Faster Guarantees of Evolutionary Algorithms for Maximization of Monotone Submodular Functions

Victoria G. Crawford
University of Florida
vcrawford01@ufl.edu

Abstract

In this paper, the monotone submodular maximization problem (SM) is studied. SM is to find a subset of size κ from a universe of size n that maximizes a monotone submodular objective function f. We show using a novel analysis that the Pareto optimization algorithm achieves a worst-case ratio of $(1-\epsilon)(1-1/e)$ in expectation for every cardinality constraint $\kappa < P$, where $P \leq n+1$ is an input, in $\mathcal{O}(nP\ln(1/\epsilon))$ queries of f. In addition, a novel evolutionary algorithm called the biased Pareto optimization algorithm, is proposed that achieves a worst-case ratio of $(1-\epsilon)(1-1/e-\epsilon)$ in expectation for every cardinality constraint $\kappa < P$ in $\mathcal{O}(n\ln(P)\ln(1/\epsilon))$ queries of f. Further, the biased Pareto optimization algorithm can be modified in order to achieve a worst-case ratio of $(1-\epsilon)(1-1/e-\epsilon)$ in expectation for cardinality constraint κ in $\mathcal{O}(n\ln(1/\epsilon))$ queries of f. An empirical evaluation corroborates our theoretical analysis of the algorithms, as the algorithms exceed the stochastic greedy solution value at roughly when one would expect based upon our analysis.

1 Introduction

A function $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ defined on subsets of a ground set U of size n is monotone submodular if it possesses the following two properties: (i) For all $A \subseteq B \subseteq U$, $f(A) \leq f(B)$ (monotonicity); (ii) For all $A \subseteq B \subseteq U$ and $x \notin B$, $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$ (submodularity). Monotone submodular set functions are found in many applications in machine learning and data mining. Applications of SM include influence in social networks [Kempe *et al.*, 2003], data summarization [Mirzasoleiman *et al.*, 2013], dictionary selection [Das and Kempe, 2011], and monitor placement [Soma and Yoshida, 2016]. As a result, there has been much recent interest in optimization problems involving monotone submodular functions. One such optimization problem is the NP-hard Submodular Maximization Problem (SM), defined as follows.

Problem 1 (Submodular Maximization Problem (SM))**.** *Let $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ be a monotone submodular function defined on subsets of the ground set U of size n, and $f(\emptyset) = 0$. Given a budget $\kappa \in [0, n]$, SM is to find $\text{argmax}_{|X| \leq \kappa} f(X)$.*

An instance of SM is referred to as SM(f, κ). It is assumed that the function f is provided as a value oracle, which when queried with a set X returns the value of $f(X)$. Time is measured in queries of f, as is the convention in submodular optimization [Badanidiyuru *et al.*, 2014].

To approximate SM, the standard greedy algorithm is very effective. Nemhauser and Wolsey [1978] showed that the standard greedy algorithm achieves the best ratio of $(1-1/e)$ for SM in $\mathcal{O}(nk)$ queries to f. In addition, faster versions of the greedy algorithm have been developed for SM [Badanidiyuru *et al.*, 2014; Mirzasoleiman *et al.*, 2015]. In particular, the stochastic greedy algorithm (SG) of Mirzasoleiman *et al.* [2015] achieves ratio $1 - 1/e - \epsilon$ in expectation in $O(n\ln(1/\epsilon))$ queries to f.

Alternatively, one may take the Pareto optimization approach to SM: Instead of maximizing f for a cardinality constraint κ, SM is re-formulated as a bi-objective optimization problem where the goal is to both maximize f as well as minimize cardinality. Instead of a single solution, we seek a pool of solutions none of which dominate another[1]. Greedy algorithms can be used to develop such a pool, however previous works [Friedrich and Neumann, 2014; Qian *et al.*, 2015b] have employed bi-objective evolutionary algorithms because they iteratively improve the entire pool of solutions and can be run indefinitely. The evolutionary algorithm PARETO OPTIMIZATION (PO) has previously been shown to find a $1 - 1/e$ approximate solution to SM(f, κ) for all $\kappa < P$, where $P \leq n+1$ is an input, in expected $\mathcal{O}(nP^2)$ queries to f [Friedrich and Neumann, 2014]. Further, PO has been demonstrated to make significant empirical improvements over the standard greedy algorithms for SM [Qian *et al.*, 2015b]. But as the size of data has grown exponentially in recent times, a query complexity that is cubic in n (for $P = \Omega(n)$) makes these evolutionary algorithms a less attractive option.

1.1 Contributions

In this work, a novel analysis is provided for the algorithm PO, and it is proven that PO achieves a worst-case ratio of

[1] In this context, a solution Y dominates X if $f(X) \leq f(Y)$, $|X| \geq |Y|$, and at least one of the two inequalities is strict.

$(1-\epsilon)(1-1/e)$ in expectation for every instance SM(f,κ) with $\kappa < P$, where $P \leq n+1$ is an input, in $\mathcal{O}(nP\ln(1/\epsilon))$ queries of f. This removes a factor of P from the query complexity of Friedrich and Neumann [2014]. This novel analysis has potential to improve the query complexity of other problems in monotone submodular optimization beyond SM [Qian *et al.*, 2015a; Qian *et al.*, 2017; Crawford, 2019; Bian *et al.*, 2020]. This result is proven in Theorem 1.

Next, a novel algorithm BIASED PARETO OPTIMIZATION (BPO) is proposed that is a similar in spirit but faster version of PO for SM. It is proven that BPO achieves a worst-case ratio of $(1-\epsilon)(1-1/e-\epsilon)$ in expectation for every instance SM(f,κ) with $\kappa < P$ in $\mathcal{O}(n\ln(P)\ln(1/\epsilon))$ queries of f. This result is proven in Theorem 2. Further, a version of BPO for a specific cardinality constraint κ, κ-BIASED PARETO OPTIMIZATION (κ-BPO), is proven to achieve a worst-case ratio of $(1-\epsilon)(1-1/e-\epsilon)$ in expectation for instance SM(f,κ) in $\mathcal{O}(n\ln(1/\epsilon))$ queries of f. This result is proven in Theorem 3. This new algorithm κ-BPO thus matches the optimal SG algorithm in terms of both approximation ratio and query complexity, while maintaining the ability of PO to continuously improve a pool of solutions.

The above theoretical results all extend to the more general setting of monotone γ-weakly submodular[2] functions [Das and Kempe, 2011], but with different approximation guarantees that depend on γ.

An empirical evaluation corroborates our theoretical analysis of the algorithms, as the algorithms exceed the SG solution value at roughly when one would expect based upon our analysis.

1.2 Additional Related Work

Evolutionary algorithms have been studied for many combinatorial optimization problems [Laumanns *et al.*, 2002; Neumann and Wegener, 2007; Friedrich *et al.*, 2010]. In particular, evolutionary algorithms have been analyzed for problems in submodular optimization including SM [Friedrich and Neumann, 2014; Qian *et al.*, 2015b; Roostapour *et al.*, 2019], submodular cover [Qian *et al.*, 2015a; Crawford, 2019], SM with more general cost constraints [Bian *et al.*, 2020], and noisy versions of SM [Qian *et al.*, 2017].

Friedrich and Neumann [2014] studied a slight variant of PO where the pool $\mathcal{S}$ is initialized to contain a random set, and $P = n$. Friedrich and Neumann proved that their variant of PO finds a $1-1/e$ approximate solution to SM(f,κ) in expected $\mathcal{O}(n^2\ln(n)+n^2\kappa)$ queries of f. It is easy to modify their analysis to see that PO finds a $1-1/e$ approximate solution to SM(f,κ) for all $\kappa < P$ in expected $\mathcal{O}(nP^2)$ queries of f. The argument of PO used in the proof of Theorem 1 of Section 2.1 is substantially different compared to the argument of Friedrich and Neumann because it analyzes the expected time until an *expected* approximation ratio is analyzed, resulting in a speedup to $\mathcal{O}(nP)$ queries of f. In addition, the result of Theorem 1 is in deterministic time due to an application of the Chernoff bound.

[2] A function $f : 2^U \to \mathbb{R}_{\geq 0}$ is γ-weakly submodular if for all $X \subseteq Y \subseteq U$, and $u \notin Y$, $\sum_{u\in Y\setminus X} \Delta f(X,u) \geq \gamma\left(f(X\cup Y)-f(X)\right)$. If $\gamma = 1$, then f is submodular.

Qian *et al.* [2015b] considered the subset selection problem, which is a special case of the monotone γ-weakly submodular maximization problem. Qian *et al.* fixed $P = 2\kappa$, and showed that for the cardinality constraint κ PO finds a $1-e^{-\gamma}$ approximate solution in expected $\mathcal{O}(n\kappa^2)$ queries of f. Their results can be generalized beyond subset selection to the monotone γ-weakly submodular maximization problem with cardinality constraint κ.

The algorithm BPO, presented in Section 2.2, uses a novel, biased selection procedure to identify sets for mutation. Because of the biased selection procedure, BPO is the first evolutionary algorithm that has an approximation guarantee in nearly linear queries of f close to that of the greedy algorithm for SM.

2 Algorithms and Theoretical Results

The theoretical contributions of the paper are presented in this section. In particular, a new theoretical analysis of the algorithm PARETO OPTIMIZATION (PO) is presented for SM in Section 2.1, the novel algorithm BIASED PARETO OPTIMIZATION (BPO) is presented and analyzed for SM in Section 2.2, and the faster modification of BPO for a specific cardinality constraint, κ-BIASED PARETO OPTIMIZATION (κ-BPO), is presented and analyzed for SM in Section 2.3. The full version of the paper includes an appendix where additional theoretical details from Section 2 are filled in.

Definitions and Notation The following notation and definitions will be used throughout Section 2. Let $f : 2^U \to \mathbb{R}_{\geq 0}$, $X \subseteq U$, and $x \in U$. (i) Marginal gain: $\Delta f(X,x) = f(X\cup\{x\}) - f(X)$. (ii) The membership of x is *flipped in* X means that if $x \in X$, then x is removed from X; and if $x \notin X$, then x is added to X. (iii) If ν is a random variable, then $\mathbb{E}[\nu]$ denotes the expected value of ν. If A is a random event, then $P(A)$ denotes the probability of A occurring. (iv) Let $\mathcal{S} \subseteq 2^U$. Then if there exists a unique $Y \in \mathcal{S}$ such that $|Y| = i$, define $\mathcal{S}[i] = Y$. If no such Y exists, or there are multiple elements of $\mathcal{S}$ of cardinality i, $\mathcal{S}[i]$ is undefined. (v) Let $X, Y \subseteq U$. Then $X \preceq Y$ if $f(Y) \geq f(X)$ and $|Y| \leq |X|$. If at least one of the two inequalities is strict, then $X \prec Y$ and Y *dominates* X. If $f(Y) = f(X)$ and $|Y| = |X|$ then it is said that X is *equivalent* to Y.

2.1 Pareto Optimization (PO)

In this section, it is proven that in time $\mathcal{O}(nP\ln(1/\epsilon))$, PO produces a $(1-\epsilon)(1-1/e)$ approximate solution in expectation for every cardinality constraint $\kappa < P$, where $P \leq n+1$ is an input. If $P = \mathcal{O}(\kappa)$ for a fixed cardinality constraint κ, then PO produces solutions for SM with similar theoretical guarantees to that of the standard greedy algorithm in the same asymptotic time, which shows the practicality of evolutionary algorithms such as PO for SM.

Description of PO

In this section, PO (Alg. 1) is described. The set $\mathcal{S} \subseteq 2^U$ is referred to as *the pool*, and each iteration of the **for** loop is referred to as *an iteration*. The pool initially contains only the empty set; its maximum size is determined by input parameter P. During each iteration, $i \in \{0,...,P-1\}$ is chosen

Algorithm 1 PO (f, P, T): PARETO OPTIMIZATION (PO)

1: **Input:** $f : 2^U \to \mathbb{R}_{\geq 0}$; $P \in \{1, ..., n+1\}$; $T \in \mathbb{Z}_{\geq 0}$.
2: **Output:** $\mathcal{S} \subseteq 2^U$.
3: $\mathcal{S} \leftarrow \{\emptyset\}$
4: **for** $t \leftarrow 1$ to T **do**
5: $\quad i \leftarrow$ UNIFORM-RANDOM$(\{0, ..., P-1\})$
6: $\quad$ **if** $\mathcal{S}[i]$ exists **then**
7: $\quad\quad B \leftarrow \mathcal{S}[i]$
8: $\quad\quad B' \leftarrow$ MUTATE (B)
9: $\quad\quad$ **if** $|B'| < P$ and $\nexists Y \in \mathcal{S}$ such that $B' \preceq Y$ **then**
10: $\quad\quad\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{B'\} \setminus \{Y \in \mathcal{S} : Y \prec B'\}$
11: $\quad\quad$ **end if**
12: $\quad$ **end if**
13: **end for**
14: **return** $\mathcal{S}$

uniformly randomly (Line 5 of Alg. 1), and if $B = \mathcal{S}[i]$ exists then it is selected from $\mathcal{S}$ to be mutated, otherwise PO continues to the next iteration. The subroutine MUTATE takes B and randomly mutates it into $B' \subseteq U$ as follows: for each $u \in U$, flip the membership of u in B with probability $1/n$. Finally, if no set in the pool dominates or is equivalent to B' and $|B'| < P$, then B' is added to the pool and all sets that B' dominates are removed. There is at most one new query of f on each iteration of PO, and therefore the input T is equal to the query complexity. Pseudocode for the subroutine MUTATE is provided in the appendix.

Analysis of PO for SM

In this section, the approximation result of the algorithm PO for SM is presented. Omitted proofs are given in the appendix. The statement of Theorem 1 easily generalizes to γ-weakly submodular objectives f, where $1 - 1/e$ is replaced with $1 - 1/e^{\gamma}$.

Theorem 1. *Suppose* PO *is run with input (f, P, T), where $f : 2^U \to \mathbb{R}_{\geq 0}$ is monotone submodular, $P \in \{1, ..., n+1\}$, and $T > 8enP \ln(1/\epsilon)$. Let $\mathcal{S}$ be the pool of* PO *at the end of iteration T. Then for any $\kappa < P$,*

$$\mathbb{E}\left[\max_{X \in \mathcal{S}, |X| \leq \kappa} f(X)\right] \geq (1-\epsilon)(1-1/e) \max_{|X| \leq \kappa} f(X).$$

Overview of Proof of Theorem 1 Given SM(f, κ) with optimal solution A^*, the standard greedy algorithm iteratively picks into its solution the element in U of highest marginal gain until κ elements have been picked. Existing analyses of PO for SM [Friedrich and Neumann, 2014; Qian *et al.*, 2015b] analyze the time it takes until essentially the standard greedy algorithm randomly occurs within PO. If instead of iteratively picking the element of highest marginal gain, a uniformly random element of A^* (possibly already chosen) is picked into the solution until κ elements are picked (this could be viewed as an idealized version of the stochastic greedy algorithm of Mirzasoleiman *et al.* [2015]) then the same approximation guarantee as the standard greedy algorithm $(1 - 1/e)$ is achieved in expectation. In the proof of Theorem 1, the expected time until this second algorithm randomly occurs within PO is analyzed, which is a factor of P faster.

Proof. Line numbers referenced are those in Algorithm 1. Throughout the proof of Theorem 1, the probability space of all possible runs of PO with the stated inputs is considered. Let $\kappa < P$, and let $A^* = \arg\max_{|X| \leq \kappa} f(X)$. We may assume that $|A^*| = \kappa$, since f is monotone. In the proof of Theorem 1, the random variable ω will be used, defined inductively as follows:

(i) Before the first iteration of PO, ω is set to 0.
(ii) If the following two conditions are met, ω is incremented at the end of an iteration: 1) $B = \arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \omega\}$ is selected on Line 5; and 2) MUTATE on Line 8 results in the membership of a single element $a^* \in A^*$ being flipped (*i.e.*, it is either the case that MUTATE returns $B' = B \cup \{a^*\}$ for $a^* \in A^* \setminus B$ or $B' = B \setminus \{a^*\}$ for $a^* \in A^* \cap B$).

Intuitively, ω is used to track a solution within $\mathcal{S}$ that has a high f value relative to its cardinality. In particular, the following lemma describes the key property of ω.

Lemma 1. *At the end of every iteration of* PO

$$\mathbb{E}\left[\max_{X \in \mathcal{S}, |X| \leq \omega} f(X)\right] \geq \left(1 - \left(1 - \frac{1}{|A^*|}\right)^{\omega'}\right) f(A^*)$$

where $\omega' = \min\{\omega, P-1\}$.

A further key point is that once a solution appears in PO, i.e., it is returned by MUTATE, there always exists at least as good of a solution within $\mathcal{S}$.

Lemma 2. *Let $Y \subseteq U$ and $|Y| \leq a < P$. If Y is returned by* MUTATE *during iteration i of* PO*, then at the end of any iteration $j \geq i$ it holds that $\max\{f(X) : X \in \mathcal{S}, |X| \leq a\} \geq f(Y)$.*

Let event F be that at the completion of a run of PO, $\omega \geq \kappa$. Then it follows from Lemmas 1 and 2 that

$$\mathbb{E}\left[\max_{X \in \mathcal{S}, |X| \leq \kappa} f(X) | F\right] \geq \left(1 - \frac{1}{e}\right) f(A^*) \qquad (1)$$

Then the remainder of the proof of Theorem 1 is to deal with the probability that ω reaches κ. To this end, the following lemma states that the run of PO may be interpreted as a Bernoulli process.

Lemma 3. *Consider a run of* PO *as a series of Bernoulli trials $Y_1, ..., Y_T$, where each iteration is a trial and a success is defined to be when ω is incremented. Then $Y_1, ..., Y_T$ are independent, identically distributed Bernoulli trials where the probability of success is*

$$\frac{1}{P} \sum_{x \in A^*} \left(1 - \frac{1}{n}\right)^{n-1} \frac{1}{n} \geq \frac{|A^*|}{enP}.$$

Then Lemma 3 and the Chernoff bound can be used to prove that the probability of ω not reaching κ after $T \geq 8enP \ln(1/\epsilon)$ iterations of PO is small. This is stated in the following lemma.

Lemma 4. $P\left(\sum_{i=1}^{T} Y_i < \kappa\right) \leq \epsilon$.

Finally, Theorem 1 follows from the law of total expectation, Inequality 1 and Lemma 4. □

Algorithm 2 BPO $(f, P, T, p, \epsilon, \xi)$: BIASED PARETO OPTIMIZATION

1: **Input:** $f : 2^U \to \mathbb{R}_{\geq 0}$; $P \in \{1, ..., n+1\}$; $T \in \mathbb{Z}_{\geq 0}$; $p \in (0, 1]$; $\epsilon \in (0, 1)$; $\xi \in (0, 1)$.
2: **Output:** $\mathcal{S} \subseteq 2^U$.
3: $M \leftarrow \lceil \ln(P)/\ln(1/\xi) \rceil$
4: $\beta_i \leftarrow 0$, $\ell_i \leftarrow 0$, $H_i \leftarrow e\ln(1/\epsilon)/\xi^i \; \forall i \in \{1, ..., M\}$
5: $\mathcal{S} \leftarrow \{\emptyset\}$
6: **for** $t \leftarrow 1$ to T **do**
7: $\quad i \leftarrow$ UNIFORM-RANDOM$(\{0, ..., P-1\})$
8: $\quad$ **if** FLIP-COIN(p) = heads **then**
9: $\quad\quad j \leftarrow$ UNIFORM-RANDOM$(\{1, ..., M\})$
10: $\quad\quad i \leftarrow |\arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \beta_j\}|$
11: $\quad\quad \ell_j \leftarrow \ell_j + 1$
12: $\quad\quad$ **if** $\ell_j = H_j$ **then**
13: $\quad\quad\quad \ell_j \leftarrow 0$
14: $\quad\quad\quad \beta_j \leftarrow \beta_j + 1$
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: $\quad$ **if** $\mathcal{S}[i]$ exists **then**
18: $\quad\quad B \leftarrow \mathcal{S}[i]$
19: $\quad\quad B' \leftarrow$ MUTATE (B)
20: $\quad\quad$ **if** $|B'| < P$ and $\nexists Y \in \mathcal{S}$ such that $B' \preceq Y$ **then**
21: $\quad\quad\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{B'\} \setminus \{Y \in \mathcal{S} : Y \prec B'\}$
22: $\quad\quad$ **end if**
23: $\quad$ **end if**
24: **end for**
25: **return** $\mathcal{S}$

2.2 Biased Pareto Optimization (BPO)

BIASED PARETO OPTIMIZATION (BPO) is a novel evolutionary algorithm with nearly the same approximation results as PO for SM in faster time. Specifically, it is proven that in time $\mathcal{O}(n\ln(P)\ln(1/\epsilon))$, BPO finds a $(1-\epsilon)(1-1/e-\epsilon)$-approximate solution in expectation for every cardinality constraint $\kappa < P$, where $P \leq n+1$ is an input. Thus, BPO is faster than PO by a factor of $\Omega(P/\ln(P))$; further, it works similarly to PO but has a biased selection procedure instead of choosing uniformly randomly.

Description of BPO

In this section, BPO (Alg. 2) is presented. Pseudocode for BPO can be found in Alg. 2. In overview, BPO follows a similar iterative procedure to PO: every iteration of the **for** loop, a set in the $\mathcal{S}$ is chosen for mutation; and only sets that are not dominated by any others are kept. The difference from PO is in the selection of the set for mutation; a certain subset of sets in $\mathcal{S}$ are selected more frequently than others, as determined by the parameters $p \in (0, 1]$, $\epsilon \in (0, 1)$, and $\xi \in (0, 1)$, and the variables β_j for $j \in \{1, ..., M\}$, where $M = \lceil \ln(P)/\ln(1/\xi) \rceil$. There is at most one new query of f on each iteration of BPO, and therefore the input T is equal to the query complexity. Next, the selection process is described in detail.

Selection process During each iteration, with probability p BPO chooses j from $\{1, ..., M\}$ uniformly randomly (Line 9) and then sets $i = |\arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \beta_j\}|$ (Line 10). Otherwise i is chosen uniformly randomly from $\{0, ..., P-1\}$ (Line 7). If $B = \mathcal{S}[i]$ exists then it is selected from $\mathcal{S}$ to be mutated, otherwise BPO continues to the next iteration. Initially, $\beta_j = 0 \; \forall j \in \{1, ..., M\}$. β_j is incremented to $\beta_j + 1$ if on $H_j = e\ln(1/\epsilon)/\xi^j$ iterations since the last increment of β_j j was chosen on Line 9. The variable ℓ_j is used to determine when β_j should be incremented: β_j is incremented during an iteration if and only if ℓ_j is set to 0 on Line 13. Notice that if $p = 0$, BPO is equivalent to PO.

Analysis of BPO for SM

The approximation results of BPO for SM are now presented. Lemmas referenced in the proof of Theorem 2 can be found in the appendix. The statement of Theorem 2 easily generalizes to γ-weakly submodular objectives f, where $1 - 1/e - \epsilon$ is replaced with $1 - 1/e^\gamma - \epsilon$.

Theorem 2. *Suppose* BPO *is run with input* $(f, P, T, p, \epsilon, \xi)$ *where* $f : 2^U \to \mathbb{R}_{\geq 0}$ *is monotone submodular,* $P \in \{1, ..., n+1\}$, $T \geq \max\{\alpha n\lceil \ln(P)/\ln(1/\xi) \rceil, \beta\ln(n)\lceil \ln(P)/\ln(1/\xi) \rceil\}$, *where* $\alpha = 2e\ln(1/\epsilon)/p$ *and* $\beta = 8/p$, $p \in (0, 1]$, $\epsilon \in (0, 1)$, *and* $\xi \in (0, 1)$. *Let* $\mathcal{S}$ *be the pool of* BPO *at the end of iteration* T. *Then for any* $\kappa < P$,

$$\mathbb{E}\left[\max_{X \in \mathcal{S}, |X| \leq \kappa} f(X)\right] \geq (1-\epsilon)(1-1/e-\epsilon) \max_{|X| \leq \kappa} f(X).$$

Overview of Proof of Theorem 2 Consider SM(f, κ) with optimal solution A^*. Recall that in the proof of Theorem 1 in Section 2.1, the approximation ratio for SM(f, κ) was proven by analyzing the expected time until a variable ω reaches κ. In order for ω to be incremented during an iteration, $|\arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \omega\}|$ must be selected on Line 5, which occurs with probability $1/P$. If we instead consider an alternative version of PO where the selection is biased towards choosing $|\arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \omega\}|$ with constant probability $\alpha > 1/P$, then ω reaches κ faster. The difficulty is that the value of ω is unknown, since it depends on MUTATE flipping the membership of an $a^* \in A^*$ and nothing else. The idea behind BPO is that we can *approximately* track ω, and therefore bias the selection. In particular, for each SM(f, κ) with $\kappa < P$, there exists a β_i that is approximately equal to the corresponding ω for κ.

Proof. Proofs of lemmas used can be found in the appendix. Lines numbers referenced are those in Algorithm 2. Throughout the proof of Theorem 2, the probability space of all possible runs of BPO with the stated inputs is considered. An iteration of the for loop in BPO is simply referred to as an iteration.

Consider any $\kappa < P$. Define $A^* = \arg\max_{|X| \leq \kappa} f(X)$. Without loss of generality we may assume that $|A^*| = \kappa$, since f is monotone. There exists $q \in \{1, ..., \lceil \ln(P)/\ln(1/\xi) \rceil\}$ such that

$$\xi^q P < |A^*| \leq \xi^{q-1} P. \tag{2}$$

Then define $\omega = \beta_q$.

The ω defined here serves a similar purpose to that defined in the proof of Theorem 1; To track a solution within $\mathcal{S}$ that has a high f value relative to its cardinality, as described in the following lemma.

Lemma 5. *At the end of every iteration of* BPO

$$\mathbb{E}\left[\max_{X\in\mathcal{S},|X|\leq\omega} f(X)\right] \geq \left(1-\left(1-\frac{1-\epsilon}{|A^*|}\right)^{\omega'}\right) f(A^*)$$

where $\omega' = \min\{\omega, P-1\}$.

In addition, the property of PO detailed in Lemma 2 of Theorem 1 clearly also holds for BPO.

Define the event F to be that at the completion of a run of BPO ℓ_q has been incremented (Line 11 of Algorithm 2) $H_q\kappa$ times. If ℓ_q has been incremented $H_q\kappa$ times, then one may see that ω has been incremented κ times. Once ω reaches κ, it clearly follows from Lemmas 5 and 2 that

$$\mathbb{E}\left[f(A)|F\right] \geq \left(1-\frac{1}{e}-\epsilon\right) f(A^*) \tag{3}$$

where $A = \text{argmax}_{X\in\mathcal{S},|X|\leq\kappa} f(X)$.

We now analyze the probability that ℓ_q has been incremented $H_q\kappa$ times. To this end, we have the following lemma.

Lemma 6. *Consider a run of* BPO *as a series of Bernoulli trials* $Y_1, ..., Y_T$*, where each iteration is a trial and a success is defined to be when* ℓ_q *is incremented. Then* $Y_1, ..., Y_T$ *are independent, identically distributed Bernoulli trials where the probability of success is* $p/\lceil \ln(P)/\ln(1/\xi)\rceil$.

Finally, an analogous argument to that of Theorem 1 can be used to complete the proof of Theorem 2. In particular, we bound the probability of event F not occurring after $T \geq \max\{\alpha n\lceil \ln(P)/\ln(1/\xi)\rceil, \beta\ln(n)\lceil \ln(P)/\ln(1/\xi)\rceil\}$, where $\alpha = 2e\ln(1/\epsilon)/p$ and $\beta = 8/p$, iterations of BPO by ϵ using the Chernoff bound, and then apply the law of total expectation. The details of the argument can be found in the appendix. □

2.3 κ-Biased Pareto Optimization (κ-BPO)

If a specific cardinality constraint κ is provided, a modified version of BPO, κ-BIASED PARETO OPTIMIZATION (κ-BPO), can produce an approximate solution in expectation even faster than BPO. In this section, the algorithm κ-BPO is described, and it is proven that κ-BPO finds a $(1-\epsilon)(1-1/e-\epsilon)$-approximate solution to SM(f,κ) in $\mathcal{O}(n\ln(1/\epsilon))$ queries of f.

Description of κ-BPO

Pseudocode for κ-BPO can be found in the appendix. κ-BPO is similar to BPO except κ-BPO is only biased towards picking a single element of $\mathcal{S}$, determined by the variable β. The input parameters of κ-BPO are the same as BPO except $\xi \in (0,1)$ is not needed.

During each iteration, with probability p κ-BPO sets $i = |\arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \beta\}|$. Otherwise i is chosen uniformly randomly from $\{0, ..., P-1\}$. If $B = \mathcal{S}[i]$ exists then it is selected from $\mathcal{S}$ to be mutated, otherwise κ-BPO continues to the next iteration. Initially, $\beta = 0$. β is incremented to $\beta+1$ if on $H = en\ln(1/\epsilon)/\kappa$ iterations since the last increment of β, i was chosen to be $|\arg\max\{f(X) : X \in \mathcal{S}, |X| \leq \beta\}|$. The variable ℓ is used to determine when β should be incremented: β is incremented during an iteration if and only if ℓ is set to 0.

Analysis of κ-BPO for SM

The approximation results of κ-BPO for SM are now presented. The statement of Theorem 3 easily generalizes to γ-weakly submodular objectives f in the analogous manner as BPO.

Theorem 3. *Suppose* κ-BPO *is run with input* $(f, \kappa, P, T, p, \epsilon)$ *where* $f : 2^U \to \mathbb{R}_{\geq 0}$ *is monotone submodular,* $\kappa \in \{1, ..., n\}$ $P \in \{\kappa+1, ..., n+1\}$, $T \geq \max\{2en\ln(1/\epsilon)/p, 8\ln(n)/p\}$, $p \in (0,1]$*, and* $\epsilon \in (0,1)$*. Let* $\mathcal{S}$ *be the pool of* κ-BPO *at the end of iteration* T*. Then,*

$$\mathbb{E}\left[\max_{X\in\mathcal{S},|X|\leq\kappa} f(X)\right] \geq (1-\epsilon)(1-1/e-\epsilon)\max_{|X|\leq\kappa} f(X).$$

Proof. The proof of Theorem 3 is any easy modification of the proof of Theorem 2 and therefore details are left to the reader. The key point is that Lemma 6 should be replaced with the following lemma.

Lemma 7. *Consider a run of* κ-BPO *as a series of Bernoulli trials* $Y_1, ..., Y_T$*, where each iteration is a trial and a success is defined to be when* ℓ *is incremented. Then* $Y_1, ..., Y_T$ *are independent, identically distributed Bernoulli trials where the probability of success is* p*.*

□

3 Experimental Evaluation

In this section, the algorithms PO and κ-BPO are evaluated on instances of data summarization with submodular and non-submodular objectives f. In summary, the faster runtime for PO proven in Theorem 1 is demonstrated empirically. Also, the results demonstrate that κ-BPO quickly finds solutions better than the standard greedy algorithm, the stochastic greedy algorithm SG [Mirzasoleiman *et al.*, 2015], and PO.

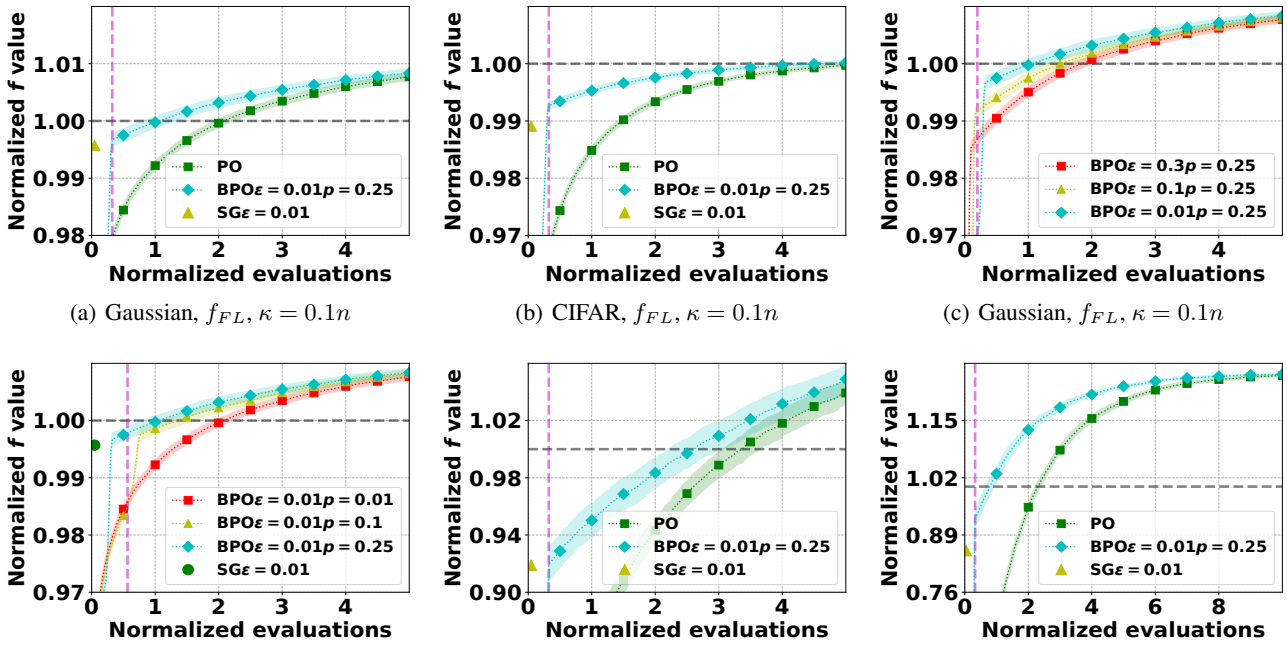
The algorithms evaluated in Section 3 are:

- the standard greedy algorithm [Nemhauser and Wolsey, 1978]
- the stochastic greedy (SG) algorithm of [Mirzasoleiman *et al.*, 2015].
- PO: the variant of the algorithm of Friedrich and Neumann [2014] as detailed in Alg. 1 and analyzed in Section 2.1.
- κ-BPO: the version of BPO that biases towards only one set in $\mathcal{S}$, based on the input κ as discussed in Section 2.3.

For both PO and κ-BPO, the parameter $P = 2\kappa$ is used on all instances.

3.1 Application

In data summarization (DS), we have a set U of data points and we wish to find a subset of U of cardinality κ that best *summarizes* the entire dataset U. $f : 2^U \to \mathbb{R}_{\geq 0}$ takes $X \subseteq U$ to a measure of how effectively X summarizes U. For the ground set U, we use: (i) A set of 10 dimensional vectors drawn from κ gaussian distributions (Gaussian), and (ii) a set of 32×32 color images from the CIFAR-100 dataset



(a) Gaussian, f_{FL}, $\kappa = 0.1n$ (b) CIFAR, f_{FL}, $\kappa = 0.1n$ (c) Gaussian, f_{FL}, $\kappa = 0.1n$

(d) Gaussian, f_{FL}, $\kappa = 0.1n$ (e) Gaussian, f_{DPP}, $\kappa = 0.1n$ (f) CIFAR, f_{DPP}, $\kappa = 0.1n$

Figure 1: In all plots the y-axis is normalized by the standard greedy value on the instance, and the x-axis is normalized by the number kn of evaluations required by the standard greedy algorithm. The dataset, objective, and value of κ are indicated in the caption of each subfigure.

[Krizhevsky *et al.*, 2009] each represented by a 3072 dimensional vector of pixels (CIFAR). For the objective f, we use: (i) The monotonic and submodular objective k-medoid objective [Kaufman and Rousseeuw, 2009] (f_{MED}), and (ii) the monotone weakly submodular objective based on Determinantal Point Process (DPP) [Kulesza *et al.*, 2012] (f_{DPP}). A lower bound on the submodularity ratio has been proven for the latter objective [Bian *et al.*, 2017].

3.2 Results

The experimental results are shown in Figure 1. All results are the mean of 50 repetitions of each algorithm; shaded regions represent one standard deviation from the mean. Objective and runtime are normalized by the objective value of and number of queries made ($n\kappa$) by the standard greedy algorithm. The value of the solution of the standard greedy algorithm is plotted as a dotted gray horizontal line $y = 1$. The time where β in κ-BPO reached κ is plotted as a vertical magenta line.

The best solution value obtained by each algorithm is shown as the rightmost point in each plot. Both κ-BPO and PO were eventually able to find better solutions than the standard greedy algorithm (i.e., normalized value > 1.0), especially on the non-submodular objective (Figures 1(e) and 1(f)). Observe that PO typically exceeds the stochastic greedy objective value within $c\kappa n$, where $c \leq 2$. This behavior corroborates our theoretical analysis that PO achieves a good solution in expectation in $O(\kappa n)$ queries. In addition, κ-BPO exceeds the SG value in cn queries. Finally, for PO, recall that the theoretical anlaysis shows that for any $\kappa < P$, the approximation ratio holds.

Because PO and κ-BPO can be terminated at any time, the running time may be compared by observing where any vertical line intersects the curves for each algorithm. The running time of the standard greedy corresponds to the line $x = 1$ (not plotted). κ-BPO reaches solution values closer to the standard greedy algorithm in significantly faster time than PO, as expected by its design. The effect of varying the parameters ε and p on the behavior of κ-BPO is shown in Figs. 1(c), 1(d), respectively: smaller ε leads to a higher initial increase but the initial increase is slower, while smaller p slows down the rate of the initial increase.

4 Conclusions

In this work, we have re-analyzed the evolutionary algorithm PO, originally analyzed for submodular maximization by Friedrich and Neumann [2014], and showed that it achieves nearly the optimal worst-case ratio in expectation on SM for any $\kappa < P$ in $O(nP)$ queries. In contrast, Friedrich and Neumann [2014] showed that the optimal worst-case ratio is achieved in expected $O(nP^2)$ queries. This improved rate of convergence is supported by an empirical evaluation.

Further, it has been shown that changing the selection process in PO results in improved query complexity to $O(n \log(P))$ to obtain the same approximation results. A variant of this algorithm κ-BPO is shown empirically to have a much faster initial rate of convergence to a good solution than PO, without sacrificing the long-term behavior of the PO algorithm.

References

[Badanidiyuru *et al.*, 2014] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2014.

[Bian *et al.*, 2017] Andrew An Bian, Joachim M Buhmann, Andreas Krause, and Sebastian Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. In *International Conference on Machine Learning (ICML)*, 2017.

[Bian *et al.*, 2020] Chao Bian, Chao Feng, Chao Qian, and Yang Yu. An efficient evolutionary algorithm for subset selection with general cost constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[Crawford, 2019] Victoria G. Crawford. An efficient evolutionary algorithm for minimum cost submodular cover. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[Das and Kempe, 2011] Abhimanyu Das and David Kempe. Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection. *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011.

[Friedrich and Neumann, 2014] Tobias Friedrich and Frank Neumann. Maximizing submodular functions under matroid constraints by evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature (PPSN)*, 2014.

[Friedrich *et al.*, 2010] Tobias Friedrich, Jun He, Nils Hebbinghaus, Frank Neumann, and Carsten Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[Kaufman and Rousseeuw, 2009] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Technical Report, University of Toronto, 2009.

[Kulesza *et al.*, 2012] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[Laumanns *et al.*, 2002] Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl, and Kalyanmoy Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *International Conference on Parallel Problem Solving from Nature (PPSN)*, 2002.

[Mirzasoleiman *et al.*, 2013] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[Mirzasoleiman *et al.*, 2015] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier Than Lazy Greedy. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[Nemhauser and Wolsey, 1978] G L Nemhauser and L A Wolsey. Best Algorithms for Approximating the Maximum of a Submodular Set Function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[Neumann and Wegener, 2007] Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40, 2007.

[Qian *et al.*, 2015a] Chao Qian, Yang Yu, and Zhi-Hua Zhou. On constrained boolean pareto optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[Qian *et al.*, 2015b] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[Qian *et al.*, 2017] Chao Qian, Jing-cheng Shi, Yang Yu, Ke Tang, and Zhi-hua Zhou. Subset Selection under Noise. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[Roostapour *et al.*, 2019] Vahid Roostapour, Aneta Neumann, Frank Neumann, and Tobias Friedrich. Pareto optimization for subset selection with dynamic cost constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[Soma and Yoshida, 2016] Tasuku Soma and Yuichi Yoshida. Maximizing monotone submodular functions over the integer lattice. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2016.