An Efficient Evolutionary Algorithm for Minimum Cost Submodular Cover

Victoria G. Crawford

Department of Computer and Information Science and Engineering, University of Florida, United States
vcrawford01@ufl.edu

Abstract

In this paper, the Minimum Cost Submodular Cover problem is studied, which is to minimize a modular cost function such that the monotone submodular benefit function is above a threshold. For this problem, an evolutionary algorithm EASC is introduced that achieves a constant, bicriteria approximation in expected polynomial time; this is the first polynomial-time evolutionary approximation algorithm for Minimum Cost Submodular Cover. To achieve this running time, ideas motivated by submodularity and monotonicity are incorporated into the evolutionary process, which likely will extend to other submodular optimization problems. In a practical application, EASC is demonstrated to outperform the greedy algorithm and converge faster than competing evolutionary algorithms for this problem.

1 Introduction

A function $f : 2^S \to \mathbb{R}_{\geq 0}$ defined on subsets of a ground set S is monotone submodular if it possesses the following two properties:

  i. For all $A \subseteq B \subseteq S$, $f(A) \leq f(B)$ (monotonicity).

  ii. For all $A \subseteq B \subseteq S$ and $x \notin B$, $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$ (submodularity).

Monotone submodular set functions and optimization problems are found in many applications in machine learning and data mining. In this paper, the NP-hard Minimum Cost Submodular Cover Problem (MCSC) is considered, which is defined as follows.

Problem 1 (Minimum Cost Submodular Cover (MCSC)). *Let S be a ground set of size n. Let $c : 2^S \to \mathbb{R}_{\geq 0}$ be a modular[1] function such that $c(X) = 0$ if and only if $X = \emptyset$, and $f : 2^S \to \mathbb{R}_{\geq 0}$ be monotone submodular. Given a threshold $\tau \leq f(S)$, MCSC is to find $\operatorname{argmin}\{c(X) : X \subseteq S, f(X) \geq \tau\}$. The function c is called the cost, while f is called the benefit.*

Applications of MCSC include data summarization [Mirza-soleiman *et al.*, 2015; 2016], active set selection [Norouzi-Fard *et al.*, 2016], recommendation systems [Guillory and Bilmes, 2011], and viral marketing in social networks [Kuhnle *et al.*, 2017].

The standard greedy algorithm[2] is an effective, efficient approximation algorithm for MCSC [Wolsey, 1982]; however, once the greedy solution has been obtained, it is unclear how it could be improved if more computational resources are available. Therefore, it is of interest to employ methods that can improve the solution quality at the expense of more runtime, while maintaining a worst-case guarantee. For this reason, an evolutionary algorithm has recently been proposed for MCSC [Qian *et al.*, 2015a].

Although random search methods such as evolutionary algorithms (EA) can find better quality solutions in practice, it is difficult to analyze the approximation quality of evolutionary algorithms. The algorithm of Qian *et al.* is able to improve upon the greedy solution in practice, but requires expected exponential time[3] to have a worst-case guarantee similar to that of the greedy algorithm. Furthermore, no evolutionary algorithm exists in prior literature for MCSC that achieves such an approximation ratio in polynomial time.

1.1 Contributions

This paper presents the novel algorithm, EASC (Evolutionary Algorithm for Submodular Cover, Alg. 1), which is the first polynomial-time evolutionary algorithm for MCSC with constant, bicriteria approximation ratio: EASC finds a solution A such that $f(A) \geq (1 - \epsilon)\tau$ and $c(A) \leq (\ln(1/\epsilon) + 1)c(A^*)$, where A^* is an optimum solution and $\epsilon \in (0, 1)$ is an input parameter. The expected time is $\mathcal{O}(n^3((c_{max}/c_{min}) \ln(1/\epsilon))^2)$, where c_{max} and c_{min} are the maximum and minimum cost of a single element $s \in S$, respectively. If c_{max}/c_{min} is bounded by a polynomial in n and ϵ is a constant, then EASC finds a near-feasible solution to MCSC with a constant approximation ratio in expected polynomial time.

In contrast to existing EAs that have been analyzed for submodular optimization problems [Qian *et al.*, 2015b; 2015a;

---

[1]The function c is modular if $c(X) = \sum_{x \in X} c(\{x\})$ for all $X \subseteq S$.

[2]The greedy algorithm is discussed in the Appendix in the full version of the paper.

[3]Time is measured in number of evaluations of f and c, as is commonly done [Badanidiyuru and Vondrák, 2014].

2017], EASC is not a generic EA for multi-objective optimization. Instead, EASC takes advantage of the structure of monotone submodular functions to quickly strengthen its population. A key idea in EASC is that the range $[0, \tau]$ is discretized into *bins* and subsets of $X \subseteq S$ are mapped to bins based on the value of $f(X)$. Solutions within a bin compete with one another using a novel measure of cost-effectiveness. Both the bin structure and notion of cost-effectiveness are designed to take advantage of monotonicity and submodularity. It is likely that these ideas have potential to be applied to monotone submodular optimization problems other than MCSC.

EASC is experimentally evaluated on instances of the Influence Threshold Problem (IT) [Goyal *et al.*, 2013; Kuhnle *et al.*, 2017] on real social network datasets. EASC is compared to both the greedy algorithm as well as the existing EA that has been analyzed for MCSC, POM [Qian *et al.*, 2015a]. Both EASC and POM are able to find better solutions than the greedy algorithm on the problem instances, which demonstrates the value of EAs for MCSC. In addition, EASC is shown to converge faster than POM on some instances.

Organization. Related work is first discussed in Section 1.2. Then, EASC is described in detail in Section 2. Theoretical results on the approximation ratio of EASC are presented in Section 3. Finally, the application and an experimental analysis of EASC is given in Section 4.

Notation. The following notation will be used throughout the paper. For $x \in S$, define $c(x) = c(\{x\})$ and $f(x) = f(\{x\})$. Define $c_{min} = \min_{x \in S} c(x)$, and $c_{max} = \max_{x \in S} c(x)$. Let $f_\tau(X) = \min\{f(X), \tau\}$ for all $X \subseteq S$. The notation for marginal gain is shortened to $\Delta f(X, x) = f(X \cup \{x\}) - f(X)$ for $X \subseteq S$ and $x \in S$. Finally, $\exp(a)$ denotes the exponential function $a \mapsto e^a$.

1.2 Related Work

Evolutionary algorithms (EAs) have previously been analyzed for submodular optimization problems [Friedrich and Neumann, 2015; Qian *et al.*, 2015a; 2015b; 2017; Friedrich *et al.*, 2018]. In general, these EAs work by maintaining a population of non-dominating[4] solutions. Iteratively, a random solution from the population is selected and mutated. If the new solution is not strictly dominated by an existing solution in the population, it is kept[5] in the population and solutions dominated by the new solution are removed from the population. These EAs are quite generic and apply broadly to multi-objective optimization problems. In contrast, EASC is designed specifically for MCSC.

Friedrich and Neumann (2014) and Qian *et al.* (2015b) analyzed similar EAs for the problem of maximizing a monotone, submodular function with respect to a cardinality constraint k. Friedrich and Neumann obtained as good an approximation ratio as the greedy algorithm in expected $\mathcal{O}(n^2(\log(n) + k))$ time and Qian *et al.* in expected $\mathcal{O}(k^2 n)$ time for this problem.

---

[4]A solution X is dominated by a solution Y if $c(Y) \leq c(X)$ and $f(Y) \geq f(X)$. The domination is strict if at least one of the inequalities is strict.

[5]In addition, the EA may require that the new solution meet some requirement such as the cost being beneath a bound.

Algorithm 1 Evolutionary Algorithm for MCSC (EASC)

Input: MCSC instance parameters $f : 2^S \rightarrow \mathbb{R}_{\geq 0}$, $c : 2^S \rightarrow \mathbb{R}_{\geq 0}$, and τ, bin parameters $\delta \in [1 - c_{min}/c(A^*), 1 - c_{min}/c(S)]$ and $\epsilon \in (0, 1)$, and number of iterations $T \in \mathbb{Z}_{\geq 0}$.
$t = 1, \mathcal{B} = \{\emptyset\}$
bin $\leftarrow$ The bin function induced by τ, δ, ϵ
$\prec \leftarrow$ The comparison operator induced by τ, bin
while $t \leq T$ **do**
    X uniformly randomly chosen from $\mathcal{B}$
    $X' = \text{mutate}(X)$
    if $\exists Y \in \mathcal{B}$ such that $\text{bin}(Y) = \text{bin}(X')$ **then**
        if $Y \prec X'$ **then**
            $\mathcal{B} = \mathcal{B} \setminus \{Y\} \cup \{X'\}$
        end if
    else
        $\mathcal{B} = \mathcal{B} \cup \{X'\}$
    end if
    $t = t + 1$
end while

Qian *et al.* (2015a) analyzed an EA for MCSC called POM (Pareto Optimization Method) [Qian *et al.*, 2015a]. Qian *et al.* proved that the population of POM would contain an $H_{c\tau}$[6] $= O(\log(c\tau))$ approximate solution for MCSC in $\mathcal{O}(Nn(\log(n) + \log(c_{max}) + N)))$ expected time, where N is the number of distinct f values in $[0, \tau]$. In order that the approximation guarantees of POM be in expected polynomial time, the number of distinct values of f in the region of $[0, \tau]$ must be bounded by a polynomial. However, this is not a realistic assumption for many applications in machine learning and data mining, where f is real-valued and easily takes on exponentially many values in the region $[0, \tau]$ [Kuhnle *et al.*, 2017; Mirzasoleiman *et al.*, 2015].

The N in the number of expected time comes from the population size of POM. Hence in POM, the population can get quite large, which in turn affects the expected time before the approximation ratio is reached. A similar issue arises when an EA for the dual problem of MCSC is analyzed [Qian *et al.*, 2017]. EASC does not have this problem as its population size is always $\mathcal{O}((c_{max}/c_{min}) \ln(1/\epsilon)n)$. There exist results on approximating a set of non-dominating solutions with a set of smaller size [Laumanns *et al.*, 2002; Horoba and Neumann, 2009]. In fact, Laumanns *et al.* and Horoba and Neumann both describe approaches of binning solutions that serves a similar purpose to the bins in EASC, though the bins in EASC are quite different; among other reasons, solutions that dominate others in the population of EASC are possible. However, it is not clear that the approaches described by Laumanns *et al.* and Horoba and Neumann could be done efficiently in this context nor could result in approximation ratios in expected polynomial time.

---

[6]c is the minimum real number making $cf(X)$ for all $X \subseteq S$ and $c\tau$ integers, and the $c\tau$ harmonic number is $H_{c\tau} = \sum_{j=1}^{c\tau} 1/j$.

2 Evolutionary Algorithm for MCSC (EASC)

In this section, the algorithm EASC (Evolutionary Algorithm for Submodular Cover) is introduced. Pseudocode for EASC can be found in Algorithm 1. EASC is designed for finding good approximate solutions to instances of MCSC efficiently. As will be shown in Section 3, if the input parameter ϵ is constant and c_{max}/c_{min} is bounded by a polynomial, then EASC provides a near-feasible solution to MCSC with a constant approximation ratio in expected polynomial time.

Fundamental to EASC is a mapping from 2^S to $\mathcal{O}((c_{max}/c_{min})\ln(1/\epsilon)n)$ bins; *bin j* is associated with the subinterval

$$\left[\left(1 - \delta^j\right)\tau, \left(1 - \delta^{j+1}\right)\tau\right] \subseteq [0, \tau],$$

and a subset $X \subseteq S$ is mapped into the bin where $\min\{f(X), \tau\}$ falls. The population $\mathcal{B} \subseteq 2^S$, which is a set of subsets of S, contains at most one subset of S per bin. The bins are discussed in more detail in Section 2.1.

The input parameters $\epsilon \in (0, 1)$ and $\delta \in [1 - c_{min}/c(A^*), 1 - c_{min}/c(S)]$, where A^* is an optimal solution to the instance, determine the number of bins and the intervals for each bin. Lower ϵ and δ values result in less bins, and hence a smaller population size. To find a δ in the required range, δ can be set to $1 - c_{min}/B$ where B is an upper bound on $c(A^*)$ such that $B \leq c(S)$. In the experiments in Section 4, B is set to the cost of the greedy solution.

The number of iterations of EASC is determined by the input parameter T. At each iteration, EASC chooses a solution $X \in \mathcal{B}$ to mutate to X', under the mutation process described in Section 2.3. If it is not the case that X' is mapped to a bin with a better solution according to comparison $\prec$, X' is added to $\mathcal{B}$, and the weaker solution is removed, if any. The comparison operator $\prec$ is discussed in Section 2.2.

The structure of the bins and the comparison operator $\prec$ are motivated by monotonicity and submodularity, as described in Sections 2.1 and 2.2.

2.1 The Bin Function

In EASC, $[0, \tau]$ is discretized into $\log_\delta(\epsilon) + 1$ intervals associated with bins. Every $X \subseteq S$ is mapped to the bin where $\min\{f(X), \tau\}$ falls: The function `bin` takes $X \subseteq S$ and returns a bin number in $\{0, ..., \log_\delta(\epsilon)\}$ as follows:

$$\texttt{bin}(X) = \begin{cases} i & \text{if } (1 - \delta^i)\tau \leq f(X) < (1 - \delta^{i+1})\tau \\ & \text{for an } i \in \{0, ..., \log_\delta(\epsilon) - 1\} \\ \log_\delta(\epsilon) & \text{if } f(X) \geq (1 - \epsilon)\tau. \end{cases}$$

The bins are depicted in Figure 1. The population $\mathcal{B}$ in EASC contains at most one solution for each bin, and therefore is bounded in size by $\log_\delta(\epsilon) + 1$. A solution $X \in \mathcal{B}$ that maps to bin $\log_\delta(\epsilon)$ (the final bin) is near-feasible: $f(X) \geq (1 - \epsilon)\tau$. It is the solution mapped to this bin that will give the approximation ratio in expected polynomial time.

Using the fact that $\delta \leq 1 - c_{min}/c(S)$, the total number of bins is bounded as follows.

Proposition 1. *The number of bins is at most*

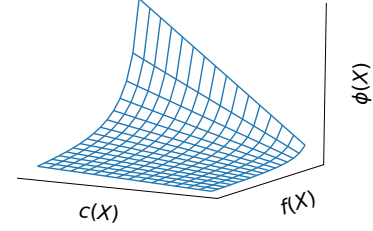$$\frac{c_{max}}{c_{min}} \ln\left(\frac{1}{\epsilon}\right) n + 1.$$

Figure 2: An illustration of the cost-effectiveness ϕ for X where `bin`$(X) \in \{1, ..., \log_\delta(\epsilon) - 1\}$. On both the $c(X)$ and $f(X)$ axis, left is greater. Lower values of $\phi(X)$ means *more* cost-effective.

Proof. It is the case that

$$\log_\delta(\epsilon) = \frac{\ln(\epsilon)}{\ln \delta} = \frac{\ln(1/\epsilon)}{-\ln \delta} \leq \frac{\ln(1/\epsilon)}{1 - \delta} \leq \frac{c(S)}{c_{min}} \ln\left(\frac{1}{\epsilon}\right).$$

Since $c(S) \leq c_{max}n$, the result follows. □

The motivation behind the interval assignment of each bin comes from the greedy algorithm for MCSC. Suppose the sequence of elements $a_1, ..., a_k$ is chosen by the greedy algorithm for the instance of MCSC. Let $A_i = \{a_1, ..., a_i\}$. It is the case[7] that for $i < k$ the marginal gain at each step is lower bounded as follows:

$$f(A_{i+1}) - f(A_i) \geq \frac{c(a_i)}{c(A^*)}(\tau - f(A_i))$$

where A^* is an optimal solution to the instance of MCSC. If $\delta \geq 1 - c_{min}/c(A^*)$ the region of each bin mimics this marginal gain. Intuitively, the bins can be thought of like steps in the greedy algorithm. EASC holds on to the best solution for each step.

2.2 Comparison Operator $\prec$

If two solutions in $\mathcal{B}$ map to the same bin, then the *weaker* solution is removed. Weaker is determined by the comparison operator $\prec$. $\prec$ uses a novel measure of cost-effectiveness, ϕ, in order to compare solutions. Let $X \subseteq S$. If `bin`$(X) = 0$ or `bin`$(X) = \log_\delta(\epsilon)$, then $\phi(X) = c(X)$. Otherwise

$$\phi(X) = c(X)/\ln\left(\frac{\tau}{\tau - f(X)}\right).$$

Notice that lower ϕ means better cost-effectiveness. Then $Y \prec X$ if and only if `bin`$(X) = $`bin`$(Y)$ and $\phi(X) < \phi(Y)$.

Figure 2 illustrates the cost-effectiveness $\phi(X)$ for varying values of $c(X)$ and $f(X)$ for `bin`$(X) \in \{1, ..., \log_\delta(\epsilon) - 1\}$. Lower values of $c(X)$ and higher values of $f(X)$ result in lower $\phi(X)$ (which means *more* cost-effective). But as $f(X)$ decreases, differences in $c(X)$ are amplified in $\phi(X)$. Therefore, c matters more when comparing solutions in lower bins, and in higher bins (excluding the last) f matters more. This encourages solutions in $\mathcal{B}$ to rise up (via mutation) to the final bin, where a $(1 - \epsilon)$-feasible solution is held.

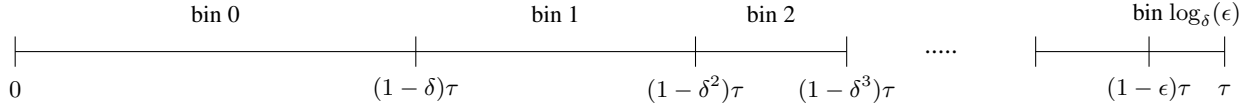[7]See the Appendix in the full version of the paper.

Figure 1: The region $[0, \tau]$ is discretized into $\log_\delta(\epsilon) + 1$ bins in EASC. Solutions $X \subseteq S$ are mapped into the bin corresponding to the region where $\min\{f(X), \tau\}$ falls. $\mathcal{B}$ contains at most 1 subset of S for each bin.

Like the bins discussed in Section 2.1, the motivation for $\prec$ is the greedy algorithm for MCSC. Again, consider the sequence of elements chosen by the greedy algorithm $a_1, ..., a_k$ and let $A_i = \{a_1, ..., a_i\}$. For $i < k$ it is the case[8] that

$$c(A_i)/\ln\left(\frac{\tau}{\tau - f(A_i)}\right) \leq c(A^*).$$

Therefore at every iteration the greedy algorithm has a solution A_i where $\phi(A_i) \leq c(A^*)$. In a sense, a solution $X \subseteq S$ such that $\phi(X) \leq c(A^*)$ has as good of cost-effectiveness as solutions picked by the greedy algorithm. $\prec$ ensures that if $\mathcal{B}$ contains a solution X such that $\phi(X) \leq c(A^*)$, then X cannot be replaced with a solution that is less good in that sense.

2.3 Mutation of Elements in $\mathcal{B}$

Random mutation of solutions in the population occur in the same fashion as existing Pareto optimization algorithms [Qian *et al.*, 2015b]. At each iteration of EASC, an element $X \in \mathcal{B}$ is chosen uniformly randomly to be mutated. X is mutated into X' as follows: Every $x \in X$ is removed from X with independent probability $1/n$. Every $x \notin X$ is added to X with independent probability $1/n$. The number of elements expected to change from X to X' is 1.

X' is added to $\mathcal{B}$ if there does not exist a $Y \in \mathcal{B}$ mapping to the same bin as X' such that $X' \prec Y$. If no such Y exists, then X' is added to $\mathcal{B}$ and any existing solution in the bin of X' is removed.

3 Approximation Results

In this section, the number of iterations before EASC contains a near-feasible solution for MCSC with an approximation ratio of $\ln(1/\epsilon) + 1$ is analyzed, where $\epsilon \in (0, 1)$ is an input parameter of EASC. Each iteration of EASC involves exactly one evaluation each of f and c. If time is measured in evaluations of f and c, as is commonly done [Badanidiyuru and Vondrák, 2014], then the expected time is a constant times the expected number of iterations.

The approximation guarantee in Theorem 1 is a *bicriteria approximation guarantee*, which means that both the feasibility constraint[9] and the minimum cost are approximated. Algorithms with bicriteria approximation guarantees have previously been considered for submodular optimization problems [Iyer and Bilmes, 2013].

If ϵ is assumed to be a constant and c_{max}/c_{min} bounded by a polynomial in n, then Theorem 1 shows that EASC finds a near-feasible solution with a constant approximation ratio in expected polynomial iterations.

Theorem 1. *Suppose that we have an instance of MCSC with optimal solution $A^* \neq \emptyset$, and EASC is run indefinitely with input $\epsilon \in (0, 1)$ and $\delta \in [1 - c_{min}/c(A^*), 1 - c_{min}/c(S)]$. Then $\mathcal{B}$ contains a set A in bin $\log_\delta(\epsilon)$ such that $f(A) \geq (1 - \epsilon)\tau$ and*

$$c(A) \leq \left(\ln\left(\frac{1}{\epsilon}\right) + 1\right)c(A^*),$$

where A^* is an optimum solution, in expected number of iterations at most

$$en\left(\left(\frac{c_{max}}{c_{min}}\right)\ln\left(\frac{1}{\epsilon}\right)n + 1\right)^2.$$

Once a solution that fits the criteria of Theorem 1 appears in bin $\log_\delta(\epsilon)$, it cannot be replaced by one that does not since the comparison operator $\prec$ compares based on only c in the last bin. Notice that Theorem 1 does not contradict the optimality of the $\ln(n)$-approximation ratio for the set cover problem [Feige, 1998], since the guarantee is bicriteria. The same bicriteria approximation guarantee in Theorem 1 holds for the greedy algorithm[10].

The proof of Theorem 1 tracks *cost-effective solutions* in $\mathcal{B}$ over the duration of EASC. A set X is cost-effective if it satisfies one of the following (mutually exclusive) conditions:

i. $\mathtt{bin}(X) < \log_\delta(\epsilon)$ and $\phi(X) \leq c(A^*)$.

ii. $\mathtt{bin}(X) = \log_\delta(\epsilon)$ and $c(X) \leq (\ln(1/\epsilon) + 1) c(A^*)$.

Once a cost-effective set is in the final bin, a solution that meets the criteria of Theorem 1 is in the population. By design, EASC never replaces a cost-effective solution in its population with one that is not cost-effective. In addition, because of the requirement that $\delta \geq 1 - c_{min}/c(A^*)$, the bins are structured tightly enough so that there is a significant probability that cost-effective solutions mutate into cost-effective solutions in strictly greater bins. Together, these points enable EASC to contain a cost-effective solution in its final bin in polynomial expected iterations.

The following lemmas will be used to prove Theorem 1. The lemmas are not novel to this work, but have previously been used to analyze the approximation guarantee of the greedy algorithm for MCSC [Goyal *et al.*, 2013]. Proofs of the lemmas are included in the Appendix in the full version of the paper.

Lemma 1. *Suppose that we have an instance of MCSC with optimal solution $A^* \neq \emptyset$. Let $X \subseteq S$ and $x^* = argmax_{x \in S} \Delta f_\tau(X, x)/c(x)$. Then*

$$\tau - f_\tau(X \cup \{x^*\}) \leq \left(1 - \frac{c(x^*)}{c(A^*)}\right)(\tau - f_\tau(X)).$$

---

[8]See the Appendix in the full version of the paper.

[9]The set X is feasible iff $f(X) \geq \tau$.

[10]This result was originally proven for an influence application by Goyal *et al.* (2013) but holds for general MCSC. See the Appendix in the full version of the paper for the proof.

Lemma 2. *Suppose that we have an instance of MCSC with optimal solution $A^* \neq \emptyset$. Let $X \subseteq S$ such that $f(X) < \tau$ and $x^* = argmax_{x \in S} \Delta f_\tau(X, x)/c(x)$. Then $c(x^*) \leq c(A^*)$.*

Proof of Theorem 1. Recall that notation is defined in Section 1. For brevity, let $r = \log_\delta(\epsilon)$ be the final bin.

There always exists at least one cost-effective solution in $\mathcal{B}$: The empty set is cost-effective since $\phi(\emptyset) = c(\emptyset) = 0 < c(A^*)$, and the empty set is never removed from $\mathcal{B}$ because there does not exist $Y \subseteq S$ such that $c(Y) < 0 = c(\emptyset)$. Note that this means if a solution is removed from $\mathcal{B}$, it can be assumed that it did not correspond to bin 0.

Define an infinite sequence ℓ_t, $t \in \{1, 2, ...\}$, where ℓ_t is the max value in $\{0, ..., r\}$ such that there exists a cost-effective solution $X \in \mathcal{B}$ where $\text{bin}(X) = \ell_t$ at the beginning of iteration t of EASC.

Part One. First, it is shown that the sequence ℓ_t is non-decreasing. Let X be the cost-effective set corresponding to ℓ_t at the beginning of iteration t. If X is not removed from $\mathcal{B}$ during the tth iteration, then clearly $\ell_t \leq \ell_{t+1}$.

Suppose X is removed from $\mathcal{B}$ during the tth iteration. Then X was replaced with X' such that $X \prec X'$ and $\text{bin}(X) = \text{bin}(X')$. Let $b = \text{bin}(X) = \text{bin}(X')$. As explained above, $b \neq 0$. Suppose $b = r$. Then $X \prec X'$ implies that $c(X') < c(X) \leq (\ln(1/\epsilon) + 1)\, c(A^*)$. Therefore X' is cost-effective. If $b \in \{1, ..., r - 1\}$. Then $X \prec X'$ implies that $\phi(X') < \phi(X) \leq c(A^*)$ and hence X' is also cost-effective. In both of these cases, $\ell_t = \ell_{t+1}$.

Part Two. Second, it is shown that if $\text{bin}(X) < r$ and

$$x^* = \text{argmax}_{x \in S} \frac{\Delta f_\tau(X, x)}{c(x)},$$

then $\text{bin}(X \cup \{x^*\}) > \text{bin}(X)$.

Let $a = \text{bin}(X)$. Lemma 1 and that $\delta \geq 1 - c_{min}/c(A^*)$ implies

$$f_\tau(X \cup \{x^*\}) \geq (1 - \delta)\tau + \delta f_\tau(X).$$

By definition of the bins $f_\tau(X) \geq (1 - \delta^a)\tau$, and therefore

$$f_\tau(X \cup \{x^*\}) \geq (1 - \delta^{a+1})\tau.$$

Since $a < r$, it is the case that $\text{bin}(X \cup \{x^*\}) \geq a + 1$.

Part Three. Third, it is shown that if X is cost-effective, $\text{bin}(X) < r$, and x^* defined as in Part Two, then $X \cup \{x^*\}$ is cost-effective. Let $a = \text{bin}(X)$ and $b = \text{bin}(X \cup \{x^*\})$. To show the cost-effectiveness of $X \cup \{x^*\}$, four cases are analyzed based on the values of a and b.

Case (i): $b < r$ and $a = 0$. In this case, $X = \emptyset$ as explained at the beginning of the proof, and $X \cup \{x^*\} = \{x^*\}$. Lemma 1 states that

$$\tau - f_\tau(x^*) \leq \left(1 - \frac{c(x^*)}{c(A^*)}\right)\tau \leq \exp\left(-\frac{c(x^*)}{c(A^*)}\right)\tau$$

which can be re-arranged to see that $\phi(x^*) \leq c(A^*)$ and hence $X \cup \{x^*\}$ is cost-effective.

Case (ii): $b < r$ and $a > 0$. Since $b, a < r$ it is the case that $f_\tau(X \cup \{x^*\}) = f(X \cup \{x^*\})$ and $f_\tau(X) = f(X)$. Lemma 1 gives that

$$\tau - f(X \cup \{x^*\}) \leq \exp\left(-\frac{c(x^*)}{c(A^*)}\right)(\tau - f(X)).$$

Using the upper bound on $\tau - f(X)$ given by re-arranging $\phi(X) \leq c(A^*)$ implies that

$$\tau - f(X \cup \{x^*\}) \leq \exp\left(-\frac{c(X \cup \{x^*\})}{c(A^*)}\right)\tau$$

which may be re-arranged to see that $X \cup \{x^*\}$ is cost-effective.

Case (iii): $b = r$ and $a > 0$. X being cost-effective and $a < r$ imply that $c(X \cup \{x^*\}) =$

$$c(X) + c(x^*) \leq \ln\left(\frac{\tau}{\tau - f(X)}\right)c(A^*) + c(x^*).$$

By Lemma 2, $c(x^*) \leq c(A^*)$. Therefore $X \cup \{x^*\}$ is cost-effective.

Case (iv): $b = r$ and $a = 0$. Then $X = \emptyset$ as explained at the beginning of the proof. Then $c(X \cup \{x^*\}) = c(x^*) \leq c(A^*)$ by Lemma 2, and therefore $X \cup \{x^*\}$ is cost-effective.

Part Four. It is now shown that if at iteration t, the cost-effective set X associated with $\ell_t < r$ is mutated into $X \cup \{x^*\}$, then $\ell_t < \ell_{t+1}$.

Suppose X is mutated into $X \cup \{x^*\}$ on iteration t. $X \cup \{x^*\}$ is cost-effective by Part Three. Let $b = \text{bin}(X \cup \{x^*\})$. By Part Two, $b > \ell_t$. If there does not exist $Y \in \mathcal{B}$ at the beginning of iteration t such that $\text{bin}(Y) = b$, then $X \cup \{x^*\}$ is added to $\mathcal{B}$. Then $\ell_{t+1} = b$.

Suppose there does exist a $Y \in \mathcal{B}$ at the beginning of iteration t such that $\text{bin}(Y) = b$. Y is not cost-effective by definition of ℓ_t. Then if $b = r$,

$$c(X \cup \{x^*\}) \leq (\ln(1/\epsilon) + 1)c(A^*) < c(Y)$$

and so $Y \prec X \cup \{x^*\}$. Then Y is replaced with $X \cup \{x^*\}$ in $\mathcal{B}$. If $b < r$, and recalling that $b \neq 0$ as explained at the beginning of the proof, then

$$\phi(X \cup \{x^*\}) \leq c(A^*) < \phi(Y).$$

Therefore again $Y \prec X \cup \{x^*\}$, so Y is replaced with $X \cup \{x^*\}$ in $\mathcal{B}$. In both cases, $\ell_{t+1} = b$.

Part Five. Finally, the expected number of iterations until $\ell_t = r$ is analyzed. Once $\ell_t = r$, the solution in $\mathcal{B}$ mapping to bin r satisfies the conditions of the theorem statement.

Suppose it is the beginning of iteration t of EASC such that $\ell_t < r$. Then with probability at least

$$\frac{1}{|\mathcal{B}|}\frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en(r+1)}$$

the set X corresponding to ℓ_t will be chosen and mutated into $X \cup \{x^*\}$. By Part Four, if this occurs $\ell_t < \ell_{t+1}$. By Part One, the sequence is non-decreasing. This means that the expected number of steps for ℓ_t to reach r is at most $enr(r + 1)$. The bound on r given by Proposition 1 of Section 2.1 gives the theorem statement. □

4 Experimental Analysis

In this section, EASC is experimentally evaluated on instances of the Influence Threshold Problem (IT) [Goyal *et al.*, 2013], a special case of MCSC. EASC is compared to the greedy
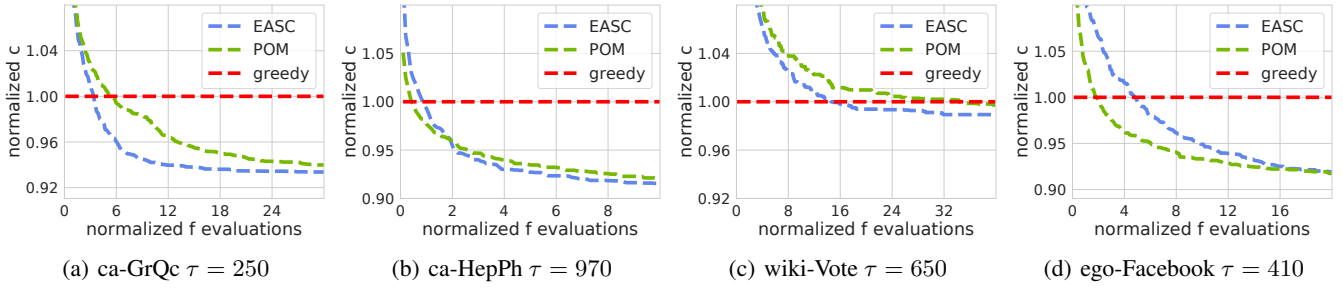
(a) ca-GrQc $\tau = 250$ (b) ca-HepPh $\tau = 970$ (c) wiki-Vote $\tau = 650$ (d) ego-Facebook $\tau = 410$

Figure 3: Over the duration of EASC and POM, the minimum cost of any solution in the population with f value above $(1-\epsilon)\tau$ is plotted. Both cost and number of f evaluations are normalized by that of the greedy algorithm.

algorithm and POM [Qian *et al.*, 2015a]. In all experiments, EASC and POM find solutions of lower cost than the greedy algorithm, and in most cases EASC converges faster than POM to a low cost. Code to run the experiments is publicly available at https://gitlab.com/vcrawford/easc.git.

4.1 Application and Setup

The experiments are run on instances of the Influence Threshold Problem, defined as follows. Let $G = (V, E)$ be a social network where vertices V represents users, and directed edges E represent social connections. Activation of users in the social network starts from an initial seed set and then propagates across "live edges" according to the independent cascade model [Kempe *et al.*, 2003], in which every edge $(u, v) \in E$ has an independent probability p_{uv} of being live.

For every user $v \in V$, there is a cost c_v of seeding that user. The cost of seeding a set X is $c(X) = \sum_{x \in X} c_x$. The function $f(X)$ is the expected number of users that will become active if X is seeded; f is monotone submodular [Kempe *et al.*, 2003]. Then the IT problem is defined as follows: given an activation threshold τ, find argmin$\{c(X) : X \subseteq S, f(X) \geq \tau\}$.

The experiments are run on four real social networks from SNAP [Leskovec and Krevl, 2015]: ca-GrQc ($n = 5242$), ca-HepPh ($n = 12008$), wiki-Vote ($n = 7115$), and ego-Facebook ($n = 4039$). The independent cascade model is used to model activation from a seed set for the above four social networks with constant edge probabilities $p = 0.07$, $p = 0.02$, $p = 0.04$, and $p = 0.013$, respectively.

Computing the expected activation $f(X)$ under the independent cascade model is #P-hard [Chen *et al.*, 2010]. Instead of evaluating f directly, the reverse influence sampling approach [Borgs *et al.*, 2014] with 100,000 samples is used in order to approximate f. The same set of samples is used for all algorithms on a data set.

The cost function $c(X) = \sum_{x \in X} c_x$, where every node v in the social network is assigned a cost c_v that is its outgoing degree perturbed by random multiplicative noise[11] In particular, node v with outdoing degree d has cost $1 + (1 + |\xi|)d$ where $\xi \sim \mathcal{N}(\mu = 0, \sigma = 0.5)$.[12]

---

[11]Under this model of cost, social network users with more outgoing edges are generally more expensive to seed, but individual preferences factor into the price of seeding via random noise.

[12]A normal distribution with mean 0 and standard deviation 0.5.

The instance of MCSC for each social network is run with a different threshold τ. The greedy algorithm is run with input τ and $\epsilon = 0.05$. EASC is run with τ, $\epsilon = 0.05$, and $\delta = 1 - c_{min}/B$ where B is the cost of the output of the greedy algorithm when run with τ and $\epsilon = 0$. POM is run with threshold $(1 - \epsilon)\tau$ for fair comparison with EASC, although POM is not a bicriteria algorithm. EASC and POM are run 3 times on each instance, and the results are averaged.

4.2 Results

The experimental results are plotted in Figure 3. At small intervals over the duration of EASC and POM, the minimum cost of any solution in the population with f value above $(1 - \epsilon)\tau$ is plotted. The cost and the number of f evaluations are normalized by that of the greedy algorithm. That is, if the greedy algorithm returned a set G, then the costs plotted are normalized by $c(G)$ and the number of function evaluations are normalized by $n|G|$.

In all experiments, EASC and POM were able to find a better solution than the greedy algorithm. In ca-HepPh (Figure 3(b)), both EASC and POM find a better solution than the greedy algorithm *in less f evaluations*. These results demonstrate an ability to improve on the solution quality of the greedy algorithm.

In ca-GrQc (Figure 3(a)) and wiki-Vote (Figure 3(c)), EASC finds a better solution more quickly than POM throughout the entire experiment. In contrast, in ca-HepPh (Figure 3(b)) and ego-Facebook (Figure 3(d)) POM finds a better solution more quickly in the earlier stages of the experiment, but EASC catches up and either outperforms POM for the remainder of the iterations (caHepPh) or converges to about the same performance. This behavior may be explained by the fact that at first POM has a smaller population since every solution competes with every other solution, unlike EASC where competition is restricted to within bins. But as the population of POM gets larger (at least 3 times larger than EASC at the end of these experiments), the improvement of POM is slowed.

Acknowledgements

References

[Badanidiyuru and Vondrák, 2014] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1497–1514. SIAM, 2014.

[Borgs *et al.*, 2014] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 946–957. SIAM, 2014.

[Chen *et al.*, 2010] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

[Feige, 1998] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[Friedrich and Neumann, 2014] Tobias Friedrich and Frank Neumann. Maximizing submodular functions under matroid constraints by multi-objective evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 922–931. Springer, 2014.

[Friedrich and Neumann, 2015] Tobias Friedrich and Frank Neumann. Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evolutionary computation*, 23(4):543–558, 2015.

[Friedrich *et al.*, 2018] Tobias Friedrich, Andreas Göbel, Francesco Quinzan, and Markus Wagner. Heavy-tailed mutation operators in single-objective combinatorial optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 134–145. Springer, 2018.

[Goyal *et al.*, 2013] Amit Goyal, Francesco Bonchi, Laks VS Lakshmanan, and Suresh Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social network analysis and mining*, 3(2):179–192, 2013.

[Guillory and Bilmes, 2011] Andrew Guillory and Jeff A Bilmes. Simultaneous learning and covering with adversarial noise. In *ICML*, volume 11, pages 369–376, 2011.

[Horoba and Neumann, 2009] Christian Horoba and Frank Neumann. Additive approximations of pareto-optimal sets by evolutionary multi-objective algorithms. In *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 79–86. ACM, 2009.

[Iyer and Bilmes, 2013] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013.

[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.

[Kuhnle *et al.*, 2017] Alan Kuhnle, Tianyi Pan, Md Abdul Alim, and My T Thai. Scalable bicriteria algorithms for the threshold activation problem in online social networks. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

[Laumanns *et al.*, 2002] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[Leskovec and Krevl, 2015] Jure Leskovec and Andrej Krevl. {SNAP Datasets}:{Stanford} large network dataset collection. 2015.

[Mirzasoleiman *et al.*, 2015] Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In *Advances in Neural Information Processing Systems*, pages 2881–2889, 2015.

[Mirzasoleiman *et al.*, 2016] Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In *Advances in Neural Information Processing Systems*, pages 3594–3602, 2016.

[Norouzi-Fard *et al.*, 2016] Ashkan Norouzi-Fard, Abbas Bazzi, Ilija Bogunovic, Marwa El Halabi, Ya-Ping Hsieh, and Volkan Cevher. An efficient streaming algorithm for the submodular cover problem. In *Advances in Neural Information Processing Systems*, pages 4493–4501, 2016.

[Qian *et al.*, 2015a] Chao Qian, Yang Yu, and Zhi-Hua Zhou. On constrained boolean pareto optimization. In *IJCAI*, pages 389–395, 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*, pages 1774–1782, 2015.

[Qian *et al.*, 2017] Chao Qian, Jing-Cheng Shi, Yang Yu, and Ke Tang. On subset selection with general cost constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-2017)*, pages 2613–2619, 2017.

[Wolsey, 1982] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.