \arg\max_{a \in \mathbf{t}} h(a|\mathbf{s}; \mathcal{Z}_l)$.
Output: s. $\{s$ is a (ν, δ)-best element$\}$

Algorithm 4 Aggressive elimination algorithm

Input: $\mathbf{s}$, $\mathcal{R}$, $\mathcal{Z}$, ν, δ
$\mathbf{t} = \mathcal{R}$.
$\mathbf{t}_0 = \mathbf{t}, t = |\mathbf{t}_0|, \phi(t) = \sqrt{\frac{6 \log t}{t^{3/4}}}, T(t) = \left\lceil \frac{\log t + 4 \log 2}{4 \log \frac{1}{\delta + \phi(t)}} \right\rceil$.
for $\ell \in \{0, 1, 2, \ldots, T(t)\}$ **do**
  Sample a subset $\mathcal{Z}_l \subseteq \mathcal{Z}$ with $l = (\ell+1)\left\lceil \frac{2}{\nu^2} \log \frac{1}{\delta} \right\rceil$ samples .
  For each element $a \in \mathbf{t}_\ell$, compute $h(a|\mathbf{s}; \mathcal{Z}_l)$ and sort the elements in descending order based on the value of $h(a|\mathbf{s}; \mathcal{Z}_l)$
  $\mathbf{t}_{\ell+1} \leftarrow$ the top-$(|\mathbf{t}_\ell| \lfloor \delta + \phi(|t|) \rfloor)$ elements in $\mathbf{t}_\ell$.
end for
Output: the set of elements $\mathbf{t}_{T(t)+1}$.